LEONARDO ALMEIDA DE ARAÚJO

GERAÇÃO DE ISOSSUPERFÍCIES PELOS ALGORITMOS *MARCHING CUBES* E VARIAÇÕES

LAVRAS – MG

2014

LEONARDO ALMEIDA DE ARAÚJO

GERAÇÃO DE ISOSSUPERFÍCIES PELOS ALGORITMOS *MARCHING CUBES* E VARIAÇÕES

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof. D.Sc. Sanderson L. Gonzaga de Oliveira

LAVRAS – MG

2014

LEONARDO ALMEIDA DE ARAÚJO

GERAÇÃO DE ISOSSUPERFÍCIES PELOS ALGORITMOS MARCHING CUBES E VARIAÇÕES

Monografia de graduação apresentada ao Colegiado do Curso de Bacharelado em Ciência da Computação, para obtenção do título de Bacharel.

APROVADA em 11 de julho de 2014.

Bel. Jéssica Renata Nogueira

Me. Frederico Santos de Oliveira

Prof. Dr. Sanderson L. Gonzaga de Oliveira (Orientador)

LAVRAS-MG
2014

Dedico esta monografia aos meus pais Vicente e Lucilene, que sempre fizeram de tudo por mim e me educaram com os ensinamentos mais importantes da minha vida.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por minha vida e por tudo que eu tenho.

Agradeço aos meus pais pelo amor que sempre me deram, por serem as pessoas mais importantes para mim e por sempre fazerem tudo o que podiam para me ajudar. Agradeço ao meu irmão Luciano por ser um grande amigo. Agradeço aos meus avós maternos José e Ercy, por sempre me darem força e me ajudar muito em tudo que precisei. Agradeço aos meus avós paternos Benedito *(in memorian)* e Sebastiana *(in memorian)*, por serem um grande exemplo, pelos conselhos e lembranças boas que deixaram.

Agradeço ao meu professor e orientador Sanderson, por me passar muito conhecimento, por sempre me ajudar em meus assuntos acadêmicos, pelas conversas e pela paciência que teve comigo. Agradeço a todos os professores e funcionários do departamento de Ciência da Computação da UFLA. Agradeço também ao professor Carlos Dietrich, por me ajudar muito neste trabalho.

Agradeço aos meus amigos Bruno e Mateus pela grande amizade, pela confiança e por sempre estarem juntos comigo quando preciso. Agradeço a todos meus amigos da República Tatu Rodando por serem uma família pra mim em todo tempo da faculdade. Agradeço a todos os demais amigos pelo companheirismo e por serem pessoas tão boas para mim.

Agradeço também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

RESUMO

Neste trabalho, descrevem-se o algoritmo *Marching Cubes* e as suas principais variações, para a geração de isossuperfícies. O algoritmo *Marching Cubes* extrai malhas poligonais de imagens tridimensionais. Mostram-se 14 variações do *Marching Cubes* que apresentam malhas com melhor qualidade, custo computacional menor ou melhor suavização que o *Marching Cubes*. Realizam-se experimentos com o *Marching Cubes* e 6 de suas variações. Nos experimentos realizados o algoritmo Macet apresentou a melhor qualidade de malha, seguido do algoritmo *Dual Contouring*. O algoritmo Issue apresentou o melhor custo computacional e o algoritmo *SurfaceNets* gerou malhas com melhor suavização.

Palavras-Chave: isossuperfícies; poligonizadores; *Marching Cubes*; computação gráfica; visualização científica.

ABSTRACT

In this paper we describe the algorithm Marching Cubes and its main variations, for generation of isosurfaces. The Marching Cubes algorithm extracts polygonal mesh of three-dimensional images. Presents 14 variations Marching Cubes presenting meshes with better quality, lower computational cost and better smoothing the Marching Cubes. Experiments are performed with the Marching Cubes and 6 of its variations. In the experiments the Macet algorithm presented the best quality mesh, then comes the algorithm Dual Contouring. The Issue algorithm presented the best computational cost and SurfaceNets algorithm generated meshes with better smoothing.

Keywords: isosurfaces; algorithms; Marching Cubes; computer graphics ; scientific visualization.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A criação e a manipulação de imagens auxiliam no entendimento de estruturas complexas, como, por exemplo, a visualização de órgãos do corpo humano. Imagens são atraentes, comparadas a outras formas de informação. A possibilidade de visualizar e manipular imagens e estruturas tridimensionais permite a compreensão e a análise de enormes quantidades de informações de natureza espacial e temporal, explorando-se a grande capacidade humana de raciocinar e de se comunicar visualmente. O processamento de imagens se mostra de grande importância devido aos grandes acervos de imagens nas mais diferentes áreas e aplicações, tornando-se uma ferramenta útil para o avanço de vários ramos científicos (SCHMITZ, 2009).

Neste trabalho, será abordado o algortimo *Marching Cubes* (LORENSEN; CLINE, 1987) e algumas de suas variações. O *Marching Cubes* é um poligonizador que extrai uma isossuperfície de uma textura 3D. Tal visualização é interessante, por exemplo, em imagens médicas, para avaliar o estado de algumas estruturas internas do corpo. O algoritmo *Marching Cubes* é aplicado em imagens de tomografia computadorizada, ressonância magnética, modelagem e efeitos especiais 3D. Vários estudos foram realizados com base no algortimo *Marching Cubes* a fim de serem desenvolvidas modificações que gerassem malhas de melhor qualidade que as malhas geradas pelo *Marching Cubes* ou com custo computacional menor que o *Marching Cubes*.

1.1 Objetivos

O objetivo deste trabalho é apresentar o poligonizador de isossuperfícies *Marching Cubes* e suas principais variações. Também é objetivo deste trabalho explicar 14

variações do algoritmo *Marching Cubes*, verificar e comparar as qualidades das isossuperfícies geradas por 6 variações do *Marching Cubes*: *Marching Tetrahedra* (SHIRLEY; TUCHMAN, 1990; TREECE; PRAGER; GEE, 1999), Issue (SHEN *et al.*, 1996), *SurfaceNets* (GIBSON, 1998), *DualSurfaceNets* (LEVENTON; GIBSON, 1999), *Dual Coutouring* (JU *et al.*, 2002) e Macet (DIETRICH, 2008). Para verificar a qualidade da malha, utilizam-se as métricas de ângulos internos mínimo e máximo (BABUSKA; AZIZ, 1976) e a métrica de razão entre os raios do triângulo (SCHREINER *et al.*, 2006).

Também é objetivo deste trabalho verificar e comparar os custos computacionais das variações do *Marching Cubes* citadas acima. Serão abordados aspectos que diferem o algoritmo *Marching Cubes* de suas variações, a fim de ressaltar quais melhorias uma variação traz em relação ao *Marching Cubes* e também a outras variações.

1.2 Motivações

O uso de imagens tridimensionais é de grande ajuda para a resolução de problemas reais em diversas áreas. A utilização de isossuperfícies auxilia na compreensão da topologia de imagens tridimensionais. Logo, a geração de isossuperfície e a proposta de novos poligonizadores para realizar essa geração é um assunto de grande importância em computação gráfica e visualização científica.

O algoritmo *Marching Cubes* é uma das publicações mais citadas nas pesquisas em computação gráfica (DIETRICH, 2008). Ao longo do tempo, algumas variações do *Marching Cubes* foram criadas para gerar resultados melhores que o *Marching Cubes*, quer seja em menor custo computacional, quer seja em geração de isossuperfíceis de melhor qualidade. Compreender as variações de um algo-

ritmo é importante para se saber em que situações determinada variação terá um melhor resultado.

1.3 Organização do Trabalho

Este trabalho é dividido em cinco partes, que são: introdução, referencial teórico, metodologia, resultados e conclusão. A introdução é apresentada no capítulo 1. O referencial teórico é apresentado no capítulo 2. Já a metodologia adotada para a elaboração do trabalho é apresentada no capítulo 3. Os resultados obtidos são encontrados no capítulo 4. As conclusões deste trabalho são encontradas no capítulo 5.

2 REFERENCIAL TEÓRICO

Neste capítulo, descreve-se na seção 2.1, o conceito de isossuperfície e de poligonizadores de isossuperfícies. Na seção 2.2, na página 20, descreve-se o algoritmo *Marching Cubes*. Na seção 2.3, na página 23, são apresentadas as principais variações do *Marching Cubes*. Na seção 2.3.15, na página 46, é apresentado um resumo do capítulo.

2.1 Isossuperfícies

Uma isossuperfície é um conjunto de nível de uma função contínua cujo domínio é um espaço tridimensional. A utilização de isossuperfícies auxilia na compreensão da topologia de imagens tridimensionais, da mesma forma que isolinhas auxiliam na compreensão da topologia de imagens bidimensionais. A figura 2.1 é um exemplo de visualização de uma isossuperfície de uma imagem tridimensional.

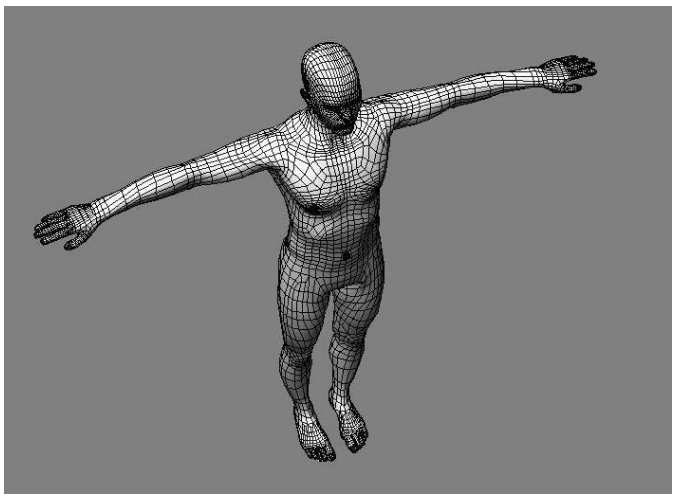


Figura 2.1: Exemplo de uma isossuperfície. A imagem foi gerada no *software* livre MakeHuman 1.0 (MAKEHUMAN, 2001).

A visualização de isossuperfícies é, normalmente, realizada por meio de renderização indireta, que é um processo de geração de uma imagem a partir de um modelo, em que uma malha é utilizada para representar uma superfície. Grosso modo, uma malha é um grafo que forma um conjunto de polígonos. Os métodos que geram malhas a partir de isossuperfícies são conhecidos como poligonizadores (NING; BLOOMENTHAL, 1993). Pode-se considerar que o algoritmo *Marching Cubes* e suas variações são os poligonizadores mais utilizados (NEWMAN; YI, 2006).

2.1.1 A qualidade da isossuperfície

A análise da qualidade da isossuperfície gerada pelo algoritmo *Marching Cubes* é realizada por duas abordagens. A primeira abordagem está relacionada ao modo de como a isossuperfície intersecciona uma célula da imagem. A segunda abordagem está relacionada ao modo de como cada triângulo é construído pelo *Marching Cubes*. O resultado dessas duas abordagens é um conjunto de métricas que determinam a qualidade final da isossuperfície gerada pelo algoritmo *Marching Cubes*.

As métricas de qualidade escolhidas têm duas características principais, sendo elas: (1) nenhuma métrica leva em consideração a área de um triângulo da isossuperfície e (2) todas as métricas consideram o triângulo equilátero como o triângulo ótimo (DIETRICH, 2008). A escolha do triângulo equilátero como triângulo de referência segue a compreensão geral sobre triângulos de boa qualidade, ou seja, triângulos com ângulos internos que não se aproximam de 0° ou 180° (SANTOS; SCHEER, 2006).

A qualidade da isossuperfície é determinada pela qualidade do pior triângulo da isossuperfície. As métricas de qualidade utilizadas neste trabalho são as seguintes:

- *Ângulos internos mínimo e máximo* (BABUSKA; AZIZ, 1976), métrica baseada no cálculo do maior e do menor ângulo interno do triângulo. Quanto menor for o tamanho do menor ângulo interno do triângulo, pior será sua qualidade. Quanto maior for o tamanho do maior ângulo interno do triângulo, pior será sua qualidade.

