Pontifícia Universidade Católica
do Rio de Janeiro

Jordan Rodrigues Rangel

Extração de Isosuperfícies com Subdivisão Adaptativa de Malhas de Hexaedros Levemente Côncavos

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 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Jordan Rodrigues Rangel

Extração de Isosuperfícies com Subdivisão Adaptativa de Malhas de Hexaedros Levemente Côncavos

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. Aprovada pela Comissão Examinadora abaixo.

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

Prof. Luiz Henrique de Figueiredo
IMPA

Prof. Marcelo Gattass
Departamento de Informática – PUC-Rio

Dr. Frederico Rodrigues Abraham
PUC-Rio

Rio de Janeiro, 29 de Setembro de 2020

Jordan Rodrigues Rangel

Graduado em Ciência da Computação pela Pontifícia Universidade Católica do Rio de Janeiro (Rio de Janeiro, Brasil) em 2017, tornou-se bolsista de mestrado associado ao Instituto Tecgraf de Desenvolvimento de Software Técnico-Científico da PUC-Rio em 2018, onde iniciou seu desenvolvimento profissional em Computação Gráfica. Durante o mestrado, direcionou sua pesquisa para extração de isosuperfícies.

Ficha Catalográfica

Rangel, Jordan Rodrigues

Extração de Isosuperfícies com Subdivisão Adaptativa de Malhas de Hexaedros Levemente Côncavos / Jordan Rodrigues Rangel; orientador: Waldemar Celes Filho. – Rio de Janeiro: PUC-Rio, Departamento de Informática, 2020.

v., 57 f: il. color. ; 30 cm

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

Inclui bibliografia

1. Informática – Teses. 2. Isosuperfícies;. 3. Geometria Computacional;. 4. Marching Cubes;. 5. Superfícies curvas;. 6. Subdivisão adaptativa;. 7. Modelo de reservatório de petróleo;. 8. Dual Contouring;. I. Celes Filho, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Dedico esta dissertação à minha noiva e à minha mãe.

Agradecimentos

Agradeço primeiramente a Deus e à Santíssima Virgem Maria, Mãe de Deus e Mãe nossa.

Agradeço à minha noiva, à minha família, especialmente minha mãe, e aos meus amigos.

Agradeço imensamente ao meu orientador Waldemar Celes por me aceitar como orientando e aos professores que tive ao longo do curso e que contribuíram com a minha formação.

Agradeço a todos os membros do MOVE, grupo de liderança católica da PUC-Rio, por toda a formação e amizade.

Agradeço, ainda, aos meus colegas do Instituto Tecgraf sempre dispostos a ajudar, aos funcionários do Departamento de Informática, das secretarias e reitorias da PUC-Rio, da Sodexo, aos comerciantes dentro e ao redor da PUC e a todos os demais que, com seu trabalho honesto e digno, possibilitaram os meus estudos.

Por fim, agradeço à PUC-Rio, ao Instituto Tecgraf e ao CNPq pelos auxílios, possibilitando-me uma dedicação mais exclusiva à minha pesquisa.

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

Resumo

Rangel, Jordan Rodrigues; Celes Filho, Waldemar. **Extração de Isosuperfícies com Subdivisão Adaptativa de Malhas de Hexaedros Levemente Côncavos**. Rio de Janeiro, 2020. 57p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A extração e visualização de isosuperfícies de campos escalares são importantes para inspeções e análises de modelos em diversas áreas. Uma isosuperfície é representada por uma malha de triângulos que aproxima um conjunto de nível do volume de dados. O foco deste trabalho é a extração e visualização de isosuperfícies de modelos de reservatório de petróleo, representados por malhas de hexaedros levemente côncavos. Para uma melhor representação das isosuperfícies, optou-se pela substituição de triângulos planares por superfícies curvas. Para assegurar a extração de superfícies contínuas e suaves, este trabalho propõe o uso de "envelopes" para determinação das normais. A técnica proposta é implementada em GPU com uso de subdivisão adaptativa das superfícies.

Palavras-chave

Isosuperfícies; Geometria Computacional; Marching Cubes; Superfícies curvas; Subdivisão adaptativa; Modelo de reservatório de petróleo; Dual Contouring;

Abstract

Rangel, Jordan Rodrigues; Celes Filho, Waldemar (Advisor). **Isosurface Extraction with Adaptive Tessellation from Hexahedral Meshes with Slightly Concave Cells**. Rio de Janeiro, 2020. 57p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The extraction and visualization of isosurfaces of scalar fields are important for inspections and analysis of models in several areas. An isosurface is represented by a mesh of triangles that approximates a level set of a data volume. The main focus of this work is the extraction and visualization of isosurfaces of black oil reservoir models, represented by hexahedral meshes with slightly concave cells. For a better representation of the isosurfaces, we have opted to replace planar triangles for curved patches. To ensure the extraction of continuous and smooth surfaces, this work proposes the use of "envelopes" to determine normals. The proposed technique is implemented in GPU with the usage of adaptive subdivision of patches.

Keywords

Isosurfaces; Computational Geometry; Marching Cubes; Curved patches; Adaptive tessellation; Black oil reservoir model; Dual Contouring;

Sumário

Lista de figuras

1
Introdução

Na área de Computação Gráfica, visualização refere-se à projeção de uma imagem gerada a partir do processamento de dados armazenados e estruturados computacionalmente. Visualização pode ser categorizada basicamente em duas classes de algoritmos: a direta e a indireta. A primeira refere-se ao *rendering* realizado diretamente a partir de um conjunto de dados de entrada, enquanto a segunda conta com um construto intermediário, como uma malha de triângulos. Um exemplo de algoritmo de visualização direta é o *Volume Ray-Tracing* (3) e um de visualização indireta é o *Marching Cubes* (25).

Há diversos interesses que levam a ser produzidos *softwares* para a visualização de dados volumétricos tridimensionais (i.e., dados amostrados em grades estruturadas ou malhas não estruturadas no espaço 3D (33)). Os casos de uso de tais programas incluem: a reconstrução de alguma parte do corpo humano a partir de cortes de tomografia computadorizada, como o da Figura 1.1, e um meio para analisar a anatomia e processos fisiológicos do corpo humano; a renderização de superfícies de fluidos em animações e simulações; e a demonstração de quais regiões possuem um certo valor de alguma propriedade (e.g., temperatura ou densidade de locais do subsolo), além de várias outras possíveis aplicações.

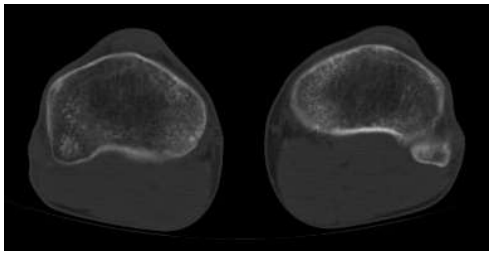


Figura 1.1: Uma imagem de tomografia computadorizada (TC) dos joelhos de uma pessoa (12).

É possível realizar a visualização indireta por meio da implementação de algoritmos de extração de isosuperfícies, como o *Surface Nets* (16) e o *Marching Cubes* (25), amplamente pesquisado e utilizado tanto na academia

Figura 1.2: Visualização de uma isosuperfície de dados provenientes de TC por meio do *Marching Cubes*.

quanto na indústria. Esses algoritmos geram uma superfície de interesse a partir de um conjunto de dados estruturados na forma de grade, como é o caso da isosuperfície extraída de dados oriundos de TC de um joelho, conforme ilustrado na Figura 1.2. O usuário pode, então, visualizar e analisar a região para chegar a novas conclusões que antes seriam difíceis com os dados brutos de entrada.

Na verdade, a superfície gerada é uma aproximação de um conjunto de pontos de certo valor e consiste em uma malha composta normalmente de triângulos. Entretanto, nem sempre os triângulos, que são planares, representam bem tal região, pois esta muitas vezes é curva. Para remediar o problema, é possível substituir os triângulos por superfícies curvas por meio de algoritmos que utilizam também a informação das normais, como o *Curved Point Normal Triangles* (36), que usa superfícies paramétricas de Bézier.

É bem comum que o tamanho do domínio do problema seja grande. A consequência disto é que a extração da isosuperfície pode ser lenta, o que impacta ou até impossibilita que a aplicação execute em tempo real. A programação paralela por meio da GPU e a utilização da técnica de subdivisão adaptativa melhoram imensamente a velocidade da extração e da visualização.

Os algoritmos de extração citados esperam como entrada uma grade regular uniforme. Entretanto, o domínio de interesse desta dissertação é reservatórios de modelos de simulação de petróleo, representados por malhas de hexaedros levemente côncavos com faces não planares, de modo que é preciso fazer uma adaptação dos procedimentos de extração de isosuperfícies.

O trabalho realizado e descrito nesta dissertação consiste na pesquisa e no desenvolvimento de algoritmos, especialmente do MC, para a extração e visualização em tempo real de isosuperfícies de modelos de reservatório de petróleo. A contribuição deste trabalho reside no desenvolvimento de uma solução que abrange uma adaptação paralelizada do MC para esse tipo de

malhas, a substituição de triângulos por superfícies paramétricas com subdivisão adaptativa, e o cálculo apropriado da normal nos vértices com a aproximação linear do gradiente por meio da matriz Jacobiana e nas superfícies curvas através da nova proposta da técnica do envelope.

Alguns trabalhos relacionados a esta dissertação são discutidos no Capítulo 2. Os algoritmos de extração de isosuperfície mais conhecidos são introduzidos e comparados no Capítulo 3. Já no Capítulo 4, expõe-se o que é subdivisão adaptativa e como superfícies curvas podem ser definidas parametricamente com polinômios de Bernstein. No Capítulo 5, os modelos de simulação de reservatório de petróleo são explicados, assim como adaptar para essas malhas tudo o que foi explorado nas seções anteriores. Os resultados finais são mostrados no Capítulo 6 e alguns aspectos da implementação são discutidos. Finalmente, a conclusão da dissertação e propostas de trabalhos futuros são comentados no Capítulo 7.

2
Trabalhos Relacionados

Marching Cubes (25) é um algoritmo amplamente estudado e desenvolvido desde sua invenção em 1987 por *Lorensen e Cline*. Sua patente expirou em 2005. Trata-se de um algoritmo de extração de isosuperfície usado para visualização indireta. Ele consiste em encontrar os pontos de certo valor por meio de interpolação das arestas das células, dentro das quais submalhas de triângulos são geradas de acordo com a classificação da célula para aproximar um conjunto de nível.

Como cada célula possui 8 vértices, há 256 configurações possíveis. Para diminuir essa quantidade, o artigo original do *Marching Cubes* (MC) propõe aproveitar simetrias de rotação e de complementaridade (i.e., células com vértices de sinais trocados). Entretanto, segundo *Dürst* (14), a proposta inicial possui o problema de múltiplas triangulações possíveis, além de defeitos topológicos, como a descontinuidade da malha gerada, por causa de ambiguidades, da tabela auxiliar de configuração das células ser incompleta e do uso das estratégias de diminuir o número de configurações.

As ambiguidades de face ocorrem quando os sinais dos vértices da face são alternados (i.e., uma diagonal positiva e a outra negativa). Nessas situações, mais de uma triangulação é possível. Para solucionar essas ambiguidades, *Montani et al.* (26) e *Zhou, Shu e Kankanhalli* (39) recomendam não usar a simetria de complementaridade, *Nielson e Hamann* (31) criaram um algoritmo chamado *Asymptotic Decider* (AD), baseado em interpolação bilinear, e adicionaram novos casos de configuração na tabela auxiliar, e *Gelder e Wilhelms* (34) definiram hipóteses relacionadas à consistência do gradiente nos vértices da face.

Já as ambiguidades do interior acontecem em casos nos quais os vértices de uma das diagonais principais da célula compartilham o mesmo sinal com um túnel possivelmente conectando a ambos (isosuperfície homeomorfa a um cilindro (10)). Nessas circunstâncias, há mais de uma triangulação que pode ser feita. *Natarajan* (29), em 1994, propõe resolver a ambiguidade por meio da consideração de pontos de sela do interpolante trilinear, além de acrescentar novos casos na tabela.

Uma outra solução para resolver a ambiguidade do interior foi dada por

Chernyaev (9), que criou um teste baseado no AD e uma nova tabela com 33 casos, razão pela qual o nome do algoritmo modificado é *Marching Cubes 33* (MC33).

Mesmo com todos os avanços e modificações, os métodos ainda não estavam completos nem com todos os defeitos topológicos resolvidos. A fim de completar o MC33, *Lewiner et. al* (22) propõem uma implementação, que foi depois corrigida por *Custódio* (10).

Entretanto, segundo *Vega et. al* (35), *Lewiner et. al* (22) não interpretaram corretamente o teste do interior proposto no MC33 e *Custódio* (10) enganou-se ao afirmar que o teste falha. *Vega et. al* (35) propõem uma nova implementação otimizada em termos de desempenho e memória.

O MC pode não conseguir aproximar bem arestas afiadas (*sharp edges*) e silhuetas da isosuperfície. Uma tentativa de solucionar esta dificuldade foi feita por *Kobbelt et al.* (19), que modificaram o MC e esperam como entrada a distância a uma superfície dada. Os algoritmos do tipo *Dual Contouring* (DC) costumam representar melhor estas características agudas. O primeiro algoritmo desta classe foi o *Surface Nets* (SN), de *Gibson* (16). Enquanto no MC os vértices da isosuperfície são inseridos nas arestas das células, nos algoritmos de DC eles são inseridos dentro das células, que é a razão pela qual é possível ter maior fidelidade em relação às arestas agudas da isosuperfície.

