Pontifícia Universidade Católica
do Rio de Janeiro

Aruquia Barbosa Matos Peixoto

Um esquema de aproximação para superfícies implícitas com redefinição dos pontos de geração da Octree associada

Tese de Doutorado

Tese apresentada ao Programa de Pós–graduação em Engenharia Mecânica do Departamento de Engenharia Mecânica da PUC–Rio como requisito parcial para obtenção do título de Doutor em Engenharia Mecânica

Orientador: Prof. Marcelo Dreux
Co–orientador: Prof. Carlos Antonio de Moura

Rio de Janeiro
Março de 2013

Pontifícia Universidade Católica
do Rio de Janeiro

Aruquia Barbosa Matos Peixoto

Um esquema de aproximação para superfícies implícitas com redefinição dos pontos de geração da Octree associada

Tese apresentada ao Programa de Pós–graduação em Engenharia Mecânica do Departamento de Engenharia Mecânica do Centro Técnico Científico da PUC–Rio como requisito parcial para obtenção do título de Doutor em Engenharia Mecânica. Aprovada pela comissão examinadora abaixo assinada.

Prof. Marcelo Dreux
Orientador
Departamento de Engenharia Mecânica – PUC–Rio

Prof. Carlos Antonio de Moura
Co–orientador
Universidade do Estado do Rio de Janeiro

Prof. Bruno Feijo
Departamento de Informática – PUC-Rio

Pedro Mário Cruz e Silva
Instituto Tecgraf – PUC-Rio

Prof. Aura Conci
Universidade Federal Fluminense

Prof. Luiz Eduardo Azambuja Sauerbronn
Universidade Federal do Rio de Janeiro

Prof. Esteban Walter Gonzalez Clua
Universidade Federal Fluminense

Prof. José Eugênio Leal
Coordenador do Centro Técnico Científico – PUC–Rio

Rio de Janeiro, 06 de março de 2013

Aruquia Barbosa Matos Peixoto

Graduou–se em Matemática na Universidade Federal do Rio de Janeiro (UFRJ) realizando Iniciação Científica sobre métodos numéricos em equações diferenciais parciais no LNCC. Cursou o mestrado em Engenharia de Sistemas e Computação na COPPE/UFRJ, tendo a dissertação de mestrado "Simplificação de Superfícies Implícitas Não-Compactas com Preservação de Topologia" sido apresentada no Workshop de Teses e Dissertações do SIBGRAPI. Trabalhou como professora substituta na UERJ, onde coorientou monografias de fim de curso, iniciação científica e especialização com os professores Paulo Rogerio Sabini e Carlos A. de Moura. Alguns desses trabalhos receberam prêmios de menção honrosa, nas Jornadas de Iniciação Científica no IMPA e na Semana de Iniciação Científica da UERJ (SEMIC/UERJ). Durante o doutorado apresentou trabalhos nos congressos a seguir: CNMAC 2009, CNMAC 2010, VII WVC 2011 (Workshop de Visão Computacional), USNCCM 2011 (US National Congress on Computational Mechanics) e WCCM 2012 (World Congress on Computational Mechanics).

Ficha Catalográfica

Peixoto, Aruquia Barbosa Matos

Um esquema de aproximação para superfícies implícitas com redefinição dos pontos de geração da *Octree* associada / Aruquia Barbosa Matos Peixoto; orientadores: Marcelo Dreux, Carlos Antonio de Moura. – 2013.

v., 92 f: il. (color.) ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2013.

Inclui bibliografia.

1. Engenharia Mecânica – Teses. 2. Superfícies Implícitas. 3. Poligonalização. 4. Geometria Computacional. I. Dreux, Marcelo. II. Moura, Carlos Antonio de. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. IV. Título.

CDD: 621

Em memória do meu querido amigo, amor, companheiro e marido por mais de uma década,
Paulo Rogerio Sabini

Agradecimentos

Os agradecimentos por conseguir realizar esse trabalho, tão pouco tempo depois da perda do meu marido, são muito especiais.

Inicialmente agradeço aos meus orientadores Marcelo Dreux e Carlos de Moura pela revisão cuidadosa do texto, e também por todo o apoio antes e durante o doutorado, muito obrigada por tudo. Falando de orientadores, é claro que eu não poderia esquecer dos orientadores do mestrado e da graduação com quem tanto aprendi, Luiz Velho e Jaime Rivera.

Ao CNPq e à PUC-Rio, pelos auxilios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus colegas do CEFET/NI por tornarem o ambiente de trabalho tão agradável.

Gostaria de agradecer aos meus amigos, que estiveram sempre ao meu lado e me deram forças para continuar: Alessandra, Joice, Fabio, Cassia, Flavia, Luciana, Alexandre, Armando, Teresa, Raimundo e Mannu.

Aos meus novos amigos, é muito bom lembrar que existem pessoas que podem iluminar nossa vida: Paulo Varandas, Thiago e Hoang.

É claro que não posso esquecer da minha querida family-in-heart, que mesmo sofrendo muito com a perda do filho, irmão e cunhado sempre esteve ao meu lado: meus sogros Sadi e Velires, meus cunhados Fabio, Rodrigo e Américo e as minhas cunhadas Marcia, Kathiane e Jussan.

E por ultimo, mas muito importantes para mim, os meus amores, meus sobrinhos-afilhados: Jasmim, Tales, Taila, Maximus e Calebe. Com eles eu conseguia me lembrar que a palavra amor fez, e faz, parte da minha vida.

Resumo

Peixoto, Aruquia Barbosa Matos; Dreux, Marcelo; Moura, Carlos Antonio de. **Um esquema de aproximação para superfícies implícitas com redefinição dos pontos de geração da *Octree* associada**. Rio de Janeiro, 2013. 92p. Tese de Doutorado — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Neste trabalho é apresentado um método adaptativo de poligonalização de superfícies implícitas, a Grade Adaptativa, associado a uma *octree* como estrutura de dados. Os pontos da subdivisão dos cubos dessa *octree* não estão sempre no centro, como na definição clássica, porém mais próximos à superfície. Para obter essa característica, os valores da função são considerados na região de poligonalização. Os pontos da malha que aproximam a superfície são conectados utilizando as arestas dos cubos referentes às folhas dessa *octree*. As arestas pertencentes à interseção de três ou quatro cubos são testadas com relação a sua intersecção pela superfície. Elas são obtidas por meio de uma redefinição do conceito de Arestas Mínimas. O método apresentado nesta tese conduz a resultados mais precisos que aqueles obtidos com métodos em que a *octree* tenha sua subdivisão sempre no centro, como o *Dual Contouring*. O melhor posicionamento da grade torna possível captar mais detalhes com o mesmo nível de subdivisões, pois nesse caso há mais precisão no posicionamento dos vértices da malha e, além disso, mais cubos intersectam a superfície, gerando mais pontos da malha.

Palavras–chave

Superfícies Implícitas; Poligonalização; Geometria Computacional.

Abstract

Peixoto, Aruquia Barbosa Matos; Dreux, Marcelo (Advisor); Moura, Carlos Antonio de (Co–advisor). **An approximation scheme for implicit surfaces by redefining the associated *Octree* points**. Rio de Janeiro, 2013. 92p. DSc. Thesis — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

This work presents an adaptive method of polygonization for implicit surfaces, the Adaptive Grid, associated to an *octree* as a data structure. The *octree* cubes subdivision points are not always set in the center, as in the classical definition, but closer to the surface. In order to obtain this property, the values of the function are considered in the region of polygonization. The mesh points that approximate the surface are connected using the cubes edges related to the leaves in the *octree*. The edges that belong to the intersection of three or four cubes are tested regarding to its intersection with the surfaces. They are obtained by a redefinition of the Minimal Edges algorithm. The method presented in this work leads to more precise results than the methods for which the *octree* has its subdivision always in the center, as *Dual Contouring*. The better positioning of the grid allows to capture more details with the same subdivision level, because in this case there is more accuracy in the mesh vertex positioning and moreover, more cubes intersect the surface, generating more mesh points.

Keywords

Implicit Surfaces; Polygonization; Computational Geometry.

Sumário

1
Introdução

Superfícies Implícitas possuem uma vasta utilização em Computação Gráfica. Seu emprego vai desde a modelagem de superfícies, para aplicação em filmes e jogos, até a reconstrução de dados adquiridos por meio da ressonância magnética ou da tomografia computadorizada. Em Computação Gráfica usualmente trabalha-se em um espaço tridimensional e as superfícies implícitas são estruturas bidimensionais. Portanto, elas têm uma dimensão a menos do que o espaço em que estão imersas. Procurar os pontos que geram a malha construída para fornecer uma aproximação de uma superfície implícita é um problema análogo ao de encontrar raízes de funções reais de uma variável real.

Um dos métodos mais simples para achar zeros de funções de $\mathbb{R}$ em $\mathbb{R}$ é o da Bissecção (1). Neste método, dado um intervalo inicial, toma-se o ponto no seu centro, gerando sua divisão em dois intervalos de mesmo comprimento. As duas regiões são avaliadas para determinar em qual delas se pode garantir que a função intercepta o eixo x. A outra região é descartada. O processo é repetido até que seja satisfeito um critério de parada, que pode estar associado ao número de iterações, ao comprimento do último intervalo obtido, ou aos valores da função nos extremos desse intervalo, ou ainda a uma combinação dessas condições.

O método da Bissecção tem uma implementação simples, mas sua convergência é linear e, portanto, muito lenta. Isso ocorre porque cada intervalo é dividido no centro, não sendo levada em consideração a possível localização da raiz. Outros métodos que consideram o comportamento da função foram posteriormente desenvolvidos, como os métodos do Ponto Fixo, Newton-Raphson e da Secante (2). Esses métodos dividem o intervalo inicial em dois intervalos de comprimentos diferentes, de tal forma que o ponto escolhido para a geração dos mesmos não esteja necessariamente no centro, porém mais próximo da raiz.

Neste trabalho é proposto um método de poligonalização de superfícies implícitas utilizando a estrutura de dados espaciais *octree* (3), com a grade

associada a esta estrutura de dados convergindo para a superfície, este método é denominado Grade Adaptativa. Assim como os métodos de busca de raízes evoluíram de um método em que a subdivisão é feita sempre no centro para métodos que levam em conta o perfil da função, neste trabalho é proposta uma alternativa aos métodos clássicos de geração da *octree* que utilizam os pontos de subdivisão sempre no centro do cubo, como o *Dual Contouring* (4). Trata-se de um método em que o perfil da superfície é considerado, como pode ser visto nas figuras 1 a) e b).

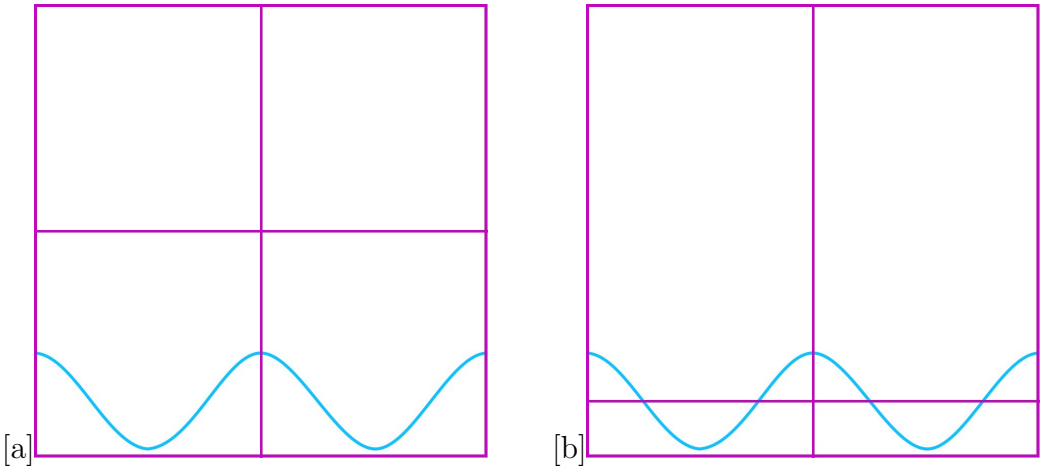


Figura 1: Geração da grade. a) Poligonalização com a subdivisão no centro e, portanto, distante da superfície. b) Subdivisão proposta neste trabalho, que leva em consideração a posição da superfície.

Para gerar a subdivisão com os pontos de subdivisão próximos à superfície são necessárias duas modificações, em relação aos métodos existentes: como definir o ponto de subdivisão e redefinir as conexões entre cubos vizinhos.

Com o método de poligonalização apresentado neste trabalho é possível obter resultados mais precisos com menos subdivisões na *octree*, como pode ser visto na figura 2, onde uma superfície implícita senoidal é reconstruída em três subdivisões da *octree*, com o *Dual Contouring* e com a Grade Adaptativa. O *Dual Contouring*, por escolher as subdivisões sempre no centro do cubo, que neste caso fica distante da superfície, reconstrói a superfície como um plano, figuras 2 a) e b). Com o algoritmo proposto, como as subdivisões estão próximas da superfície, com as mesmas três subdivisões do cubo original torna-se claro o comportamento senoidal da superfície, figuras 2 c) e d).

Os demais capítulos da tese estão organizados da seguinte maneira:

O capítulo 2 apresenta os conceitos básicos, como os de *octree* e de poligonalização adaptativa. O capítulo 3 trata dos principais métodos de poligonalização de superfícies. Os capítulos 4 e 5 apresentam o método proposto, a Grade Adaptativa. No capítulo 4 é discutido como é feita a subdivisão da grade,

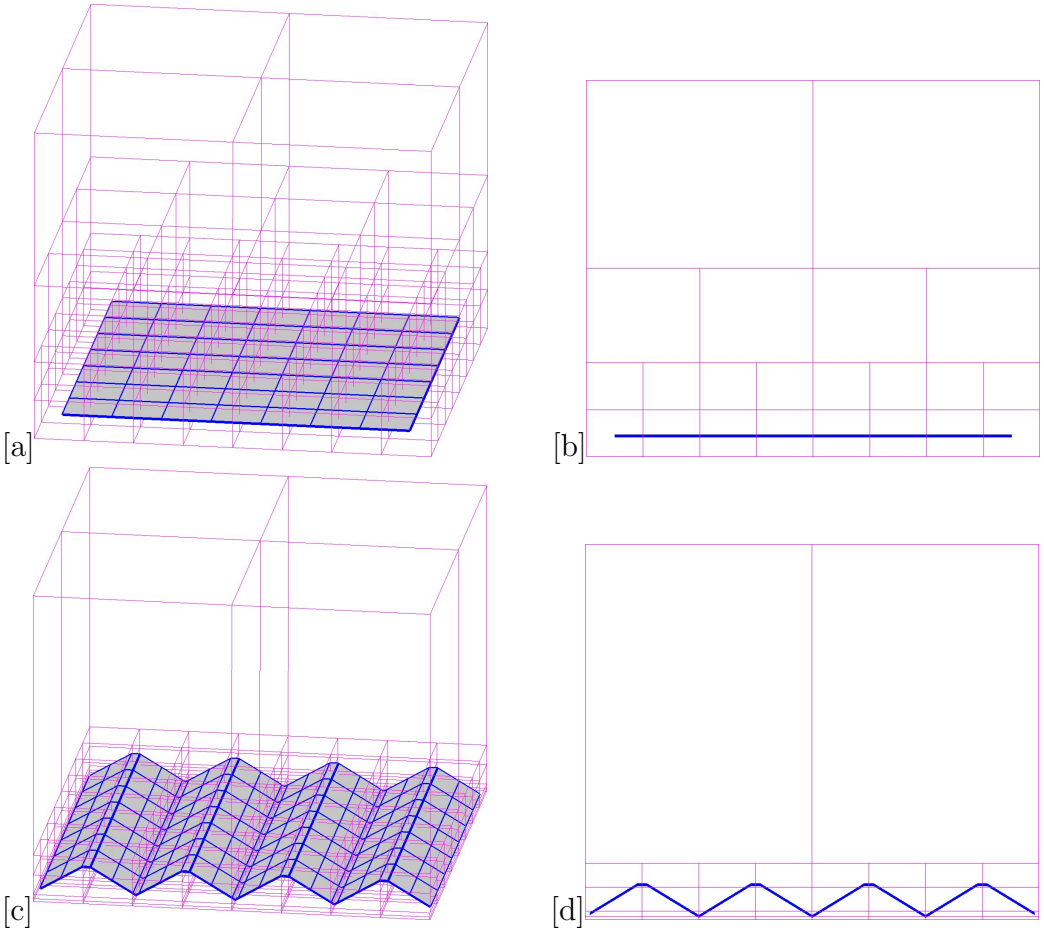


Figura 2: Poligonalização da mesma superfície implícita com o *Dual Contouring* e com o método proposto nesta tese. a) e b) Vista em perspectiva e vista de frente, respectivamente, da superfície poligonalizada com o *Dual Contouring*. c) e d) Vista em perspectiva e vista de frente, respectivamente, da superfície poligonalizada com o método proposto nesta tese.

através de uma escolha mais criteriosa do ponto de subdivisão, e no capítulo 5 é mostrado como fazer a conexão dos pontos para gerar a malha. No capítulo 6 são mostradas superfícies geradas com a Grade Adaptativa, e, finalmente, o capítulo 7 expõe as conclusões deduzidas e propõe trabalhos futuros.

2
Conceitos Preliminares

Em Computação Gráfica, representações de superfícies implícitas podem ser obtidas a partir de uma função, que pode ser dada por uma expressão algébrica, como, por exemplo, $f(x, y, z) = x^2 + yz$, e é chamada de função implícita, ou a partir de uma amostragem, como em uma ressonância magnética em que os valores de densidade de um corpo são armazenados nos vértices de uma grade. A diferença entre estas duas representações é que, ao usar uma expressão algébrica, é possível (teoricamente) ter o valor exato da função em cada ponto do espaço. Já no caso em que os dados são introduzidos a partir de uma amostragem nos vértices de uma grade, para ter o valor em um ponto fora da grade é feita uma interpolação de valores da amostragem próximos a esse ponto e o resultado é o valor aproximado da função implícita nesse ponto.

Independentemente da forma como a superfície implícita é definida, para fazer a poligonalização é utilizada uma grade, que pode ser retangular ou não, em cujos vértices são feitas amostragens dos valores da superfície. Nesta tese está sendo considerado que, sem perda de generalidade, são procurados os pontos em que o valor da função é zero, esta é a superfície de nível definida por $f(x, y, z) = 0$. Portanto, quando houver referência a pontos com sinais distintos, isto equivale a dizer que existem dois pontos $(x1, y1, z1)$ e $(x2, y2, z2)$, tais que a função calculada tem valor positivo em um dos pontos e negativo no outro, o que indica que estes pontos estão em lados opostos com relação à superfície. O valor procurado, neste caso zero, é chamado de isolevel da superfície de nível.

Se a superfície resultante ao fixar o isolevel é compacta, como por exemplo um toro, a superfície divide o espaço em três partes:

- Interior – São os pontos que estão contidos dentro da superfície e têm valor $f(x, y, z) < 0$.
- Exterior – São os pontos que estão fora da superfície e têm valor $f(x, y, z) > 0$.

– Superfície – São os pontos que estão na superfície e têm valor $f(x, y, z) = 0$.

Se a superfície não é compacta, como por exemplo um plano, não é possível definir os pontos como interior ou exterior à superfície, mas sim pontos de lados distintos.

Para criar a poligonalização é gerada uma grade, que na figura 3 é retangular. Assume-se que a grade é fina o bastante para captar os detalhes da superfície, de tal forma que se a superfície passa de um elemento para outro da grade sua passagem é captada pelos vértices, como pode ser visto na figura 3 a), mas caso ela não seja detectada como vemos na parte da direita na figura 3 b), não há necessariamente uma mudança na sua topologia, pois os pontos que foram detectados são conectados, havendo, entretanto, uma perda na informação geométrica, que neste caso é a curvatura da superfície.

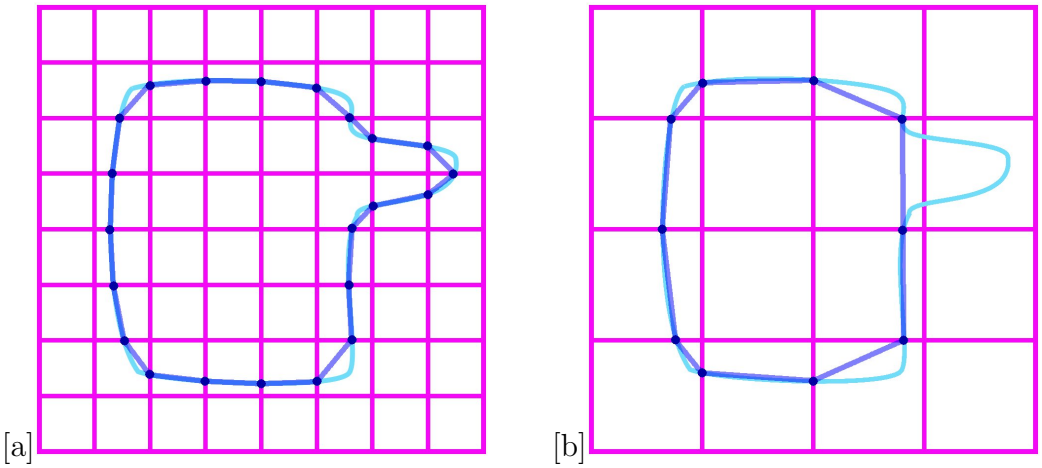


Figura 3: A mesma superfície, que está em azul claro, poligonalizada com duas grades distintas. a) Com a grade mais refinada são captados mais detalhes da superfície. b) Com uma grade menos refinada alguns detalhes são perdidos, mas neste caso a topologia não é alterada.