- *Relação de raios do triângulo* (SCHREINER *et al.*, 2006), métrica baseada na razão entre os raios do círculo circunscrito (R) e o raio do círculo inscrito (r) no triângulo. Neste trabalho, o valor da razão dos raios do triângulo (P) é normalizada no intervalo fechado [0,1], onde $P = 2r/R$ e o triângulo equilátero tem qualidade 1.

2.2 Algoritmo *Marching Cubes*

O algoritmo *Marching Cubes* foi projetado para gerar isossuperfícies em um campo escalar tridimensional. Ele é conhecido pelo seu baixo custo computacional e robustez.

O algoritmo utiliza a técnica de divisão e conquista para localizar a superfície de determinado objeto. Um isovalor é passado para o algoritmo. O isovalor é a referência que o algoritmo utiliza para formar a isossuperfície de um objeto. Quanto maior o isovalor, maior será a quantidade de detalhes da isossuperfície gerada. O algoritmo particiona o objeto em diversas células interligadas, todas com um tamanho pré-estabelecido. Quanto menor a célula, maior o tempo de processamento e a precisão. Cada célula é formada por oito pixels, como mostrado na figura 2.2. Cada pixel é representado por um vértice da célula (LORENSEN; CLINE, 1987).

Essa célula é uma subdivisão espacial de um determinado objeto em partes menores. Uma célular limita o espaço de busca do algoritmo *Marching Cubes*,

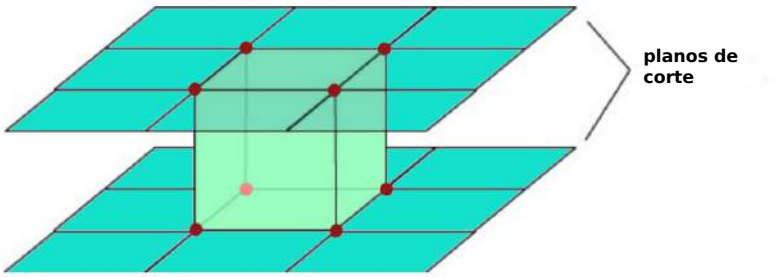


Figura 2.2: Planos de corte de uma célula utilizado no algoritmo *Marching Cubes*.

onde a isossuperfície gerada tem um comportamento mais simples. Uma célula também pode ser chamada de cubo.

Godoi (2012, p. 33) explica que o algoritmo *Marching Cubes* percorre todas as células que a imagem foi dividida. Para verificar se uma célula faz intersecção com a isosssuperfície, o algoritmo verifica se algum vértice dessa célula excede ou equivale ao isovalor informado. Os vértices das células, que possuem valor maior ou igual ao isovalor, são ditos ativados e considera-se que estejam dentro da isossuperfície. Os vértices das células com valor menor ao isovalor são ditos desativados e considera-se que estejam fora da isossuperfície. Na figura 2.3, ilustra-se um exemplo de uma célula com valores atribuídos a seus vértices.

Dessa forma, pela localização dessas interseções, é possível determinar a topologia de uma superfície por meio de triangulações. Pelo *Marching Cubes*, pode-se enumerar 256 situações diferentes para a representação de superfícies para cada célula. Por rotações e simetrias, todos os casos podem ser agrupados em 15 famílias, mostradas na figura 2.4. Essas 15 famílias servem como índice, que indica a posição e a quantidade de triângulos que formarão trechos da isossuperfície dentro de cada célula.

A isossuperfície será gerada depois que o algoritmo *Marching Cubes* analisar e estabelecer trechos da isossuperfície dentro de todas as células da imagem. O processamento de cada célula é independente do processamento das demais célu-

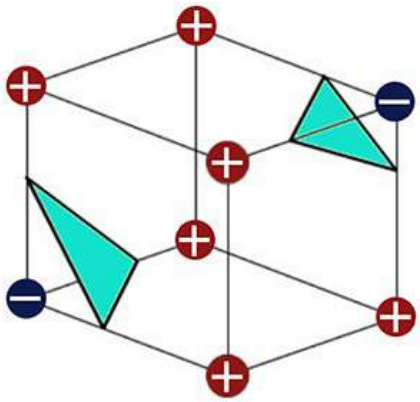


Figura 2.3: Um exemplo de geração de isossuperfície por uma célula do *Marching Cubes*. Os vértices com sinais positivos representam vértices ativados e os vértices com sinais negativos representam vértices desativados.

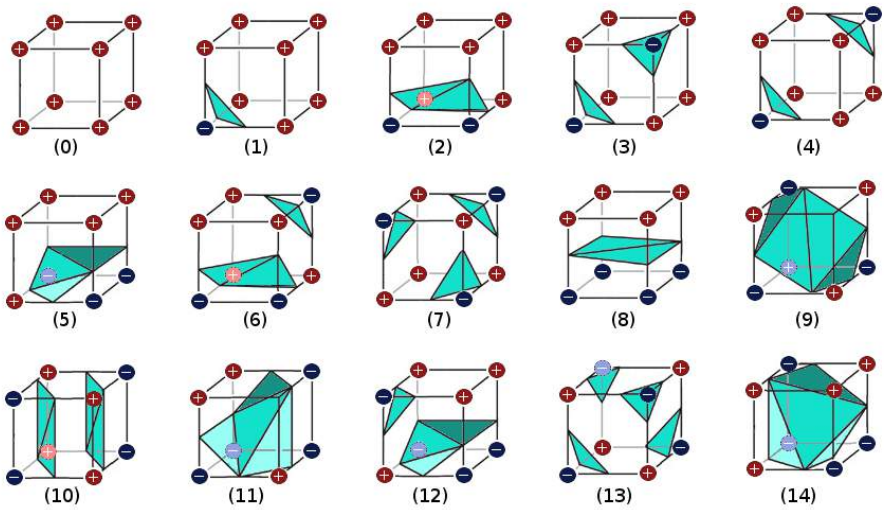


Figura 2.4: Os 15 casos representativos do *Marching Cubes*.

las e inclui passos computacionalmente simples. Essas características motivaram a melhoria do *Marching Cubes* em outros algoritmos de forma que fosse alcançada redução do custo computacional ou melhor qualidade na geração da isosssuperfície.

2.3 Variações do algoritmo *Marching Cubes*

O algoritmo *Marching Cubes* é referência na poligonização de isossuperfícies em imagens 3D (DIETRICH, 2008). Diversos estudos sobre esse assunto fizeram com que fossem criadas variações do *Marching Cubes*. Nas subseções a seguir, descrevem-se as principais variações do *Marching Cubes*:

- *Marching Tetrahedra* (SHIRLEY; TUCHMAN, 1990; TREECE; PRAGER; GEE, 1999), descrito na subseção 2.3.1;
- *The Asymptotic Decider* (NIELSON; HAMANN, 1991), apresentado na subseção 2.3.2;
- *Algoritmo de Itoh-Koyamada* (ITOH; KOYAMADA, 1995), abordado na subseção 2.3.3;
- *Issue* (SHEN *et al.*, 1996), descrito na subseção 2.3.4;
- *Algoritmo de Fujishiro-Maeda-Sato-Takeshima* (FUJISHIRO *et al.*, 1996), descrito na subseção 2.3.5;
- *SurfaceNets* (GIBSON, 1998), apresentado na subseção 2.3.6;
- *DualSurfaceNets* (LEVENTON; GIBSON, 1999), abordado na subseção 2.3.7;
- *Dual Coutouring* (JU *et al.*, 2002), descrito na subseção 2.3.8;
- *Extended Marching Cubes* (KOBBELT *et al.*, 2001), descrito na subseção 2.3.9;
- *Algoritmo de Lewiner-Lopes-Vieira-Tavares* (LEWINER *et al.*, 2003), apresentado na subseção 2.3.10;

- *Dual Marching Cubes* (NIELSON, 2004), abordado na subseção 2.3.11;
- *Algoritmo de Zhang-Bajaj-Sohn* (ZHANG; BAJAJ; SOHN, 2005), abordado na subseção 2.3.12;
- *HistoPyramids* (DYKEN *et al.*, 2008), descrito na subseção 2.3.13;
- Macet (DIETRICH, 2008; DIETRICH *et al.*, 2008; DIETRICH *et al.*, 2009b; DIETRICH *et al.*, 2009a), apresentado na subseção 2.3.14.

Em algumas variações, foram utilizadas figuras com objetos em duas dimensões apenas para exemplificar, de uma forma mais simples, o funcionamento de determinado algoritmo.

2.3.1 Algoritmo *Marching Tetrahedra*

O algoritmo *Marching Tetrahedra* (SHIRLEY; TUCHMAN, 1990) foi projetado para melhorar a qualidade da isossuperfície em relação ao *Marching Cubes* e de reduzir situações imprecisas geradas pelo *Marching Cubes*. O que muda entre o *Marching Tetrahedra* em relação ao *Marching Cubes* é que cada célula da grade do *Marching Tetrahedra* é subdividido em tetraedros, como pode ser observado na figura 2.5. Essa subdivisão da célula pode ser em cinco, em seis ou em vinte e quatro tetraedros, com suas respectivas tabelas de intersecção. O *Marching Tetrahedra* gera isossuperfícies mais fiéis aos dados originais que o *Marching Cubes*, porém precisa de mais memória (SHIRLEY; TUCHMAN, 1990).

No *Marching Cubes*, existem 256 diferentes casos possíveis para a triangulação de uma célula, que podem ser reduzidos para 15 casos por simetria. Com o *Marching Tetrahedra*, são analisados tetraedros em vez de células. Tetraedros só podem ter 16 diferentes casos possíveis de triangulações, que podem ser reduzidos para três casos, por simetria (TREECE; PRAGER; GEE, 1999). Essa redução

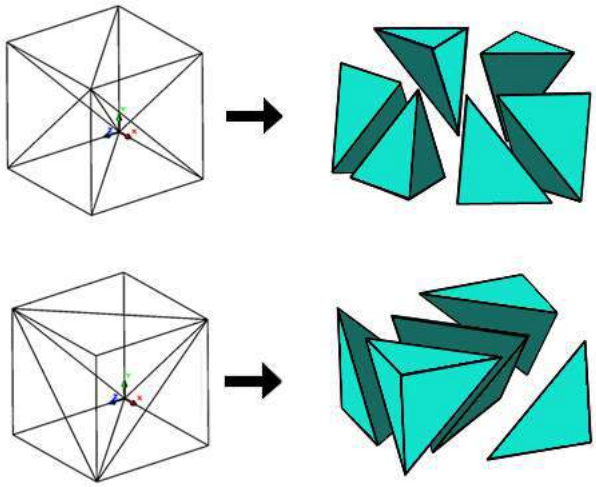


Figura 2.5: Representação de duas células divididas em seis e em cinco tetraedros, respectivamente.

de casos diminui as ocorrências de triangulos de má qualidade, e faz com que o algoritmo *Marching Tetrahedra* gere isossuperfícies com maior qualidade em relação ao algoritmo *Marching Cubes*. Porém, como cada célula é dividida em vários tetraedros, o custo de memória e de processamento é elevado (SHIRLEY; TUCHMAN, 1990).

Foram detectados alguns problemas no *Marching Tetrahedra*. O primeiro problema é que a divisão de uma célula em vários tetraedros apresenta uma ambiguidade. Essa ambiguidade faz com que a simetria da subdivisão da célula altere o formato das células vizinhas, a fim de alinhar as faces dos tetraedros para formar a isossuperfície. Isso ocorre porque, em alguns casos, a divisão de uma célula não se encaixa com a divisão de outra célula vizinha. Outro problema é que o *Marching Tetrahedra* cria um número ainda maior de triângulos que o *Marching Cubes* para um determinado conjunto de dados. Esses problemas foram corrigidos por Treece, Prager e Gee (1999). Eles utilizaram uma técnica de agrupamento de vértices que resolveu o problema de ambiguidade entre células vizinhas e reduziu o número de triângulos em relação à primeira versão do *Marching Tetrahedra*. Com isso, o

algoritmo *Marching Tetrahedra* ainda gera mais polígonos e utiliza mais memória que o algoritmo *Marching Cubes*; porém, gera uma isossuperfície com melhor qualidade que o *Marching Cubes*.