O problema do SN é que a isosuperfície gerada pode não ser *manifold* para algumas configurações. Para sanar isto, *Nielson* (30) desenvolveu em 2004 o *Dual Marching Cubes* (DMC), um algoritmo também do tipo DC. A diferença entre o SN e o DMC é que o SN só produz um único vértice por célula, enquanto o DMC pode gerar múltiplos vértices, dentre os quais o vértice mais apropriado é escolhido para compor a superfície a ser gerada segundo uma tabela de configuração e a aresta bipolar interna em questão.

Entretanto, ainda há alguns casos nos quais a superfície pode não ser *manifold* mesmo com o DMC. Estes casos ocorrem quando há certas configurações de células adjacentes com facetas ambíguas. Para lidar com estas situações, *Wenger* realizou algumas modificações neste algoritmo, criando um novo chamado *Manifold Dual Marching Cubes* (38). A ideia é substituir essas configurações por outras que não levem a superfícies que não sejam *manifolds*.

Para melhor aproximação do conjunto de nível, é possível substituir os triângulos gerados pelos algoritmos de extração de isosuperfície por superfícies curvas. *Vlachos et al.* (36) propõem o *Curved Point-Normal Triangles*, substituindo os triângulos por superfícies cúbicas triangulares de Bézier com variação quadrática da normal, que se consegue por meio de superfícies quadráticas e também triangulares de Bézier.

Todavia, há defeitos visuais que podem ser causados tanto pelo uso inadequado da interpolação quadrática em certos casos, como no caso de curvas quadráticas ou cúbicas sem inflexão, quanto pela falta de coplanaridade entre uma aresta e suas respectivas normais. Este problema é enunciado e discutido por *Lee e Jen* (21).

Há outras abordagens que também substituem polígonos planares por superfícies curvas. *Phong Tessellation* (6), por exemplo, é uma técnica que substitui triângulos por superfícies quadráticas triangulares de Bézier com interpolação linear da normal. Estas características, a princípio, não permitem que tratem de inflexões na geometria de forma realística. Nesse mesmo trabalho do *Phong Tessellation* (6), *Boubekeur e Alexa* também propõem um método para tornar a subdivisão adaptativa utilizando a posição da câmera.

Enquanto o *PN Triangles* é um método utilizado na execução de *shaders* na GPU e substitui em tempo real os triângulos por superfícies paramétricas, existem aqueles que subdividem os polígonos de entrada e geram uma nova geometria mais suave, como é o caso do algoritmo de subdivisão de superfície e *Catmull-Clark* (8).

Propostas como o algoritmo de aproximação de subdivisão de superfícies de *Loop et. al* (24) e o algoritmo de construção de superfícies suaves *Myles et. al* (28) lidam com outros tipos de polígonos, não apenas triângulos, como quadriláteros e pentágonos.

Há também algoritmos que lidam com superfícies definidas com funções não-polinomiais, como no caso da superfície de *Gregory* de *Walton e Meek* (13), que é quártico, utiliza a técnica de *blending* de superfícies e no qual é preciso lidar com um problema de singularidade. A ideia é que, assim, seja garantida uma continuidade G^1. Por apresentar tais características, esta superfície é mais custosa de se computar.

Existem os métodos denominados globais que também conseguem alcançar continuidade G^1 ou C^1, o que não é o caso do *PN Triangles*, que é local e só garante C^0. Um exemplo de método global é o proposto por *Moreton e Séquin* (27), que utiliza técnicas de otimização numérica. Tais métodos também costumam apresentar um desempenho pior.

As malhas referentes a reservatórios de petróleo são de hexaedros levemente côncavos e não são necessariamente regulares nem uniformes. Para conseguir calcular a normal como sendo o gradiente do campo escalar, é preciso calculá-la no espaço paramétrico e, então, trazê-la ao espaço do mundo por meio da inversa da matriz Jacobiana, conforme proposto por *Barroso* em seu trabalho de simulação de fluidos (4).

3
Extração de Isosuperfície

Como dito na introdução deste documento, a visualização indireta conta com um construto intermediário, a partir do qual ocorrerá a renderização. A sua geração, no caso desta dissertação, é realizada por meio de algoritmos de extração de isosuperfície.

Os algoritmos descritos neste capítulo são o *Marching Cubes* e alguns do tipo *Dual Contouring*, como o *Surface Nets*. Todos recebem como entrada um isovalor de interesse e uma amostra de um campo escalar composta por vértices tridimensionais de uma grade regular associados a escalares. Como resultado, retornam uma malha de triângulos que representa a isosuperfície (38).

Para melhor compreensão do problema e do programa produzido, é necessário esclarecer antes noções basilares, o que será feito na seção a seguir.

3.1
Conceitos preliminares

A primeira definição a ser apresentada é a do campo escalar, que é uma função que associa um valor escalar real para cada ponto de $\Re^d$, onde d é a dimensão do domínio. No caso de interesse, $d = 3$.

Sejam o campo escalar $\phi : \Re^d \rightarrow \Re$ e o isovalor $\sigma \in \Re$, o conjunto $\{x : \phi(x) = \sigma\}$ é chamado de um conjunto de nível de ϕ e pode ser representado por $\phi^{-1}(\sigma)$. Se ϕ é uma função contínua, então o conjunto de nível separa $\Re^d$ em dois conjuntos de pontos, aqueles com valor escalar menor que o isovalor e os que têm valor maior.

Em duas dimensões, um conjunto de nível é chamado de curva de nível. Já no caso tridimensional, pode receber o nome de superfície de nível ou, como é chamado na área de visualização de volumes, isosuperfície, que geralmente se refere a uma superfície construída a partir de um conjunto finito de pontos de entrada, cada um associado a um valor escalar. Estes pontos são uma amostra de algum campo escalar e, neste caso, de forma mais estrita, a isosuperfície é uma aproximação do conjunto de nível (38). A Figura 3.1 ilustra uma isosuperfície extraída a partir de amostras geradas por meio de tomografias computadorizadas.

Ainda no caso de três dimensões, quando se tem explicitamente um campo escalar ϕ que seja uma função implícita, o conjunto de nível pode ser chamado de superfície implícita, como exemplificado na Figura 3.2.

O processo de gerar essa aproximação de um conjunto de nível a partir de dados volumétricos chama-se extração de isosuperfície, que é implementada por meio de algoritmos como *Marching Cubes* e *Surface Nets*. Tais dados estão estruturados em forma de grade (*grid*). Cada vértice está associado a um valor escalar.

Figura 3.1: Visualização de uma isosuperfície referente a um dente.

Figura 3.2: Extração de uma superfície implícita de gênero igual a dois.

Um vértice é classificado como positivo se seu valor é maior ou igual ao isovalor. Caso contrário, ele é classificado como negativo. Uma aresta é dita bipolar se possuir um vértice positivo e outro negativo. Isto significa, no caso de um campo contínuo, que este passa pela aresta. Uma célula ativa é aquela que possui pelo menos uma aresta bipolar. Dito de outro modo, trata-se de uma célula cortada por ϕ.

Cada vértice e aresta são identificados por um índice local único, conforme a Figura 3.3 demonstra. Como há oito vértices em uma célula, a configuração de uma célula pode ser representada por um *byte*, cada *bit* representando se o respectivo vértice é positivo ou negativo. Tal identificador é utilizado pelos

algoritmos de extração para poder acessar tabelas auxiliares (*look-up tables*), que informam qual superfície deve ser gerada para aquela célula específica. Os vértices a serem criados nas arestas bipolares e que formam os triângulos que aproximam a isosuperfície dentro da célula são chamados de isovértices.

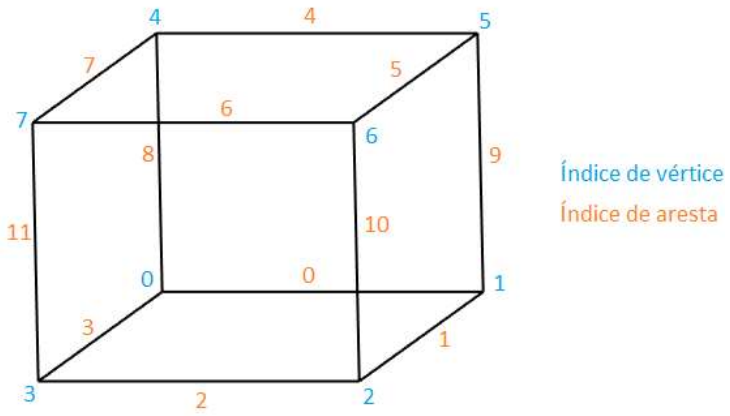


Figura 3.3: Ordem local dos vértices e das arestas de uma célula (7).

É importante ressaltar a suposição de que a entrada de um algoritmo de extração de isosuperfície consiste em um conjunto finito de pontos associados a valores amostrados a partir de um campo escalar ϕ, além do isovalor. Espera-se também que ϕ seja uma função contínua.

Por fim, uma última definição a ser tratada é a de *manifold*, que, segundo *Wenger* (38), é uma formalização matemática do conceito intuitivo de uma superfície. Um *k-manifold* é um conjunto de pontos que se assemelha localmente a uma bola aberta $\mathbb{B}^k$ de raio 1 centrada na origem. $\mathbb{B}^1$ é um segmento de linha aberto e $\mathbb{B}^2$ é um disco aberto. Um *k-manifold* com fronteira é um conjunto de pontos que se assemelha localmente a $\mathbb{B}^k$ em união com a fronteira. Exemplos de *2-manifold* são discos e polígonos convexos.

3.2 Marching Cubes

Marching Cubes (MC), criado por *Lorensen e Cline* (25), é o algoritmo mais conhecido e pesquisado na área de extração de isosuperfície. Tal como os demais algoritmos, o MC recebe como entrada um isovalor e dados volumétricos estruturados em forma de grade, onde cada vértice está associado a um valor escalar. O procedimento conta ainda com uma tabela auxiliar para gerar uma superfície em cada célula segundo sua configuração, definida pela classificação

de seus vértices em relação ao isovalor. O pseudocódigo do MC está descrito no Algoritmo 1.

As células são processadas de forma independente. Primeiro, os vértices são percorridos para averiguar se são positivos ou negativos em relação à isosuperfície e, durante este processo, constrói-se o identificador de 8 *bits* que representa a configuração da célula. Caso se trate de uma célula ativa, gera-se os triângulos com a utilização de uma tabela auxiliar, na qual a linha a ser usada é identificada pelo *byte* formado e contém quais são as arestas bipolares onde devem ser postos os isovértices que compõem os triângulos da superfície. Em outras palavras, cada linha da tabela representa uma possível configuração da célula e determina os triângulos a serem construídos.

Tais vértices são criados por meio de uma interpolação linear. Sejam σ o isovalor, v_i o isovértice que se deseja gerar, v_n o vértice negativo da aresta bipolar, v_p o vértice positivo, e e_n e e_p os escalares associados a estes vértices, a posição de v_i pode ser encontrada por meio da seguinte fórmula: $(1-t)v_n + tv_p$, onde $t = (\sigma - e_n)/(e_p - e_n)$.

Da mesma forma, as normais dos isovértices também podem ser calculadas através de uma interpolação linear. Entretanto, para isso, as normais dos vértices da grade devem estar definidas. Caso não estejam, é possível computá-las como sendo a normalização do vetor gradiente, que é perpendicular à superfície e é definida pelas derivadas parciais do campo escalar. Uma forma de aproximar o gradiente é por meio do método das diferenças finitas centrais, utilizado tanto pelo artigo original (25) quanto neste trabalho e expresso matematicamente a seguir:

$$D_x(i,j,k) = \frac{\Phi(i+1,j,k) - \Phi(i-1,j,k)}{\Delta x},$$

$$D_y(i,j,k) = \frac{\Phi(i,j+1,k) - \Phi(i,j-1,k)}{\Delta y},$$

$$D_z(i,j,k) = \frac{\Phi(i,j,k+1) - \Phi(i,j,k-1)}{\Delta z},$$

$$n(i,j,k) = \frac{(D_x, D_y, D_z)}{||(D_x, D_y, D_z)||},$$

onde D_x, D_y e D_z são as derivadas parciais aproximadas que compõem o

gradiente no vértice identificado pela tripla (i, j, k), $\Delta x, \Delta y$ e Δz são os comprimentos das células em cada um dos eixos, $\Phi(i, j, k)$ é o valor do campo escalar amostrado no vértice e $n(i, j, k)$ a sua respectiva normal. Nas bordas, onde falta o vértice adjacente para calcular a derivada parcial, utiliza-se como substituto o valor $\Phi(i, j, k)$ amostrado no próprio vértice.

Algoritmo 1: Marching Cubes

```
Entrada:
  σ - um isovalor
  G - uma grade regular 3D
Saída:
  P - um array de triângulos
T ← Pré-processamento da tabela auxiliar
para cada célula (i, j, k) faça
    config = 0
    c = 0
    enquanto c < 8 faça
        v_c = G[(i, j, k)][c]
        se v_c.escalar < σ então
            config = config | (1«c)
        c++
    para cada tripla de arestas (a_1, a_2, a_3 ∈ T[config]) faça
        triângulo = {}
        para cada a da tripla a_1, a_2, a_3 faça
            v_n = a.v_n
            v_p = a.v_p
            e_n = v_n.escalar
            e_p = v_p.escalar
            t = (σ − e_n)/(e_p − e_n)
            v = InterpolaçãoLinear(v_p, v_n, t)
            triângulo = triângulo ∪ v
        P = P ∪ triângulo
Retorna P
```

Dado que a configuração de uma célula é identificada por meio da classificação dos vértices, que consiste em positivo ou negativo, existem $2^8 = 256$ configurações possíveis. Destes, há dois casos nos quais não há geração de isosuperfície, que são as configurações onde todos os vértices possuem as mesmas classificações, isto é, todos são positivos ou todos são negativos.