Após sua geração a grade é percorrida, e em cada um dos seus elementos são testados os valores dos seus vértices. Para uma grade fina o bastante para captar a superfície, existem três casos possíveis com relação aos valores da função nos vértices do quadrado:

– Todos os valores são maiores do que zero – Neste caso a superfície não passa por este quadrado.
– Todos os valores são menores do que zero – Também neste caso a superfície não passa por este quadrado.
– Valores menores e maiores do que zero – Neste caso a superfície passa por este quadrado.

Para gerar a poligonalização é feita uma amostragem, e para guardar os dados dessa amostragem é usada uma grade. Os dois tipos de grade mais utilizados são: simpliciais, (5) e (6), e não-simpliciais, (7) e (8), (9) e (4). Uma grade simplicial é composta por tetraedros em $\mathbb{R}^3$, e em $\mathbb{R}^2$ por triângulos, e a grade não-simplicial mais utilizada é composta em $\mathbb{R}^3$ por cubos e em $\mathbb{R}^2$ por quadrados. Na figura 4, temos exemplos de grades simpliciais e não simpliciais.

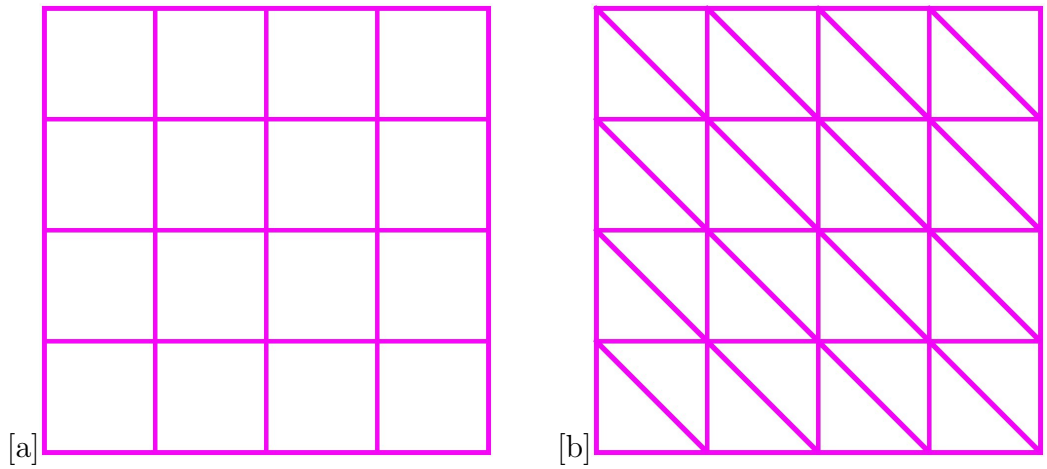


Figura 4: Grades utilizadas na poligonalização em $\mathbb{R}^2$. a) Grade não simplicial. b)Grade simplicial.

Uma malha tetraedral tem a característica de não gerar ambiguidades, pois seus elementos são simpliciais de dimensão 3, mas a implementação destas malhas é mais trabalhosa. Uma malha formada por cubos tem uma implementação mais simples, porém ela pode gerar ambiguidades. Estas ambiguidades ocorrem pois, dado um mesmo posicionamento de pontos dentro e fora da superfície, podem ser geradas duas poligonalizações distintas, como é mostrado na figura 5.

Existem vários métodos para resolver os casos de ambiguidade. Alguns dividem o cubo em tetraedros, e outros não fazem esta subdivisão, mas usam métodos matemáticos para resolver esta ambiguidade, como (10). Neste trabalho é usada uma malha não simplicial.

2.1 Poligonalização Uniforme X Adaptativa

Outra característica importante para a geração da malha é com relação à utilização de uma poligonalização uniforme ou adaptativa.

Como o próprio nome induz, a poligonalização uniforme constrói uma partição uniforme do espaço original, posicionando planos nos eixos x, y e z. Cada elemento dessa partição é um cubo, que chamaremos de voxel. Essa

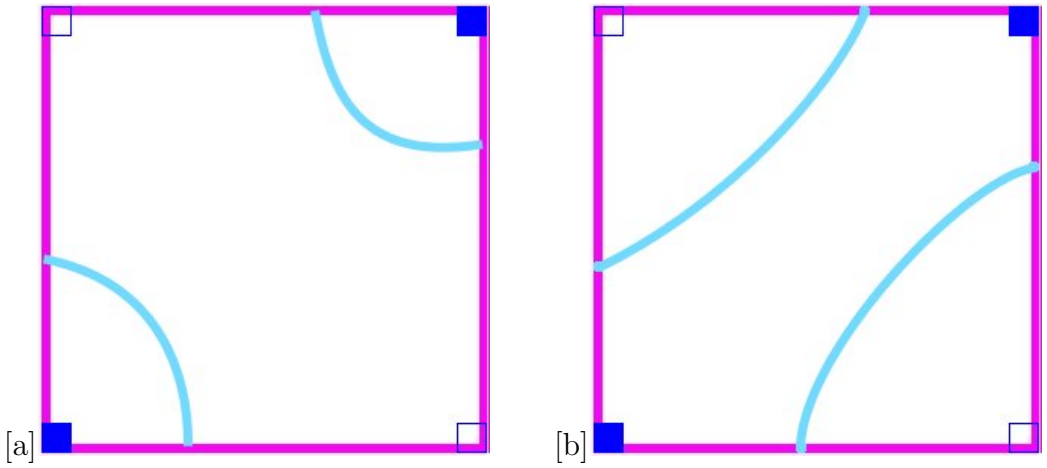


Figura 5: Duas possíveis poligonalizações para o mesmo posicionamento de pontos dentro e fora da superfície. Os vértices preenchidos pertencem ao interior da superfície, e os não preenchidos estão no exterior.

poligonalização é mais simples de implementar, mas tem como desvantagem o fato de gerar uma única partição para todo o espaço, mesmo em regiões pelas quais a superfície não passa, como é possível ver na figura 6 a).

Por outro lado, a poligonalização adaptativa parte do espaço original e gera a sua subdivisão de tal forma que as regiões com mais informações, como, por exemplo, regiões com mais curvatura, são mais subdivididas, como pode ser visto na figura 6 b). Nessa poligonalização é utilizada uma estrutura de dados, associada à subdivisão, que é usualmente uma *octree*, o que será mostrado com mais detalhes na próxima seção.

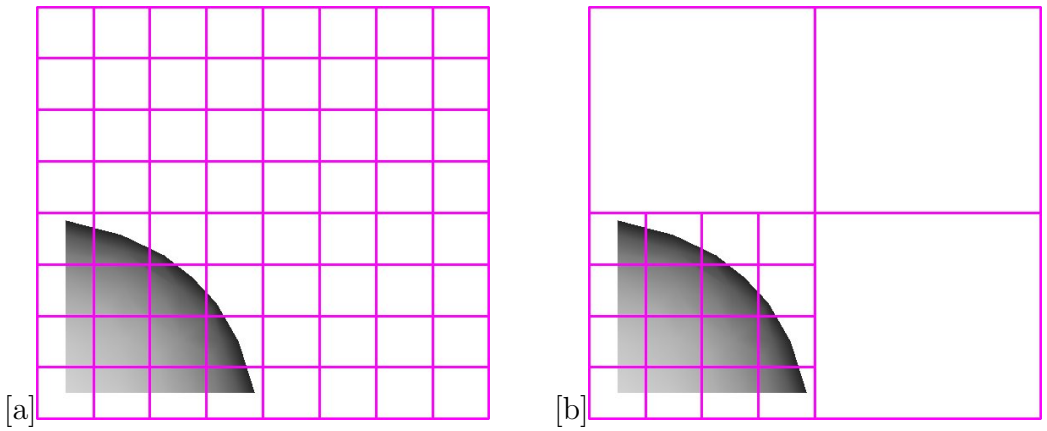


Figura 6: Poligonalização de uma superfície a) Com uma grade uniforme. b) Com uma grade adaptativa.

Para gerar a poligonalização, os voxels são percorridos e para cada voxel que intersecta a superfície são gerados triângulos da poligonalização, ou é inserido um ponto, que posteriormente será conectado com os pontos vizinhos. Na construção da triangularização, para cada aresta que tem sinais distintos

os pontos são posicionados nas arestas do voxel, usando uma interpolação dos valores dos vértices desta aresta para determinar um ponto que aproxime o ponto da superfície que passa por ela.

Ao gerar os triângulos da poligonalização posicionando os vértices da malha nas arestas dos voxels, existem restrições com relação a dois voxels que compartilhem uma aresta. A figura 7 mostra dois cubos vizinhos que se encontram em uma aresta. Como um dos cubos está subdividido, o cálculo do posicionamento do vértice da malha no cubo maior contém um erro maior do que do que aquele associado ao cálculo do posicionamento do vértice no cubo menor, que é resultante da subdivisão do cubo à direita. Com isto surgem posições distintas para o vértice na aresta que conecta estes dois cubos, como pode ser visto na figura 7. Isto gera uma alteração na topologia da superfície original, que é contínua, pois os vértices não se conectam e geram dois segmentos de reta descontínuos.

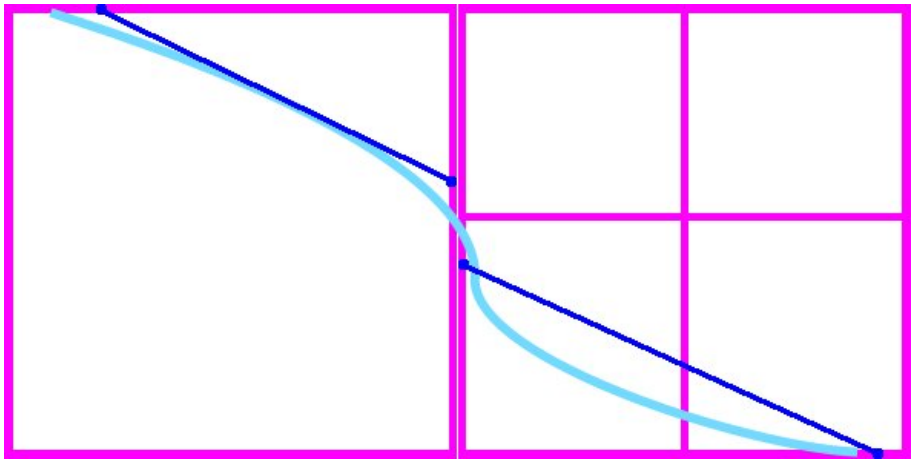


Figura 7: A superfície original está em azul claro, e em azul escuro estão os segmentos de reta usados para gerar a poligonalização. Neste caso os dois segmentos de reta, que se encontram no centro da figura, não se conectam.

Quando os pontos da poligonalização são posicionados no interior do voxel, e não nas suas aresta, não há restrição com relação a voxels vizinhos, a poligonalização pode ser adaptativa, mas neste caso a poligonalização deve ser feita em duas fases: na primeira etapa os pontos são posicionados, e a segunda etapa consiste em conectar estes pontos gerando a malha da poligonalização.

Para conectar os pontos resultantes da amostragem da superfície em cada cubo, é necessário ter uma forma de identificar os cubos vizinhos, que se encontram em aresta.

2.2 Estruturas de Dados Kd-tree e Octree

Construída uma poligonalização adaptativa, a grade correspondente está associada a uma estrutura de dados. As estruturas de dados mais utilizadas são a kd-tree (11) e a *octree* (12). Essas estruturas de dados são do tipo árvore, pois são grafos acíclicos e conexos.

Neste trabalho é utilizada uma *octree*, mas para facilitar a compreensão do motivo da escolha dessa estrutura de dados, é feita uma comparação mais detalhada entre uma *octree* e uma kd-tree.

Tanto a *octree* quanto a kd-tree associam a subdivisão de um cubo a uma árvore, onde cada nó corresponde a um cubo resultante da subdivisão. Neste trabalho toma-se uma poligonalização em $\mathbb{R}^3$, mas para facilitar a compreensão do funcionamento dessas estruturas, nesta seção elas são visualizadas em $\mathbb{R}^2$. Daí resultam uma kd-tree em $\mathbb{R}^2$ e uma *quadtree*, que é a restrição de uma *octree* a $\mathbb{R}^2$.

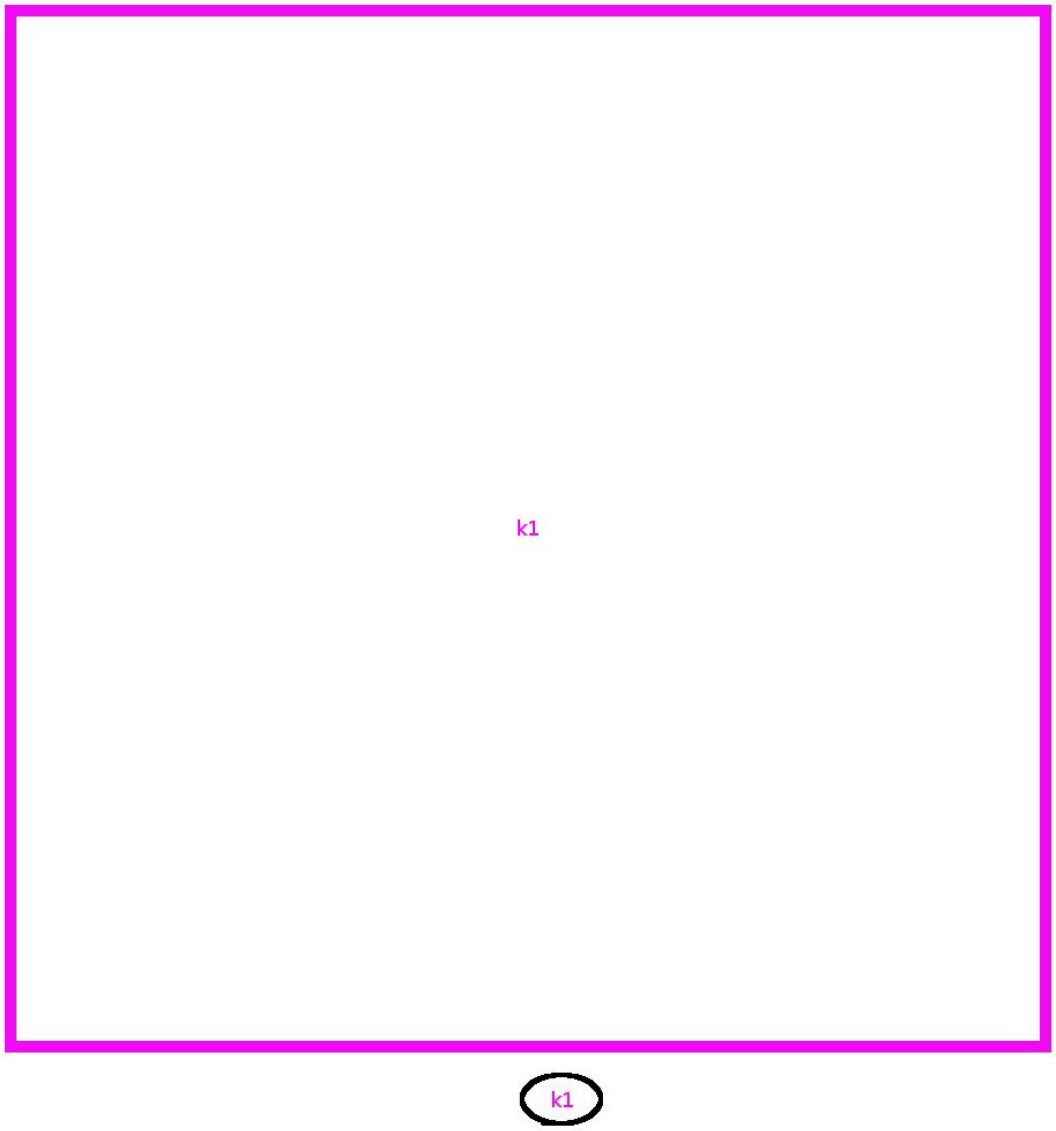
Uma kd-tree é uma árvore binária em que cada nó ou tem dois filhos ou não tem filhos. O nó inicial é associado ao cubo inicial, que é a região onde é feita a poligonalização. A cada passo é utilizada uma reta em $\mathbb{R}^2$ (ou um plano em $\mathbb{R}^3$), que gera a subdivisão em dois conjuntos. Essas retas são paralelas aos eixos x e y e são introduzidas na mesma sequência, alternando suas direções, sendo seu posicionamento definido a partir da posição da superfície.

As figuras 8 a 12 mostram a sequência para efetuar a poligonalização de um quadrado em $\mathbb{R}^2$, relativa a uma curva hipotética, e a kd-tree resultante desta subdivisão. Nesse caso são feitas duas subdivisões no eixo x e duas subdivisões no eixo y, o que resultou em uma árvore cheia com quatro níveis de profundidade.

A primeira subdivisão é feita no eixo x, como mostra a figura 9. Pode-se observar que nesse caso a *octree* gera dois nós no segundo nível, correspondentes aos dois retângulos resultantes da subdivisão.

Ao fazer a segunda subdivisão, no eixo y mostrado na figura 10, pode-se observar que as duas retas verticais estão em posições distintas, em relação ao eixo x. Ao contrário do caso da *quadtree*, figura 16, em que estas retas ficam na mesma posição.

A terceira subdivisão, figura 11, é feita no eixo y e gera oito retângulos, que estão representados pelos oito nós no terceiro nível da kd-tree. Finalmente a figura 12 mostra a quarta subdivisão, ela tem como resultado dezesseis novos retângulos, que estão representados na kd-tree pelos oito nós no quarto nível.



[a]

[b]

Figura 8: Cubo original. a) Cubo sem subdivisão. b) *Kd-tree* somente com o nó raíz.

Para acessar um retângulo resultante da subdivisão no eixo x seguida de uma subdivisão no eixo y é necessário percorrer dois níveis na kd-tree. Além disso, os algoritmos para testar se dois cubos se encontram em uma aresta, ou face, não são tão simples como os algoritmos correspondentes para uma *octree*.

