Luiz Felipe Netto

Algoritmo de corte com preservação de contexto para visualização de modelos de reservatório

Dissertação de Mestrado

Departamento de Informática
Programa de Pós–graduação em Informática

Rio de Janeiro
Setembro de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Luiz Felipe Netto

Algoritmo de corte com preservação de contexto para visualização de modelos de reservatório

Dissertação de Mestrado

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

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro
Setembro de 2016

Pontifícia Universidade Católica
do Rio de Janeiro

Luiz Felipe Netto

Algoritmo de corte com preservação de contexto para visualização de modelos de reservatório

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática — PUC-Rio

Prof. Ricardo Guerra Marroquim
UFRJ

Prof. Marcelo Gattass
Departamento de Informática — PUC-Rio

Prof. Helio Côrtes Vieira Lopes
Departamento de Informática — PUC-Rio

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 16 de setembro de 2016

Luiz Felipe Netto

Graduou-se em Ciência da Computação pela Universidade Federal de Santa Maria (UFSM), onde trabalhou em projetos relacionados a Computação Gráfica no Laboratório de Computação Aplicada (LaCA). Em 2014 ingressou no programa de mestrado em Informática na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Durante o mestrado, trabalhou com pesquisa e desenvolvimento junto ao grupo de visualização do Instituto Tecgraf.

Ficha Catalográfica

Netto, Luiz Felipe

Algoritmo de corte com preservação de contexto para visualização de modelos de reservatório / Luiz Felipe Netto; orientador: Waldemar Celes Filho. — Rio de Janeiro : PUC-Rio, Departamento de Informática, 2016.

71 f: il. (color.); 29,7 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2016.

Inclui bibliografia.

1. Informática – Teses. 2. Algoritmo de corte. 3. Foco-contexto. 4. Renderização em tempo real. I. Celes Filho, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Primeiramente, agradeço a minha família por estarem sempre ao meu lado, me incentivando a seguir nesse caminho. Aos meus pais, Patrícia e Pai Neno, e minha maninha Lara, por todo amor, carinho e fé depositada em mim. Aos meus tios avós, Olinda e Clóvis, por sempre me ajudarem nos momentos difíceis com aquela ligação ou recado inesperado. Ao tio Backer, meu irmão mais velho, que acha o título de Mestre o melhor de todos.

Aos meus colegas desde a faculdade, Schirmer e Schardong, pela ajuda para me estabelecer no Rio.

Aos colegas do Tecgraf, em especial ao Bernardo, Fred, Christiano e Espinha, que sempre estiverem disponíveis para sanar quaisquer dúvidas e foram fundamentais durante essa pesquisa.

Ao meu orientador, Waldemar, por compartilhar de seu conhecimento, pela oportunidade e incentivo no desenvolvimento deste trabalho.

Ao Instituto Tecgraf e a PUC-RIO, pelo apoio financeiro, que viabilizaram a realização deste trabalho.

Resumo

Netto, Luiz Felipe; Celes Filho, Waldemar. **Algoritmo de corte com preservação de contexto para visualização de modelos de reservatório**. Rio de Janeiro, 2016. 71p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A simulação numérica de reservatório de petróleo é um processo amplamente utilizado na indústria de óleo e gás. O reservatório é representado por um modelo de células hexaédricas com propriedades associadas, e a simulação numérica procura prever o fluxo de fluído dentro do modelo. Especialistas analisam os resultados dessas simulações através da inspeção, num ambiente gráfico interativo, do modelo tridimensional. Neste trabalho, propõe-se um novo algoritmo de corte com preservação de contexto para auxiliar a inspeção do modelo. O principal objetivo é permitir que o especialista visualize o entorno de poços. Os poços representam o objeto de interesse que deve estar visível e o modelo tridimensional (o contexto) é preservado na medida do possível no entorno desses poços. Desta forma, torna-se possível avaliar a variação de propriedades associadas às células na vizinhança do objeto de interesse. O algoritmo proposto explora programação em placa gráfica e é válido para objetos de interesse arbitrários. Propõe-se também uma extensão do algoritmo para que a seção de corte seja desacoplada da câmera, permitindo analisar o modelo cortado de outros pontos de vista. A eficácia do algoritmo proposto é demonstrada através de resultados baseados em modelos reais de reservatório.

Palavras–chave

Algoritmo de corte; Foco-contexto; Renderização em tempo real; Visualização de reservatórios.

Abstract

Netto, Luiz Felipe; Celes Filho, Waldemar (Advisor). **Cutaway algorithm with context preservation for reservoir model visualization**. Rio de Janeiro, 2016. 71p. MSc. Dissertation — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Numerical simulation of black oil reservoir is widely used in the oil and gas industry. The reservoir is represented by a model of hexahedral cells with associated properties, and the numerical simulation is used to predict the fluid behavior in the model. Specialists make analysis of such simulations by inspecting, in a graphical interactive environment, the tridimensional model. In this work, we propose a new cutaway algorithm with context preservation to help the inspection of the model. The main goal is to allow the specialist to visualize the wells and their vicinity. The wells represent the object of interest that must be visible while preserving the tridimensional model (the context) in the vicinity as far as possible. In this way, it is possible to visualize the distribution of cell property together with the object of interest. The proposed algorithm makes use of graphics processing units and is valid for arbitrary objects of interest. It is also proposed an extension to the algorithm to allow the cut section to be decoupled from the camera, allowing analysis of the cut model from different points of view. The effectiveness of the proposed algorithm is demonstrated by a set of results based on actual reservoir models.

Keywords

Cutaway algorithm; Focus-context; Real time rendering; Reservoir visualization.

Sumário

Lista de figuras

Lista de tabelas

It is our responsibility as scientists, knowing the great progress and great value of a satisfactory philosophy of ignorance, the great progress that is the fruit of freedom of thought, to proclaim the value of this freedom, to teach how doubt is not to be feared but welcomed and discussed, and to demand this freedom as our duty to all coming generations.

Richard Phillips Feynman, *The Pleasure of Finding Things Out.*

1
Introdução

Ao longo do campo de reservatórios de petróleo encontram-se, principalmente, fluídos de água, óleo e gás. A extração de óleo e gás ocorre, primeiramente, com a perfuração de poços produtores, através da pressão natural que expele o fluído pelo poço. Em um segundo momento, quando a pressão natural não é suficiente, métodos secundários são utilizados na extração do óleo. Poços injetores são inseridos para efetuar o controle do fluxo dos fluídos em direção a poços produtores, através de métodos como injeção de água e gás. A colocação de poços ao longo de reservatórios de petróleo deve ocorrer de forma planejada, buscando a maximização da produção e do tempo de atividade do mesmo. Esse planejamento ocorre, extensivamente, com base na simulação numérica de modelos de reservatório de petróleo, sendo uma etapa relevante para industria de óleo e gás.

Os modelos de reservatório de petróleo são representados por células hexaédricas, com propriedades associadas através de simulações numéricas, que buscam prever o fluxo do fluído em seu interior. Ambientes interativos são utilizados na análise desses modelos, provendo ferramentas como gráficos, visualizadores bidimensionais e tridimensionais, auxiliando especialistas na compreensão do reservatório e suas propriedades (Figura 1.1). Nesse contexto, técnicas de visualização são desenvolvidas, fornecendo novos pontos de vista sobre as propriedades do reservatório.

Neste trabalho, propõe-se um algoritmo de corte baseado em visualização de Foco+Contexto, com o objetivo de auxiliar a inspeção do entorno de poços do reservatório de petróleo. Esse corte é realizado com base em um conjunto de poços selecionados pelo usuário, representando o foco, sempre visível, e removendo parte do contexto, as células do modelo. Para isso, gera-se um mapa de profundidade, a partir do foco, representando a superfície de corte. Ao desenhar o modelo tridimensional, o contexto, os fragmentos que estão entre essa superfície e o observador são descartados. Para preencher os espaços revelados nesse processo, desenha-se a superfície de corte. Esse desenho é baseado em um método que testa, para cada uma das posições do mapa, se ocorre interseção com o modelo.

Inicialmente, o algoritmo de corte é orientado à posição do observador,

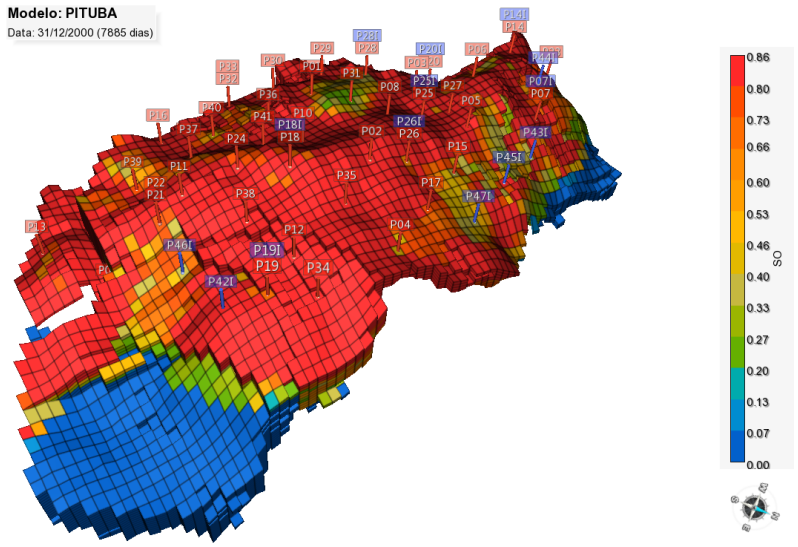
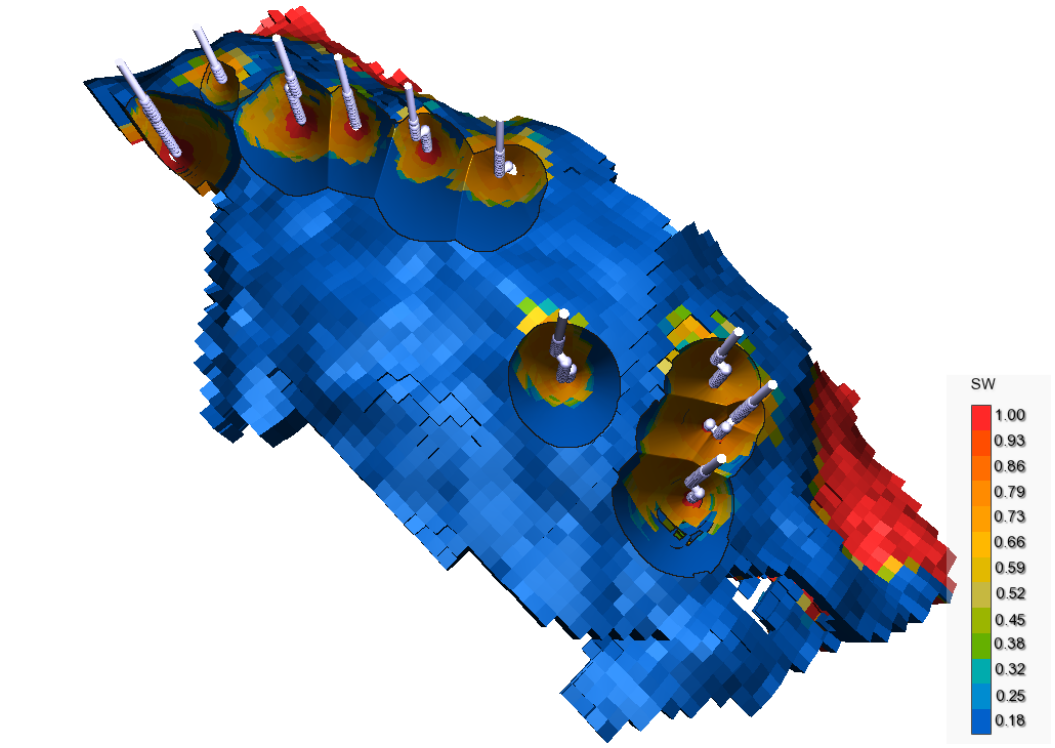


Figura 1.1 – Visualização de um modelo de reservatório de petróleo com poços injetores e produtores em um ambiente interativo. A propriedade saturação de óleo (SO) está associada as células do modelo.

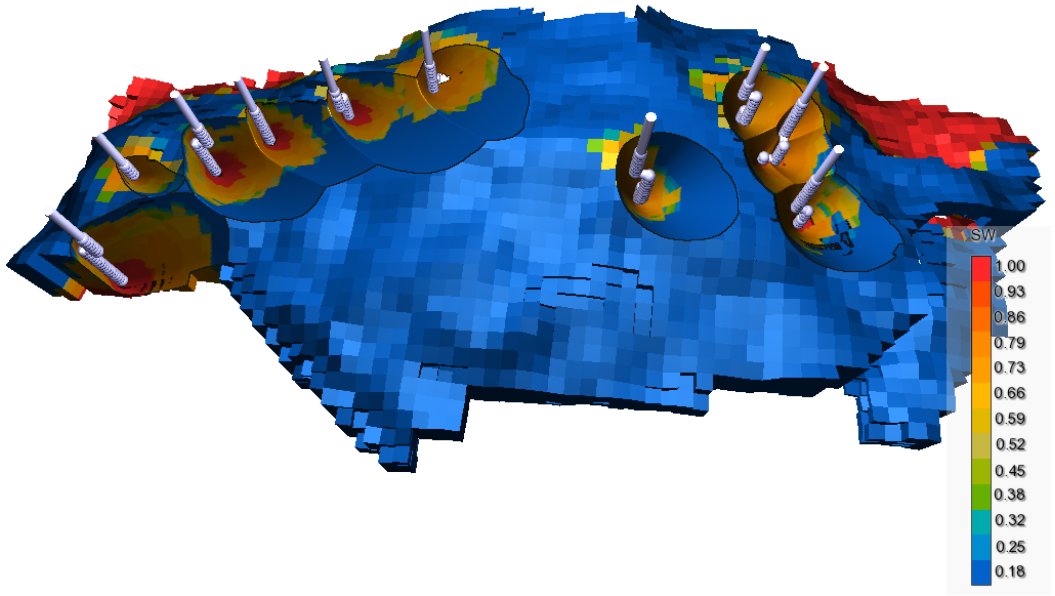
com controle de abertura e do desenho do aramado das células do modelo. Propõe-se uma extensão desse algoritmo para prover o ganho de paralaxe de movimento, no qual o corte é desacoplado da posição do observador, podendo ser visualizado a partir de outros pontos de vista.

Os resultados comprovam a eficácia dos algoritmos propostos que extensivamente usam programação em placa gráfica. A partir das imagens geradas, é possível observar as propriedades associadas às células no entorno dos objetos de interesse e a variação de suas intensidades em diferentes passos de tempo. O desempenho do algoritmo ocorre em taxas de quadro por segundo interativas desde modelos de pequeno porte até modelos massivos. Quando comparado com outros cortes de Foco+Contexto aplicados a reservatório de petróleo, o algoritmo diferencia-se por minimizar a quantidade de contexto removida e por não requerer o desenho de todas células do modelo, mas somente as faces externas. A Figura 1.2 apresenta o resultado obtido na visualização de um modelo de reservatório de petróleo com um conjunto de poços injetores definidos como objetos de interesse.

Esta dissertação está organizada da seguinte forma: no Capítulo 2 discute-se os trabalhos relacionados aos algoritmos propostos. No Capítulo 3 é apresentado o primeiro algoritmo proposto e, no Capítulo 4, um segundo algoritmo é proposto como extensão para esse. O Capítulo 5 apresenta os resultados obtidos, onde são discutidas a qualidade, o desempenho e duas possíveis aplicações para os algoritmos propostos. No Capítulo 6, concluí-se o documento, apresentando as considerações finais e possíveis trabalhos futuros.



(a)



(b)

Figura 1.2 – Modelo de reservatório de petróleo com a propriedade saturação de água (SW) associada as células. (a) Corte orientado ao observador com poços injetores definidos como objetos de interesse e (b) visualização do mesmo corte a partir de outro ponto de vista.

2
Trabalhos relacionados

Neste capítulo, são apresentados trabalhos relacionados à visualização de Foco+Contexto. Destacam-se cortes do tipo *cutaway*, que podem ser classificados como estáticos e interativos. Em seguida, são apresentados trabalhos com aplicação em modelos de reservatório de petróleo.

2.1
Visualização de Foco+Contexto

A visualização de Foco+Contexto inclui uma ampla gama de técnicas para visualizar modelos volumétricos e poligonais de diversas áreas, como Arquitetura, Biologia e Engenharia, em que o objetivo é destacar o foco sem remover totalmente o contexto. Essas técnicas, das quais destacamos lentes mágicas, vistas explodidas e *cutaway*, são tradicionalmente utilizadas em livros técnicos, científicos e educacionais, elaboradas por ilustradores com domínio no assunto, que decidem cuidadosamente como cortar ou mover partes desses modelos para que as estruturas de interesse sejam destacadas [4]. Essas técnicas são naturalmente aplicáveis na área de visualização computadorizada, onde diversos trabalhos propõem soluções dinâmicas e interativas para sua obtenção.

Lentes mágicas [5–9] permitem que o usuário explore os modelos através de uma lupa que serve como uma ferramenta para a aplicação de vários filtros, como *zoom* e transparência (*ghosting*), dando ênfase ou revelando detalhes do modelo. *Vistas explodidas* [10–12] auxiliam na compreensão de modelos complexos através da deformação e separação entre seus objetos, com ênfase no objeto em foco, mantendo uma relação espacial coerente. O trabalho de Martins Filho et al. [13] aplica vistas explodidas em modelos de reservatório de petróleo representados por *corner-points*. Eles desenvolveram um sistema que faz uso de uma estrutura de dados chamada árvore de explosão para relacionar polígonos convexos disjuntos, evitando a sobreposição de células.

2.2
Cortes do tipo *cutaway*

Cortes do tipo *cutaway* removem porções oclusoras do modelo para revelar o foco, ao passo que preservam informações do contexto. Para isto,

divide-se os objetos do modelo em dois conjuntos: objetos de interesse, o foco, e objetos secundários, o contexto. Os trabalhos podem ser categorizados em ambientes para visualização interativa e ambientes para gerar ilustrações estáticas.

Feiner e Seligmann [14] introduziram algoritmos para identificar objetos de interesse oclusos em uma cena e visualizá-los através de cortes e *ghosting*, fazendo uso de uma abordagem baseada em *z-buffer* e outras duas baseadas no algoritmo de volume de sombra [15]. Diepstraten et al. [16] desenvolveram métodos para renderização interativa de cortes, que são classificados em *break-away*, um corte estreito que remove somente o que está à frente do objeto de interesse, e *cutout*, um corte amplo que remove uma quantidade maior do contexto, revelando o objeto de interesse e seus arredores. Coffin e Höllerer [17] permitem a realização de um corte interativo em geometrias oclusoras com o uso de operações de *constructive solid geometry* (CSG) através de um *stencil-buffer*.

Bruckner e Gröler [18] produziram um sistema interativo para gerar ilustrações estáticas, semelhantes às encontradas em livros, através de renderização volumétrica direta. Li et al. [19] desenvolveram um sistema que automatiza a geração dos cortes, bastando dizer qual o formato do corte, cilíndrico ou retangular, e o objeto de interesse, e permite que o usuário faça ajustes finos na visualização. O sistema produz ilustrações com legendas para identificar os objetos revelados pelo corte e aplica filtros para destacar os contornos dos objetos.

2.2.1 Superfície de corte representada por uma textura de profundidade

Uma abordagem para a obtenção desses cortes é utilizar uma superfície de corte representada por um mapa de profundidade, o qual é gerado a partir da profundidade dos objetos de interesse. Objetos secundários, que estão entre a superfície e o observador, são removidos para a obtenção do corte. Sistemas interativos de visualização volumétrica [20, 21] introduziram este conceito, que logo foi adaptado para malhas poligonais [22], com aplicações em modelos geológicos [23] e modelos de reservatório de petróleo [24]. Esta abordagem também foi aplicada na construção de um sistema que permite a compreensão de modelos CAD através de perfurações no modelo, revelando estruturas internas, e suporta múltiplas vistas para dar orientação [25].

Burns e Finkelstein [22] produziram cortes adaptativos com o formato dos objetos de interesse, com controle da abertura do corte e que são dependentes do observador. A superfície de corte é gerada com uma transformada de

distância aplicada aos objetos de interesse através do algoritmo de *jump-flooding* [1]. Este algoritmo representa um paradigma que explora o paralelismo da placa gráfica para transferir informações associadas com uma posição de uma grade bidimensional para todas outras em $\log n$ passos, onde n é a maior dimensão desta. Cada posição sendo processada avalia as informações contidas em seus 8 vizinhos a uma distância de l igual a maior potência de 2 menor que sua maior dimensão, para o primeiro passo, $l = l/2$, para o segundo passo, $l = l/4$, para o terceiro passo, e assim por diante, até que $l = 1$. A cada passo, se algum vizinho contiver alguma informação de interesse, esta é copiada (Figura 2.1).

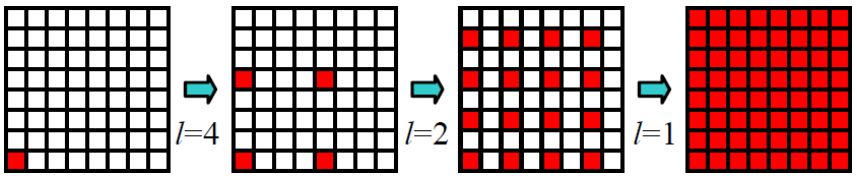


Figura 2.1 – Propagação de informação com *jump-flooding* em uma grade 8×8, tal que $n = 8$. Ilustração retirada de [1].

O processo descrito acima possibilita a obtenção da superfície de corte com uma taxa de quadros por segundo (*frames per second*, FPS) interativa, mais eficiente que o método tradicional para determinar transformadas de distância aproximada em CPU [26]. A superfície de corte é definida, por Burns e Finkelstein [22], com base na grade de *pixels* da janela, realizando as seguintes observações:

– A transformada de distância de uma imagem B é determinada através de uma função que calcula, para cada *pixel* p, a distância d a todas sementes $q \in B$, guardando a menor:

$$d(p) = \min_{q \in B} \|q - p\| \tag{2.1}$$

– Sabendo a maior distância d_{max} entre dois pontos, determinada pelos extremos opostos na diagonal da grade de *pixels*, torna possível definir a seguinte função:

$$d(p) = \max_{q \in B} (d_{max} - \|q - p\|) \tag{2.2}$$

– Tendo como entrada uma textura I com a profundidade das *back-faces* dos objetos de interesse em cada pixel $z(p)$ e um fator de escala m, define-se:

$$d(p) = \max_{q \in I} (z(p) - m\,\|q - p\|) \tag{2.3}$$

– A Equação (2.3) representa uma superfície de inclinação $m = \tan(\theta)$ com o plano da imagem, sobre cada *pixel* p ao longo do eixo-z até a profundidade $z(p)$.

Para aplicação do corte, a cena com os objetos secundários é desenhada e fragmentos que estão entre o observador e a superfície de corte são descartados. As paredes dos objetos cortados são desenhadas utilizando a cor das *back-faces*, reveladas pela operação de descarte, e a normal é derivada da superfície de corte. Por fim, são desenhados os objetos de interesse para completar a visualização.

Observa-se que em uma cena com projeção ortográfica, o ângulo θ entre a superfície de corte e o vetor direção do observador é determinado por m. Em uma cena com projeção perspectiva isso é válido apenas no espaço da janela. Ainda considerando esta projeção, eles observaram que distorções ocorrem no corte quando o observador se aproxima ou se afasta dos objetos de interesse. Isso ocorre devido às variações dos valores da profundidade no espaço da janela. Considerando o fator de escala da matriz de projeção $PM_{sz} = ((z_{near} + z_{far})/(z_{near} - z_{far}))$, foi proposta a seguinte correção para substituir m na Equação (2.3):

$$m(p) = \frac{-(PM_{sz} + z(p))}{\tan(\theta)} \tag{2.4}$$

2.2.2 Cortes em modelos de reservatório de petróleo

Usualmente, modelos de reservatório de petróleo são representados por malhas discretas de células hexaédricas semi-estruturadas. A malha ser semi-estruturada indica que a relação entre células vizinhas é definida por ponteiros, através de uma estrutura de dados topológicas. A malha representa uma grade tridimensional de hexaedros irregulares, onde as células são identificadas por índices $[i, j, k]$, podem ser ativas ou inativas, e possuem propriedades associadas que são mapeadas durante a visualização.

A renderização da malha do reservatório pode ser obtida a partir da triangulação de suas faces. Cada célula é representada por 8 nós (Figura 2.2), compondo seis faces, que podem ser classificadas em faces internas e externas. **Faces externas** são aquelas que fazem parte da borda externa do modelo, incluindo vizinhas de células inativas e vizinhas a falhas geológicas que podem estar no interior do modelo. Uma maneira de otimizar drasticamente a visualização é desenhar a malha somente com as faces externas, já que as faces internas não são visíveis se o observador estiver localizado fora do modelo.

Células inativas não são de interesse para simulação e geralmente não são renderizadas.