Por fim, é importante ressaltar uma propriedade da isosuperfície gerada por meio do MC, que é a qualidade de ser *2-manifold* com fronteira, desde que não haja vértices associados a escalares iguais ao isovalor σ. Neste caso, é possível que a superfície seja *não-manifold*.

3.3 Dual Contouring

Existe uma classe de algoritmos de extração de isosuperfície que serve de alternativa ao *Marching Cubes* (MC), chamada *Dual Contouring* (DC). Uma das maiores diferenças entre ambos é onde se põe cada isovértice gerado, que no caso do MC é na aresta, enquanto no DC é no interior da célula.

Dessa diferença é que se origina o termo *dual*. No MC, se um cubo é ativo, traça-se um segmento de linha de uma aresta bipolar a outra, enquanto no DC, se a aresta é bipolar, traça-se um segmento de linha de cubo ativo a outro.

Essa característica diferenciada permite que o DC possa representar melhor características mais afiadas ou agudas (e.g., *sharp edges*). Entretanto, nem todos os algoritmos do tipo DC conseguem garantir o *2-manifold* com fronteira.

3.3.1 Surface Nets

Gibson (16) desenvolveu o *Surface Nets* (SN), o primeiro algoritmo do tipo *Dual Contouring* (DC) e o mais simples. Assim como o *Marching Cubes* (MC), espera-se como entrada um isovalor σ e um campo escalar amostrado por meio de vértices organizados em forma de uma grade regular tridimensional.

O procedimento é, em síntese, o seguinte: primeiro, adiciona-se um isovértice v_i para cada célula ativa; depois, para cada aresta bipolar interna (i.e., compartilhada por quatro células), gera-se um quatrilátero composto pelos isovértices das células ao redor, como ilustrado na Figura 3.4.

A posição de v_i é definida como sendo o centróide dos lugares onde a isosuperfície corta as arestas bipolares da célula. Seja a uma aresta bipolar, v_a o isovértice da aresta a e $|A|$ a quantidade de arestas bipolares da célula, o centróide é calculado por $\sum_a v_a/|A|$.

A posição das interseções entre a isosuperfície e as arestas bipolares (v_a) é calculada da mesma forma como os isovértices são computados no MC: pela interpolação linear dos vértices que compõem a aresta.

Seguindo apenas os passos citados para a execução do algoritmo, o SN tem como saída uma malha de quadriláteros, quando o que se deseja é uma malha de triângulos. Portanto, é preciso acrescentar um passo a mais e converter os quatriláteros em triângulos.

Não há garantia de que os vértices que formam o quadrilátero sejam coplanares. Nos casos em que são, basta subdividir o polígono pela diagonal em dois triângulos. Isto não é adequado nos demais casos, pois é possível que disto decorra uma isosuperfície com auto-interseções (17).

Para solucionar esse problemas, *Ju e Udeshi* (17) sugerem subdividir o quadrilátero em quatro triângulos. Seja v_a o resultado da interpolação de uma aresta bipolar interna a, os triângulos são formados por v_a e por isovértices de duas células adjacentes que compartilham a, conforme mostrado na Figura 3.4.

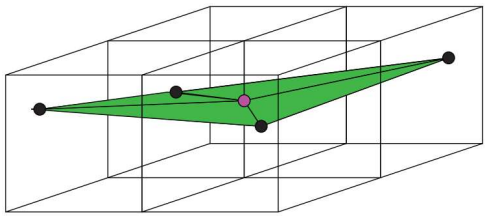


Figura 3.4: Subdivisão de quadrilátero gerado pelo *Surface Nets* em quatro triângulos (38). O ponto rosa é o v_a e os pretos são isovértices das células que compartilham a.

Apesar de não haver ambiguidade no SN, este algoritmo não garante uma isosuperfície *2-manifold* com fronteira. Um exemplo é o caso de uma célula com doze arestas bipolares, fazendo com que seja possível que o isovértice faça parte da composição de doze quadriláteros. Há outros casos onde a isosuperfície não é *manifold*, como nos em que há células com configurações que levam à ambiguidade no MC (38).

3.3.2 Manifold Dual Marching Cubes

O *Dual Marching Cubes* (DMC) é uma evolução do *Surface Nets* (SN). Para tentar resolver o problema do *não-manifold* no SN, *Nielson* (30) propõe considerar mais de uma possibilidade de isovértice por célula ativa. O pseudocódigo do DMC é apresentado no Algoritmo 2.

A escolha de qual vértice utilizar é feita baseando-se em uma tabela auxiliar, que é uma correspondente da utilizada pelo *Marching Cubes* (DC), razão da opção pelo termo *dual* (38). Dadas a configuração da célula e a aresta bipolar interna em questão, a tabela auxiliar indica quais arestas bipolares da respectiva célula devem ser consideradas no cálculo do centróide.

Apesar disso, há ainda alguns casos nos quais o DMC só gera apenas um isovértice e, por isso, pode formar uma isosuperfície que não seja *manifold*. Estes casos ocorrem quando duas configurações de células estão presentes em células adjacentes e compartilham uma face ambígua.

Para solucioná-los, *Wenger* (38) relata uma pequena modificação do algoritmo, dando origem ao chamado *Manifold Dual Marching Cubes* (MDMC). Basicamente, a alteração consiste na troca das superfícies utilizadas apenas

Algoritmo 2: Dual Marching Cubes

Entrada:
σ - um isovalor
G - uma grade regular 3D
Saída:
P - um *array* de triângulos
$T \leftarrow$ Pré-processamento da tabela auxiliar
para cada *aresta bipolar interna a* **faça**
  $quadrilátero = \{\}$
  para cada *célula (i,j,k) ao redor de a* **faça**
    $config = 0$
    $c = 0$
    enquanto $c < 8$ **faça**
      $v_c = G[(i, j, k)][c]$
      se $v_c.escalar < \sigma$ **então**
        $config| = (1 « c)$
      $c++$
    $arestas\_bipolares = T[config][a]$
    $vértices\_interpolados = \forall ab \in arestas\_bipolares \rightarrow$
     $InterpolaçãoLinear(ab, \sigma)$
    $v_i = Centroide(vértices\_interpolados)$
    $quadrilátero = quadrilátero \cup v_i$
  $triângulos = Subdivisão(quadrilátero, a)$
  $P = P \cup triângulos$
Retorna P

quando houver células adjacentes com essas configurações problemáticas. Para isto, conta-se com uma segunda tabela auxiliar que indica quais triângulos devem ser formados nesses casos. No Algoritmo 2, a modificação seria justamente o acréscimo de uma checagem para ver em que caso se encontra e usar a segunda tabela se houver essas células com configurações ambíguas.

Tal como no MC, a malha construída pelo MDMC só será *2-manifold* com fronteira se não houver vértices que estejam associados a escalares iguais ao isovalor.

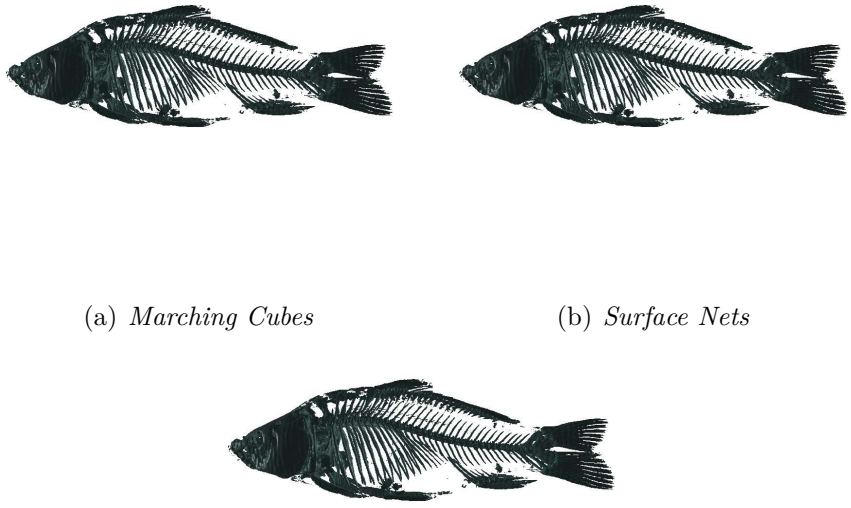
3.4 Considerações

Nesta seção, serão discutidas algumas considerações sobre os algoritmos vistos, além de algumas conclusões e comparações. Uma delas é a solução do problema de haver vértices com escalar igual ao isovalor e quando a amostra é composta apenas por escalares inteiros, o que é bem comum quando a origem dos dados é aparelhos de escaneamento. Para estes casos, basta transformar o isovalor em um valor real somando-o com 0.5 (38). Fazendo esta modificação, os algoritmos *Marching Cubes* (MC) e *Manifold Dual Marching Cubes* (MDMC)

passam a conseguir extrair isosuperfícies *manifold* também nesses casos.

O resultado das três técnicas se encontra na Figura 3.5. Apesar do MC e MMDC gerarem malha *manifold*, elas podem ainda assim alterar a topologia da isosuperfíce. Um exemplo disto está na Figura 3.7, que mostra que enquanto o *Surface Nets* (SN) produz algumas regiões pinçadas, o MC e o MDMC separam esses lugares, gerando partes de superfícies desacopladas entre si, o que também aparece na Figura 3.6.

Uma outra diferença que precisa ser ressaltada é que o MC tende a gerar triângulos mais finos e menores do que os demais algoritmos, possivelmente tornando a malha inadequada para posterior uso em métodos de elementos finitos, além de possivelmente criar artefatos visuais que distraem (38).



(a) *Marching Cubes* (b) *Surface Nets*

(c) *Manifold Dual Marching Cubes*

Figura 3.5: Extração de isosuperfície de uma carpa com $\sigma = 1150.5$.

Tendo estudado e analisado os algoritmos de extração de isosuperfície, e chegado o momento de escolher qual método seria levado adiante para adaptar ao caso de malhas de hexaedros levemente côncavos, optou-se pelo MC pelas seguintes razões:

- O MC é o algoritmo mais estudado e com mais variações e adaptações na área de extração de isosuperfície, o que sinaliza que ajustá-lo a esses tipos de malhas de hexaedros seja mais adequado.
- Enquanto nos algoritmos de *Dual Contouring* (DC) percorre-se as arestas bipolares internas, e é preciso verificar no *Manifold Dual Marching Cubes*

(a) *Marching Cubes*

(b) *Surface Nets*

(c) *Manifold Dual Marching Cubes*

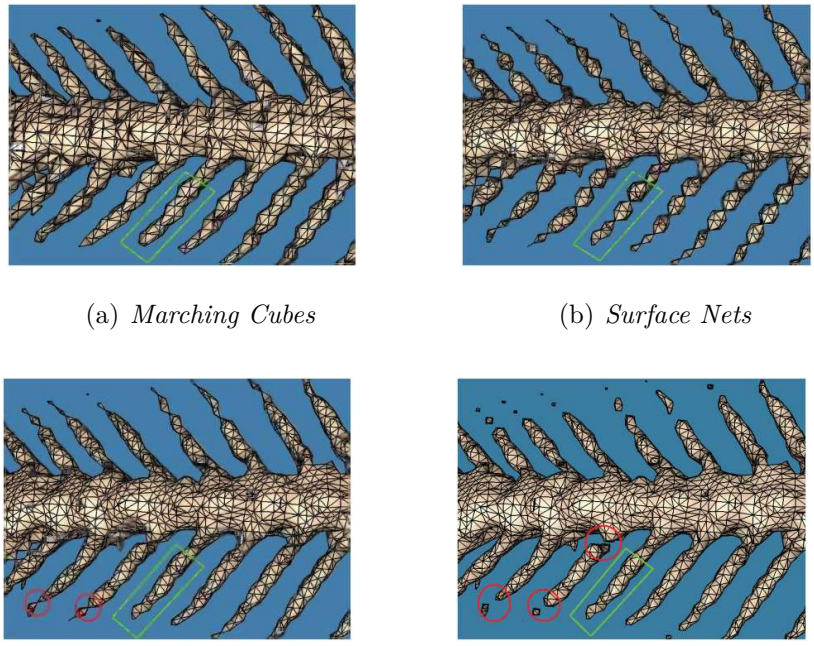
Figura 3.6: Extração de isosuperfície de uma carpa com $\sigma = 1150.5$. Visualização com *zoom*.

as células adjacentes para evitar o *não-manifold*, no MC as células são percorridas independentemente, o que facilita o desenvolvimento de um programa paralelizado, importante para execução rápida do mesmo.

– Pode ocorrer nos modelos de reservatório de petróleo que duas células adjacentes não compartilhem totalmente uma face, como exemplificado na Figura 5.5, e, consequentemente, não possuam em comum algumas arestas que compartilhariam caso se tratasse de uma malha regular. Há também o problema das células inativas. Estes fatores impactariam fortemente a base dos algoritmos do tipo DC: as arestas bipolares internas. Assim, adaptar um desses algoritmos seria mais difícil do que o MC.

– Não se verificou uma grande diferença de qualidade nem de estética no resultado dos algoritmos, considerando que as malhas foram geradas com muitos polígonos por conta dos *datasets* volumosos que foram utilizados. Apesar do SN ter um resultado esteticamente um pouco melhor, as malhas produzidas por este procedimento não são garantidamente *manifold*.

– O MC é mais simples de se implementar do que os demais algoritmos.



(a) *Marching Cubes* (b) *Surface Nets*

(c) *Dual Marching Cubes* (d) *Manifold Dual Marching Cubes*

Figura 3.7: Extração de isosuperfície de uma carpa com $\sigma = 1150.5$ feita por *Wenger* (38). Os dados foram subamostrados na proporção de $8:1$ para averiguar melhor as diferenças entre os resultados dos algoritmos.