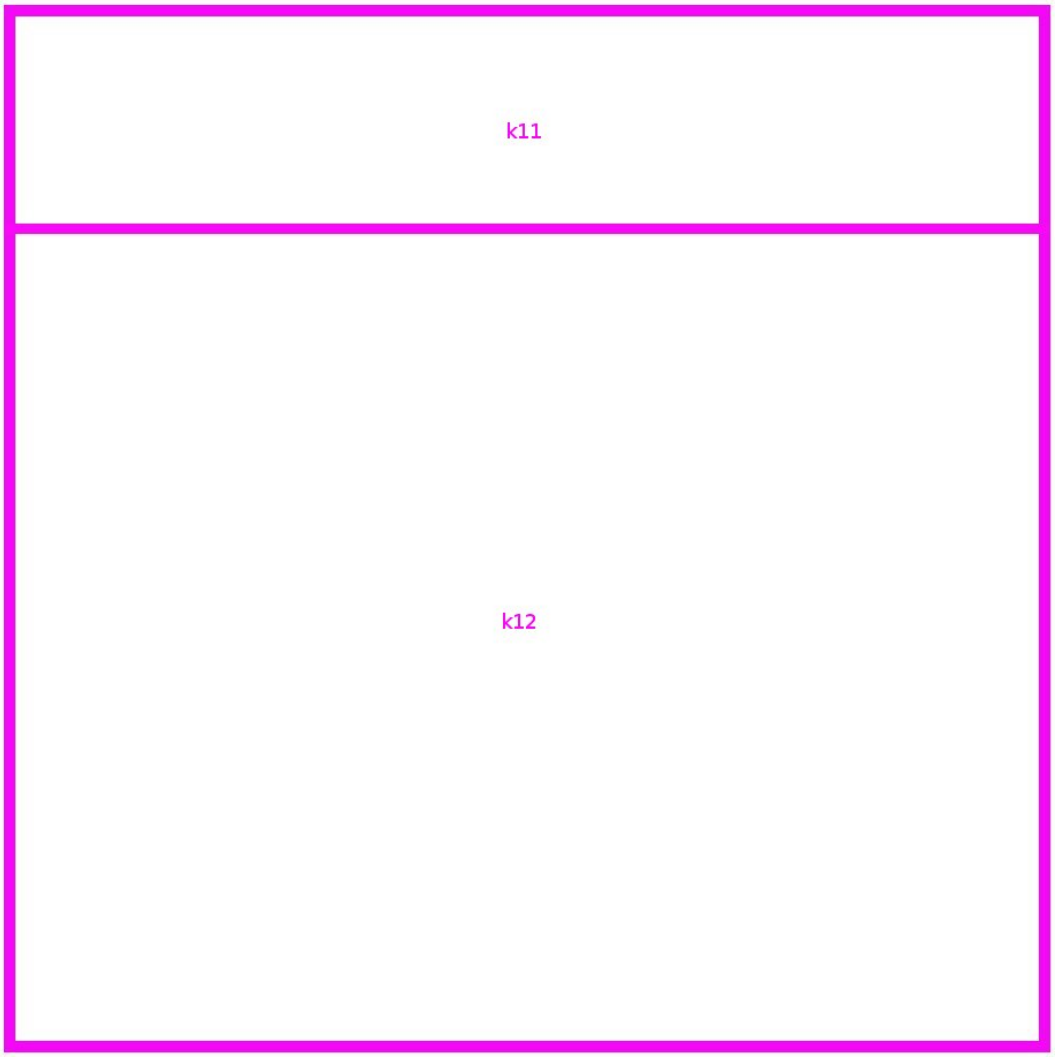
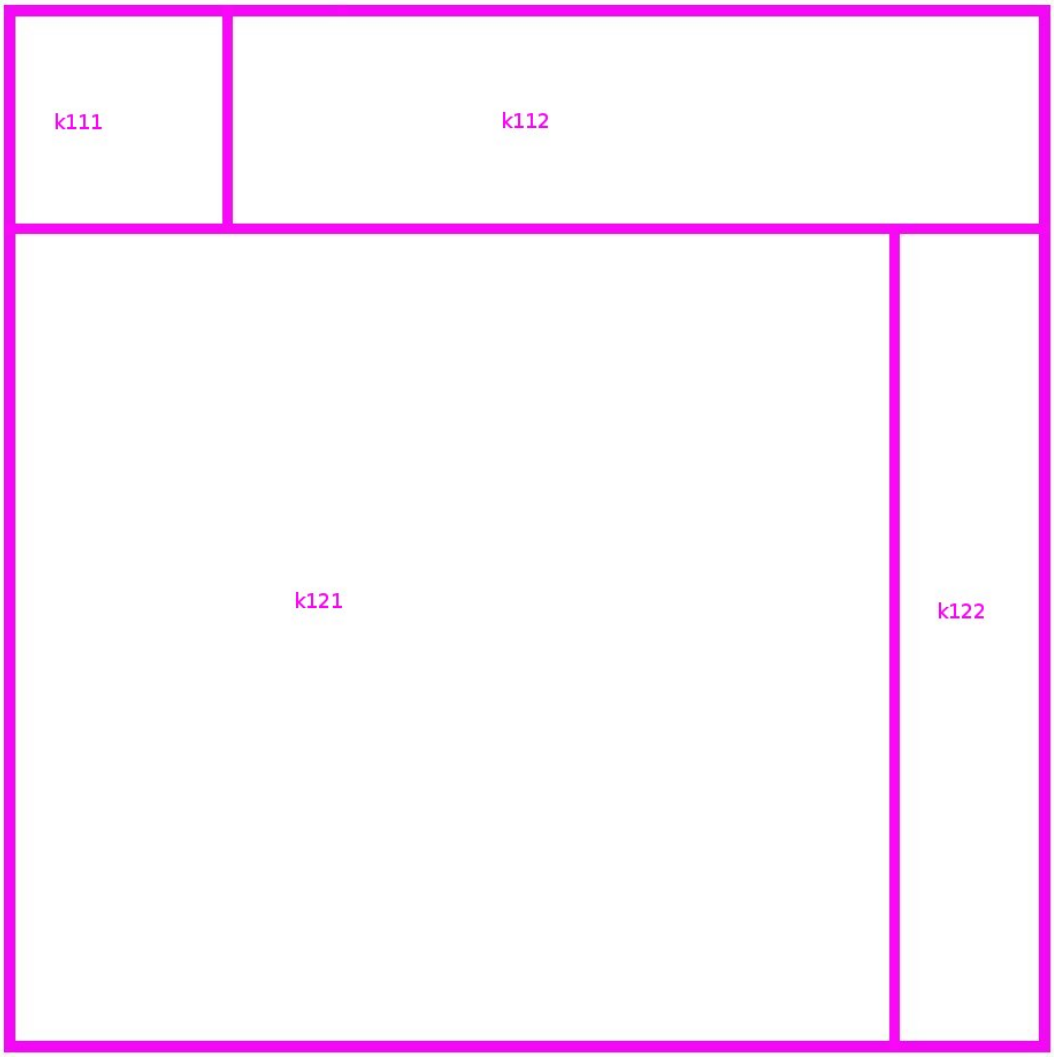
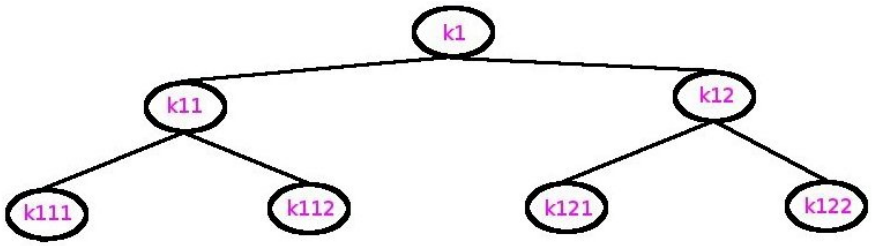


Figura 9: Cubo com a primeira subdivisão, no eixo y. a) Retângulos resultantes da subdivisão. b) *Kd-tree* com dois nós no segundo nível.

A definição de *octree* pode ser vista no artigo (12) e também no livro (3). Sua restrição em $\mathbb{R}^2$ é uma *quadtree* e seu funcionamento é análogo ao descrito para uma *quadtree*. A *octree* foi originalmente utilizada para obter uma partição do espaço, a partir de um sólido que contém o interior e o bordo



[a]

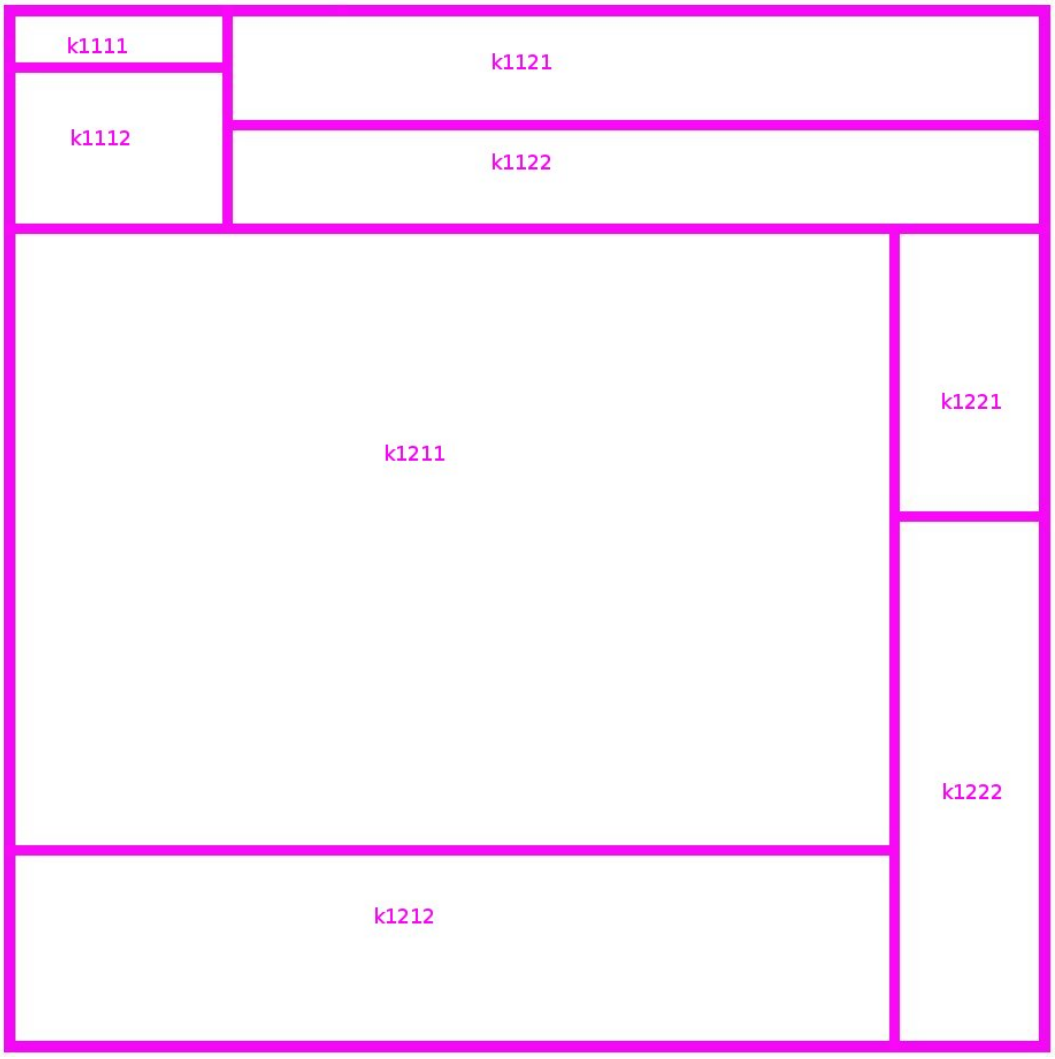


[b]

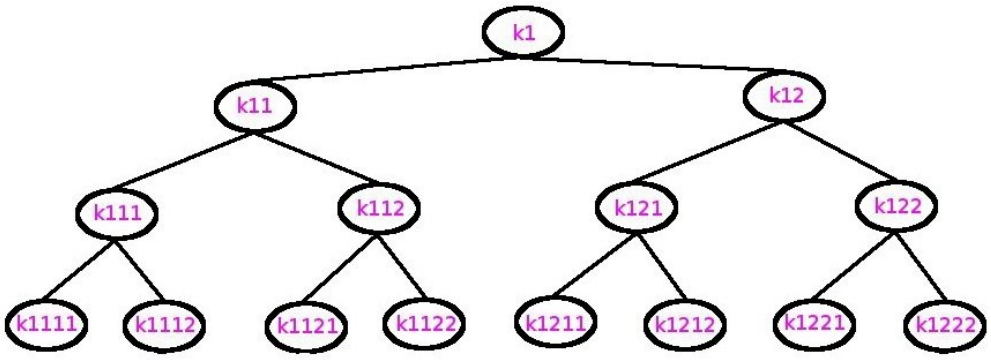
Figura 10: Cubo com a segunda subdivisão, no eixo x. a) Retângulos resultantes da subdivisão. b) *Kd-tree* com quatro nós no terceiro nível.

de uma superfície, como pode ser visto na figura 13.

Uma *octree* é uma estrutura de dados, como uma pilha ou uma fila, em que são guardadas informações resultantes da subdivisão de um cubo. Assim como a kd-tree, a *octree* é uma estrutura de dados do tipo árvore, pois é



[a]



[b]

Figura 11: Cubo com a terceira subdivisão, no eixo y. a) Retângulos resultantes da subdivisão. b) *Kd-tree* com oito nós no quarto nível.

um grafo acíclico e conexo. Entretanto na *octree*, para cada nó só há duas possibilidades: ou o nó não tem filhos, ou tem oito filhos. Ao contrario da kd-tree, que posiciona um plano de cada vez, na *octree* é escolhido um ponto no interior do cubo, onde são posicionados os planos que dividem este cubo e são

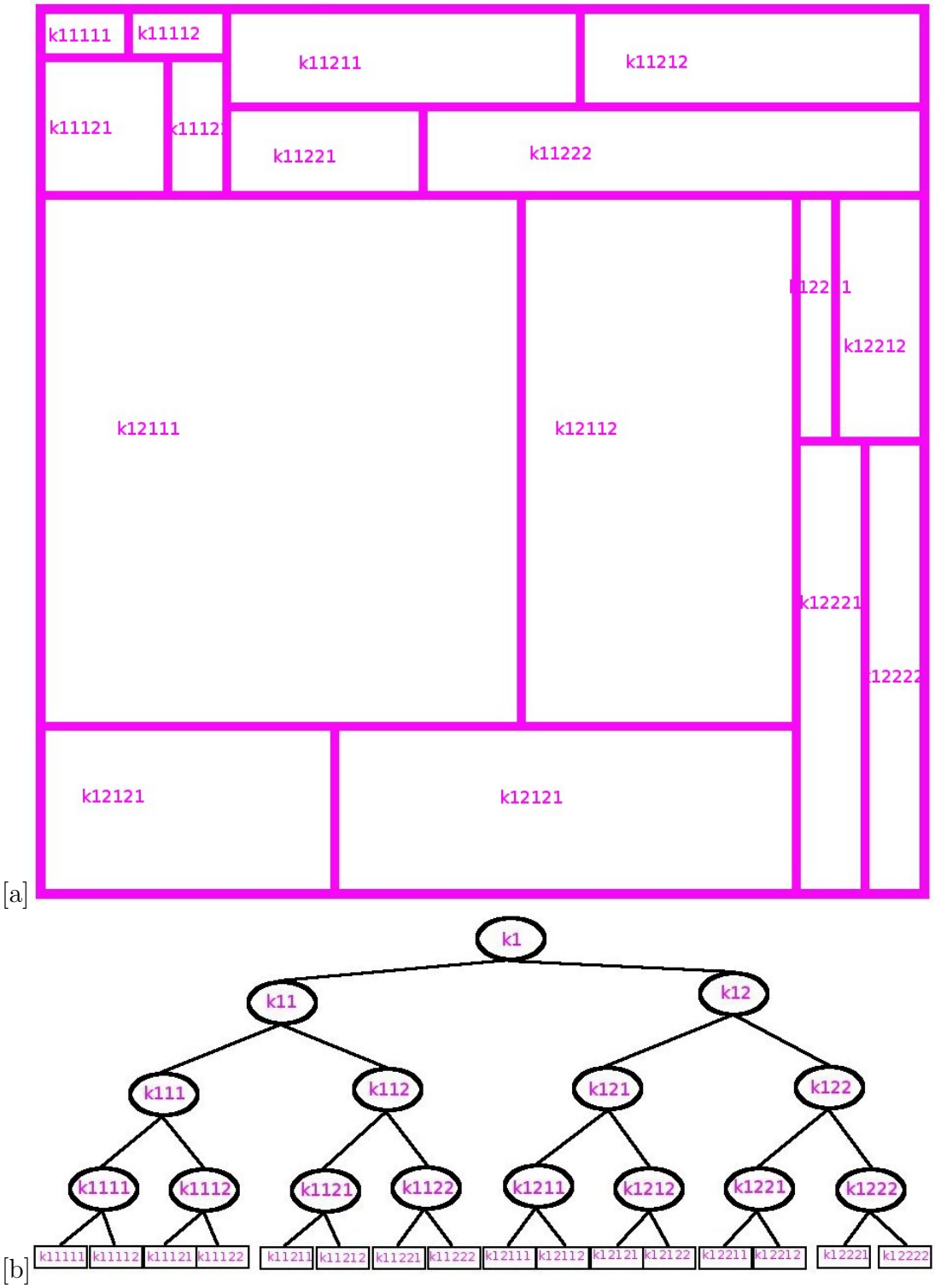


Figura 12: Cubo com a terceira subdivisão, no eixo x. a) Retângulos resultantes da subdivisão. b) *Kd-tree* com dezesseis nós no quinto nível.

paralelos aos planos xy, xz e yz.

Na definição original de *octree*, um cubo é subdividido sempre no seu centro, e em cada uma das oito regiões resultantes dessa subdivisão é testado se a superfície intersecta essa região. Caso intersecte, ela é novamente subdividida

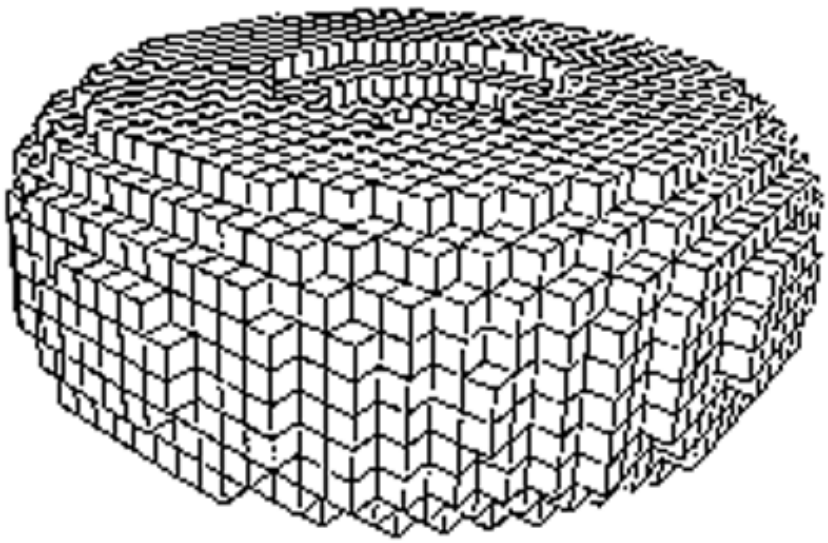
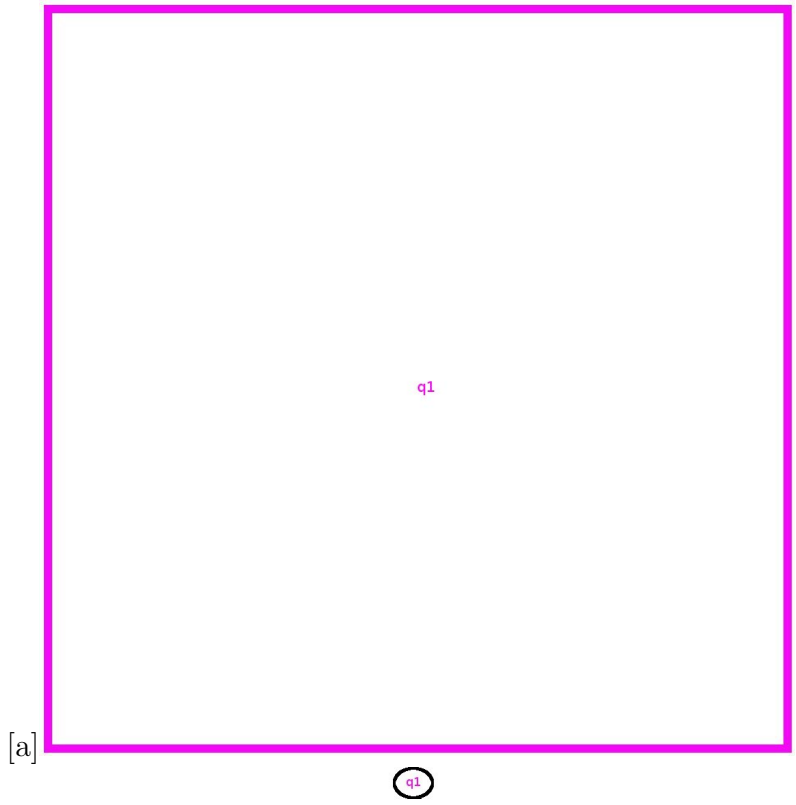


Figura 13: Geração de um sólido com seu interior e bordas, utilizando uma *octree*. Referência (3).

a partir do seu centro e o processo continua até um critério de parada, que pode ser o tamanho do cubo ou a profundidade da árvore.

Nas figuras 14 a 16 é mostrada a geração de uma *quadtree*, com a subdivisão do quadrado original, também relativa a uma curva hipotética, e a estrutura de dados associada a essa subdivisão. Neste exemplo, são feitas duas subdivisões, o que resulta em duas partições no eixo x e duas no eixo y. Na figura 15 b) é feita uma partição no eixo x e uma no eixo y. É possível ver que essas partições se encontram em um ponto do quadrado, que é o seu centro, pois é usada a definição original de *quadtree*. E ao contrário da kd-tree, que para fazer duas subdivisões nos eixos x e y requeria uma árvore com quatro níveis de profundidade, neste caso essas duas subdivisões nos eixos x e y requerem uma *quadtree* com somente dois níveis.



[a]

[b]

Figura 14: Cubo original. a) Cubo sem subdivisão. b) *Quadtree* somente com o nó raíz.

No caso de uma *octree*, para acessar um retângulo resultante da subdivisão nos eixos x, y e z, basta seguir para o nível inferior. Além disso, os algoritmos para testar se dois cubos se encontram em uma aresta ou face, são rápidos e simples.

Apesar de suas aplicações para poligonalização adaptativa, o fato de ser a subdivisão construída sempre a partir do centro do cubo pode gerar gastos desnecessários, gerando subdivisões em que os cubos resultantes não são os melhores para a poligonalização. Para a poligonalização de uma dada superfície implícita, a correspondente função implícita fornece informações em todo o espaço de poligonalização. Essa informação pode ser utilizada para posicionar melhor a grade, evitando restringir o ponto de subdivisão sempre ao centro, o qual pode não estar situado tão próximo da superfície.