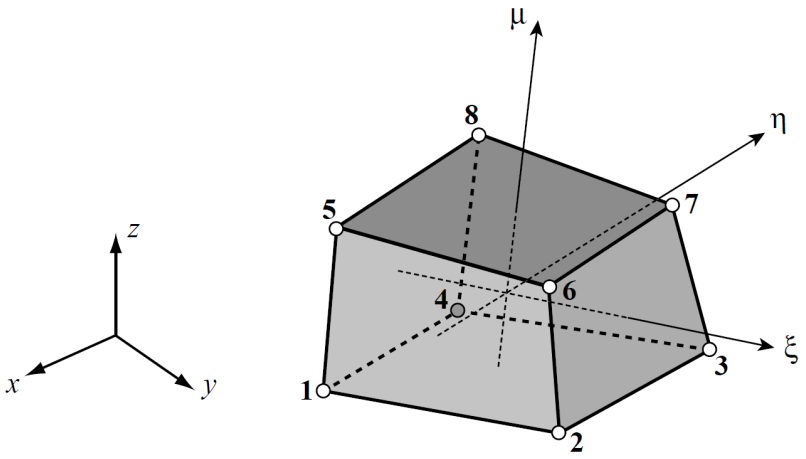


Figura 2.2 – Hexaedro de 8 nós. Ilustração retirada de [2].

Até o presente momento, temos conhecimento de apenas dois trabalhos aplicando cortes em modelos de reservatório de petróleo. No trabalho de Martins Filho et al. [27] foi desenvolvida uma ferramenta que permite a aplicação de cortes, com o uso de filtros para definir as células de interesse, combinado com o efeito de transparência nas células secundárias. Um gráfico XY foi utilizado para aplicação de dois filtros, células com propriedades dentro dos limites dos filtros são categorizadas como Células Selecionadas e as demais como Células Não-Selecionadas. Para aplicação do corte, um raio é traçado para cada um dos centroides das Células Selecionadas. Se a distância entre o raio e cada Célula Não-Selecionada for menor que uma determinada tolerância, esta é completamente descartada.

Carvalho et al. [24] propõem um método para aplicação de cortes utilizando os princípios definidos em [23], onde o corte gerado possui formato de caixa com base na profundidade das caixas envolventes (*bounding box*) das células de interesse. Para auxiliar a análise do corte, o método provê um modo chamado de vista congelada (*freeze view*), que permite rotacionar o modelo sem que o corte seja alterado, com ganho de percepção de profundidade do efeito de paralaxe. O corte é realizado em três estágios: a criação da superfície de corte, o desenho das células secundárias com aplicação do corte e, por fim, o desenho das células de interesse.

Nesta dissertação, é utilizada a estrutura de dados topológicas TopS [28, 29] para representar a malha do modelo. Essa estrutura é compacta, somente os vértices e as células da malha são representados explicitamente, ao passo que faces e arestas são representadas implicitamente. A estrutura também é completa, por permitir a obtenção de todas as relações topológicas

de adjacência. Por exemplo, dado um vértices, é possível obter os vértices adjacentes ou determinar se uma face é compartilhada entre duas células adjacentes. Posteriormente, essa funcionalidade é explorada para realizar a extração das faces externas do modelo.

2.3 Mapeamento de campos escalares de malhas não estruturadas

Franceschin [3] propôs uma técnica para realizar mapeamento de interseções de seções de cortes arbitrárias em malhas não estruturadas fazendo uso de programação em placa gráfica. Superfícies arbitrárias são renderizadas e os fragmentos gerados pelo processo de rasterização tem suas posições testadas com as células do modelo de reservatório, sendo desenhados com a cor do campo escalar associado à célula que ocorrer interseção. Para garantir a eficiência do calculo de interseção foi utilizada uma grade regular como estrutura de aceleração. Cada *voxel* da grade regular possui uma lista de células do modelo, que estão associadas a uma lista de vértices e de propriedades. A grade regular e as listas são computadas em uma etapa de pré-processamento e então enviadas para placa gráfica para que possam ser acessadas em tempo de renderização.

O teste de interseção na determinação da localização de um ponto arbitrário (x, y, z) no modelo de reservatório, ou seja, a determinação de qual célula contém um fragmento, prossegue com as seguintes etapas:

1. Obter a posição do fragmento no espaço do objeto
2. Acessar o *voxel* correspondente na grade regular
3. Carregar o conjunto de vértices para cada célula no *voxel*
4. Verificar se o fragmento está dentro da célula definida pelos vértices carregados através de um método numérico
5. Acessar o valor da propriedade correspondente se ocorrer interseção

No método numérico, converte-se posições do espaço físico, com coordenadas (x, y, z), para o espaço paramétrico, com coordenadas naturais (ξ, μ, η). Este espaço é regular e as coordenadas naturais assumem valores em $[-1, 1]$, onde funções de forma (*shape functions*) são utilizadas para realizar o mapeamento para o espaço físico (Figura 2.3).

Para determinar se uma posição física corresponde a um ponto dentro da célula, basta verificar se os valores obtidos para (ξ, μ, η) estão dentro do intervalo $[-1, 1]$. Visto que a coordenada natural da posição do fragmento foi

encontrada, basta mapear as propriedades do campo escalar. Este método é executado no *fragment shader* para cada célula da lista contida no *voxel* da grade regular que ocorrer interseção.

Alguns resultados obtidos por [3] estão ilustrados na Figura 2.4. É possível observar o mapeamento da interseção em polígonos arbitrários, as cores das propriedades e o aramado (*wireframe*) correspondente são renderizados corretamente. Esta técnica apresenta bom desempenho, tendo como principal limitação a necessidade de armazenar todos vértices do modelo na memória da placa gráfica.

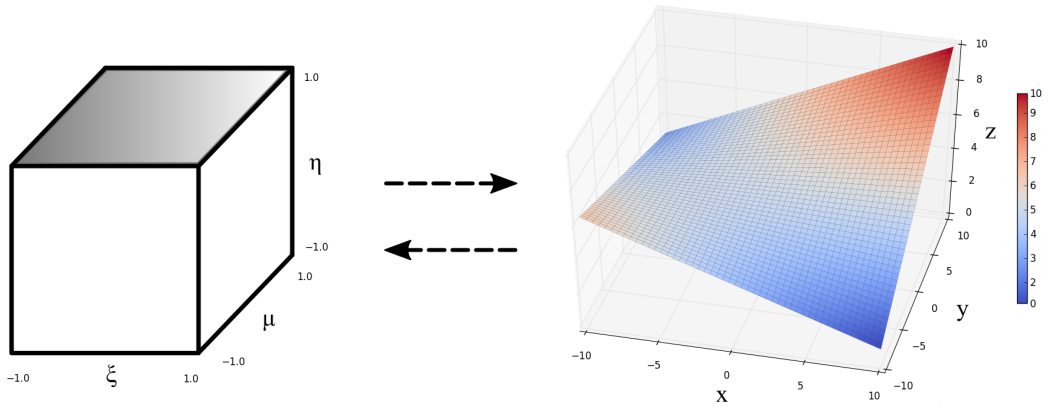


Figura 2.3 – Mapeamento da face topo (em cinza) entre o espaço paramétrico (esquerda) e o espaço físico (direita). Neste, a superfície da face é curva e as cores representam a intensidade em z.

2.4 Mapeamento de texturas com elevação

O algoritmo de mapeamento de relevo (*relief mapping*) [30] pode ser utilizado para renderizar polígonos arbitrários com aspectos tridimensionais, dado um mapa de profundidade associados a cada um desses polígonos. Esse algoritmo é relevante por apresentar uma possível abordagem na reconstrução da superfície de corte representada por um mapa de profundidade.

O mapeamento é realizado através de um programa de fragmento. O mapa de relevo pode ser representado pelas componentes RGBα com precisão de 32 bits, onde RGB representam as componentes do mapa de normal e α representa o mapa de profundidade. Dois passos são executados no espaço da textura do mapa de relevo: uma busca linear na direção do vetor do observador para encontrar o primeiro ponto abaixo do relevo, seguida de uma busca binária entre a posição encontrada e a posição anterior para aproximar com precisão o ponto de interseção.

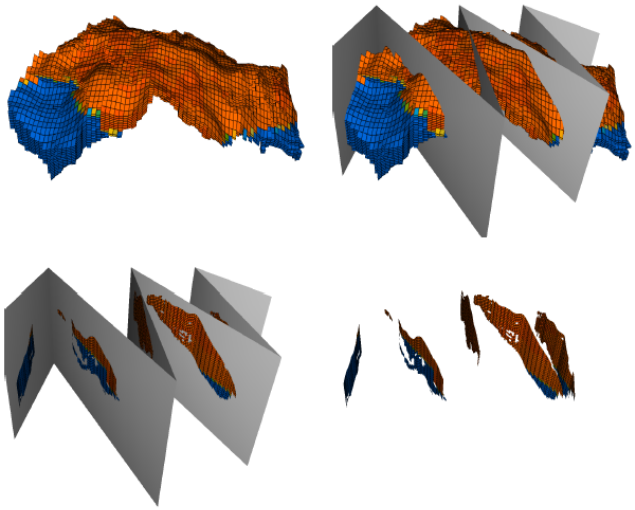


Figura 2.4 – Mapeamento do campo escalar em uma superfície arbitrária. Ilustração retirada de [3].

O tamanho do passo na busca linear determina o nível de detalhe que o mapeamento de relevo apresenta, o que resulta em artefatos na visualização da superfície mapeada quando o tamanho do passo é maior que algumas estruturas do relevo. Para contornar isso, Policarpo e Oliveira [31] modificaram o algoritmo para percorrer um mapa de cones ao invés de realizar a busca linear. O mapa de cones é obtido em pré-processamento, sendo representado por uma componente adicional no mapa de profundidade, que define, para cada posição, a razão da largura pela altura de um cone circular que intercepta o relevo apenas uma vez.

2.5 Proposta

Os algoritmos propostos nos Capítulos 3 e 4, diferenciam-se dos demais trabalhos aplicados a modelos de reservatório de petróleo, por produzirem cortes com base na profundidade dos objetos de interesse. A superfície de corte gerada minimiza a quantidade de contexto a ser removida. O desenho do modelo é realizado somente com suas faces externas, explorando-se o método proposto por Franceschin [3] para realizar o mapeamento do campo escalar no desenho da superfície de corte. Para auxiliar a percepção de profundidade é possível desacoplar o corte do observador, de maneira que este não seja mais gerado a cada quadro, provendo o ganho de paralaxe de movimento. Neste caso, utilizamos um algoritmo baseado na técnica de *relief map* [30] para aproximar a superfície de corte. Ainda, os objetos de interesse podem ser definidos pelo desenho de quaisquer objetos, no contexto da visualização de reservatórios de petróleo, expandindo a possibilidade de aplicações deste trabalho.

3
Algoritmo de corte orientado ao observador

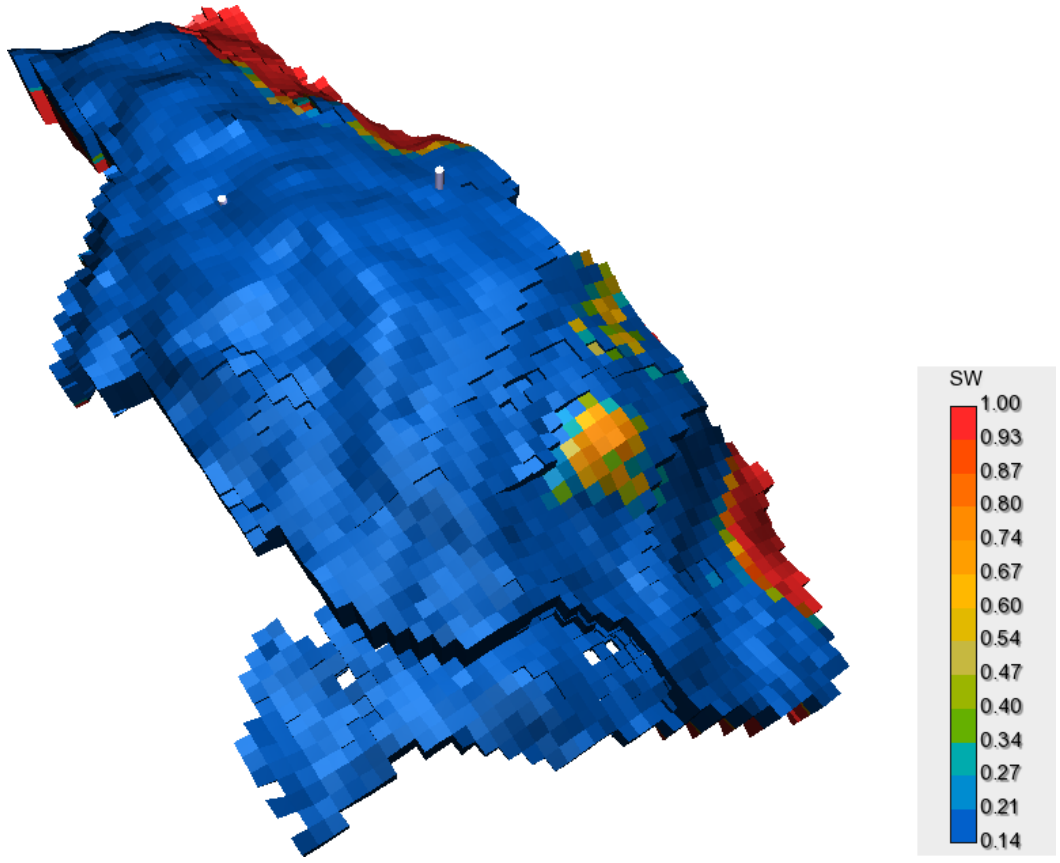
Neste capítulo, propõe-se um algoritmo de corte orientado ao observador, o qual pode ser utilizado por especialistas para avaliar as propriedades no entorno de objetos de interesse, com ênfase em poços de reservatório de petróleo. O algoritmo é baseado na superfície de corte proposta por Burns e Finkelstein [22] e o mapeamento das propriedades na superfície do corte conforme proposto por Franceschin [3]. A Figura 3.1 apresenta o corte aplicado a um modelo de reservatório com dois poços como objetos de interesse.

3.1
Algoritmo proposto

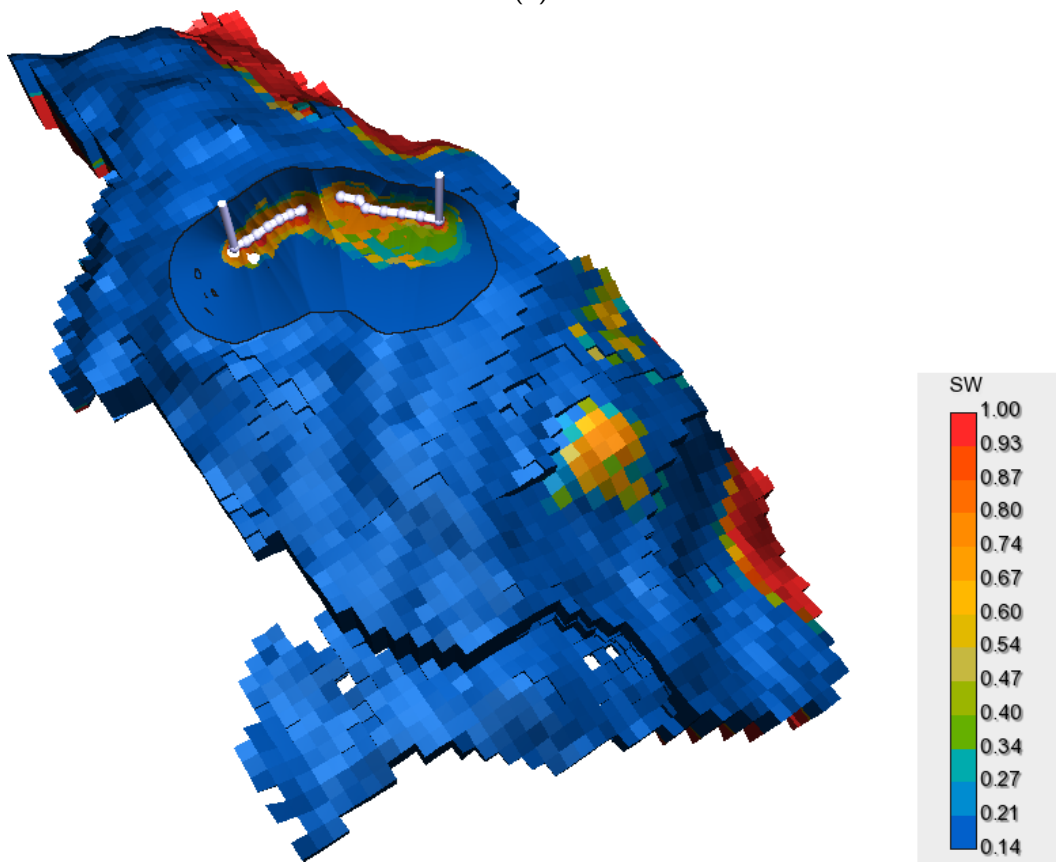
O algoritmo proposto é realizado através de uma série de passadas utilizando renderização para textura. A composição final da cena é obtida com renderização para janela ao aplicar iluminação e desenhar os objetos de interesse. Alguns parâmetros podem ser controlados livremente, como a posição do observador e a abertura do corte.

A iluminação é efetuada através da técnica de *deferred shading*, a qual é eficiente por aplicar cálculos de iluminação apenas uma vez por *pixel*, ao contrário de técnicas tradicionais, que efetuam *shading* em todos fragmentos candidatos a ocupar esse pixel. São criados inicialmente 3 *buffers*, chamados coletivamente de *G-buffers*, com as dimensões da janela para armazenar as posições, as cores e as normais da geometria (Figura 3.2). Os *buffers* são representados por componentes RGBα com precisão de 32 bits, utilizados ao longo de todo processo de renderização para textura do modelo. Desta forma, ao fim de cada passada em que a geometria é desenhada, somente os atributos dos fragmentos mais próximos à câmera estarão escritos nos *G-buffers*.

Alguns pré-requisitos devem ser satisfeitos antes de iniciarmos o algoritmo de corte. Deve-se definir o conjunto de objetos de interesse, seja este apenas um poço, vários poços ou quaisquer outros objetos, caracterizando o foco. O conjunto de objetos secundários é definido por todas células do modelo de reservatório, e eventuais demais objetos, caracterizando o contexto.



(a)



(b)

Figura 3.1 – (a) Modelo de reservatório de petróleo com saturação de água (SW) associada às células e dois poços injetores. (b) Poços definidos como objetos de interesse revelados pelo algoritmo de corte.

Na inicialização é gerada a grade regular uniforme necessária para realizar o mapeamento do campo escalar na superfície de corte com a técnica proposta por Franceschin [3]. As dimensões da grade (n_x, n_y, n_z) influenciam significantemente no desempenho do cálculo de interseção. Um modelo muito complexo exige uma grade com alta resolução, ao passo que um modelo simples, com pequena variação de tamanho das células, é eficiente com uma grade com menor resolução. Para fazer uso desta técnica, basta definir o valor da variável k, que indica a resolução da grade com base nas dimensões topológicas (n_i, n_j, n_k) do modelo:

$$n_x = k\, n_i \qquad n_y = k\, n_j \qquad n_z = k\, n_k$$

Após o valor de k ser definido, a grade regular uniforme é criada e enviada para placa gráfica. De maneira geral, a medida que k cresce o desempenho aumenta, porém, maior será o espaço alocado na memória da placa gráfica. O teste de interseção, descrito na Seção 2.3, é realizado em tempo de renderização.

Podemos sumarizar o algoritmo completo proposto em seis etapas:

1. Desenhar as *back-faces* dos objetos de interesse (OI)
2. Computar a superfície de corte
3. Desenhar o modelo com descarte de fragmentos
4. Desenhar a superfície de corte no modelo
5. Aplicar iluminação
6. Desenhar as *front-faces* dos objetos de interesse

Para auxiliar a compreensão de cada etapa nas seções a seguir, será possível observar o estado atual do corte no modelo com iluminação já aplicada, ainda que esta seja explicada posteriormente. A próxima seção detalha a criação da superfície de corte, a qual compõe as duas primeiras etapas. A Seção 3.3 identifica algumas limitações ao desenhar o modelo de reservatório de petróleo e como podemos contorná-las. A Seção 3.4 apresenta o desenho da superfície de corte com mapeamento do campo escalar e a adaptação realizada na técnica de [3]. Por fim, detalhes sobre a iluminação, pós-processamento e o desenho de objetos de interesse são abordados na Seção 3.5.

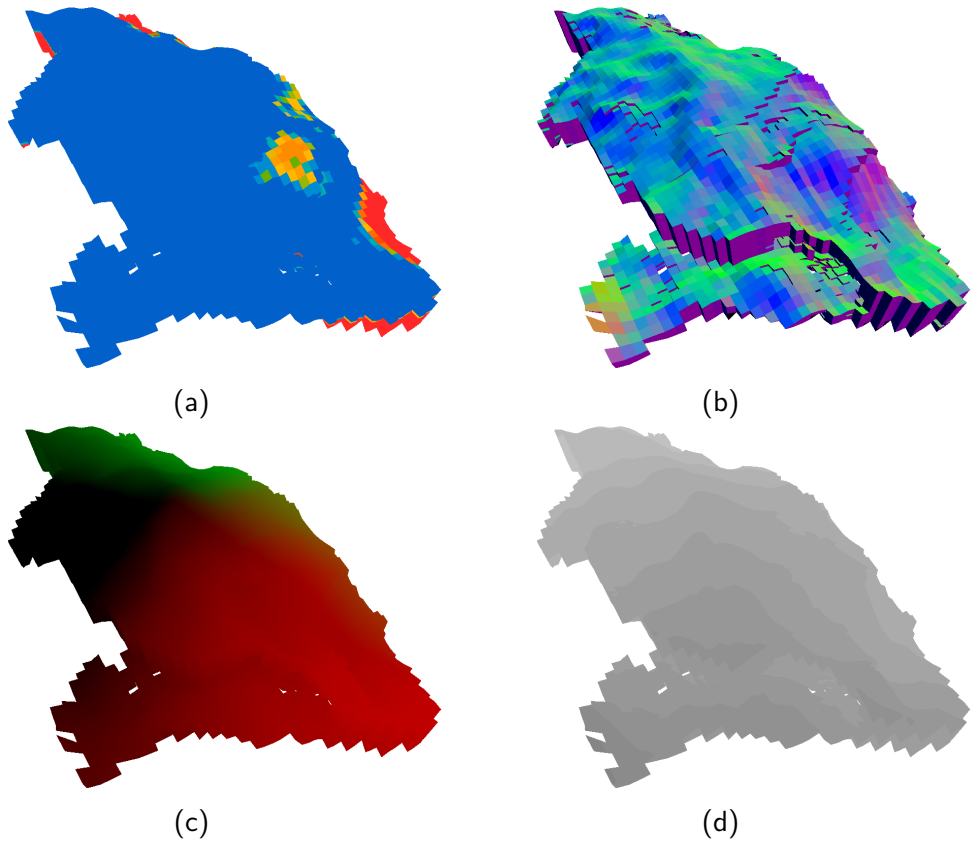


Figura 3.2 – Representação dos *G-buffers* para um modelo de reservatório. (a) Mapa de cor, (b) mapa de normal, (c) mapa de posição e (d) profundidade extraída da componente α do mapa de posição.

3.2 Gerando a superfície de corte

A superfície de corte é representada por um mapa de profundidade com as dimensões da janela. Esse mapa resulta da aplicação da transformada de distância num conjunto de sementes, os quais são representados pelos *pixels* das *back-faces* dos objetos de interesse (OI) renderizados. Observa-se que devido a natureza desse processo, objetos que estão fora do volume de visão não são renderizados e, portanto, não contribuem para a superfície de corte gerada (Figura 3.3).

O algoritmo de *jump-flooding* [1], descrito na Seção 2.2.1, é utilizado para obtenção deste mapa, para o qual são criados dois *buffers* com as dimensões da janela. Estes *buffers* são representados por componentes RGB com precisão de 32 bits. As componentes RG armazenam a posição da semente mais próxima e a componente B armazena a profundidade da superfície de corte calculada, ambas no espaço da janela. Antes de prosseguir com este algoritmo, as *back-faces* dos objetos de interesse são renderizadas para textura com a função de z modificada para guardar os fragmentos com maior profundidade, preenchendo

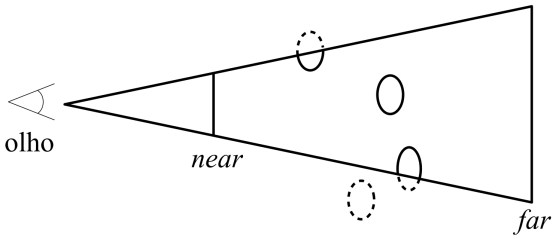


Figura 3.3 – Partes dos objetos de interesse fora do volume de visão, tracejados, não influenciam o corte por não serem rasterizados.

um dos *buffers* e que será o mapa de sementes inicial. Inicia-se o algoritmo anexando este mapa para leitura e o outro *buffer* para escrita na renderização para textura. Durante cada passo, para cada *pixel* p sendo processado, avalia-se a profundidade dos 8 vizinhos com a Equação (2.3), aplicando a correção de perspectiva da Equação (2.4). Com isso, obtemos a seguinte equação:

$$d(p) = \max_{q \in I} \left(z(p) + \frac{(PM_{sz} + z(p)) \, \|q - p\|}{\tan(\theta)} \right) \tag{3.1}$$

Ao fim de cada passo, intercala-se o mapa a ser lido com o *buffer* anexado na renderização para textura. O último *buffer* escrito representa o mapa de profundidade associado à superfície do corte. A Figura 3.4 apresenta 4 estágios deste algoritmo aplicado aos objetos de interesse representados por dois poços.

