



Pontifícia Universidade Católica
do Rio de Janeiro

Gisele Cristina da Cunha Holtz

**Representação semi-implícita de modelos
discretos de reservatórios naturais de petróleo**

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Engenharia Civil, do Departamento de Engenharia Civil e Ambiental da PUC-Rio.

Orientador : Prof. Luiz Fernando Martha
Coorientador: Prof. Waldemar Celes

Rio de Janeiro
Junho de 2025



Pontifícia Universidade Católica
do Rio de Janeiro

Gisele Cristina da Cunha Holtz

**Representação semi-implícita de modelos
discretos de reservatórios naturais de petróleo**

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Engenharia Civil da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Luiz Fernando Martha

Orientador

Departamento de Engenharia Civil e Ambiental – PUC-Rio

Prof. Waldemar Celes

Coorientador

Departamento de Informática – PUC-Rio

Prof. Joaquim Bento Cavalcante Neto

UFC

Prof. Anselmo Montenegro

UFF

Prof. Elisa Sotelino

Departamento de Engenharia Civil e Ambiental – PUC-Rio

Frederico Abraham

Instituto Tecgraf – PUC-Rio

Rio de Janeiro, 25 de Junho de 2025

Todos os direitos reservados. A reprodução, total ou parcial do trabalho, é proibida sem a autorização da universidade, do autor e do orientador.

Gisele Cristina da Cunha Holtz

Graduou-se em Engenharia Civil pelo Centro Universitário de Volta Redonda (UniFOA) em 2002. Concluiu o mestrado em Engenharia Civil pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) em 2005. Atua como pesquisadora no Instituto Tecgraf/PUC-Rio.

Ficha Catalográfica

Holtz, Gisele Cristina da Cunha

Representação semi-implícita de modelos discretos de reservatórios naturais de petróleo / Gisele Cristina da Cunha Holtz; orientador: Luiz Fernando Martha; coorientador: Waldemar Celes. – 2025.

67 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil e Ambiental, 2025.

Inclui bibliografia

1. Modelo discreto. 2. Representação implícita. 3. Visualização de reservatório de petróleo. I. Martha, Luiz Fernando. II. Celes, Waldemar. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Civil e Ambiental. IV. Título.

Agradecimentos

A Deus, pela certeza de Seu amor incondicional.

Aos meus filhos, Guilherme e Matheus, que são o meu grande amor e a razão do meu viver. Mesmo diante da minha ausência em tantos momentos, sempre me incentivaram com carinho e compreensão.

Ao meu marido Júlio, pelo companherismo e paciência inestimáveis.

Aos meus pais, Osmar e Fátima, que são meu porto seguro. À Laci, pelos cuidados e carinho de uma verdadeira mãe. Aos meus irmãos, Patrícia e Gustavo, pela amizade e apoio incondicional. Aos meus sobrinhos, Bernardo, Nina e Pedro, que preenchem minha vida com alegria. Aos meus sogros, Vera e Júlio, meus cunhados, Marcella, Natália e Rafael, e à minha madrinha, Ismaete, pela torcida constante ao longo dessa caminhada.

Ao professor Luiz Fernando Martha, orientador deste trabalho, expressei minha profunda gratidão pela confiança e pela oportunidade de retomar este doutorado. Mais do que orientador acadêmico, teve um papel decisivo na minha trajetória profissional e sempre foi uma fonte de inspiração.

Ao professor Waldemar Celes, coorientador deste trabalho, por ter aceitado o desafio de participar dessa caminhada com tanta dedicação, paciência e generosidade. Suas orientações foram extremamente valiosas nesse processo, sempre marcadas pela clareza e pelo pragmatismo. Seu incentivo constante foi decisivo para a conclusão deste trabalho.

Aos meus amigos, pelos momentos de descontração que trouxeram leveza aos meus dias, pelo acolhimento nos momentos difíceis e pela presença constante.

Ao Tecgraf, pelo apoio tecnológico essencial ao desenvolvimento desta tese e por oferecer um ambiente que valoriza a pesquisa, marcado pelo espírito colaborativo e pela troca constante de conhecimento. Aos amigos que fiz nesse espaço, com quem aprendi imensamente, cresci como profissional e encontrei apoio nos momentos mais desafiadores.

Ao Departamento de Engenharia Civil e Ambiental e à PUC-Rio, pelo suporte acadêmico e pela contribuição essencial para meu crescimento intelectual e profissional ao longo desses anos.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Holtz, Gisele Cristina da Cunha; Martha, Luiz Fernando; Celes, Waldemar. **Representação semi-implícita de modelos discretos de reservatórios naturais de petróleo**. Rio de Janeiro, 2025. 67p. Tese de Doutorado – Departamento de Engenharia Civil e Ambiental, Pontifícia Universidade Católica do Rio de Janeiro.

Simulação numérica de reservatórios naturais de petróleo é amplamente empregada pela indústria de óleo e gás para previsão e planejamento da produção. A simulação de reservatórios baseia-se em uma malha topologicamente estruturada de células hexaédricas, que modela características geológicas complexas. A visualização desses modelos discretos em larga escala é essencial para uma melhor inspeção e análise dos resultados das simulações, o que representa um desafio devido ao tamanho e à complexidade de sua geometria. Enquanto a visualização convencional do modelo é feita a partir do desenho das faces externas, técnicas mais avançadas, como a visualização volumétrica, demandam a representação do modelo completo na GPU e um eficiente algoritmo de localização de pontos. Trabalhos anteriores propuseram representações do modelo baseadas em GPU, mas aplicações em tempo real ainda são limitadas por questões de desempenho. Este trabalho apresenta uma representação semi-implícita compacta de modelos discretos de reservatórios, que combina um campo de coordenadas topológicas com um mapa de profundidade, permitindo a implementação de algoritmos de visualização eficientes e precisos. Experimentos computacionais demonstram as vantagens da representação proposta em termos de uso de memória e desempenho, mantendo a qualidade visual.

Palavras-chave

Modelo discreto; Representação implícita; Visualização de reservatório de petróleo.

Abstract

Holtz, Gisele Cristina da Cunha; Martha, Luiz Fernando (Advisor); Celes, Waldemar (Co-Advisor). **Semi-implicit representation of discrete black oil reservoir models**. Rio de Janeiro, 2025. 67p. Tese de Doutorado – Departamento de Engenharia Civil e Ambiental, Pontifícia Universidade Católica do Rio de Janeiro.

Numerical black oil reservoir simulation is widely employed in the oil and gas industry for planning and predicting field explorations. Reservoir simulation relies on a topologically structured grid of hexahedral cells, modeling complex geological features. Visualization of these large-scale simulated discrete models is essential for inspection and analysis but presents challenges due to their size and geometric complexity. While conventional methods focus on rendering external faces, advanced techniques, like volume rendering, require full model representation on the GPU and an efficient point-location algorithm. Previous works proposed GPU-based model representations, but real-time applications remain limited by performance. Simplified alternatives improve speed but at the cost of accuracy. This work introduces a compact semi-implicit representation of discrete reservoir models that combines a topological coordinate field and a depth map to enable the implementation of fast and accurate visualization algorithms. Computational experiments demonstrate the advantage of the proposed representation regarding memory usage and performance while preserving visual quality.

Keywords

Discrete model; Implicit representation; Black oil reservoir visualization.

Sumário

1	Introdução	13
2	Trabalhos relacionados	17
2.1	Modelagem geométrica	20
2.1.1	Representação explícita	20
2.1.2	Representação implícita	21
3	Modelos Discretos de Reservatório	22
4	Representação Semi-Implícita de Reservatório	26
4.1	Campo de coordenadas topológicas (CCT)	26
4.1.1	Busca geométrica	30
4.1.2	Método de Newton-Raphson	32
4.1.3	Coordenadas topológicas ótimas	34
4.2	Mapa de profundidade	36
4.3	Atributos do modelo	37
4.4	Memória requerida	38
4.5	Localização de pontos na <i>GPU</i>	39
4.6	Tratamento das descontinuidades	40
5	Resultados	50
5.1	Modelos	50
5.2	Memória requerida	52
5.3	Desempenho e acurácia na localização de pontos	54
5.4	Visualização volumétrica	57
5.4.1	Análise da performance	58
5.4.2	Análise da qualidade	59
5.5	Linhas de corrente	60
6	Conclusão	64
	Referências bibliográficas	65

Lista de figuras

Figura 1.1	Visualização convencional de modelos discretos de reservatório, representados por uma grade topologicamente estruturada, com as complexidades da geometria destacadas na seção transversal: descontinuidades, células inativas e células degeneradas.	14
Figura 1.2	Técnicas de visualização baseadas em shaders.	15
Figura 3.1	Identificação das células de um modelo discreto de reservatório.	22
Figura 3.2	Características da representação de um modelo discreto de reservatório.	23
Figura 3.3	Seção 2D de um modelo discreto de reservatório, com as complexidades da geometria destacadas na imagem.	24
Figura 3.4	Representação de células de um modelo discreto de reservatório.	24
Figura 4.1	Representação semi-implícita para modelos contínuos.	27
Figura 4.2	<i>CCT</i> representado pela grade regular envolvendo a malha irregular de hexaedros.	29
Figura 4.3	Cálculo das coordenadas topológicas dos vértices da grade ilustrada em 2D: (a) Malha irregular do reservatório e grade do <i>CCT</i> ; (b) Voxels na área de influência de uma célula; (c) Cálculo da coordenada topológica para os vértices da área de influência. (d) Coordenadas topológicas atribuídas a todos os vértices da grade.	30
Figura 4.4	Geração de amostras pelas células ativas para o processo de otimização.	35
Figura 4.5	Estrutura de dados do mapa de profundidade na <i>GPU</i> sem tratamento de descontinuidades.	37
Figura 4.6	Estrutura de dados do Mapa de Ativas e do Mapa de Propriedades Escalares na <i>GPU</i> .	37
Figura 4.7	Propriedade suavizada do pixel calculada pela média ponderada da propriedade das células ativas mais próximas a ele.	38
Figura 4.8	Gráfico da variação do espaço de memória da parcela implícita em relação ao espaço de memória total da representação semi-implícita de acordo com o valor de α .	39
Figura 4.9	Fluxograma do algoritmo de localização de pontos.	39
Figura 4.10	Ilustração do processo de busca geométrica pela camada que contém o ponto.	40
Figura 4.11	Descontinuidade na geometria do modelo, dentro de um <i>voxel</i> , causada por uma falha geológica.	41
Figura 4.12	Representação do modelo por múltiplos blocos.	42
Figura 4.13	Mapa 2D de falhas no plano $i - j$, ilustrando o prolongamento das falhas.	42

Figura 4.14 Ilustração da aplicação do algoritmo de inundação para a identificação das colunas dos blocos.	43
Figura 4.15 Mapa de profundidade com descontinuidade.	43
Figura 4.16 Fluxograma do algoritmo de obtenção de z nas células vizinhas à falha através do MP com descontinuidade.	44
Figura 4.17 Fluxograma do algoritmo de localização de pontos.	45
Figura 4.18 Mapa de colunas de uma região altamente fraturada do modelo, resultando em um grande número de blocos, sendo cada um representado por uma cor diferente.	46
Figura 4.19 Estrutura de aceleração: (a) Modelo subdividido em blocos, identificados por diferentes cores, e grade regular 2D da estrutura de aceleração; (b) Cálculo da interseção da grade do CCT de cada bloco com as células da estrutura de aceleração; (c) Células da estrutura de aceleração contendo a lista de blocos com o qual há interseção.	46
Figura 4.20 Estrutura de dados da estrutura de aceleração na GPU.	47
Figura 4.21 Alocação dos blocos na textura única do atlas com o CCT de todos os blocos do modelo: (a) Inicialização da textura com as maiores dimensões dos blocos; (b) e (e) Armazenamento do bloco no espaço disponível; (c) e (d) Expansão da textura para alocação dos blocos;	48
Figura 4.22 Estrutura de dados da representação semi-implícita considerando descontinuidades e dos atributos do modelo.	49
Figura 5.1 Detalhes dos modelos reais de reservatório de petróleo usados nos experimentos.	51
Figura 5.2 Vista em mapa da geometria irregular das células nas direções i e j : (a) e (b) Modelo M1; (c) e (d) Modelo M2.	52
Figura 5.3 Erros de aproximação nas coordenadas topológicas (i, j) recuperadas para os quatro primeiro modelos de reservatório. Os gráficos mostram a distribuição dos erros superiores a 0,05 (mais de 99,5% das amostras apresentam erros abaixo desse limite). Nenhum erro acima de 0,05 foi observado para os três últimos modelos.	55
Figura 5.4 Distribuição atualizada dos erros maiores que 0,05 nas coordenadas topológicas (i, j) recuperadas após modificação da resolução do campo de coordenada topológica (CCT) para os modelos M4 e M5.	56
Figura 5.5 Algoritmo de traçado de raios, com amostras geradas ao longo do raio.	57
Figura 5.6 Comparação de performance dos modelos entre a representação proposta e os métodos PLTS e LIBC.	59

- Figura 5.7 Percentual de pixels (relativo à imagem inteira) em cada intervalo de diferença de cor de acordo com a Tabela 5.6, durante a rotação do modelo. A qualidade da imagem obtida com a solução proposta é comparada com a imagem do método *LIBC*, adotando como referência a imagem gerada através do método *PLTS*. Linhas contínuas são associadas à solução proposta e linhas tracejadas ao método *LIBC*. Observe que os gráficos usam a escala logarítmica no eixo vertical para facilitar a visualização. 61
- Figura 5.9 Linhas de corrente obtidas por meio do algoritmo de localização de pontos proposto. 62
- Figura 5.8 Imagens representativas do algoritmo de visualização volumétrica na renderização da saturação de óleo do modelo M1. As imagens das diferenças ilustram o ganho de qualidade da nossa abordagem em comparação com a estratégia *LIBC*. As cores indicam a diferença em cada pixel, conforme a Tabela 5.6. 63

Lista de tabelas

Tabela 5.1	Características dos modelos reais usados nos experimentos: Identificação, resolução da malha de hexaedros, número de células ativas e número de blocos requeridos pela representação semi-implícita.	51
Tabela 5.2	Tempo (minutos) detalhado e total na construção do CCT. Foram medidos os tempos nas seguintes etapas: algoritmo de <i>Bin Packing</i> para organização dos blocos na textura, busca geométrica para cálculo das coordenadas topológicas exatas e aplicação do método dos mínimos quadrados para a determinação das coordenadas topológicas ótimas.	53
Tabela 5.3	Comparação do uso de memória, em MB, entre a estrutura de dados de incidência convencional, o método ResGPU [12] e a representação semi-implícita. Os valores entre parênteses indicam o uso relativo de memória da representação semi-implícita em comparação com os outros métodos.	53
Tabela 5.4	Comparação de performance na localização de pontos nos modelos de reservatório com a estrutura ResGPU e com a representação semi-implícita. O valor representa o tempo médio necessário para a localização de 10^8 pontos na <i>GPU</i> .	54
Tabela 5.5	Porcentagem de pontos que foram classificados incorretamente como externos.	56
Tabela 5.6	Tabela de classificação dos intervalos do Delta E de acordo com a percepção da diferença de cores, mostrando as cores adotadas para visualizá-las.	60

Lista de Abreviaturas

CCT – Campo de Coordenada Topológica

GPU – *Graphics Processing Unit*

LIBC – *Linear Interpolation with Barycentric Coordinates*

MP – Mapa de Profundidade

PLTS – *Point Location with Topological Search*

VA – Vetor de Ativas

1

Introdução

Simulação numérica de reservatórios naturais de petróleo é amplamente empregada pela indústria de óleo e gás para previsão do comportamento e planejamento das estratégias de produção de campos de petróleo. O padrão da indústria é a aplicação do método das diferenças finitas em uma malha topologicamente estruturada composta por células hexaédricas. A geometria do modelo discreto do reservatório é bastante complexa, pois as células são dispostas de forma a respeitar a geologia do domínio, com seus horizontes e falhas. Com o aumento do poder computacional e a necessidade de representar campos de larga escala, essas simulações geram uma enorme quantidade de dados que precisam ser interpretados e analisados de forma criteriosa, permitindo identificar padrões de comportamento do reservatório, avaliar a eficiência das estratégias de produção e prever o desempenho futuro, embasando, assim, as tomadas de decisão.

A visualização convencional de um modelo discreto de reservatório é feita a partir da renderização das faces externas das células da fronteira da malha (Figura 1.1), utilizando algoritmos convencionais de desenho de polígonos, que baseiam-se no desenho de primitivas gráficas, como triângulos e quadriláteros, cuja geometria é determinada pela posição dos vértices. Esse tipo de visualização pode ser facilmente implementada com o uso de uma estrutura de dados convencional que seja capaz de identificar faces não compartilhadas entre células, ou seja, faces externas. Essas faces são transferidas para a *GPU* e renderizadas. Porém, técnicas mais elaboradas de visualização são essenciais para revelar a distribuição das grandezas físicas no interior do volume. Essas técnicas empregam renderização baseada em programação em placa gráfica (*GPU programming*) e requerem a localização de pontos em relação à malha, isto é, dada a posição $[x, y, z]$ de um ponto, deseja-se identificar a célula $[I, J, K]$ do modelo que o contém e a coordenada paramétrica $[s, t, r]$ do ponto nessa célula. Esse algoritmo de localização de pontos implementado numa estrutura explícita convencional de incidência de células é extremamente ineficiente, exigindo o emprego de estruturas de aceleração. Tanto a representação desses modelos na GPU de forma compacta, quanto o uso de algoritmos eficientes de localização de pontos, são desafios encontrados para alcançar uma visualização

interativa na aplicação dessas técnicas.

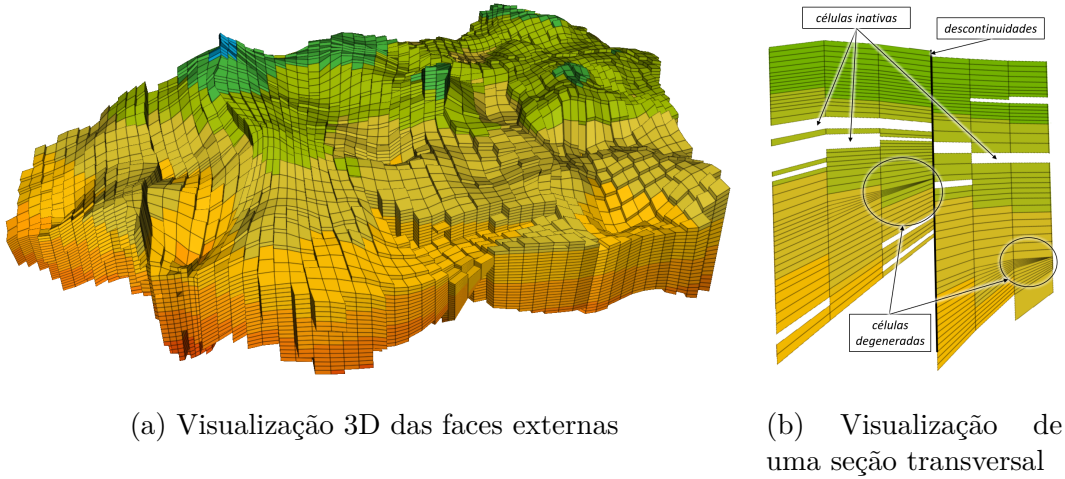


Figura 1.1: Visualização convencional de modelos discretos de reservatório, representados por uma grade topologicamente estruturada, com as complexidades da geometria destacadas na seção transversal: descontinuidades, células inativas e células degeneradas.

Franceschin et al.[12] exploram a estrutura topológica dos modelos de reservatório para propor uma representação compacta própria para GPU, denominada *ResGPU*. Os autores apresentam um algoritmo de localização de pontos que pode ser utilizado na implementação de diferentes técnicas de visualização. Caso o algoritmo de localização de pontos seja suficientemente eficiente, o modelo de reservatório discreto pode ser interpretado como uma textura procedural mapeada em superfícies arbitrárias (Figuras 1.2a, 1.2b e 1.2c). Netto e Celes [23] exploram essa representação para propor uma técnica de visualização de corte com preservação do contexto, revelando a distribuição de grandezas físicas nas proximidades dos poços em modelos de reservatório (Figura 1.2d). No entanto, em um estudo experimental sobre visualização volumétrica de modelos de reservatório, conduzido por Campagnolo e Celes [7], o desempenho apresentado pela estrutura *ResGPU* foi insuficiente para a renderização do modelo em tempo real (Figura 1.2e). Os autores, então, propuseram a simplificação da geometria do modelo, aproximando cada face de célula por um par de triângulos, o que introduz imprecisões na imagem final.