As figuras 17 e 18 mostram a poligonalização adaptativa usando uma *octree*. Na figura 17 é utilizado o *Dual Contouring*, com a subdivisão da *octree* sempre no centro, e na figura 18 é utilizado o método proposto neste trabalho,

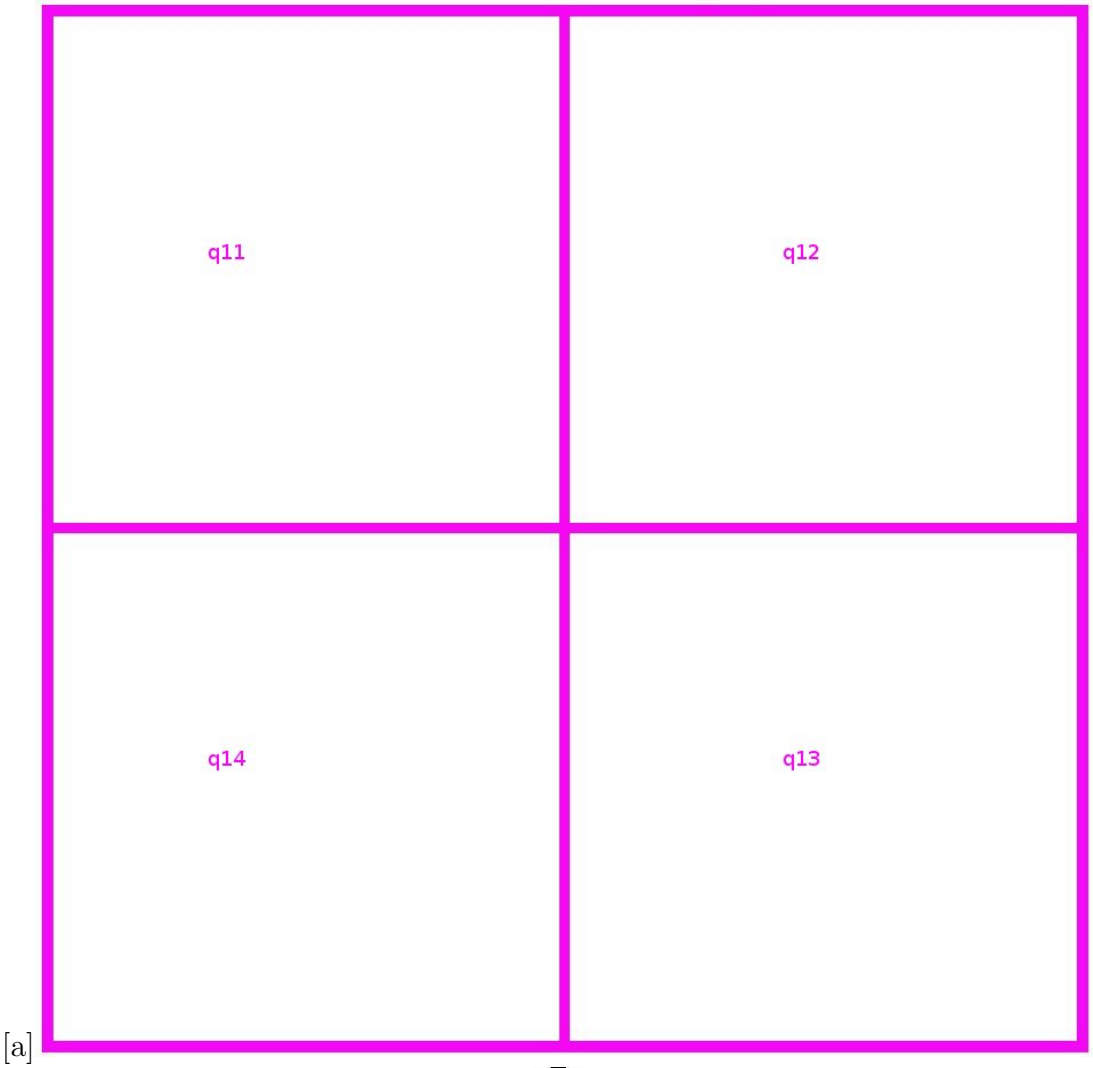


Figura 15: Cubo com a primeira subdivisão. a) Retângulos resultantes da subdivisão. b) *Quadtree* com quatro nós no segundo nível.

que posiciona o ponto de subdivisão da grade próximo à superfície

Na figura 17 é possível ver que na primeira subdivisão, figura 17 b), a grade está afastada da superfície. Na segunda subdivisão, figura 17 c), a grade resultante está distante e portanto não foram captados os detalhes da curva. Por isso a poligonalização resultante dessa grade é uma reta, enquanto a curva é modelada por uma senoide, figura 17 d).

Apesar de ser este método adaptativo, o posicionamento da subdivisão

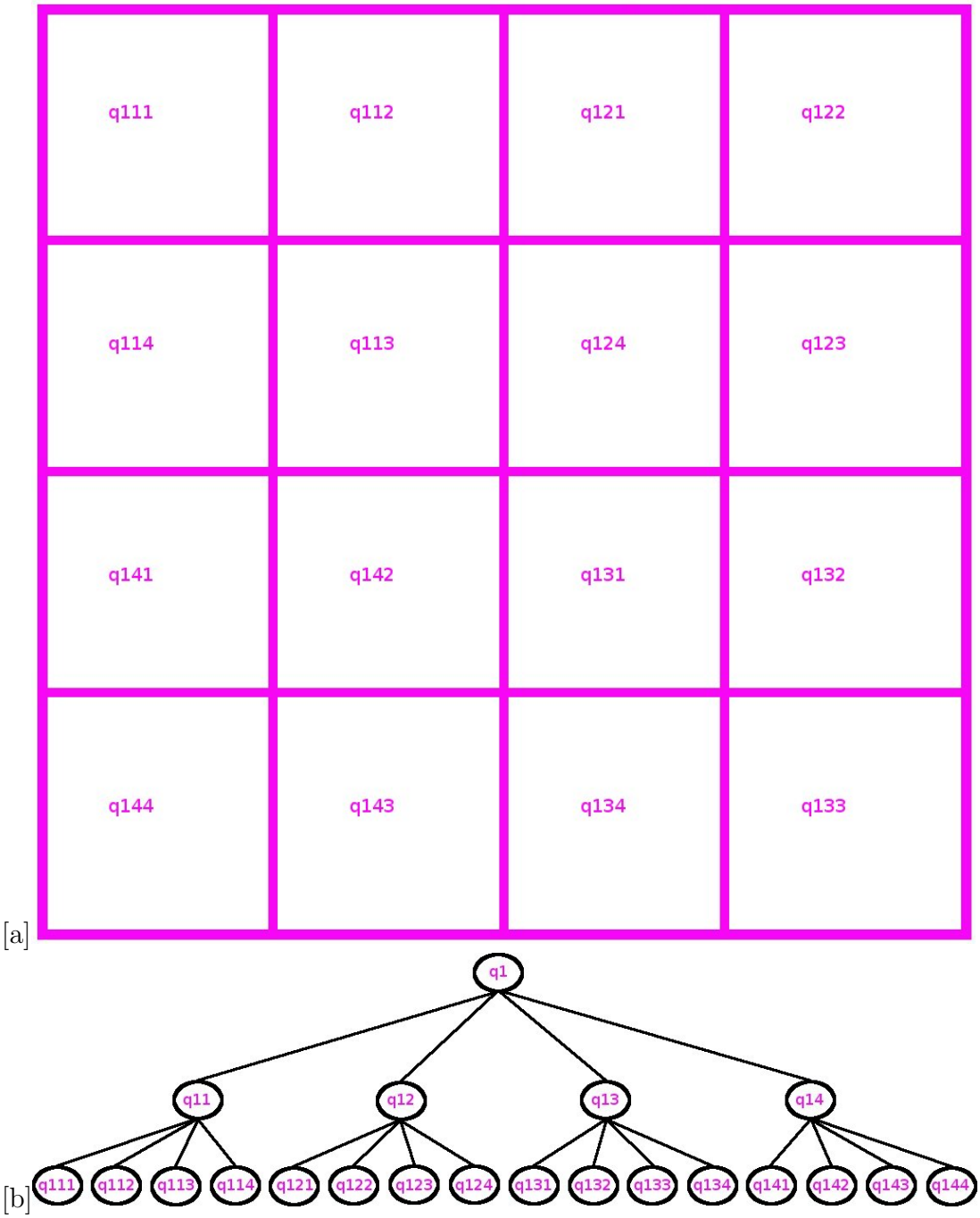


Figura 16: Cubo com a segunda subdivisão. a) Retângulos resultantes da subdivisão. b) *Quadtree* com dezesseis nós no terceiro nível.

no centro do cubo leva a subdivisões desnecessárias, e neste caso a uma amostragem que perde detalhes da superfície.

Na figura 18, é mostrada a poligonalização da mesma curva apresentada na figura 17, com os pontos de subdivisão da grade definidos a partir de dados associados à posição da curva. Em 18 b) a primeira subdivisão já está bem próxima da curva, e com a segunda subdivisão é possível ver que a grade em 18 c) está bem diferente da grade em 17 c), e intersecta a curva em vários

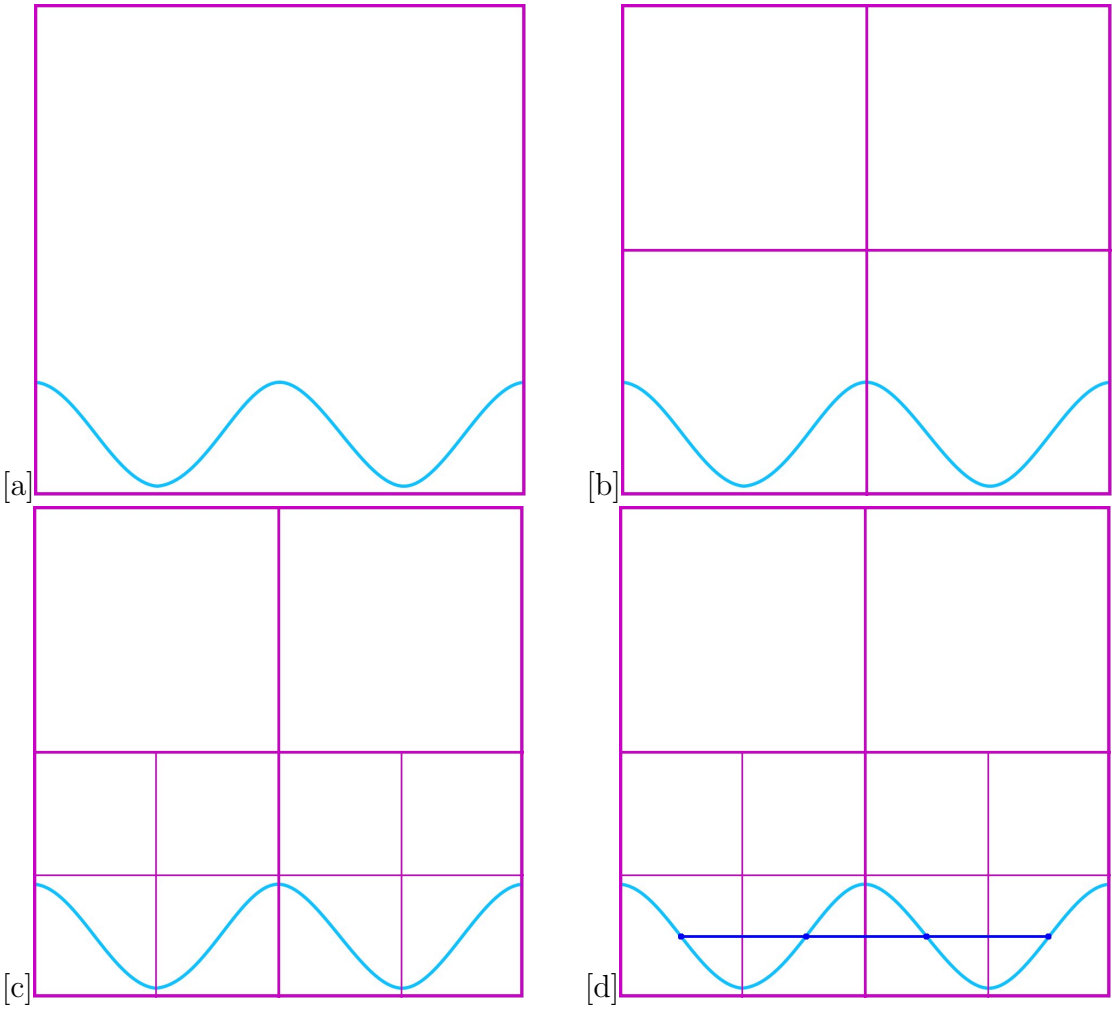


Figura 17: As subdivisões da região de poligonalização estão em lilás, e a curva está em azul. a) Quadrado de poligonalização. b) Quadrado com uma subdivisão. c) Quadrado com duas subdivisões. d) Em azul escuro está o resultado desta poligonalização, que é uma reta.

pontos. Como resultado, a poligonalização consegue captar mais detalhes da curva, como pode ser visto na figura 18 d), em que o resultado se aproxima bastante da curva original.

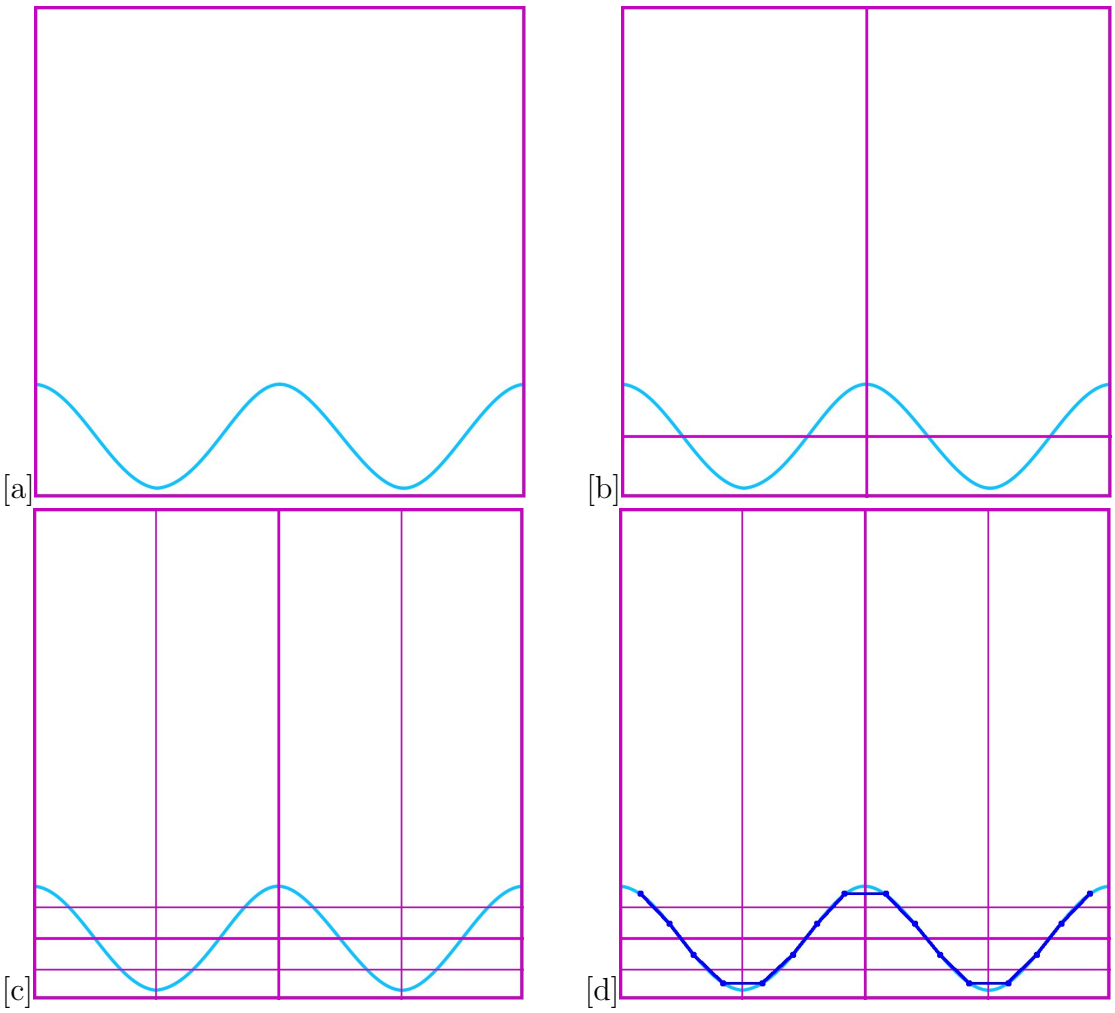


Figura 18: As subdivisões da região de poligonalização, representadas em lilás, e a curva original, representada em azul claro. a) Quadrado de poligonalização. b) Quadrado com uma subdivisão. c) Quadrado com duas subdivisões. d) A curva resultante dessa poligonalização, em azul escuro, que é bem próxima à curva original.

3
Métodos Anteriores

Neste capítulo são descritos métodos de poligonalização de superfícies implícitas relacionados a este trabalho. Iniciando com o *Marching Cubes*, um método que utiliza uma grade uniforme, passa pelos métodos *Extended Marching Cubes* e *SurfaceNets*, ambos métodos de poligonalização uniforme. Estes já contêm mudanças na poligonalização que possibilitam chegar a métodos adaptativos cuja estrutura de dados associada é uma *octree*. São métodos desse último tipo o *Dual Contouring*, mostrado no fim do capítulo, e o método desenvolvido nesta tese, a Grade Adaptativa.

3.1
Marching Cubes

Este é um método de poligonalização uniforme que apesar de suas limitações é muito utilizado, tanto pela sua simplicidade de compreensão quanto pela facilidade de implementação. Ele está descrito em (7), utiliza uma grade uniforme, que é gerada pela partição dos eixos x, y e z, e resulta em uma poligonalização da superfície por uma malha triangular.

No artigo original, a partição é percorrida duas vezes: na primeira os voxels que intersectam a superfície são identificados e marcados, e na segunda passagem são gerados os triângulos da poligonalização, nos voxels que foram marcados no primeiro passo. As implementações atuais percorrem a grade somente uma vez, onde para cada voxel que intersecta a superfície são gerados os triângulos da malha.

Em cada voxel que intersecta a superfície, são testados quais vértices estão dentro ou fora da superfície e sua configuração é comparada com uma tabela com 15 casos, pois segundo o artigo original, a menos de rotação, todas as possibilidades se restringiam a estes casos.

As configurações descritas no artigo original podem ser vistas na figura 19. É possível ver que para todas as configurações os vértices da malha estão nos bordos dos voxels, nas suas arestas, e não são colocados vértices da malha no interior do voxel.

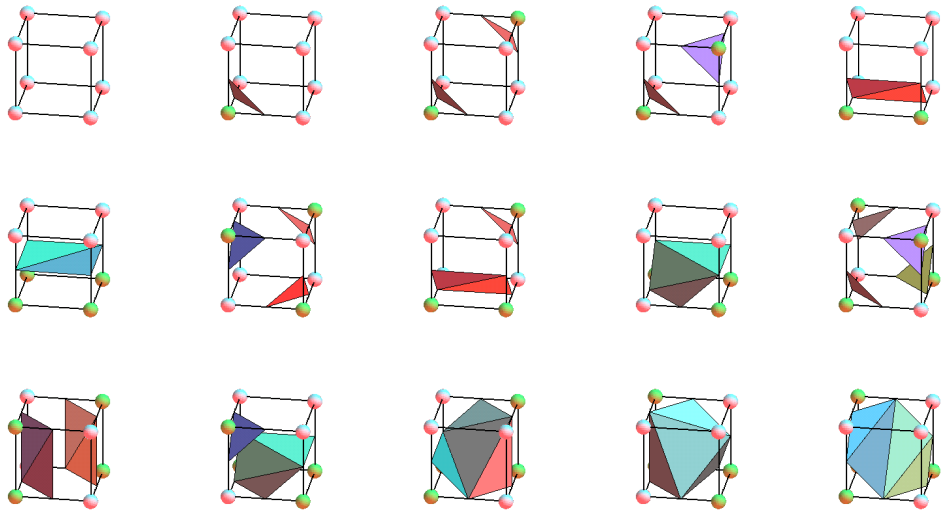


Figura 19: Configurações originais do *Marching Cubes*

Em trabalhos posteriores, foi mostrado que os 15 casos descritos no artigo original não cobrem todas as possibilidades. Este método possui casos de ambiguidades, em que dada uma configuração, é possível ter mais de uma poligonalização, como pode ser visto na figura 20. Algumas soluções foram propostas, como dividir o cubo em que há a possibilidade de ambiguidade em tetraedros, que não admitem ambiguidade, ou utilizar métodos numéricos, como o *Asymptotic Decider* (10) ou mais recentemente (13) que estende e implementa a solução dada por (14).

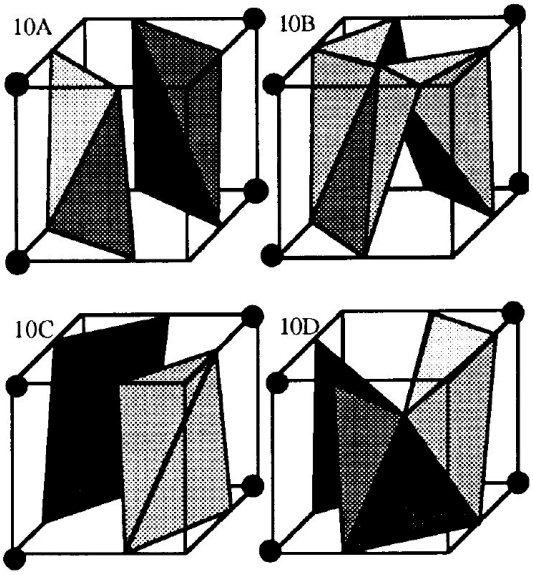
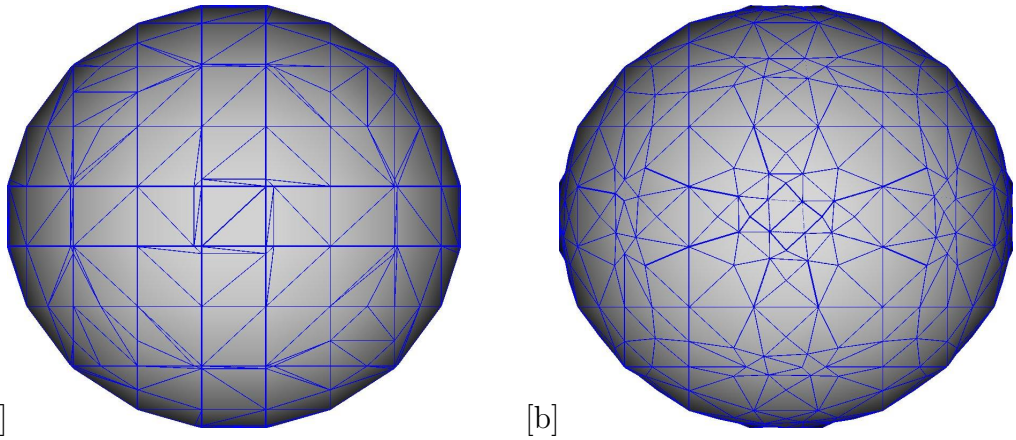


Figura 20: Ambiguidade no *Marching Cubes*. Neste caso uma mesma configuração pode gerar quatro poligonalizações distintas. Referência [Nielson 1991].