4
Subdivisão Adaptativa e Superfícies Curvas

Em malha de hexaedros, a parte da isosuperfície que transpassa uma célula da grade de amostra volumétrica pode ser curva, mesmo no caso de uma grade regular e uniforme. Assim, os triângulos gerados pelo *Marching Cubes* (MC) podem não aproximar tão bem a isosuperfície por serem figuras planares.

Uma solução para melhor representá-la é substituir os triângulos por superfícies (*patches*) paramétricas. Uma técnica para se fazer isto é a subdivisão das primitivas de entrada em primitivas menores, criando novos pontos para serem manipulados e melhor representarem uma superfície curva definida por alguma fórmula matemática.

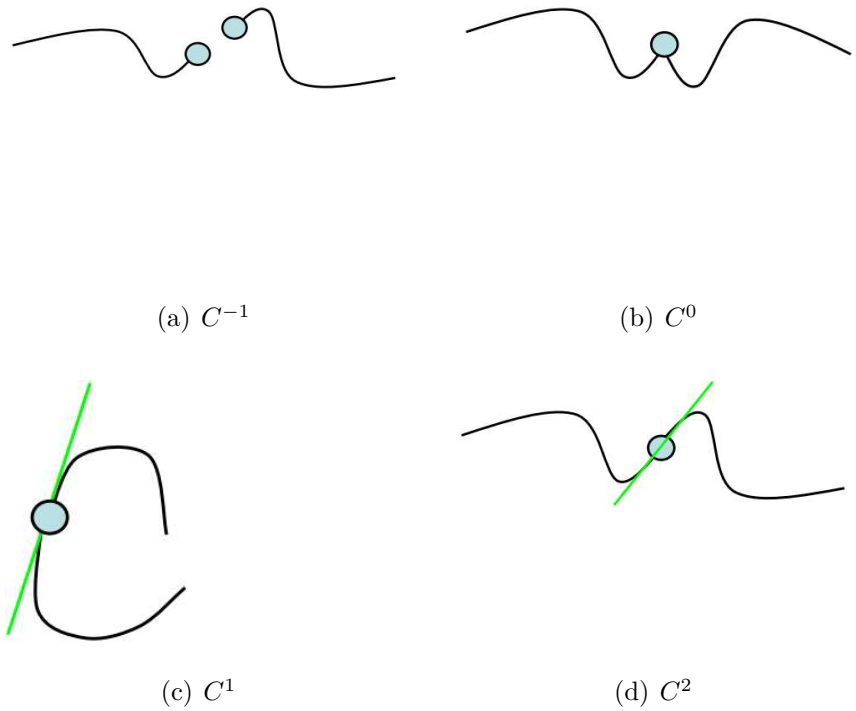
No caso do *OpenGL*, o processo de subdivisão (*tessellation*) consiste em três etapas no *pipeline* da API: o *tessellation control shader* (TCS), o *tessellation primitive generator* (TPG) e o *tessellation evaluation shader* (TES). O primeiro recebe as primitivas de entrada, manipula os vértices e possivelmente cria novos, além de definir o quanto a primitiva será subdividida na etapa seguinte. Ocorre, então, a subdivisão de fato por meio do TPG, uma parte não-programável do pipeline do *OpenGL*. Por fim, as primitivas resultantes desta etapa servem como entrada do passo seguinte, o TES, que transforma os pontos segundo a definição matemática de uma superfície curva.

Além do benefício de poder representar melhor curvas e silhuetas, a técnica de subdivisão pode ser utilizada para mostrar mais polígonos e detalhes quanto mais próximo estiver a câmera da malha e menos quanto ela estiver mais afastada, melhorando, assim, o desempenho do programa.

4.1
Continuidades Paramétrica e Geométrica

Um conceito importante a ser tratado e que permeia o assunto deste capítulo é o de continuidade, mais especificamente continuidade paramétrica C^n e continuidade geométrica G^n, esta última também conhecida como continuidade visual V^n. O n determina o grau de continuidade e de suavidade. Quanto maior for o n, maior a suavidade. Este tema surge a partir da necessidade de unir suavemente curvas ou superfícies vizinhas.

A continuidade paramétrica C^n está mais fortemente associada à parametrização do que apenas o formato da curva ou superfície. C^0 significa que as curvas paramétricas (ou superfícies) são contínuas, isto é, estão conectadas, não há um buraco ou pulo entre duas curvas. Se a junção de duas curvas paramétricas forma uma curva C^1, isto quer dizer que ela é C^0 e sua derivada também é contínua. Uma curva composta por outras duas é dita ser de continuidade C^2 quando for C^0, C^1 e quando a respectiva derivada de segunda ordem for contínua. E assim por diante em relação aos demais n. Esses diferentes graus de continuidade são demonstrados na Figura 4.1.



(a) C^{-1} (b) C^0

(c) C^1 (d) C^2

Figura 4.1: Exemplos de continuidade paramétrica (32). C^{-1} é referente à falta de continuidade.

As continuidades G^n e C^n são conceitos semelhantes, apesar deste último ser mais rigoroso e da G^n ser mais referente ao formato da curva ou da superfície, conforme pode ser visto na Figura 4.2. A composição de duas superfícies (ou curvas) é G^0 e C^0 quando elas estiverem ligadas ($C^0 = G^0$). Uma superfície é dita G^1 se, além de G^0, ela tiver continuidade da tangente, ou seja, as tangentes das duas superfícies que a formam forem paralelas. Ela é dita C^1 se, além de G^1, as tangentes tiverem o mesmo comprimento (módulo). Ademais, G^2 e G^3 referem-se à continuidade da curvatura (i.e., compartilham um mesmo centro de curvatura ao longo da interseção) e sua aceleração (derivada da curvatura em respeito ao comprimento de arco) entre duas superfícies conectadas, respectivamente.

As propostas de superfícies curvas *Phong Tessellation* (6) e *PN Trian-*

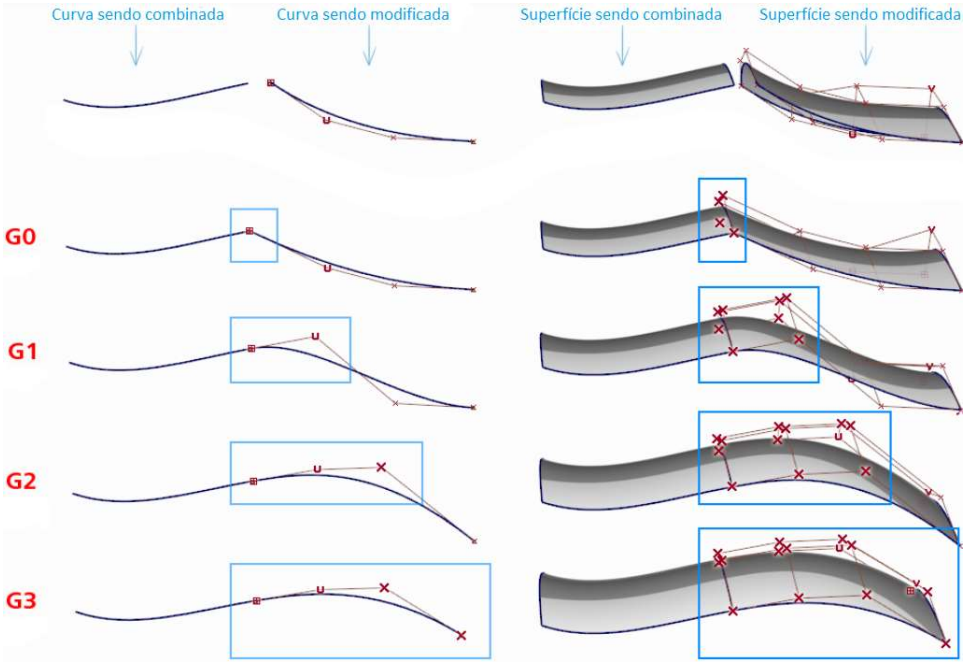


Figura 4.2: Demonstração de continuidade geométrica em diversos graus (2).

gles (37) não dependem de informações da vizinhança e possuem continuidade G^0. Já o PNG1 (15) utiliza informações dos triângulos adjacentes para criar uma superfíce G^1. A superfície de *Gregory* de *Walton e Meek* (13) conseguem a mesma continuidade utilizando superfícies quárticas definidas a partir de funções racionais, o que gera uma dificuldade de singularidade que ocorre nos vértices do triângulo que precisa ser tratada.

4.2 Superfícies paramétricas

O *Curved Point Normal Triangles*, ou apenas *PN Triangles*, criado por *Vlachos et. al* (36), refere-se a um método baseado em subdivisão para substituir triângulos por superfícies cúbicas triangulares de Bézier com variação quadrática da normal. A combinação dessas superfícies cúbicas é de continuidade G^0.

O objetivo do *PN Triangles* (PNT) foi melhorar a qualidade de malhas em programas de entretenimento digital artísticos, como jogos, não no nível de qualidade de um designer, mas suavizando vincos e silhuetas. Os termos "*Point*" e "*Normal*" foram escolhidos porque as coordenadas e as normais são as únicas informações necessárias referentes aos vértices dos triângulos de entrada para a implementação do PNT.

Sejam (u, v, w) as coordenadas baricêntricas de um vértice de um triân-

gulo, com $u, v, w \geq 0$ e $w = 1 - u - v$, a superfície cúbica triangular de Bézier b é definida da seguinte forma:

$$\begin{aligned} b(u,v,w) = & \sum_{i+j+k=3} b_{i,j,k} \frac{3!}{i!j!k!} u^i v^j w^k \\ = & \; b_{300}u^3 + b_{030}v^3 + b_{003}w^3 \\ + & \; b_{210}3u^2v + b_{120}3v^2u + b_{201}3u^2w \\ + & \; b_{021}3v^2w + b_{102}3w^2u + b_{012}3w^2v \\ + & \; b_{111}6uvw \end{aligned}$$

onde os coeficientes $b_{300}, b_{030}, b_{003}, b_{210}, b_{120}, b_{201}, b_{021}, b_{102}, b_{012}$ e b_{111} são chamados de pontos de controle e determinam a forma da superfície.

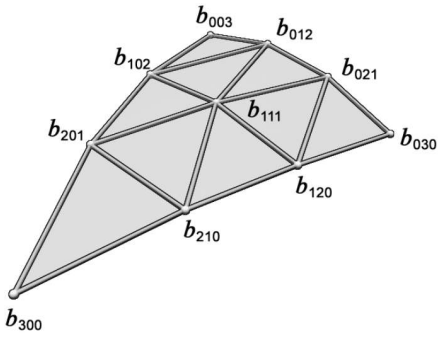


Figura 4.3: Pontos de controle de uma superfície cúbica de Bézier (36).

Os coeficientes b_{300}, b_{030} e b_{003} são os vértices originais P_1, P_2 e P_3 do triângulo de entrada. Para calcular cada um dos pontos de controle $b_{210}, b_{120}, b_{201}, b_{021}, b_{102}$ e b_{012}, computa-se primeiramente um ponto intermediário na aresta do triângulo com distância de 1/3 do vértice original mais próximo (e.g., $\frac{2}{3}P_i + \frac{1}{3}P_{i+1}$). O próximo e último passo é projetar ortogonalmente o ponto calculado no plano definido pelo vértice original mais próximo e a respectiva normal.

Tal procedimento é expresso matematicamente do jeito a seguir:

$$
\begin{aligned}
b_{210} &= (2P_1 + P_2 - w_{12}N_1)/3, \\
b_{120} &= (2P_2 + P_1 - w_{21}N_2)/3, \\
b_{201} &= (2P_1 + P_3 - w_{13}N_1)/3, \\
b_{021} &= (2P_2 + P_3 - w_{23}N_2)/3, \\
b_{102} &= (2P_3 + P_1 - w_{31}N_3)/3, \\
b_{012} &= (2P_3 + P_2 - w_{32}N_2)/3,
\end{aligned}
$$

onde $w_{ij} = (P_j - P_i) \cdot N_i$.

Por fim, para encontrar o ponto de controle central b_{111}, calcula-se primeiramente o centro $C = (P_1 + P_2 + P_3)/3$ e a média dos coeficientes intermediários $M = (b_{210} + b_{120} + b_{201} + b_{021} + b_{102} + b_{012})/6$. Depois, computa-se a diferença entre M e C e desloca-se a metade desta diferença a partir de M. De outro modo, $b_{111} = M + (M - C)/2$.

A normal da geometria gerada com a utilização do PNT não é garantida de ser contínua, o que pode causar defeitos visuais. *Vlachos et. al* (36) propõem, então, calcular a normal por meio de uma superfície quadrática independente que traria essa continuidade para efeitos de *shading*. Isto significa que esta normal não é a verdadeira normal da geometria. Trata-se de uma superfície quadrática triangular de Bézier. Segue a definição dada originalmente no artigo:

$$
\begin{aligned}
n(u,v,w) &= \sum_{i+j+k=2} n_{ijk} u^i v^j w^k \\
&= n_{200}u^2 + n_{020}v^2 + n_{002}w^2 \\
&+ n_{110}uv + n_{011}vw + n_{101}uw.
\end{aligned}
$$

Entretanto, a equação acima está incorreta e foi, infelizmente, copiada por vários outros artigos. Conforme exposto posteriormente em uma apresentação realizada por dois dos autores do documento original, *Vlachos e Peters* (37), a fórmula correta é a seguinte:

$$\begin{aligned} n(u,v,w) &= \sum_{i+j+k=2} n_{ijk} \frac{2!}{i!j!k!} u^i v^j w^k \\ &= n_{200}u^2 + n_{020}v^2 + n_{002}w^2 \\ &+ n_{110}2uv + n_{011}2vw + n_{101}2uw \end{aligned}$$

Para deduzir esta equação, é preciso antes expressar a fórmula genérica de uma superfície triangular de Bézier, que é a seguinte:

$$p(u,v,w) = \sum_{i+j+k=n} p_{ijk} B^n_{ijk},$$

onde n é o grau da superfície, p_{ijk} é um ponto de controle e B^n_{ijk} é um polinômio de Bernstein, definido por $B^n_{ijk} = \frac{n!}{i!j!k!} u^i v^j w^k$. Assim, percebe-se que estava faltando o termo $\frac{2!}{i!j!k!}$ na proposta inicial da superfície quadrática da normal.