2.3.2 Algoritmo *The Asymptotic Decider*

O algoritmo *The Asymptotic Decider* (NIELSON; HAMANN, 1991) foi projetado para corrigir um problema de ambiguidade na geração de isossuperfícies pelo algoritmo *Marching Cubes*. Esse problema de ambiguidade pode acontecer em algumas células e faz com que a isossuperfície gerada fique inconsistente em relação ao objeto sendo reconstruído. Essa ambiguidade, em certos casos, não faz tanta diferença no resultado final da isossuperfície. Com isso, o *Marching Cubes*, por não tratar essa ambiguidade, é mais rápido que o algoritmo *The Asymptotic Decider* (NIELSON; HAMANN, 1991).

Segundo Junior (2008), o problema de ambiguidade na geração da isossuperfície pelo *Marching Cubes* pode ocorrer de uma forma similar ao problema de ambiguidade que também pode ocorrer na geração de isolinhas em imagens bidimensionais. Em imagens tridimensionais, o problema ocorre nas situações em que dois vértices opostos em relação a uma diagonal de uma célula estão ativados. Isso é chamado de ambiguidade interna. Em imagens bidimensionais, o problema de ambiguidade ocorre nas situações em que apenas dois vértices da diagonal de uma face estão ativados. Isso é chamado de ambiguidade de face e está relacionado com o algoritmo *Marching Squares* (ROSENSTROM, 2009), que é uma versão do *Marching Cubes* para a geração de isolinhas em imagens bidimensionais. O problema de ambiguidade faz com que uma célula tenha duas formas "distintas" de geração de isossuperfície e geram um problema no algoritmo *Marching Cubes*. Esse problema faz com que o algoritmo não diferencie, em determinadas células, o lado de fora do lado de dentro da isossuperfície, e causa problemas na quali-

dade da malha gerada. Na figura 2.6, ilustra-se um exemplo desse problema de ambiguidade em uma célula tridimensional e em uma célula bidimensional, com os mesmos vértices ativados.

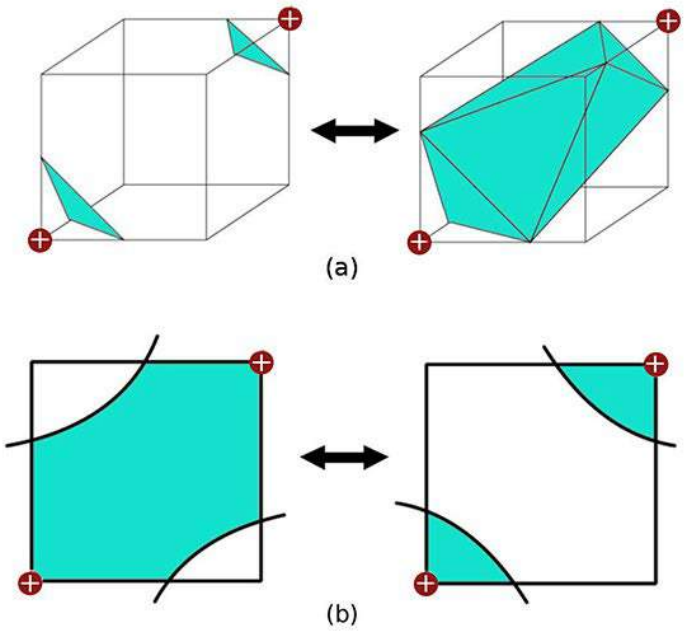


Figura 2.6: Nos dois casos, (a) e (b), os mesmos vértices das células estão ativados. Um vértice ativado é um vértice que está dentro da isossuperfície. O problema de ambiguidade é que trechos da isossuperfície em uma célula podem ser gerados com formatos distintos, mesmo que os vértices ativados sejam os mesmos.

Segundo Nielson e Hamann (1991), o algoritmo *The Asymptotic Decider* resolve as configurações ambíguas que ocorrem no *Marching Cubes*. O algoritmo *The Asymptotic Decider* analisa a posição em que a isossuperfície intersecciona as arestas de determinada célula. Por essa posição, é obtido um ponto dentro da célula. Por esse ponto, é obtida uma distância desse ponto até os vértices ativados. Por essa distância, o algoritmo decide se o conjunto de vértices ativados na célula está separado pela isossuperfície. Com isso, aquele problema de ambiguidade do *Marching Cubes* não ocorre no algoritmo *The Asymptotic Decider*. O algoritmo

The Asymptotic Decider gera malhas de melhor qualidade e perde em desempenho computacional em relação ao algoritmo *Marching Cubes*.

2.3.3 Algoritmo de Itoh-Koyamada

O algoritmo de Itoh e Koyamada (1995) foi projetado para ter melhor desempenho computacional na geração de isossuperfícies que o algoritmo *Marching Cubes*. A ideia principal do algoritmo de Itoh-Koyamada é analisar o mínimo possível de células que não serão interseccionadas pela isossuperfície a ser gerada. Com menos células a serem analisadas, mais rápido o algoritmo gera a isossuperfície. As células no contexto do algoritmo de Itoh-Koyamada são as mesmas células do contexto do algoritmo *Marching Cubes*.

O algoritmo de Itoh-Koyamada marca aleatoriamente uma célula na imagem e define essa célula como ponto inicial. É realizada uma verificação nos valores dos vértices dessa célula para saber se ficará dentro ou fora da isossuperfície a ser gerada. Essa célula é marcada com um valor máximo se estiver do lado de dentro da isossuperfície ou é marcada com um valor mínimo se estiver do lado de fora da isossuperfície. Depois que a célula é marcada, o algoritmo marca aleatoriamente outra célula a ser analisada. Quando um ponto de máximo e um ponto de mínimo são marcados, o algoritmo percorre em linha reta as células vizinhas desses dois pontos até encontrar uma célula de intersecção entre os valores de máximo e mínimo. Essa célula corresponde à célula que intersecta a isossuperfície.

Na figura 2.7, ilustra-se em uma imagem bidimensional, um exemplo do procedimento do algoritmo de Itoh-Koyamada na análise de células dessa imagem. As células marcadas com 'x' representam as células que passaram pelo processo de análise do algoritmo de Itoh-Koyamada. Na imagem, nota-se que muitas células que não pertencem ao objeto que se quer poligonizar não são analisadas pelo algoritmo.

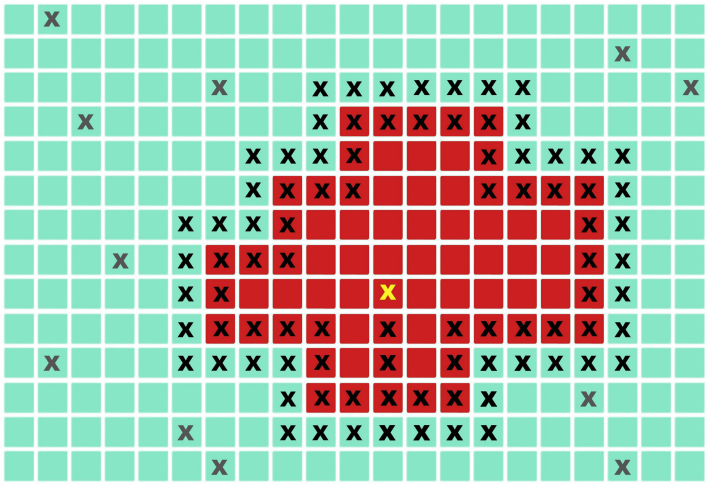


Figura 2.7: Cada quadrado verde representa uma célula da imagem que não faz parte da região a ser detectada. Cada quadrado vermelho representa uma célula da região a ser detectada. As células marcadas com 'x' de cor cinza representam as células que foram analisadas pelo algoritmo de Itoh-Koyamada que, diferentemente do algoritmo *Marching Cubes*, não analisa todas as células da imagem em sequência. A célula marcada com com 'x' de cor amarela representa a primeira célula da região a ser detectada que o algoritmo de Itoh-Koyamada encontrou. As células marcadas com 'x' de cor preta representam aquelas que foram percorridas pelo algoritmo de Itoh-Koyamada e são as células em que a isolinha (ou isossuperfície em um modelo 3D correspondente) será gerada. As células a serem analisadas são definidas aleatoriamente até que uma célula da região a ser detectada seja encontrada. Então, as células vizinhas dessas células são analisadas em alguma direção (no exemplo, na direção para baixo), até que se encontre uma célula que limita as duas regiões. Essa célula que limita as duas regiões é uma célula que intersecta a isossuperfície. Em seguida, as células vizinhas à célula de intersecção são analisadas e aquelas que também limitam as duas regiões são marcadas. Esse processo se repete até que todas as células que interseccionam a região sejam detectadas e marcadas.

O algoritmo de Itoh-Koyamada, ao encontrar uma célula que intersecta a isossuperfície, gera triângulos dentro dessa célula. Esse processo também é realizado no algoritmo *Marching Cubes* para dividir uma célula e gerar trechos da isossuperfície dentro dela. Após a geração de um trecho da isossuperfície em uma determinada célula que intersecciona a isossuperfície, todas as células vizinhas dessa

célula também são analisadas. Se uma célula que ainda não foi analisada também for uma célula de intersecção com a isossuperfície, o processo de geração de triângulos dentro dessa célula é realizado novamente. Esse processo é repetido até que todas as células de intersecção sejam encontradas. Quando todas as células de intersecção forem encontradas, a isossuperfície é gerada pela ligação de todas as triangulações que foram realizadas.

Testes realizados por Itoh e Koyamada (1995) indicam que quanto maior o número de células de uma imagem, maior é o número de células que não pertencem à região de intersecção da isossuperfície. O algoritmo de Itoh-Koymada analisa um número pequeno de células que não pertence à região de intersecção da isossuperfície.

Dessa forma, quanto maior o número de células que uma imagem for dividida, maior será o desempenho computacional que o algoritmo de Itoh-Koymada terá em relação ao algoritmo *Marching Cubes*. O algoritmo de Itoh-Koymada não é tão preciso quanto o algoritmo *Marching Cubes*. Consequentemente, como descrito, gera isossuperfícies com menos qualidade que o *Marching Cubes*.

2.3.4 Algoritmo *Issue*

O algoritmo *Isosurfacing in Span Space with Utmost Efficiency* (Issue) de Shen *et al.* (1996) é um aperfeiçoamento do algoritmo *Near Optimal Isosurface Extraction* (Noise) de Livnat, Shen e Johnson (1996). O *Issue* foi projetado para gerar isossuperfícies mais rápido que o algoritmo *Marching Cubes*.

O algoritmo *Issue* é baseado em uma representação denominada *Span Space*, em que cada célula da imagem é mapeada por dois pontos. Esses dois pontos representam as coordenadas x e y da célula, respectivamente, a valores mínimo e máximo da célula. Dessa forma, se um dado isovalor, estiver entre os pontos mínimo e máximo de uma célula, essa célula é considerada ativa. Quando todas

as células ativas são determinadas, a isossuperfície é gerada dentro de cada uma dessas células, de acordo com o valor dos pontos *x* e *y* de cada célula.

O algoritmo *Issue* mapeia todas as células em um espaço bidimensional para conseguir localizar rapidamente as células ativas, sem a necessidade de analisar todo o volume. Com isso, o algoritmo *Issue* tem melhor desempenho computacional em relação ao algoritmo *Marching Cubes*. Porém, no algoritmo *Issue*, não se realiza uma poligonização de cada célula ativa seguindo uma tabela de casos, como no algoritmo *Marching Cubes*. Isso acarreta uma geração de malha de qualidade pior que o algoritmo *Marching Cubes*. O *Issue* gera uma isossuperfície de má qualidade devido ao fato do algoritmo não poligonizar cada célula ativa.