Outras modificações foram propostas para o *Marching Cubes*. Entre elas

temos (15), que propõem, para cada configuração, conectar os vértices de forma que os triângulos gerados sejam mais próximos de triângulos equiláteros, inclusive com a possível inserção de um vértice no interior do voxel. Em (16), figura 21, foi proposta uma suavização global na malha percorrendo os triângulos e, se necessário, modificando a posição dos seus vértices, para evitar triângulos com ângulos demasiadamente pequenos.



[a] [b]

Figura 21: Modificação da malha, proposta por (16).

Pelo fato de ser a grade uniforme, para que o *Marching Cubes* gere uma malha com mais detalhes da superfície é necessária a construção de uma grade muito refinada, mesmo em regiões que não possuem muitas informações, como regiões com menor curvatura. Outra limitação deste método é que, como os vértices da malha são posicionados na borda do voxel, se existirem detalhes no interior do voxel, como um bico ou quina, este não será detectado.

3.2 Extended Marching Cubes

O método *Extended Marching Cubes* foi apresentado em (8). Trata-se de uma modificação do *Marching Cubes* que utiliza uma grade uniforme e funciona em duas etapas. Para este método é necessária uma estrutura de dados topológica, como half-edge (17), para percorrer a malha.

Na primeira etapa, para cada voxel, são testados os vetores normais, dados pela função implícita, nos pontos da interseção da superfície com o voxel. Caso o ângulo entre estes vetores seja maior do que um limite dado, isto indica que dentro deste voxel a superfície tem um bico ou uma quina. Um ponto da malha no interior da grade é, então, posicionado para representar este artefato utilizando os vetores normais para posicionar este vértice, como pode ser visto na figura 22.

Deve-se tomar cuidado com o ponto inserido para que ele não ultrapasse os limites do voxel. Caso isto ocorra, é feita uma interpolação para posicionar este ponto dentro dos limites do voxel ao qual ele pertence.

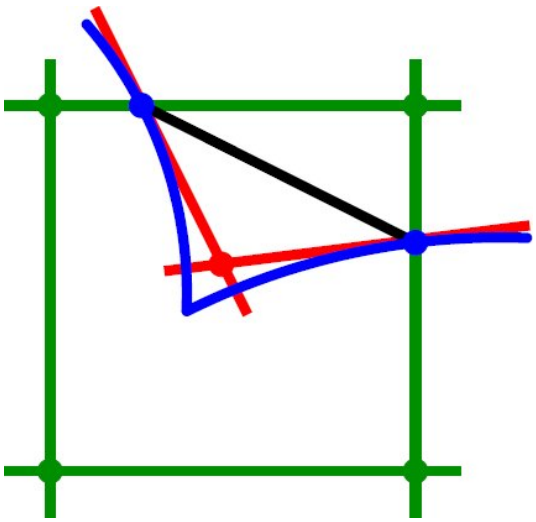


Figura 22: A grade esta em verde e a superfície pode ser vista em azul. Em preto está a poligonalização pelo *Marching Cubes*. O ponto em vermelho no interior da grade é a interseção das retas perpendiculares aos vetores normais. Figura (8).

A segunda etapa consiste em percorrer a malha gerada na primeira etapa, usando a estrutura de dados half-edge para testar se vértices que estão em triângulos que dividem uma aresta estão marcados. Caso estejam, uma mudança nas conexões da malha é feita para conectar estes pontos e gerar as arestas e vértices da superfície implícita, como pode ser visto na figura 23. A poligonalização com o *Marching Cubes*, figura 23 a), gera os bordos da superfície com baixa qualidade, já a poligonalização com o *Extended Marching Cubes*, figura 23 b), gera uma superfície com bordos que representam melhor a superfície original.

O *Extended Marching Cubes* consegue captar mais detalhes, como arestas e vértices, sem a necessidade de fazer mais subdivisões na grade, mas como posiciona vértices da malha na borda dos voxels, impossibilita fazer uma poligonalização adaptativa.

3.3 SurfaceNets

O método *SurfaceNets* é apresentado em (9) e apesar de ter sido publicado antes do *Extended Marching Cubes*, nesta tese ele é apresentado depois, pois a sua idéia principal leva à poligonalização adaptativa.

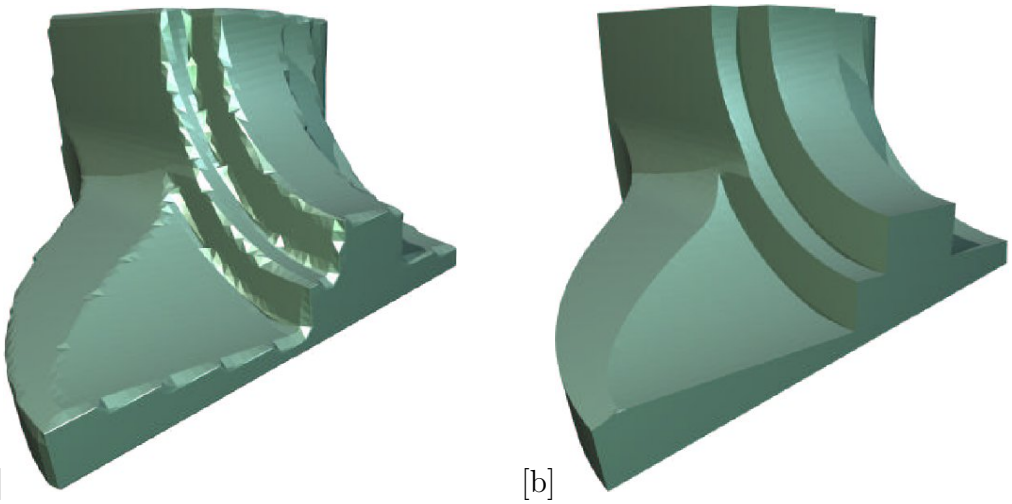


Figura 23: Poligonalização de uma superfície implícita. a) Poligonalização com o *Marching Cubes* b) Poligonalização com o *Extended Marching Cubes*. Figura (8).

SurfaceNets utiliza uma grade uniforme, e é um método interativo. Ao contrário dos métodos anteriores, os vértices da malha são posicionados no interior do voxel, inicialmente no seu centro. A conexão dos pontos é feita visitando todos os voxels sequencialmente, como no *Marching Cubes*.

Uma função de relaxamento é utilizada, iterativamente, para posicionar estes vértices sobre a superfície, de forma a mantê-los no interior do voxel original e garantir que a topologia da superfície não seja alterada. Um exemplo do funcionamento deste método pode ser visto na figura 24.

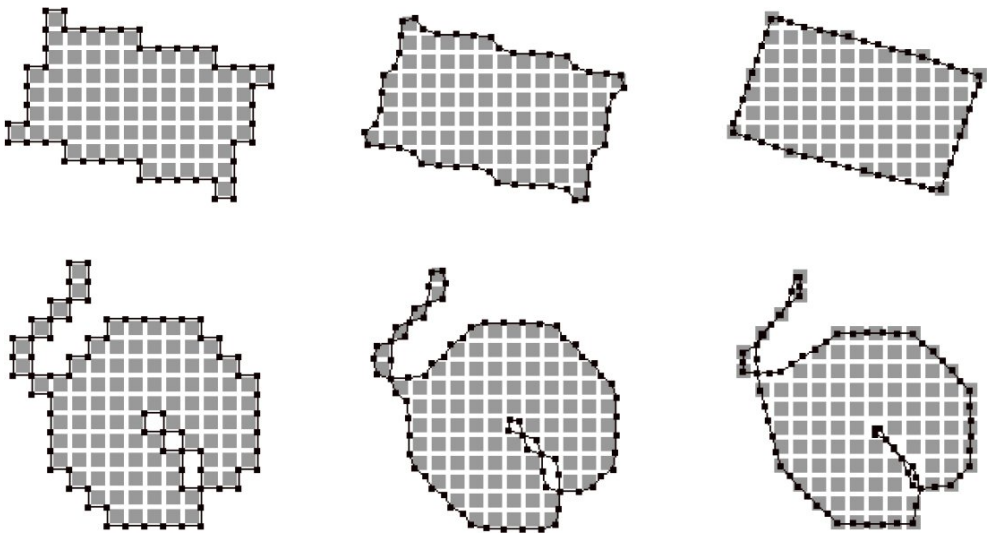


Figura 24: A primeira coluna mostra a superfície com os seus vértices posicionados no centro da grade. As duas colunas seguintes mostram o posicionamento dos vértices usando a função de relaxamento. Figura (9).

Modificações deste método são utilizadas para a geração de malhas para

o Método de Elementos Finitos, como em (18).

Este método tem as condições necessárias para fazer a poligonalização adaptativa, pois não utiliza pontos no bordo do voxel. Por outro lado, ele não explora essa capacidade e funciona com uma partição uniforme, da mesma forma que o *Marching Cubes*.

3.4 Dual Contouring

O *Dual Contouring* é apresentado em (4), e é descrito com mais detalhes no relatório técnico (19). Este método é adaptativo e usa uma *octree* como estrutura de dados associada à partição do espaço. Para a geração dos polígonos percorre-se a *octree* de forma recursiva.

Este método possui duas etapas: na primeira é feita a geração da *octree*, posicionando os vértices da malha no interior das folhas que intersectam a superfície; a segunda etapa consiste em percorrer esta *octree* para conectar os vértices da malha, com a poligonalização sendo gerada por triângulos e/ou retângulos.

Na primeira etapa, em que é gerada a subdivisão do espaço, esta subdivisão é feita até ser alcançada uma profundidade máxima definida pelo usuário, ou até algum outro critério de parada, como a curvatura da superfície, por exemplo. Por ser um método adaptativo, a *octree* vai ser gerada de acordo com características da superfície, portanto não é necessário gerar uma *octree* completa. Em cada folha da *octree* que intersecta a superfície, um vértice da malha é posicionado utilizando os valores da função implícita nos vértices do voxel referente a esta folha.

Os vértices são conectados analisando as Arestas Mínimas, definidas como as arestas em que três ou quatro cubos, que correspondem a folhas na *octree*, se encontram. Como a subdivisão sempre é feita no centro do cubo, a Aresta Mínima pode ser definida como a aresta pertencente ao cubo que corresponde à folha com maior profundidade na *octree*, como pode ser visto na figura 25. Nestas arestas os sinais dos seus vértices são analisados, e caso possuam sinais distintos isto indica que a superfície passa por esta aresta. Neste caso, os pontos pertencentes aos cubos que se conectam nesta aresta são conectados, formando um polígono da superfície aproximada. Como nesta tese é feita uma modificação da definição original de Aresta Mínima, são mostrados com mais detalhes o seu funcionamento no próximo capítulo.

Em conjunto com o *Dual Contouring*, é apresentado um método de simplificação de superfícies para manter a topologia da superfície original.

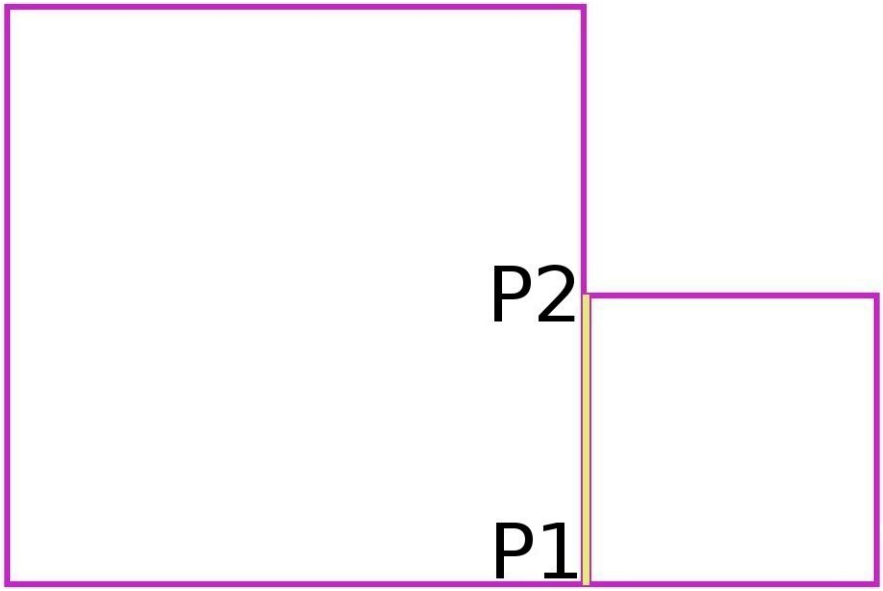


Figura 25: A Aresta Mínima é dada pelos vértices P1 e P2, que são extremos do cubo à direita. Na *octree* ele corresponde a uma folha situada a um nível abaixo do da folha correspondente ao cubo à esquerda.

Este método de simplificação foi estendido para o caso em que a superfície é não-compacta por (20), e para o caso de um cilindro em que toda uma seção (circunferência) degenera em uma curva simples (21).

Várias outras modificações foram propostas para o *Dual Contouring*, entre elas a descrita em (22), que propõe uma forma de colocar mais de um ponto por folha, conseguindo desta forma, separar componentes distintas da superfície que estão em um mesmo voxel. Outra modificação é proposta por (23) que propõem um método de simplificação que mantém a topologia para superfícies poligonalizadas com múltiplos vértices da malha em cada folha.

A *octree* utilizada é a clássica, onde o ponto da subdivisão de um cubo é posicionado sempre no seu centro. Apesar de ser um método adaptativo, dependendo da posição da superfície, podem ser necessárias várias subdivisões para captar detalhes da superfície, pois em um dado estágio a grade pode ainda estar distante da superfície.

No capítulo seguinte é apresentado o método proposto nesta tese, em que a grade converge para a superfície, sendo possível obter resultados mais precisos com menos subdivisões da grade.

4
Grade Adaptativa: Geração da Octree

O método apresentado nesta tese, a Grade Adaptativa, consiste em dois passos, para cada superfície gerada: definir a *octree*, com as suas subdivisões, e percorrer esta *octree* para conectar os vértices da malha.

O objetivo do método é tornar a grade adaptativa no sentido de que ela se aproxima da superfície graças à escolha dos pontos da subdivisão. Para isto, a cada passo da subdivisão é escolhido um ponto próximo da superfície, e não o ponto central do cubo, como é tradicional ao utilizar uma *octree* em uma poligonalização adaptativa. A etapa de geração da *octree* é descrita neste capítulo.

Após a geração da *octree* e o posicionamento dos vértices da malha nas folhas que intersectam a superfície, ela é percorrida para conectar esses vértices. Essa conexão é feita com o emprego das Arestas Mínimas, que são resultantes do encontro de três ou quatro cubos relacionados às folhas da *octree* em torno de uma aresta. Esta etapa de conexão dos pontos da malha é descrita com mais detalhes no capítulo 5.

Nas duas subseções seguintes são revistas a definição clássica da *octree* (3) e de dois métodos para busca de raízes (Bissecção e Posição Falsa). Os conceitos utilizados nesses dois métodos levam à definição do ponto de subdivisão do cubo proposta neste trabalho.

4.1
Definição Clássica de Octree

A *octree* é uma estrutura de dados, geometricamente ela é associada em $\mathbb{R}^3$ à subdivisão de um cubo do espaço, onde cada nó corresponde a um cubo da subdivisão. A subdivisão é construída escolhendo um ponto no interior do cubo a ser subdividido. Por esse ponto passam os três planos paralelos aos planos coordenados que dividem o cubo em consideração.

O cubo inicial da poligonalização é associado ao primeiro nó da *octree*, o nó raiz. Cada cubo é subdividido em oito cubos, o que faz com que o nó

correspondente a este cubo na *octree* tenha oito filhos, que estão associados aos oito cubos resultantes da subdivisão.

Na definição clássica da *octree*, a subdivisão é feita sempre no centro do cubo, como pode ser visto na sua definição original (3) e na sua aplicação à poligonalização de superfícies implícitas (4). A figura 26 a) mostra um cubo subdividido, gerando oito novos cubos. Esse cubo é o equivalente geométrico a uma *octree* com raíz e oito filhos, como mostrado na figura 26 b).

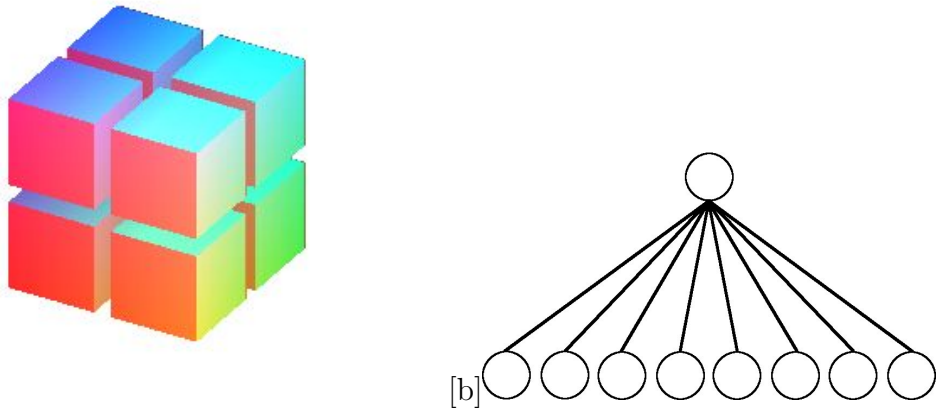


Figura 26: a) Cubo com uma subdivisão. b) *Octree* com os oito nós resultantes da subdivisão.

Em $\mathbb{R}^2$ a estrutura de dados correspondente à *octree* é a *quadtree*, que, por ser bidimensional, é mais fácil de visualizar em um texto impresso. Para uma *octree*, cada nó ou não tem nenhum filho ou tem oito filhos, enquanto em uma *quadtree*, que está associada a um retângulo, cada nó ou não tem nenhum filho ou tem quatro filhos, que estão associados aos quatro retângulos resultantes da subdivisão, como pode ser visto na figura 27.

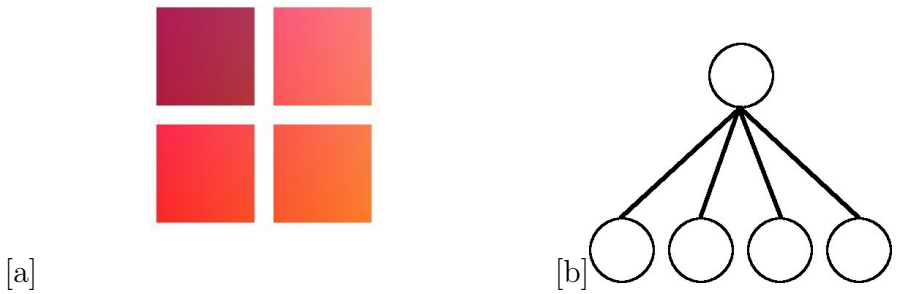


Figura 27: a) Quadrado com uma subdivisão. b) *Quadtree* com os quatro nós resultantes da subdivisão.

A *octree*, pela sua estrutura, pode ser geometricamente associada a poligonalização adaptativa. Regiões que requerem mais detalhes, como regiões

com alta curvatura, podem ter mais níveis de subdivisão e regiões com menos detalhes, ou que não interseciona a superfície, podem ter menos níveis de subdivisão.

Uma *octree* gerada apenas com os pontos da subdivisão no centro do cubo a ser particionado pode resultar em um número excessivo de níveis de subdivisão, apesar de estarem os vértices do cubo distantes da região onde passa a superfície. Esse fato é ilustrado na figura 28, que mostra o perfil de uma função modelada por um cosseno e que está concentrada na região inferior do retângulo. A figura 28 a) mostra uma *quadtree* gerada com os pontos de subdivisão no centro do cubo e a figura 28 b) mostra uma *quadtree* gerada com a subdivisão proposta nesta tese, em que o valor da função é levado em conta para posicionar o ponto de subdivisão.

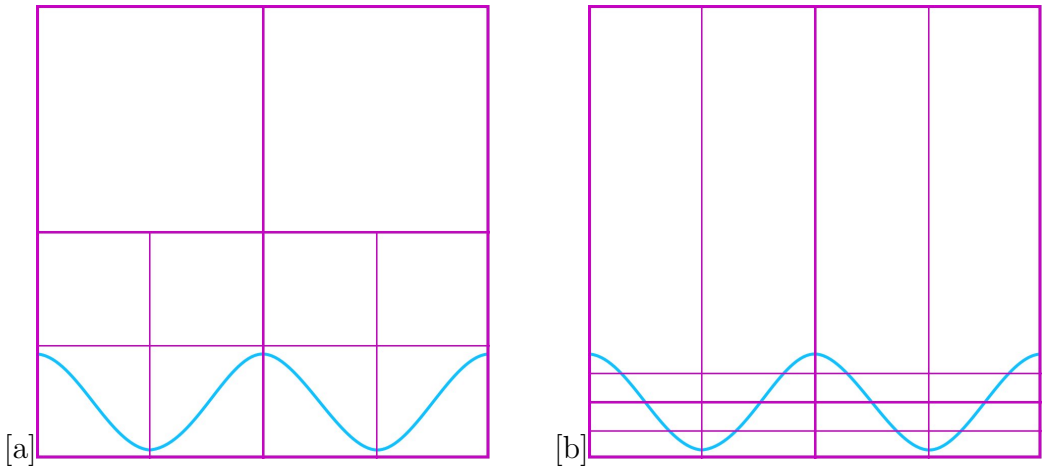


Figura 28: a) *Quadtree* gerada com a subdivisão no centro. b) *Quadtree* gerada com a subdivisão mais próxima à superfície.