Para encontrar os coeficientes intermediários n_{110}, n_{011} e n_{101}, computa-se a média das normais dos vértices da aresta e reflete-a em relação ao plano perpendicular à aresta. Em termos matemáticos:

$$n_{110} = \frac{N_1 + N_2 - v_{12}(P_2 - P_1)}{||N_1 + N_2 - v_{12}(P_2 - P_1)||},$$

$$n_{011} = \frac{N_2 + N_3 - v_{23}(P_3 - P_2)}{||N_2 + N_3 - v_{23}(P_3 - P_2)||},$$

$$n_{101} = \frac{N_3 + N_1 - v_{31}(P_1 - P_3)}{||N_3 + N_1 - v_{31}(P_1 - P_3)||},$$

onde $v_{ij} = 2\frac{(P_j - P_i)\cdot(N_i + N_j)}{(P_j - P_i)\cdot(P_j - P_i)}$.

Uma outra forma de calcular a normal é por meio de uma interpolação linear, ou seja, $n(u,v,w) = \frac{n_{300}u + n_{030}v + n_{003}w}{||n_{300}u + n_{030}v + n_{003}w||}$. Entretanto, assim não se capta as inflexões geométricas, o que implica em um *shading* irreal.

4.3

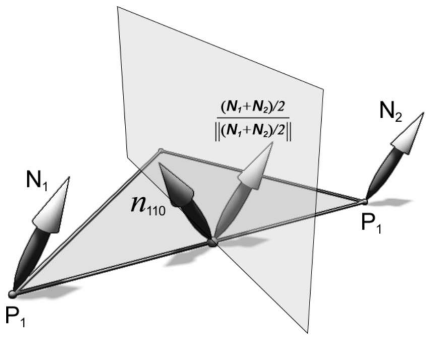


Figura 4.4: Reflexão da média das normais dos vértices uma aresta para cálculo de um coeficiente intermediário da superfície quadrática n. (36).

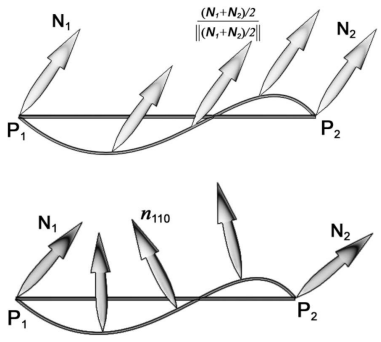


Figura 4.5: Comparação entre interpolação linear e interpolação quadrática (36).

Subdivisão Adaptativa

Como se viu na seção anterior, o *Curved Point Normal Triangles* (36) utiliza subdivisão para a reprodução de superfícies curvas, isto é, os triângulos planares gerados pelo *Marching Cubes* são substituídos por uma superfície curva composta e aproximadas por triângulos menores. Surge, então, a questão: o quanto subdividir as primitivas de entrada do algoritmo para bem aproximar as superfícies paramétricas?

Este questionamento é importante, pois, para aplicações em tempo real, é preciso sim considerar os aspectos visuais e estéticos, mas também o desempenho, tanto de velocidade de execução quanto de memória. A subdivisão adaptativa serve justamente para adequar o processo em vista de um custo benefício melhor, tentando gerar apenas a quantidade necessária de triângulos para bem aproximar a curva. Quanto mais curva for a superfície, maior a quantidade de triângulos será preciso para aproximá-la.

Um método simples e eficaz para se alcançar tal objetivo é pela comparação das normais dos vértices de cada aresta do triângulo. O processo realizado pelo *PN Triangles* (PNT) transforma cada aresta do triângulo original em uma

curva. Pressupõe-se que quanto mais as normais dos vértices que definem a aresta apontarem para a mesma direção e sentido, mais plano e semelhante ao triângulo original é a nova superfície, enquanto quanto menos parecidos forem quanto ao sentido e direção, mais curva é a superfície.

O primeiro passo é calcular o cosseno entre as duas normais da aresta. Este valor serve como medida do quanto as normais são semelhantes ou não. Supondo normais unitárias, basta utilizar o produto interno para encontrar o resultado. Seja $t = n_i \cdot n_j$, então $t \in [-1, 1]$. Quanto mais próximo de 1, mais elas apontam na mesma direção e sentido. E quanto mais próximo de -1, menos apontam.

O parâmetro t, então, pode ser utilizado em alguma função para calcular o quanto uma aresta deve ser subdividida. Um exemplo é transformar t em $t' = (-t + 1)0.5$, de modo que $t' \in [0, 1]$ e quanto maior for t' menos as normais estão na mesma direção e sentido, e então usá-lo em uma função linear, como $l(t') = kt'$, onde k é o número máximo de subdivisões que se pretender ter por aresta.

Um outro exemplo é a utilização de uma função exponencial, transformando primeiro t em $t' = (t + 1)0.5$ e depois t' em $t'' = (1 - t)0.45 + t'$, que pode ser usado, então, em $e(t'') = 6^{1/t''} - 5$, cujo resultado define o quanto uma aresta deve ser subdivida em. A Tabela 4.1 exemplifica o cálculo de subdivisões com essa função. A proposta da utilização de uma função exponencial tem o intuito de tornar a subdivisão adaptativa ainda mais sensível às silhuetas e contornos de modo a direcionar mais recursos computacionais a essas regiões.

t	***t'***	***t"***	***e(t")***	**Resultado**
-1.000	0.000	0.450	48.607	20.000
-0.750	0.125	0.519	26.626	20.000
-0.500	0.250	0.588	16.111	16.111
-0.250	0.375	0.656	10.337	10.337
0.000	0.500	0.725	6.839	6.839
0.250	0.625	0.794	4.558	4.558
0.500	0.750	0.863	2.984	2.984
0.750	0.875	0.931	1.849	1.849
1.000	1.000	1.000	1.000	1.000

Tabela 4.1: Demonstração do número de subdivisões para diferentes valores de t. A coluna *Resultado* é o mínimo entre $e(t'')$ e k, o número de subdivisões máximo, a fim de respeitar o limite. Neste exemplo, $k = 20$.

Outro método, também simples, mas dependente da câmera, utilizado por *Boubekeur e Alexa* no *Phong Tessellation* (6), consiste em calcular a diferença normalizada entre o vértice e a posição da câmera, realizar o produto interno entre esta direção e a normal do vértice para medir a inclinação pela qual se vê

a superfície, subtrair o módulo do resultado de 1 e multiplicar isto pelo número de subdivisões máximo que se deseja ter:

$$s = \left(1 - \left|N_i \cdot \frac{C - P_i}{||C - P_i||}\right|\right) k,$$

onde P_i é um vértice do triângulo, N_i a respectiva normal, C a posição da câmera e k o número máximo de subdivisões.

Entretanto, s é um valor por vértice, enquanto a subdivisão ocorre por aresta. Assim, para cada aresta, é preciso calcular o s associado a cada um dos vértices e escolher o maior entre os dois.

É importante ressaltar que é preciso estar atento aos limites tanto inferiores quanto superiores do quanto as arestas dos triângulos devem ser subdivididas. No *OpenGL*, o parâmetro associado à quantidade de subdivisões de uma aresta, na verdade indica quantos segmentos devem formá-la. Por exemplo, o valor 1 significa um segmento de reta, que é a própria aresta original, ou seja, não há subdivisões. Já o valor 2 representa dois segmentos e, portanto, uma subdivisão, e assim por diante.

Deste modo, o limite inferior é 1 à priori, significando que não deve haver subdivisões naquela aresta, enquanto o limite superior é um escalar $k > 1$ que não seja grande o suficiente para pesar a aplicação.

Ademais, destaca-se a necessidade de que triângulos adjacentes devem compartilhar as normais referentes à aresta que possuem em comum, que também deve ser subdividida pelo mesmo montante. Caso contrário, haverá buracos e interseção entre polígonos da superfície gerada.

Além de especificar o quanto as arestas devem ser subdivididas, o *OpenGL* permite definir o quanto o triângulo deve ser dividido interiormente. É um valor associado a quantos anéis interiores devem ser gerados. Uma escolha possível para este valor é escolher o maior entre os três paramêtros definidos para as arestas. Uma outra é a média entre tais valores. A Figura 4.6 demonstra como é um triângulo é subdivido segundo os fatores escolhidos.

Existem outros métodos e definições matemáticas de superfícies que foram criados com intuitos semelhantes, como é o caso do *Phong Tessellation* (6), que utiliza uma superfície quadrática e interpolação linear da normal. Entretanto, esta combinação não lida adequadamente com superfícies com inflexão.

Há também estratégias que dependem do conhecimento de triângulos adjacentes, como o algoritmo de *Loop* (23). Entretanto, tal dependência, especialmente considerando uma grade não-regular e não-uniforme de hexaedros

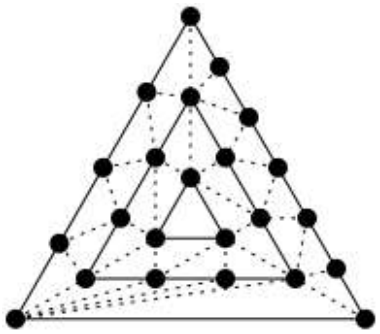


Figura 4.6: Exemplo de subdivisão de uma superfície triangular utilizando o *tessellation shader* do *OpenGL*. Os valores de subdivisão escolhidos para as arestas foram $4, 1$ e 6, e 5 para a parte interna do triângulo (18).

levemente côncavos, pode ser custosa computacionalmente e inviabilizar uma aplicação de ser executada em tempo real. O *PN Triangles* (PNT) e o *Phong Tessellation* não precisam de informações da vizinhança.

Tendo em vista a complexidade e o interesse em aplicações em tempo real, nesse trabalho optou-se por utilizar o *PN Triangles*, com uma nova proposta para o cálculo de suas normais.

4.4 Uso de Envelope para Extração de Normais

Neste trabalho, a superfície quadrática do PNT não apresentou resultados satisfatórios. Acredita-se que as razões sejam devido à independência entre as superfícies de geometria e normal, pela normal não estar definida unicamente, pela inadequação da interpolação quadrática no caso de curvas quadráticas ou curvas cúbicas sem inflexão e pela suposição de que as normais são suficientemente coplanares ao vetor que representa a aresta (21).

A estratégia tomada foi, então, a do envelope, cujo resultado é ilustrado pela Figura 4.8. A ideia consiste em calcular uma outra superfície, temporária, que não será efetivada como saída, ou seja, não gerará uma geometria que será de fato visualizada. Essa superfície temporária serve como auxílio para cálculos da normal da real superfície. A superfície temporária pode ser entendida como uma versão inflada da verdadeira, o que possibilita que sua normal seja calculada de forma mais natural, simples e eficiente pela diferença normalizada entre os pontos de ambas as superfícies. Com esta técnica, busca-se uma consistência maior nas normais dos triângulos gerados em células adjacentes. Exemplos de normais calculadas por esse método são mostradas pela Figura 4.7.

Os coeficientes $b_{300}, b_{030}, b_{003}, n_{200}, n_{020}$ e n_{002}, no caso da superfície real, consistem nos vértices e normais do triângulo original, respectivamente. No

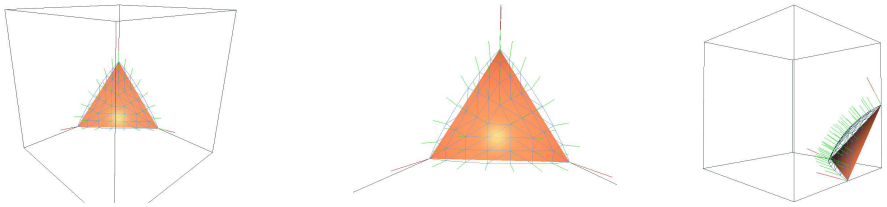


Figura 4.7: Demonstração da geometria e das normais de resultados obtidos do *Marching Cubes* (MC) com o PNT modificado pela técnica do envelope. Um único vértice é positivo, enquanto os demais são negativos. As linhas azuis são a visualização *wireframe* da superfície paramétrica curva. O triângulo coral é o gerado pelo MC. As linhas verdes são as normais da superfície curva e as vermelhas do triângulo de entrada.

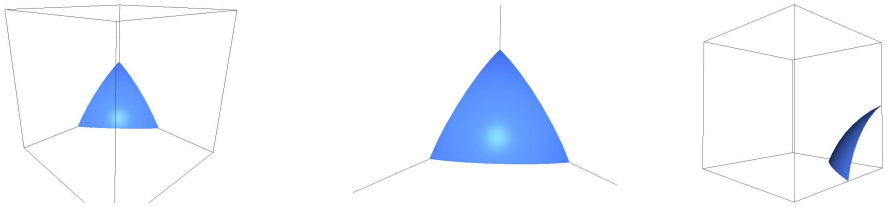


Figura 4.8: Resultados obtidos do MC com o PNT modificado pela técnica do envelope.

caso da superfície temporária auxiliar, as normais são iguais também, mas os vértices são o resultado do deslocamento das posições originais em direção da respectiva normal. As normais da superfície de saída são calculadas na última etapa do *tessellation shader* pela normalização da diferença entre os pontos da geometria associados às mesmas coordenadas (u, v, w) de ambas as superfícies.