Opcionalmente, pode-se utilizar um mapa de profundidade com resolução igual à metade das dimensões da janela de visualização. Para isso, antes do desenho dos objetos de interesse, é ativada uma *viewport* com essas dimensões e prossegue-se com o algoritmo de *jump-flooding*. Após o mapa de profundidade ter sido gerado a *viewport* é restaurada com as dimensões iniciais. A superfície de corte representada por esse mapa é acessada normalmente nas próximas etapas do algoritmo, sem que nenhuma outra modificação seja necessária. Essa abordagem objetiva ganho de desempenho com pequeno impacto na visualização.

3.3 Desenho do modelo aplicando o corte

Desenhar o modelo completo com todas suas células, o conjunto de objetos secundários, tem um custo computacional elevado. Modelos de reservatório reais apresentam um total de células na ordem de milhões. Para cada célula, a placa gráfica processa 2 triângulos por face, totalizando 12 triângulos. Uma maneira usual de otimizar o desenho de elementos fechados é desenhar somente as *front-faces*, ativando a função de descarte (*culling*) das *back-faces*, já que

elas não são visíveis.

A renderização pode ser otimizada ainda mais ao extrair as faces externas do modelo. Um *buffer* de vértices é criado em pré-processamento para armazenar a geometria destas faces através de um processo simples. Para cada face em uma célula ativa, se a célula vizinha a esta for inativa ou inexistente, adiciona-se os vértices da face ao *buffer* de vértices. Junto com os vértices, pode-se considerar outros atributos associados a geometria que sejam necessários para renderização, como normais, coordenadas de textura, e assim por diante. Após processar as faces de todas células ativas, o *buffer* de vértices gerado pode ser utilizado para renderizar o modelo.

Ambas otimizações, descritas acima, invalidam a maneira que o método de [22] colore a superfície de corte. Para contornar essa limitação, utilizamos a técnica de [3] para desenhar a superfície de corte e prosseguimos para o desenho do modelo somente com as faces externas que forem *front-faces*.

Neste momento, pode-se aplicar o corte no desenho dos objetos secundários utilizando o mapa de profundidade gerado pelas etapas anteriores. Este corte é obtido através de um *fragment shader* com operações no espaço da janela. Para cada fragmento gerado com coordenadas (x, y) e profundidade

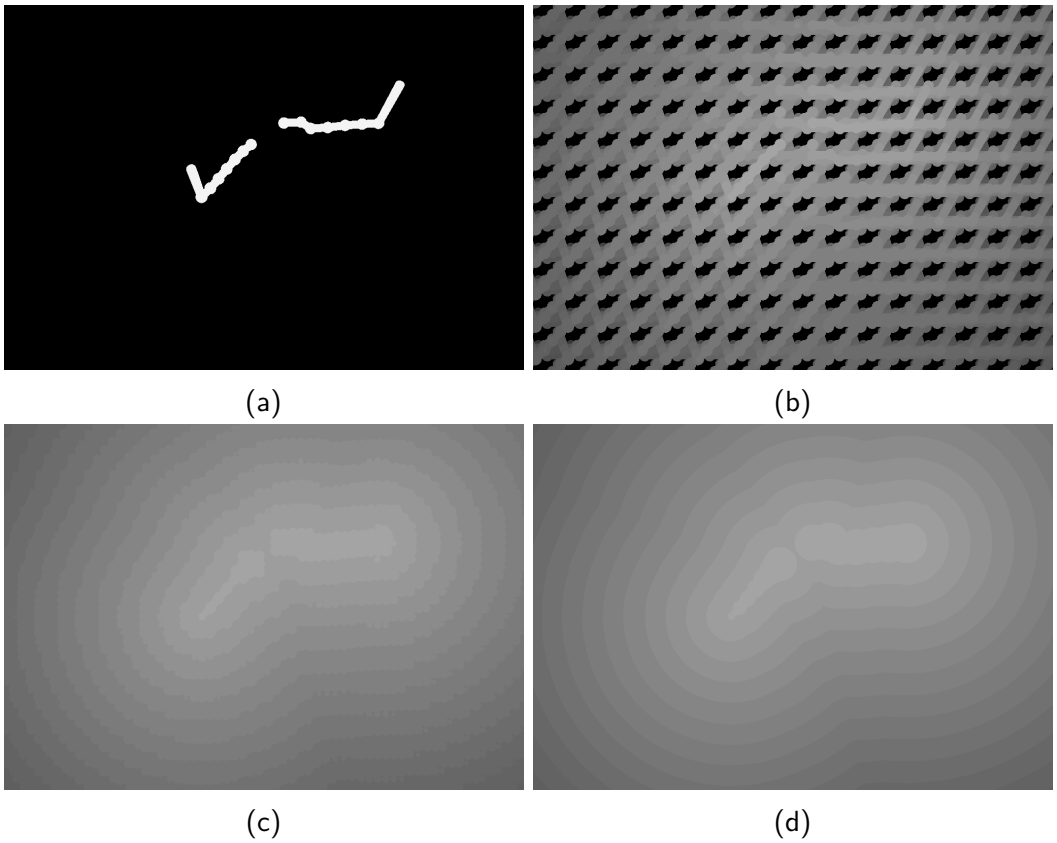


Figura 3.4 – Estágios do algoritmo de *jump-flooding* ilustrados com os valores das profundidades. A cor foi reescalada para visualização. (a) Mapa de profundidade inicial, (b) e (c) estágios intermediários, e (d) superfície de corte final.

z_f, compara-se z_f com a profundidade da superfície de corte z_c nas mesmas coordenadas (x, y). Se $z_f < z_c$, implica que o fragmento está entre a câmera do observador e a superfície de corte e, portanto, será descartado. Caso contrário, o programa de fragmento continua e os *G-buffers* são escritos com a posição, normal e cor correspondentes. Como resultado desta etapa, podemos observar na Figura 3.5 que outras faces externas estão presentes dentro do modelo são reveladas.

3.4 Desenho da superfície de corte

Faces externas em contato são encontradas em regiões de falhas geológicas, o que representa um problema na renderização dessas estruturas pelo conflito de profundidade. A técnica de mapeamento de campos escalares em seções de corte arbitrárias [3] apresenta resultados que minimizam esse problema, além de descartar a necessidade de desenhar todas células do modelo, já que o mapeamento de interseção da superfície de corte depende somente do modelo previamente armazenado em GPU.

A superfície de corte é desenhada por meio de um quadrilátero com as dimensões da janela. Utiliza-se a coordenada normalizada (x, y) de cada

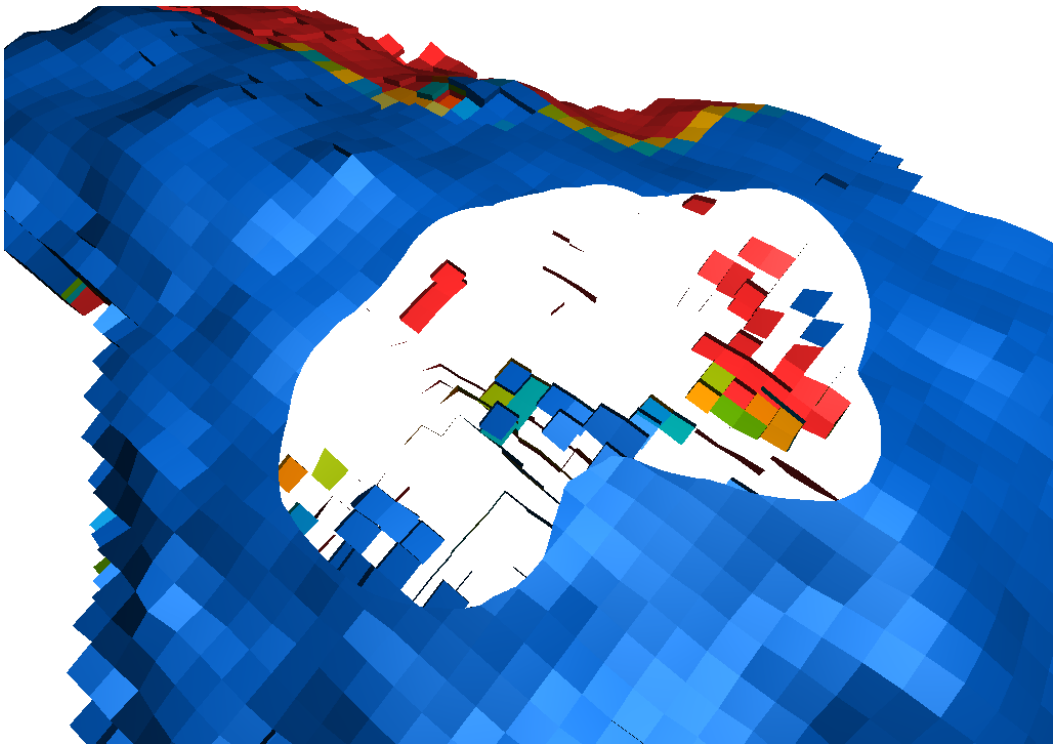


Figura 3.5 – Modelo desenhado com descarte de fragmentos revelando estruturas internas.

fragmento gerado para acessar o valor z_c do mapa de profundidade, definindo a posição $P_{jan} = (x, y, z_c)$ da superfície de corte no espaço da janela. Transforma-se P_{jan} para obter a posição P_{obj} da superfície de corte no espaço do objeto. Dada a condição de descarte no desenho dos objetos secundários, esta posição representa um espaço no mundo ainda não ocupado por outros fragmentos com profundidade igual ou inferior a superfície de corte. A posição P_{obj} é então utilizada no teste de interseção com a grade regular uniforme, conforme descrito na Seção 2.3. Se ocorrer interseção com alguma célula, o *G-buffer* é escrito com a posição P_{obj} e a cor do campo escalar correspondente. Caso contrário, se P_{obj} não tiver interseção com nenhuma célula, o fragmento é descartado, mantendo-se os valores escritos previamente. O resultado desta etapa está ilustrado na Figura 3.7.

3.4.1 Mapeamento do campo escalar em malhas não estruturadas adaptado

Ao comparar a renderização de uma célula e o mapeamento da interseção desta em uma superfície arbitrária, é possível observar pequenas diferenças próximos aos limites da célula. Isso ocorre pela diferença de precisão entre sua triangulação, uma operação linear entre os nós, e o método numérico utilizado para definir se um fragmento da superfície está dentro da célula (Figura 3.6). Neste método, a interseção entre o ponto e a célula é feita considerando-se o espaço paramétrico do elemento, sem assumir faces planares. Já na visualização do corte, isso resulta no aparecimento de pequenos espaços vazios em regiões de fronteira do corte com o modelo (Figura 3.7). A técnica de [3] foi adaptada para considerar células nas quais o método numérico converge, mas falha ao verificar se as coordenadas naturais estão dentro do intervalo de $[-1, 1]$. Para cada célula da lista contida no *voxel* da grade regular, na qual ocorrer essa situação, prossegue-se com as seguintes etapas:

1. Efetuar *clamp* entre [-1,-1] na coordenada natural calculada
2. Mapear a coordenada natural para o espaço físico e determinar a distância entre esta e a posição do fragmento
3. Armazenar a posição da célula que minimiza essa distância

A célula armazenada só será considerada como retorno do teste de interseção ao percorrer toda lista e, por fim, verificando se a distância calculada é menor que uma tolerância l definida pelo usuário. Porém, enquanto percorre-se a lista, se o método numérico convergir para uma célula e suas coordenadas naturais estiverem dentro do intervalo de $[-1, 1]$, esta terá prioridade e será

retornada pelo teste de interseção, ignorando o processo descrito acima. Em outras palavras, as posições que representam os espaços vazios serão preenchidos com a cor da célula mais próxima, desde que a distância entre a posição e a célula seja menor que l. O resultado desta abordagem pode ser visualizado na Figura 3.8.

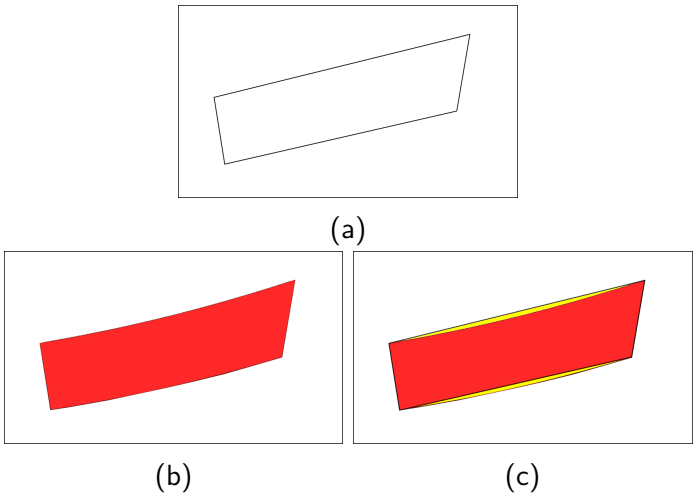


Figura 3.6 – Seções transversais de uma mesma célula: (a) obtida com triangulação, (b) obtida com mapeamento e (c) a diferença entre as duas é destacada em amarelo.

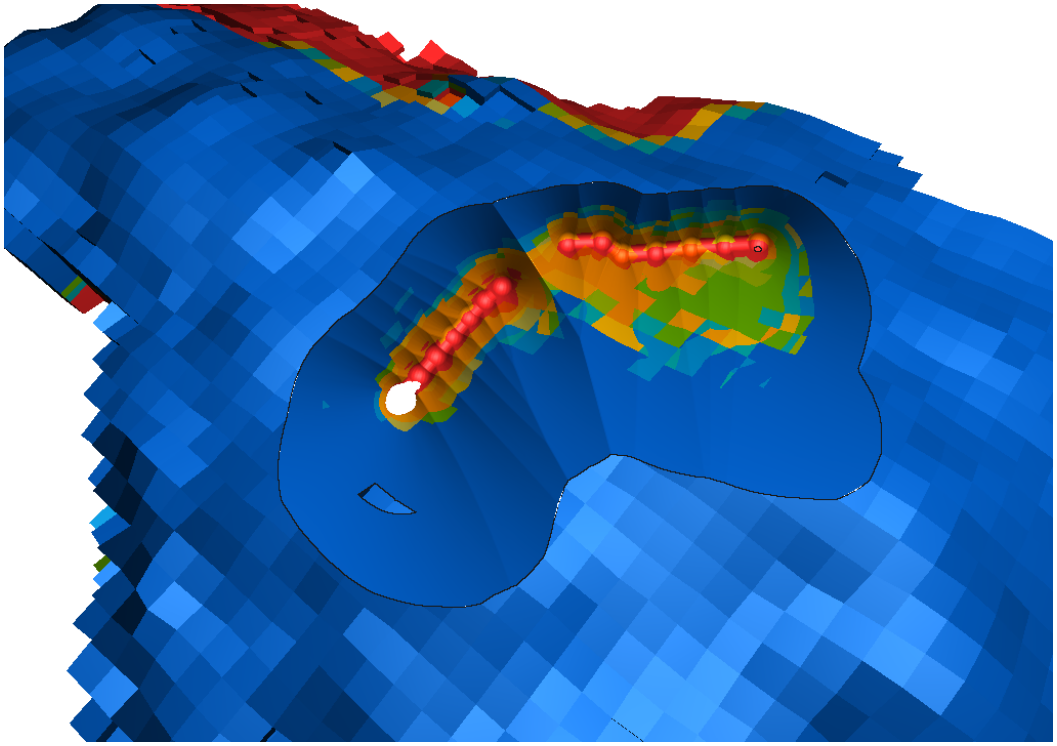


Figura 3.7 – Modelo desenhado com a superfície de corte. Destacam-se pequenos espaços vazios próximos a borda do corte.

3.5 Aplicando iluminação e desenho dos objetos de interesse

A composição final do corte é obtida efetuando os cálculos de iluminação aos mapas de cor, posição e normal, produzidos nas etapas anteriores. Esse processo toma parte com renderização para janela, onde desenha-se um quadrilátero com as dimensões da janela, tal que, cada posição do mapa será acessada para compor um *pixel*.

O cálculo de iluminação aplicado a todos objetos é realizado com base no modelo de iluminação local com componentes ambiente, difuso e especular (*ambient, diffuse, specular*, ADS). São definidas duas fontes de luz, uma acima do modelo e outra na posição da câmera. A primeira contribui com 60% da iluminação e a segunda com os 40% restantes.

Para fragmentos que pertencem a superfície de corte, é necessário calcular a sua normal tomando a coordenada (x, y) dos fragmentos vizinhos e os valores de z_c correspondentes no mapa de profundidade. Deriva-se a normal com o produto vetorial dos vetores definidos pelas posições dos 4 vizinhos diretos. Portanto, os vetores são definidos, da posição do fragmento inferior b até o superior t, e da posição do fragmento a esquerda l para o da direita r, conforme Figura 3.9.

Regiões de fronteira da superfície de corte com as faces externas do modelo são desenhadas com cor diferenciada para indicar seus limites. Para cada fragmento que pertence aos objetos secundários, avalia-se os *pixels* vizinhos verificando se estes pertencem a superfície de corte. Caso pertença, o fragmento é renderizado com a cor indicada.

A visualização final (Figura 3.10) é obtida com o desenho dos objetos de interesse e outros objetos que sejam necessários a visualização podem ser desenhados neste momento. Opcionalmente, pode-se desenhar as faces externas do modelo e a superfície de corte com aramado (Figura 3.11), ou apenas uma delas. Para as faces externas é tomada como base a técnica proposta por Bærentzen et al. [32]. Já o desenho do aramado na superfície de corte provém do próprio método de interseção e é desenhado somente para a camada k de cada célula, destacando os horizontes do modelo de reservatório.

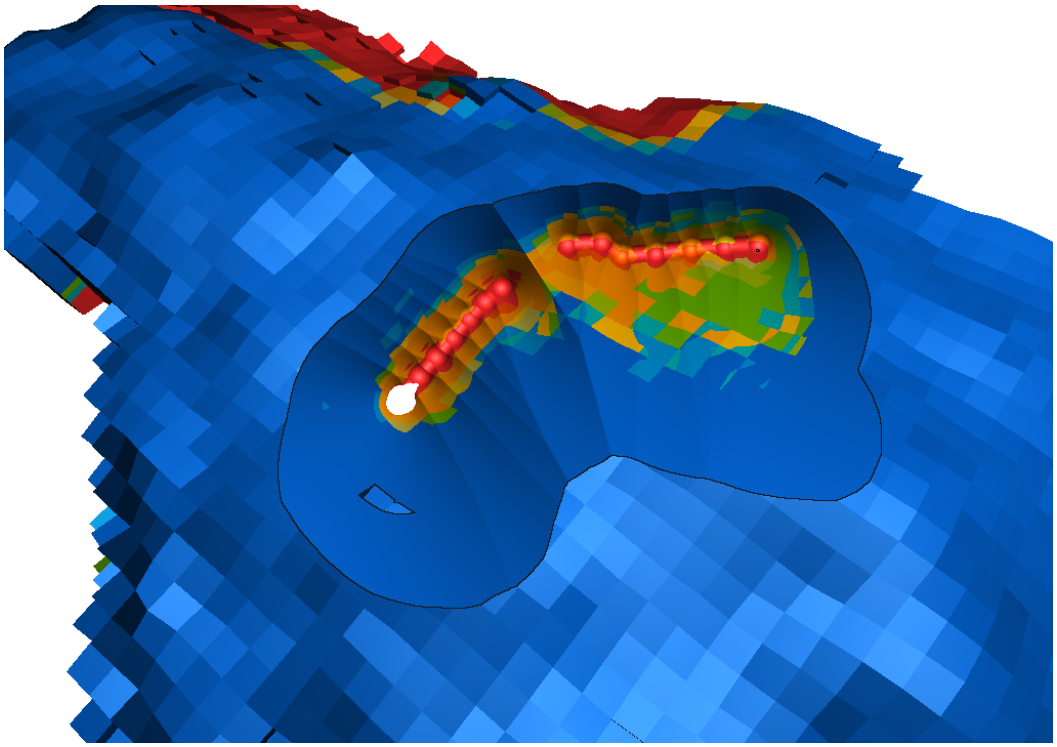
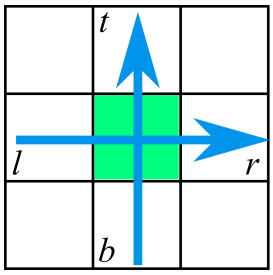
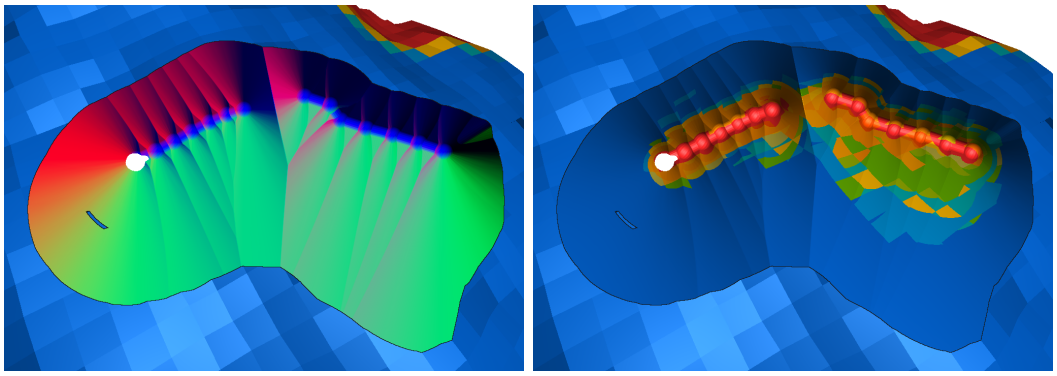


Figura 3.8 – Desenho da superfície de corte com o uso de tolerância preenchendo os espaços vazios na borda do corte.



(a)



(b) (c)

Figura 3.9 – (a) Cálculo da normal a partir da posição dos vizinhos, (b) mapa de normal destacado para superfície de corte e (b) superfície de corte iluminada.

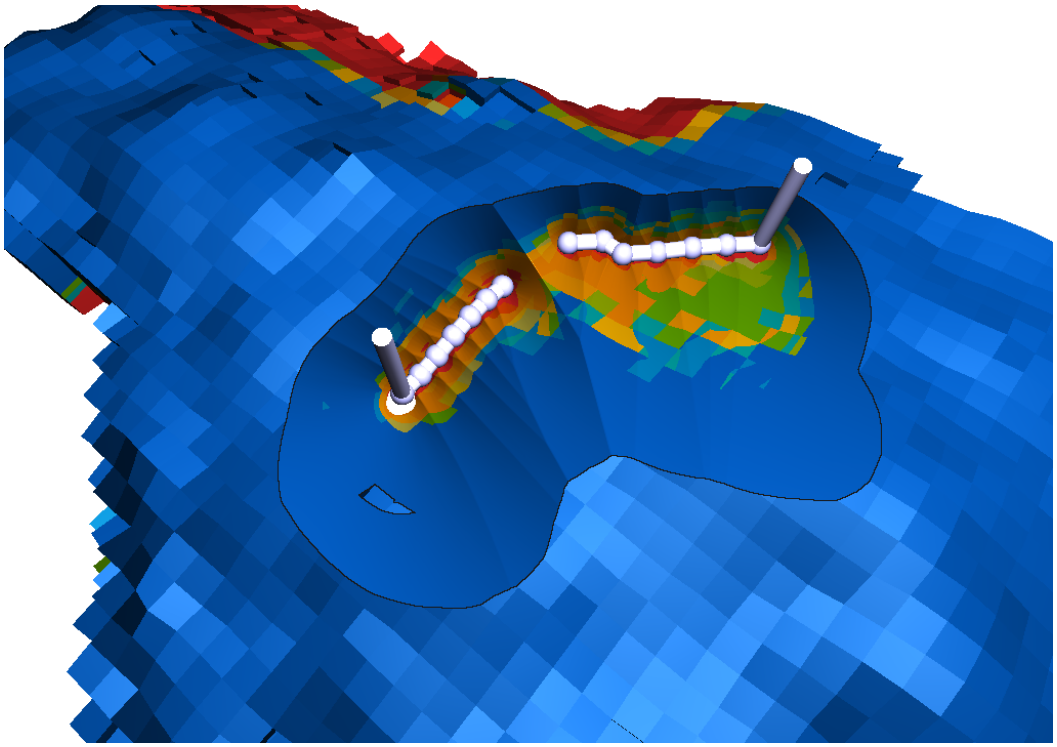


Figura 3.10 – Visualização final do algoritmo de corte proposto.