Na figura 28 b) é possível ver que a grade esta mais próxima à curva fazendo, nesse caso, com que cada retângulo intersecte a superfície, gerando assim uma malha mais refinada com o mesmo número de subdivisões da *quadtree* mostrada em 28 a).

4.2
Encontrando Raízes de Função

Nesta seção são mostrados dois métodos para procurar raízes de uma função. Eles estão associados à subdivisão da *octree* sempre no seu centro, método da Bissecção, e à subdivisão proposta neste trabalho, método da Falsa Posição, em que a função implícita que define a superfície é utilizada para posicionar o ponto de subdivisão do cubo. Neste trabalho, sem perda

de generalidade, a superfície é definida por $f(x, y, z) = 0$, e portanto são procuradas as raízes da função $f(x, y, z)$.

4.2.1 Método da Bissecção

Neste método, inicialmente é dado um intervalo inicial $[a, b]$ em que os valores da função têm sinais opostos nos extremos deste intervalo, $f(a) * f(b) < 0$. Este é um método iterativo em que a cada iteração é escolhido um ponto c no centro do intervalo e é calculado o produto dos valores da função neste ponto c com os extremos do intervalo. Nesta seção, a cada iteração o ponto c é chamado de Pi, onde i é o índice da iteração.

Se $f(a) * f(c) < 0$, então redefine-se $b = c$, caso contrário é redefine-se $a = c$ e este passa a ser o novo intervalo. O algoritmo continua até um critério de parada, que pode ser o número de iterações, o comprimento do intervalo resultante ou o valor do módulo da função no ponto do centro do intervalo.

Na figura 29 é mostrada uma sequência de iterações para o cálculo da raíz da função $-x^3 + 9x^2 - 6x - 26$ no intervalo $[0, 6]$, cuja raiz é $x_r = 2.518$. Neste caso é possível ver que a convergência é muito lenta, pois mesmo tendo a informação do valor da função em todos os pontos do domínio, esta informação não é utilizada para o cálculo da raíz, pois a subdivisão é feita sempre no centro.

A figura 29 a) mostra o gráfico da função no intervalo $[0, 6]$; o resultado da primeira iteração é mostrado na figura 29 b), nela o valor de $P1$ é 3 e $f(P1) = 10$. A distância entre o valor encontrado na primeira iteração e a raiz é $|P1–x_r| = 0.482$.

Como o método da Bissecção não leva em conta, para a definição do novo subintervalo, os valores da função considerada, na segunda iteração, que é mostrada na figura 29 c), o valor de $P2$ é 1.5 e $f(P2) = -18.125$, fazendo $|P2–x_r| = 1.018$, que é maior do que a distância entre a raiz e o valor de $P1$, obtido na primeira iteração. Na terceira iteração, que pode ser vista na figura 29 d), o valor de $P3$ é 2.250, com $f(P3) = -5.328$. Portanto, ao final de 3 iterações a distância entre a raiz e o valor de $P3$ é $|P3–x_r| = 0.268$.

A forma como o ponto da subdivisão de uma *octree* é escolhido, na sua definição clássica, é similar a uma iteração do método da Bissecção, pois a subdivisão é feita sempre no centro do cubo, não sendo levados em conta os valores da função implícita.

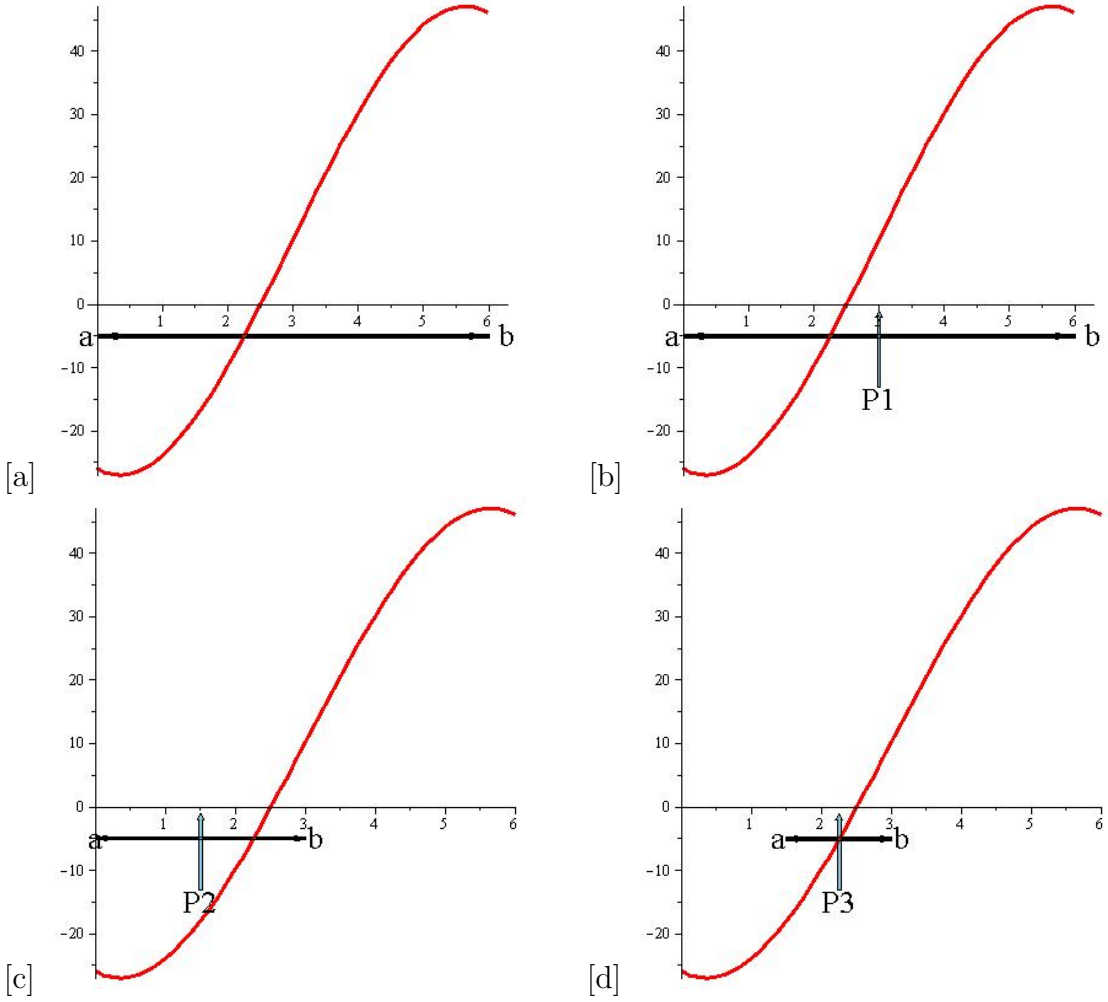


Figura 29: Método da Bissecção com 3 iterações. a) Curva original. b) Primeira iteração. c) Segunda iteração. d) Terceira iteração.

4.2.2 Método da Posição Falsa

Esse método, como o da Bisseção, inicia-se com um intervalo $[a, b]$, em cujos extremos a função assume valores com sinais opostos. Esse intervalo é subdividido para achar uma raiz da função, mas o ponto escolhido para fazer a divisão do intervalo não é o ponto médio deste intervalo, e sim um ponto calculado levando em conta o valor da função nos extremos do intervalo e utilizando a interpolação linear.

A escolha do ponto c para a subdivisão é dada por $c = \frac{a*f(b)-b*f(a)}{f(b)-f(a)}$. Da mesma forma que no método da Bissecção, é feito o teste $f(a) * f(c)$; se o resultado for negativo, faz-se $b = c$, caso contrário, $a = c$. O processo continua até um critério de parada, como para Bissecção. A cada iteração, o ponto c é chamado de Pi, sendo i o índice da iteração.

Nesse método deve-se observar na escolha do ponto c que o valor da

função é levado em conta. Para exemplificar e comparar a sua convergência com o método da Bissecção, na figura 30 é usada a mesma função $-x^3+9x^2-6x-26$ e o mesmo intervalo $[0, 6]$.

A figura 30 a) mostra a curva original no intervalo $[0, 6]$. Em 30 b) é mostrada a primeira iteração deste método, que resulta em $P1 = 2.167$ e $f(P1) = -6.921$. Com isso a distância entre o valor encontrado na primeira iteração e a raiz é $|P1–x_r| = 0.351$. Na segunda iteração, mostrada na figura 30 c), tem-se $P2 = 2.668$ e $f(P2) = 3.065$, que resulta em $|P2–x_r| = 0.150$. Finalmente na figura 30 d) é mostrada a terceira iteração, que tem $P3 = 2.514$ e $f(P3) = -0.088$, resultando em $|P3–x_r| = 0.004$.

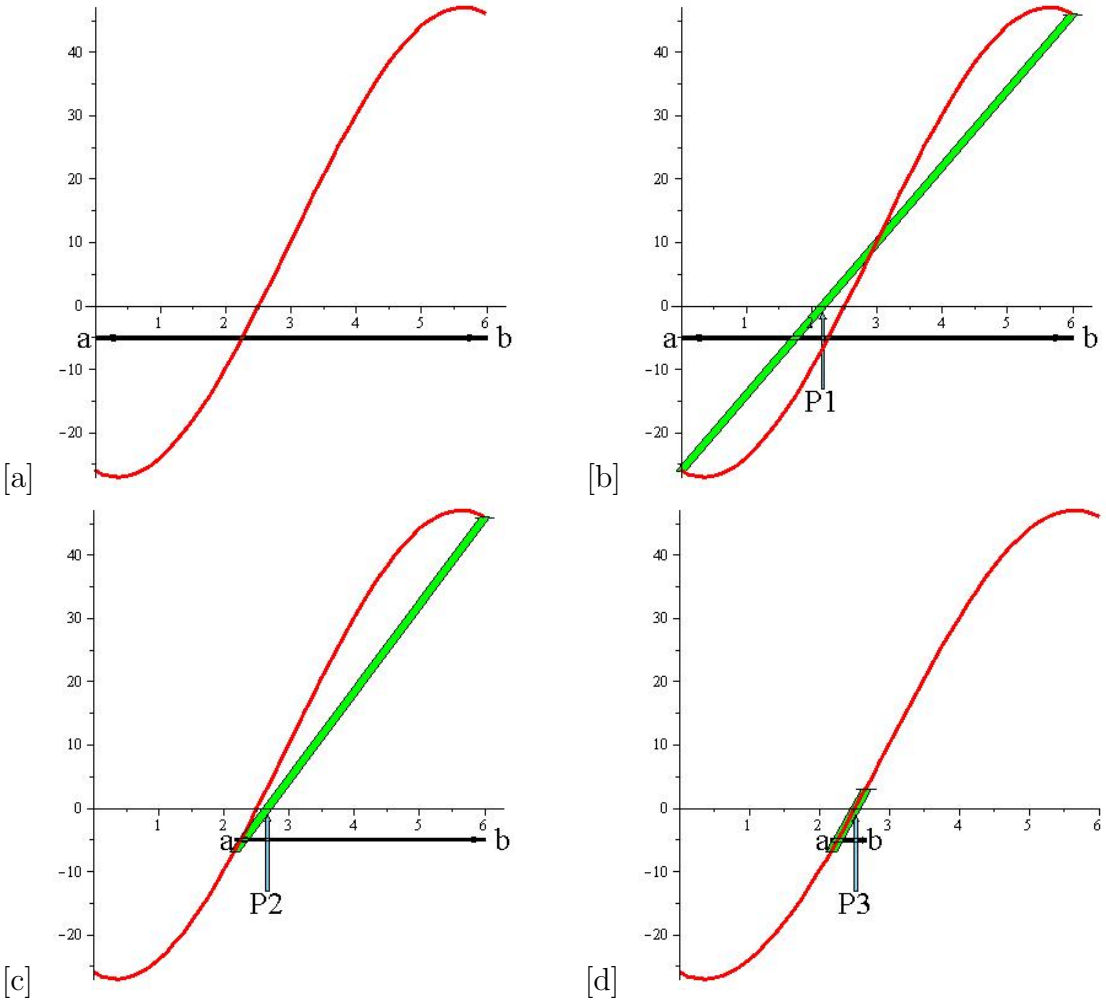


Figura 30: Método da Falsa Posição com 3 iterações. a) Curva original. b) Primeira iteração. c) Segunda iteração. d) Terceira iteração.

Portanto, ao final de três iterações do método da Falsa Posição, a diferença entre os valores aproximado e exato da raiz é 0.004, enquanto o mesmo número de iterações utilizando o método da Bissecção resulta em uma diferença de 0.268.

Neste trabalho é proposto um método de subdivisão da *octree* que leva em conta os valores da função: o ponto de subdivisão da *octree* não tem que ser no centro do cubo, como na definição clássica, mas pode ser obtido com uma variação do Método da Falsa Posição, com uma ou várias iterações.

4.3 Gerando o Ponto da Subdivisão da Octree

A subdivisão da *octree* é efetuada usando valores da função implícita de forma a determinar um ponto do cubo que esteja mais próximo da superfície. Num cubo a ser subdividido há duas possibilidades para os valores da função calculados nos vértices:

- Todos os vértices têm o mesmo sinal.
- Os vértices têm sinais distintos.

Nestes dois casos, este cubo pode conter uma parte da superfície, como pode ser visto nas figuras 31 e 32. Na figura 31 a) é mostrada uma circunferência totalmente contida no retângulo, o que faz com que os vértices tenham todos o mesmo sinal, pois estão fora da circunferência. Nesse caso, a subdivisão é feita no seu centro, como pode ser visto na figura 31 b).

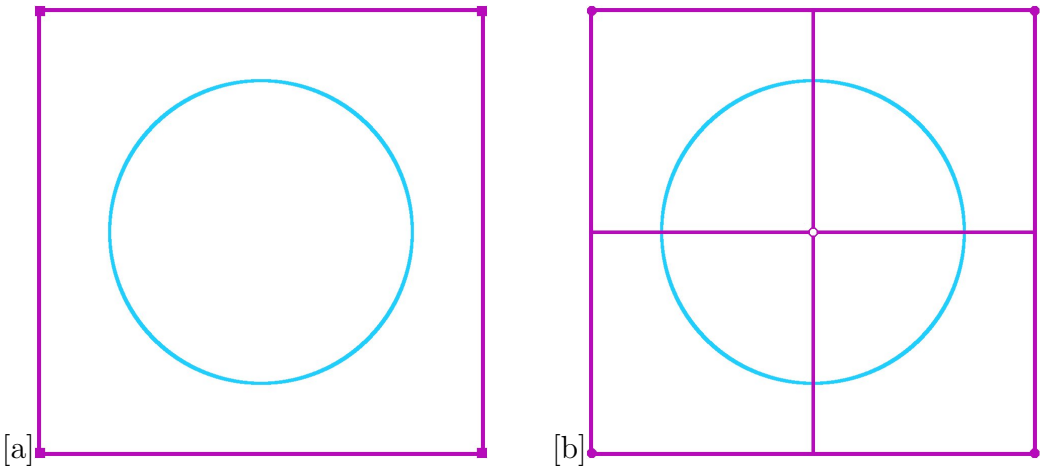


Figura 31: a) Circunferência totalmente contida no quadrado. b) Subdivisão feita no seu centro.

A figura 32 a) mostra a porção de uma circunferência parcialmente contida no retângulo, mas que engloba apenas um dos seus vértices. Neste caso um dos vértices tem sinal distinto dos demais. A informação da função implícita nos vértices é utilizada para posicionar o ponto de subdivisão, como pode ser visto na figura 32 b).

Além do valor da função implícita nos vértices do cubo, o gradiente também pode ser usado para posicionar o ponto de subdivisão, como pode

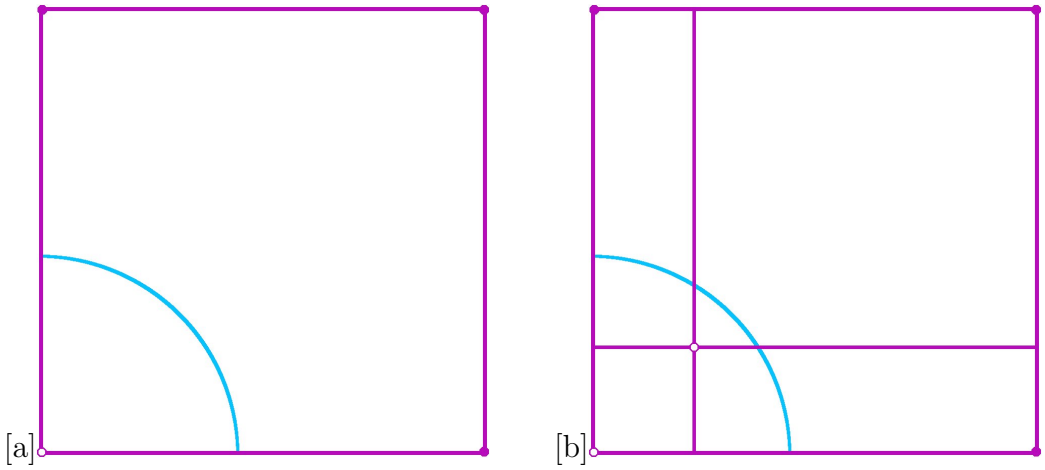


Figura 32: a) Circunferência englobando um dos vértices do quadrado. b) Subdivisão usando as informações dos vértices do quadrado e, neste caso, fora do seu centro.

ser visto na figura 33. Em a) é mostrado o ponto resultante da interseção dos vetores tangentes à curva, em verde são mostradas as retas tangentes e em magenta o ponto resultante da interseção destas.

Além do posicionamento inicial do ponto de subdivisão o vetor gradiente também pode ser utilizado de forma iterativa para reposicionar esse ponto mais próximo à superfície, como pode ser visto na figura 33 b).

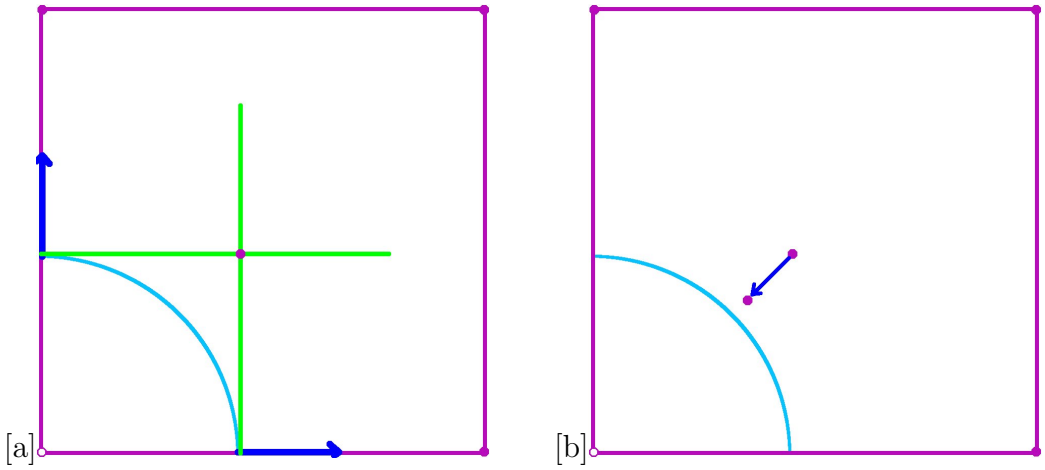


Figura 33: a) Posicionamento inicial do ponto de subdivisão utilizando o vetor gradiente. b) Reposicionamento do ponto de subdivisão utilizando o gradiente para posicioná-lo mais próximo à curva.

Quando o ponto de subdivisão do cubo é escolhido fora do centro é necessário tomar cuidado para que não ocorram situações que levem a degeneração da subdivisão. São eles:

– Folga dos lados da subdivisão com relação ao cubo original

– Convergência somente por um dos lados da superfície

– Posicionamento do ponto de subdivisão fora da superfície

Como estes são critérios importantes para a geração da grade, cada um deles será discutido com mais detalhes a seguir.

4.3.1
Folga dos lados da subdivisão com relação ao cubo original

Ao escolher o ponto da subdivisão é necessário levar em conta que o retângulo resultante desta subdivisão não pode ser degenerado, com um dos lados com comprimento muito próximo de zero, e portanto com o ponto de subdivisão muito próximo de uma das arestas, como mostra a figura 34 a). Para evitar esta degeneração é definida uma folga em torno dos lados do quadrado, fazendo com que o ponto de subdivisão só possa ser posicionado em uma região definida no seu interior, como mostra a região em branco na figura 34 b).

Este valor da folga pode ser definido como uma porcentagem do valor dos lados do retângulo. Ao defini-lo como 50%, a única região válida é o centro do cubo e o algoritmo de subdivisão funciona de forma análoga à definição clássica de geração da *octree*.

O reposicionamento do ponto de subdivisão pode ser feito utilizando o vetor normal, ou pode ser definido como o ponto válido mais próximo com relação a cada um dos eixos x, y e z.