5
Malhas de Hexaedros Levemente Côncavos

As malhas de maior interesse deste trabalho são as utilizadas para modelar e simular reservatórios de petróleo. Neste capítulo, introduz-se os modelos de reservatório, descreve-se como são essas malhas de hexaedros levemente côncavos e como o método deste trabalho é adaptado e aplicado a elas.

5.1
Modelo de reservatório de petróleo

Segundo *Dake* (11), um reservatório de petróleo é gerado natural e gradualmente pela acumulação de hidrocabornetos dentro de rochas porosas, formadas por meio da sedimentação ao longo de milhões de anos. O processo de extração de petróleo se dá por meio da perfuração de um reservatório, aproveitando-se das pressões naturais presentes nas rochas. A comódite deve, então, ser encaminhada até uma linha de produção para fins de armazenamento, processamento e transporte.

De acordo com *Abraham* (1), simuladores numéricos estão sendo cada vez mais utilizados e demandados pela indústria de petróleo a fim de otimizar o lucro minimizando o custoso processo de extração. Isto se dá pelo planejamento que é possível fazer graças às informações e previsões realizadas pelas simulações.

Um modelo comum utilizado pela indústria petrolífera para a simulação de um reservatório de petróleo é definido por uma malha discreta de hexaedros organizados topologicamente em uma grade tridimensional, conforme ilustrado na Figura 5.1. Cada uma das células são identificadas por índices (i, j, k), o que permite encontrar a vizinhança somando ou subtraindo 1 de uma das dimensões, isto é, de um dos índices (1).

Os hexaedros que compõem a malha do modelo são, em geral, de tamanhos distintos e compostos por faces não planares, de modo que células vizinhas podem não compartilhar faces como em uma grade regular uniforme. Isto é uma representação das falhas geológicas decorrentes de descontinuidades na elevação. Ademais, algumas células podem ser categorizadas como inativas para simulações referentes a certas propriedades, podendo gerar conjuntos desconectados e irregulares de células, conforme pode ser visto na Figura 5.2.

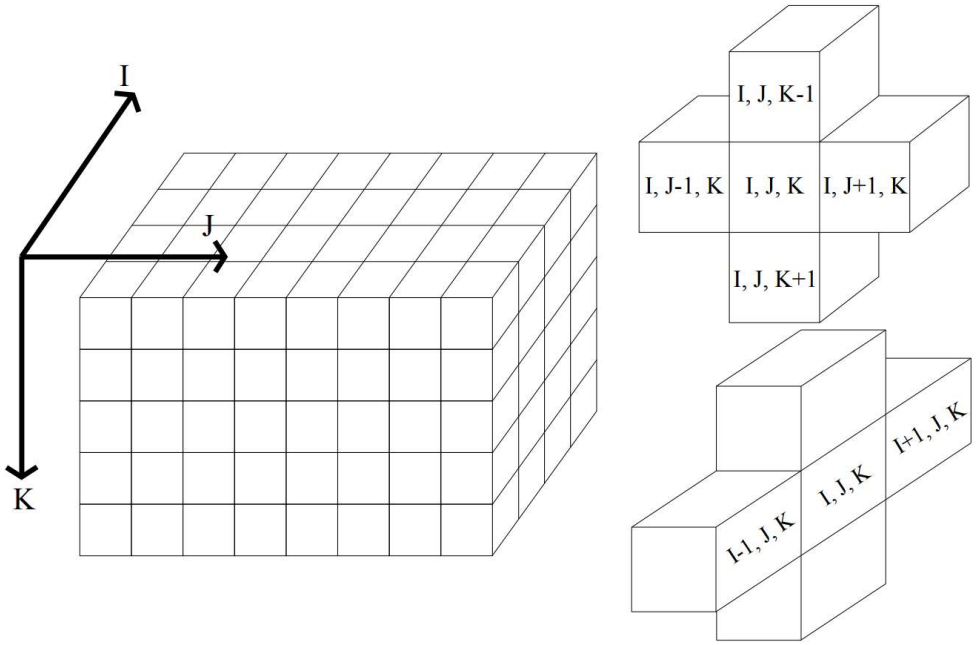


Figura 5.1: Organização topológica da malha usada em modelos de simulação de reservatório de petróleo (1).

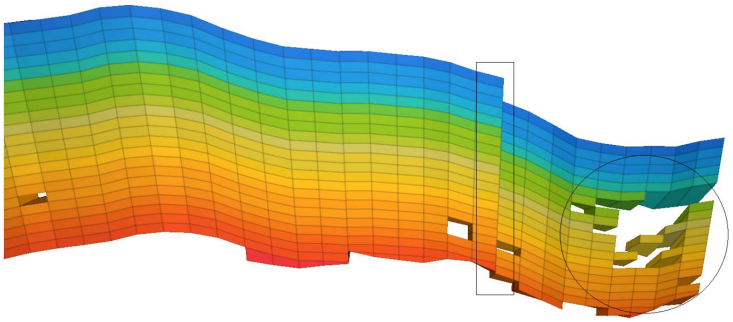


Figura 5.2: Demonstração de irregularidades de um modelo de reservatório por meio de um recorte lateral. Células inativas estão circuladas à direita e uma região de falha geológica está marcada à esquerda (20).

Há diversos elementos físicos do reservatório que podem ser analisados, como pressão e saturação de óleo. A simulação de um modelo associa tais propriedades às células do modelo, tornando-as amostras de um campo escalar. Essas amostras podem ser calculadas ao longo de vários passos de tempo pela simulação.

Como os valores das propriedades estão presentes apenas nas células e o *Marching Cubes* (MC) espera que estejam nos vértices, é preciso realizar

um pré-processamento para calcular os escalares nos vértices. Isto é feito da seguinte forma: para cada vértice da malha, computa-se a média dos escalares das células que o contêm.

No que diz respeito a estruturas de dados, o MC comum utiliza uma matriz tridimensional que guarda os valores amostrados nos vértices, as posições dos mesmos e suas respectivas normais, e representa a grade regular uniforme. Já na versão adaptada para as malhas de modelos de reservatório, é necessário um vetor (*array*, ou uma lista) com todas as informações de cada vértice (posição, normal, escalar), além de uma matriz tridimensional que guarda para cada célula 8 índices referentes ao vetor de vértices, indicando quais são os vértices que compõem o hexaedro. É preciso fazer essa alteração porque as células adjacentes podem não compartilhar uma face.

5.2 Cálculo das Normais

O *Marching Cubes* e o método de diferenças finitas centrais funcionam em grades regulares e uniformes. Entretanto, isto não é mais o caso quando se considera uma malha de hexaedros de faces não planares. Esta seção se destina a discutir quais são as mudanças necessárias para calcular as normais considerando todas as particularidades de um modelo de reservatório, citadas na seção anterior, que já tratou de outras modificações que precisaram ser realizadas.

A primeira mudança é referente à necessidade de adaptar o espaço do mundo para um espaço paramétrico que seja regular e uniforme onde os cálculos possam ser devidamente realizados, conforme ilustrado na Figura 5.3. Para fazer isto, a ideia consiste em utilizar a matriz Jacobiana, tal como feito por *Barroso* (4). Ela é composta pelas derivadas parciais de primeira ordem de uma função vetorial v e aproxima linearmente v da melhor forma possível na vizinhança de um ponto específico, ou seja, localmente.

Supondo que a associação entre os espaços seja uma bijeção contínua no interior do domínio da simulação, pode-se definir as seguintes relações (4):

$$f_s = f_x x_s + f_y y_s + f_z z_s,$$

$$f_t = f_x x_t + f_y y_t + f_z z_t,$$

$$f_p = f_x x_p + f_y y_p + f_z z_p,$$

onde f_x é a derivada do campo escalar f na direção X. É possível também fazer o mapeamento inverso, isto é, o retorno do espaço paramétrico para o

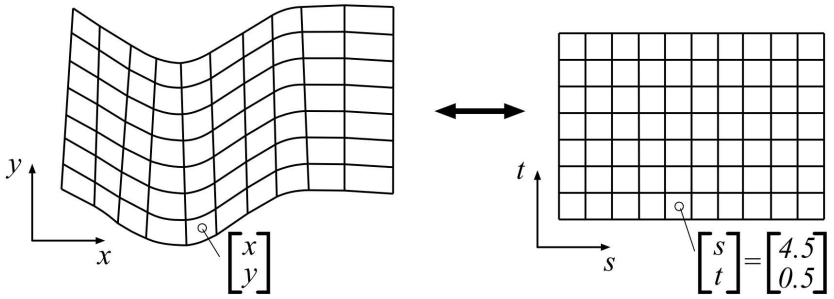


Figura 5.3: Mapeamento de uma posição no espaço do mundo para o espaço paramétrico no caso de duas dimensões (4).

espaço do mundo, como formulado a seguir:

$$f_x = f_s s_x + f_t t_x + f_p p_x,$$
$$f_y = f_s s_y + f_t t_y + f_p p_y,$$
$$f_z = f_s s_z + f_t t_z + f_p p_z.$$

As equações acima podem ser reescritas de modo mais compacto por meio da notação matricial:

$$[\nabla f]_{(s,t,p)} = J[\nabla f]_{(x,y,z)},$$
$$[\nabla f]_{(x,y,z)} = J^{-1}[\nabla f]_{(s,t,p)},$$

$$J = \begin{pmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \\ \frac{\partial x}{\partial p} & \frac{\partial y}{\partial p} & \frac{\partial z}{\partial p} \end{pmatrix} = \begin{pmatrix} [\nabla x]_{(s,t,p)} & [\nabla y]_{(s,t,p)} & [\nabla z]_{(s,t,p)} \end{pmatrix},$$

$$J^{-1} = \begin{pmatrix} \frac{\partial s}{\partial x} & \frac{\partial t}{\partial x} & \frac{\partial p}{\partial x} \\ \frac{\partial s}{\partial y} & \frac{\partial t}{\partial y} & \frac{\partial p}{\partial y} \\ \frac{\partial s}{\partial z} & \frac{\partial t}{\partial z} & \frac{\partial p}{\partial z} \end{pmatrix} = \begin{pmatrix} [\nabla s]_{(x,y,z)} & [\nabla t]_{(x,y,z)} & [\nabla p]_{(x,y,z)} \end{pmatrix},$$

onde J é a matriz Jacobiana, J^{-1} a sua inversa, $[\nabla x]_{(s,t,p)}, [\nabla y]_{(s,t,p)}$ e $[\nabla z]_{(s,t,p)}$ os gradientes das direções X, Y e Z no sistema de coodenadas (s,t,p), $[\nabla s]_{(x,y,z)}, [\nabla t]_{(x,y,z)}$ e $[\nabla p]_{(x,y,z)}$ os gradientes das direções S, T e P no sistema de coordenadas (x,y,z).

No caso concreto das normais nos vértices da malha, que estão no espaço do mundo, calcula-se, primeiramente, os gradientes das direções X, Y e Z, que formarão a matriz Jacobiana, e a normal no espaço paramétrico. O próximo passo é, então, usar a inversa da matriz para transformar a normal do espaço paramétrico na normal do espaço do mundo. Matematicamente, essa transformação é expressa da seguinte forma:

$$n_{(x,y,z)} = J^{-1} n_{(s,t,p)},$$

onde $n_{(s,t,p)}$ é a normal no espaço paramétrico e $n_{(xyz)}$ é a normal no espaço do mundo.

O cálculo da normal no espaço paramétrico se dá por meio do método de diferenças finitas centrais, já discutido na seção 3.2. Entretanto, algumas alterações devem ser feitas. Como nas malhas de células hexaédricas levemente côncavas de modelos de reservatórios há geometria irregular, adapta-se a técnica para utilizar o vértice mais próximo da célula vizinha, conforme demonstrado pelas Figuras 5.4 e 5.5. Já nos casos cuja célula adjacente é inativa, trata-se da mesma forma das bordas.

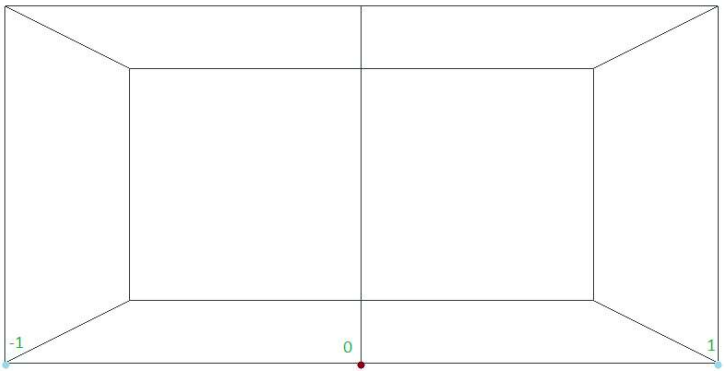


Figura 5.4: Exemplo de duas células adjacentes que compartilham toda uma face. Escolhe-se os vértices identificados por 1 e -1 para calcular a normal do vértice 0.

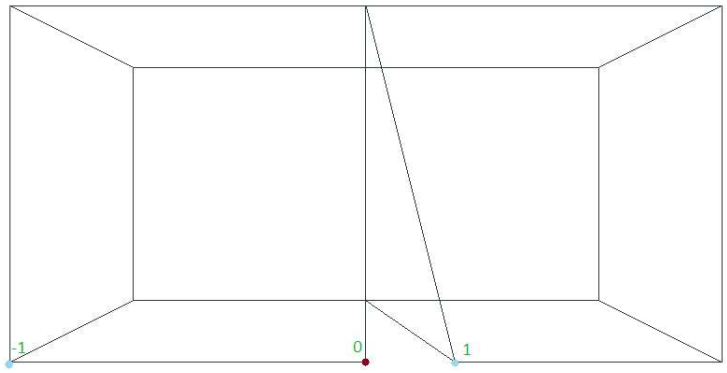


Figura 5.5: Exemplo de duas células adjacentes que não compartilham totalmente uma face. Escolhe-se os vértices identificados por 1 e -1 para calcular a normal do vértice 0.