2.3.5 Algoritmo de Fujishiro-Maeda-Sato-Takeshima

O algoritmo de Fujishiro-Maeda-Sato-Takeshima (FUJISHIRO *et al.*, 1996) foi projetado para gerar malhas com melhor qualidade que o algoritmo *Marching Cubes*. O algoritmo de Fujishiro-Maeda-Sato-Takeshima utiliza as técnicas do algoritmo *The Asymptotic Decider* (NIELSON; HAMANN, 1991), descrito na seção 2.3.2, na página 26, a tabela de casos do algoritmo *Marching Cubes* e a técnica *octree* de subdivisão de uma célula em oito células iguais. O algoritmo de Fujishiro-Maeda-Sato-Takeshima perde em desempenho computacional em relação ao algoritmo *Marching Cubes* e gera isossuperfícies com mais qualidade em relação ao algoritmo *Marching Cubes* (FUJISHIRO *et al.*, 1996).

O algoritmo de Fujishiro-Maeda-Sato-Takeshima verifica todas as células que a imagem foi dividida da mesma forma que o algoritmo *Marching Cubes*. O algoritmo de Fujishiro-Maeda-Sato-Takeshima, ao detectar uma célula de intersecção com a isossuperfície, utiliza o mesmo procedimento do algoritmo *The Asymptotic Decider* para verificar se uma célula possui algum problema de ambiguidade do *Marching Cubes*. Caso alguma célula possua o problema de ambiguidade, o algo-

ritmo de Fujishiro-Maeda-Sato-Takeshima utiliza a *octree* para que o problema de ambiguidade não ocorra. Em uma *octree*, divide-se a célula em oito células iguais. Essas oito células novas são analisadas novamente, repetindo-se o processo até que nenhuma nova célula apresente o problema de ambiguidade. A divisão pela *octree* resolve os problemas de ambiguidades e faz com que a isossuperfície gerada seja mais fiel a pequenos detalhes da imagem em relação ao algoritmo *Marching Cubes*.

2.3.6 Algoritmo *SurfaceNets*

O algoritmo *SurfaceNets* (GIBSON, 1998) gera uma representação de isossuperfície com menos contraste que a isossuperfície gerada pelo *Marching Cubes* e que permaneça fiel aos dados originais. O primeiro passo do algoritmo *SurfaceNets* é localizar quais células são interceptadas pela isossuperfície. Essas células são chamadas de ativas. É colocado um ponto no centro de cada umas dessas células ativas. Após todas as células serem inspecionadas, é utilizada uma *função de minimização* para ligar esses pontos entre as células e formar a isossuperfície. Uma função de minimização possui o limite do seu domínio em formato parabólico. A ligação entre os pontos é entre células com face em comum. Essa ligação de pontos realizada por meio da função de minimização faz com que a isossuperfície tenha pouco contraste.

O algoritmo *SurfaceNets* gera isossuperfícies com poucos contrastes e com poucas irregularidades. Na figura 2.8, exemplifica-se o algoritmo *SurfaceNets* em uma imagem bidimensional. Uma desvantagem do algoritmo *SurfaceNets* é que pode eliminar pequenas estruturas que podem ser importantes na aplicação. Por exemplo, alguma anatomia de uma superfície de um órgão humano pode ser desconsiderada.

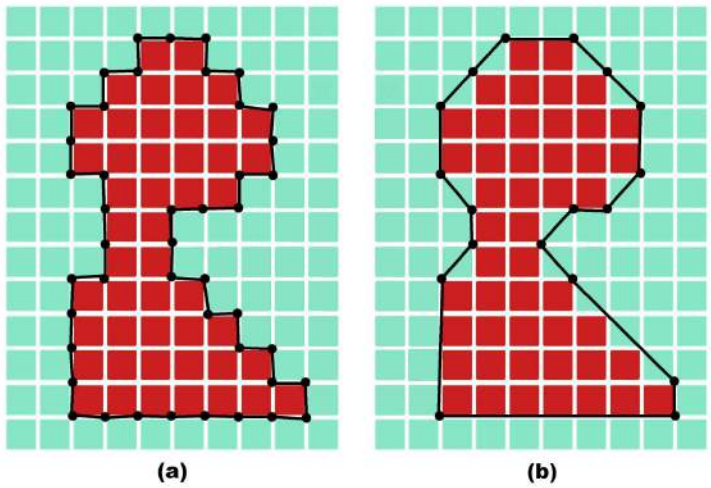


Figura 2.8: Exemplo do algoritmo *SurfaceNets* em imagem bidimensional. Em (a), ilustram-se duas regiões contornadas pelo algoritmo *Marching Cubes*. Em (b), ilustram-se duas regiões contornadas pelo algoritmo *SurfaceNets*.

O algoritmo *SurfaceNets* realiza os procedimentos do *Marching Cubes* sem utilizar a tabela de 256 posições do *Marching Cubes*. Uma isossuperfície gerada pelo algoritmo *SurfaceNets* possui triângulos com qualidade superior a uma isossuperfície gerada pelo algoritmo *Marching Cubes*, porém não se pode afirmar que o *SurfaceNets* gera uma isossuperfície final com qualidade superior ao *Marching Cubes* porque o *SurfaceNets* elimina detalhes do objeto sendo reconstruído. O custo computacional do algoritmo *SurfaceNets* é similiar ao custo computacional do algoritmo *Marching Cubes* (GIBSON, 1998).

2.3.7 Algoritmo *Dual SurfaceNets*

O algoritmo *Dual SurfaceNets* (LEVENTON; GIBSON, 1999) é uma extensão do algoritmo *SurfaceNets*. O *SurfaceNets* gera modelos a partir de uma única varredura na imagem e o *Dual SurfaceNets* gera modelos a partir da combinação de duas ou mais varreduras na imagem.

O algoritmo *Dual SurfaceNets* foi proposto para gerar isossuperfícies de resolução alta da imagem em análise em um curto espaço de tempo. Em alguns casos, por exemplo, em uma simulação cirúrgica, a falta de informações com imagens de resolução alta pode resultar em problemas.

No algoritmo *SurfaceNets*, pequenos detalhes de um objeto na imagem são perdidos por gerar isossuperfícies com pouco contraste. No algoritmo *Dual SurfaceNets*, também perdem-se pequenos detalhes; porém, um pouco menos em relação ao algoritmo *SurfaceNets*.

O *Dual SurfaceNets* combina várias informações do algoritmo *SurfaceNets*, a fim de gerar uma isossuperfície de qualidade melhor que o algoritmo *SurfaceNets*. Nessa combinação de informações, ambos os conjuntos de dados precisam estar no mesmo sistema de coordenadas.

O algoritmo *Dual SurfaceNets* combina varreduras com resoluções diferentes, ou seja, uma varredura tem resolução maior do que a outra. Uma média ponderada das varreduras combinadas é realizada. Caso duas varreduras apresentem diferenças, então, é descartada a varredura com menor resolução. A varredura de maior resolução analisada é comparada novamente com outra varredura de resolução ainda maior e assim sucessivamente. Quando duas varreduras seguidas tiverem soluções similares, então, é gerada a isossuperfície.

O *Dual SurfaceNets* gera isossuperfícies mais fiéis aos dados originais do que o *SurfaceNets*. No algoritmo *Dual SurfaceNets*, algumas irregularidades da isossuperfície gerada pelo algoritmo *SurfaceNets* são diminuídas. O *Dual SurfaceNets* gera uma isossuperfície com triângulos com maior qualidade em relação aos algoritmos *Marching Cubes*, porém não se pode afirmar que o *Dual SurfaceNets* gera uma isossuperfície final com qualidade superior ao *Marching Cubes* porque o *Dual SurfaceNets* elimina detalhes do objeto sendo reconstruído.

2.3.8 Algoritmo *Dual Contouring*

O algoritmo *Dual Contouring* (JU *et al.*, 2002) é baseado no algoritmo *SurfaceNets* (GIBSON, 1998), abordado na subseção 2.3.6, na página 32. O *Dual Contouring* é mais relacionado com o *SurfaceNets* do que com o *Marching Cubes*.

A motivação dos autores do *Dual Contouring* foi a construção de um jogo de computador que permitisse a destruição interativa do cenário. O cenário é representado por um campo escalar que deveria ser poligonizado a cada modificação realizada pelo usuário. Esse jogo deu origem ao algoritmo *Dual Contouring*. O algoritmo *Dual Contouring* gera malhas com qualidade melhor e com um custo computacional pior que o algoritmo *Marching Cubes*. O algoritmo *Dual Contouring* apresenta um baixo erro de aproximação em regiões de curvatura elevada e também realiza divisão adaptativa das células.

No algoritmo *Dual Contouring*, os pontos da amostragem são colocados no interior da célula e não nas suas arestas. O *Dual Contouring* é adaptativo, ou seja, algumas regiões da imagem são mais subdivididas em células do que outras. Com isso, a posição de um ponto da amostragem não afeta os pontos de amostragem vizinhos. Isso não pode ser realizado nos algoritmos *Marching Cubes* e *Marching Tetrahedra* (JU *et al.*, 2002).

O algoritmo *Dual Contouring* parte de um espaço tridimensional inicial de apenas uma célula. Essa célula é particionada em oito células iguais, representadas por uma *octree*. Estruturas *octree* são utilizadas, principalmente, para a partição de um espaço tridimensional, dividindo-o em oito octantes de forma recursiva, representada na figura 2.9. Em uma *octree*, cada nó representa uma célula e essa célula pode ser dividida em oito células de mesmo volume, que são representadas por oito nós filhos.

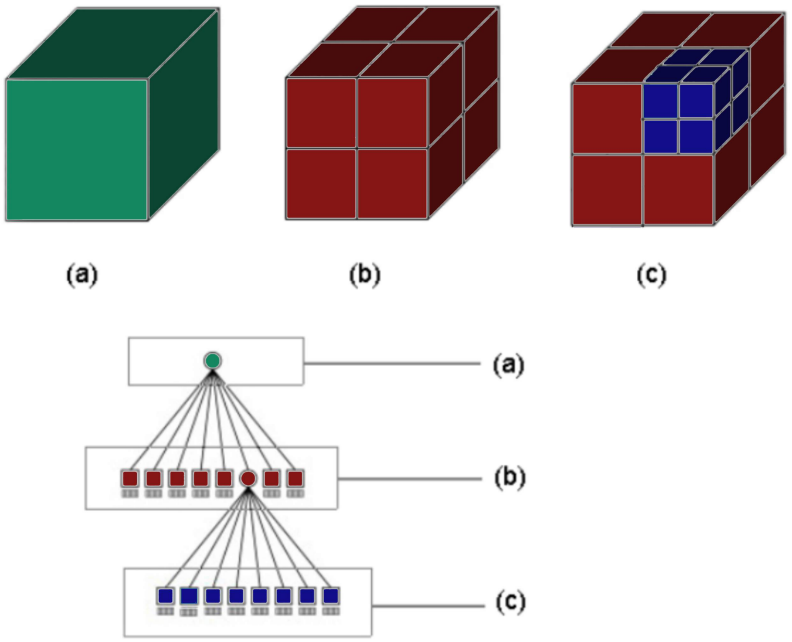


Figura 2.9: Divisão de uma célula em *octree*.

Após a criação de uma *octree* em uma célula, essa célula se torna um nó pai e é dividida em oito nós filhos. Esses oito nós filhos são testados. Se algum desses nós filhos possuir um valor que representa uma intersecção com a superfície de um objeto, então, esse nó filho se torna um nó pai ao ser dividido em oito nós filhos que serão testados. Um nó filho é excluído quando seu valor representar que a célula está fora da região do objeto. Esse processo é repetido até que não reste nenhum nó filho a ser excluído. Com isso, as células que não foram excluídas formam um objeto, no contexto de visualização. A isossuperfície desse objeto é gerada pelos polígonos resultantes de ligações entre células vizinhas, que são os nós filhos restantes. Exemplos dos algoritmos *Marching Cubes* e *Dual Contouring* são ilustrados na figura 2.10.