Visando a solução dessas questões, este trabalho propõe uma representação inovadora de modelos discretos de reservatório. Para viabilizar um eficiente algoritmo de localização de pontos, propõe-se uma representação semi-implícita, que estabelece um campo de coordenadas topológicas capaz de aproximar a geometria das células no plano horizontal, combinado com um mapa que armazena explicitamente as coordenadas de profundidade das camadas

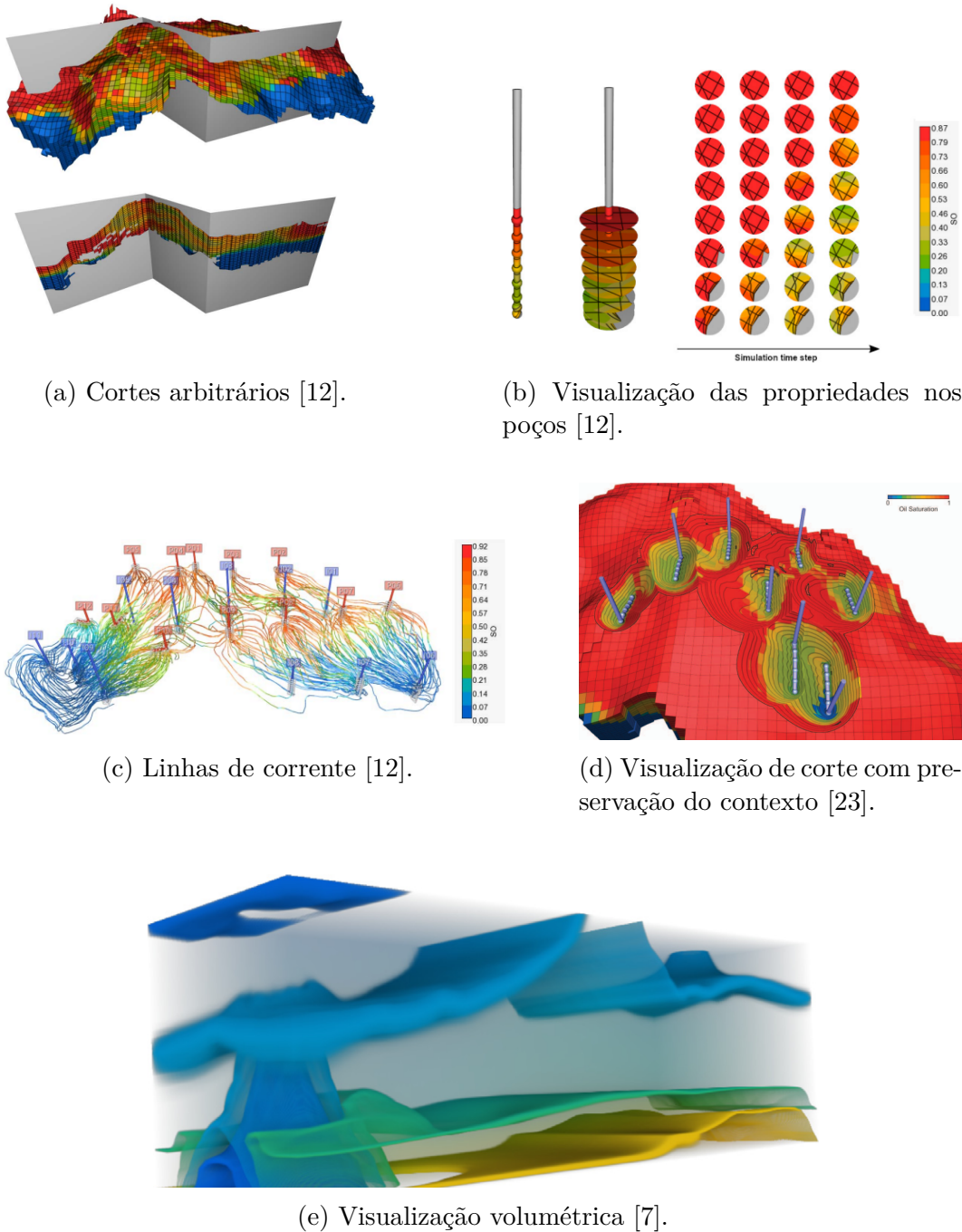


Figura 1.2: Técnicas de visualização baseadas em shaders.

geológicas. Trata-se de uma estrutura de dados mais compacta que as abordagens existentes, adequada para armazenamento em GPU, que possibilita uma significativa economia de memória e facilita a implementação de um algoritmo de localização de pontos altamente eficiente. Embora o uso de estruturas implícitas limite a reconstrução exata das geometrias do reservatório, experimentos computacionais de visualização volumétrica demonstram que a aproximação alcançada proporciona qualidade de imagem superior e melhor desempenho em comparação ao método que representa cada face da célula utilizando um

par de triângulos, uma técnica já considerada de boa precisão.

O restante deste trabalho está estruturado da seguinte forma: No Capítulo 2 é feita uma recapitulação dos trabalhos relacionados à representação de modelos volumétricos e às técnicas de localização de pontos, com destaque para os trabalhos dedicados à representação de modelos de reservatório, além de revisar os conceitos relacionados à representação implícita. Em seguida, o Capítulo 3 descreve um modelo de reservatório e o Capítulo 4 traz a proposta de representação semi-implícita de modelos de reservatório. A memória requerida e os experimentos de performance e acurácia realizados são apresentados no Capítulo 5. Por fim, o Capítulo 6 traz as conclusões da pesquisa realizada e as propostas para trabalhos futuros.

2

Trabalhos relacionados

Na modelagem de reservatórios convencionais, a estrutura de incidências denominada *Corner Point Grid* [24][25] é amplamente empregada para representar malhas de hexaedros, pois possui capacidade de adaptação a geometrias complexas, como falhas geológicas e horizontes. Topologicamente, os *corner points* correspondem aos oito vértices de cada célula da malha e são indexados por coordenadas discretas $[I, J, K]$. Cada *corner point* é definido por uma única posição geométrica $[x, y, z]$. Além da estrutura de incidências, é comum armazenar também a informação de adjacência entre células, que é representada pelos índices das células vizinhas associados a cada uma das faces. A presença de um índice de célula inválido indica que a face correspondente é uma face externa do modelo.

No pós-processamento dos resultados da simulação, a aplicação de técnicas avançadas de visualização depende tanto da viabilidade de representar o modelo na GPU, limitada pela disponibilidade de memória, principalmente quando trata-se de modelos massivos, quanto do uso de algoritmos eficientes de localização de pontos, que permitam a renderização do modelo em tempo real.

Há diversos estudos voltados para a representação de modelos volumétricos com o foco na aplicação de algoritmos de visualização. Muitos desses estudos são voltados para modelos representados por grades regulares. Beyer et al. [3] revisam trabalhos com foco na renderização de grades regulares que adotam técnicas de renderização local. Sarton et al. [26] estendem o problema a malhas não estruturadas, as quais podem ser compostas, por exemplo, por tetraedros, pirâmides e cunhas, onde são revisados trabalhos que utilizam representações hierárquicas e adaptativas e que focam na representação de modelos massivos, que não podem ser inteiramente armazenados na GPU, além de apresentar técnicas para percorrer e amostrar as malhas não estruturadas.

Pesquisas mais recentes abordam a representação de modelos voltados para a área de geociências, que demandam soluções especializadas devido a características específicas, como camadas geológicas e falhas, que devem ser adequadamente respeitadas. Graciano et al. [16] propuseram uma nova estrutura de dados, denominada QuadStack, para representar modelos de camadas

utilizando um conjunto de campos de altura comprimidos em *quadtrees*. As propriedades e a geometria do modelo são armazenadas de forma desacopladas, de forma a permitir que seja explorada a frequente redundância das propriedades de uma camada. Alguns trabalhos propõem representações específicas para modelos discretos de reservatório com malhas de hexaedros topologicamente estruturadas. As pesquisas que utilizam métodos convencionais de visualização tem por objetivo desenvolver técnicas que permitam a escalabilidade dessas soluções para modelos massivos, seja em CPU ou GPU. Uma abordagem comum é o uso de técnicas de multirresolução, onde o grande desafio é a preservação de detalhes finos e descontinuidades. O HexaShrink [24] é uma representação que combina de forma coerente diferentes esquemas de multirresolução (baseado em *wavelets*) em diferentes dimensões, que resulta em uma estrutura que preserva descontinuidades (falhas) entre escalas. Abraham e Celes [1] propõem uma técnica que utiliza um algoritmo de simplificação que explora as características de modelos de camada através da aplicação de um conjunto de operadores de colapso de colunas de células, com o objetivo de manter com precisão a geometria das superfícies da borda do modelo ao mesmo tempo que mantém a malha de hexaedros com boa qualidade, usando técnicas de multirresolução. As propriedades originais do modelo, o *wireframe* e as normais das superfícies são mapeadas nas malhas simplificadas através de mapeamento de textura. Ceballos et al. [8] apresentam uma proposta de simplificação em que oito células do nível mais refinado passam a ser representadas por uma única célula, cuja geometria é dada pelo fecho convexo dos oito vértices das quinas dessas células. Nas regiões de falha, uma heurística é proposta para limitar as possíveis simplificações, de forma a preservar as descontinuidades. A metodologia estabelece como os valores mínimo e máximo das propriedades das células de resolução mais fina são armazenados nas células de resolução mais grosseira e como esses valores são usados para decidir quais nós na *octree* devem ser renderizados.

Diversas técnicas de visualização de modelos de reservatório baseadas em programas que executam na placa gráfica requerem uma representação que, além de ser própria para armazenamento na GPU e compacta em termos de memória, ofereça suporte a um eficiente algoritmo de localização de pontos no domínio do modelo. Franceschin et al. [12] propõem uma estrutura própria para modelos de reservatório, denominada ResGPU, que além de explorar as características de malhas de hexaedros estruturadas, explora a esparsidade no domínio ijk , devido às células inativas. A incidência das células que não são vizinhas à falha é obtida implicitamente, o que proporciona uma grande economia de memória, visto que, na prática, mais de 90% das células são desse

tipo. Uma técnica de hashing espacial perfeito é utilizada para armazenar apenas as células ativas e os seus vértices incidentes, além de tratar as descontinuidades permitindo mais de uma instância dos vértices. O trabalho também propõe um método preciso de localização de pontos que adota uma grade regular como técnica de aceleração. Cada *voxel* da grade regular possui uma lista das células do modelo interceptadas por ele. Para a técnica de visualização volumétrica, que utiliza um algoritmo de traçado de raios e requer a localização de uma sequência de amostras ao longo do raio, uma busca topológica é realizada ao invés de reiniciar a localização de pontos a cada amostra, explorando a coerência espacial entre amostras subsequentes. Devido ao alto grau de distorção da geometria e à existência de células hexaédricas mal-formadas, para maior precisão no algoritmo de localização de pontos, faz uso do método iterativo Newton-Raphson para o cálculo das coordenadas paramétricas dos pontos, limitando sua aplicação a técnicas de visualização em tempo real que demandam amostragem de muitos pontos. Essa solução foi empregada com sucesso na renderização de superfícies de corte [12] e na técnica de visualização de corte com preservação do contexto [23]. Contudo, na visualização volumétrica, o algoritmo de traçado de raios, que demanda a amostragem sequencial de muitos pontos ao longo do raio, requer um algoritmo de localização de pontos ainda mais eficiente. Campagnolo e Celes [7] propuseram duas outras técnicas visando melhorar a performance do algoritmo de visualização volumétrica de modelos de reservatório. A primeira delas, avalia o uso de uma grade regular para reamostragem das propriedades escalares de grades irregulares. Essa técnica possui maior simplicidade de implementação e possibilita a utilização de qualquer algoritmo para visualização de volumes regulares. Como em qualquer malha estruturada com geometria regular, a localização de pontos é direta. Sua desvantagem é pouca precisão e necessidade de reconstrução ao modificar o campo escalar. A segunda técnica propõe que cada face do hexaedro seja representada por dois triângulos na amostragem dos pontos, possibilitando a interpolação linear com coordenadas baricêntricas ao invés de ser necessário um método iterativo para o cálculo das coordenadas topológicas de um ponto. Essa abordagem se mostrou mais eficiente que a proposta por Franceschin et al. [12], porém ainda abaixo dos requisitos necessários para uma inspeção interativa de alguns modelos, além de introduzir imprecisão devido à simplificação da geometria das células.

Neste trabalho, o uso de representação implícita para modelos discretos de reservatório é investigado. Campos de distância [19] são usualmente utilizados na representação implícita de superfícies. Wang et al. [28] propõem uma abordagem que utiliza múltiplos campos de distância com sinal, organizados

em uma árvore binária, para representar o contorno de modelos volumétricos, permitindo combinar objetos de diferentes escalas e decompor os sólidos em regiões. Recentemente, representações neurais implícitas tem sido exploradas na representação de formas 3D [11][10][13][9]. No entanto, todas essas representações implícitas descrevem apenas o contorno das estruturas, sem uma descrição contínua das células discretas que decompõem o interior de regiões volumétricas.

A seção a seguir apresenta os conceitos básicos relacionados aos tipos de representações comuns em modelagem geométrica, destacando o porquê da representação implícita ser particularmente adequada para a localização de pontos.

2.1

Modelagem geométrica

A modelagem geométrica consiste na representação matemática da forma e características geométricas de um objeto. De forma geral, existem dois tipos principais de representação: explícita e implícita. A escolha do tipo de representação a ser adotada deve ser feita com base na análise das características do tipo de processamento geométrico que será aplicado a ela, pois cada uma delas possui vantagens e desvantagens.

A seguir serão apresentados os tipos de representação utilizando exemplos de curvas bidimensionais para simplificar a compreensão dos conceitos. Serão discutidas as características particulares de cada um delas, suas vantagens e desvantagens [14] [4].

2.1.1

Representação explícita

Na representação explícita, uma variável é expressa explicitamente em função de outras variáveis. Na geometria analítica, ao utilizar coordenadas Cartesianas, as equações são representadas na forma $y = f(x)$. Entre os inúmeros exemplos possíveis de representação explícita no plano pode-se citar a equação de uma reta, $y = 2x + 1$, ou de uma parábola, $y = x^2$.

Essa representação permite gerar facilmente pontos na curva. Para cada x , basta avaliar a função $f(x)$ para obter o ponto. Sendo assim, é uma questão muito simples desenhar uma curva a partir de sua representação explícita. Porém, a verificação de pontos sobre a curva exige que seja encontrado o ponto da curva mais próximo ao ponto (x, y) consultado, o que significa que deve-se encontrar o mínimo da função distância entre um ponto da curva e o ponto, exigindo um certo esforço computacional. Não é possível representar

curvas fechadas ou com auto-interseção com uma única equação explícita. Por exemplo, a representação explícita de um círculo unitário centrado na origem em coordenadas Cartesianas é dada por duas equações:

$$y = \pm\sqrt{1-x^2} \quad (2-1)$$

Uma alternativa para representar curvas consiste no uso de coordenadas paramétricas, no qual as equações são escritas explicitamente em função de novos parâmetros, $x = x(t)$ e $y = y(t)$, reduzindo a dimensão do problema. Esse tipo de abordagem permite descrever superfícies fechadas ou com auto-interseção por meio de uma única equação, conferindo maior flexibilidade à modelagem geométrica. No exemplo do círculo unitário centrado na origem, a representação paramétrica é dada pelo intervalo de uma função paramétrica $f : [0, 2\pi] \rightarrow \mathbb{R}^2$, onde:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos t \\ \sin t \end{bmatrix} \quad (2-2)$$

2.1.2

Representação implícita

A representação implícita de uma curva não resolve explicitamente suas variáveis. Em 2D, utilizando coordenadas Cartesianas, são representadas por equações do tipo $f(x, y) = 0$. O círculo unitário centrado na origem é representado pela seguinte equação:

$$x^2 + y^2 - 1 = 0 \quad (2-3)$$

Nessa representação é difícil amostrar pontos sobre a curva, pois é necessário descobrir quais coordenadas (x, y) satisfazem a função implícita, o que torna difícil a tarefa de desenhá-la. Por outro lado, essa representação descreve todo o espaço ao redor de um objeto de forma que, ao avaliar a função implícita, seja possível determinar a posição de um ponto qualquer em relação ao objeto. O escalar resultante da avaliação da função implícita igual a zero indica que o ponto está sobre a curva. Para curvas fechadas, o sinal da função distingue pontos internos e externos, o que é uma característica importante na modelagem de sólidos. Pontos internos possuem sinal negativo e pontos externos possuem sinal positivo.

3

Modelos Discretos de Reservatório

A simulação numérica de reservatórios é tradicionalmente realizada por meio da aplicação do método das diferenças finitas no domínio do reservatório, representado por uma malha de células hexaédricas. Essas células são organizadas em uma grade topológica, respeitando as características geológicas do reservatório. Cada célula é identificada por uma tupla de coordenadas topológicas (I, J, K) , representando os índices da célula na grade (Figura 3.1). A direção K é aproximadamente alinhada à direção z no espaço Euclidiano (coordenada de profundidade). As faces do topo e da base das células hexaédricas são tipicamente não-planas para se conformar às geometrias complexas das superfícies geológicas, enquanto as faces laterais são usualmente planas. Células com um mesmo valor de K formam uma camada, e células com as mesmas coordenadas topológicas (I, J) compõem uma coluna (Figura 3.2).

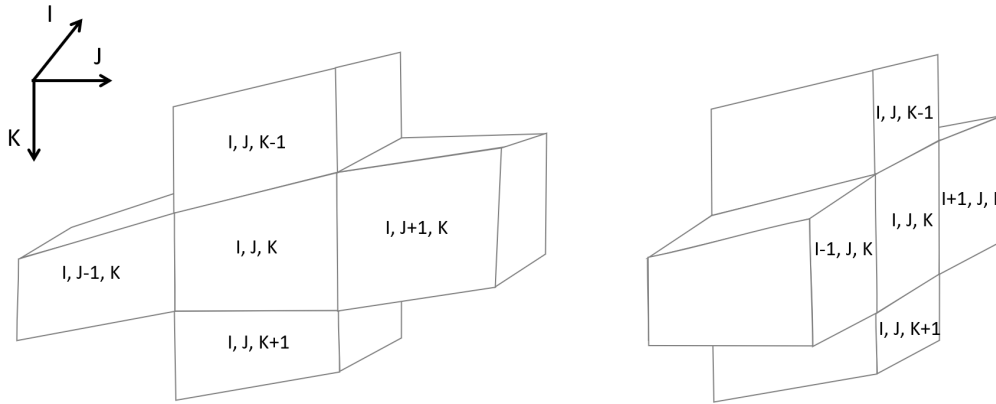
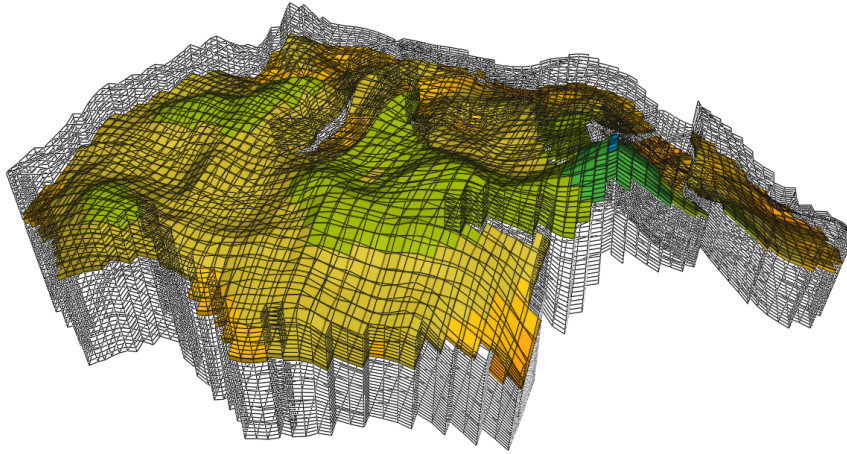
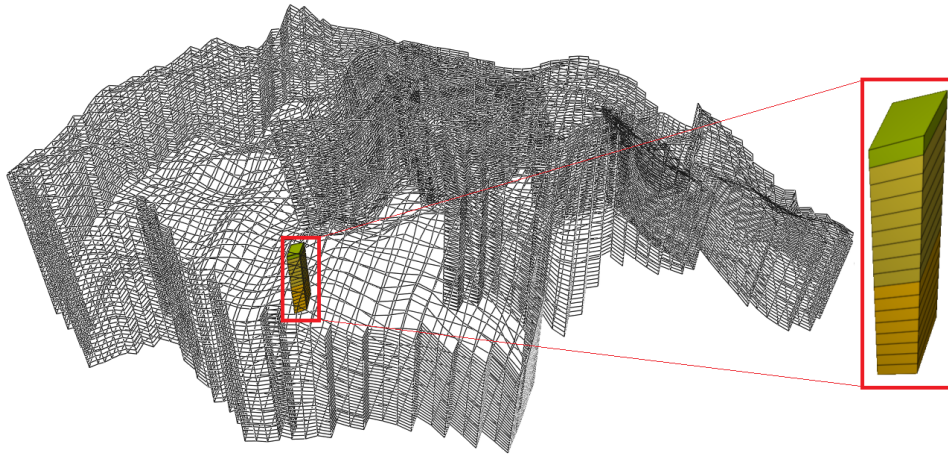


Figura 3.1: Identificação das células de um modelo discreto de reservatório.