6
Implementação e Resultados

Neste capítulo, os resultados obtidos são expostos e relatados. Os experimentos foram realizados em uma máquina com um processador i7-9750H e uma placa gráfica NVIDIA GeForce RTX 2060.

A linguagem de programação utilizada foi C++14, com a utilização extensiva das bibliotecas *Standard Template Library* (STL), composta de diversos algoritmos e estruturas, e *OpenGL Mathematics* (GLM), que é de matemática voltada para Computação Gráfica (CG). Além disso, a renderização é realizada com o uso da API de CG *Open Graphics Library* (*OpenGL*) e a sua linguagem de *shaders OpenGL Shading Language* (GLSL).

O *Compute Shader* do *OpenGL* é empregado para computar em paralelo o *Marching Cubes* (MC). O *Tessellation Shader* é usado para a substituir os triângulos gerados pelo MC pelas superfícies paramétricas curvas. Devido ao uso desses *shaders*, exige-se uma versão moderna e atualizada do *OpenGL* a partir da 4.3.

6.1
Uso de envelope para cálculo da normal

Nesta seção, são apresentados alguns resultados obtidos com a técnica do envelope. Para isto, são feitas comparações entre o *PN Triangles* original e o modificado pela proposta, e também entre superfície extraída com e sem a subdivisão adaptativa.

Lee e Jen (21) explicam que a interpolação quadrática pode ser inadequada para curvas quadráticas ou cúbicas sem inflexão. A Figura 6.1 mostra o tipo de defeito visual que pode aparecer com o uso do *PN Triangles*, mais especificamente da superfície quadrática proposta para o cálculo das normais. Já na Figura 6.2, é possível ver o resultado com a modificação da técnica do envelope sem os artefatos visuais.

A Figura 6.3 ilustra a comparação entre uma superfície extraída com e sem a subdivisão para demonstrar o quanto a subidivisão adaptativa realmente suaviza os contornos e silhuetas. Para tanto, foi utilizada uma extração de baixa resolução com as dimensões 20x20x20.

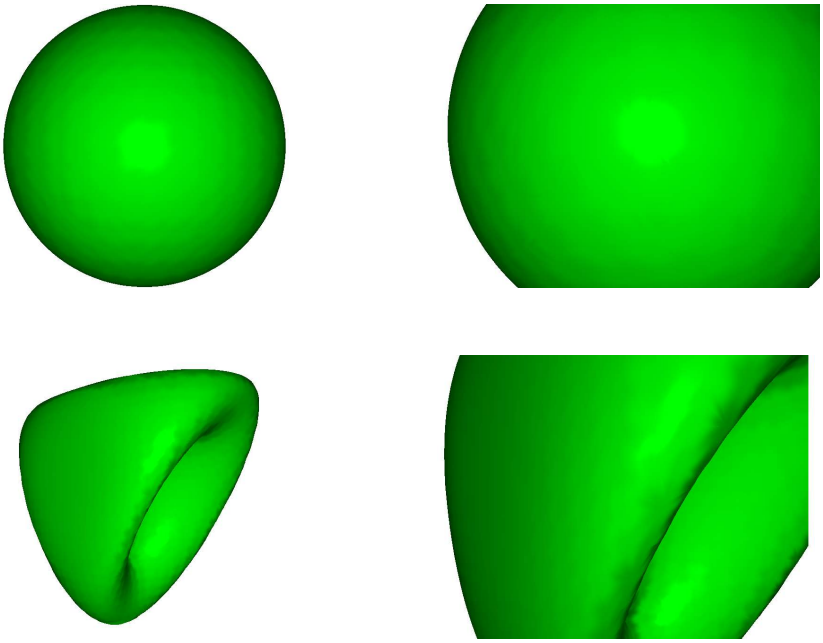


Figura 6.1: Renderização de isosuperfícies com o *PN Triangles* original. As imagens à direita são a versão com *zoom* das figuras à esquerda.

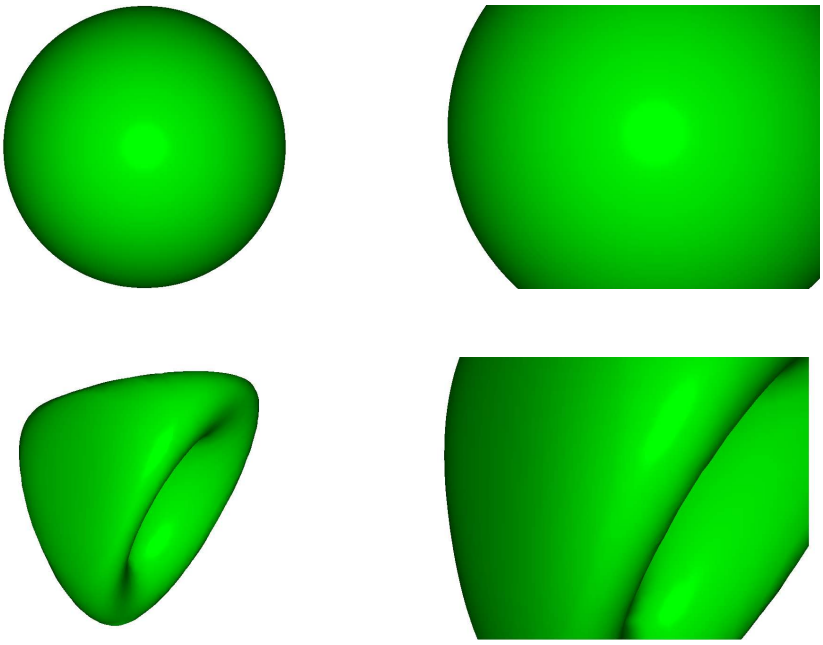


Figura 6.2: Visualização de isosuperfícies com o *PN Triangles* e a técnica do envelope. As imagens à direita são a versão com *zoom* das figuras à esquerda.

Percebeu-se uma grande diferença de desempenho entre o MC implementando no CPU sem programação paralela e no GPU com o *Compute Shader*. Para as funções implícitas testadas e dimensões de até 40 por eixo, o MC pode demorar quase um segundo no CPU e menos de um microssegundo no GPU,

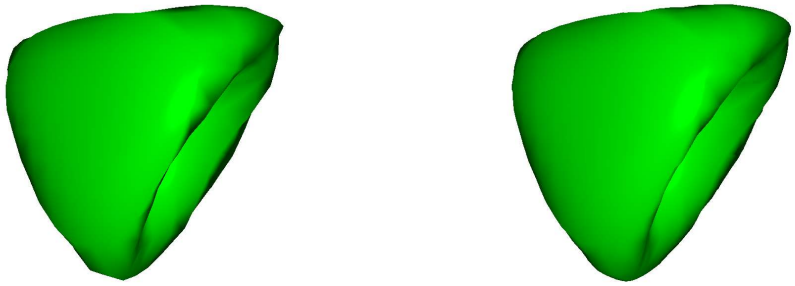


Figura 6.3: Comparação entre uma superfície extraída com e sem subdivisão adaptativa. A superfície à esquerda é sem qualquer subdivisão, enquanto à direita é com a subdivisão proposta.

como pode ser verificado na Tabela 6.1.

Superfície	Dimensões	Tempo CPU	Tempo GPU	# Triângulos
Esfera	20x20x20	370.57	0.3204	2482
Torus	30x30x30	604.00	0.2831	4688
Caixa Suave	35x35x35	483.60	0.3491	2648
De gênero 2	40x40x40	954.60	0.3679	4980

Tabela 6.1: Comparativo entre os tempos do MC no CPU e no GPU.

6.2 Aplicação em modelos de reservatório de petróleo

A primeira parte da solução consiste em ler o isovalor e o modelo de reservatório passados como parâmetro para o programa. O segundo passo consiste em percorrer o modelo para trazer as amostras das células para os vértices por meio do cálculo da média. Então, monta-se a malha tridimensional de índices e o vetor de vértices, conforme explicado na Seção 5.1.

Em seguida, computa-se a matriz Jacobiana para, com ela, encontrar as normais nos vértices dos hexaedros. Isto é feito considerando a topologia da grade do modelo em particular, que podem ter os eixos I, J e K invertidos ou não.

Posteriormente, executa-se um *Compute Shader* para computar de forma rápida um algoritmo que é uma parte do MC apenas para descobrir quantos triângulos serão gerados, informação necessária para não separar um pedaço de memória (*buffer*) muito grande para guardar a saída do verdadeiro MC, que é o procedimento executado logo em seguida, também por meio de um *Compute Shader*.

Por fim, a renderização é realizada com o auxílio de um *Tessellation Shader*, responsável por subdividir os triângulos de saída do passo anterior e

substituí-los por superfícies curvas. Todo o processo encontra-se sintetizado na Figura 6.4.

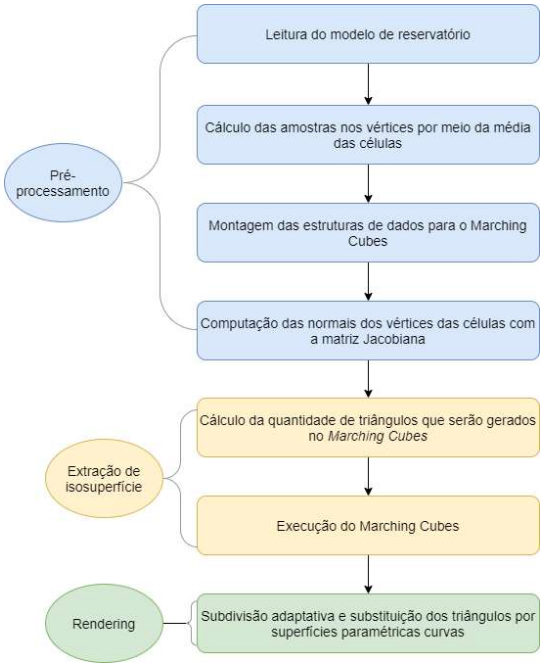


Figura 6.4: Fluxo de execução da solução desenvolvida.

Alguns dos resultados obtidos podem ser observados pelas Figuras 6.5, 6.6 e 6.7. Os modelos utilizados foram o Caixa Regular, o Pituba e o Proxy. A malha do primeiro imita uma grade regular uniforme, enquanto as demais possuem irregularidades. Optou-se por utilizar o último passo de tempo da simulação para a demonstração dos resultados.

Informações referentes aos resultados estão sumarizadas na Tabela 6.2. Percebe-se que o MC implementado com computação paralela apresentou alto desepenho com todos os modelos. Como de se esperar, o MC é mais rápido no caso da malha do modelo Caixa Regular e menos no caso do Proxy por conta da diferença nas dimensões e na quantidade de triângulos gerados.

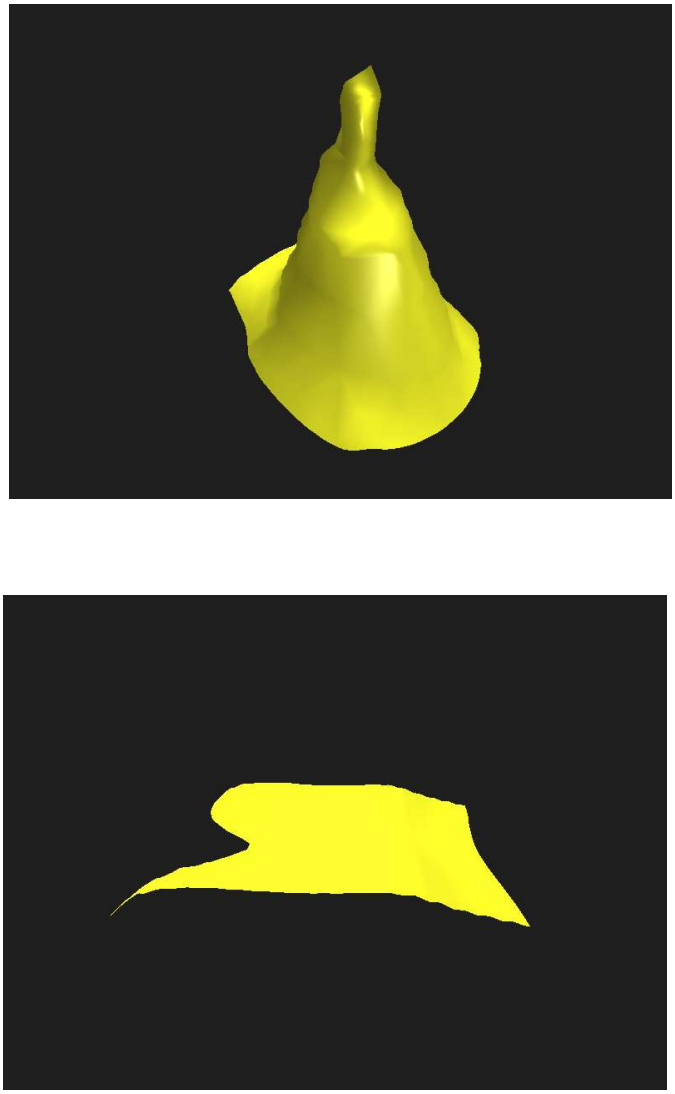


Figura 6.5: Visualização resultante do modelo Caixa Regular. O campo escalar escolhido é referente à saturação de óleo. Os isovalores são 0.135 e 0.309 à esquerda e à direita, respectivamente.