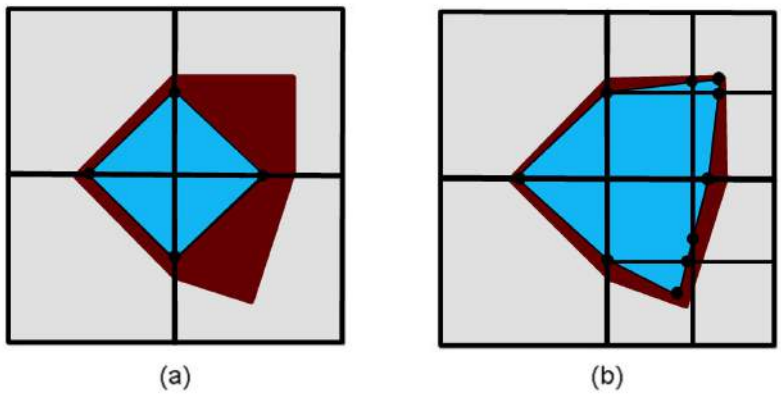


Figura 2.10: Exemplos de contornos em uma região pelo algortimo *Marching Cubes*, representado na figura (a) e pelo algoritmo *Dual Contouring*, representado na figura (b). A região vermelha representa o objeto a se gerar a isossuperfície. A região azul representa a isossuperfície gerada em cima do objeto. A região cinza representa a parte fora da isossuperfície.

2.3.9 Algoritmo *Extended Marching Cubes*

O algoritmo *Extended Marching Cubes* (KOBBELT *et al.*, 2001) foi projetado para gerar malhas com melhor qualidade em cantos e em curvaturas de um objeto do que em relação ao algoritmo *Marching Cubes*. Isso porque, com o algoritmo *Marching Cubes*, essas regiões ficam, na maioria das vezes, com muitas irregularidades. O algoritmo *Extended Marching Cubes* mantém a simplicidade do algoritmo *Marching Cubes*, porém, gera uma isossuperfície mais fiel ao objeto detectado.

O algoritmo *Extended Marching Cubes* segue as etapas iniciais do método *Marching Cubes*. Para cada célula da imagem, é utilizada a tabela do *Marching Cubes* para gerar a triangulação. A diferença entre o *Extended Marching Cubes* e o *Marching Cubes* é que os triângulos da isossuperfície gerada no algoritmo *Extended Marching Cubes* são testados um por um e no algoritmo *Marching Cubes* eles são simplesmente gerados. Nesse teste feito nos triângulos, são analisados vetores normais dentro de cada triângulo. Se o ângulo desses vetores for maior que um valor de entrada do algoritmo, valor este, definido para a geração da isossuperfície,

então, é inserido um ponto na célula a que pertence o triângulo. Pelo ângulo dos vetores normais, um ponto é marcado sobre uma aresta ou sobre um vértice da célula e sobre esse ponto será gerado a isossuperfície.

Após percorrer todas as células da imagem, é gerada uma malha pelos pontos que foram marcados nas células em um vértice ou em uma aresta. A malha é percorrida e ligações entre os pontos são realizados, modificando localmente a conexão da malha. Com essa modificação, pontos de canto ou de curvatura da imagem ficam menos irregulares em relação à malha gerada pelo *Marching Cubes*. Exemplos em duas dimensões do resultado dos algoritmos *Marching Cubes* e *Extended Marching Cubes*, são ilustrados na figura 2.11.

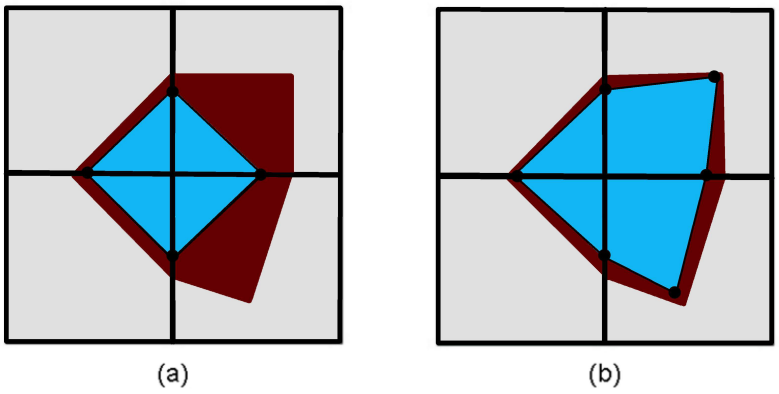


Figura 2.11: Na figura (a), ilustra-se um exemplo em imagem bidimensional da geração de um contorno pelo algoritmo *Marching Cubes*. Na figura (b), ilustra-se a geração de um contorno pelo algoritmo *Extended Marching Cubes* do mesmo exemplo ilustrado na figura (a). A região azul corresponde ao contorno gerado. A região vermelha corresponde ao objeto que se quer gerar o contorno. O contorno gerado pelo algoritmo *Extended Marching Cubes* cobre melhor a área de um mesmo objeto, em relação a um contorno gerado pelo algoritmo *Marching Cubes*.

O algoritmo *Extended Marching Cubes*, pelo fato de realizar mais cálculos que o algoritmo *Marching Cubes* e gerar mais pontos dentro das células de intersecção, demora mais para gerar a isossuperfície do que o *Marching Cubes*. Entretanto, o

algoritmo *Extended Marching Cubes* gera isossuperfície com mais qualidade que o algoritmo *Marching Cubes* (KOBBELT *et al.*, 2001).

2.3.10 Algoritmo de Lewiner-Lopes-Vieira-Tavares

O algoritmo de Lewiner-Lopes-Vieira-Tavares (LEWINER *et al.*, 2003) foi projetado para gerar isossuperfícies com qualidade melhor do que o algoritmo *Marching Cubes*. O algoritmo de Lewiner-Lopes-Vieira-Tavares é baseado em uma técnica descrita em Chernyaev (1995), chamada de *Marching Cubes 33*. Na técnica *Marching Cubes 33*, a tabela de casos do algoritmo *Marching Cubes* foi ampliada de 15 casos para 33 casos. O algoritmo de Lewiner-Lopes-Vieira-Tavares também foi projetado para não causar os problemas de ambiguidade em uma célula na geração de isossuperfícies, mesmo problema citado no algoritmo *The Asymptotic Decider*, descrito na subseção 2.3.2, na página 26.

Para essa expansão da tabela de casos do algoritmo *Marching Cubes*, alguns casos originaram outros casos. Na figura 2.12, ilustra-se um exemplo da extensão do caso 6 da tabela de casos do algoritmo *Marching Cubes* em outros dois casos.

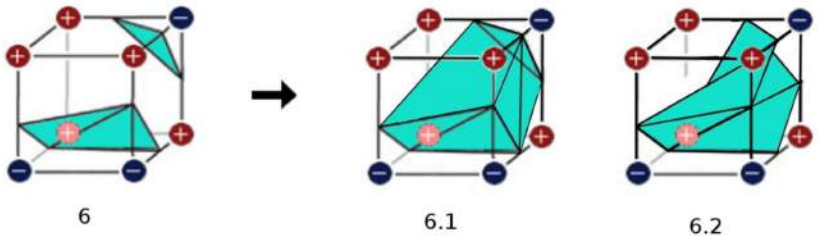


Figura 2.12: Exemplo da expansão do caso 6 da tabela de casos do algoritmo *Marching Cubes* em outros dois casos. Em uma célula com os mesmos vértices marcados, pode-se gerar trechos da isossuperfície de formas diferentes. Esses formatos diferentes dos trechos em cada uma das células são importantes para que a isossuperfície gerada seja mais fiel ao objeto a ser poligonizado.

O algoritmo de Lewiner-Lopes-Vieira-Tavares gera isossuperfícies com um pior desempenho computacional em relação ao algortimo *Marching Cubes*. O algoritmo de Lewiner-Lopes-Vieira-Tavares utiliza uma tabela de 33 casos para fazer comparações e, no pior caso, demora mais que o dobro do tempo do algoritmo *Marching Cubes*, que utiliza uma tabela de 15 casos.

2.3.11 Algoritmo *Dual Marching Cubes*

O algoritmo *Dual Marching Cubes* (NIELSON, 2004) é, praticamente, a junção das técnicas do algoritmo *Marching Cubes* com o algoritmo *Dual Contouring*. O objetivo do algoritmo *Dual Marching Cubes* é eliminar aspectos negativos produzidos por triângulos de baixa qualidade e com isso, melhorar a qualidade da isossuperfície gerada em relação ao *Marching Cubes*.

No algoritmo *Dual Marching Cubes*, aumentam-se dos 15 casos da tabela de casos do algoritmo *Marching Cubes* para 23 casos. O algoritmo *Dual Contouring* estabelece pontos no interior de uma célula analisada, que é a base para a geração da isossuperfície.

Já o algoritmo *Marching Cubes* estabelece pontos nas arestas de uma célula analisada, que também é base para a geração da isossuperfície. O algoritmo *Dual Marching Cubes* realiza uma junção do conjunto de pontos gerados pelo algoritmo *Dual Contouring*, com o conjunto de pontos gerados pelo algoritmo *Marching Cubes*. Com isso, o algoritmo *Dual Marching Cubes* gera isossuperfícies com base em mais dados que o algoritmo *Marching Cubes*.

O algoritmo *Dual Marching Cubes* permite que o usuário defina o nível de precisão da malha final. Com o *Dual Marching Cubes*, é possível atingir um bom nível de detalhamento sem a necessidade de um grande número de subdivisões, produzindo-se uma superfície com menos irregularidades que o algoritmo *Marching Cubes*. Porém, o processamento é mais lento em comparação com o

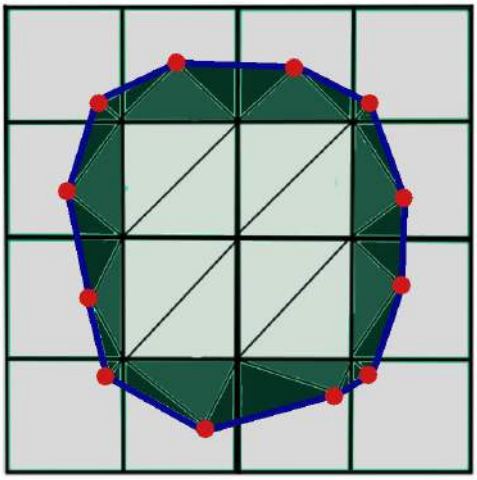
algoritmo *Marching Cubes* e o com o algoritmo *Dual Contouring* (SCHAEFER; WARREN, 2004).

2.3.12 Algoritmo de Zhang-Bajaj-Sohn

O algoritmo de Zhang, Bajaj e Sohn (2005) gera isossuperfícies baseado em cálculos que utilizam valores dos vértices das arestas de cada célula da imagem para criar pontos de amostragem dentro das células. A partir desses pontos, são geradas malhas tetraédricas ou hexaédricas em cada célula que intersecta a isossuperfície. A união dessas malhas forma a isossuperfície. Essa função que gera pontos dentro da célula é também utilizada no algoritmo *Dual Contouring*, descrito na seção 2.3.8, na página 35.

O algoritmo de Zhang-Bajaj-Sohn verifica todas as células da imagem. Após a geração de um ponto no interior de cada célula que intersecta a isossuperfície, é realizada uma ligação desses pontos com vértices da própria célula e com pontos das células adjacentes. Ao final desse processo, malhas tetraédricas ou hexaédricas são geradas e da união dessas malhas é gerada a isossuperfície. Na figura 2.13, ilustra-se, em uma imagem bidimensional, um exemplo de uma malha gerada pelo algoritmo de Zhang-Bajaj-Sohn.

Segundo Zhang, Bajaj e Sohn (2005), o algoritmo de Zhang-Bajaj-Sohn gera isossuperfícies com qualidade superior às isossuperfícies geradas pelo algoritmo *Marching Cubes*. O algoritmo de Zhang-Bajaj-Sohn gera uma isossuperfície com menos irregularidades em relação ao algoritmo *Marching Cubes*. Porém, gera isossuperfícies com um desempenho computacional pior em relação ao algoritmo *Marching Cubes*.



.

Figura 2.13: Exemplo de um contorno gerado pelo algoritmo de Zhang-Bajaj-Sohn em uma imagem bidimensional. Os pontos em vermelho correspondem aos pontos gerados dentro de cada célula que intersecta a região que se quer contornar. As duas tonalidades de verde escuro representam a parte de dentro do contorno em cada célula que intersecta a isossuperfície. A linha em azul corresponde ao contorno final.