A Figura 3.3 mostra detalhes de um modelo discreto de reservatório visto em uma seção 2D, destacando a modelagem de diversas estruturas geológicas características, já ilustradas na Figura 1.1, e que serão descritas a seguir. Tipicamente, as dimensões das células não variam significativamente nas direções I e J ; entretanto, a necessidade de modelar as camadas geológicas impõe grandes variações nas espessuras (alturas) das células ao longo da direção K . As camadas geológicas podem afinar e desaparecer, criando células de *pinch-out*, que são células que possuem arestas colapsadas, podendo apresentar espessura nula, de modo que as camadas superior e inferior entrem em contato direto.



(a) Camada composta por células com uma mesma coordenada topológica K .



(b) Coluna composta por células com as mesmas coordenadas topológicas (I, J) .

Figura 3.2: Características da representação de um modelo discreto de reservatório.

Essa configuração resulta em células vizinhas com grande variação de espessura. Movimentos relativos entre blocos de rocha resultam em falhas geológicas que introduzem descontinuidades nas direções I e J . Células não participantes da simulação de fluxo, como aquelas com porosidade zero, são definidas como inativas, proporcionando flexibilidade na modelagem de domínios de reservatórios irregulares.

Coordenadas topológicas (I, J, K) também são atribuídas aos vértices das células, de forma que a diagonal de uma célula hexaédrica é definida por seus vértices (I, J, K) e $(I + 1, J + 1, K + 1)$, conforme ilustra a Figura 3.4a. No entanto, nas regiões onde há descontinuidades nas direções I e/ou J devido à existência de falhas geológicas, ocorre deslocamentos em profundidade nessas direções entre células vizinhas (Figura 3.4b). Como resultado, em regiões com falhas, vértices que compartilham as mesmas coordenadas topológicas podem ter até quatro coordenadas Euclidianas distintas (x, y, z) .

O objetivo deste trabalho é prover suporte para um eficiente algoritmo de

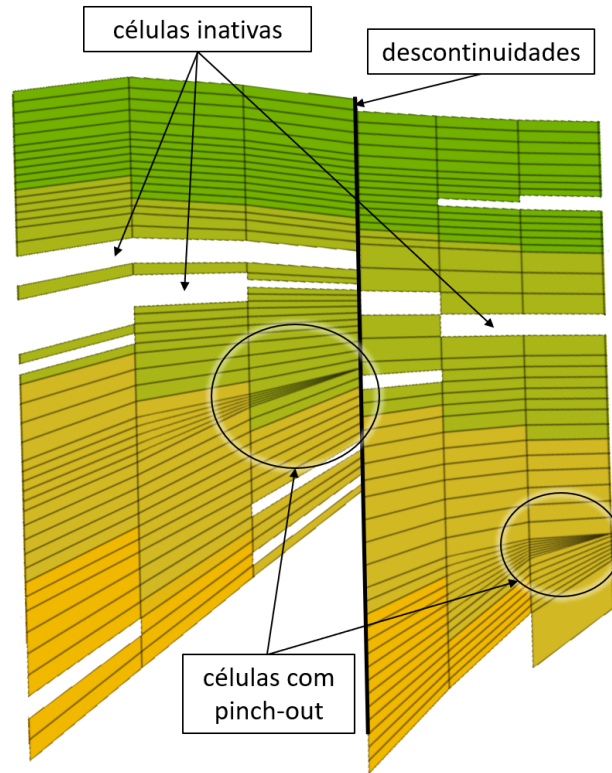
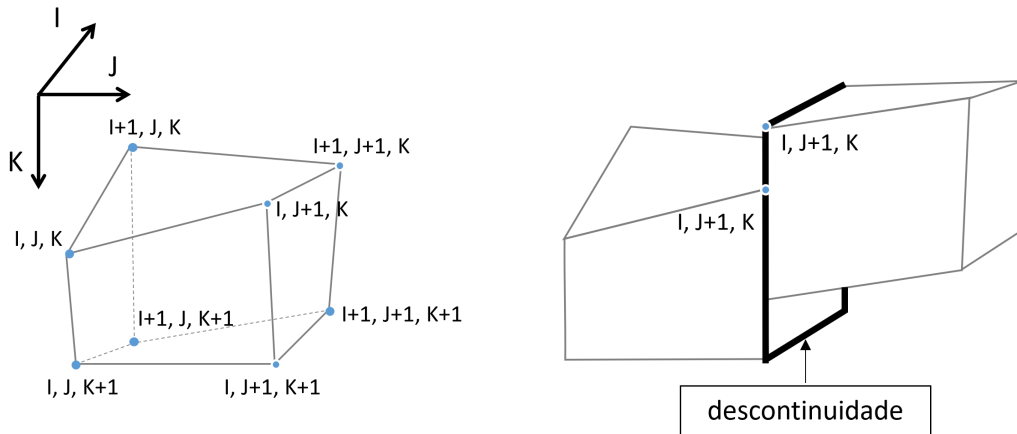


Figura 3.3: Seção 2D de um modelo discreto de reservatório, com as complexidades da geometria destacadas na imagem.



(a) Índices dos vértices de uma célula. (b) Vértices compartilham as mesmas coordenadas topológicas na descontinuidade, mas possuem coordenadas Euclidianas distintas (x, y, z) .

Figura 3.4: Representação de células de um modelo discreto de reservatório.

localização: dado um ponto com coordenada Euclidiana (x, y, z) , determinar a célula (I, J, K) que o contém e a coordenada paramétrica (s, t, r) dentro dessa célula. A existência de células com faces não-planas requer o uso de uma solução numérica para a implementação desse algoritmo. Usualmente, dado

um ponto (x, y, z) , utiliza-se o método de Newton-Raphson para encontrar as coordenadas paramétricas correspondentes (s, t, r) em uma determinada célula, o que é computacionalmente custoso.

Considerando as coordenadas paramétricas dentro de uma célula, $s, t, r \in [0, 1]$, podemos definir um sistema contínuo de coordenadas topológicas globais (i, j, k) que identifica um ponto no reservatório:

$$\begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} I \\ J \\ K \end{bmatrix} + \begin{bmatrix} s \\ t \\ r \end{bmatrix} \quad (3-1)$$

As coordenadas topológicas globais i, j e k são valores em ponto flutuante que variam de 0 até o número de células em cada direção, n_i , n_j e n_k , respectivamente.

Embora existam abordagens numéricas alternativas para simulação de reservatórios, como o uso de malhas não estruturadas com células prismáticas ou tetraédricas, este trabalho foca exclusivamente em grades topologicamente estruturadas compostas por células hexaédricas, que continuam sendo o padrão predominante na indústria.

4

Representação Semi-Implícita de Reservatório

O propósito deste trabalho é oferecer uma representação para modelos discretos de reservatório que possibilite a implementação eficiente de técnicas avançadas de visualização baseadas na *GPU*. As técnicas abordadas nesse trabalho são baseadas em localização de pontos, o que justifica a busca por uma representação implícita. Dado um ponto no espaço euclidiano (x, y, z) , a representação deve permitir o cálculo eficiente da coordenada topológica global (i, j, k) no domínio do reservatório. Apenas as células ativas importam. Células inativas não fazem parte do modelo e são ignoradas.

Devido às características dos modelos de reservatório, optou-se por uma representação semi-implícita, que será inicialmente introduzida sem considerar a existência de falhas no modelo, isto é, desconsiderando as descontinuidades nas direções I e J . Nesse caso, a estrutura de dados, resumida na Figura 4.1, combina uma representação implícita, denominada Campo de Coordenadas Topológicas (CCT), que armazena um campo contínuo global de coordenadas topológicas (i, j) , e uma parcela explícita, denominada Mapa de Profundidade (MP), que armazena explicitamente a coordenada z (profundidade) de cada coluna de vértices. Além dessas estruturas geométricas, a representação completa do modelo inclui um Vetor de Ativas, que diferencia células ativas e inativas, e Vetores de Propriedades Escalares, que armazenam informações relacionadas às propriedades escalares visualizadas. Após a apresentação da representação para modelos contínuos, a solução é expandida para tratar a existência de falhas geológicas através da divisão do modelo em blocos contínuos.

A representação semi-implícita é construída na CPU a partir da representação explícita do modelo de reservatório, usualmente composta por uma estrutura de incidência (*Corner Point Grid*) combinada com a informação de adjacência entre células.

4.1

Campo de coordenadas topológicas (CCT)

O Campo de Coordenadas Topológicas *CCT* armazena coordenadas topológicas globais (i, j) e foi inspirado em campos de distância regulares. O fato de as células da malha de hexaedros do modelo de reservatório

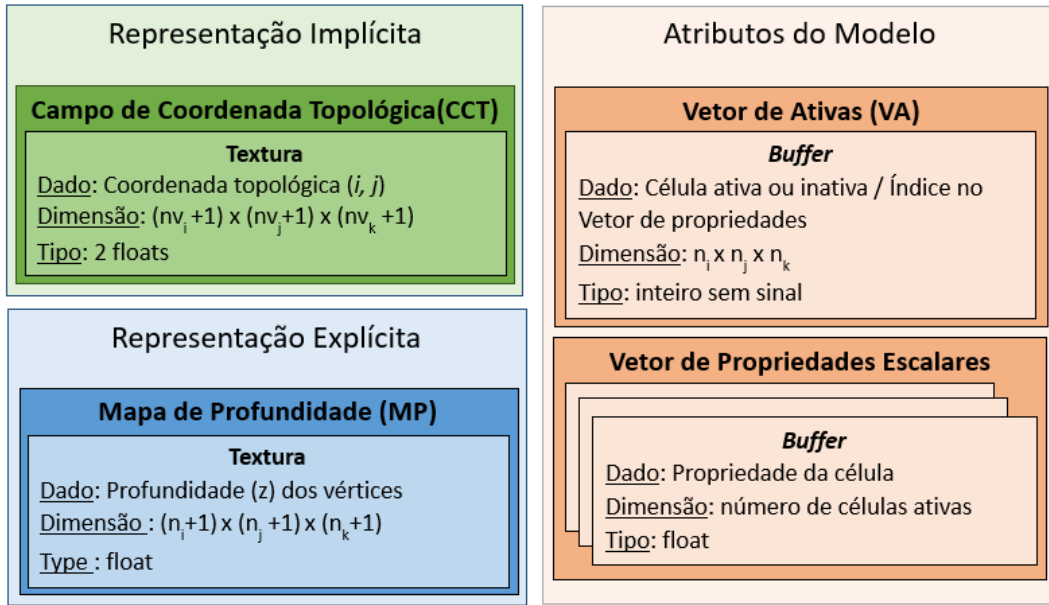


Figura 4.1: Representação semi-implícita para modelos contínuos.

apresentarem geometria mais regular nas direções i e j permite que localmente seja assumida uma aproximação linear das coordenadas topológicas nessas direções. O CCT é composto por uma grade regular tridimensional com $nv_i \times nv_j \times nv_k$ *voxels* (células da grade), cujos vértices armazenam suas respectivas coordenadas topológicas i e j . Trata-se de uma representação implícita, própria para modelos contínuos, que, através da interpolação trilinear por *voxel*, permite recuperar a coordenada topológica global (i, j) a partir da localização de um ponto (x, y, z) no voxel.

Neste texto, a malha irregular de hexaedros do modelo de simulação de reservatório será chamada simplesmente de *malha* e seus elementos hexaédricos de *células*. A grade regular que representa o CCT será referenciada como *grade* e suas células como *voxels*.

O domínio do CCT é definido por uma caixa envolvente orientada no plano $x - y$. Na direção z , entretanto, a caixa permanece alinhada ao sistema global de coordenadas. A Figura 4.2 ilustra a estrutura do CCT cobrindo o domínio de um reservatório. A caixa envolvente do modelo é calculada de acordo com o método de Análise de Componentes Principais, que identifica as direções principais de variação do conjunto de pontos localizados nos vértices das células ativas na borda do modelo. Isso é realizado por meio da decomposição da matriz de covariância, permitindo definir uma caixa envolvente orientada que melhor se ajusta à geometria do modelo [17]. A grade é orientada de acordo com essas direções principais nas coordenadas x e y , onde os eixos locais não são necessariamente alinhados ao sistema global de coordenadas. Essa abordagem reduz a redundância de espaço e melhora a

precisão da representação da grade.

A resolução da caixa está diretamente relacionada à precisão com que as coordenadas (i, j) são reconstruídas e à memória requerida. O desafio consiste em determinar uma resolução que proporcione economia de memória, sem comprometer a precisão dos resultados, mantendo o erro dentro de um limite aceitável. Foi estabelecida uma métrica que determina a resolução da grade ($nv_i \times nv_j \times nv_k$) em função da resolução da malha ($n_i \times n_j \times n_k$). Como a caixa envolvente não está necessariamente alinhada com as direções i e j do modelo, a resolução da grade nessas direções é calculada a partir do tamanho médio ($\bar{l}$) das arestas horizontais de todas as células ativas do modelo, bem como das dimensões horizontais, w e h , da caixa envolvente. Na direção k , a resolução da grade é diretamente proporcional ao número de células da malha nessa direção (n_k). Além disso, fatores α e β são introduzidos para permitir variações na resolução da grade. O fator α ajusta a resolução nas direções horizontais e o fator β ajusta a resolução na direção vertical, conforme definido pelas seguintes equações:

$$nv_i = \left\lceil \frac{\alpha w}{\bar{l}} \right\rceil \quad (4-1)$$

$$nv_j = \left\lceil \frac{\alpha h}{\bar{l}} \right\rceil \quad (4-2)$$

$$nv_k = \lceil \beta n_k \rceil \quad (4-3)$$

Como a coordenada topológica k não é extraída diretamente do *CCT* e as coordenadas topológicas i e j tendem a variar pouco verticalmente, a grade pode ser menos refinada na direção k . Por isso, foi estabelecida uma relação de $\beta = 0,5 \alpha$.

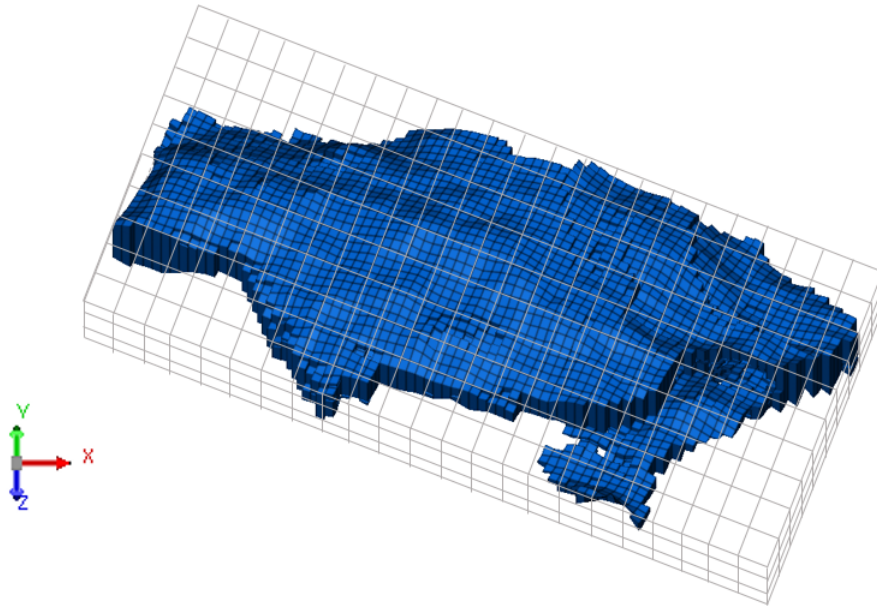


Figura 4.2: *CCT* representado pela grade regular envolvendo a malha irregular de hexaedros.

O principal desafio na construção do *CCT* é atribuir valores apropriados às coordenadas topológicas (i, j) dos vértices da grade em relação à malha. O cálculo do valor exato pode ser feito através de uma busca geométrica, onde dada a posição (x, y, z) de um vértice da grade, aplica-se o método de Newton-Raphson para determinar com precisão as coordenadas paramétricas (s, t, r) do vértice em relação à célula da malha que o contém, determinando-se assim a sua coordenada topológica (i, j) , conforme Equação 3-1. O método de Newton-Raphson também funciona se o ponto estiver fora do domínio da célula, retornando coordenadas paramétricas fora do intervalo $[0, 1]$. Essa característica é crucial para atribuir valores a vértices próximos à borda, mas localizados fora do reservatório. A reconstrução dentro dos *voxels* que intersectam parcialmente o modelo depende de que todos os valores (i, j) atribuídos a seus vértices sejam coerentes.

No entanto, essa abordagem pode ser insuficiente, pois os valores exatos (i, j) podem não ser os mais adequados para minimizar o erro da reconstrução da geometria no interior dos *voxels*. Por esse motivo, optou-se por aplicar um método de otimização baseado no método dos mínimos quadrados para determinar as coordenadas topológicas ótimas que minimizam os erros de reconstrução, utilizando como valores iniciais as coordenadas topológicas exatas obtidas pela busca geométrica, o que acelera a convergência e evita que a solução fique retida em mínimos locais.

A construção do CCT é realizada em uma etapa de pré-processamento,

uma vez que trata-se de um processo lento, sobretudo devido à otimização das coordenadas topológicas. Uma vez calculadas, as coordenadas topológicas da grade são armazenadas na *GPU* em uma textura 3D, que é uma estrutura especializada que organiza dados em um espaço tridimensional, permitindo acessos rápidos e eficientes. Ao acessar essa textura, a interpolação trilinear nativa da *GPU* é aplicada automaticamente, determinando a coordenada topológica de qualquer ponto do domínio sem a necessidade de cálculos adicionais, o que garante um processamento de alto desempenho.

A seguir, serão detalhados o algoritmo para a realização da busca geométrica, o método de Newton-Raphson, necessário para o cálculo das coordenadas paramétricas, e a otimização dos valores das coordenadas topológicas armazenadas na grade.

4.1.1

Busca geométrica

Os passos da busca geométrica, realizada em 3D, são ilustrados em 2D na Figura 4.3. A Figura 4.3a apresenta um exemplo simplificado de um modelo de reservatório, representado por sua malha irregular, juntamente com a grade regular do *CCT*. Durante a busca, percorrem-se as células ativas (I, J, K) da malha e, para cada uma delas, determina-se a área de influência, que é definida pelos voxels que interceptam a caixa envolvente da célula (Figura 4.3b).