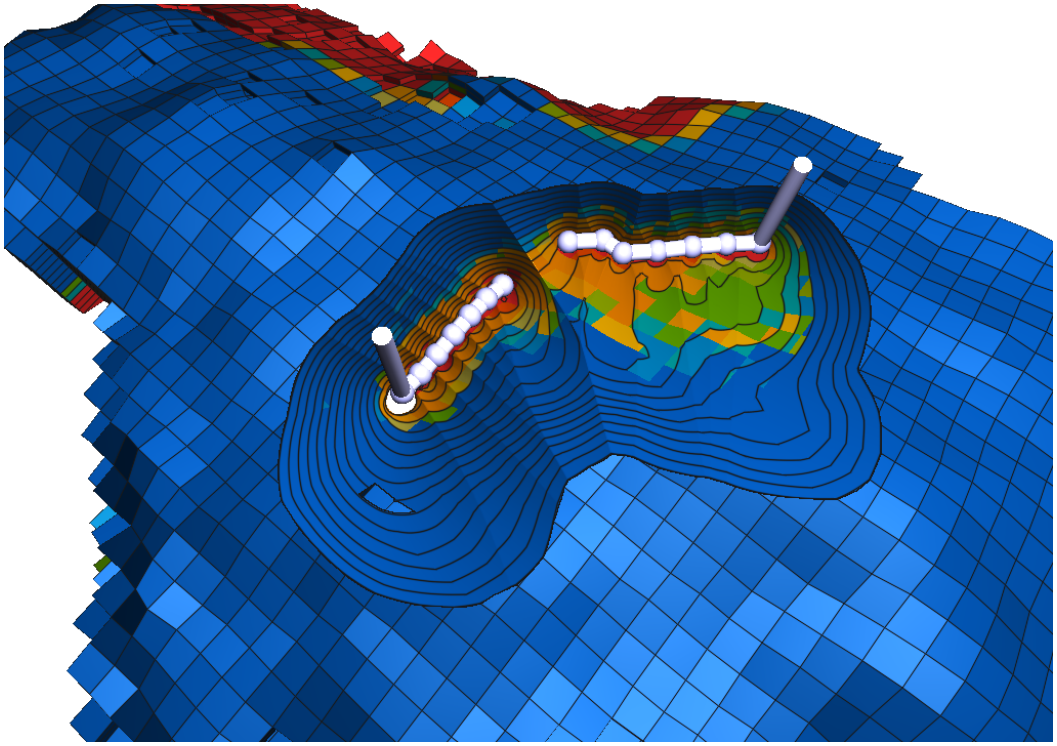


Figura 3.11 – Visualização final do algoritmo de corte proposto com aramado ativo para as faces externas e para superfície de corte.

4
Algoritmo de corte desacoplado do observador

Em uma visualização interativa, é necessário permitir que o corte gerado seja explorado a partir de diferentes pontos de vista. Quando o corte é orientado ao observador, ocorre ausência do efeito de paralaxe de movimento, prejudicando a percepção da profundidade dos objetos de interesse e dos objetos secundários. Esse efeito é perceptível quando um usuário move-se em uma cena observando um ponto fixo e objetos no seu campo de visão também aparentam se mover. Observa-se que objetos entre o olho e o ponto fixo aparentam mover-se em direção oposta à do observador, ao passo que objetos além do ponto fixo aparentam mover-se na mesma direção. O campo de movimento aparente é não uniforme e com velocidade proporcional à distância do olho ao objeto [33].

Para auxiliar na inspeção e avaliação dos objetos revelados no processo de visualização, pretende-se reconstruir o corte, previamente gerado, para diferentes posições da câmera. As posições associadas ao mapa de profundidade da superfície de corte encontram-se no espaço da janela, para avaliá-las a partir da posição da câmera atual, é possível explorar as transformações do espaço em coordenadas homogêneas (Figura 4.1), utilizadas no pipeline gráfico. Uma posição no espaço do objeto é transformada para o espaço do olho ao multiplicá-la pela matriz de *modelview* (M_v). Em seguida, obtém-se a posição no espaço de *clip* multiplicando o resultado pela matriz de projeção (M_p), bastando mapeá-la para obter a posição no espaço da janela (J_{map}). Esse processo é invertível, partindo de uma posição no espaço do janela, basta mapear para o espaço de *clip* (C_{map}), e multiplicar pelas matrizes inversas de projeção (M_p^{-1}) e, na sequência, de *modelview* (M_v^{-1}), para obter a posição no espaço do olho e do objeto, respectivamente.

Neste capítulo, propõe-se um algoritmo de corte desacoplado do observador, o qual estende o algoritmo de corte proposto no Capítulo 3, ao passo que mantém o mesmo objetivo, visando melhorar sua interpretação. Utiliza-se uma abordagem baseada na técnica de *relief map* [30] para gerar a visualização do corte. A Figura 4.2 apresenta o corte aplicado a um modelo de reservatório com dois poços como objetos de interesse.

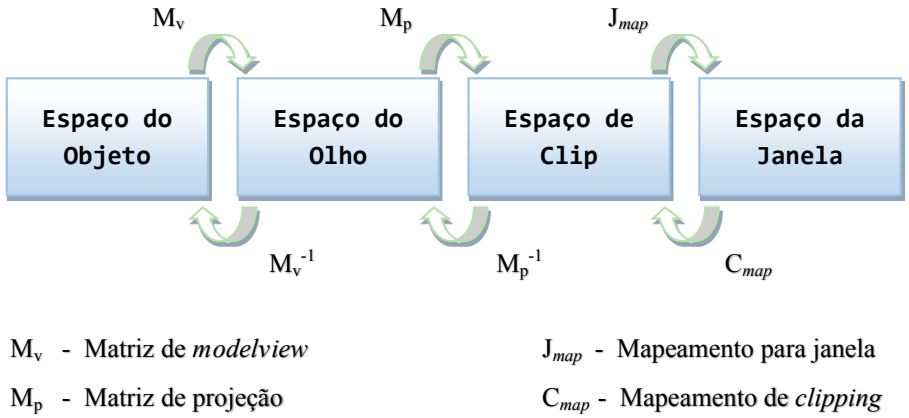


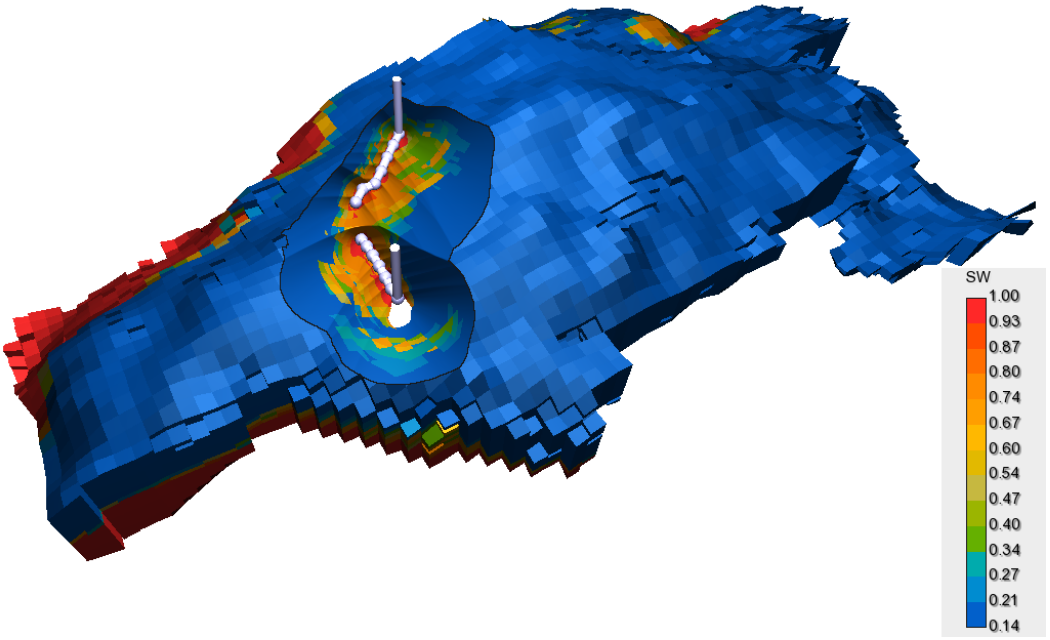
Figura 4.1 – Transformações do espaço de coordenadas homogêneas.

4.1 Algoritmo proposto

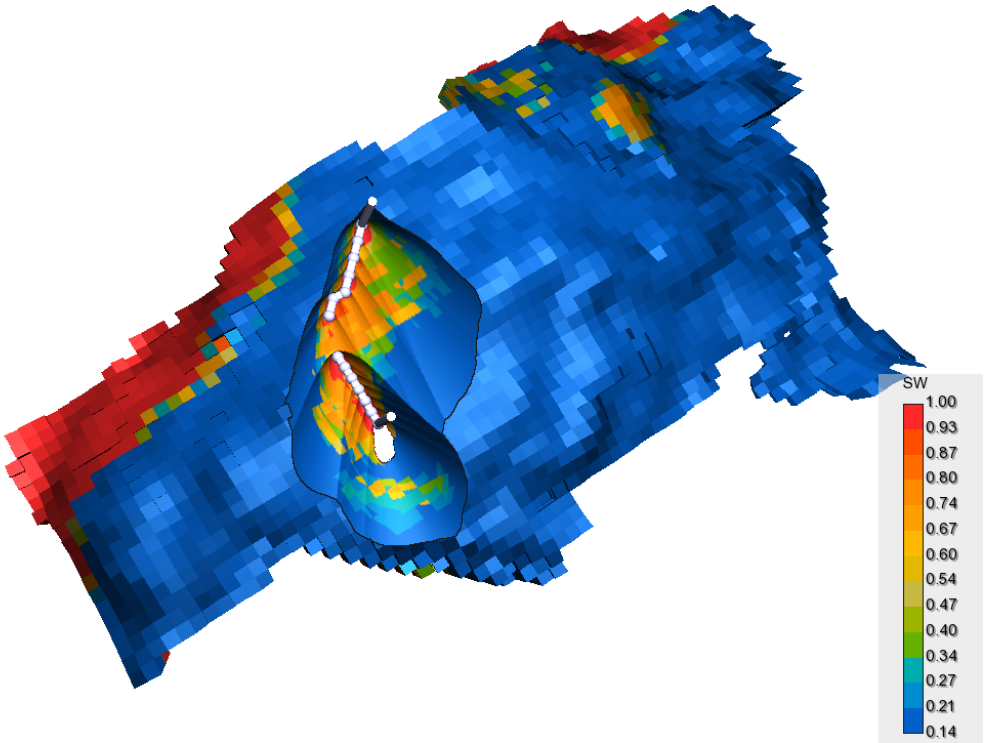
Modificou-se o algoritmo proposto no Capítulo 3 para incorporar um modo de visualização em que o corte é desacoplado do observador. Quando este modo é ativado, a superfície de corte é gerada uma única vez, produzindo um mapa de profundidade que será utilizado para reconstruir o corte. Também é realizado apenas um mapeamento do campo escalar, com base nesta superfície, produzindo um mapa de cor. Por fim, é mantida a renderização em múltiplas passadas e o uso da técnica de *deferred shading* para aplicar iluminação. O algoritmo completo proposto consiste das seguintes etapas:

1. Gerar o corte orientado ao observador (Capítulo 3) até que seja solicitado o modo desacoplado
2. Gerar o mapa de profundidade e o mapa de cor
3. Desenhar o modelo com a superfície de corte
   (a) Aplicar o corte via descarte de fragmentos
   (b) Aproximar a superfície de corte via traçado de raio
4. Aplicar iluminação e desenhar os OI

A próxima seção apresenta os requisitos necessários para dar início ao algoritmo proposto. A Seção 4.3 detalha a operação de descarte de fragmentos utilizando o mapa de profundidade gerado previamente. A Seção 4.4 descreve como podemos gerar o corte e aproximá-la com o observador em novas posições.



(a)



(b)

Figura 4.2 – (a) Modelo de reservatório de petróleo com corte aplicado e (b) modelo com corte aplicado visto a partir de outra posição.

4.2 Pré-requisitos

Estando em curso a visualização do corte orientado ao observador, deve-se ativar o modo desacoplado para inspecionar o corte a partir de outras posições. Nesse momento, a matriz de projeção M_{rp} e a matriz de *modelview* M_{rv}, utilizadas na iteração que antecede a ativação deste modo, são armazenadas separadamente, para que seja possível reconstruir o espaço de referência em que o corte foi gerado. A superfície de corte agora é fixa e definida pelos mapas de profundidade e de cor, também gerados nessa iteração. Os mapas mantêm as características descritas anteriormente, com as dimensões da janela onde, para cada coordenada (x, y), estão associados valores de z no espaço da janela, ao mapa de profundidade, e valores do campo escalar em uma escala de cores, ao mapa de cor (Figura 4.3). O formato e a abertura do corte só podem ser modificados ao desativar o modo desacoplado. Portanto, durante a execução do algoritmo, as seguintes matriz estão disponíveis:

- M_{rp}: Matriz de projeção no espaço de referência
- M_{rv}: Matriz de *modelview* no espaço de referência
- M_p: Matriz de projeção corrente
- M_v: Matriz de *modelview* corrente

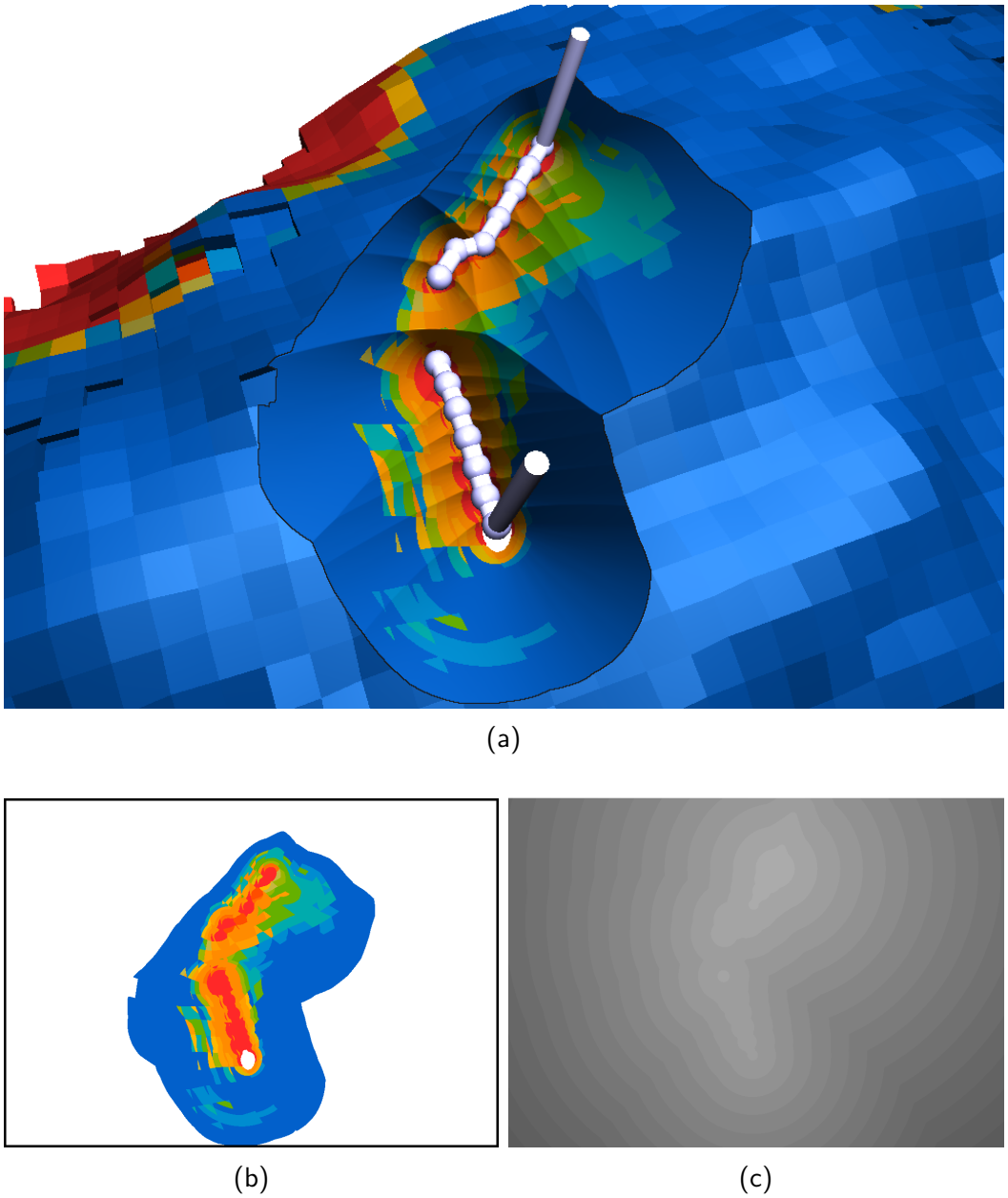
4.3 Desenho do modelo com a superfície de corte

O desenho do conjunto de objetos secundários ocorre de forma usual, como descrito na Seção 3.3, somente com as *front-faces* das faces externas do modelo. Porém, o procedimento de comparação da profundidade com a superfície de corte no *fragment shader* é modificado. Cada fragmento é definido por sua posição $P_j = (x, y, z)$ no espaço da janela corrente, onde (x, y) representam a coordenada do fragmento e z sua profundidade. Mapea-se P_j para o espaço de *clip* corrente P_{clip}, para então transformá-lo para o espaço de *clip* de referência P_{rclip}, no qual o corte foi gerado:

$$P_{rclip} = \Big(M_{rp} \times M_{rv}\Big)\Big(M_v^{-1} \times M_p^{-1}\Big)P_{clip} \tag{4.1}$$

Então, após a divisão de perspectiva de P_{rclip} pela a coordenada homogênea, verificamos se as componentes (x, y, z) obtidas estão em $[-1, 1]$. Isso indica que a posição do fragmento corrente corresponde a uma posição válida no espaço de referência. Neste caso, podemos prosseguir com o mapeamento dessa posição para o espaço da janela de referência, obtendo $P_r = (x_r, y_r, z_r)$. Utiliza-se as coordenadas (x_r, y_r), para acessar o valor z_c da superfície de corte no

mapa de profundidade. Se $z_r < z_c$, então o fragmento está entre a câmera do observador e a superfície de corte e, portanto, como seria descartado no espaço de referência, é descartado no espaço corrente, revelando um espaço vazio a ser preenchido pelo desenho da superfície de corte. Caso contrário, o programa de fragmento continua e os *G-buffers* são escritos com a posição, normal e cor correspondentes à face externa do modelo no espaço corrente.



(a)

(b) (c)

Figura 4.3 – (a) Corte orientado ao observador e a superfície de corte associada, representada pelo (b) mapa de cor, e pelo (c) mapa de profundidade, cujas cores estão escaladas para destacar a variação de profundidade.

4.4 Desenhando a superfície de corte

A área revelada pelo processo de descarte (Figura 4.4) não pode ser diretamente preenchida pelo mapa de profundidade associado à superfície de corte. Este mapa foi gerado no espaço da janela de referência (Figura 4.5) e possui uma resolução definida pelas dimensões desta. Ainda que cada ponto desse mapa possa ser discretizado e armazenado em um *buffer* de vértices para ser renderizado como malha, a quantidade de vértices seria proporcional a essa resolução, aumentando o custo para renderização e necessitando uma abordagem mais complexa para completar os espaços gerados pela variação da profundidade.

Uma maneira mais simples para desenhar a superfície de corte é aproximar a sua profundidade através de traçado de raios primários (*ray casting*) no espaço da janela de referência, ao identificar que o fragmento corrente deve ser descartado. Cada raio deve percorrer o volume de visão (*frustum*), seguindo a direção definida pela posição do observador até a posição do fragmento, em uma busca linear com passos fixos, verificando se o valor da profundidade ao longo do raio é maior que a profundidade associada à superfície de corte. Quando isso ocorrer, determinamos que já houve interseção. Logo, para aproximar o ponto de interseção utiliza-se busca binária com a posição anterior e a posterior à interseção, onde os valores de profundidade são comparados novamente. Em poucos passos, que reduzem o intervalo de busca, ocorre convergência com uma boa aproximação da superfície de corte.

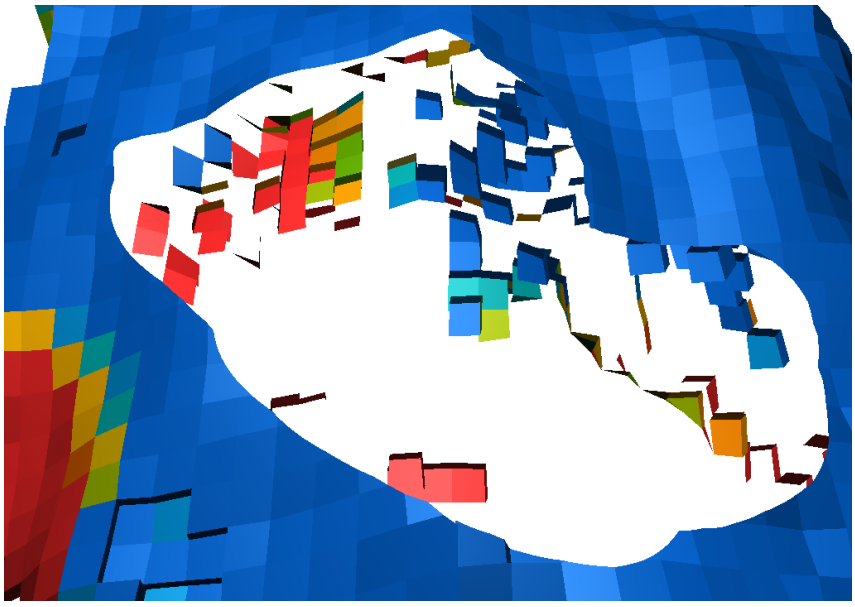


Figura 4.4 – Desenho do modelo aplicando descarte de fragmentos com base em uma superfície de corte gerada em outro espaço de referência.

Detalhando o procedimento, para cada fragmento que for descartado pela comparação de profundidade da etapa anterior, define-se um raio com a seguinte equação paramétrica:

$$R(t) = O + t\vec{d} \tag{4.2}$$

A origem do raio O é definida pela posição do fragmento P_r previamente transformada para o sistema de referência do corte (Seção 4.3). Porém, não necessariamente é possível transformar a posição do observador, pois a nova câmera pode estar atrás da câmera de referência. Portanto, deriva-se a direção do raio, $\vec{d}$, tomando a posição do fragmento corrente $P_j = (x, y, z)$ e outro ponto com posição $Q_j = (x, y, z_d)$, tal que, $z_d = z - 0,001$. Transformando ambos para o espaço da janela em que o corte foi gerado, obtém-se, respectivamente, P_r e Q_r. Portanto, a direção é dada por $\vec{d} = P_r - Q_r$. Reescrevendo a Equação (4.2), temos:

$$R(t) = P_r - t(P_r - Q_r) \tag{4.3}$$

Pretende-se percorrer a trajetória do raio variando t, no intervalo de $[t_{min}, t_{max}]$, ao passo que a profundidade associada com sua posição é comparada com a superfície de corte. Quando $t = 0$, temos o ponto de partida da busca linear em P_r. Porém, para determinar o tamanho do passo δ de variação em t, é necessário obter o valor para t_{max} em que o raio intercepta o *frustum*, ou seja, o seu limite de propagação. Como o *frustum* no espaço da janela pode ser visto como uma caixa alinhada com os eixos coordenados (AABB), basta realizar um teste de interseção do raio com os planos da caixa (*slab method*) para determinar t_{max}. Temos que o tamanho do passo é dado por:

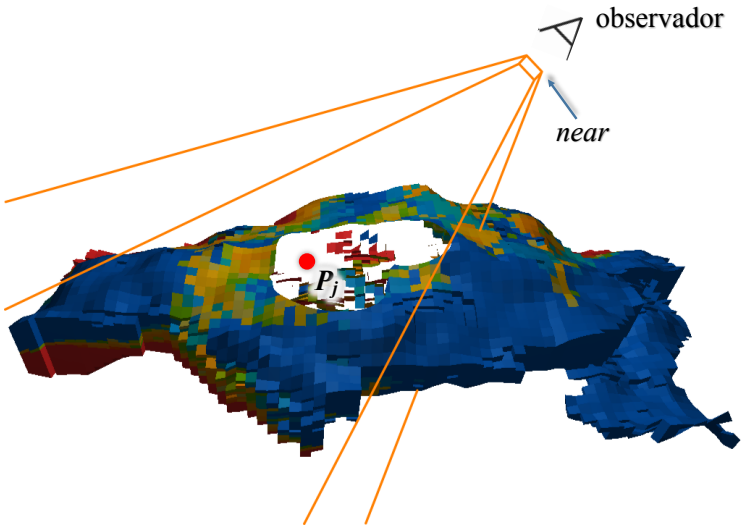


Figura 4.5 – *Frustum* (em laranja) do espaço de referência em que um corte foi gerado. Na região de descarte esta ilustrado em vermelho a posição de um fragmento no espaço corrente.

$$\delta = \frac{t_{max}}{n} \tag{4.4}$$

O número total de passos n deve ser grande o suficiente, minimizando δ, para que pequenos detalhes na superfície de corte não sejam perdidos durante a busca linear. Dada a posição ao longo do raio $R = (x, y, z)$ a cada passo, avalia-se a profundidade z e a profundidade z_c, extraída do mapa de profundidade utilizando as componentes (x, y) dessa posição:

$$E_{val}(R) = z - z_c \tag{4.5}$$

Caso $E_{val} > 0$, já temos interseção, pois o raio está além da superfície de corte (Figura 4.6). Portanto, prossegue-se com busca binária entre a posição anterior P_{prev} e a posição na qual já ocorreu interseção $P_{post} = R$, fazendo uso da Equação (4.5) para avaliar as posições candidatas a definir um novo intervalo de busca. O funcionamento do algoritmo de busca binária é apresentado através de seu pseudo-código na Tabela 4.1. Em 6 passos, o equivalente a dividir o intervalo inicial em 2^6 partes, obtém-se boa uma aproximação para posição de interseção do raio com a superfície de corte. O processo completo, busca linear e busca binária, está ilustrado na Figura 4.7 para uma superfície arbitrária no espaço da janela de referência. Ao final, os *G-buffers* são escritos com a posição aproximada e a cor, que é acessada a partir dessa posição no mapa de cor da superfície de corte. Entretanto, se o mapa de cor não tiver valores associados a posição aproximada, o fragmento é descartado.