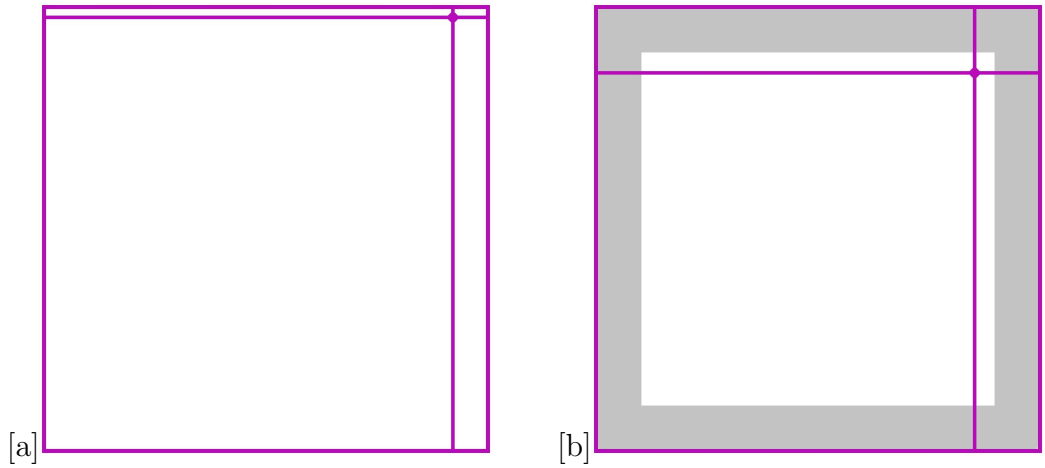


Figura 34: a) Ponto de subdivisão muito próximo a uma das arestas do cubo. b) Folga da subdivisão em cinza e região válida em branco.

4.3.2
Convergência somente por um dos lados da superfície

Neste caso é necessário ter cuidado para não deixar a subdivisão ser feita sempre em uma mesma região definida pela superfície - dentro ou fora. O objetivo é evitar uma convergência muito boa por um lado enquanto o outro lado tem um espaço grande sem pontos da subdivisão e, consequentemente, o posicionamento do ponto de subdivisão gere um erro elevado e desnecessário.

A figura 35 a) mostra a subdivisão feita sempre no interior da superfície e a figura 35 b) mostra a subdivisão feita alternadamente nas regiões interior e exterior à superfície. Na figura 35 b) é possível ver que o retângulo R1 tem arestas menores e está mais próximo à superfície do que o seu equivalente na figura 35 a), possibilitando uma poligonalização mais precisa da superfície. Além disso, com a poligonalização sendo feita alternadamente no interior e no exterior da superfície são gerados mais retângulos que intersectam a superfície. Na figura 35 a) sete retângulos intersectam a superfície, enquanto na figura 35 b) são nove retângulos.

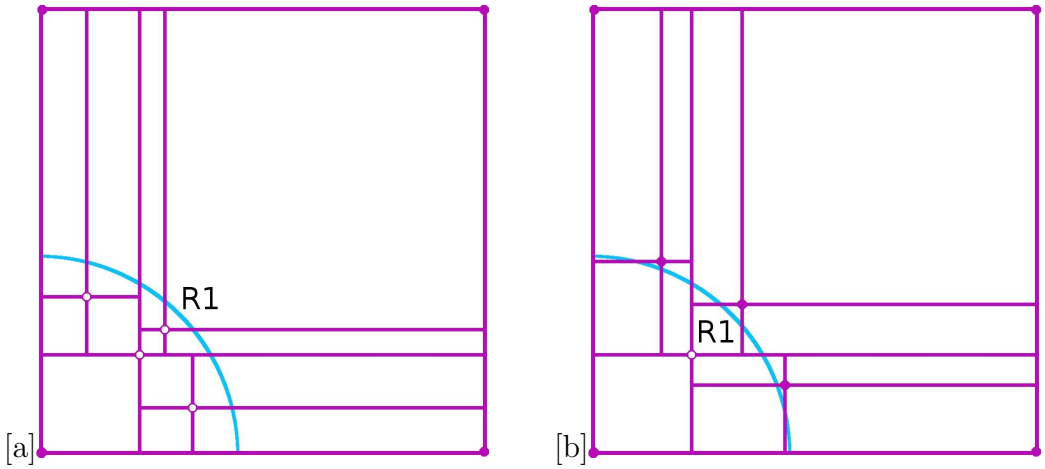


Figura 35: a) Todas as subdivisões estão no interior da superfície. b) Subdivisão inicial no interior da superfície, com as subdivisões seguintes na região exterior.

Para uma melhor convergência é necessário posicionar os pontos de subdivisão alternadamente na região interior e exterior à superfície. Para isso é utilizado o vetor gradiente iterativamente, reposicionando o ponto de subdivisão.

4.3.3
Posicionamento do ponto de subdivisão fora da superfície

Se, ao fazer a subdivisão do cubo, o ponto da subdivisão estiver sobre a superfície, isto resultará em uma degeneração, pois como será visto com

mais detalhes no próximo capítulo, os valores da função implícita nos vértices das Arestas Mínimas são usados para saber se estes vértices possuem sinais distintos, e portanto se estas arestas intersectam a superfície. Para evitar que o ponto de subdivisão esteja sobre a superfície, basta mover o ponto da subdivisão para a região interior ou exterior da superfície, podendo ser usado o vetor gradiente para posicionar este ponto, como pode ser visto na figura 36.

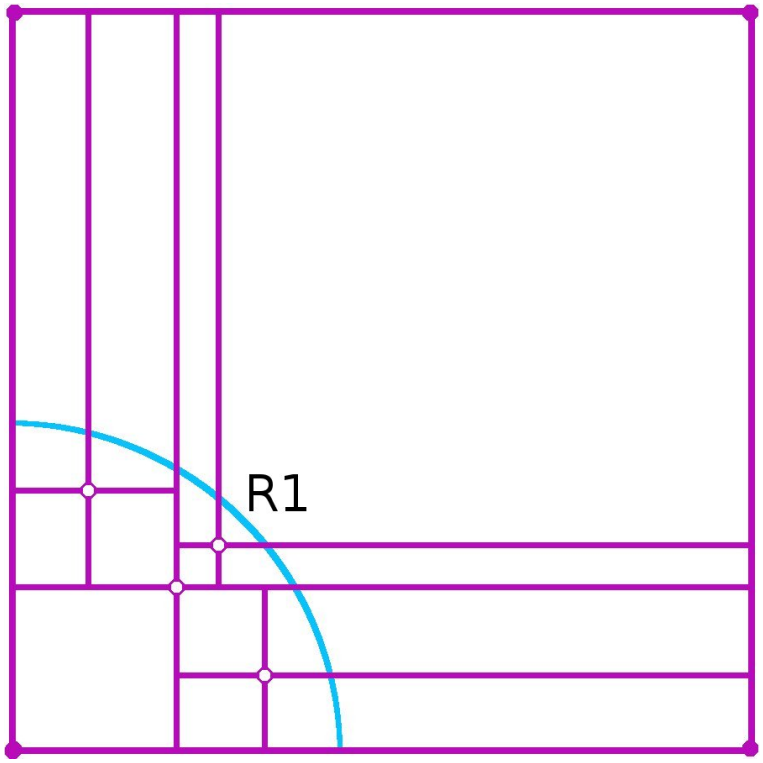


Figura 36: Reposicionamento do ponto de subdivisão para que não esteja sobre a curva.

4.3.4 Posicionando o Ponto da Subdivisão com Extremos do Cubo com Sinais Distintos

O posicionamento do ponto de subdivisão neste trabalho é feito em duas etapas. Inicialmente um ponto é posicionado utilizando as informações dos vértices do cubo, e na segunda etapa sua posição é modificada usando o vetor gradiente.

Para o posicionamento inicial, os valores da função implícita nos vértices do cubo são analisados e, nas arestas cujos extremos tenham vértices com mudança de sinal é feita uma interpolação linear para calcular o ponto desta aresta por onde passa a superfície. Após calcular estas interseções, o ponto

de subdivisão inicial pode ser escolhido somente calculando uma média dos valores de interseção nas coordenadas x, y e z.

A segunda etapa, que pode ser opcional, considera o ponto de subdivisão calculado na primeira etapa e usa o vetor gradiente para reposicioná-lo mais próximo à superfície, inclusive determinando se ele deve estar na região interior ou exterior à superfície. Esta segunda etapa pode ser repetida mais de uma vez a fim de encontrar uma melhor posição para este ponto. O critério de parada pode ser o número de iterações definido pelo usuário, a distância entre dois pontos da sequência de aproximação ser menor do que um valor determinado, ou que o ponto se mantenha na região desejada, dentro ou fora da superfície.

A figura 37 a) mostra um posicionamento inicial, utilizando somente os pontos de interseção da curva com as arestas do quadrado, sem o uso do gradiente. As figuras 37 b) e c) mostram o reposicionamento deste ponto utilizando o vetor gradiente, e a figura 37 d) mostra o posicionamento final deste ponto de subdivisão com a grade gerada nesta subdivisão.

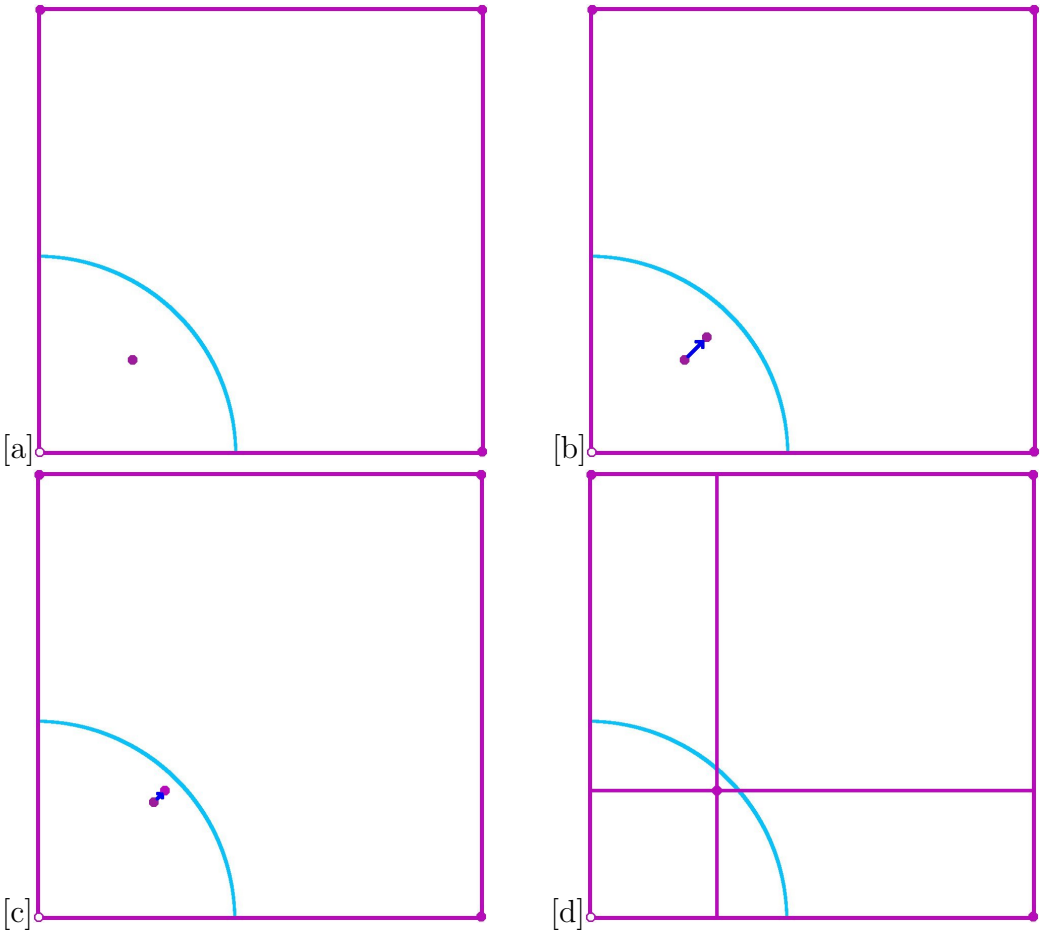


Figura 37: a) Posicionamento inicial do ponto de subdivisão. b) Primeiro reposicionamento utilizando o vetor gradiente. c) Segundo reposicionamento utilizando o vetor gradiente. d) Grade gerada e seu ponto de subdivisão.

O posicionamento do ponto de subdivisão da superfície é a primeira etapa do método apresentado neste trabalho, a Grade Adaptativa. Esta etapa é utilizada para a geração da *octree*, e para encontrar os pontos que geram a malha.

Para cada folha da *octree* que intersecta a superfície, os pontos da malha são calculados por meio de uma interpolação linear dos pontos de interseção nas arestas das folhas. Após o cálculo dos pontos da malha eles devem ser conectados e para isso serão utilizadas as arestas dos cubos que estão relacionados na *octree* às suas folhas. A definição de Aresta Mínima apresentada no *Dual Contouring* (4) não funciona para a *octree* gerada com a subdivisão proposta neste capítulo. No próximo capítulo é apresentada então a modificação proposta nesta tese para a Aresta Mínima, que na verdade engloba e estende a definição de Aresta Mínima utilizada pelo *Dual Contouring*.

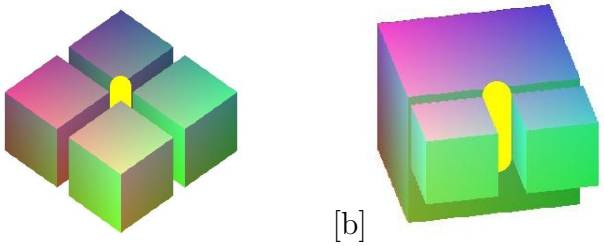
5
Grade Adaptativa: Redefinição de Arestas Mínimas

No capítulo anterior foi descrita a construção da *octree*, de forma que os pontos da subdivisão não estejam obrigatoriamente no seu centro. Neste capítulo é mostrado como conectar os pontos da malha gerada pela *octree*. Para isso é necessário redefinir o conceito de Arestas Mínimas, apresentado no *Dual Contouring*.

Este capítulo inicia apresentando a definição original de Arestas Mínimas. Em seguida são apresentados casos em que a tentativa de utilizar essa definição em conjunto com a subdivisão proposta neste trabalho falha. Finaliza apresentando a modificação proposta neste trabalho para a Aresta Mínima, destinada a octrees cujos pontos de subdivisão não estão necessariamente no centro.

5.1
Definição de Arestas Mínimas no Dual Contouring

As Arestas Mínimas são definidas como as arestas resultantes do encontro de três ou quatro cubos, que correspondem a folhas na *octree*. Portanto, uma Aresta Mínima não tem subdivisões, uma vez que os cubos a que ela pertence não são subdivididos. A figura 38 mostra dois casos em que três ou quatro cubos podem se encontrar em uma Aresta Mínima, em amarelo. Em 38 a) é possível ver que os quatro cubos não têm subdivisão e se encontram em uma única aresta comum aos quatro. Em 38 b) é possível ver o caso em que três cubos se encontram em torno de uma Aresta Mínima. Neste caso, é possível observar que tal aresta tem o comprimento do lado dos dois cubos menores.



[a] [b]

Figura 38: Encontro de cubos em torno de uma Aresta Mínima. a) Quatro cubos. b) Três cubos.

A figura 39 a) mostra dois cubos subdivididos no seu centro. É possível ver que a aresta central (vertical, em destaque na figura) não é uma Aresta Mínima, pois está subdividida. E que ela contém duas Arestas Mínimas, numeradas como 1 e 2. Na definição clássica de *octree*, como a subdivisão é feita sempre no centro do cubo, dois cubos que correspondam a folhas no mesmo nível da *octree* têm o mesmo tamanho, e se eles se encontram em uma aresta, os extremos V1 e V2 desta aresta são os mesmos, como pode ser visto na figura 39 b).

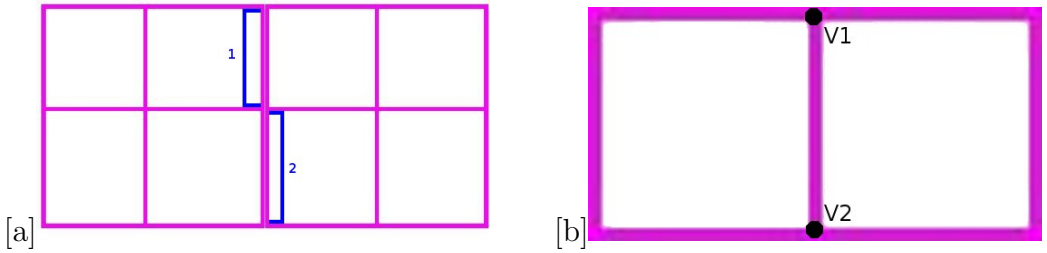


Figura 39: Aresta Mínima. a) Aresta, no centro, que contém duas Arestas Mínimas. b) Extremos da Aresta Mínima 1, V1 e V2.

O algoritmo para percorrer a *octree* e chegar às arestas mínimas é um algoritmo recursivo que parte do nó raiz e percorre seus filhos até chegar às folhas da *octree*, encontrando as arestas resultantes do encontro dos cubos referentes a estas folhas, e que portanto não têm subdivisão.

A cada subdivisão de um cubo, somente são consideradas as arestas no interior do cubo que está sendo subdividido, pois tais arestas são resultantes da interseção dos cubos resultantes dessa subdivisão. As arestas na face desse cubo somente são levadas em conta se estiverem no interior de um outro cubo, que deve estar em um nível superior na *octree*. Caso contrário, elas estão no bordo do domínio de poligonalização e são resultantes da conexão de somente dois cubos tendo um espaço vazio ao seu redor, como pode ser visto na figura 40. Nessa figura, é possível observar que esta aresta é resultante de somente dois cubos. Portanto, em torno dessa aresta podem existir no máximo dois pontos da malha e consequentemente não é possível gerar um elemento da malha em torno dela, seja ele um triângulo ou um retângulo.

Existem três formas de gerar novas arestas:

- No interior da subdivisão de um cubo.
- Na subdivisão de pelo menos um dos dois cubos que se encontram em uma face.
- Na subdivisão de pelo menos um dos três ou quatro cubos que se encontram em uma aresta.

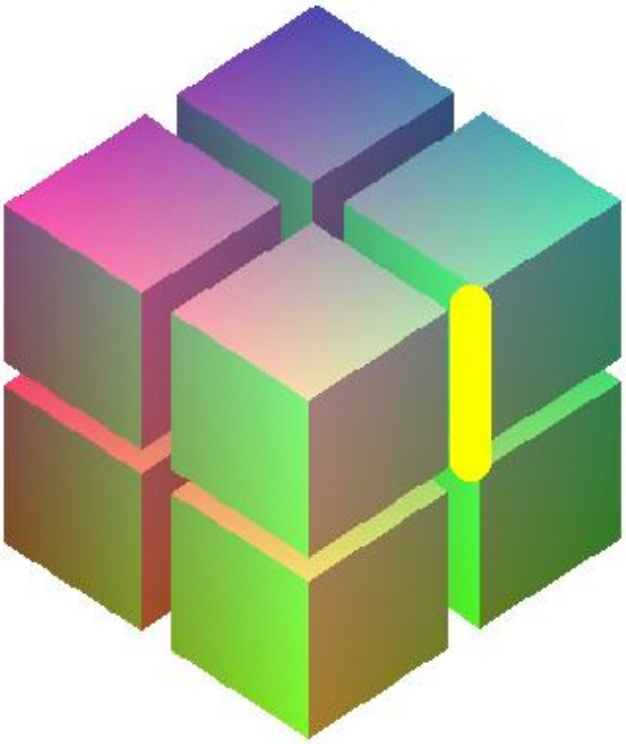


Figura 40: A aresta na face do cubo, em amarelo, não é utilizada na poligonalização da superfície.

Estes três casos são analisados a seguir, inicialmente de acordo com a definição original de Aresta Mínima, com a subdivisão sempre no centro do cubo, e depois é apresentada a modificação proposta nessa tese, em que a posição de subdivisão está próxima à superfície.

5.1.1 Arestas Geradas a Partir da Subdivisão de um Cubo

Quando um cubo é subdividido, ele gera oito novos cubos como pode ser visto na figura 41 b). Além disso, são geradas doze faces, definidas pelo encontro de cubos dois a dois, sendo quatro faces em cada um dos planos xy, xz e yz mostradas na figura 41 c). E por fim, a figura 41 d) representa as seis arestas que são geradas nessa subdivisão, definidas pelo encontro de cubos quatro a quatro, sendo duas na direção do eixo x, duas na direção do eixo y e duas na direção do eixo z.