Para cada vértice dos voxels dessa área de influência, aplica-se o método de Newton-Raphson, descrito em detalhes na próxima seção, para o cálculo da coordenada paramétrica (s, t, r) em relação à célula corrente.

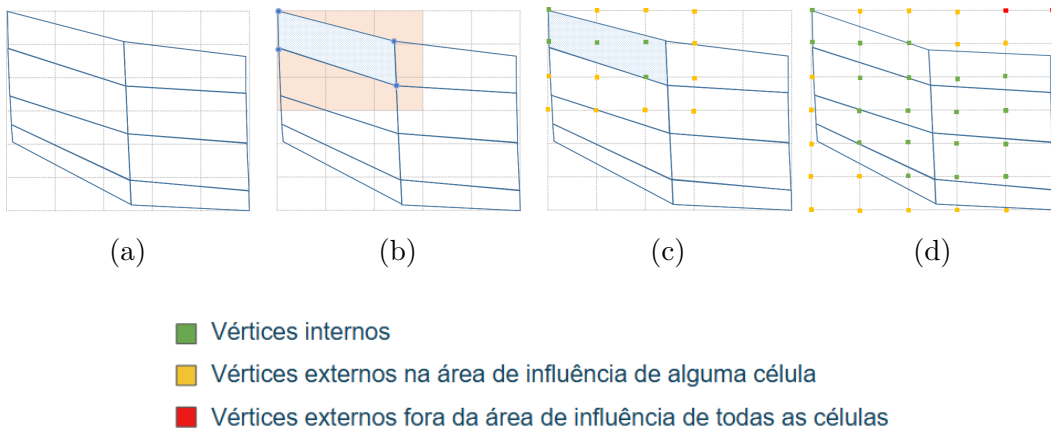


Figura 4.3: Cálculo das coordenadas topológicas dos vértices da grade ilustrada em 2D: (a) Malha irregular do reservatório e grade do *CCT*; (b) Voxels na área de influência de uma célula; (c) Cálculo da coordenada topológica para os vértices da área de influência. (d) Coordenadas topológicas atribuídas a todos os vértices da grade.

Uma vez determinada a coordenada paramétrica do vértice, calcula-se a respectiva coordenada topológica pela Equação (3-1). Caso as coordenadas paramétricas estejam contidas no intervalo $[0, 1]$, o vértice está dentro da célula corrente, como ilustrado pelos vértices verdes da Figura 4.3c. Nesse cenário, atribui-se ao vértice da grade o valor da coordenada topológica calculada e registra-se que a busca foi concluída para ele. Assim, em iterações subsequentes, esses vértices serão ignorados mesmo que esteja dentro da área de influência de outra célula. Por outro lado, se ao menos uma das coordenadas paramétricas estiver fora do intervalo $[0, 1]$, significa que o vértice está fora da célula, como os vértices em amarelo na Figura 4.3c. Nesse caso, escolhe-se a menor distância topológica do vértice às faces da célula corrente. Se ainda não houver uma coordenada topológica atribuída, o vértice recebe a coordenada topológica calculada e registra-se a distância topológica do vértice à célula. Caso contrário, verifica-se se a célula corrente está mais próxima do vértice do que a célula registrada em iterações anteriores. Se for o caso, a coordenada topológica do vértice é atualizada, garantindo que, para vértices da grade situados fora do modelo, a atribuição seja baseada na célula mais próxima.

A existência de células degeneradas na borda do modelo pode, em alguns casos, impedir o cálculo da coordenada topológica pelo método numérico de vértices da grade localizados fora do domínio, acarretando que vértices adjacentes a voxels que interceptam o modelo permaneçam com coordenadas (i, j) indefinidas após a busca geométrica. Sendo assim, esses vértices são identificados e suas respectivas coordenadas topológicas são determinadas por meio de busca nas células associadas aos vértices vizinhos internos ao modelo.

Após a conclusão dessa etapa, os vértices externos da grade que não estiverem dentro da área de influência de nenhuma célula (vértices vermelhos na Figura 4.3d) ainda não possuem valores de coordenadas topológicas. Atribui-se a eles o menor valor finito representável em ponto flutuante de 32 bits, aproximadamente $-3,40 \times 10^{38}$. Esse número está muito além do intervalo típico das coordenadas topológicas reais, que variam de 0 até, no máximo, 10^3 ; consequentemente, qualquer coordenada topológica recuperada em pontos localizados nesses *voxels* será negativa, permitindo classificar imediatamente o ponto como externo ao domínio. Em teoria, um ponto situado em um *voxel* externo pode recuperar uma coordenada topológica positiva se estiver extremamente próximo de um vértice que, embora esteja fora do modelo, tenha recebido valores positivos por ser adjacente a *voxels* que interceptam o modelo. Mesmo nesse cenário, a diferença de ordens de grandeza entre o valor negativo atribuído aos vértices externos ($-3,40 \times 10^{38}$) e as coordenadas topológicas reais limita essa possibilidade de erro a uma região imperceptível

dentro do *voxel*.

4.1.2

Método de Newton-Raphson

O Método de Newton-Raphson é um método numérico iterativo, que utiliza a derivada da função para refinar sucessivamente uma estimativa inicial. Trata-se de um dos métodos numéricos mais utilizados para encontrar raízes de funções não lineares, ou seja, valores de x que satisfazem $f(x) = 0$.

Até este ponto, as coordenadas paramétricas s, t, r foram consideradas no intervalo $[0, 1]$. No entanto, o Método de Newton-Raphson implementado opera com coordenadas paramétricas (ξ, η, ζ) no intervalo $[-1, 1]$, o que é uma convenção comum em formulações numéricas para facilitar a interpolação e o cálculo das derivadas. Após a convergência do processo iterativo, as coordenadas obtidas no intervalo $[-1, 1]$ são convertidas para o intervalo $[0, 1]$, garantindo compatibilidade com a representação paramétrica adotada anteriormente:

$$s = \frac{\xi + 1}{2} \quad (4-4)$$

$$t = \frac{\eta + 1}{2} \quad (4-5)$$

$$r = \frac{\zeta + 1}{2} \quad (4-6)$$

A partir dessas coordenadas paramétricas (ξ, η, ζ) , a posição do ponto no espaço físico (x, y, z) pode ser determinada a partir das coordenadas dos vértices da célula (x_v, y_v, z_v) por meio das seguintes funções de interpolação:

$$x = f(\xi, \eta, \zeta) = \sum_{v=1}^8 N_m(\xi, \eta, \zeta) x_m \quad (4-7)$$

$$y = g(\xi, \eta, \zeta) = \sum_{v=1}^8 N_m(\xi, \eta, \zeta) y_m \quad (4-8)$$

$$z = h(\xi, \eta, \zeta) = \sum_{v=1}^8 N_m(\xi, \eta, \zeta) z_m \quad (4-9)$$

onde, m é o índice do vértice da célula. No caso de células hexaédricas, as funções de forma N_m são dadas por:

$$N_1 = \frac{1}{8}(1 - \xi)(1 - \eta)(1 - \zeta) \quad (4-10a)$$

$$N_2 = \frac{1}{8}(1 - \xi)(1 + \eta)(1 - \zeta) \quad (4-10b)$$

$$N_3 = \frac{1}{8}(1 + \xi)(1 + \eta)(1 - \zeta) \quad (4-10c)$$

$$N_4 = \frac{1}{8}(1 + \xi)(1 - \eta)(1 - \zeta) \quad (4-10d)$$

$$N_5 = \frac{1}{8}(1 - \xi)(1 - \eta)(1 + \zeta) \quad (4-10e)$$

$$N_6 = \frac{1}{8}(1 - \xi)(1 + \eta)(1 + \zeta) \quad (4-10f)$$

$$N_7 = \frac{1}{8}(1 + \xi)(1 + \eta)(1 + \zeta) \quad (4-10g)$$

$$N_8 = \frac{1}{8}(1 + \xi)(1 - \eta)(1 + \zeta) \quad (4-10h)$$

Para o mapeamento inverso, onde deseja-se determinar a coordenada paramétrica (ξ, η, ζ) dado (x, y, z) , recai-se num sistema não linear. Uma forma de resolver esse problema é transformá-lo num problema de determinação de raízes simultâneas, e daí usar o método de Newton-Raphson. As funções cujas raízes (ξ, η, ζ) queremos determinar são:

$$u(\xi, \eta, \zeta) = x - f(\xi, \eta, \zeta) = 0 \quad (4-11)$$

$$v(\xi, \eta, \zeta) = y - g(\xi, \eta, \zeta) = 0 \quad (4-12)$$

$$w(\xi, \eta, \zeta) = z - h(\xi, \eta, \zeta) = 0 \quad (4-13)$$

O método de Newton-Raphson para determinação de raízes de funções escalares é um processo iterativo definido por:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (4-14)$$

onde n representa o índice da iteração.

No caso do elemento hexaédrico, trabalhamos com funções de três variáveis e, portanto, a derivada da função é expressa pela matriz Jacobiana:

$$J = \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \zeta} \\ \frac{\partial v}{\partial \xi} & \frac{\partial v}{\partial \eta} & \frac{\partial v}{\partial \zeta} \\ \frac{\partial w}{\partial \xi} & \frac{\partial w}{\partial \eta} & \frac{\partial w}{\partial \zeta} \end{bmatrix} \quad (4-15)$$

E a iteração de Newton-Raphson, para o problema, é expressa por:

$$\begin{bmatrix} \xi_{n+1} \\ \eta_{n+1} \\ \zeta_{n+1} \end{bmatrix} = \begin{bmatrix} \xi_n \\ \eta_n \\ \zeta_n \end{bmatrix} - J^{-1} \begin{bmatrix} u(\xi_n, \eta_n, \zeta_n) \\ v(\xi_n, \eta_n, \zeta_n) \\ w(\xi_n, \eta_n, \zeta_n) \end{bmatrix} \quad (4-16)$$

4.1.3

Coordenadas topológicas ótimas

O método dos mínimos quadrados [15] é aplicado para otimizar as coordenadas topológicas (i_v, j_v) dos nv vértices da grade, de forma a obter o melhor ajuste possível considerando um conjunto de amostras. Esse método minimiza a soma dos quadrados dos resíduos, ou seja, as diferenças entre as predições do sistema e os dados amostrados. O sistema para i e para j são resolvidos de forma independente.

Para a construção do conjunto de dados amostrados usado na otimização, são geradas ns amostras, tanto internas quanto externas ao domínio do modelo, considerando apenas as células ativas da malha. Para cada amostra, são anotadas suas coordenadas topológicas (i_s, j_s) e suas coordenadas Cartesianas (x_s, y_s, z_s) . O objetivo é reconstruir as coordenadas (i_s, j_s) de todas as amostras baseando-se nos valores armazenados nos vértices do *voxel* (i_v, j_v) . Para cada *voxel*, a reconstrução é realizada através da interpolação trilinear:

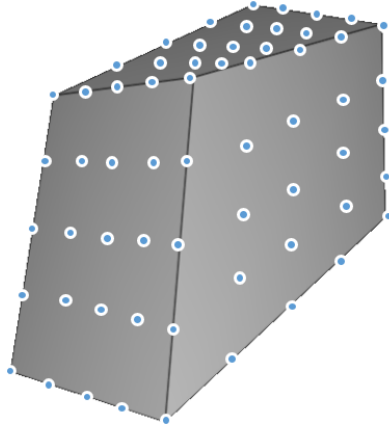
$$i_s(s, t, r) = \sum_{m=0}^8 N_m(s, t, r) i_{v_m} \quad (4-17)$$

$$j_s(s, t, r) = \sum_{m=0}^8 N_m(s, t, r) j_{v_m} \quad (4-18)$$

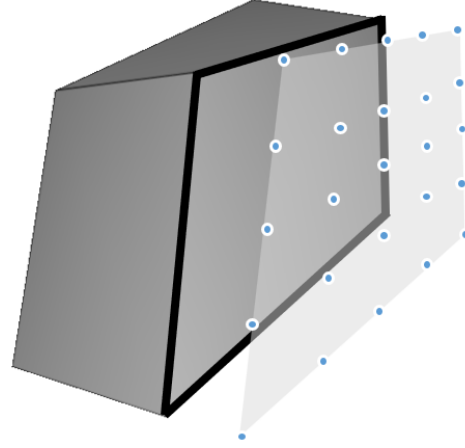
onde N_m são as funções de forma associadas a um elemento finito hexaédrico, nesse caso, o *voxel*, e as coordenadas paramétricas das amostras (s, t, r) , que são calculadas a partir de suas coordenadas Cartesianas (x_s, y_s, z_s) .

Para cada célula ativa do reservatório são geradas amostras nos vértices, nas arestas, nas faces e no interior da célula, conforme ilustra a Figura 4.4a. As amostras são distribuídas no espaço paramétrico $[-1, 1]$, com espaçamentos δ_h e δ_v aplicados, respectivamente, às direções horizontais (plano i - j) e vertical (k). Os valores padrão adotados foram estabelecidos empiricamente como sendo $\delta_h = \delta_v = 0,5$. Caso a distância média entre as amostras em alguma direção ultrapasse 33% da dimensão do *voxel* correspondente, o respectivo parâmetro δ é reduzido pela metade, até que a densidade de amostragem esteja compatível com a resolução espacial do modelo. Além disso, a geração de amostras externas, fora do intervalo $[-1, 1]$, é importante para garantir uma reconstrução precisa nos *voxels* que interceptam parcialmente o modelo

de reservatório. Essas amostras são extrapoladas a partir das células ativas da borda do modelo, seguindo o mesmo espaçamento adotado para as amostras internas.



(a) amostras internas.



(b) amostras externas extrapoladas na direção das faces de borda.

Figura 4.4: Geração de amostras pelas células ativas para o processo de otimização.

A partir das amostras, o sistema inconsistente a ser resolvido pelo método dos mínimos quadrados é construído:

$$\mathbf{A} \cdot \mathbf{i}_v = \mathbf{i}_s \quad (4-19)$$

onde:

- $\mathbf{A}$ é a matriz composta pelos coeficientes $a_{s,v}$ das funções de forma, com ns (número de amostras) linhas e nv (número de vértices) colunas;
- $\mathbf{i}_v$ é o vetor das coordenadas topológicas dos nv vértices da grade, a ser determinado pelo método dos mínimos quadrados;
- $\mathbf{i}_s$ é o vetor das coordenadas topológicas observadas das ns amostras, obtidas a partir das posições reais dos pontos amostrados no modelo.

Explicitamente, a matriz esparsa $\mathbf{A}$, o vetor de incógnitas $\mathbf{i}_v$ e o vetor de observações $\mathbf{i}_s$ podem ser representados como:

$$\begin{bmatrix} a_{0,0} & a_{0,1} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & a_{1,nv-2} & 0 \\ 0 & a_{2,1} & 0 & \cdots & 0 & a_{2,nv-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{ns-2,0} & 0 & 0 & \cdots & 0 & a_{ns-2,nv-1} \\ 0 & 0 & 0 & \cdots & a_{ns-1,nv-2} & a_{ns-1,nv-1} \end{bmatrix} \begin{bmatrix} i_{v0} \\ i_{v1} \\ i_{v2} \\ \vdots \\ i_{nv-2} \\ i_{nv-1} \end{bmatrix} = \begin{bmatrix} i_{s0} \\ i_{s1} \\ i_{s2} \\ \vdots \\ i_{ns-2} \\ i_{ns-1} \end{bmatrix} \quad (4-20)$$

Para garantir uma boa estabilidade numérica e um ajuste confiável, é fundamental que o número de amostras seja significativamente maior que o número de vértices da grade, ou seja, $ns \gg nv$. Isso assegura que o sistema linear tenha redundância suficiente para minimizar os erros de reconstrução. Além disso, um número maior de amostras melhora a regularização da solução, tornando a otimização mais robusta e permitindo uma interpolação mais precisa das coordenadas dentro dos *voxels*.

A matriz $\mathbf{A}$ é esparsa, contendo apenas oito valores não nulos por linha, correspondentes aos termos das funções de forma associados aos oito vértices do *voxel*. Essa esparsidade reflete a estrutura local das relações entre os vértices da grade e as amostras, garantindo que cada amostra influencie apenas um subconjunto reduzido de vértices.

Cada vértice contribui para a reconstrução dentro dos seus oito *voxels* vizinhos. Próximo à fronteira do reservatório, observou-se a existência de poucas amostras localizadas nas proximidades de determinados vértices. Essa configuração de subamostragem leva a um processo de otimização impreciso. Por esse motivo, os vértices que possuem poucas amostras (nos experimentos realizados, foram adotadas três amostras como valor limite) foram eliminados do processo de otimização, atribuindo a eles diretamente as coordenadas exatas previamente calculadas como valor final. Adicionalmente, para *voxels* com menos amostras do que um limite mínimo (nos experimentos realizados, utilizou-se dezesseis amostras), foram incluídas amostras dos *voxels* vizinhos, atribuindo coordenadas paramétricas fora do intervalo $[-1, 1]$. Essas heurísticas contribuem para uma solução numericamente mais robusta.

4.2

Mapa de profundidade

O mapa de profundidade (MP) corresponde à parcela explícita da solução proposta. As coordenadas z (profundidade) dos vértices do modelo são armazenadas em uma textura tridimensional com dimensões igual ao número de vértices do modelo em cada direção ($n_i + 1$, $n_j + 1$, $n_k + 1$), conforme ilustra Figura 4.5. Dada a coordenada topológica (i, j, k) de qualquer ponto no domínio do modelo, calcula-se a coordenada de textura correspondente e, utilizando-se o filtro trilinear nativo da *GPU*, recupera-se a cota z do ponto com um único acesso à essa textura.

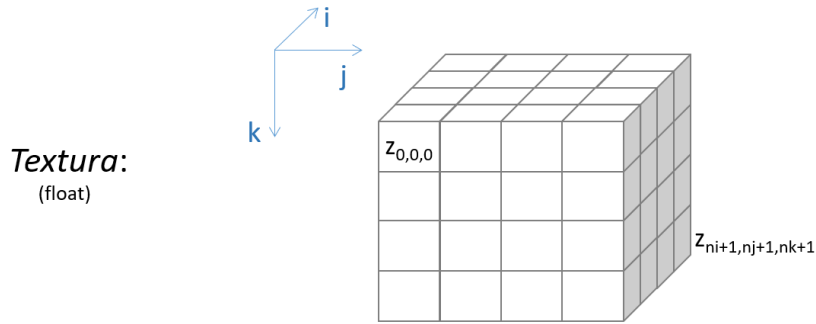


Figura 4.5: Estrutura de dados do mapa de profundidade na *GPU* sem tratamento de discontinuidades.

4.3

Atributos do modelo

A visualização do modelo tem como objetivo permitir a análise e interpretação dos resultados da simulação numérica, o que exige a representação das propriedades físicas e dos dados simulados associados às células ativas da malha. Para isso, é necessário que os atributos das células, correspondentes à identificação de células ativas e suas propriedades escalares também sejam armazenadas em *GPU*. O vetor de ativas (VA) é representado por um *buffer* de tamanho igual ao número total de células do modelo ($ni \times nj \times nk$). Por outro lado, o vetor de propriedades escalares é representado por um *buffer* de tamanho igual ao número células ativas. O vetor de ativas armazena, na posição referente a cada célula ativa, o respectivo índice (id) no vetor de propriedades escalares, onde os valores das propriedades (v) são efetivamente registrados. As células inativas são identificadas por conterem, no vetor de ativas, um índice inválido (Figura 4.6).

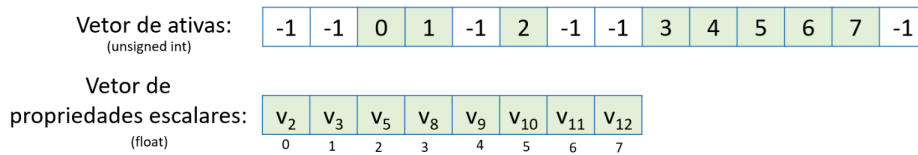


Figura 4.6: Estrutura de dados do Mapa de Ativas e do Mapa de Propriedades Escalares na *GPU*.