2.3.13 Algoritmo *HistoPyramids*

O algoritmo *HistoPyramids* (DYKEN *et al.*, 2008) é baseado em uma estrutura de dados hierárquica que permite a expansão e a compactação de dados. O algoritmo *HistoPyramids* foi projetado para gerar isossuperfícies com qualidade melhor que a gerada pelo algoritmo *Marching Cubes*.

O algoritmo *HistoPyramids* consiste em duas fases. Na primeira fase, os elementos de entrada são mapeados do domínio tridimensional para o domínio bidimensional. É criada uma pilha de texturas em uma estrutura de dados em pirâmide. Essas texturas são valores de células armazenadas no mesmo nível. A cada nível

da pilha, o tamanho da textura de um nível inferior é quatro vezes maior que o tamanho da textura de um nível superior (DYKEN *et al.*, 2008). A maior textura, que é a base da pilha, corresponde à textura de base da pirâmide. Os dados dessa base da pirâmide, de acordo com os seus valores, são compactados para um nível menor, que é compactado para outro nível ainda menor e assim, sucessivamente, até chegar a um nível mínimo, chamado de topo da pirâmide. Na figura 2.14, ilustra-se o processo de compactação dos valores das células em uma estrutura em pirâmide.

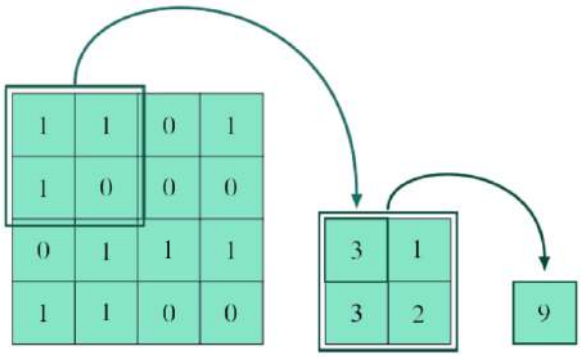


Figura 2.14: Processo de formação da pirâmide. O último nível representa o valor do total de elementos de saída da pirâmide.

Na segunda fase, o algoritmo expande os dados compactados da pirâmide até encontrar algum valor que foi informado. O algoritmo *HistoPyramids* compacta e expande os dados armazenados na pirâmide de acordo com um valor de entrada do algoritmo. O que define a compactação ou a expansão dos dados é um valor informado para a geração da isossuperfície. Caso o valor informado for alto em relação aos valores da imagem, será realizada uma expansão dos dados. Caso o valor informado for baixo em relação aos valores da imagem, será realizada uma compactação dos dados.

No algoritmo *HistoPyramids*, a geração de isossuperfície começa no último nível, ou topo da pirâmide. Os valores de cada nível são analisados até se encontrar um valor referente a um valor que foi informado. A célula que possuir um valor correspondente ao valor informado será interseccionada pela isossuperfície gerada. Caso os valores de um nível da pirâmide não correspondam com o valor informado, é realizada uma expansão do nível atual. Essa expansão cria um novo nível, inferior ao nível atual e os valores desse novo nível serão analisados. Esse processo de expansão de níveis é realizado até que todos os valores de um nível correspondam ao valor informado. Assim, uma isossuperfície pode ser gerada de uma forma mais organizada e com qualidade superior ao algoritmo *Marching Cubes*. Mas isso leva a um desempenho computacional pior em comparação com o algoritmo *Marching Cubes*.

2.3.14 Algoritmo Macet (*Marching Cubes with Edge Transformation*)

O algoritmo Macet (DIETRICH, 2008; DIETRICH *et al.*, 2008; DIETRICH *et al.*, 2009a; DIETRICH *et al.*, 2009b) gera malhas de melhor qualidade, com a mesma simplicidade e a mesma robustez que o algoritmo *Marching Cubes*. Mais especificamente, segundo Dietrich (2008), o Macet gera malhas com qualidade de aproximação igual ou melhor que a qualidade do algoritmo *Marching Cubes* e, o mais importante, gera malhas em que os triângulos de baixa qualidade ainda têm qualidade igual ou superior às malhas geradas pelo algoritmo *Marching Cubes*. Porém, o Macet apresenta um custo computacional maior que o algoritmo *Marching Cubes*.

Segundo Labelle e Shewchuk (2007), o *Marching Cubes* é capaz de extrair triângulos de qualidade alta e baixa. O que difere as duas situações é o modo como a célula intersecciona a isossuperfície. Dessa forma, ao analisar a relação da célula com a isossuperfície, é possível identificar e evitar as situações em que

triângulos de qualidade baixa são gerados e, consequentemente, malhas de boa qualidade podem ser geradas.

O maior mérito dos autores do Macet foi decompor os 256 casos do *Marching Cubes* em um *grupo de arestas*. Os autores do Macet mostraram que a disposição das arestas em cada grupo é o fator responsável pela qualidade dos triângulos gerados. São oito grupos de arestas, mostrados na figura 2.15, que correspondem a oito formas possíveis de um triângulo interseccionar três arestas da célula.

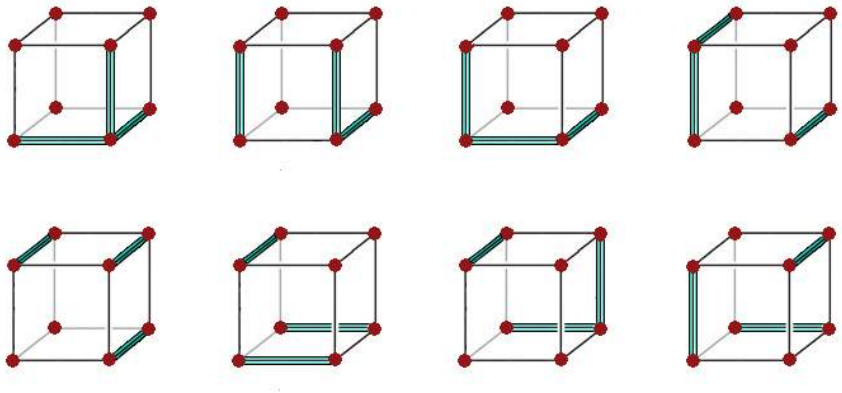


Figura 2.15: Oito grupos de arestas que abrangem todos os grupos de arestas ativas possíveis que podem gerar triângulos no interior de uma célula cúbica.

Com os experimentos realizados, Dietrich (2008) e colaboradores (DIETRICH *et al.*, 2009b) mostraram que alguns dos oito grupos de arestas são responsáveis pela maior parte de triângulos de baixa qualidade gerados pelo *Marching Cubes*. O estudo do grupo de arestas tornou possível identificar e reduzir os casos em que ocorre a maior parte da formação de triângulos de baixa qualidade. Há três modificações para a malha ser gerada de forma melhor.

1. A primeira modificação é na tabela de triangulações do *Marching Cubes*. Nos casos representativos do *Marching Cubes*, 8 dos 15 casos admitem triangulações alternativas, sem que a topologia da malha seja alterada. Isso

possibilita reduzir a probabilidade de uma célula produzir triângulos de baixa qualidade. Outra modificação é realizada para algum dos oito grupos de arestas que apresentam triângulos de baixa qualidade. Essa modificação é realizada nos vértices desses triângulos.

2. A segunda modificação é na deformação da grade do *Marching Cubes*. O formato de cada célula influi no formato da célula vizinha. A deformação da célula elimina algumas conexões próximas de células vizinhas. Isso resulta no colapso de alguns triângulos de baixa qualidade e melhora a consistência da malha gerada.

3. A terceira modificação é no deslocamento dos vértices de intersecção ao longo das arestas ativas. A qualidade de cada triângulo formado dentro da célula é determinada pela posição dos vértices de intersecção ao longo das arestas ativas. Com isso, a modificação do posicionamento do vértice na aresta resulta na transformação de um triângulo de má qualidade em um triângulo de boa qualidade.

Segundo Dietrich (2012), o algoritmo Macet ainda é um trabalho incompleto e precisa de algumas melhorias. Mais simplicidade na implementação e mais qualidade na isossuperfície gerada são as duas principais melhorias possíveis no algoritmo.

2.3.15 Resumo

Na seção 2.3, foram apresentados o algoritmo *Marching Cubes* e 14 variações do algoritmo *Marching Cubes*. Todas as variações do *Marching Cubes* apresentam algumas características diferentes das demais variações. Na figura 2.16, mostram-se todas as variações do algoritmo *Marching Cubes* descritas neste trabalho e suas relações de qualidade e custo computacional com o *Marching Cubes*.

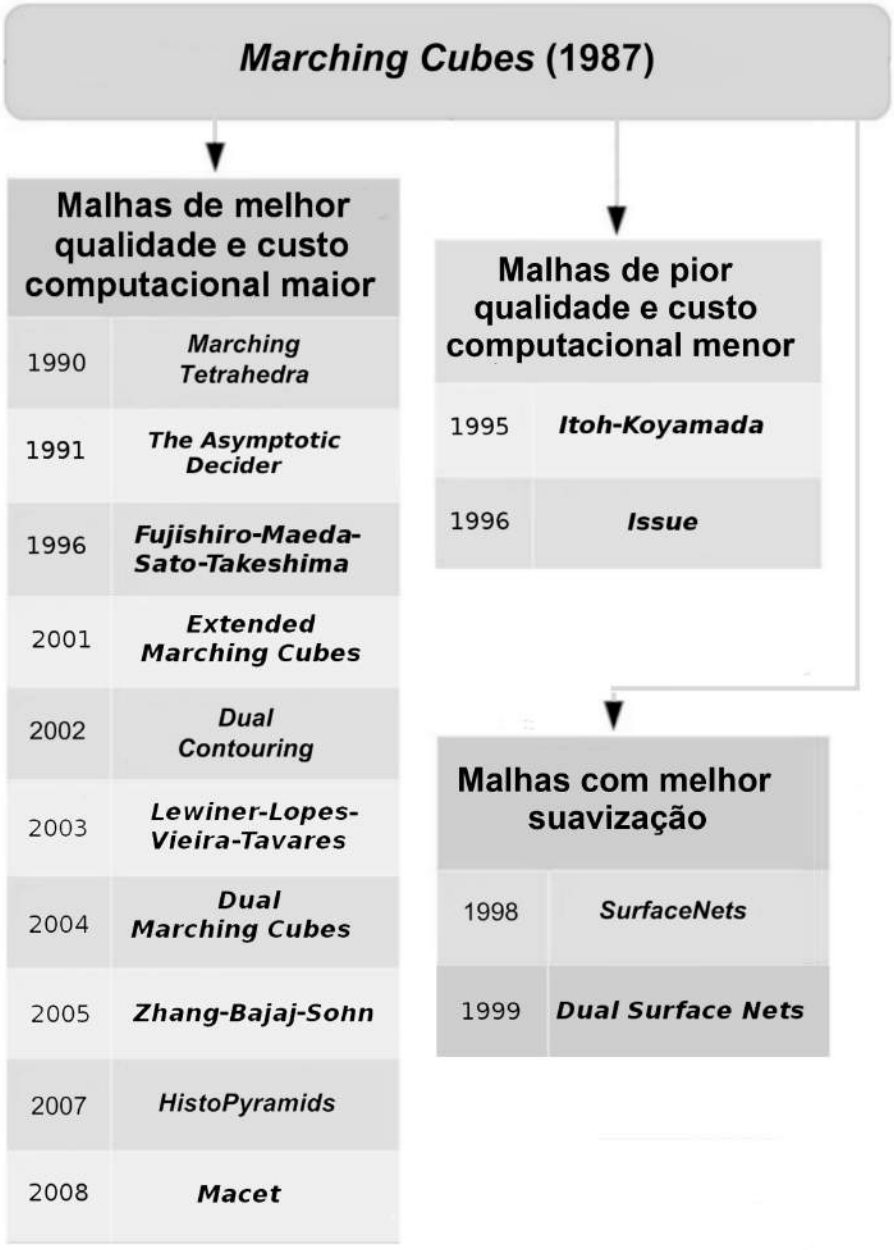


Figura 2.16: Variações do *Marching Cubes*.

3 METODOLOGIA

Neste capítulo, será explicada a metodologia utilizada para o desenvolvimento deste trabalho. Serão explicados os materiais utilizados para a elaboração desse trabalho e a forma de obtenção dos mesmos.