Modelo	**Dimensões**	**Isovalor**	**Tempo do MC**	**# Triângulos**
Caixa de Sapato	60x20x10	0.1350	0.6853ms	304
Caixa de Sapato	60x20x10	0.3090	0.6787ms	638
Pituba	83x45x23	0.7000	0.8967ms	7452
Pituba	83x45x23	0.4207	0.8381ms	6835
Proxy	173x176x49	0.68	4.2769ms	52517

Tabela 6.2: Informações referentes aos resultados obtidos com os modelos Caixa de Sapato, Pituba e Proxy. O tempo do *Marching Cubes* refere-se tanto à contagem de triângulos quanto o algoritmo em si.

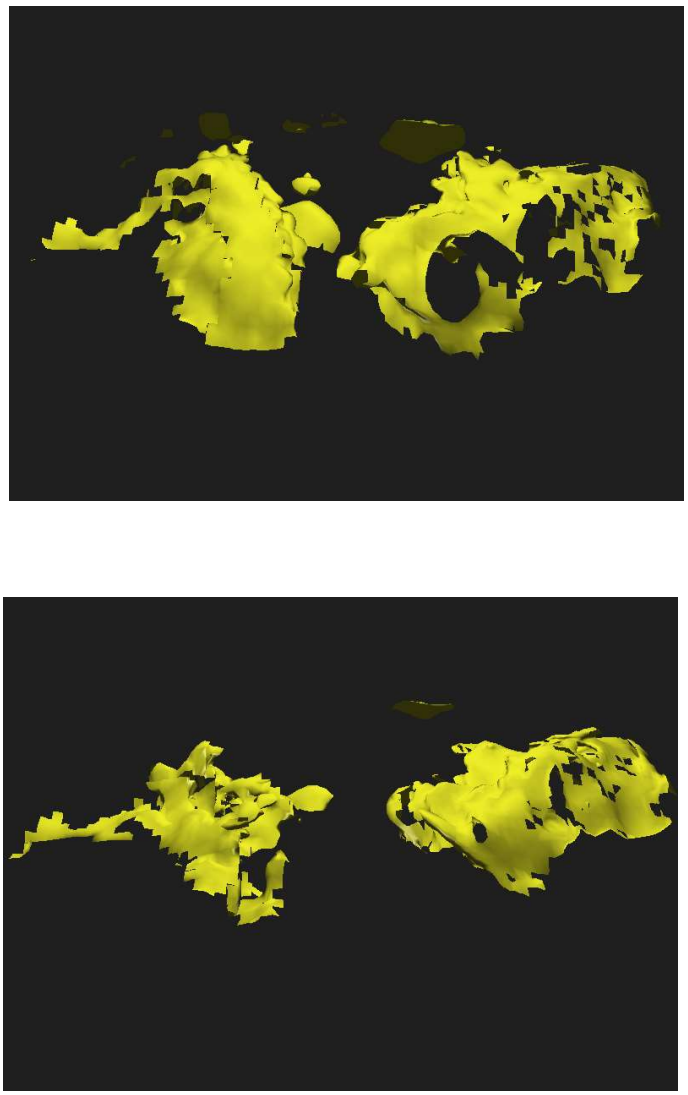


Figura 6.6: Visulização resultante do modelo Pituba. O campo escalar escolhido é referente à saturação de óleo. Os isovalores são 0.7 e 0.4207 à esquerda e à direita, respectivamente.

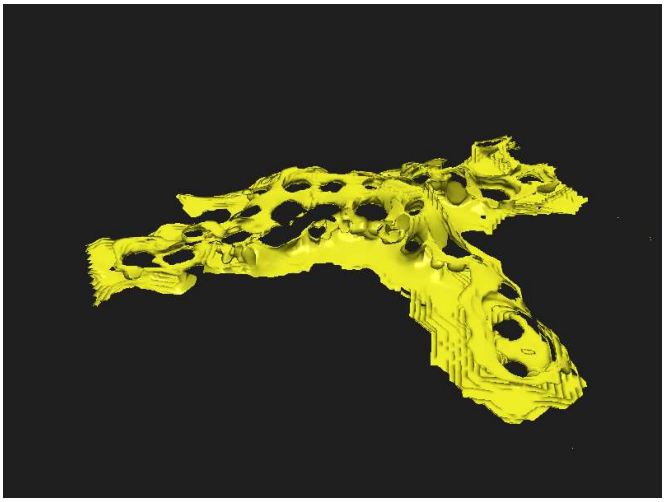


Figura 6.7: Visulização resultante do modelo Proxy. O campo escalar escolhido é referente à saturação de óleo. O isovalore de interesse é 0.68.

7
Conclusão e Trabalhos Futuros

Nesta dissertação, elaborou-se uma aplicação de execução em tempo real para a extração de isosuperfície de modelos de reservatórios naturais de petróleo. Discutiu-se os algoritmos mais conhecidos de extração e, por meio de comparações, definiu-se que o MC seria o mais apropriado para adaptar ao caso de interesse.

Foi elaborada uma modificação do *PN Triangles* (36) para o cálculo das normais das superfícies de Bézier. A nova proposta, a técnica do envelope, consiste em computar uma versão inflada da superfície a ser gerada e encontrar as normais por meio da normalização da diferença entre a superfície inflada e a original.

As características dos modelos de reservatório foram analisadas a fim que de uma adaptação do MC fosse feita para que funcionasse em malhas de células hexaédricas levemente côncavas. As modificações incluem o cálculo das amostras nos vértices por meio da média e o uso de um vetor de vértices e de uma matriz tridimensional que guarda oito índices por célula referentes ao vetor. Além disso, para definir as normais nos vértices, a inversa da matriz Jacobiana é utilizada para mapear a normal no espaço paramétrico, calculada pelo método de diferenças finitas centrais, para o espaço do mundo.

Conseguiu-se obter uma solução de rápida execução em tempo real por meio do uso intensivo de programação paralela por meio dos *Compute Shaders* e por meio de uma subdivisão adaptativa.

Alguns dos trabalhos futuros que podem ser realizados incluem:

- Existem múltiplas propostas que tentam consertar os problemas de ambiguidade e de topologia presentes no algoritmo original do MC. *Vegas et al.* (35) pretendem encerrar a questão com um novo teste do interior de uma célula. Um avanço relevante desta dissertação seria a adaptação do MC apresentado por esse artigo para malhas de modelos de reservatório.
- Comparar a curvatura das superfícies geradas pelo *PN Triangles* modificado pela técnica do envelope e outras propostas de superfícies curvas, como feito em (5).
- É possível que haja alguns defeitos visuais no *PN Triangles* por conta da inadequação da superfície proposta em certos casos e por falta de

coplanaridade, conforme discutido por *Lee e Jen* (21). Uma solução que combine a técnica do envelope com essa abordagem pode apresentar resultados ainda mais promissores.

– Pesquisar e analisar a possibilidade de adaptar ainda mais o MC para renderizar isosuperfícies com múltiplas camadas de forma translúcida e sem comprometer o desempenho da aplicação com algoritmos de ordenação lentos ou com estruturas de dados pesadas.

– Aplicar e analisar o emprego da técnica proposta em modelos massivos de reservatórios utilizados na indústria.

Referências bibliográficas

[1] ABRAHAM, F.. **Visualização de Modelos Massivos de Reservatórios Naturais de Petróleo**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2011.

[2] Autodesk. **Continuity 1: G0, G1, G2, G3**. Online; accessed 21-Sep-2020.

[3] AVILA, L.; KAUFMAN, A.. **Volumetric ray tracing.** p. 11–18, 01 1995.

[4] BARROSO, V.. **Simulação eficiente de fluidos no espaço paramétrico de malhas estruturadas tridimensionais**. Tese de Doutorado, February 2014.

[5] BOSCHIROLI, M.; FÜNFZIG, C.; ROMANI, L. ; ALBRECHT, G.. **A comparison of local parametric c0 bézier interpolants for triangular meshes**. Computers & Graphics, 35:20–34, 02 2011.

[6] BOUBEKEUR, T.; ALEXA, M.. **Phong tessellation**. ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2008), 27(5), 2008.

[7] BOURKE, P.. **Polygonising a scalar field**, 1994. Online; accessed 21-Sep-2020.

[8] CATMULL, E.; CLARK, J.. **Recursively generated b-spline surfaces on arbitrary topological meshes**. Computer-Aided Design, 10(6), 1978.

[9] CHERNYAEV, E. V.. **Marching cubes 33: Construction of topologically correct isosurfaces**. Technical report, 1995.

[10] CUSTÓDIO, L.. **Revisitando o Marching Cubes 33: Garantias Topológicas e Qualidade de Malha**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2014.

[11] DAKE, L. P.. **Fundamentals of Reservoir Engineering**. Elsevier Science, 1978.

[12] DEPARTMENT OF RADIOLOGY, U. O. I.. **Knee with anterior tibial osteotomy**. Online; accessed 22-Sep-2020.

[13] DJ, W.; DS, M.. **A triangular g1 patch from boundary curves**. Computer-Aided Design, 28:113–123, 1996.

[14] DÜRST, M. J.. **Additional reference to marching cubes**. Computer Graphics, 22(2):72–73, 1988.

[15] FÜNFZIG, C.; MÜLLER, K.; HANSFORD, D. ; FARIN, G.. **Png1 triangles for tangent plane continuous surfaces on the gpu**. In: PROCEEDINGS OF GRAPHICS INTERFACE 2008, GI '08, p. 219–226, CAN, 2008. Canadian Information Processing Society.

[16] GIBSON, S. F. F.. **Constrained elastic surface nets: Generating smooth surfaces from binary segmented data**. In: Wells, W. M.; Colchester, A. ; Delp, S., editors, MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION — MICCAI'98, p. 888–898, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[17] JU, T.; UDESHI, T.. **Intersection-free contouring on an octree grid**. 01 2006.

[18] Khronos. **Tessellation - OpenGL Wiki**. Online; accessed 21-Sep-2020.

[19] KOBBELT, L. P.; BOTSCH, M.; SCHWANECKE, U. ; SEIDEL, H.-P.. **Feature sensitive surface extraction from volume data**. In: PROCEEDINGS OF THE 28TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '01, p. 57–66, New York, NY, USA, 2001. Association for Computing Machinery.

[20] LEAL, E.. **Traçado de linhas de fluxo em modelos de reservatórios naturais de petróleo baseado em métodos numéricos adaptativos**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, 2015.

[21] LEE, Y.-C.; JEN, C.-W.. **Improved quadratic normal vector interpolation for realistic shading**. The Visual Computer, 17:337–352, 2001.

[22] LEWINER, T.; LOPES, H.; VIEIRA, A. ; TAVARES, G.. **Efficient implementation of marching cubes' cases with topological guarantees**. Journal of Graphics Tools, 8, 04 2012.

[23] LOOP, C.. **Smooth Subdivision Surfaces Based on Triangles**. Tese de Doutorado, January 1987.

[24] LOOP, C.; SCHAEFER, S.; NI, T. ; CASTAÑO, I.. **Approximating subdivision surfaces with gregory patches for hardware tessellation**. ACM Trans. Graph., 28, 12 2009.

[25] LORENSEN, W. E.; CLINE, H. E.. **Marching cubes: A high resolution 3d surface construction algorithm**. SIGGRAPH Comput. Graph., 21(4):163–169, Aug. 1987.

[26] MONTANI, C.; SCATENI, R. ; SCOPIGNO, R.. **A modified look-up table for implicit disambiguation of marching cubes**. The Visual Computer, 10(6):353–355, 1994.

[27] MORETON, H. P.; SÉQUIN, C. H.. **Functional optimization for fair surface design**. In: PROCEEDINGS OF THE 19TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH '92, p. 167–176, New York, NY, USA, 1992. Association for Computing Machinery.

[28] MYLES, A.; NI, T. ; PETERS, J.. **Fast parallel construction of smooth surfaces from meshes with tri/quad/pent facets**. Comput. Graph. Forum, 27:1365–1372, 07 2008.

[29] NATARAJAN, B.. **On generating topologically consistent isosurfaces from uniform samples**. The Visual Computer, 11:52–62, 2005.

[30] NIELSON, G.. **Dual marching cubes**. p. 489– 496, 11 2004.

[31] NIELSON, G.; HAMANN, B.. **The asymptotic decider: Resolving the ambiguity in marching cubes**. p. 83–91, 413, 11 1991.

[32] OF ENGINEERING & TECHNOLOGY, R. S.. **Mechanical engineering 308 computer aided design and analysis**. Online; accessed 21-Sep-2020.

[33] PAIVA, A. C. D.; SEIXAS, R. D. B. ; GATTASS, M.. **Introdução à visualização volumétrica**, Jan 1999.

[34] VAN GELDER, A.; WILHELMS, J.. **Topological considerations in isosurface generation**. ACM Trans. Graph., 13(4):337–375, Oct. 1994.

[35] VEGA, D.; ABACHE, J. ; COLL, D.. **A fast and memory saving Marching Cubes 33 implementation with the correct interior test**. Journal of Computer Graphics Techniques (JCGT), 8(3):1–18, August 2019.

[36] VLACHOS, A.; PETERS, J.; BOYD, C. ; MITCHELL, J. L.. **Curved pn triangles**. In: IN SYMPOSIUM ON INTERACTIVE 3D GRAPHICS, p. 159–166, 2001.

[37] VLACHOS, A.; PETERS, J.; BOYD, C. ; MITCHELL, J. L.. **Curved pn triangles**. In: PROCEEDINGS OF THE 2001 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS, I3D '01, p. 159–166, New York, NY, USA, 2001. Association for Computing Machinery.

[38] WENGER, R.. **Isosurfaces: Geometry, Topology, and Algorithms**. An A K Peters Book. CRC Press, 2013.

[39] ZHOU, C.; SHU, R. ; KANKANHALLI, M. S.. **Handling small features in isosurface generation using marching cubes**. Computers & Graphics, 18(6):845–848, 1994.