Embora a visualização dessas propriedades possa ser feita de forma discreta, célula por célula, uma visualização suavizada proporciona uma compreensão mais intuitiva dos padrões do modelo. Para isso, o valor da propriedade no ponto (x, y, z) é calculado como uma média ponderada da propriedade (v_c) das oito células mais próximas, sendo consideradas no cálculo apenas as células

ativas (Equação (4-21)).

$$v = \frac{\sum_{c=1}^{n_c} w_c v_c}{\sum_{c=1}^{n_c} w_c} \quad (4-21)$$

Os pesos w_c são o inverso da distância paramétrica do ponto ao centro de cada célula, e n_c é o numero de células vizinhas ativas. A Figura 4.7 ilustra em 2D as células mais próximos ao ponto destacado em vermelho.

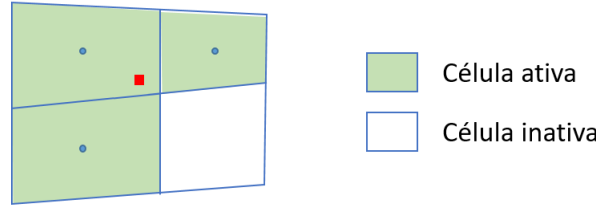


Figura 4.7: Propriedade suavizada do pixel calculada pela média ponderada da propriedade das células ativas mais próximas a ele.

4.4

Memória requerida

O espaço de memória da parcela implícita (memória do CCT), em relação ao espaço total de memória requerida pela representação semi-implícita, sem levar em conta o armazenamento das propriedades escalares, permanece constante para um dado valor de α , independente da resolução da malha. Para essa análise, assume-se uma simplificação para o cálculo aproximado da memória do CCT e adota-se $nv_i = \alpha n_i$, $nv_j = \alpha n_j$ e $nv_k = 0,5 \alpha n_k$. Com isso, a memória das estruturas CCT, MP e VA é calculada pelas seguintes equações, respectivamente:

$$M_{\text{CCT}} = 2,0 (\alpha n_i + 1) (\alpha n_j + 1) (0,5 \alpha n_k + 1) \quad (4-22)$$

$$M_{\text{MP}} = (n_i + 1) (n_j + 1) (n_k + 1) \quad (4-23)$$

$$M_{\text{VA}} = n_i n_j n_k \quad (4-24)$$

A Figura 4.8 apresenta um gráfico com a curva da variação dessa relação em função de α . Com $\alpha = 1$, as dimensões horizontais dos *voxels* se aproximam das dimensões horizontais das células, resultando em uma boa aproximação. Com essa configuração, a memória requerida pelo CCT corresponde a aproximadamente 33% da representação completa (desconsiderando o armazenamento das propriedades escalares).

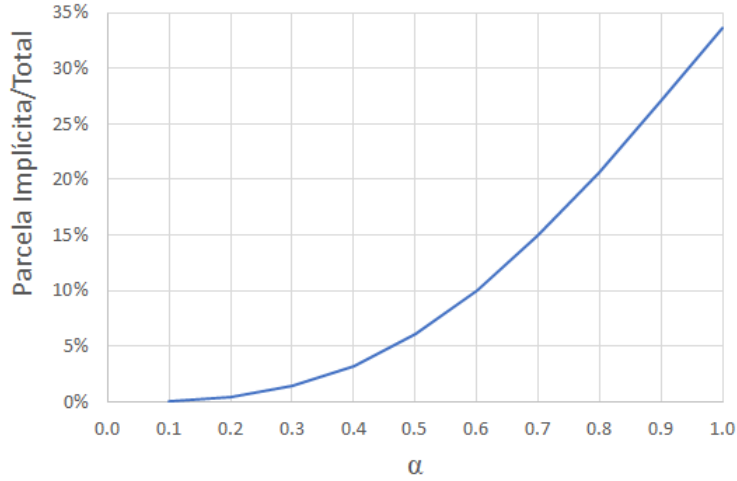


Figura 4.8: Gráfico da variação do espaço de memória da parcela implícita em relação ao espaço de memória total da representação semi-implícita de acordo com o valor de α .

4.5

Localização de pontos na *GPU*

A localização de pontos consiste na conversão das coordenadas globais de um ponto (x, y, z) em coordenadas topológicas (i, j, k) em relação à malha. A Figura 4.9 ilustra o processo proposto.

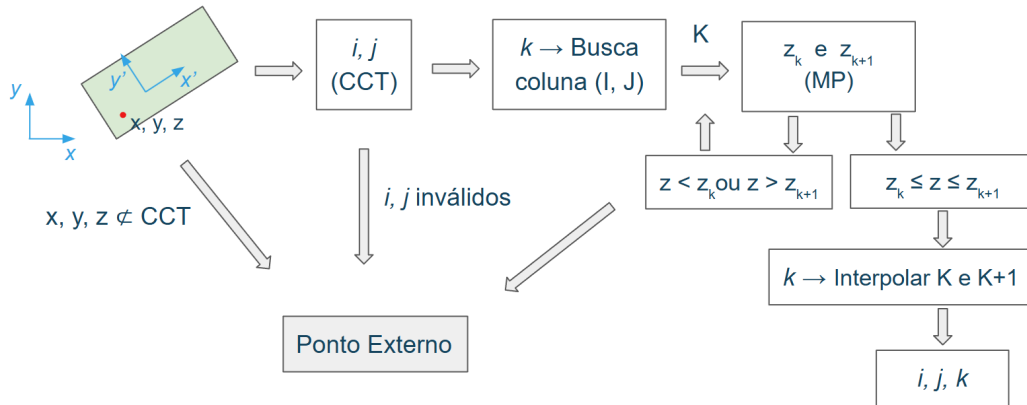


Figura 4.9: Fluxograma do algoritmo de localização de pontos.

Na *GPU*, as coordenadas globais são primeiramente transformadas para o espaço da caixa orientada do CCT e, caso o ponto esteja contido na caixa, calcula-se sua posição normalizada em relação à ela, que corresponde à coordenada de textura utilizada para acessar a textura 3D do CCT. Com um único acesso, utilizando-se o filtro de interpolação trilinear nativo da placa gráfica, as coordenadas topológicas armazenadas nos vértices do *voxel* que

contém o ponto são interpoladas e os valores reconstruídos das coordenadas topológicas (i, j) no ponto são recuperados. Os índices da coluna da malha (I, J) onde o ponto se localiza correspondem à parte inteira de (i, j) , enquanto a parte fracionária define suas coordenadas paramétricas (s, t) .

Uma busca geométrica na coluna (I, J) é realizada para a determinação da coordenada topológica k (Figura 4.10). A busca inicia-se na camada inicial K_0 , a partir da qual o algoritmo obtém as coordenadas nas faces do topo e da base da célula corrente, z_k e z_{k+1} , respectivamente. Essas cotas são recuperadas ao acessar a textura do mapa de profundidade, utilizando-se o filtro trilinear nativo da *GPU*, nas posições dadas pelas coordenadas topológicas (i, j) já recuperadas e pelas coordenadas k do topo e da base da célula corrente. Considera-se que o ponto pertence à célula corrente quando $z_k \leq z \leq z_{k+1}$ (Figura 4.10). Caso contrário, a busca continua nas células vizinhas acima ($z \leq z_k$) ou abaixo ($z \geq z_{k+1}$) da célula corrente até que a célula que contenha o ponto seja encontrada. A princípio, uma busca binária poderia acelerar esse processo. No entanto, algoritmos podem explorar coerência espacial entre os pontos. Algoritmos baseados em traçado de raios, por exemplo, geralmente possuem uma boa estimativa inicial para K_0 , o que reduz significativamente o esforço de busca. Uma vez que a célula contendo o ponto é identificada, a coordenada k do ponto é calculada através de interpolação linear entre z_K e z_{K+1} em relação à coordenada z do ponto. Caso as coordenadas calculadas (i, j, k) correspondam a uma célula inativa ou estejam fora do domínio do reservatório, o ponto é considerado fora do modelo.

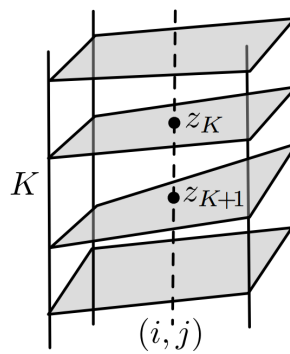


Figura 4.10: Ilustração do processo de busca geométrica pela camada que contém o ponto.

4.6

Tratamento das discontinuidades

Uma premissa assumida pelo *CCT* é que, localmente, a geometria do modelo nas direções i e j é contínua, podendo ser aproximada por uma inter-

poluição trilinear dentro de um *voxel*. No entanto, essa suposição deixa de ser válida em regiões com falhas geológicas, onde ocorrem grandes descontinuidades, como pode ser observado na Figura 4.11.



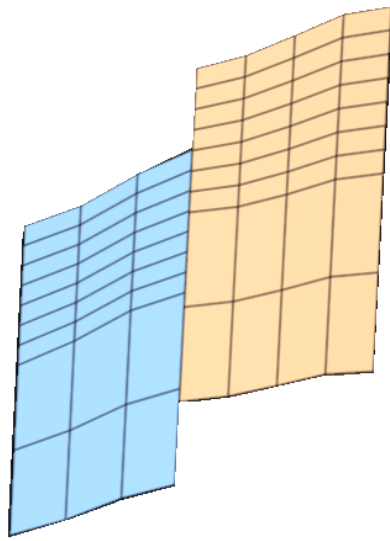
Figura 4.11: Descontinuidade na geometria do modelo, dentro de um *voxel*, causada por uma falha geológica.

Para viabilizar a representação de modelos com descontinuidades, propõe-se a divisão do modelo em blocos de células, cada um composto por regiões contínuas do modelo e representado por um CCT independente (Figura 4.12). As falhas são usadas para delimitar os limites entre esses blocos. Dado que existem descontinuidades apenas no plano $(i-j)$, todas as células de uma mesma coluna (I, J) do modelo pertencem obrigatoriamente a um mesmo bloco. Com isso, os blocos podem ser identificados por um único mapa $n_i \times n_j$, no qual cada entrada indica o bloco ao qual pertence cada coluna (I, J) .

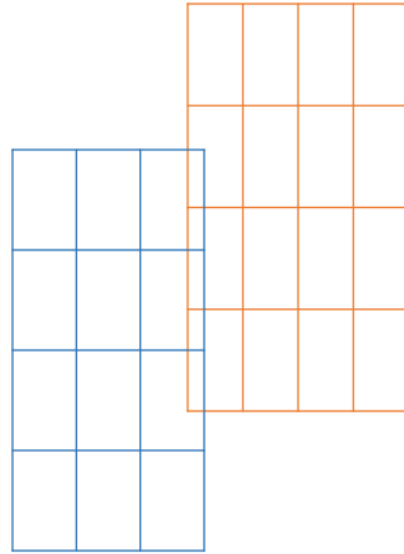
Em uma fase anterior à identificação dos blocos, as falhas internas são prolongadas até a borda do modelo ou até a interseção com outra falha, criando regiões contínuas do modelo delimitadas pelas falhas (Figura 4.13). Cada uma dessas regiões constitui um bloco, formado pelo conjunto de colunas de células contidas em seu interior.

A identificação dos blocos é realizada através de um algoritmo de inundação (*flooding algorithm*) com busca em profundidade que atribui um número de identificação único a todas as colunas conectadas pertencentes a um mesmo bloco, conforme ilustrado na Figura 4.14.

A representação de múltiplos blocos também exige alterações no mapa de profundidade (MP). As descontinuidades geométricas ao longo das falhas impedem que células vizinhas compartilhem as mesmas faces, o que implica que um mesmo vértice (I, J, K) localizado na falha pode apresentar até quatro diferentes valores de z (Figura 4.15b). Isso exige uma adaptação na estrutura de dados do MP, uma vez que, na textura, é armazenado um único valor de z

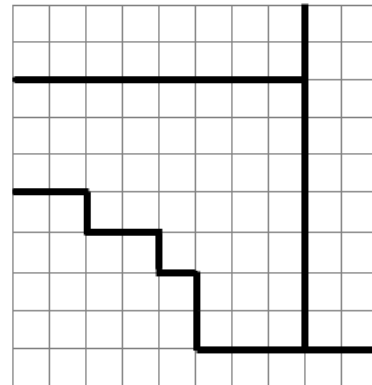
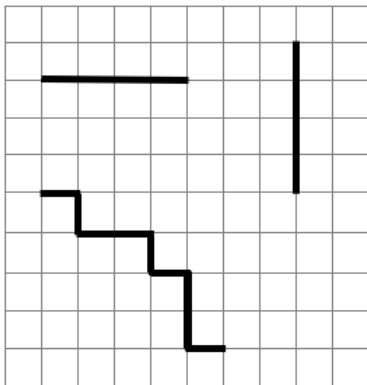


(a) Regiões contínuas do modelo



(b) Grade do CCT dos blocos.

Figura 4.12: Representação do modelo por múltiplos blocos.

Figura 4.13: Mapa 2D de falhas no plano $i - j$, ilustrando o prolongamento das falhas.

por vértice. Um *buffer* auxiliar é utilizado para armazenar os quatro valores de z dos vértices de falha, sempre seguindo a mesma ordenação, conforme ilustra a Figura 4.15c. Na posição da textura referente a um vértice de falha, ao invés de armazenar a profundidade, é armazenado um valor (L) que codifica o índice (id) do respectivo vértice no *buffer* de vértices de falha. A estrutura de dados do MP adaptada para a representação das falhas é ilustrada na Figura 4.15d.

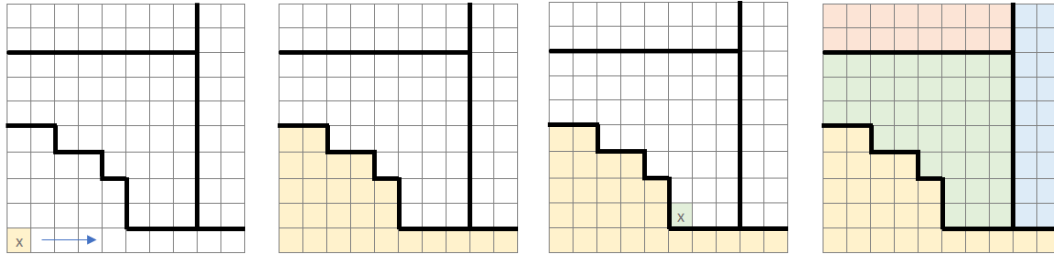
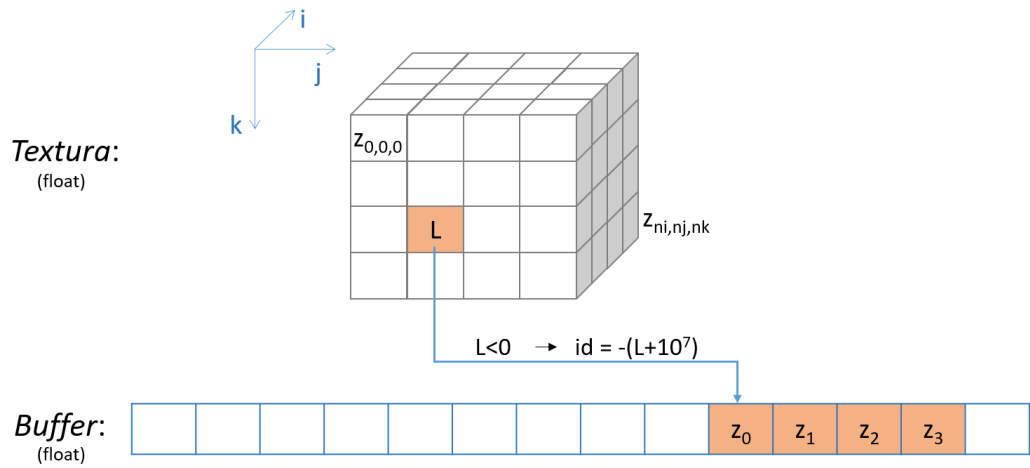
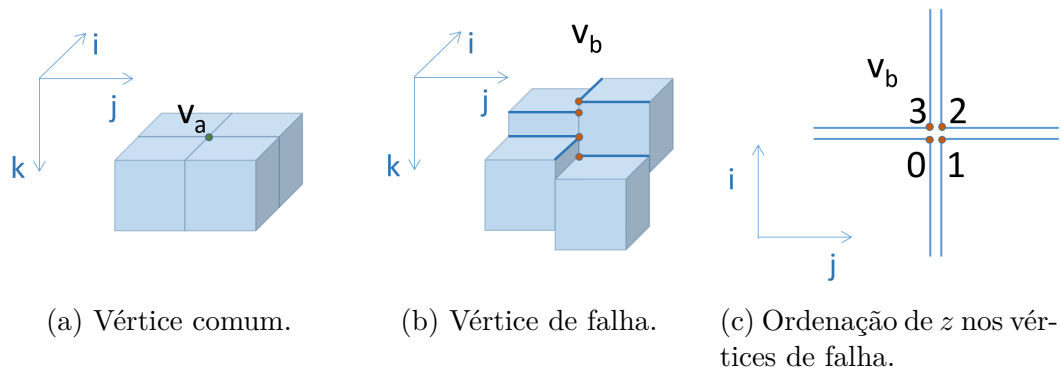


Figura 4.14: Ilustração da aplicação do algoritmo de inundação para a identificação das colunas dos blocos.



(d) Estrutura de dados do mapa de profundidade na *GPU* com tratamento de descontinuidade.

Figura 4.15: Mapa de profundidade com descontinuidade.

O valor codificado (L) é sempre negativo para diferenciá-lo das profundidades regulares, as quais são sempre positivas. Além disso, a codificação incorpora uma constante de grande magnitude para garantir que a interpolação na textura do MP de pontos localizados em células vizinhas à falha resulte sempre em valores negativos:

$$L = -(id + 10^7) \quad (4-25)$$

Nessas células, a profundidade z no ponto deve ser calculada à parte (Figura 4.16). Para isso, dado um ponto definido pela coordenada topológica (i, j, k) , primeiro determina-se, com a Equação 3-1, os índices (I, J, K) da célula que o contém e as respectivas coordenadas paramétricas (s, t, r) . As profundidades dos oito vértices da célula são consultadas na textura do MP. Para os vértices de falha que armazenam valores negativos, o valor real da profundidade é obtido através do *buffer* de vértices de falha. O valor negativo corresponde ao valor codificado (L) e a posição (id) do vértice de falha no *buffer* é determinada a partir da Equação 4-25. Após o cálculo da profundidade z de todos os vértices da célula, interpola-se trilinearmente esses valores, utilizando-se as funções de forma apresentadas nas Equações 4-10, a fim de determinar a profundidade z no ponto.

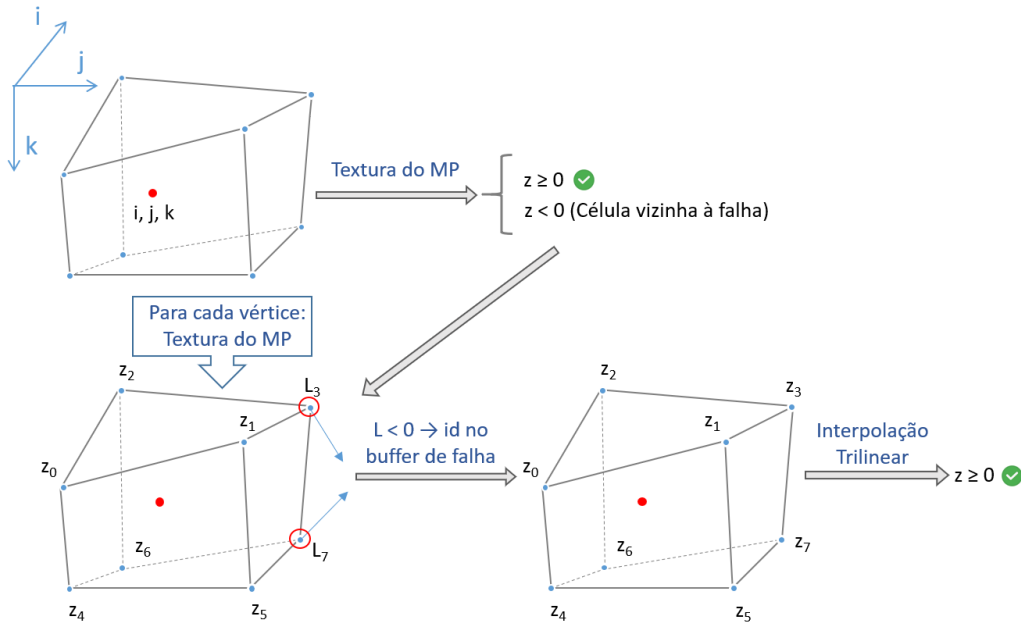


Figura 4.16: Fluxograma do algoritmo de obtenção de z nas células vizinhas à falha através do MP com descontinuidade.