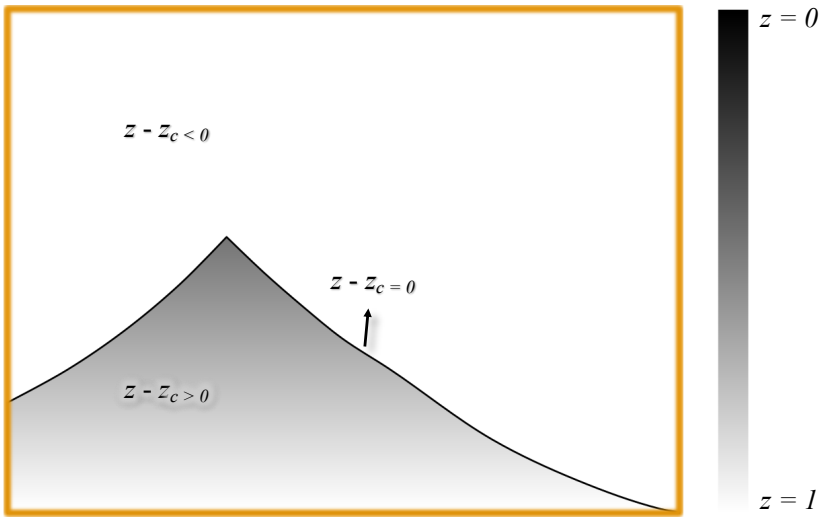


Figura 4.6 – Relação entre a profundidade da superfície de corte z_c, representada por uma linha com cor preta, no espaço da janela e as possíveis profundidades dos fragmentos z que estão abaixo da superfície, região em tons cinza, e acima, região em branco.

Tabela 4.1 – Pseudo-código da busca binária realizada para aproximar a posição da superfície de corte.

```
1.  P_1 ← P_prev
2.  P_2 ← P_post
3.  for i ← 0, i < 6 do
4.     P_query ← (P_1+P_2)/2
5.     if E_val(P_1) * E_val(P_query) ≤ 0 then
6.        P_2 ← P_query
7.     else if E_val(P_2) * E_val(P_query) ≤ 0 then
8.        P_1 ← P_query
9.     else
10.       discard
11.    end if
12. end for
13. return P_query
```

4.5 Aplicando iluminação e desenho dos objetos de interesse

A visualização final é composta de maneira análoga à descrita na Seção 3.5, consumindo os valores associados aos *G-buffers* para aplicar iluminação com base no modelo de iluminação local e componentes ADS, destacar os contornos do corte e, na sequência, desenhar os objetos de interesse. As normais associadas à superfície de corte também são determinadas nesta etapa com base na posição dos fragmentos vizinhos. A Figura 4.8 apresenta o desenho da aproximação da superfície de corte e a Figura 4.9 o resultado final com os objetos de interesse.

Observa-se que, no algoritmo proposto, o corte não é gerado se o observador estiver localizado dentro dos limites do *hull* do modelo, já que a superfície de corte é desenhada realizando traçado de raios somente quando ocorre o descarte de fragmentos do conjunto de objetos secundários. O corte apresentará baixo nível de detalhes quando o modo desacoplado é ativado em uma posição em que o observador esteja muito distante do modelo e, após, aproximar-se da superfície de corte. Isso ocorre devido a resolução inicial do mapa, definida pela quantidade de *pixels* que este estiver ocupando no corte orientado ao observador. Ainda, se ao ativar o modo desacoplado a superfície do corte estiver além dos limites da janela de visualização, não será possível reconstruí-la posteriormente, deixando um espaço vazio pela descontinuidade do corte (Figura 4.10).

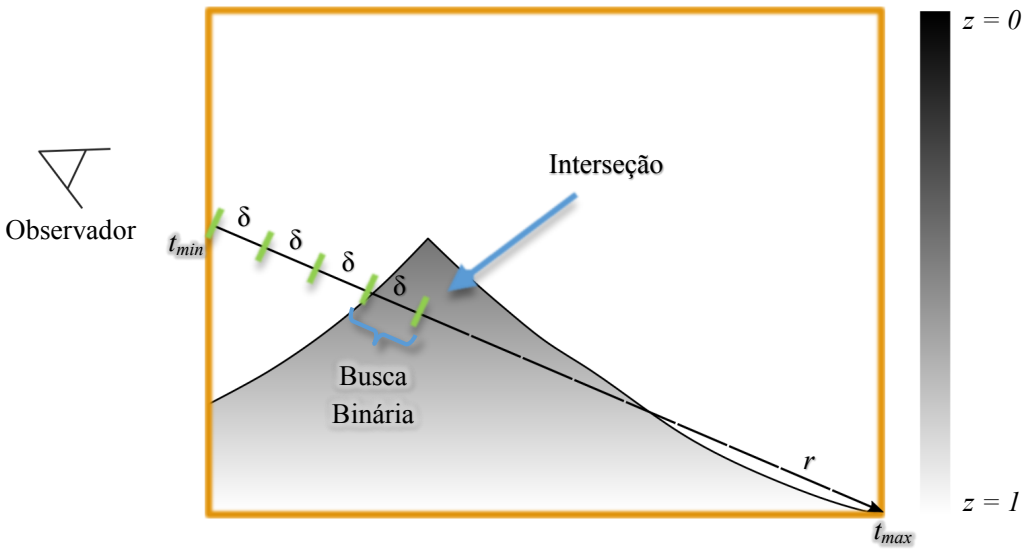


Figura 4.7 – Traçado de raio no espaço da janela de referência com busca linear até a primeira interseção, em destaque, seguido da busca binária.

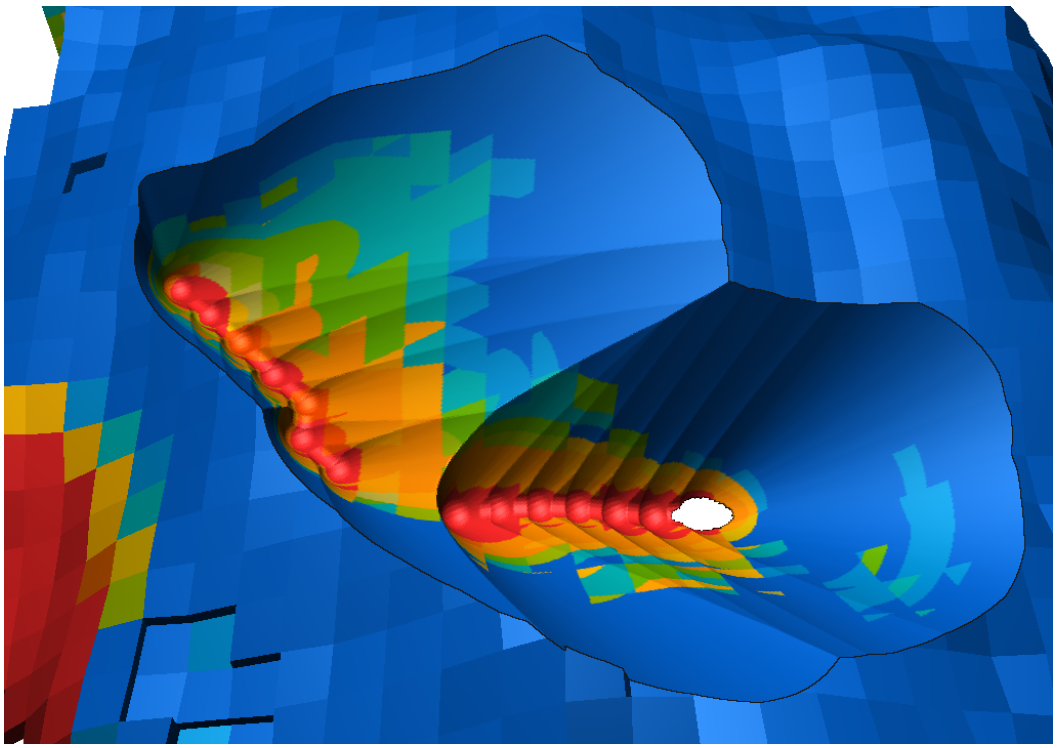


Figura 4.8 – Aproximação da superfície de corte com traçado de raios, efetuando busca linear e busca binária.

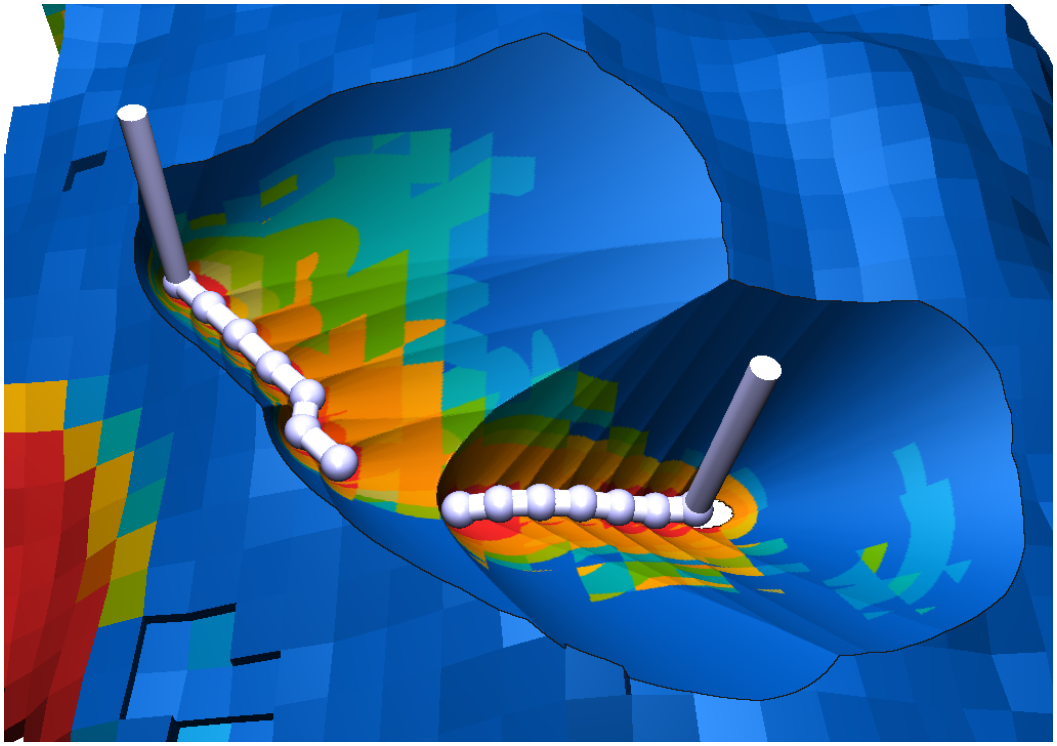
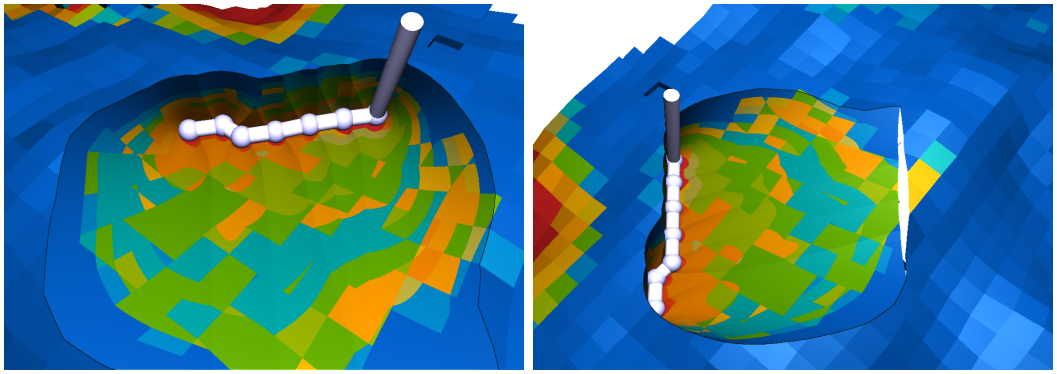


Figura 4.9 – Composição final do corte com desenho dos objetos de interesse.



(a) (b)

Figura 4.10 – (a) Superfície de corte ultrapassando os limites da janela de visualização e (b) descontinuidade na reconstrução do corte a partir de outro ponto de vista.

5
Resultados e discussões

Neste capítulo, são apresentados os resultados experimentais para os algoritmos propostos. Para isso, foi implementado um sistema utilizando a linguagem C++ em conjunto com a API OpenGL e a linguagem de programação em placa gráfica GLSL. A partir desta implementação, foi realizada uma série de experimentos para avaliar a qualidade e o desempenho dos algoritmos, apresentados na Seção 5.1 e na Seção 5.3, respectivamente. Por fim, apresentamos os resultados obtidos para possíveis aplicações, ao definir outros elementos do reservatório, que não poços, como objetos de interesse, na Seção 5.2.

5.1
Qualidade

Visando avaliar a qualidade do algoritmo proposto, cortes foram aplicados em um modelo de reservatório de petróleo com diferentes configurações de poços. Esse modelo é composto por aproximadamente 29 mil células ativas, das quais são extraídas 16 mil faces externas. São utilizadas algumas combinações de poços como objetos de interesse, onde foram escolhidos números variados de poços injetores ou de poços produtores. Com o objetivo de facilitar a distinção entre os poços e a escala de cor escolhida para representar a intensidade das propriedades associadas às células do modelo, os poços injetores são desenhados em um tom azul claro, enquanto os poços produtores em um tom vermelho claro. Para a apresentação desses resultados, foram geradas imagens com resolução de 1024×720 *pixels*. Em alguns casos, a escala de cor foi omitida para não interferir na visualização do corte. As imagens obtidas para os cortes orientados ao observador são apresentados na Seção 5.1.1 e para os cortes desacoplados do observador na Seção 5.1.2.

5.1.1
Cortes orientados ao observador

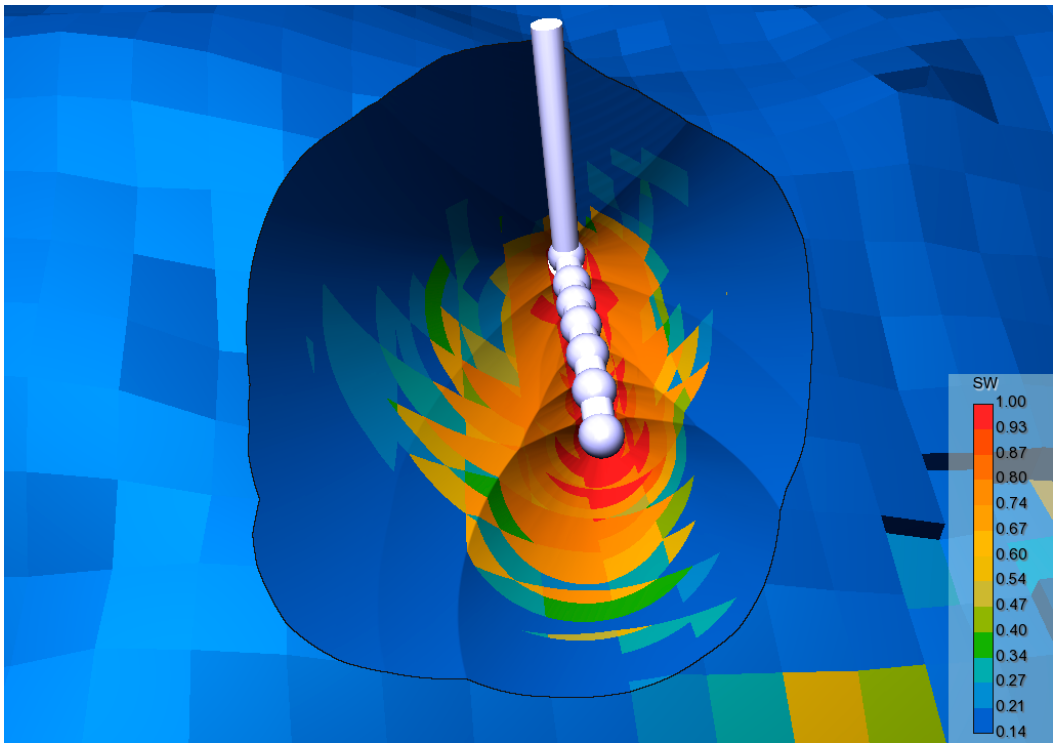
A seguir são apresentados cortes orientados ao observador aplicados a modelos em dois passos de tempo sequenciais. O objetivo é validar a qualidade

do algoritmo de corte e destacar a variação das propriedades associadas as células no entorno dos objetos de interesse.

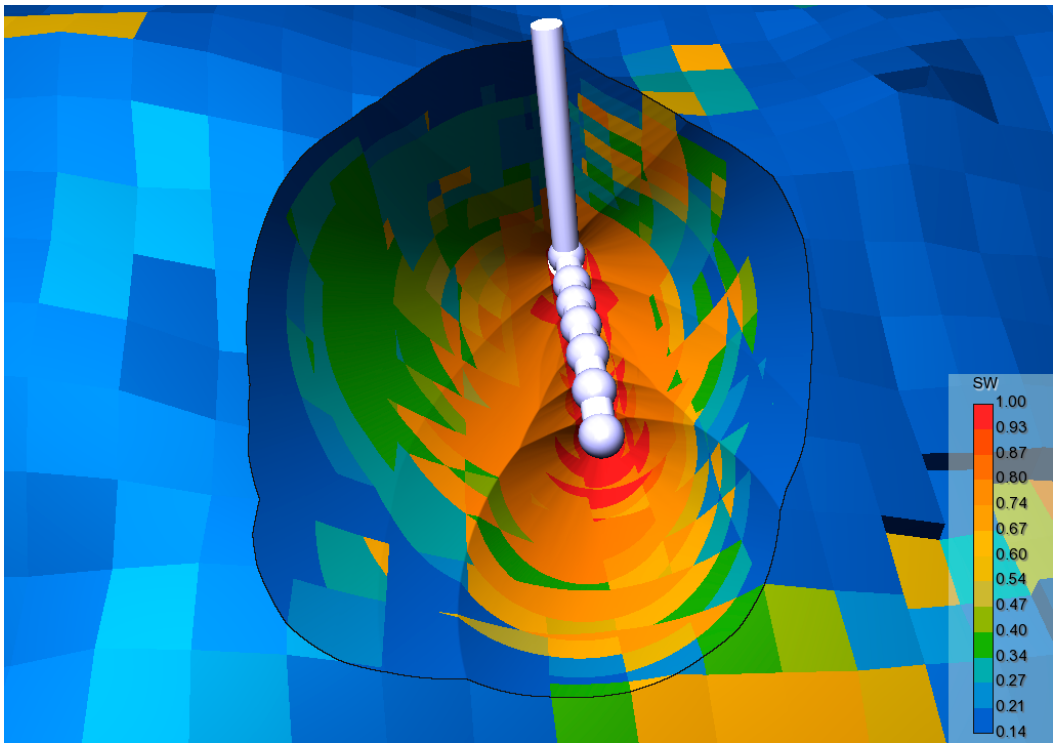
Na Figura 5.1, podemos observar o aumento da saturação de água (SW) nas proximidades de um poço injetor, definido como objeto de interesse, em um passo de tempo anterior (Figura 5.1a) e posterior (Figura 5.1b). Células cortadas são corretamente desenhadas, como na Figura 5.1b, observável na continuidade das mesmas em regiões de fronteira com a superfície de corte. Também é importante notar como a posição das fontes de luz auxilia na interpretação do corte.

Um resultado semelhante, ao descrito acima, é obtido ao definir o conjunto de objetos de interesse com múltiplos poços injetores, na Figura 5.2. Da mesma forma, próximos a poços produtores, definidos como objetos de interesse, observa-se o decréscimo da saturação de óleo (SO) em seu entorno, nas Figuras 5.3 e 5.4. Principalmente para os casos com múltiplos poços, é perceptível como a superfície de corte delimita a região de influência dos objetos individuais, o que acaba minimizando a quantidade de contexto removida.

As Figuras 5.5 a 5.8 apresentam o desenho do aramado das faces externas do modelo e dos horizontes da superfície de corte, delimitando o espaço de cada célula e destacando as falhas geológicas presentes, respectivamente, para os cortes das Figuras 5.1b, 5.2b, 5.3b e 5.4b.

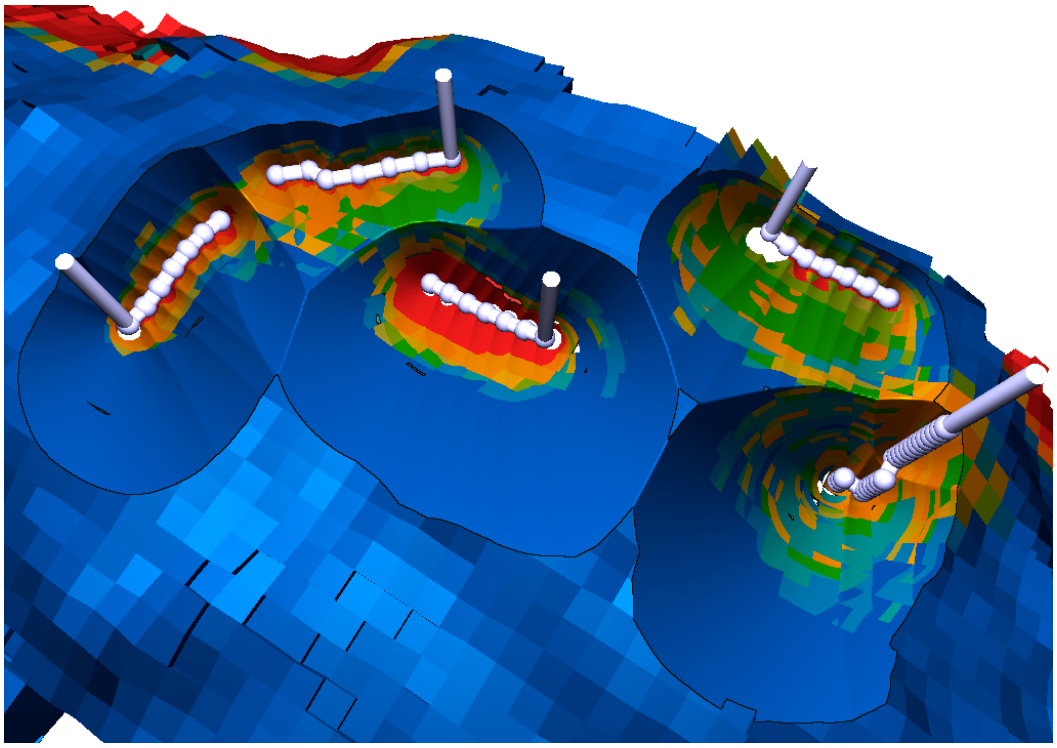


(a)

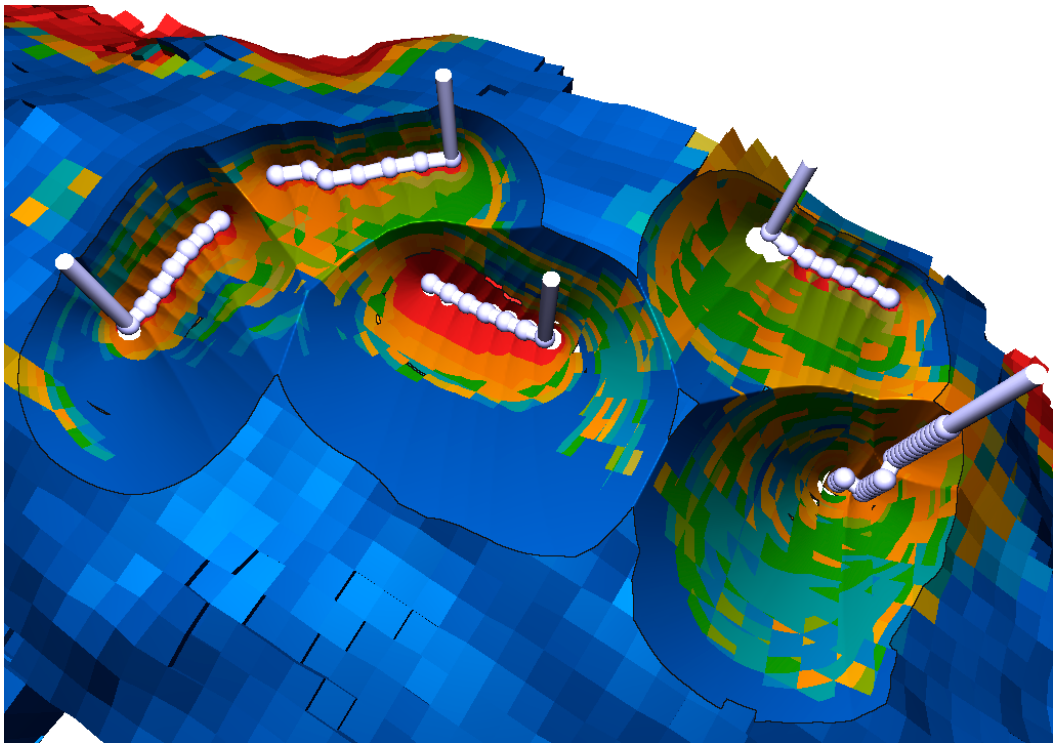


(b)

Figura 5.1 – Corte aplicado a um modelo de reservatório, com um poço injetor definido como objeto de interesse, em dois passos de tempo diferentes.