A realização da pesquisa ocorreu no laboratório LaMP - Laboratório de Matemática Computacional e Processamento Gráfico do DCC-UFLA. Para a obtenção de informação, foram utilizados livros, artigos e periódicos. A principal base de pesquisa de periódicos foi a da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Na seção 3.1, aborda-se a classificação para o tipo de pesquisa. Na seção 3.2, descrevem-se as ferramentas utilizadas nos testes. Na seção 3.3, são apresentados os métodos utilizados neste trabalho.

3.1 Classificação para o tipo de pesquisa

Quanto à natureza, essa pesquisa se caracteriza como uma pesquisa aplicada. Segundo Jung (2009), trabalhos teóricos e experimentais sobre fundamentos de fenômenos e fatos com finalidade de aplicação em particular são adequados a uma pesquisa aplicada. Neste trabalho, a geração de isossuperfícies, as propriedades do algoritmo *Marching Cubes* e de suas variações, e a geração de isossuperfícies para testes fazem com que essa pesquisa seja classificada como aplicada.

Quanto aos objetivos, essa pesquisa é classificada como descritiva, pois tem como finalidade observar, registrar e analisar os fenômenos, fatores e ou variáveis. Quanto aos procedimentos, essa pesquisa classifica-se como experimental. Isso porque, assim como neste trabalho, em uma pesquisa experimental, ocorrem simulações e modelagens. Além disso, nesta pesquisa, ocorre a manipulação das

variáveis em um ambiente artificial, isto é, criado e preparado especificamente para a experimentação.

3.2 Ferramentas utilizadas

Na geração de isossuperfícies, utilizou-se um computador HP Pavilion DV6 Notebook PC®, processador Intel(R) Core(TM)2 Duo CPU T6500 @ 2.1GHz (2 CPUs)™, memória RAM de 4096 MB DDR3 1333MHz, cache de 3MB. O sistema operacional utilizado foi o Windows 8.1 Pro 64 bits (6.3, compilação 9600) e Ubuntu 12.04 LTS 64 *bits* com *kernel* versão 3.2.0-38-generic. Utilizou-se, no sistema Windows, o Microsoft Visual Studio 2010 SharePoint Developer Tools 10.0.30319 e no sistema Linux o CodeBlocks 10.5, os dois com *plugin* para C/C++ para a implementação dos testes computacionais.

3.3 Métodos

Para a geração de isossuperfícies pelo algoritmo *Marching Cubes* e suas variações, utilizou-se um projeto computacional desenvolvido por Dietrich (2008). Nesse projeto, o algoritmo *Marching Cubes* ou uma de suas variações é escolhido antes de ser realizado a execução. Uma imagem tridimensional é passada como parâmetro para o algoritmo escolhido. Um valor que representa o nível da isossuperfície a ser gerada também é passada como parâmetro.

4 RESULTADOS

Neste texto, foram apresentadas 14 variações do *Marching Cubes* no capítulo 2. Como pode ser visto em Newman e Yi (2006), há diversas variações do *Marching Cubes*. Segundo Dietrich (2013), seis das mais importantes variações do *Marching Cubes* são: Macet, *Dual Contouring*, *Marching Tetrahedra*, *SurfaceNets*, *Dual SurfaceNets* e o Issue. Neste capítulo, apresentam-se os resultados dos experimentos realizados com o *Marching Cubes* e as seis variações citadas anteriormente. Dos algoritmos descritos neste trabalho que não foram analisados nos experimentos, foram tiradas conclusões obtidas dos artigos e periódicos dos autores desses algoritmos.

Os experimentos foram realizados com dois conjuntos de dados escalares. O primeiro conjunto de dados representa um motor e o segundo conjunto de dados representa um silício modelado. Os dois conjuntos de dados estão disponíveis em http://www9.informatik.uni-erlangen.de/External/vollib/.

A implementação dos códigos dos algoritmos testados foi baseada no projeto de Dietrich (2008). A implementação não contém otimizações, com o objetivo de não beneficiar nenhuma das variações testadas. Em todos os algoritmos, um mesmo isovalor de entrada foi adotado. Esse isovalor correspondente ao nível que será gerada a isossuperfície.

Os experimentos retornam dois resultados de cada algoritmo. O primeiro resultado é o tempo gasto de cada algoritmo para a geração da isossuperfície, medido em milissegundo. O segundo resultado é a qualidade da isossuperfície gerada, medida por três valores. Esses três valores correspondem às métricas de qualidade de triângulos, utilizadas neste trabalho. O primeiro valor representa a métrica de menor ângulo interno dos triângulos da isossuperfície gerada. O segundo valor representa a métrica de maior ângulo interno dos triângulos da isossuperfície ge-

rada. Esses dois primeiros valores são dados em graus. O terceiro valor representa uma medida de qualidade chamada "razão dos raios" determinada pela razão entre os raios dos círculos circunscrito e inscrito de cada triângulos da isossuperfície gerada. Nessa terceira métrica, os seus valores são normalizados no intervalo fechado [0,1], em que valores mais próximos de 1 possuem melhor qualidade.

Na tabela 4.1, apresentam-se os valores do primeiro teste. Na figura 4.1, ilustram-se isossuperfícies geradas de cada algoritmo no primeiro teste. Na tabela 4.2, apresentam-se os valores do segundo teste. Na figura 4.2, ilustram-se isossuperfícies geradas de cada algoritmo no segundo teste.

Tabela 4.1: Resultado de experimentos realizados para avaliar o custo computacional e a qualidade da isossuperfície gerada em cada algoritmo. A métrica 1 representa o valor da métrica de menor ângulo interno. A métrica 2 representa o valor da métrica de maior ângulo interno. A métrica 3 representa o valor da métrica "razão dos raios". A malha tridimensional usada no experimento representa a figura de um motor.

Algoritmo	Tempo (ms.)	Métrica 1	Métrica 2	Métrica 3
Marching Cubes	72489,4	0,3578098	178,8062	0,020110
Dual Contouring	79256,3	0,7480970	169,9729	0,099682
Macet	147962,2	13,7432000	133,8701	0,292740
Marching Tetrahedra	220138,1	10,4387880	145,8380	0,222650
SurfaceNets	81206,8	0,5427008	174,3340	0,147668
Dual SurfaceNets	83872,1	0,5384211	175,4356	0,129490
Issue	49341,9	0,0001078	179,4088	0,003611

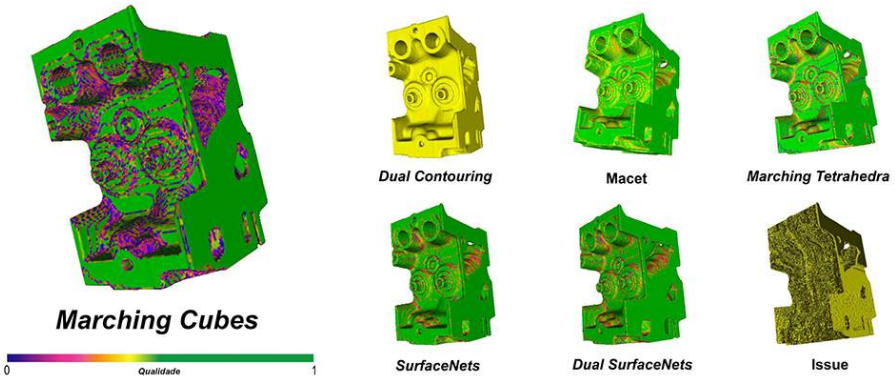


Figura 4.1: Isossuperfícies geradas pelos poligonizadores em um conjunto de dados que representa um motor. As cores da isossuperfície representam a qualidade dos triângulos gerados da determinada região.

Tabela 4.2: Resultado de experimentos realizados para avaliar o custo computacional e a qualidade da isossuperfície gerada em cada algoritmo. A métrica 1 representa o valor da métrica de menor ângulo interno. A métrica 2 representa o valor da métrica de maior ângulo interno. A métrica 3 representa o valor da métrica "razão dos raios". A malha tridimensional usada no experimento representa a figura de um silício modelado.

Algoritmo	Tempo (ms.)	Métrica 1	Métrica 2	Métrica 3
Marching Cubes	3382,32	0,4248009	177,5964	0,097553
Dual Contouring	3506,55	0,6338092	170,5955	0,192257
Macet	11745,90	15,7608205	128,0893	0,403850
Marching Tetrahedra	13912,57	11,6731431	155,3988	0,350111
SurfaceNets	3234,54	0,5376485	171,0348	0,363719
Dual SurfaceNets	3304,72	0,5293840	172,9927	0,353094
Issue	1765,33	0,0003720	179,2009	0,008831

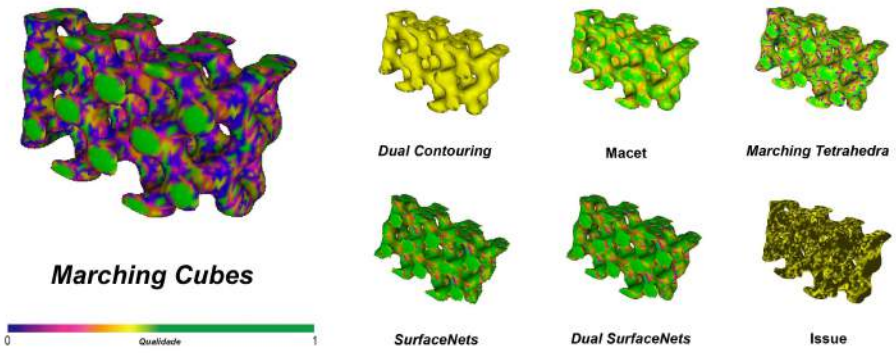


Figura 4.2: Isossuperfícies geradas pelos poligonizadores em um conjunto de dados que representa um silício modelado. As cores da isossuperfície representam a qualidade dos triângulos gerados da determinada região.

5 CONCLUSÕES

Diversos algoritmos poligonizadores de isossuperfície têm o algoritmo *Marching Cubes* como base e buscam gerar malhas de melhor qualidade, melhorar a visualização ou ganhar em custo computacional em relação ao *Marching Cubes*. Pelos valores resultantes dos experimentos, nota-se que existe uma relação entre custo computacional e qualidade de malha gerada. Nessa relação, quanto mais qualidade uma variação ganha em relação ao *Marching Cubes*, maior será o seu custo computacional em relação ao *Marching Cubes*. Na figura 5.1, mostram-se as relações de melhoria de qualidade e custo computacional do algoritmo *Marching Cubes* e de três variações que foram testadas.

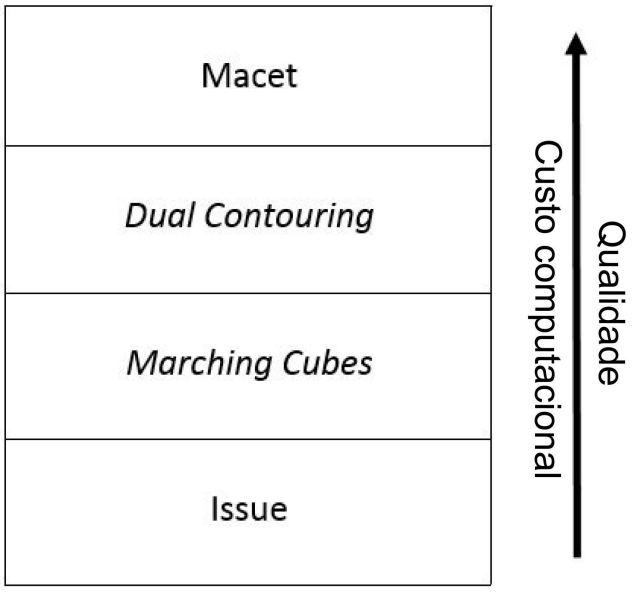


Figura 5.1: Relação de qualidade e custo computacional do Macet, *Dual Contouring*, *Marching Cubes* e Issue.

Numa comparação entre os algoritmos Macet e *Marching Tetrahedra*, pelos valores da tabela 4.1 e 4.2, nota-se que o algoritmo Macet é superior ao *Marching Tetrahedra* em qualidade e também em custo computacional. Por essa razão, o *Marching Tetrahedra* tem um indício de ser descartado.