Embora, em teoria, o método de identificação de pontos no interior de células vizinhas à falha possa falhar quando o ponto avaliado está extremamente próximo de um vértice regular dessa célula — situação em que a interpolação trilinear poderia resultar num valor de profundidade positivo — isso não se verifica na prática. Essa situação é análoga ao observado no CCT ao recuperar a coordenada topológica em pontos externos ao modelo, conforme discutido anteriormente.

Na localização de pontos em modelos com múltiplos blocos, é realizada

uma busca por bloco, sendo que dentro de cada bloco, o processo é praticamente o mesmo já descrito previamente para modelos sem descontinuidades, exceto pela forma que a coordenada z é obtida quando o ponto está numa célula vizinha à falha. (Figura 4.17). A diferença principal é que, ao finalizar a busca em um bloco, caso o ponto não seja localizado em nenhuma de suas células, a busca continua nos demais blocos do modelo até que o ponto seja encontrado ou que todos os blocos tenham sido percorridos. A localização de pontos fora do modelo representa um caso extremo, onde é necessário percorrer todos os blocos para constatar que o ponto não pertence ao modelo.

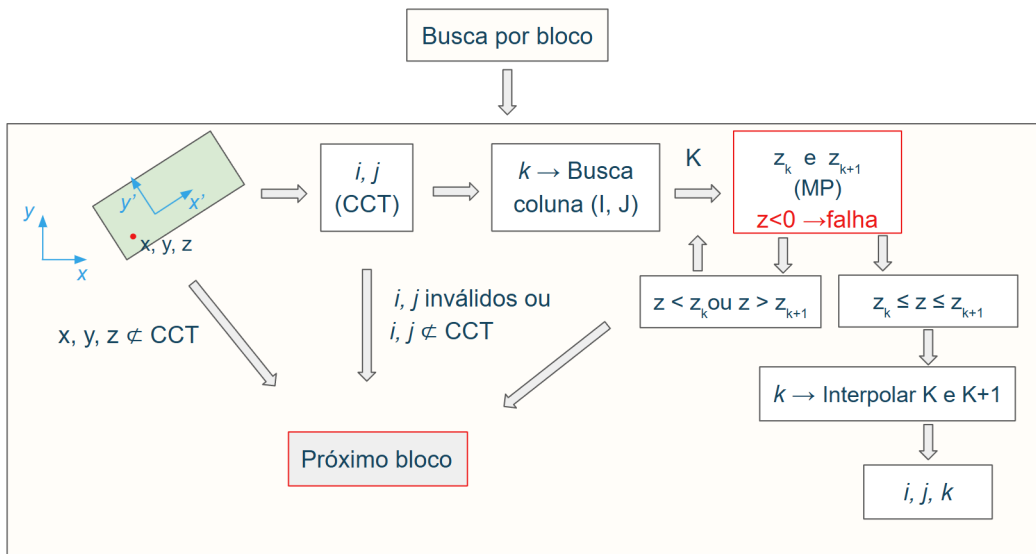


Figura 4.17: Fluxograma do algoritmo de localização de pontos.

Em modelos altamente falhados, o número de blocos pode ser muito grande, com alguns blocos contendo apenas poucas colunas (Figura 4.18). Na localização de pontos, verificar cada bloco para determinar se o ponto está contido ou não torna-se impraticável. Diante disso, foi elaborada uma estrutura de aceleração que consiste em uma única grade 2D cobrindo todo o domínio do reservatório. Essa grade 2D subdivide o plano $i - j$ em células regulares, cada uma contendo a lista de blocos com os quais faz interseção (Figura 4.19). Como as grades dos CCTs de cada bloco são alinhadas com o eixo z , a interseção é determinada apenas pela projeção dos blocos nas direções x e y . Em cada célula, a lista de blocos é ordenada em ordem decrescente pelo número de colunas que compõem cada bloco. Essa ordenação permite que blocos com maior número de células sejam processados primeiro na localização de pontos, otimizando o desempenho nessa etapa. A natureza bidimensional da estrutura de aceleração resulta em um uso marginal de memória.

Dada as coordenadas (x, y) de um ponto, calcula-se o *voxel* da grade 2D

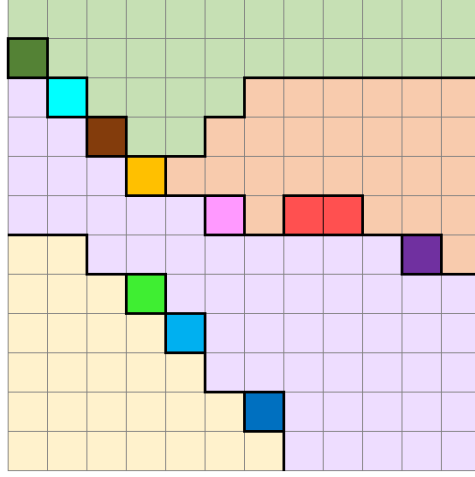


Figura 4.18: Mapa de colunas de uma região altamente fraturada do modelo, resultando em um grande número de blocos, sendo cada um representado por uma cor diferente.

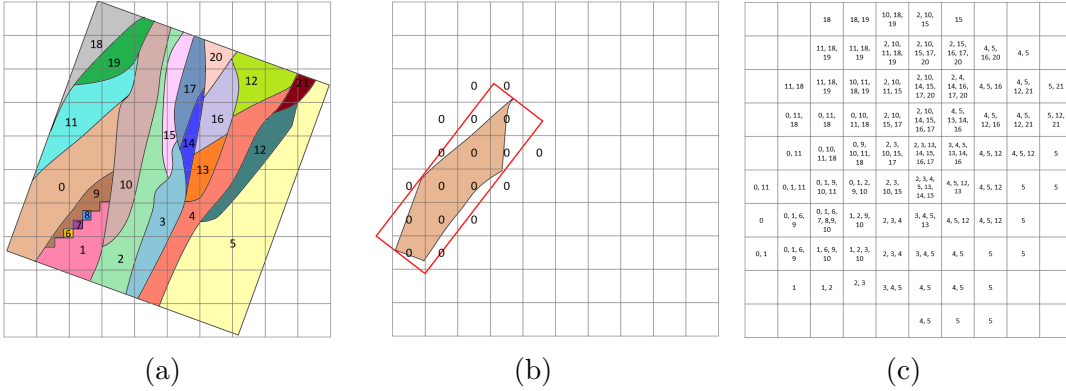


Figura 4.19: Estrutura de aceleração: (a) Modelo subdividido em blocos, identificados por diferentes cores, e grade regular 2D da estrutura de aceleração; (b) Cálculo da interseção da grade do CCT de cada bloco com as células da estrutura de aceleração; (c) Células da estrutura de aceleração contendo a lista de blocos com o qual há interseção.

onde ele se localiza. Com essa informação, tem-se a lista dos blocos nos quais a busca deve ser realizada. Dessa forma, pontos localizados fora do modelo, que não pertencem à caixa orientada de nenhum bloco, são imediatamente descartados.

São utilizados dois *buffers* para armazenar as informações da estrutura de aceleração. O primeiro *buffer* armazena, para cada *voxel*, o número de blocos (n) interceptados e o índice (id) correspondente no segundo *buffer*. O segundo *buffer* contém os índices dos blocos associados a cada *voxel* (Figura 4.20).

Um atlas de texturas tridimensional, que reúne várias texturas menores, é utilizado para armazenar os CTTs de todos os blocos na *GPU* em uma única estrutura. Essa abordagem apresenta um desafio em termos de alocação

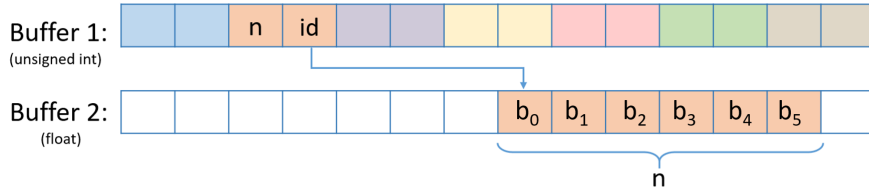


Figura 4.20: Estrutura de dados da estrutura de aceleração na GPU.

eficiente do espaço. A disposição das informações de cada bloco na textura assemelha-se a um problema clássico de otimização combinatória conhecido como *Bin Packing*. No contexto do atlas de textura, o problema envolve a organização de um conjunto de itens de tamanhos diferentes — neste caso, os blocos — dentro de um único recipiente expansível, que corresponde à textura 3D.

Como os blocos são separados apenas por descontinuidades verticais, as dimensões das texturas dos blocos na direção k tendem a ser semelhantes, enquanto as maiores variações ocorrem nas direções i e j . Sendo assim, o problema pode ser simplificado para um problema bidimensional, onde os blocos são posicionados ao longo do plano $i - j$. Para lidar com esse desafio, considera-se que a textura do atlas é expansível apenas na direção i . Foi adotada a heurística *First Fit Decreasing* [18], onde os blocos são ordenados pelo maior tamanho na direção j e inseridos no primeiro espaço disponível que tenha capacidade suficiente para acomodá-lo. A ordenação ajuda a otimizar o uso do espaço, garantindo que os blocos maiores sejam alocados primeiro, o que pode reduzir a fragmentação e melhorar a compactação dos dados.

As dimensões iniciais do atlas são definidas a partir das maiores dimensões (nv_i, nv_j, nv_k) dos blocos nas direções (i, j, k) , respectivamente. Essa escolha inicial visa acomodar os blocos mais volumosos de forma eficiente desde o início. Quando não há espaço suficiente para acomodar um bloco, a textura é expandida na direção i . A Figura 4.21 ilustra o processo em um modelo com quatro blocos. Dado um bloco, para a verificação da existência de espaço disponível no atlas, deve-se seguir os seguintes passos:

1. Obter dimensões do bloco a ser posicionado.
2. Percorrer todas as posições possíveis na textura do atlas nas direções i e j .
3. Verificar se a posição está livre.
4. Checar sobreposição com blocos já posicionados.

5. Retornar à posição encontrada se não houver sobreposição.
6. Sinalizar necessidade de expandir a textura se nenhuma posição válida for encontrada.

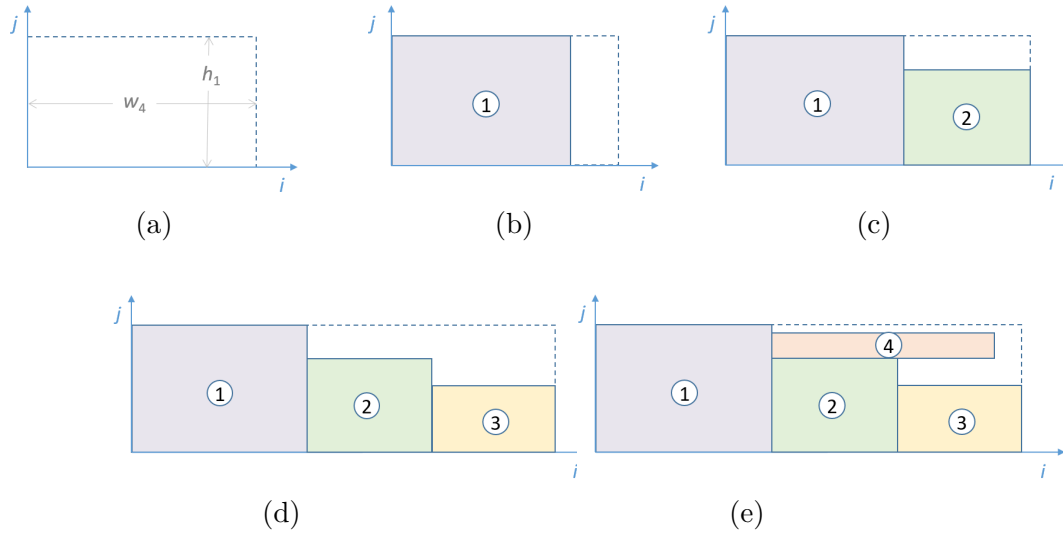


Figura 4.21: Alocação dos blocos na textura única do atlas com o CCT de todos os blocos do modelo: (a) Inicialização da textura com as maiores dimensões dos blocos; (b) e (e) Armazenamento do bloco no espaço disponível; (c) e (d) Expansão da textura para alocação dos blocos;

A estrutura de dados da representação semi-implícita considerando as descontinuidades é ilustrada na Figura 4.22.

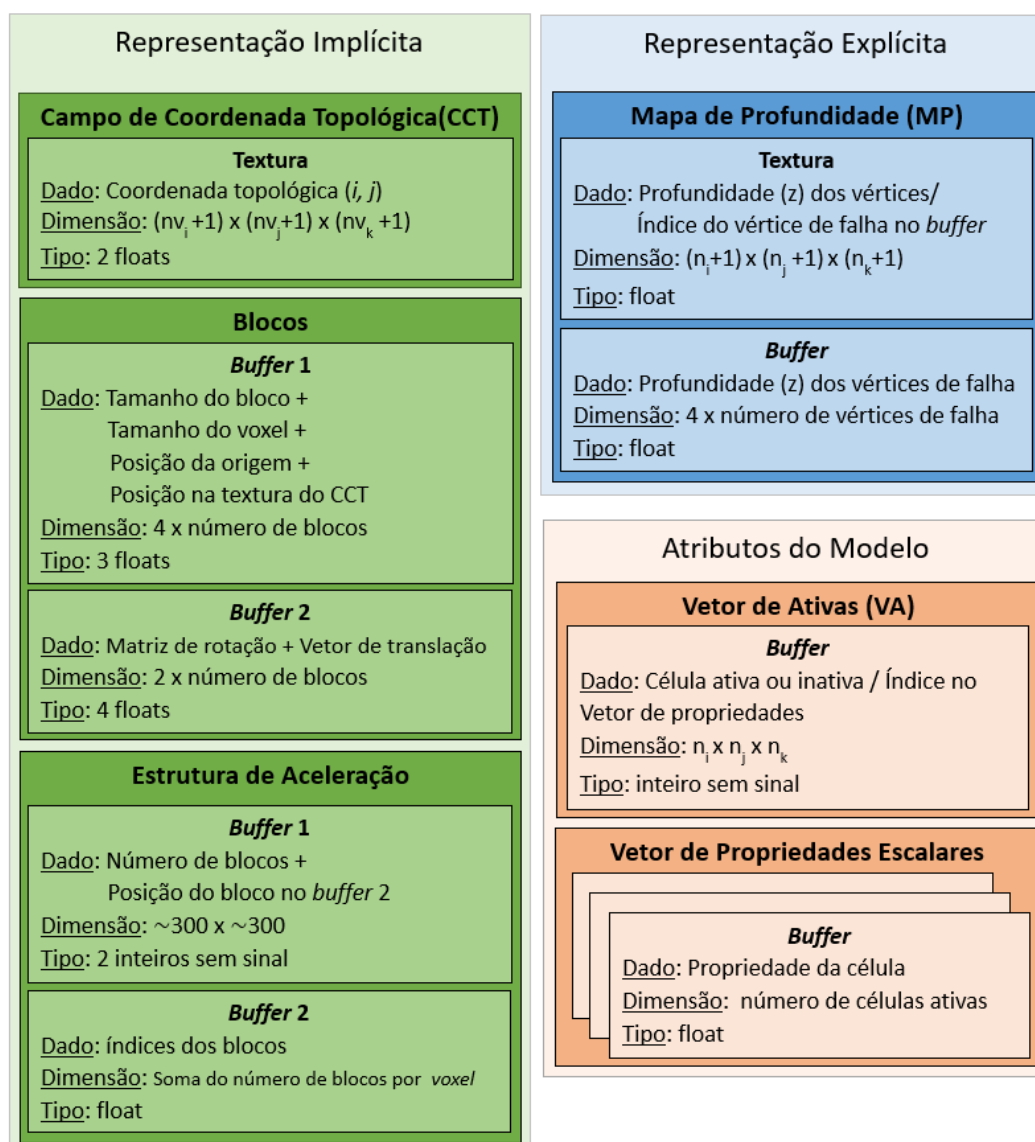


Figura 4.22: Estrutura de dados da representação semi-implícita considerando descontinuidades e dos atributos do modelo.

5

Resultados

Um conjunto de experimentos foram realizados para avaliar a representação semi-implícita proposta. Todos os testes computacionais foram realizados em um processador Intel Core i9-13900KF 3.00 GHz e 128 GB de memória RAM, com uma placa de vídeo NVIDIA GeForce RTX 4090, que possui 24 GB de memória dedicada.

A seguir, na primeira seção, são descritos os modelos utilizados nos experimentos e, na seção seguinte, apresenta-se a memória requerida para o armazenamento em GPU da proposta, comparando com diferentes representações. Na sequência, é realizada a análise da precisão e eficiência na localização de pontos utilizando-se a representação semi-implícita através da avaliação de um conjunto de amostras geradas randomicamente no domínio de cada modelo. Posteriormente, é avaliada a aplicação da solução proposta na renderização dos modelos através da visualização volumétrica, tendo sido analisados o desempenho computacional e a qualidade da imagem gerada, como forma de avaliar o impacto do erro nas coordenadas topológicas na qualidade da visualização dos modelos. Por fim, como um exemplo mais geral de aplicação, além do escopo da visualização, apresenta-se uma aplicação para traçado de linhas de corrente através da solução proposta.

5.1

Modelos

Foram utilizados sete modelos reais de reservatório, com diferentes ordens de grandeza e características, para avaliar a representação proposta. A Tabela 5.1 apresenta as dimensões da malha de hexaedros de cada modelo, o número de células ativas de cada malha e o número de blocos necessários para representá-los com a estrutura semi-implícita.

Os modelos M1, M2, M3, M4, M5 e M6 são modelos utilizados na análise de fluxo do reservatório. O modelo M7, por outro lado, é projetado para a análise de captura e armazenamento de carbono (CCS - Carbon Capture and Storage). Os dois primeiros modelos são pequenos, sendo o modelo M1 usado principalmente em contextos acadêmicos como referência para simulação de reservatórios. No entanto, esses modelos apresentam geometrias complexas, o

Tabela 5.1: Características dos modelos reais usados nos experimentos: Identificação, resolução da malha de hexaedros, número de células ativas e número de blocos requeridos pela representação semi-implícita.

Modelo	Resolução			Número de células ativas	Número de blocos
	n_i	n_j	n_k		
M1	46	69	30	42.353	9
M2	142	39	25	66.559	10
M3	60	133	77	246.637	16
M4	201	341	71	1.134.131	77
M5	281	330	57	1.265.688	9.194
M6	346	352	100	6.245.219	72
M7	343	326	367	16.667.310	4.100

que impõe desafios à acurácia da representação implícita. O terceiro modelo representa um campo petrolífero de porte médio. Os três modelos seguintes correspondem a campos atuais de grande porte, apresentando geometrias mais regulares. O modelo M5 requer um número elevado de blocos, devido à existência de inúmeras falhas geológicas. O último modelo, por ser um modelo de CCS, é tipicamente de grande escala, pois precisa simular a dispersão do CO₂ em formações geológicas e garantir a segurança e a eficácia do armazenamento a longo prazo [5]. A Figura 5.1 apresenta vistas com detalhes dos modelos, ilustrando suas características geométricas. Observa-se que a geometria das células nas direções i e j tende a ser mais irregular em modelos com menor resolução. A Figura 5.2 exibe detalhes da geometria irregular encontrada nos modelos M1 e M2.