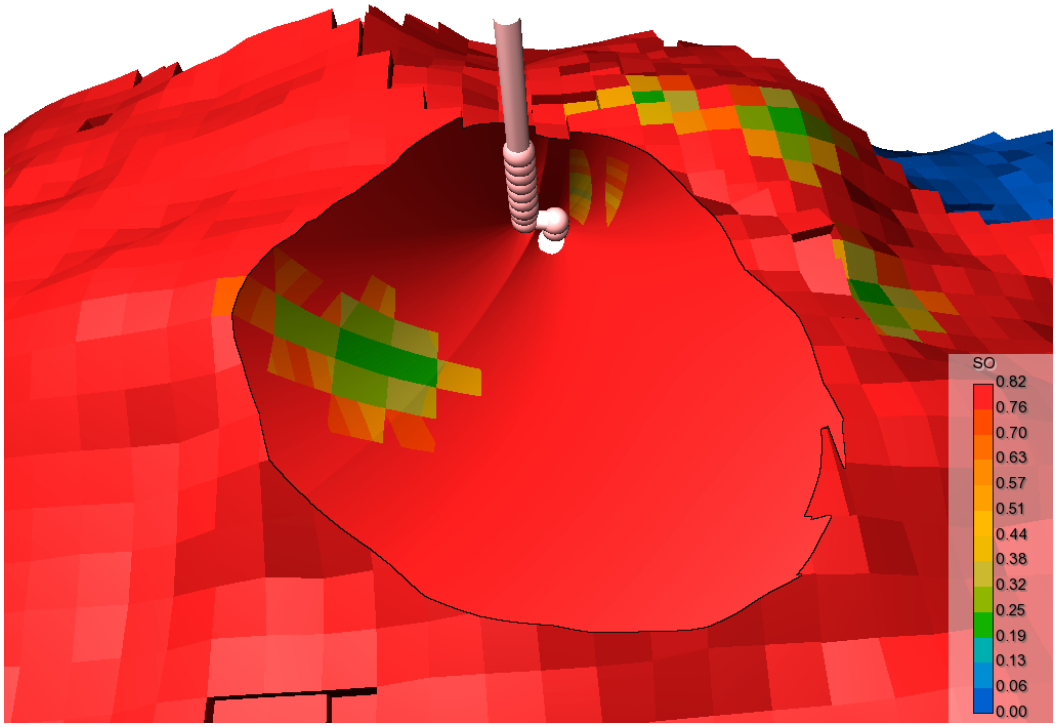


(a)

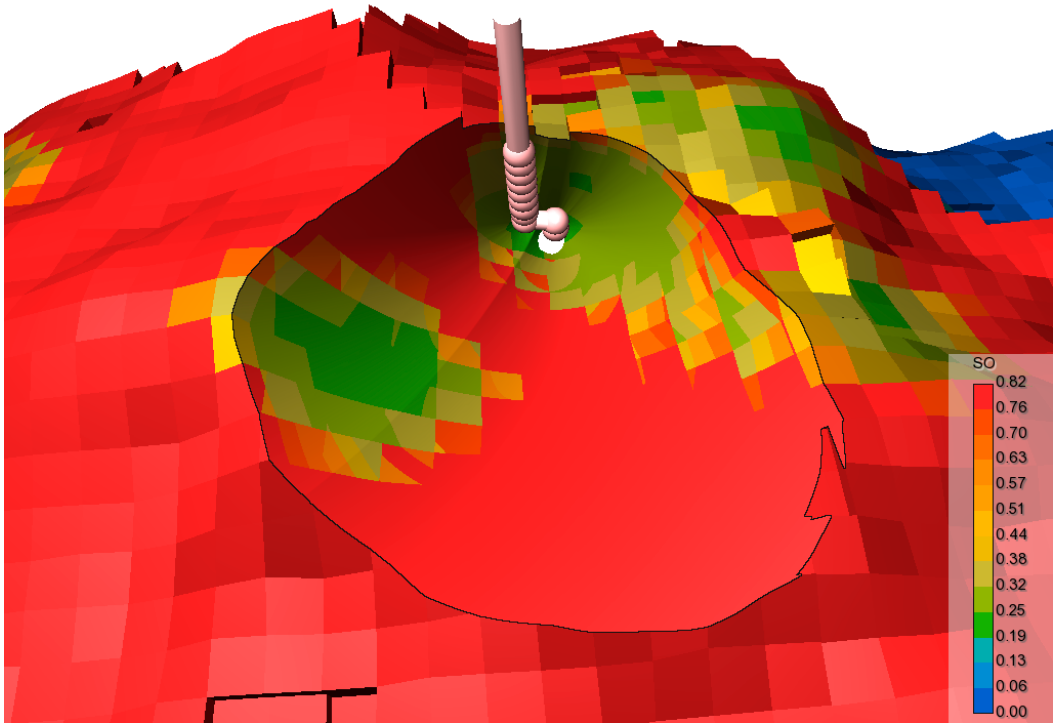


(b)

Figura 5.2 – Corte aplicado a um modelo de reservatório de petróleo em dois passos de tempo, com a propriedade de saturação de água (SW) associada as células, e múltiplos poços injetores definidos como objetos de interesse.

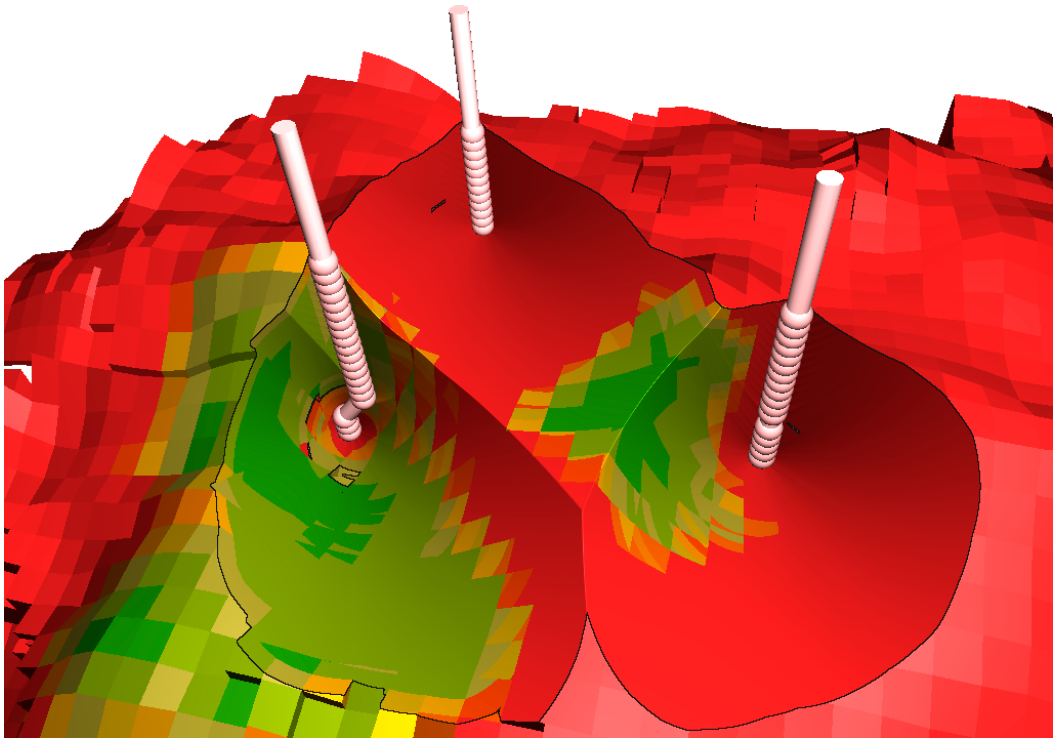


(a)

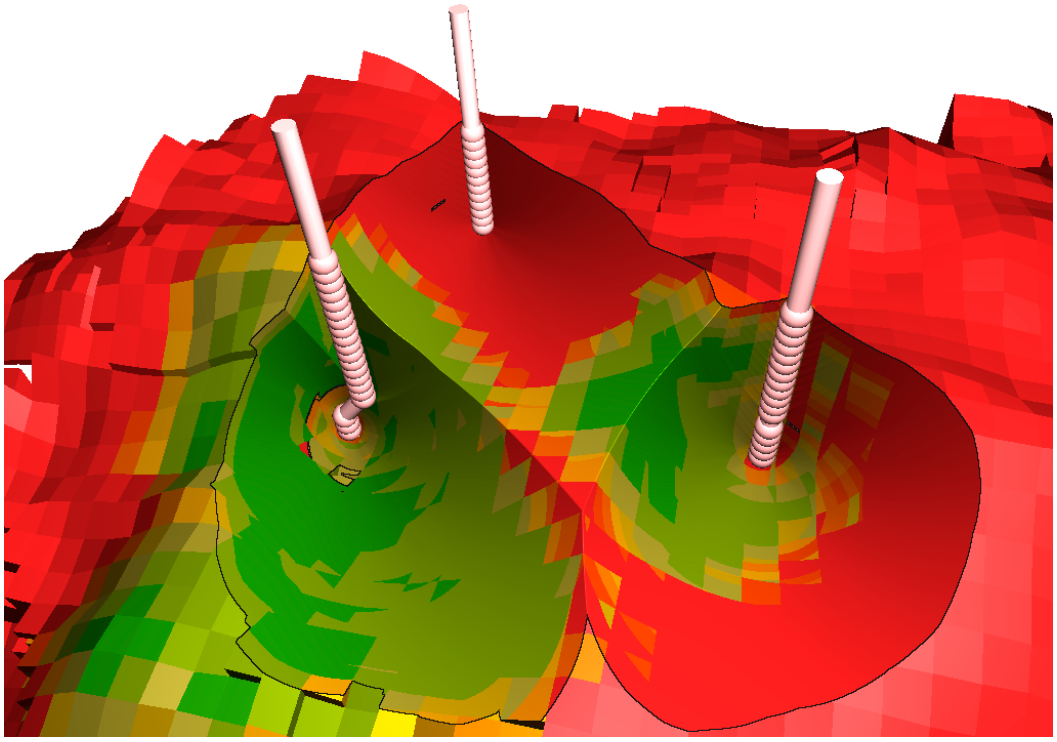


(b)

Figura 5.3 – Corte aplicado a um modelo de reservatório de petróleo em dois passos de tempo, com a propriedade de saturação de óleo (SO) associada as células, e um poço produtor definido como objeto de interesse.



(a)



(b)

Figura 5.4 – Corte aplicado a um modelo de reservatório de petróleo em dois passos de tempo, com a propriedade de saturação de óleo (SO) associada as células, e três poços produtores definidos como objetos de interesse.

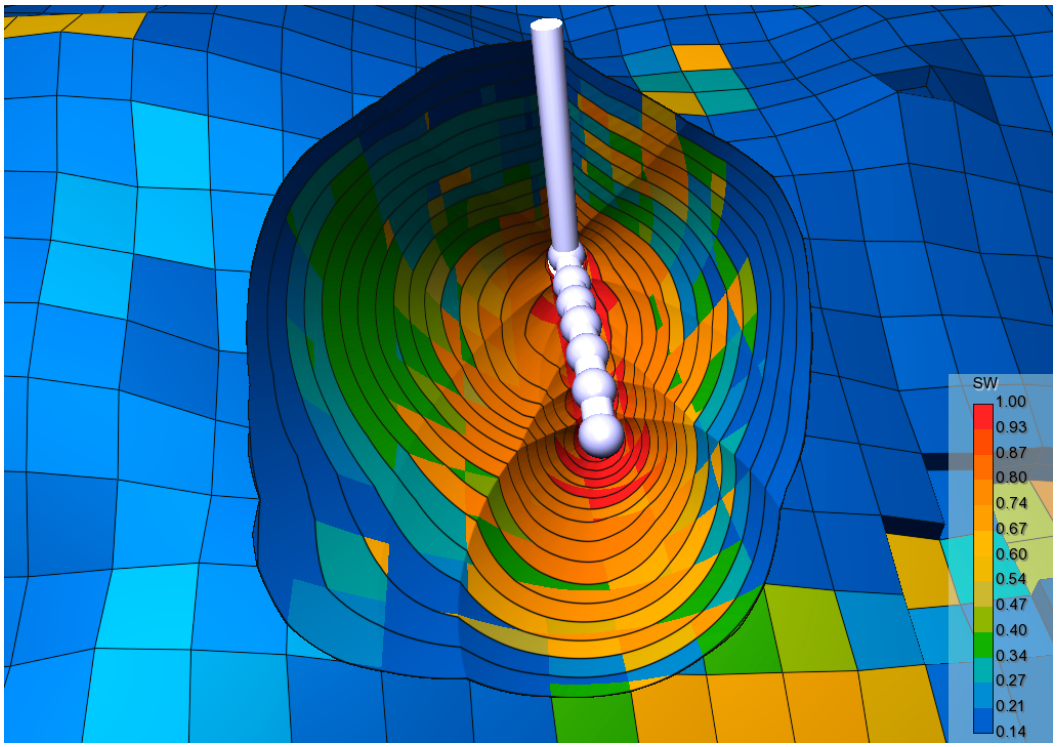


Figura 5.5 – Mesmo corte da Figura 5.1b renderizado com aramado nas faces externas e os horizontes na superfície de corte.

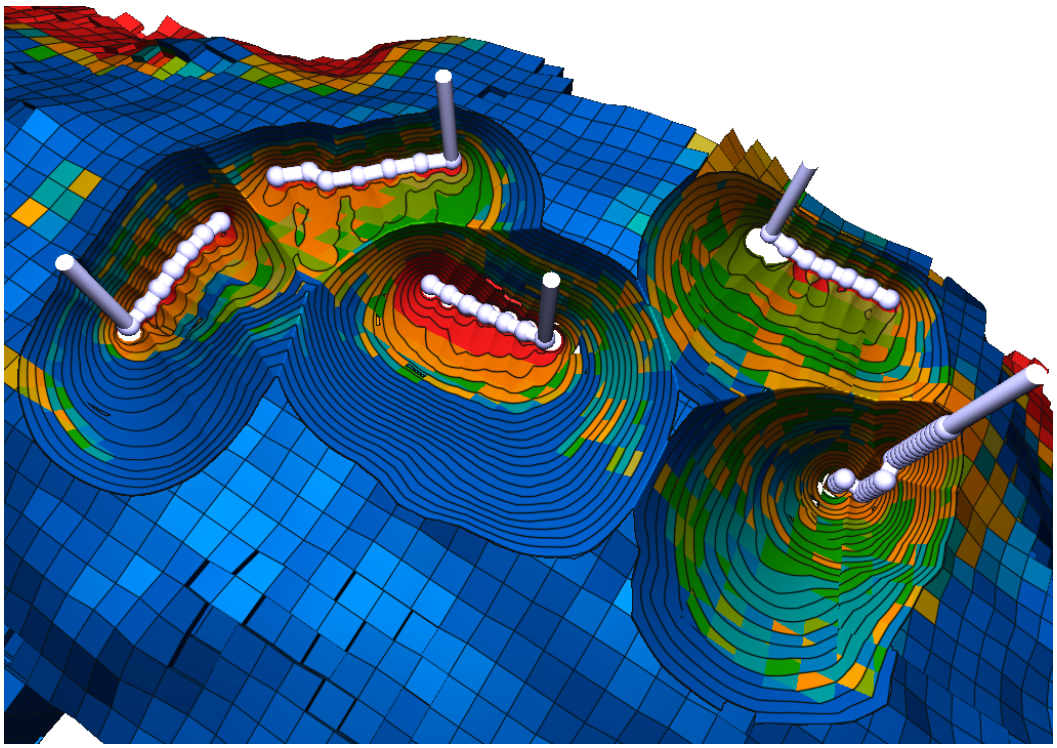


Figura 5.6 – Mesmo corte da Figura 5.2b renderizado com aramado nas faces externas e os horizontes na superfície de corte.

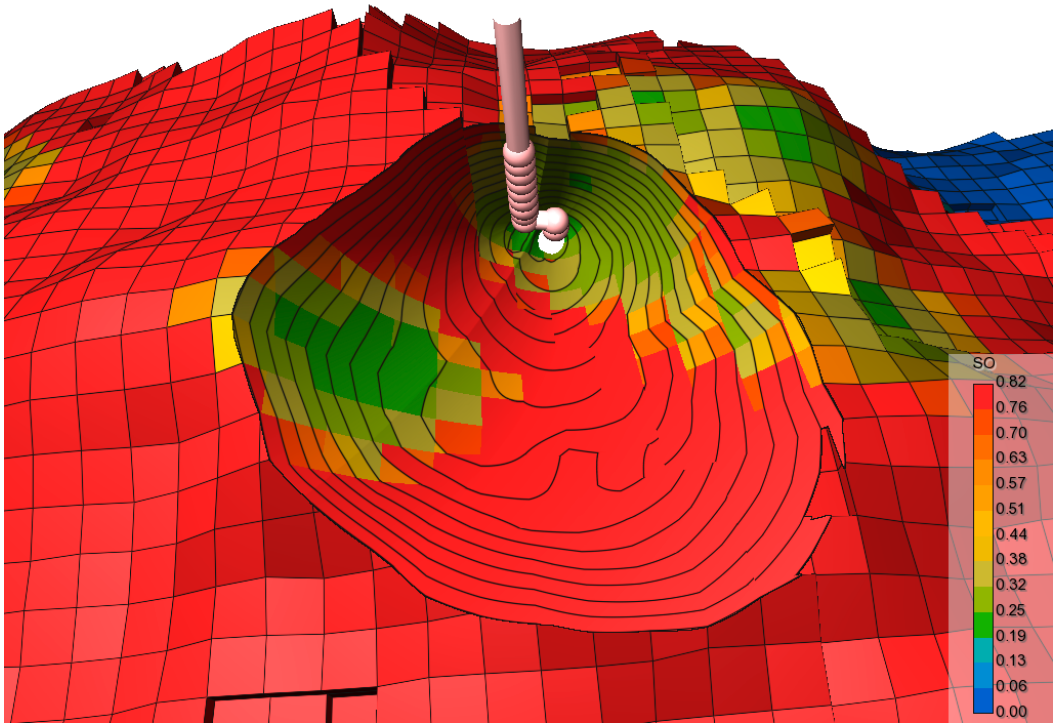


Figura 5.7 – Mesmo corte da Figura 5.3b renderizado com aramado nas faces externas e os horizontes na superfície de corte.

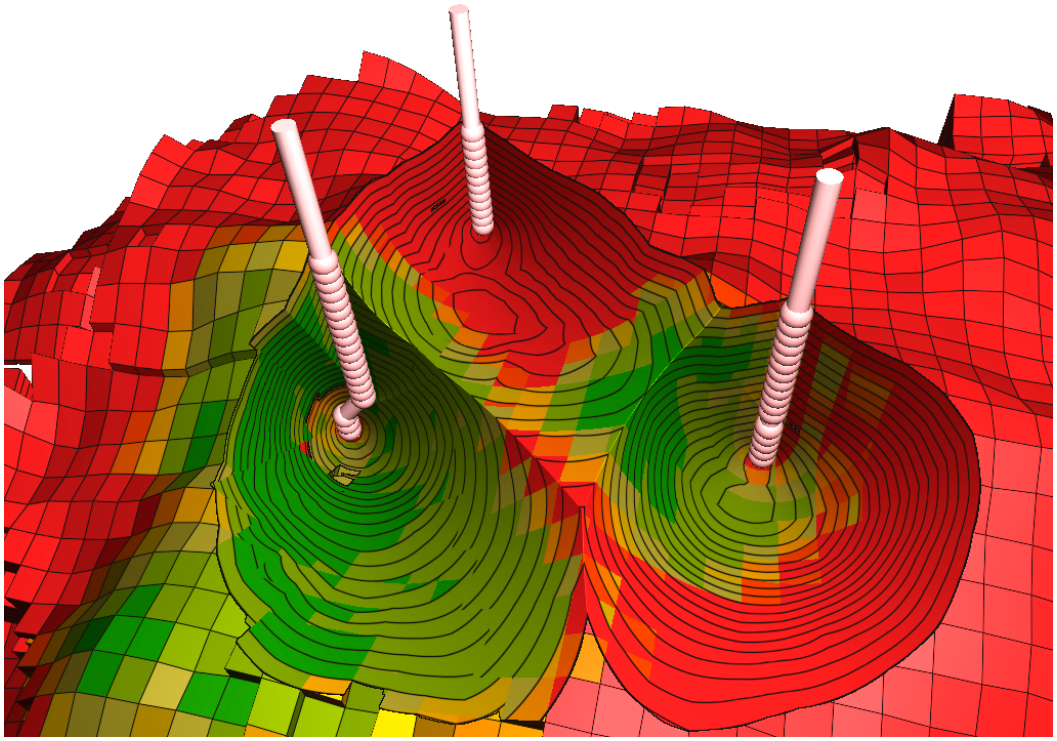


Figura 5.8 – Mesmo corte da Figura 5.4b renderizado com aramado nas faces externas e os horizontes na superfície de corte.

5.1.2
Corte desacoplado do observador

Para validar a qualidade do algoritmo de corte desacoplado do observador, primeiro são apresentadas imagens com o corte orientado ao observador e, na sequência, o mesmo corte desacoplado, visto de outro ponto de vista.

Na Figura 5.9, observa-se um corte com múltiplos poços injetores como objetos de interesse e com a propriedade de saturação de água associada às células. Nas Figuras 5.10a e 5.10b, são apresentadas duas possíveis interações, após desacoplar o corte, aproximando-se a câmera para inspecionar o entorno dos poços em questão. Observa-se que como o corte foi gerado em um ponto distante da posição da câmera atual, o corte mantém qualidade para uma correta interpretação, mesmo que alguns serrilhados possam ser observados nos limites de células na superfície de corte, resultado da aproximação da posição da superfície de corte.

A propriedade de saturação de óleo é inspecionada através de um corte de referência gerado para um conjunto de poços produtores como objeto de interesse na Figura 5.11a e a visualização do corte desacoplado na Figura 5.11b. Nesse caso, o desenho do aramado está ativo somente para as faces externas do modelo.

Por fim, na Figura 5.12a, um corte é gerado próximo aos objetos de interesse, com o desenho do aramado ativo para as faces externas do modelo e para superfície de corte. Na sequência, Figura 5.12b, a câmera é afastada, permitindo a inspeção do corte em um contexto geral.