Os algoritmos *SurfaceNets* e *Dual Surface Nets* geram malhas com melhor suavização em relação ao *Marching Cubes*. Os algoritmos *SurfaceNets* e *Dual Surface Nets* não objetivam gerar uma superfície que seja o mais próximo possível da imagem original. Isso significa que uma imagem gerada após um processo de suavização não é, necessariamente, de melhor qualidade que a imagem original. Grosso modo, uma imagem gerada após um processo de suavização apresenta pouco serrilhado e cantos nítidos. Com isso, entende-se aqui que uma isossuperfície de qualidade significa ser fiel ao objeto sendo reconstruído ou detectado. Isso porque os dados de um objeto a ser reconstruído podem não conter elementos suficientes para que os cantos sejam bem definidos e uma suavização pode ser necessária. Os resultados dos testes para os algoritmos *SurfaceNets* e *Dual SurfaceNets* não podem ser comparados com os resultados dos testes dos demais algoritmos. O *SurfaceNets* e o *Dual SurfaceNets*, ao realizarem o processo de suavização, informações sobre o objeto são perdidas na imagem. Com isso, o valor dos resultados de seus testes não podem ser comparados com os valores das outras variações que não perdem informações. Numa comparação entre os algoritmos *SurfaceNets* e *Dual SurfaceNets*, pelos valores da tabela 4.1 e 4.2, nota-se que o algoritmo *SurfaceNets* é um pouco superior ao *Dual SurfaceNets* em qualidade e também em custo computacional. Por essa razão, o *SurfaceNets* tem um indício de ser melhor que o *Dual SurfaceNets*.

O algoritmo *Dual Contouring* apresenta um custo computacional similar ao custo computacional do *Marching Cubes*. Porém, a qualidade da isossuperfície gerada pelo *Dual Contouring* é bastante superior a qualidade da isossuperfície gerada pelo *Marching Cubes*. Veja as métricas 1, 2 e 3 nas tabelas 4.1 e 4.2.

Além disso, nas figuras 4.1 e 4.2, o *Marching Cubes* gerou triângulos com boa qualidade, mas também muitos triângulos de má qualidade. Por outro lado, o *Dual Contouring* gerou, em geral, triângulos de qualidade média nos dois experimentos.

Em algumas situações, precisa-se gerar isossuperfícies com a maior qualidade possível, sem se preocupar com o custo computacional. Como por exemplo, na área médica, na geração de uma isossuperfície de um órgão. Nesse caso, o algoritmo Macet se destaca. Em situações, como por exemplo, em jogos eletrônicos em tempo real, precisa-se gerar isossuperfícies em um curto espaço de tempo e a qualidade da isossuperfície não é tão importante. Nesse caso o algoritmo *Issue* se destaca. Em situações em que a perda de informações pode ocorrer, mas precisa-se gerar isossuperfícies mais bem acabadas visualmente, o algoritmo *SurfaceNets* se destaca. Na figura 5.2, apresenta-se as melhores utilizações para os algoritmos Macet, *Dual Contouring*, Issue e *SurfaceNets*.

Melhor utilização	**Poligonizador**
Ótima qualidade de isossuperfície sem se importar com o custo computacional	Macet
Boa relação entre benefício (ou seja, qualidade da isossuperfíce) por custo computacional	*Dual Contouring*
Baixo custo computacional sem se importar com a qualidade da isossuperfície	Issue
Isossuperfície com boa visualização	*SurfaceNets*

Figura 5.2: Relação de melhor utilização do Macet, *Dual Contouring*, Issue e *SurfaceNets*.

5.1 Trabalhos Futuros

Em trabalhos futuros, almeja-se realizar novos experimentos com outras variações do *Marching Cubes* que não participaram dos experimentos deste trabalho. Pretende-se realizar novos experimentos com mais de dois conjuntos de dados. Também verificar outras métricas de qualidade a fim de confirmar os resultados obtidos nesse trabalho. Também pretende-se utilizar aplicações práticas de geração de isossuperfícies e testar qual algoritmo é mais apropiado para determinada aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

BABUSKA, I.; AZIZ, A. K. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, New Providence, New Jersey, USA, v. 13, n. 2, p. 214–226, 1976.

CHERNYAEV, E. V. Marching cubes 33: Construction of topologically correct isosurfaces. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. Protvino, Moscow Region, Russia: 142284, 1995. (Technical Report CN/95-17), p. 1–8.

DIETRICH, C. A. *Grupos de Arestas: Uma Nova Abordagem para Entender a Qualidade da Malha Gerada pelo Marching Cubes e suas Variantes*. Tese (Doutorado) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre - RS, 2008.

DIETRICH, C. A. *Geração de isossuperfícies e variações do algortimo Marching Cubes*. 15 de setembro 2012. Comunicação por e-mail.

DIETRICH, C. A. *Geração de isossuperfícies e variações do algortimo Marching Cubes*. 10 de agosto 2013. Comunicação pessoal.

DIETRICH, C. A.; SCHEIDEGGER, C. E.; COMBA, J. L. D.; NEDEL, L. P.; SILVA, C. T. Edge groups: an approach to understanding the mesh quality of marching methods. *IEEE transactions on visualization and computer graphics*, Los Alamitos, CA, USA, v. 14, n. 6, p. 1651–1658, 2008.

DIETRICH, C. A.; SCHEIDEGGER, C. E.; COMBA, J. L. D.; NEDEL, L. P.; SILVA, C. T. Marching cubes without skinny triangles. *Computing in science and engineering*, New York, NY, USA, v. 11, n. 2, p. 82–87, 2009.

DIETRICH, C. A.; SCHEIDEGGER, C. E.; SCHREINER, J.; COMBA, J. L. D.; NEDEL, L. P.; SILVA, C. T. Edge transformations for improving mesh quality of

marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, New York, NY, USA, v. 15, n. 1, p. 150–159, 2009.

DYKEN, C.; ZIEGLER, G.; THEOBALT, C.; SEIDEL, H.-P. High-speed marching cubes using histopyramids. *Computer Graphics Forum*, Blackwell Publishing, Oxford, UK, v. 27, n. 8, p. 2028–2039, September 2008.

FUJISHIRO, I.; MAEDA, Y.; SATO, H.; TAKESHIMA, Y. Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 2, n. 2, p. 144–155, jun. 1996.

GIBSON, S. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In: WELLS, W.; COLCHESTER, A.; DELP, S. (Ed.). *Medical Image Computing and Computer-Assisted Interventation*. Springer Berlin Heidelberg, 1998. (Lecture Notes in Computer Science, v. 1496), p. 888–898. ISBN 978-3-540-65136-9. Disponível em: <http://dx.doi.org/10.1007-/BFb0056277>.

GODOI, W. C. *Reconhecimento de Padrões 3D em Tomografia Industrial*. 1–164 p. Tese (Doutorado) — Instituto de Informática, Universidade Federal do Paraná, Curitiba, Paraná, 2012.

ITOH, T.; KOYAMADA, K. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 1, n. 4, p. 319–327, dez. 1995. ISSN 1077-2626. Disponível em: <http://dx.doi.org/10.1109/2945.485619>.

JU, T.; LOSASSO, F.; SCHAEFER, S.; WARREN, J. Dual contouring of hermite data. In: . New York, NY, USA: ACM, 2002. v. 21, n. 3, p. 339–346.

JUNG, C. F. *Metodologia aplicada a projetos de pesquisa: Sistemas de Informação e Ciência da Computação*. 2009. Disponível em http://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CDUQFjAA&url=http%3A%2F%2Fwww.jung.pro.br%2Fmoodle%2Fmod%2Fresource%2Fview.php%3Fid%3D102&ei=RQEtUbitMoSs8ASupYHwDw&usg=AFQjCNFRrix4NNXuNHO4q_lwPT-iPSInWg&bvm=bv.42965579,d.eWU. Acessado em 10 de fevereiro de 2014.

JUNIOR, V. P. *Métodos implícitos para a reconstrução de superfícies a partir de nuvens de pontos*. 132 p. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo, 2008.

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*. New York, NY, USA: ACM, 2001. (SIGGRAPH '01), p. 57–66. ISBN 1-58113-374-X.

LABELLE, F.; SHEWCHUK, J. R. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph.*, ACM, New York, NY, USA, v. 26, n. 3, jul. 2007.

LEVENTON, M. E.; GIBSON, S. F. F. Model generation from multiple volumes using constrained elastic surfacenets. In: *Proceedings of the 16th International Conference on Information Processing in Medical Imaging*. London, UK, UK: Springer-Verlag, 1999. (IPMI '99), p. 388–393. ISBN 3-540-66167-0. Disponível em: <http://dl.acm.org/citation.cfm?id=645596.660712>.

LEWINER, T.; LOPES, H.; VIEIRA, A. W.; TAVARES, G. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, v. 8, p. 379–386, 2003.

LIVNAT, Y.; SHEN, H.-W.; JOHNSON, C. R. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 2, n. 1, p. 73–84, mar. 1996.

LORENSEN, W. E.; CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 21, n. 4, p. 163–169, ago. 1987.

MAKEHUMAN. *MakeHuman team - Open Source tool for making 3D characters.* 2001. Disponível em: http://www.makehuman.org. Acesso em: 10 de outubro de 2013.

NEWMAN, T. S.; YI, H. A survey of the marching cubes algorithm. *Computers & Graphics*, v. 30, n. 5, p. 854 – 879, 2006. ISSN 0097-8493.

NIELSON, G. M. Dual marching cubes. In: *Proceedings of the conference on Visualization*. Washington, DC, USA: IEEE Computer Society, 2004. p. 489–496.

NIELSON, G. M.; HAMANN, B. The asymptotic decider: resolving the ambiguity in marching cubes. In: *Proceedings of the 2nd conference on Visualization '91*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991. (VIS '91), p. 83–91. ISBN 0-8186-2245-8. Disponível em: <http://dl.acm.org/citation.cfm?id=949607.949621>.

NING, P.; BLOOMENTHAL, J. An evaluation of implicit surface tilers. *IEEE Comput. Graph. Appl.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 13, n. 6, p. 33–41, nov. 1993.

ROSENSTROM, S. *Detecting edge pixels with Marching Squares Algorithm*. 2009. Disponível em: http://www.sakri.net/blog/2009/05/28/detecting-edge-pixels-with-marching-squares-algorithm/. Acesso em 21/04/2012.

SANTOS, A. M. dos; SCHEER, S. *Uma medida para avaliação global da qualidade de malhas triangulares não-estruturadas e uma métrica de qualidade local para seus elementos: A relação perimetral*. 20 p. Dissertação (Mestrado) — Universidade Federal do Paraná – UFPR, Centro de Estudos da Engenharia Civil – CESEC /UFPR – 81531-980 - Curitiba – PR, Brasil, 2006.

SCHAEFER, S.; WARREN, J. Dual marching cubes: Primal contouring of dual grids. In: *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*. Washington, DC, USA: IEEE Computer Society, 2004. p. 70–76. ISBN 0-7695-2234-3.

SCHMITZ, L. A. *Analysis and Acceleration of High Quality Isosurface Contouring*. 80 p. Dissertação (Mestrado) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, out. 2009.

SCHREINER, J.; SCHEIDEGGER, C. E.; FLEISHMAN, S.; SILVA. Direct (re) meshing for efficient surface processing. *Computer graphics forum*, Wiley Online Library, v. 25, n. 3, p. 527–536, 2006.

SHEN, H.-W.; HANSEN, C. D.; LIVNAT, Y.; JOHNSON, C. R. Isosurfacing in span space with utmost efficiency (issue). In: *Proceedings of the 7th conference on Visualization*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1996. p. 287. ISBN 0-89791-864-9.

SHIRLEY, P.; TUCHMAN, A. A polygonal approximation to direct scalar volume rendering. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 24, n. 5, p. 63–70, nov. 1990.

TREECE, G. M.; PRAGER, R. W.; GEE, A. H. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, v. 23, p. 583–598, 1999.

ZHANG, Y.; BAJAJ, C.; SOHN, B.-S. 3d finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, v. 194, n. 48–49, p. 5083 – 5106, 2005.