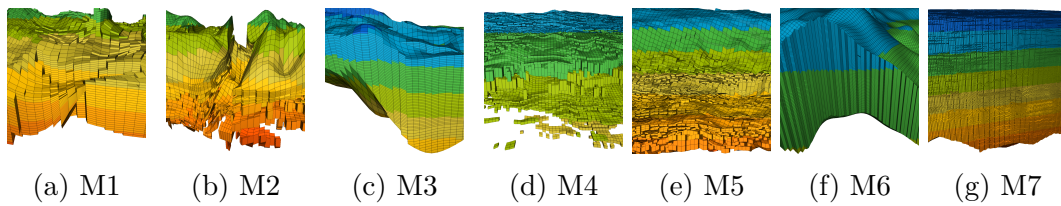


Figura 5.1: Detalhes dos modelos reais de reservatório de petróleo usados nos experimentos.

Conforme mencionado no capítulo anterior, a construção do CCT é um processo lento e, por isso, é realizada em uma etapa de pré-processamento. A Tabela 5.2 apresenta o tempo detalhado gasto nesse processo para os modelos utilizados na avaliação dos resultados. A etapa que demanda maior tempo é a otimização das coordenadas topológicas. Embora essa demora seja uma

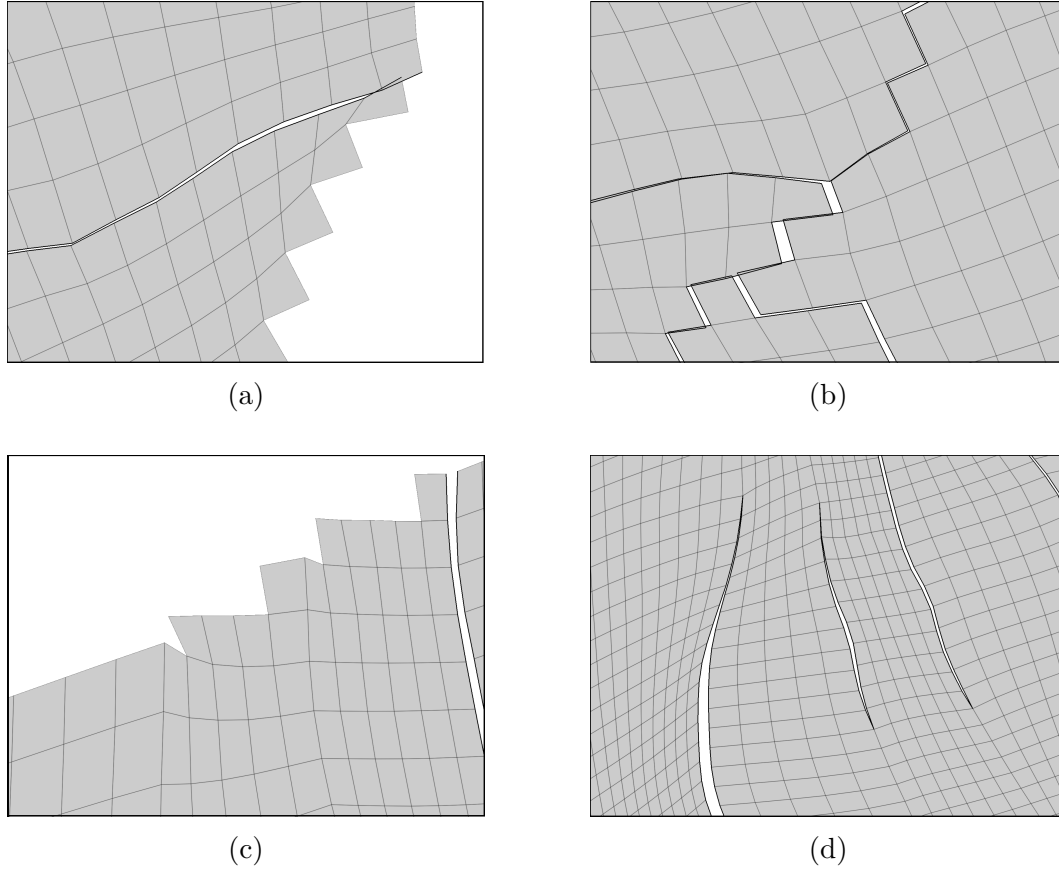


Figura 5.2: Vista em mapa da geometria irregular das células nas direções i e j : (a) e (b) Modelo M1; (c) e (d) Modelo M2.

limitação para a construção da representação proposta em tempo real, ela não compromete a aplicação da solução, uma vez que, após a construção do modelo de reservatório, sua geometria raramente é modificada durante as simulações. Dessa forma, o CCT necessita ser construído uma única vez para cada modelo.

5.2

Memória requerida

Um requisito fundamental para que seja viável armazenar na *GPU* a representação de um modelo de reservatório de larga escala é que ela seja compacta, apresentando um consumo de memória reduzido. Como visto anteriormente, a resolução do CCT varia de acordo com o fator α adotado (Equações 4-1, 4-2 e 4-3). Para a avaliação dos resultados foi adotado $\alpha = 1$ na construção do CCT, exceto quando explicitamente mencionado um valor diferente. A memória requerida pela solução proposta foi comparada a duas outras estruturas: a primeira é a estrutura de dados convencional, que armazena explicitamente as incidências e as adjacências das células; a segunda é a *ResGPU*, proposta por *Franceschin et al.* [12], que explora as características de malhas de hexaedros estruturadas e obtém, de forma implícita, a incidência e a adja-

Tabela 5.2: Tempo (minutos) detalhado e total na construção do CCT. Foram medidos os tempos nas seguintes etapas: algoritmo de *Bin Packing* para organização dos blocos na textura, busca geométrica para cálculo das coordenadas topológicas exatas e aplicação do método dos mínimos quadrados para a determinação das coordenadas topológicas ótimas.

Modelo	Bin Packing	Coordenada exatas	Coordenadas ótimas	Total
M1	0.0	0.0	4.1	4.1
M2	0.0	0.0	5.2	5.2
M3	0.0	0.0	13.7	13.7
M4	0.0	0.3	106.2	106.5
M5	14.6	0.6	215.1	230.3
M6	0.0	5.1	272.3	277.4
M7	2.1	35.5	688.7	726.3

cência das células não vizinhas às falhas. Entretanto, não foi possível construir a representação *ResGPU* para o modelo M7, que é o maior modelo. A Tabela 5.3 apresenta a memória requerida por cada uma dessas estruturas e a comparação entre a representação proposta e as demais. Observa-se que, para todos os modelos analisados, a representação semi-implícita reduz significativamente o consumo de memória.

Tabela 5.3: Comparação do uso de memória, em MB, entre a estrutura de dados de incidência convencional, o método ResGPU [12] e a representação semi-implícita. Os valores entre parênteses indicam o uso relativo de memória da representação semi-implícita em comparação com os outros métodos.

Modelo	Incidência	ResGPU	Semi-implícita
M1	3,18	3,73	1,47 (46% 39%)
M2	5,08	6,06	2,67 (52% 44%)
M3	17,21	18,11	9,83 (55% 52%)
M4	84,24	106,29	26,57 (77% 62%)
M5	104,54	149,72	79,63 (76% 53%)
M6	406,71	185,99	111,65 (27% 60%)
M7	1363,87	-	406,04 (30% -)

5.3

Desempenho e acurácia na localização de pontos

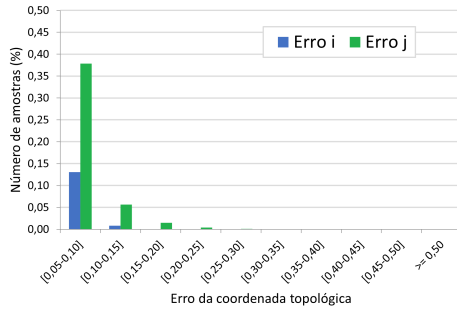
Para os testes de desempenho computacional e acurácia das coordenadas localizadas, foram gerados randomicamente 10^8 pontos localizados no interior das células ativas de cada modelo. Para cada célula (I, J, K) , foi selecionado um número de pontos proporcional ao seu volume. Cada amostra é composta pelas coordenadas Cartesianas do ponto (x, y, z) , obtido a partir de coordenadas (s, t, r) aleatórias, e as correspondentes coordenadas topológica globais originais do ponto (i, j, k) . Esses pontos foram transferidos para a GPU e, para a avaliação de desempenho, foi medido o tempo gasto na localização de todos eles com a solução proposta e com o método *Point Location with Topological Search (PLTS)* que é próprio para a estrutura ResGPU [12]. O experimento foi repetido 100 vezes para a determinação do tempo de execução médio. Como mostrado na Tabela 5.4, a representação semi-implícita apresentou desempenho até nove vezes superior ao da proposta ResGPU, resultando em um ganho de performance significativo. Observa-se também que o tempo gasto para localizar os pontos com a representação proposta não varia de forma significativa, apesar das complexidades geométricas do modelo. Os tempos do método *PLTS* apresentam um desvio padrão de 7,84 ms, contra um valor de 0,73 ms da estrutura proposta. Esse fato ilustra claramente a vantagem de se utilizar uma representação semi-implícita.

Tabela 5.4: Comparação de performance na localização de pontos nos modelos de reservatório com a estrutura ResGPU e com a representação semi-implícita. O valor representa o tempo médio necessário para a localização de 10^8 pontos na GPU.

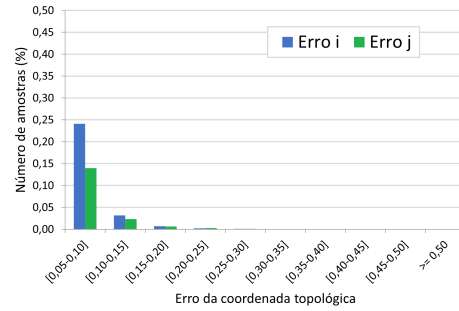
Modelo	Tempo (milissegundos)		Ganho de Desempenho
	ResGPU	Proposta	
M1	34,3	4,9	7,1
M2	38,7	4,5	8,4
M3	38,5	4,8	8,0
M4	58,1	5,5	10,6
M5	39,3	6,9	5,7
M6	36,9	5,4	6,8
M7	-	5,6	-

Para avaliar o erro de aproximação da solução proposta, foram medidas as diferenças nas coordenadas topológicas (i, j) recuperadas através da localização

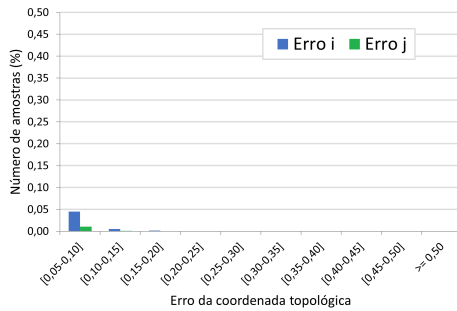
de pontos. A Figura 5.3 apresenta a distribuição dos erros superiores a 0,05 em i e j (de modo geral, mais de 99,5% das amostras apresentam erros abaixo de 0,05) para os quatro primeiros modelos. Para os três últimos modelos, não foram observados erros superiores a esse limite. Os erros são mais significativos nos dois primeiros modelos, devido às geometrias complexas sendo aproximadas por interpolação trilinear, e no modelo M4, em função de pequenos aglomerados isolados de células próximos à fronteira do reservatório.



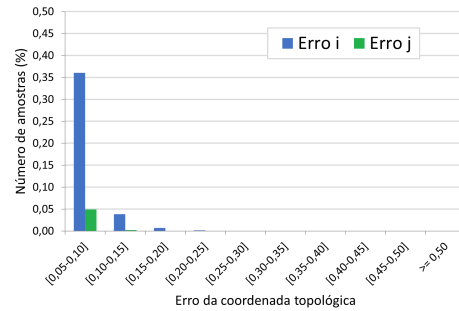
(a) M1



(b) M2



(c) M3



(d) M4

Figura 5.3: Erros de aproximação nas coordenadas topológicas (i, j) recuperadas para os quatro primeiros modelos de reservatório. Os gráficos mostram a distribuição dos erros superiores a 0,05 (mais de 99,5% das amostras apresentam erros abaixo desse limite). Nenhum erro acima de 0,05 foi observado para os três últimos modelos.

Conforme mencionado anteriormente, a estrutura de dados do campo de coordenadas topológicas (CCT) corresponde a aproximadamente 33% da memória necessária para a representação completa do modelo. Isso permite ajustar a resolução do CCT para equilibrar melhor o uso de memória e a acurácia. Por exemplo, a distribuição mostrada na Figura 5.4 demonstra como os erros diminuem quando a resolução do CCT é duplicada para o modelo M4 ($\alpha = 2$, aumentando o uso total de memória em 25%) e, por outro lado, como

os erros permanecem aceitáveis mesmo quando a resolução do CCT é reduzida pela metade no modelo M5 ($\alpha = 0,5$, reduzindo o uso de memória em 17%).

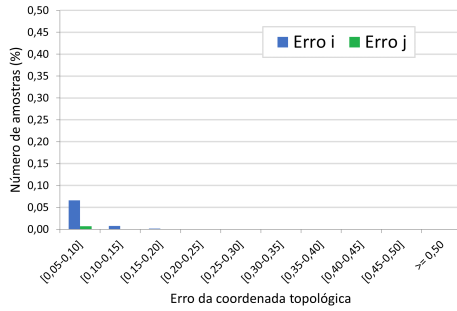
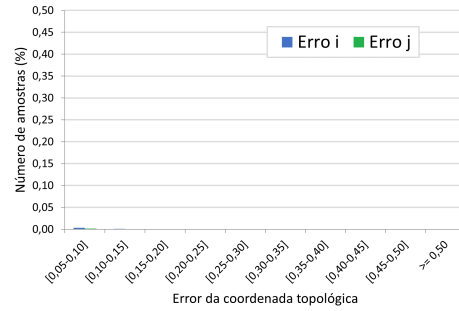
(a) M4 com $\alpha = 2$ (b) M5 com $\alpha = 0,5$

Figura 5.4: Distribuição atualizada dos erros maiores que 0,05 nas coordenadas topológicas (i, j) recuperadas após modificação da resolução do campo de coordenada topológica (CCT) para os modelos M4 e M5.

Esses erros nas coordenadas topológicas (i, j) induzem imprecisões no plano do horizonte. As coordenadas (i, j) recuperadas também indicam a coluna onde será realizada a busca pela coordenada de profundidade. No entanto, não foram observados erros significativos na coordenada de profundidade, embora a coordenada topológica k possa naturalmente apresentar discrepâncias em regiões com células degeneradas: em células com espessura próxima ou igual a zero, diferenças nos valores de k não implicam erros nas coordenadas z .

Um ponto adicional de atenção é a identificação incorreta de pontos internos ou externos ao modelo. Pontos internos próximos à fronteira do modelo podem ser classificados erroneamente como externos, e vice-versa. A Tabela 5.5 apresenta o percentual de pontos internos classificados como externos no experimento original para os seis modelos. Como veremos, essa classificação incorreta é a principal causa de artefatos visuais em imagens geradas por técnicas de visualização. No entanto, a quantidade de pontos nessa situação permanece dentro de limites aceitáveis.

Tabela 5.5: Porcentagem de pontos que foram classificados incorretamente como externos.

M1	M2	M3	M4	M5	M6	M7
0.16%	0.27%	0.08%	0.29%	0.15%	0.01%	0.05%

5.4

Visualização volumétrica

É necessário avaliar se os erros relatados na seção anterior impactam significativamente o resultado das técnicas de visualização do modelo ou se a economia de memória e o ganho de desempenho justificam a abordagem proposta. Para essa avaliação, foi escolhida a técnica de visualização volumétrica, uma das mais exigentes em termos de recursos para inspecionar a variação de propriedades escalares no interior do volume. Os algoritmos mais comuns de visualização volumétrica são baseados em traçado de raios, onde um raio é traçado para cada pixel da tela e a cor resultante é obtida a partir da combinação da cor de várias amostras avaliadas ao longo do raio (Figura 5.5) [22].

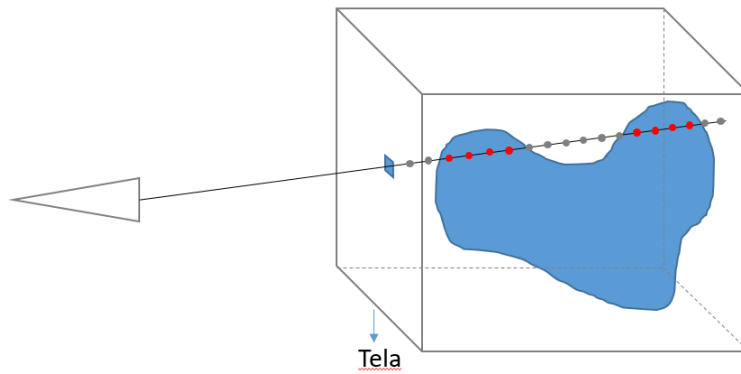


Figura 5.5: Algoritmo de traçado de raios, com amostras geradas ao longo do raio.

O trabalho de Campagnolo e Celes [7] é utilizado como referência nessa análise. Nele, a estrutura de dados ResGPU [12] é utilizada na representação dos modelos de reservatório e são comparadas três estratégias do algoritmo de visualização volumétrica baseado em traçado de raios: *Structured Resampled Grid (SRG)*, *Linear Interpolation with Barycentric Coordinates (LIBC)* e *Point Location with Topological Search (PLTS)*. A estratégia *SRG* foi excluída da análise devido à baixa qualidade das imagens geradas. A abordagem *LIBC* melhora o desempenho ao aproximar cada face da célula com um par de triângulos, enquanto a *PLTS* utiliza um algoritmo preciso de localização de pontos baseado em Newton-Raphson, proporcionando alta acurácia (utilizado aqui como referência) em troca de um baixo desempenho.

No experimento realizado, as estratégias *LIBC* e *PLTS* são comparadas com a representação semi-implícita na localização dos pontos ao longo do raio. O passo de integração, que representa a distância entre as amostras, foi fixado em $0,5\bar{h}$ — sendo $\bar{h}$ a altura média das células ($\bar{h}$) de cada modelo — o que seria o suficiente para gerar ao menos uma amostra na maioria das

células interceptadas pelo raio. Na representação semi-implícita, explorou-se a coerência espacial entre as amostras ao longo do raio e, para cada amostra, definiu-se a camada inicial K_0 da busca geométrica realizada na coluna (I, J) que contém o ponto como sendo a mesma em que a amostra anterior se encontrava.

5.4.1

Análise da performance

O desempenho de cada método foi medido ao rotacionar incrementalmente o modelo 180° em torno dos eixos X, Y e Z, sequencialmente. A Figura 5.6 apresenta, para cada modelo, a comparação de desempenho alcançada entre as diferentes estratégias analisadas. Em todos os casos, a solução proposta supera significativamente as duas outras estratégias em termo de performance, o que corrobora os resultados apresentados anteriormente na localização do conjunto de pontos gerados aleatoriamente.



Figura 5.6: Comparação de performance dos modelos entre a representação proposta e os métodos PLTS e LIBC.

5.4.2 Análise da qualidade

Na análise de qualidade, são comparadas as imagens obtidas pela visualização volumétrica. Para a quantificação do erro das imagens é adotada a métrica CIEDE2000, que é usada para mensurar a diferença de cor entre pixels, levando em conta que a percepção de diferenças de cor não é uniforme. A métrica considera ajustes de tonalidade, croma e luminosidade, proporcionando uma avaliação precisa da discrepância visual entre as imagens [20]. Para uma análise qualitativa do erro é utilizada a Tabela 5.6, que define intervalos

de referência para a classificação da percepção das diferenças de cor [27, 2]. A tabela também mostra as cores que foram associadas a cada intervalo na visualização dos erros.

Tabela 5.6: Tabela de classificação dos intervalos do Delta E de acordo com a percepção da diferença de cores, mostrando as cores adotadas para visualizá-las.

Delta E	Percepção	Cor
≤ 1	Não perceptível pelos olhos humanos	
1 - 2	Perceptível através de atenta observação	
2 - 10	Perceptível à primeira vista	
11 - 49	As cores são mais semelhantes do que opostas	
50 - 100	As cores são mais opostas do que semelhantes	

A Figura 5.7 apresenta o percentual de pixels em cada intervalo de classificação do Delta-E (Tabela 5.6), em cada quadro durante a rotação do modelo, adotando a imagem obtida com o método *PLTS* como referência no cálculo do erro. Observa-se que, no geral, o método proposto é mais preciso que o método *LIBC*. Apenas em alguns pontos de vista específicos, o contrário acontece. Os intervalos de erro mais críticos, em geral, concentram-se próximo à borda do reservatório, não comprometendo a acurácia.