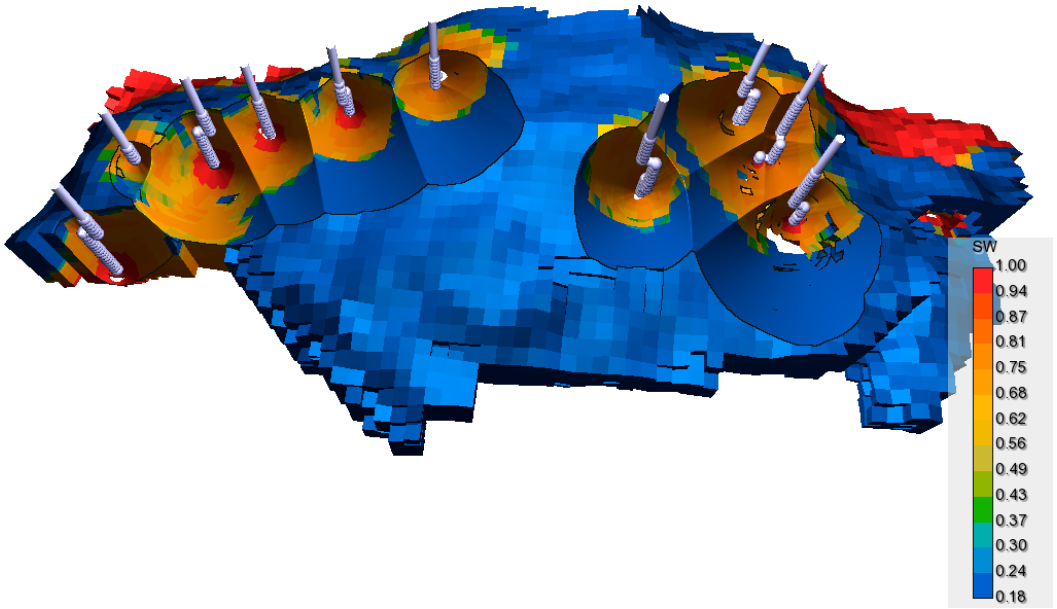
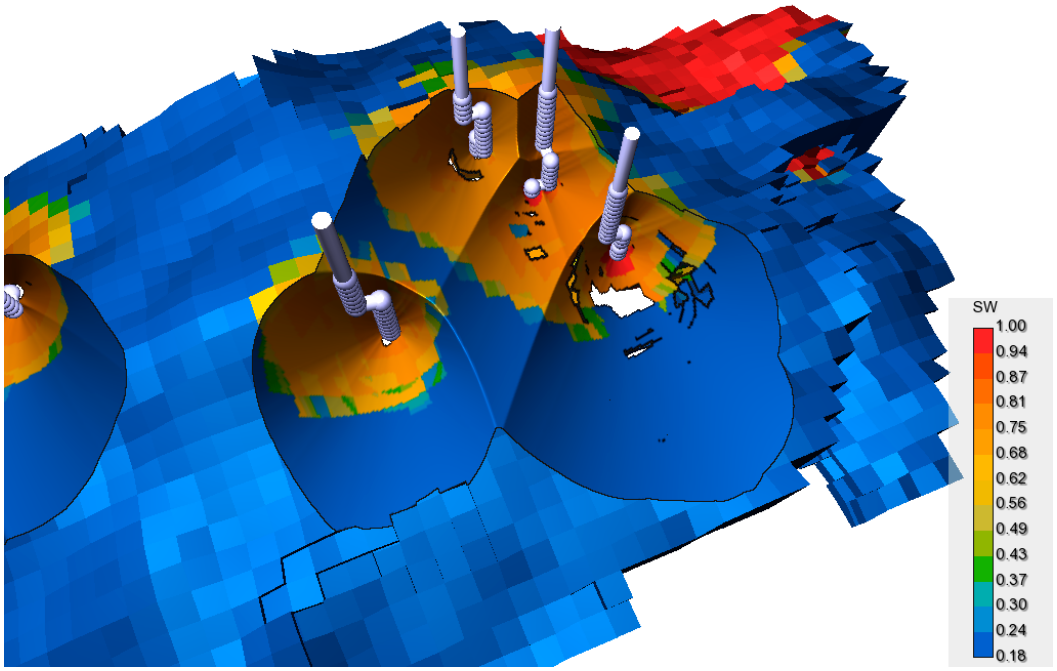
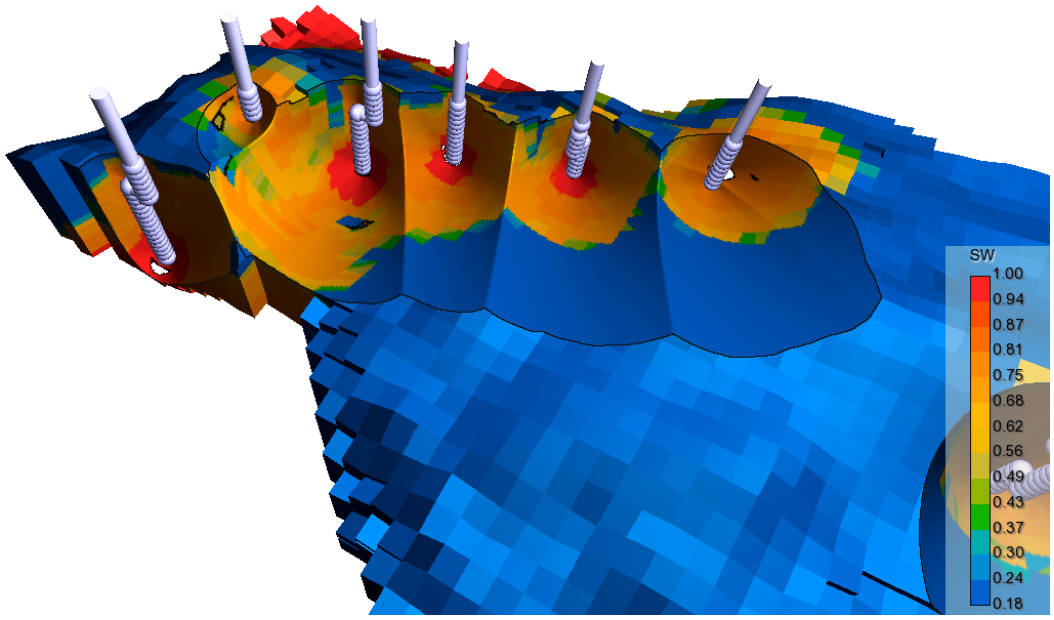


Figura 5.9 – Corte orientado ao observador com poços injetores definidos como objetos de interesse.

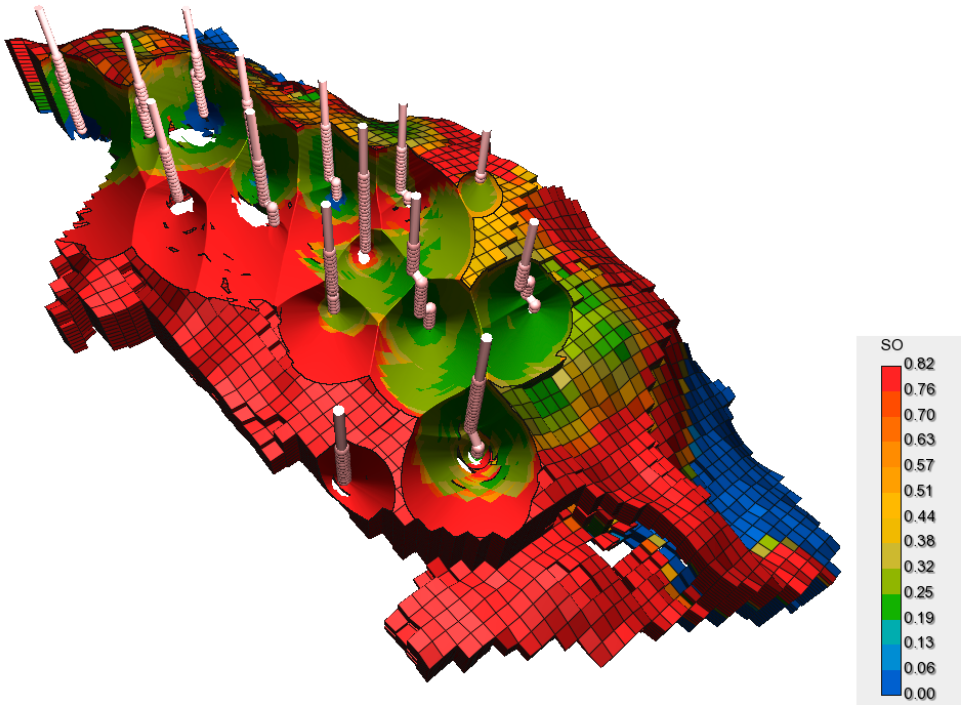


(a)

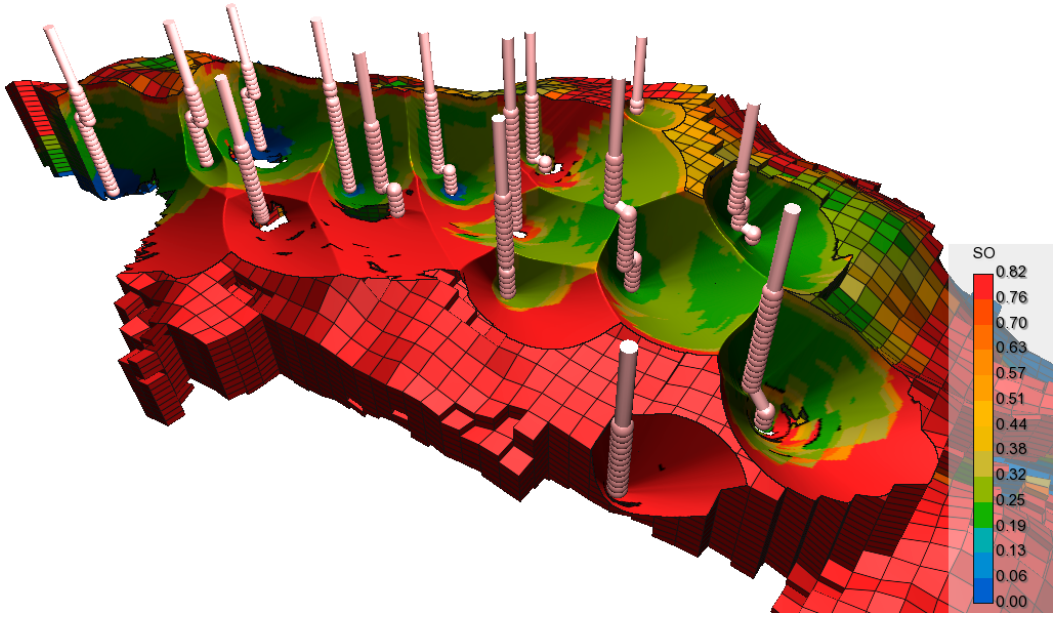


(b)

Figura 5.10 – Visualização dos objetores de interesse a partir de diferentes posições desacoplando o corte gerado na Figura 5.9.

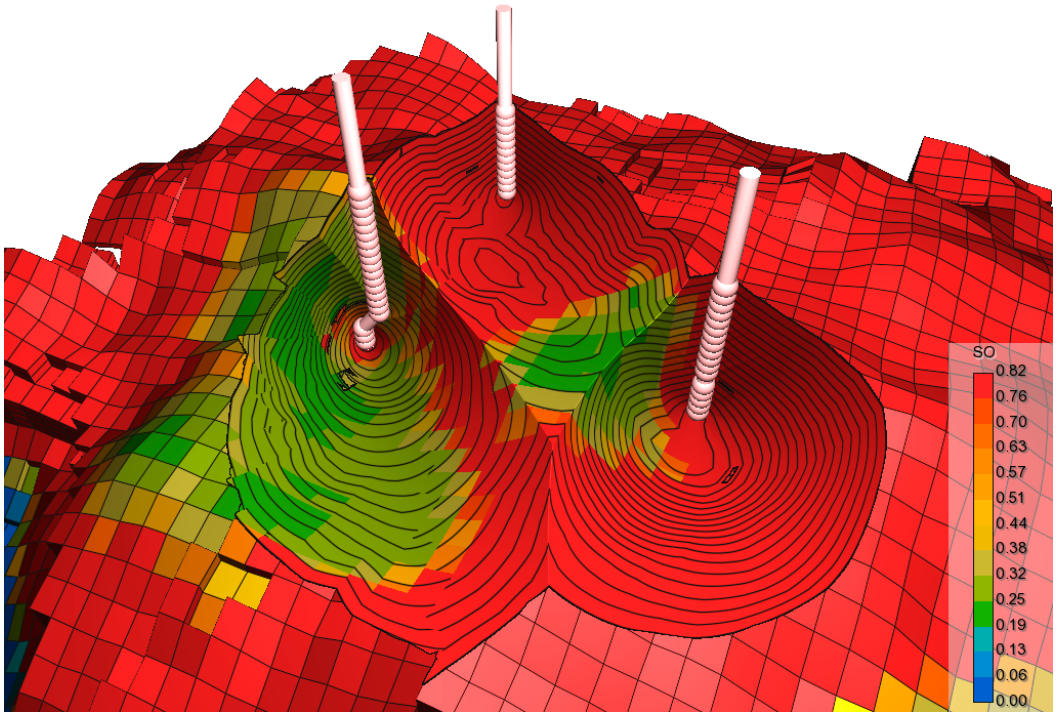


(a)

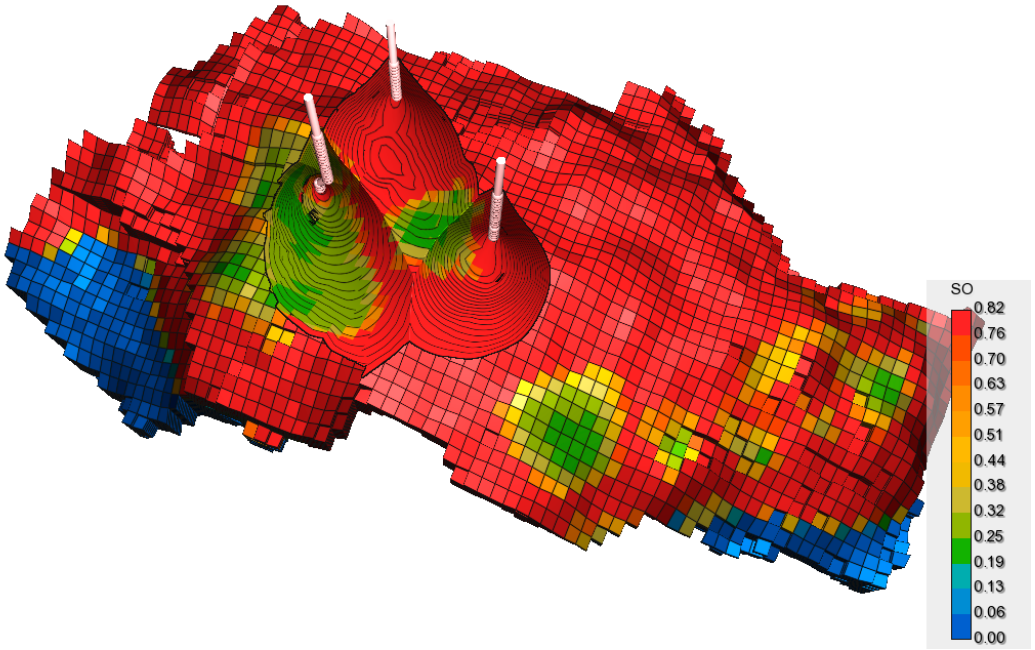


(b)

Figura 5.11 – (a) Corte orientado ao observador com poços produtores como objetos de interesse e (b) corte desacoplado.



(a)



(b)

Figura 5.12 – (a) Corte orientado ao observador com poços produtores como objetos de interesse e (b) corte desacoplado.

5.2
Outros objetos de interesse

Os algoritmos propostos podem ser naturalmente integrados a ambientes de produção. Os dados entre diferentes ferramentas podem ser relacionadas, através de, por exemplo, *brushing and linking*, para uma inspeção mais completa. Nesta seção, são apresentadas, brevemente, resultados para duas possibilidades de extensão, realizando a troca dos objetos de interesse por células ou por linhas de fluxo.

Células podem ser selecionadas através de filtros que definem, para uma propriedade, um valor mínimo e máximo. Qualquer célula nesse intervalo é considerada um objeto de interesse. De maneira similar, a seleção pode ocorrer para seus índices (i, j, k), delimitados por intervalos em cada uma das coordenadas. Ambas seleções podem ser resultados de *brushing and linking* com, por exemplo, gráficos XY. Na Figura 5.13, é apresentando o corte obtido, cujas células com a propriedade SW e valores entre 0,83 e 0,91 são definidas como objetos de interesse. Na Figura 5.14, outro corte é produzido a partir de um subconjunto de células, selecionadas com base em seu índices.

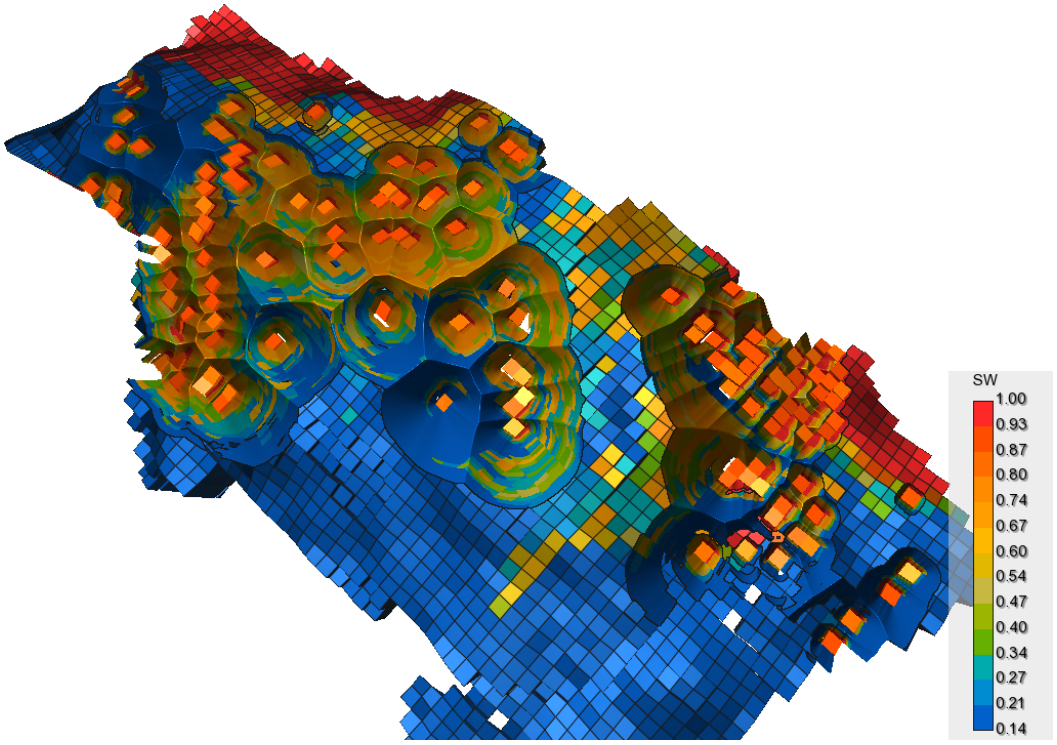


Figura 5.13 – Corte gerados com base em um filtro de propriedades, selecionando subconjuntos de células como objetos de interesse.

Linhas de fluxo ilustram o comportamento do fluído no interior do reservatório, sendo mapeadas conforme as necessidades do usuário, por exemplo, com as cores das propriedades das células em seu caminho, com diferentes propriedades ou com cores aleatórias para facilitar a identificação de cada linha. Utilizando essa última para realizar o desenho das linhas de fluxo, na Figura 5.15, é apresentado o corte obtido definindo como objetos de interesse as linhas que representam o fluxo entre um poço produtor e dois poços injetores.

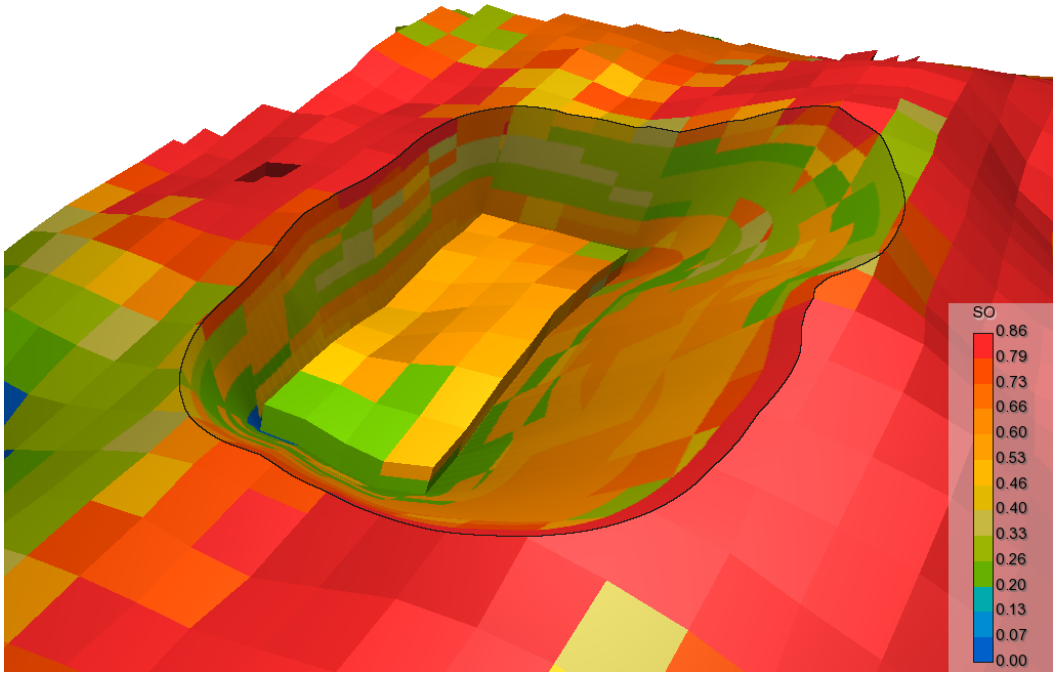
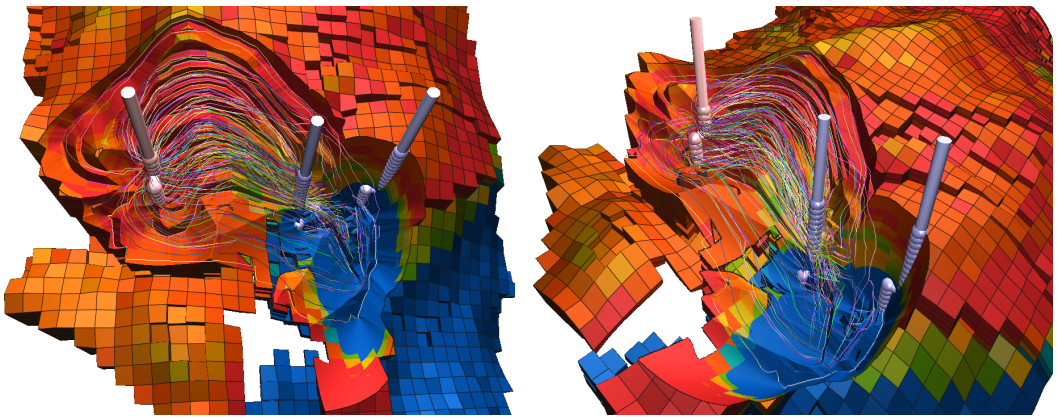


Figura 5.14 – Cortes gerados com um filtro baseado em índices, selecionando subconjuntos de células como objetos de interesse.



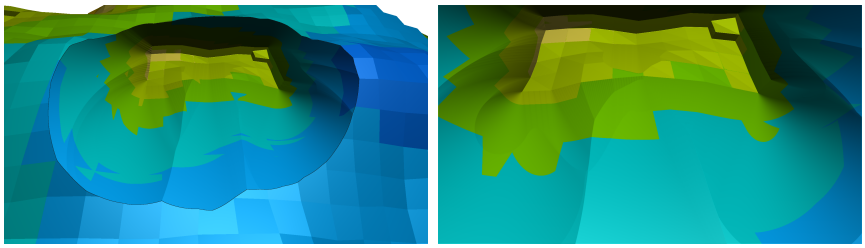
(a) (b)

Figura 5.15 – Linhas representando o fluxo entre um poço produtor e dois poços injetores em (a) um corte orientado ao observador e (b) o mesmo corte desacoplado.

5.3 Desempenho

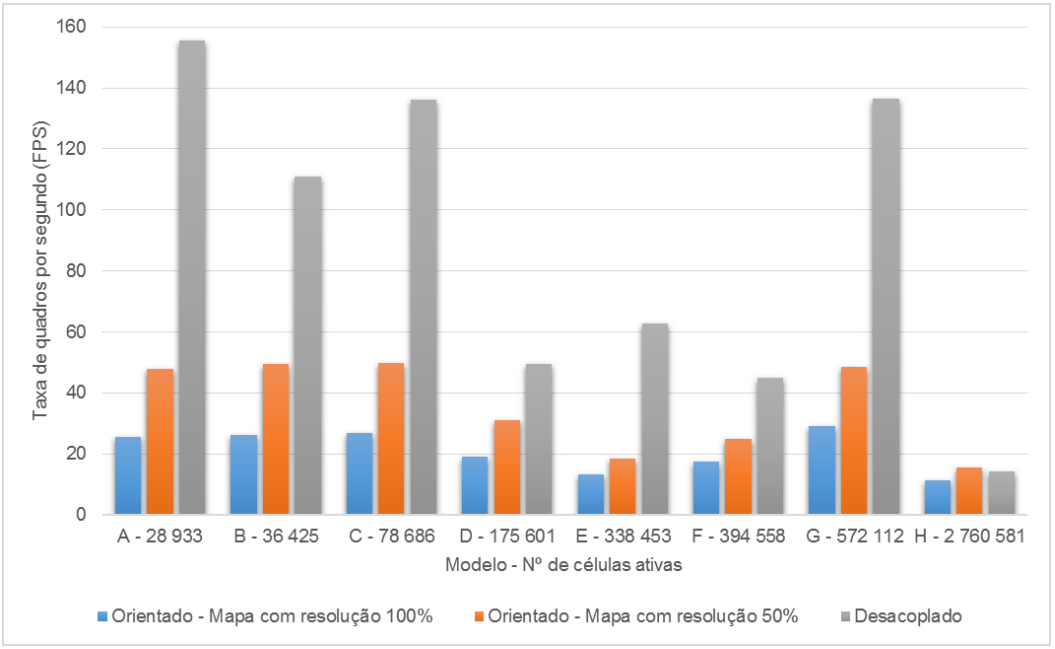
Esta seção apresenta a análise dos experimentos realizados para testar o desempenho dos algoritmos propostos. Os experimentos foram realizados em um processador Intel $i5$-4440 de $3.1GHz$ com $8GB$ de memória RAM-DDR3, utilizando uma placa de vídeo NVIDIA GeForce GTX 760 com $2GB$ de VRAM. São testados 8 modelos, nomeados de A a H, com total de células ativas variando de 29 mil a 572 mil, e o último com $2,76$ milhões de células ativas. Alguns padrões são definidos para todos os experimentos: a janela de visualização possui dimensões de 1920×1080, a grade regular é construída com um fator de resolução $k = 3$, a superfície de corte e os objetos secundários são renderizados com aramado, utiliza-se uma propriedade comum a todos os modelos, os objetos de interesse são definidos por um subconjunto de células e todas medidas de tempo são representadas por suas médias.

Inicialmente, na Figura 5.17, são apresentados resultados para cortes em dois gráficos separados, nos quais o desenhado da superfície de corte ocupa, respectivamente, 50% e 100% da área definida pelas dimensões da janela ativa. Na prática, o primeiro deve representar o caso médio de uso dos algoritmos de corte (Figura 5.16a), e o segundo representa o pior caso para desempenho dos algoritmos (Figura 5.16b). Em cada gráfico, são comparados os resultados relacionados a taxa de quadros por segundo (FPS) de cada algoritmo para os modelos utilizados e a contagem de células ativas. Os algoritmos de corte estão organizados em três colunas por modelo e foram executados a partir do mesmo ponto de vista. A primeira coluna representa o algoritmo de corte orientado ao observador, no qual o mapa de profundidade da superfície de corte, gerado na segunda etapa do algoritmo, tem resolução igual 100% das dimensões da janela. A segunda coluna também representa o algoritmo de corte orientado ao observador, porém, o mapa tem resolução igual a 50% das dimensões da janela. A terceira coluna representa o algoritmo de corte desacoplado do observador, gerado a partir da superfície de corte com resolução igual a 100%.

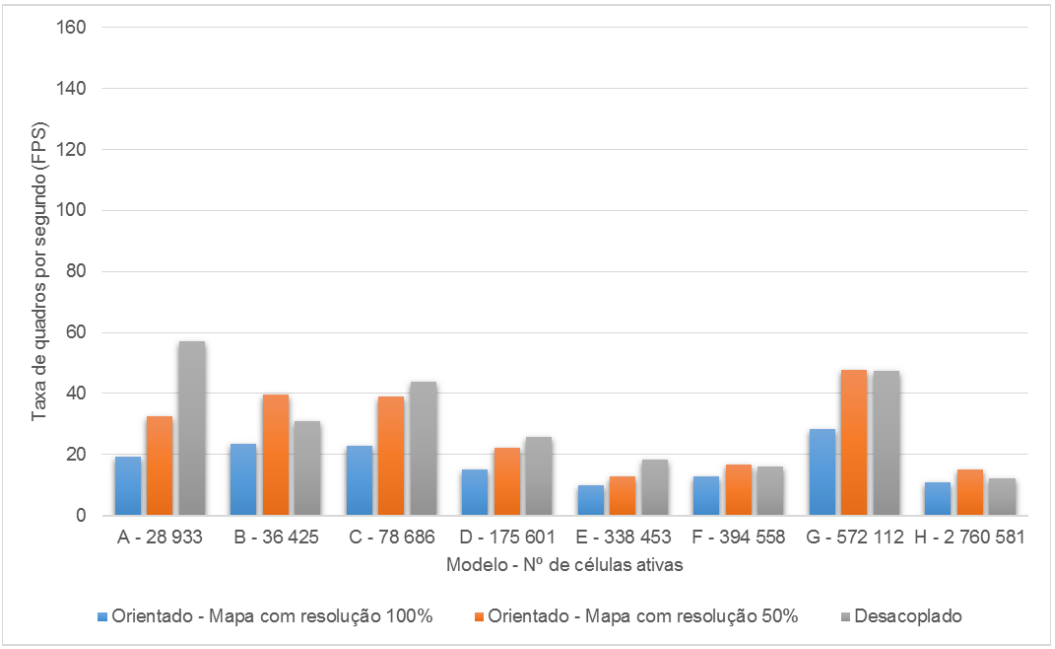


(a) (b)

Figura 5.16 – Cortes gerados com a superfície de corte ocupando (a) 50% e (b) 100% da janela ativa.



(a)



(b)

Figura 5.17 – Comparação das taxas de quadro por segundo de 8 modelos diferentes. Para cada modelo, são testados 3 algoritmos. Em (a) os cortes ocupam 50% da janela e em (b) ocupam 100% da janela.

Com os resultados da Figura 5.17, fica explícito o ganho de desempenho proporcionado pelo uso do mapa de profundidades com menor resolução. O desempenho dos algoritmos de corte para o caso médio está acima de 20 FPS para a maioria dos modelos, estando abaixo desse limite nos três algoritmos somente para o modelo H. Observa-se que, ainda que haja alguma relação entre a quantidade de células ativas de cada modelo e a taxa de quadros, esta não é o fator dominante, já que o modelo G, com 572 mil células ativas, apresenta desempenho semelhante ao do modelo C, com 78 mil células ativas. Uma breve comparação de tempo para a etapa que gera a superfície de corte, utilizando um mapa de profundidade com resolução igual a 100% e 50% das dimensões da janela, é apresentado na Figura 5.18. As medidas de tempo são menores e mais estáveis no mapa gerado com resolução igual a 50%.

Ocorre uma boa diferença na taxa de quadros do algoritmo de corte desacoplado, quando comparado com os dois primeiros, na Figura 5.17a, para a maioria dos modelos. O desempenho desse algoritmo está relacionado ao desenho das faces externas e a quantidade de traçados raio realizados, os quais ocorrem sempre que fragmentos do modelo são descartados pela comparação de profundidade com a superfície de corte. Portanto, para os resultados da Figura 5.17b, nos quais a quantidade de fragmentos descartados é aproximadamente o dobro do que para os cortes gerados na Figura 5.17a, a diferença na taxa de quadros desse algoritmo com os outros tende a ser pequena, quando não mais cara, devido à quantidade de traçados de raio efetuados.

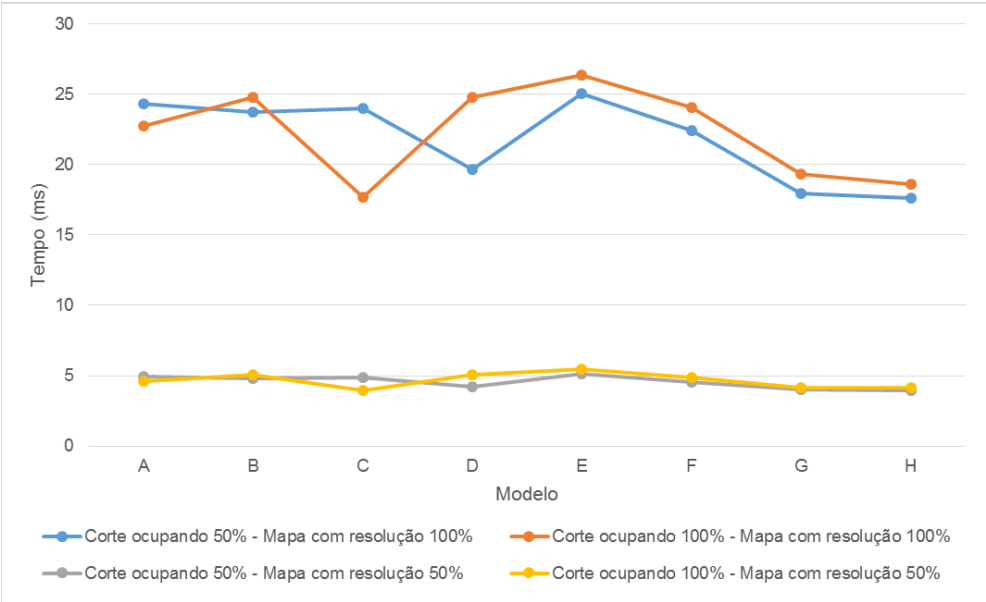


Figura 5.18 – Comparação entre o custo para gerar a superfície de corte variando as dimensões da superfície de corte e variando a área que o corte ocupa na visualização, com base nas dimensões da janela.

Nas Tabelas 5.1 e 5.2, são apresentadas as medidas de tempo para cada etapa do algoritmo de corte orientado ao observador na renderização de um quadro, com a intenção de esclarecer a relação entre essas etapas e os dados dos modelos. As medidas de tempo foram extraídas de um corte cuja área ocupa toda a janela, utilizando um mapa de profundidade com as dimensões da janela e assumindo os padrões definidos no parágrafo inicial deste capítulo. Essas etapas, descritas no Capítulo 3, são:

1. Desenhar as *back-faces* dos objetos de interesse (OI)
2. Computar a superfície de corte
3. Desenhar o modelo com descarte de fragmentos
4. Desenhar a superfície de corte no modelo
5. Aplicar iluminação
6. Desenhar as *front-faces* dos objetos de interesse

Analisando as Tabelas 5.1 e 5.2, está claro que as etapas 1, 5 e 6 resultam em pequeno impacto no algoritmo, mantendo-se estável para todos modelos, com medidas abaixo de 3% do tempo total. Nota-se a discrepância entre o total de células ativas e o total de faces externas extraídas em cada modelos. Para melhor comparar o resultado das outras etapas, é produzido um gráfico, na Figura 5.19, com base nos valores associados à essa tabela.

No gráfico da Figura 5.19, são comparados os tempos das etapas 2, 3 e 4, e da soma total de todas etapas. O eixo-x é ordenado com base no número de faces externas de cada modelo. Observa-se que a etapa 2 mantém-se constante, já que seu desempenho depende, principalmente, das dimensões da janela. O custo da etapa 3, o desenho das faces externas do modelo, tende a crescer com a quantidade das mesmas, estando relacionada com o número de primitivas enviadas para placa gráfica. Já o custo associado a etapa 4, a determinação da interseção de cada fragmento da superfície de corte com o modelo, é mais complexa de ser avaliada, pois depende da localização do fragmento na grade regular.

Tabela 5.1 – Comparação de desempenho de cada etapa do algoritmo de corte orientado ao observador. As medidas de tempo são expressas em porcentagem e o tempo total em milissegundos.

Modelo		A	B	C	D	E	F	G	H
Células ativas		28 933	36 425	78 686	175 601	338 453	394 558	572 112	2 760 581
Faces externas		16 492	24 652	60 356	139 278	361 118	267 752	45 342	480 998
1. Back-faces	(%)	0,28	0,43	0,10	0,26	0,55	0,42	0,13	0,04
2. JFA	(%)	48,28	55,76	43,46	34,80	25,03	33,95	50,65	16,70
3. Modelo	(%)	4,99	11,04	6,89	28,01	12,82	16,42	5,33	49,94
4. Interseção	(%)	44,14	30,22	46,93	35,37	60,46	47,50	41,07	32,32
5. Def. Shading	(%)	2,26	2,46	2,59	1,51	1,05	1,64	2,79	0,99
6. Obj. de Int.	(%)	0,06	0,09	0,02	0,05	0,09	0,06	0,02	0,01
Tempo total	(ms)	47,16	44,39	40,74	71,29	105,38	70,89	38,19	111,39

Tabela 5.2 – Comparação de desempenho de cada etapa do algoritmo de corte orientado ao observador. As medidas de tempos são expressos em milissegundos.

Modelo	A	B	C	D	E	F	G	H
Células Ativas	28 933	36 425	78 686	175 601	338 453	394 558	572 112	2 760 581
Faces Externas	16 492	24 652	60 356	139 278	361 118	267 752	45 342	480 998
1. Back-faces	0,13	0,19	0,04	0,19	0,58	0,30	0,05	0,04
2. JFA	22,77	24,75	17,71	24,81	26,38	24,07	19,34	18,60
3. Modelo	2,36	4,90	2,81	19,97	13,51	11,64	2,04	55,63
4. Interseção	20,81	13,42	19,12	25,21	63,72	33,67	15,68	36,01
5. Def. Shading	1,07	1,09	1,06	1,08	1,15	1,16	1,07	1,11
6. Obj. de Int.	0,03	0,04	0,01	0,04	0,09	0,05	0,01	0,02
Total	47,16	44,39	40,74	71,29	105,38	70,89	38,19	111,39

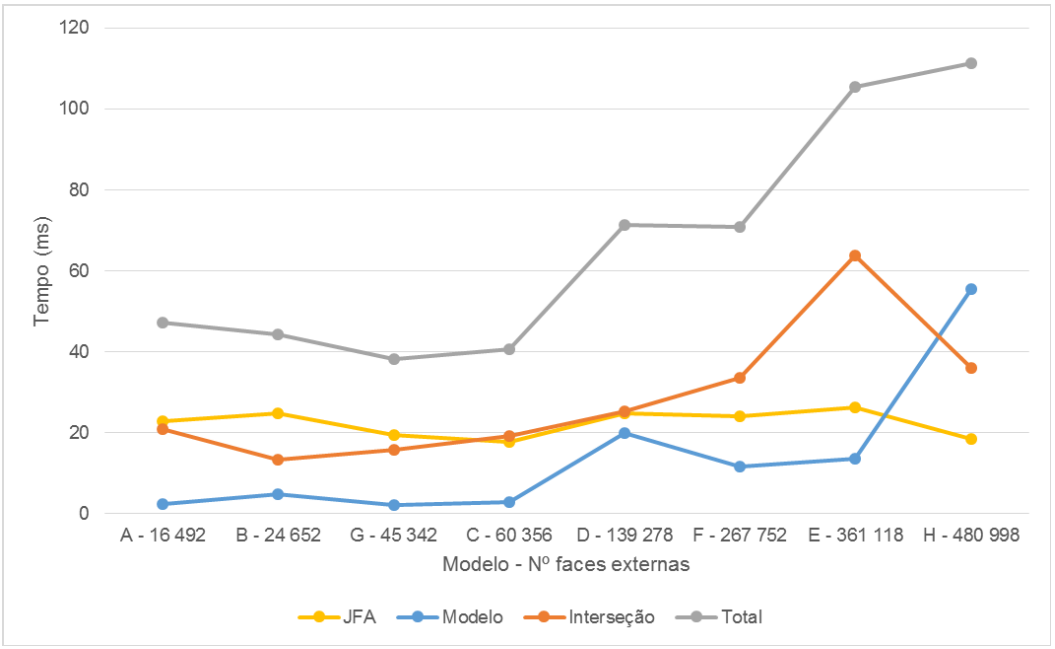


Figura 5.19 – Comparação entre as medidas de tempo das etapas do algoritmo de corte orientado ao observador.

6
Conclusão

Neste trabalho, foi proposto um novo algoritmo de corte aplicado a modelos de reservatório de petróleo. O corte é gerado pela interatividade do usuário com o modelo, definindo um conjunto de objetos de interesse a ser inspecionado. O controle da abertura do corte e a maneira como a superfície de corte é gerada, a partir da profundidade dos objetos de interesse, minimiza significantemente a quantidade de contexto, as células do modelo, a serem cortadas. O desenho da superfície de corte é eficiente, explorando um algoritmo que efetua o cálculo de interseção de fragmentos desta na GPU. Isso remove a necessidade de desenhar todas células do modelo, prosseguindo apenas com o desenho de suas faces externas, o que reduz drasticamente a quantidade de primitivas enviadas para placa gráfica. O algoritm@articleID, author = author, title = title, journaltitle = journaltitle, date = date, o é flexível, permitindo que qualquer objeto relevante ao contexto do modelo de reservatórios possa ser utilizado como objeto de interesse. Poços foram explorados como os principais objetos de interesse ao longo desse trabalho; como exemplos de aplicação, foram também definidos conjunto de células e linhas de fluxo.

Os resultados compravam a eficácia dos algoritmos propostos, produzindo cortes em taxas de quadro por segundo interativas mesmo para modelos massivos. A qualidade do corte é comprovada através de várias combinações de poços como objetos de interesse, produzindo diferentes cortes visualizados de diversas posições. Como exemplo de informações extraídas com esses cortes, às células do modelo foram associadas com diferentes propriedades e foi possível observar a variação da intensidade das mesmas de acordo com o passo de tempo corrente. Através da análise de desempenho, observou-se que a quantidade de primitivas, o número de faces externas enviadas para placa gráfica, representam o provável gargalo de desempenho no sistema desenvolvido.

Como continuação do trabalho, idealiza-se a integração dos algoritmos desenvolvidos a um sistema de produção e a avaliação por especialistas na área de engenharia de petróleo. O algoritmo de corte desacoplado pode ser melhorado substituindo a etapa de busca linear por busca em mapa de cones, como proposto por [31]. O desenho das faces externas para modelos massivos representa uma etapa de elevado custo durante a renderização, técnicas de

nível de detalhe (*level of detail, LOD*) e descarte (*culling*) podem vir a ser exploradas para alivar este problema. Pretende-se também, explorar técnicas que promovem maior percepção de profundidade na visualização dos cortes como, por exemplo, oclusão ambiente.

Referências bibliográficas

[1] RONG, G.; TAN, T.-S.. **Jump flooding in GPU with applications to voronoi diagram and distance transform**. In: PROCEEDINGS OF THE 2006 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, I3D '06, p. 109–116. ACM, 2006.

[2] FELIPPA, C. A.. **Advanced finite element methods**, 2013. `http://www.colorado.edu/engineering/CAS/courses.d/AFEM.d/`.

[3] FRANCESCHIN, B. B.. **Visualização de seções de corte arbitrárias de malhas não estruturadas**. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2012.

[4] AGRAWALA, M.; LI, W. ; BERTHOUZOZ, F.. **Design principles for visual communication**. Commun. ACM, 54(4):60–69, 2011.

[5] BIER, E. A.; STONE, M. C.; PIER, K.; BUXTON, W. ; DEROSE, T. D.. **Toolglass and magic lenses: The see-through interface**. In: PROCEEDINGS OF THE 20TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '93, p. 73–80, New York, NY, USA, 1993. ACM.

[6] LAMAR, E.; HAMANN, B. ; JOY, K. I.. **A magnification lens for interactive volume visualization**. In: COMPUTER GRAPHICS AND APPLICATIONS, 2001. PROCEEDINGS. NINTH PACIFIC CONFERENCE ON, p. 223–232, 2001.

[7] WANG, Y. S.; LEE, T. Y. ; TAI, C. L.. **Focus+Context visualization with distortion minimization**. IEEE Transactions on Visualization and Computer Graphics, 14(6):1731–1738, Nov 2008.

[8] MONCLÚS, E.; DÍAZ, J.; NAVAZO, I. ; VÁZQUEZ, P.-P.. **The virtual magic lantern: An interaction metaphor for enhanced medical data inspection**. In: PROCEEDINGS OF THE 16TH ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY, VRST '09, p. 119–122, New York, NY, USA, 2009. ACM.

[9] BARIČEVIĆ, D.; LEE, C.; TURK, M.; HÖLLERER, T. ; BOWMAN, D. A.. **A hand-held ar magic lens with user-perspective rendering**. In: MIXED AND AUGMENTED REALITY (ISMAR), 2012 IEEE INTERNATIONAL SYMPOSIUM ON, p. 197–206, Nov 2012.

[10] LI, W.; AGRAWALA, M. ; SALESIN, D.. **Interactive image-based exploded view diagrams**. In: PROCEEDINGS OF GRAPHICS INTERFACE 2004, GI '04, p. 203–212, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[11] BRUCKNER, S.; GROLLER, M. E.. **Exploded views for volume data**. IEEE Transactions on Visualization and Computer Graphics, 12(5):1077–1084, Sept 2006.

[12] KARPENKO, O.; LI, W.; MITRA, N. ; AGRAWALA, M.. **Exploded view diagrams of mathematical surfaces**. IEEE Transactions on Visualization and Computer Graphics, 16(6):1311–1318, Nov 2010.

[13] FILHO, Z. M.; BRAZIL, E. V. ; SOUSA, M. C.. **Exploded view diagrams of 3D grids**. In: 2015 28TH SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES, p. 242–249, Aug 2015.

[14] FEINER, S. K.; SELIGMANN, D. D.. **Cutaways and ghosting: satisfying visibility constraints in dynamic 3D illustrations**. The Visual Computer, 8(5):292–302, 1992.

[15] CROW, F. C.. **Shadow algorithms for computer graphics**. SIGGRAPH Comput. Graph., 11(2):242–248, July 1977.

[16] DIEPSTRATEN, J.; WEISKOPF, D. ; ERTL, T.. **Interactive cutaway illustrations**. Computer Graphics Forum, 22(3):523–532, 2003.

[17] COFFIN, C.; HOLLERER, T.. **Interactive perspective cut-away views for general 3D scenes**. In: 3D USER INTERFACES (3DUI'06), p. 25–28, March 2006.

[18] BRUCKNER, S.; GROLLER, E.. **VolumeShop: an interactive system for direct volume illustration**. In: IEEE VISUALIZATION, 2005. VIS 05, p. 671–678, 2005.

[19] LI, W.; RITTER, L.; AGRAWALA, M.; CURLESS, B. ; SALESIN, D.. **Interactive cutaway illustrations of complex 3D models**. In: ACM SIGGRAPH 2007 PAPERS, SIGGRAPH '07. ACM, 2007.

[20] VIOLA, I.; KANITSAR, A. ; GROLLER, M. E.. **Importance-driven volume rendering**. In: PROCEEDINGS OF THE CONFERENCE ON VISUALIZATION '04, VIS '04, p. 139–146. IEEE Computer Society, 2004.

[21] BURNS, M.; HAIDACHER, M.; WEIN, W.; VIOLA, I. ; GROELLER, E.. **Feature Emphasis and Contextual Cutaways for Multimodal Medical Visualization**. In: Museth, K.; Moeller, T. ; Ynnerman, A., editors, EUROGRAPHICS/ IEEE-VGTC SYMPOSIUM ON VISUALIZATION. The Eurographics Association, 2007.

[22] BURNS, M.; FINKELSTEIN, A.. **Adaptive cutaways for comprehensible rendering of polygonal scenes**. In: ACM SIGGRAPH ASIA 2008 PAPERS, SIGGRAPH Asia '08, p. 154:1–154:7. ACM, 2008.

[23] LIDAL, E. M.; HAUSER, H. ; VIOLA, I.. **Design principles for cutaway visualization of geological models**. In: PROCEEDINGS OF THE 28TH SPRING CONFERENCE ON COMPUTER GRAPHICS, SCCG '12, p. 47–54. ACM, 2012.

[24] DE CARVALHO, F. M.; BRAZIL, E. V.; MARROQUIM, R. G.; SOUSA, M. C. ; OLIVEIRA, A.. **Interactive cutaways of oil reservoirs**. Graphical Models, 84:1 – 14, 2016.

[25] PINDAT, C.; PIETRIGA, E.; CHAPUIS, O. ; PUECH, C.. **Drilling into complex 3D models with gimlenses**. In: PROCEEDINGS OF THE 19TH ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY, p. 223–230. ACM, 2013.

[26] BORGEFORS, G.. **Distance transformations in digital images**. Computer Vision, Graphics, and Image Processing, 34(3):344 – 371, 1986.

[27] FILHO, Z. M.; BRAZIL, E. V.; SOUSA, M. C.; CARVALHO, F. D. ; MARROQUIM, R.. **Cutaway applied to corner point models**. In: WORKSHOP ON INDUSTRY APPLICATIONS (WGARI) IN SIBGRAPI 2012 (XXV CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES), Ouro Preto, MG, Brazil, august 2012.

[28] CELES, W.; PAULINO, G. H. ; ESPINHA, R.. **A compact adjacency-based topological data structure for finite element mesh representation**. International Journal for Numerical Methods in Engineering, 64(11):1529–1556, 2005.

[29] CELES, W.; PAULINO, G. H. ; ESPINHA, R.. **Efficient Handling of Implicit Entities in Reduced Mesh Representations**. Journal of Computing and Information Science in Engineering, 5(4):348–359, Aug. 2005.

[30] POLICARPO, F.; OLIVEIRA, M. M. ; COMBA, J. A. L. D.. **Real-time relief mapping on arbitrary polygonal surfaces**. In: PROCEEDINGS OF THE 2005 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, I3D '05, p. 155–162, New York, NY, USA, 2005. ACM.

[31] POLICARPO, F.; OLIVEIRA, M. M.. **Relaxed cone stepping for relief mapping**. GPU gems, 3:409–428, 2007.

[32] BÆRENTZEN, A.; NIELSEN, S. L.; GJØL, M.; LARSEN, B. D. ; CHRISTENSEN, N. J.. **Single-pass wireframe rendering**. In: ACM SIGGRAPH 2006 SKETCHES, p. 149. ACM, 2006.

[33] GLASSNER, A. S.. **Principles of digital image synthesis**. Morgan Kaufmann, 1995.