A Figura 5.8 ilustra o ganho na acurácia da solução proposta quando comparada ao método *LIBC*, comparando a qualidade da imagem final obtida de diferentes pontos de vista. Conforme observado, em alguns pontos de vista específicos, a solução proposta apresenta resultados piores em poucos pixels próximos à borda dos blocos devido aos erros de classificação de pontos internos e externos. No entanto, a representação semi-implícita proporciona uma considerável melhoria na qualidade das imagens em toda a extensão do modelo.

5.5

Linhas de corrente

O traçado de linhas de corrente é um exemplo mais geral de aplicação da representação semi-implícita. Linhas de corrente são trajetórias que representam o caminho seguido por partículas em um campo de velocidades. Por definição, é uma curva cuja tangente é igual ao vetor velocidade do fluxo em qualquer instante [21]. Em reservatórios de petróleo, seu traçado é crucial para a análise dos padrões de fluxo, contribuindo para a identificação de zonas de

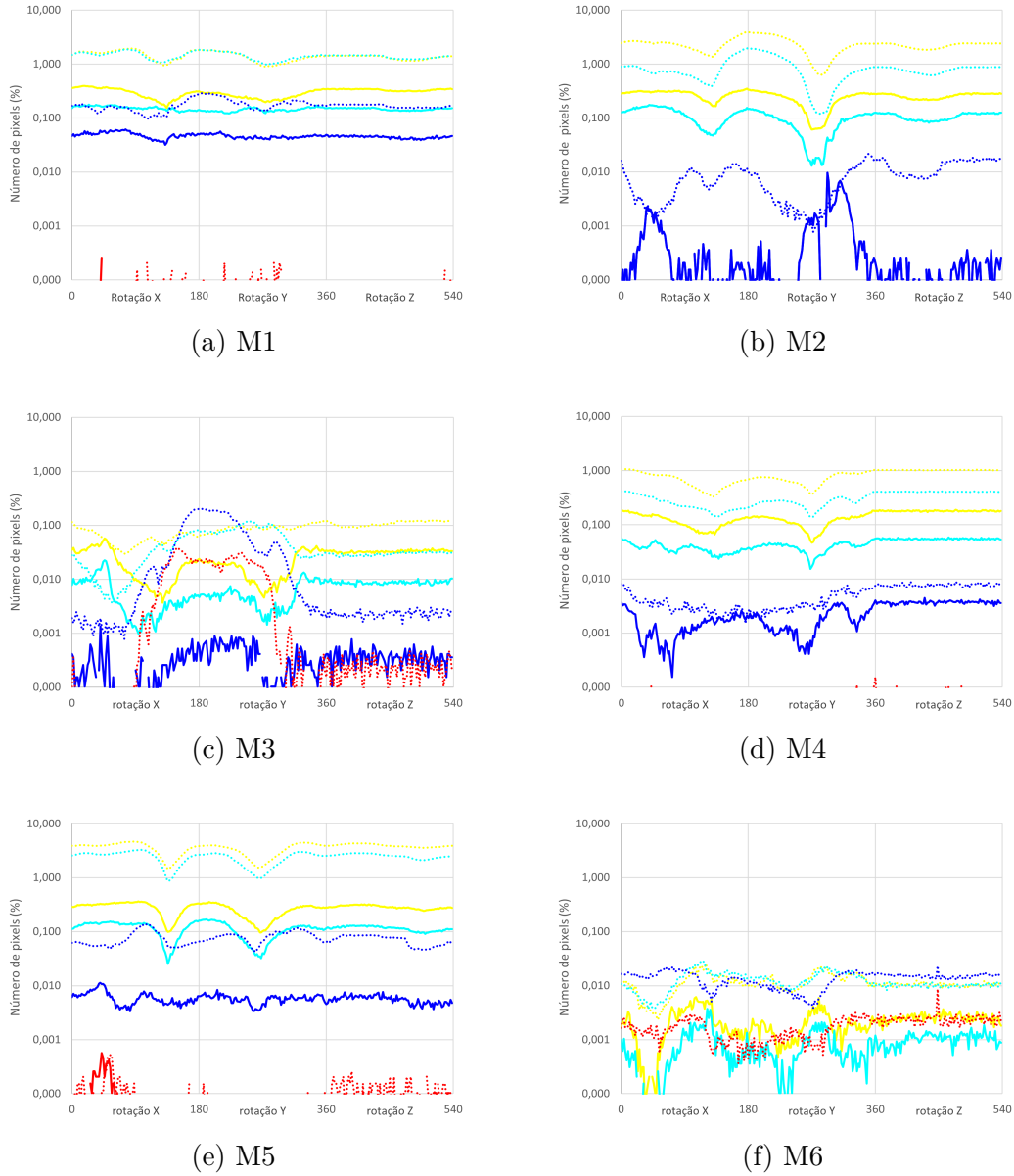


Figura 5.7: Percentual de pixels (relativo à imagem inteira) em cada intervalo de diferença de cor de acordo com a Tabela 5.6, durante a rotação do modelo. A qualidade da imagem obtida com a solução proposta é comparada com a imagem do método *LIBC*, adotando como referência a imagem gerada através do método *PLTS*. Linhas contínuas são associadas à solução proposta e linhas tracejadas ao método *LIBC*. Observe que os gráficos usam a escala logarítmica no eixo vertical para facilitar a visualização.

canalização e estagnação do fluxo, além de fornecer subsídios para estratégias de extração e otimização da produção.

O cálculo das linhas de corrente é realizado por meio da integração numérica das equações diferenciais que regem o movimento dos fluidos. Nesse processo, a determinação da posição das partículas em cada instante é fundamental, o que demanda uma eficiente técnica de localização de pontos para avaliar o campo de velocidades. Entre os métodos empregados para essa in-

tegração, destaca-se o método de Runge-Kutta de quarta ordem [6], o qual, a cada passo, realiza múltiplas avaliações do campo de velocidades e atualiza a posição da partícula por meio de uma combinação ponderada dessas avaliações, proporcionando elevada precisão. Esse método é amplamente utilizado devido ao bom equilíbrio entre precisão e custo computacional.

O algoritmo de localização de pontos proposto neste trabalho foi empregado no traçado das linhas de corrente e demonstrou ótimo desempenho. Os resultados obtidos evidenciam que o método é eficaz na determinação das trajetórias dos fluidos, mesmo em regiões de fluxo complexo, contribuindo para uma análise detalhada dos padrões de escoamento nos reservatórios de petróleo (Figura 5.9).

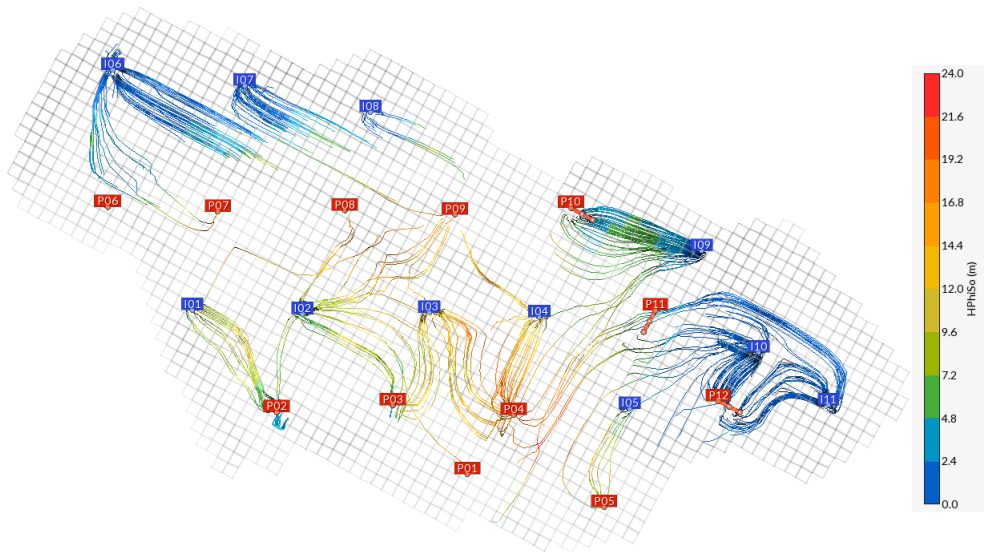
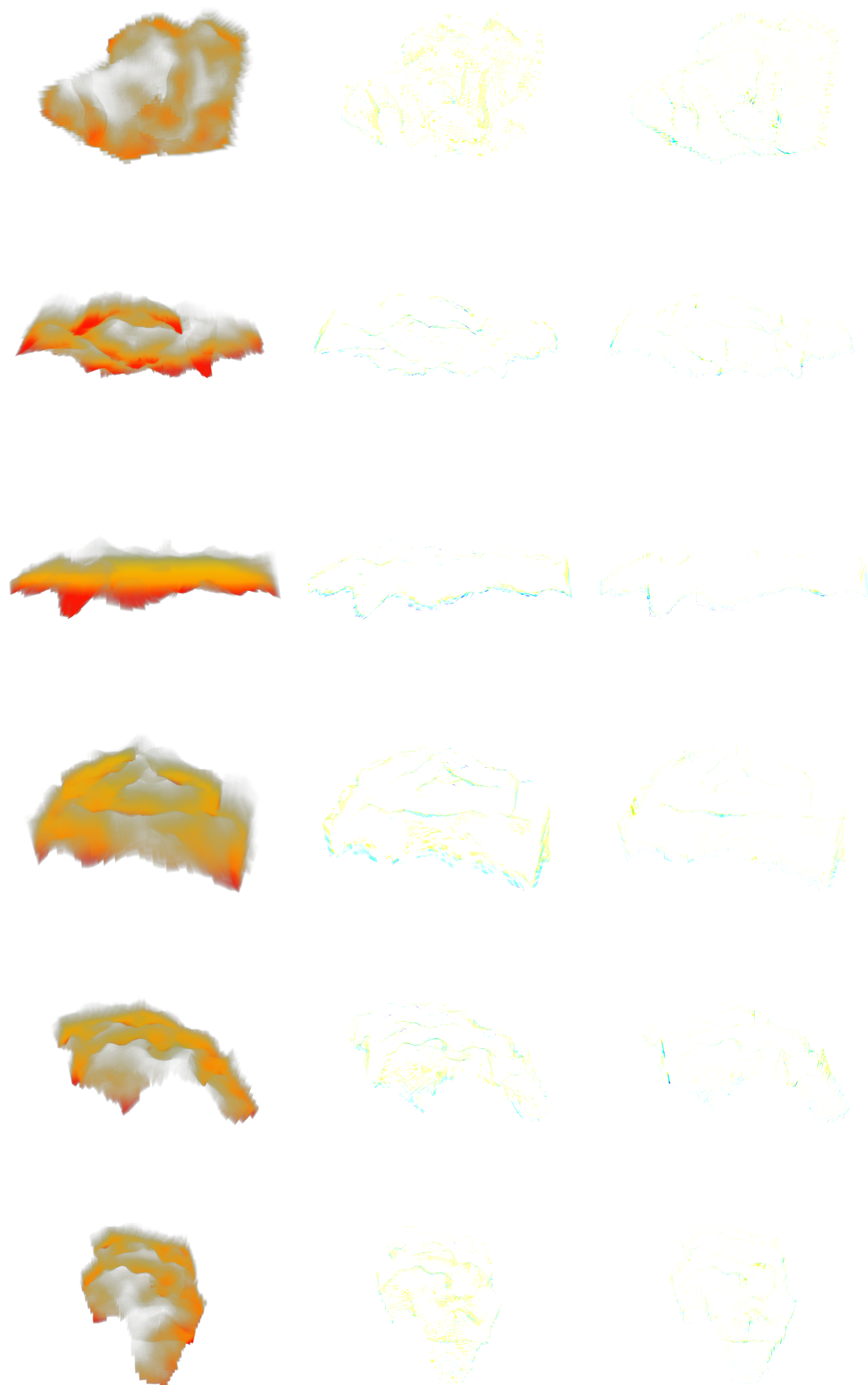


Figura 5.9: Linhas de corrente obtidas por meio do algoritmo de localização de pontos proposto.



(a) Imagens de referên- (b) Diferenças com o mé- (c) Diferenças com o mé-
cia obtidas com o método todo *LIBC* todo proposto
PLTS

Figura 5.8: Imagens representativas do algoritmo de visualização volumétrica na renderização da saturação de óleo do modelo M1. As imagens das diferenças ilustram o ganho de qualidade da nossa abordagem em comparação com a estratégia *LIBC*. As cores indicam a diferença em cada pixel, conforme a Tabela 5.6.

6

Conclusão

Foi apresentada uma nova representação para modelos discretos de reservatório, denominada representação semi-implícita, voltada para técnicas de visualização baseadas em GPU. Foi desenvolvido um eficiente algoritmo de localização de pontos, próprio para essa estrutura, o qual combina um campo de coordenadas topológicas implícitas, que utiliza a interpolação trilinear nativa da *GPU*, com um mapa de profundidade explícito. Descontinuidades causadas por falhas geológicas são tratadas por meio da decomposição do modelo em blocos, mantendo o desempenho mesmo em modelos altamente fraturados.

Comparações e testes demonstram as vantagens da representação semi-implícita proposta, considerando seis modelos reais de reservatórios. A abordagem alcança uma redução significativa no uso de memória — em média, 50% em relação à estrutura de incidências e à estrutura ResGPU [12] — sem comprometer a acurácia, além de obter um ganho médio de desempenho no algoritmo de localização de pontos de aproximadamente sete vezes em relação à estrutura de dados ResGPU. Experimentos de renderização volumétrica apresentaram taxas de atualização de quadros superiores e maior fidelidade da imagem gerada quando comparados a estratégias anteriores que adotam alguma aproximação do modelo.

Trabalhos futuros podem avaliar formas de melhorar o tempo de construção da solução proposta. Em uma outra abordagem, novas pesquisas podem explorar a extensão de representações implícitas neurais de superfícies discretas para volumes. Além disso, uma representação totalmente implícita pode trazer vantagens significativas; nesse contexto, investigações sobre a perda de expressividade de representações puramente implícitas na modelagem geológica mostram-se necessárias.

Referências bibliográficas

- [1] ABRAHAM, F.; CELES, W.. **Multiresolution visualization of massive black oil reservoir models.** The Visual Computer, 35(6):837–848, June 2019.
- [2] AUNG, S. S.; KYU, Z. M.. **Modified codebook algorithm with kalman filter for foreground segmentation in video sequences.** In: 2017 INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND COMMUNICATION (ICSPPC), p. 332–336, 2017.
- [3] BEYER, J.; HADWIGER, M. ; PFISTER, H.. **State-of-the-art in gpu-based large-scale volume visualization.** Comput. Graph. Forum, 34(8):13–37, Dec. 2015.
- [4] BOTSCH, M.; KOBELT, L.; PAULY, M.; ALLIEZ, P. ; LÉVY, B.. **Polygon mesh processing.** CRC press, 2010.
- [5] BUI, M.; ADJIMAN, C. S.; BARDOW, A.; ANTHONY, E. J.; BOSTON, A.; BROWN, S.; FENNELL, P. S.; FUSS, S.; GALINDO, A.; HACKETT, L. A. ; OTHERS. **Carbon capture and storage (ccs): the way forward.** Energy & Environmental Science, 11(5):1062–1176, 2018.
- [6] BUTCHER, J. C.. **Numerical methods for ordinary differential equations.** John Wiley & Sons, 2016.
- [7] CAMPAGNOLO, L. Q.; CELES, W.. **An experimental study on volumetric visualization of black oil reservoir models.** Computers & Graphics, 93:84–94, 2020.
- [8] CEBALLOS, LOÏC; CONCHE, B. D. G. P. D.. **Visualization of large scale reservoir models.** In: Patel, D., editor, INTERACTIVE DATA PROCESSING AND 3D VISUALIZATION OF THE SOLID EARTH, p. 209–232. Springer International Publishing, 2021.
- [9] CHEN, Z.; ZHANG, H.. **Learning implicit fields for generative shape modeling.** In: PROCEEDINGS OF THE IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), June 2019.

- [10] CHIBANE, J.; ALLDIECK, T. ; PONS-MOLL, G.. **Implicit functions in feature space for 3d shape reconstruction and completion.** CoRR, abs/2003.01456, 2020.
- [11] CHIBANE, J.; MIR, M. A. ; PONS-MOLL, G.. **Neural unsigned distance fields for implicit function learning.** In: Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. ; Lin, H., editors, **ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS**, volumen 33, p. 21638–21652. Curran Associates, Inc., 2020.
- [12] FRANCESCHIN, B.; ABRAHAM, F.; NETTO, L. F. ; CELES, W.. **Gpu-based rendering of arbitrarily complex cutting surfaces for black oil reservoir models.** In: 2019 32ND SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI), p. 131–138, 2019.
- [13] GENOVA, K.; COLE, F.; SUD, A.; SARNA, A. ; FUNKHOUSER, T.. **Local deep implicit functions for 3d shape.** In: **PROCEEDINGS OF THE IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)**, June 2020.
- [14] GOLDMAN, R.. **Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling.** Elsevier, 2002.
- [15] GOLUB, G. H.; VAN LOAN, C. F.. **Matrix computations.** JHU press, 2013.
- [16] GRACIANO, A.; RUEDA, A. J.; POSPÍŠIL, A.; BITTNER, J. ; BENES, B.. **Quadstack: An efficient representation and direct rendering of layered datasets.** *IEEE Transactions on Visualization and Computer Graphics*, 27(9):3733–3744, 2021.
- [17] JOLLIFFE, I. T.. **Principal Component Analysis.** Springer, 2nd edition, 2002.
- [18] JOHNSON, D. S.; DEMERS, A.; ULLMAN, J. D.; GAREY, M. R. ; GRAHAM, R. L.. **Worst-case performance bounds for simple one-dimensional packing algorithms.** *SIAM Journal on Computing*, 3(4):299–325, 1974.
- [19] JONES, M.; BAERENTZEN, J. ; SRAMEK, M.. **3d distance fields: a survey of techniques and applications.** *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, 2006.

- [20] LUO, M. R.; CUI, G. ; RIGG, B.. **The development of the CIE 2000 colour-difference formula: CIEDE2000**. Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur, 26(5):340–350, 2001.
- [21] MACHADO, M. V. B.. **Modelagem numérica de reservatórios de petróleo: Prática integrada de simulação**. Petrobras: Rio de Janeiro, Brazil, 2023.
- [22] MCREYNOLDS, T.; BLYTHE, D.. **Advanced graphics programming using OpenGL**. Elsevier, 2005.
- [23] NETTO, L. F.; CELES, W.. **A new focus+context visualization technique for inspecting black oil reservoir models**. In: 2021 34TH SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI), p. 25–32, 2021.
- [24] PEYROT, J.-L.; DUVAL, L.; PAYAN, F.; BOUARD, L.; CHIZAT, L.; SCHNEIDER, S. ; ANTONINI, M.. **Hexashrink, an exact scalable framework for hexahedral meshes with attributes and discontinuities: multiresolution rendering and storage of geoscience models**. Computational Geosciences, 23(4):723–743, aug 2019.
- [25] RØE, P.; HAUGE, R.. **A volume-conserving representation of cell faces in corner point grids**. Computational Geosciences, 20(3):453–460, jun 2016.
- [26] SARTON, J.; ZELLMANN, S.; DEMIRCI, S.; GÜDÜKBAY, U.; ALEXANDRE-BARFF, W.; LUCAS, L.; DISCHLER, J.-M.; WESNER, S. ; WALD, I.. **State-of-the-art in Large-Scale Volume Visualization Beyond Structured Data**. Computer Graphics Forum, 42(3):491–515, June 2023.
- [27] SCHUESSLER, Z.. **Delta e: Learn**. URL: <http://zschuessler.github.io/DeltaE/learn/>. Acessado em 18/08/2024.
- [28] WANG, L.; YU, Y.; ZHOU, K. ; GUO, B.. **Multiscale vector volumes**. ACM Trans. Graph., 30(6):1–8, Dec. 2011.