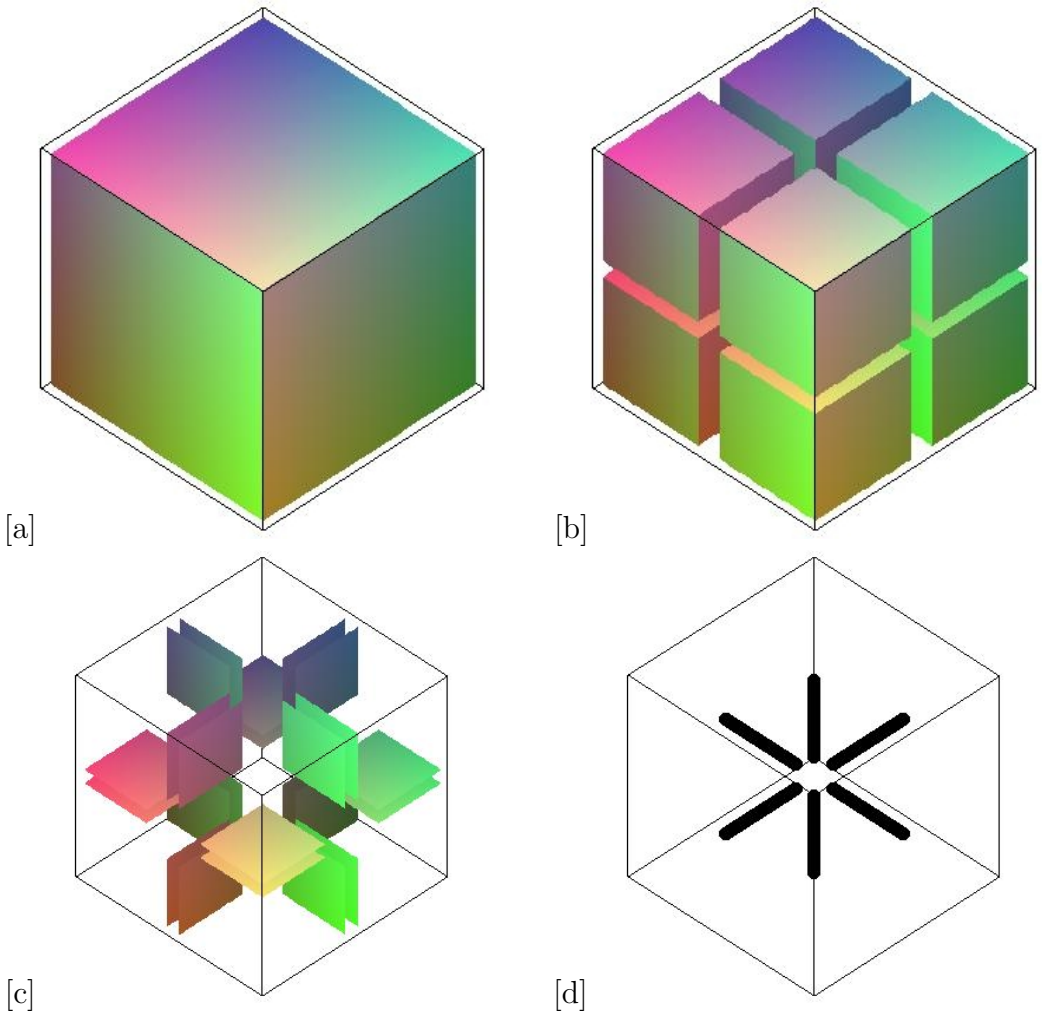


Figura 41: Subdivisão de um cubo. a) Cubo original. b) Os oito cubos resultantes da subdivisão. c) Doze faces formadas pelo encontro de cubos dois a dois. d) Seis arestas formadas pelo encontro de cubos quatro a quatro.

5.1.2
Arestas Geradas a Partir da Subdivisão de Faces

Cada encontro de dois cubos em uma face pode gerar novas arestas, se pelo menos um deles for subdividido. As arestas resultantes do encontro destes dois cubos estão na face em que eles se encontram.

A figura 42 a) mostra dois cubos que se encontram em uma face. Ao dividir um dos cubos são geradas quatro novas faces, resultantes do encontro de cubos dois a dois nessa face, como pode ser visto nas figuras 42 b) e c). Cada uma dessas faces é resultante do encontro de um dos quatro cubos resultantes da subdivisão do cubo à esquerda com o cubo à direita, o qual não foi subdividido.

Com a subdivisão do cubo também são geradas quatro novas arestas que, neste caso, são resultantes do encontro de três cubos, dois deles resultantes da

subdivisão do cubo à esquerda, pois somente ele foi subdividido, como pode ser visto na figura 42 d).

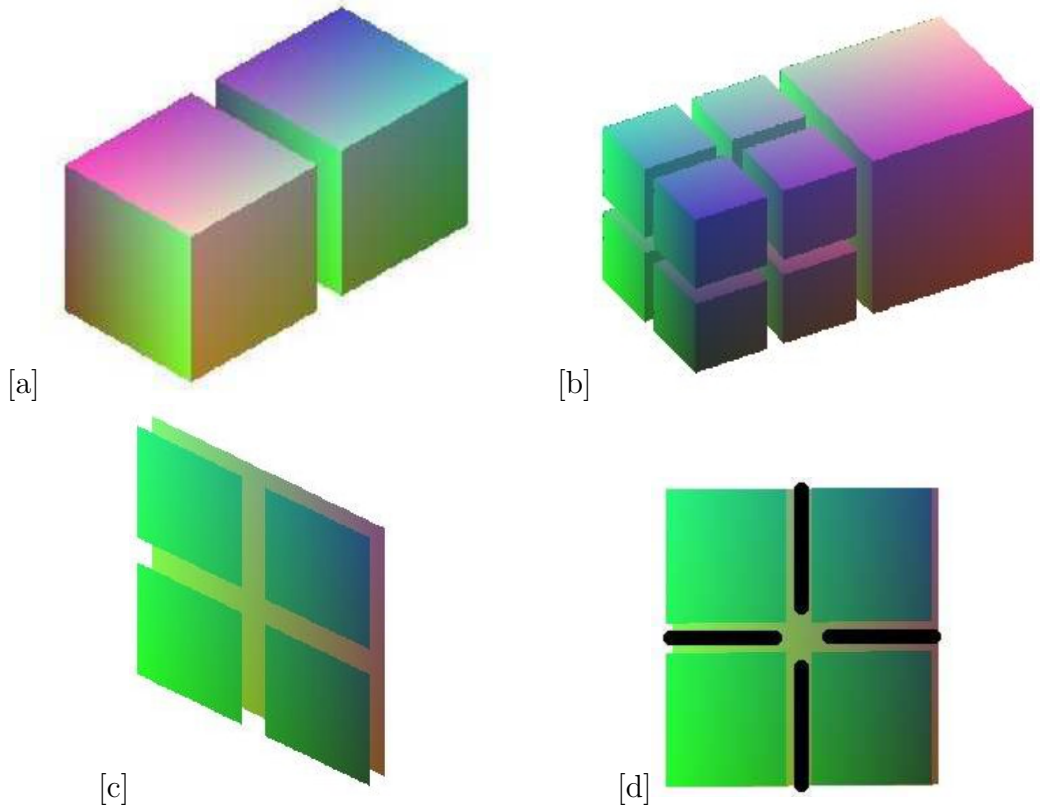


Figura 42: Subdivisão de um dos dois cubos que se encontram em uma face. a) Cubos originais. b) Subdivisão do cubo à esquerda. c) As quatro faces resultantes da subdivisão. d) As quatro arestas resultantes da subdivisão.

No caso mostrado na figura 42, como a subdivisão utilizada é a clássica, com o ponto de subdivisão sempre no centro do cubo, caso os dois cubos fossem subdivididos esta subdivisão resultaria no mesmo número de faces e arestas. Se o cubo à direita também tivesse sido subdividido, os quatro cubos à esquerda se encontrariam com outros quatro cubos à direita, mas as faces resultantes da subdivisão manteriam a mesma posição das faces resultantes da subdivisão somente do cubo à esquerda. No caso das arestas, com a subdivisão também do cubo à direita, cada uma delas seria resultante do encontro de quatro cubos.

5.1.3
Arestas Geradas a partir da Subdivisão de uma Aresta

Como uma aresta é definida pelo encontro de três ou quatro cubos, se um deles for subdividido são geradas duas novas arestas, como pode ser visto na figura 43 c) e d). Também neste caso, como a subdivisão utilizada é a clássica - com o ponto de subdivisão sempre no centro do cubo-, essas subdivisões vão se encontrar e serão geradas duas novas arestas, independentemente de todos ou somente um dos cubos ser subdividido.

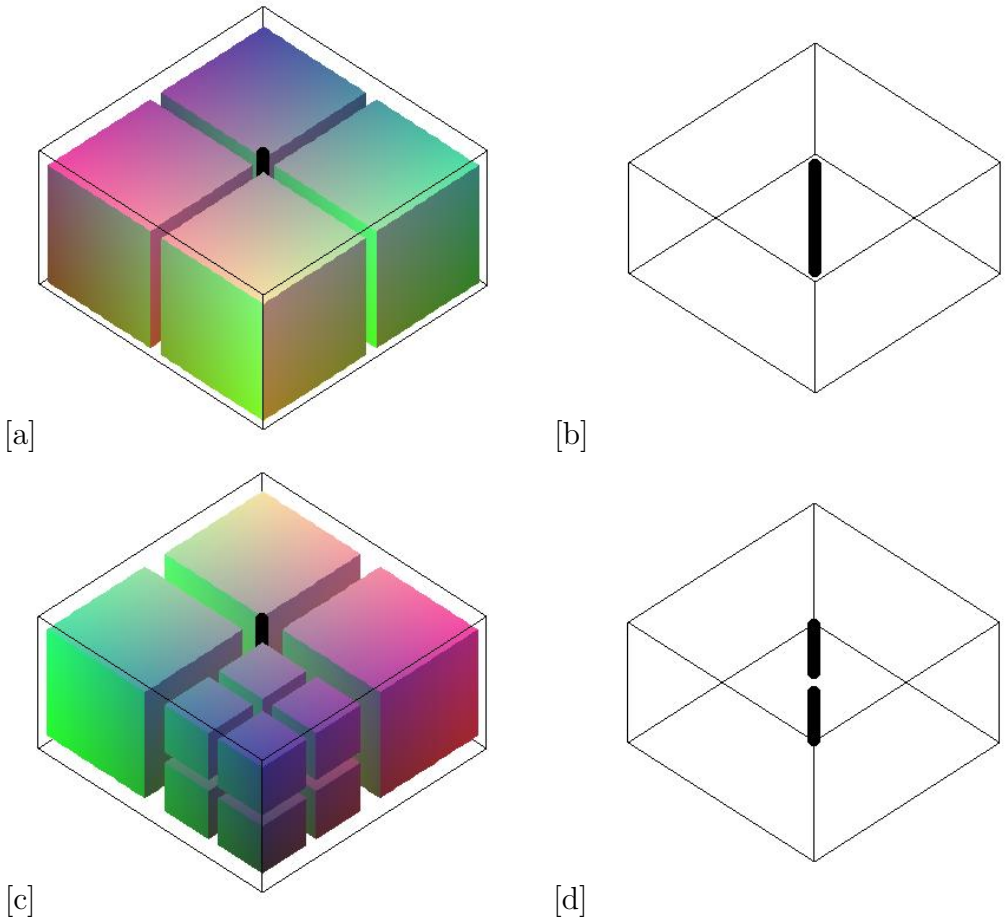


Figura 43: Subdivisão de uma aresta. a) Quatro cubos se encontrando em uma aresta. b) Aresta definida pelo encontro dos quatro cubos. c) Subdivisão de um dos quatro cubos, resultando em duas novas arestas. d) As duas arestas resultantes da subdivisão do cubo.

5.2
Pseudo-código de Arestas Mínimas para o Dual Contouring

A partir do exposto, é possível apresentar o algoritmo de conexão de arestas, que é recursivo e percorre toda a *octree* até chegar às suas folhas. Para saber se a superfície passa por uma Aresta Mínima os valores da função implícita são calculados nos seus vértices. Caso possuam sinais distintos, a superfície passa por essa aresta, e os pontos da malha que estão nos cubos que nela se encontram são conectados.

O algoritmo para percorrer a *octree* e encontrar as arestas mínimas é recursivo e composto de três funções: $Procura\_cubo(q1)$, $Procura\_face(q1, q2)$ e $Procura\_aresta(q1, q2, q3, q4)$. Estas chamadas são todas recursivas e percorrem os cubos, as faces (que são definidas como o encontro de dois cubos) e as arestas (que são definidas como o encontro de três ou quatro cubos) até chegar às folhas da *octree*, encontrando as arestas mínimas.

A divisão de um cubo gera oito novos cubos enumerados como pode ser visto na figura 44.

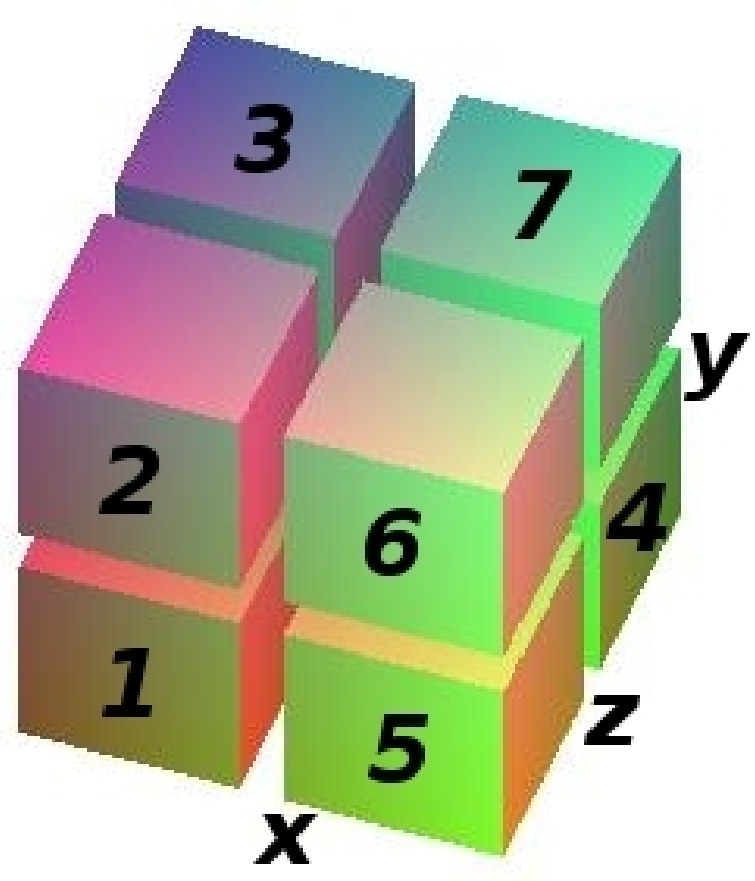


Figura 44: Subdivisão de um cubo e a enumeração dos cubos resultantes da subdivisão.

5.2.1 Procura_cubo(q1)

Quando um cubo é subdividido, ele gera oito novos cubos, doze novas faces e seis novas arestas, portanto a função $Procura\_cubo$ faz oito chamadas à função $Procura\_cubo$, doze chamadas à função $Procura\_face$ e seis chamadas à função $Procura\_aresta$.

O pseudo-código para a função $Procura\_cubo$ é dado abaixo, onde $q1$ é o cubo original e $q1 \rightarrow pi$ é cada um dos oito cubos resultantes da subdivisão de $q1$:

Procura_cubo(q1) {
$If(q1 \rightarrow p0 \; != NULL)$ {
$Procura\_cubo(q1 \rightarrow p0)$;
$Procura\_cubo(q1 \rightarrow p1)$;
$Procura\_cubo(q1 \rightarrow p2)$;
$Procura\_cubo(q1 \rightarrow p3)$;
$Procura\_cubo(q1 \rightarrow p4)$;
$Procura\_cubo(q1 \rightarrow p5)$;
$Procura\_cubo(q1 \rightarrow p6)$;
$Procura\_cubo(q1 \rightarrow p7)$;

$Procura\_face(q1 \rightarrow p0, q1 \rightarrow p4)$;
$Procura\_face(q1 \rightarrow p1, q1 \rightarrow p5)$;
$Procura\_face(q1 \rightarrow p2, q1 \rightarrow p6)$;
$Procura\_face(q1 \rightarrow p3, q1 \rightarrow p7)$;
$Procura\_face(q1 \rightarrow p0, q1 \rightarrow p3)$;
$Procura\_face(q1 \rightarrow p1, q1 \rightarrow p2)$;
$Procura\_face(q1 \rightarrow p4, q1 \rightarrow p7)$;
$Procura\_face(q1 \rightarrow p5, q1 \rightarrow p6)$;
$Procura\_face(q1 \rightarrow p0, q1 \rightarrow p1)$;
$Procura\_face(q1 \rightarrow p2, q1 \rightarrow p3)$;
$Procura\_face(q1 \rightarrow p4, q1 \rightarrow p5)$;
$Procura\_face(q1 \rightarrow p6, q1 \rightarrow p7)$;

$Procura\_aresta(q1 \rightarrow p0, q1 \rightarrow p1, q1 \rightarrow p4, q1 \rightarrow p5)$;
$Procura\_aresta(q1 \rightarrow p2, q1 \rightarrow p3, q1 \rightarrow p6, q1 \rightarrow p7)$;
$Procura\_aresta(q1 \rightarrow p0, q1 \rightarrow p3, q1 \rightarrow p4, q1 \rightarrow p7)$;

$Procura\_aresta(q1 \rightarrow p1, q1 \rightarrow p2, q1 \rightarrow p5, q1 \rightarrow p6);$
$Procura\_aresta(q1 \rightarrow p0, q1 \rightarrow p1, q1 \rightarrow p2, q1 \rightarrow p3);$
$Procura\_aresta(q1 \rightarrow p4, q1 \rightarrow p5, q1 \rightarrow p6, q1 \rightarrow p7);$
}
}

5.2.2 Procura_face(q1,q2)

A função *Procura_cubo* só procura as arestas mínimas localizadas no interior de um cubo. Portanto, para encontrar as arestas em uma face resultante do encontro de dois cubos, devem ser feitas chamadas à $Procura\_face$. Em uma face, se um ou ambos os cubos são subdivididos, são geradas quatro novas faces e quatro novas arestas. Portanto são feitas quatro chamadas à $Procura\_face$ e quatro chamadas à $Procura\_Aresta$.

É utilizada uma função auxiliar $Encontra\_Face\_Face$, que guarda no vetor $cubos\_face$ os índices das subdivisões dos dois cubos, $q1$ e $q2$, que se conectam nessa face.

O pseudo-código para a função $Procura\_face$ é dado abaixo:

Procura_face(q1,q2) {
$If((q1 \rightarrow p0 != NULL)||(q2 \rightarrow p0 != NULL))$ {
$Encontra\_Face\_Face(q1, q2, cubos\_face);$
$if(q1 \rightarrow p0 != NULL)$ {
$cubo[0][0] = q1 \rightarrow p[cubos\_face[0][0]];$
$cubo[0][1] = q1 \rightarrow p[cubos\_face[0][1]];$
$cubo[0][2] = q1 \rightarrow p[cubos\_face[0][2]];$
$cubo[0][3] = q1 \rightarrow p[cubos\_face[0][3]];$
}$else$ {
$cubo[0][0] = cubo[0][1] = cubo[0][2] = cubo[0][3] = q1;$
}

$if(q2 \rightarrow p[0] != NULL)$ {
$cubo[1][0] = q2 \rightarrow p[cubos\_face[1][0]];$
$cubo[1][1] = q2 \rightarrow p[cubos\_face[1][1]];$
$cubo[1][2] = q2 \rightarrow p[cubos\_face[1][2]];$

```
            cubo[1][3] = q2 → p[cubos_face[1][3]];
        }else {
            cubo[1][0] = cubo[1][1] = cubo[1][2] = cubo[1][3] = q2;
        }

        Procura_Face(cubo[0][0], cubo[1][0]);
        Procura_Face(cubo[0][1], cubo[1][1]);
        Procura_Face(cubo[0][2], cubo[1][2]);
        Procura_Face(cubo[0][3], cubo[1][3]);

        Procura_Aresta(cubo[0][0], cubo[0][1], cubo[1][0], cubo[1][1]);
        Procura_Aresta(cubo[0][1], cubo[0][2], cubo[1][1], cubo[1][2]);
        Procura_Aresta(cubo[0][2], cubo[0][3], cubo[1][2], cubo[1][3]);
        Procura_Aresta(cubo[0][3], cubo[0][0], cubo[1][3], cubo[1][0]);
    }
}
```

5.2.3 Procura_aresta(q1,q2,q3,q4)

Uma aresta pode ser formada pelo encontro de três ou quatro cubos. Ela será subdividida se pelo menos um dos cubos que se encontram nela for subdividido. A função *Procura_aresta* gera duas novas chamadas à *Procura_aresta*.

Na função *Procura_aresta* se uma aresta é resultante do encontro de três cubos então dois dos *qis* são iguais; caso contrário, eles são todos distintos, correspondendo aos quatro cubos. É utilizada uma função auxiliar *Encontra_Aresta_Aresta* que, dados três ou quatro cubos que se encontram nela, $q1$, $q2$, $q3$ e $q4$, guarda os índices dos cubos correspondentes às arestas inferior e superior, pois elas podem se encontrar no eixo x, y ou z.

```
Procura_Aresta(q1, q2, q3, q4) {
    if((q1 → p0 != NULL)||(q2 → p0 != NULL)
            ||(q3 → p0 != NULL)||(q4 → p0 != NULL)) {
        Encontra_Aresta_Aresta(q1, q2, q3, q4, cubos_aresta);
        if(q1 → p0 != NULL) {
            q11 = q1 → p[cubos_aresta[0][0]];
```

```
            q12 = q1 → p[cubos_aresta[0][1]];
        }else {
            q11 = q12 = q1;
        }

        if(q2 → p0 != NULL) {
            q21 = q2 → p[cubos_aresta[1][0]];
            q22 = q2 → p[cubos_aresta[1][1]];
        }else {
            q21 = q22 = q2;
        }

        if(q3 → p0 != NULL) {
            q31 = q3 → p[cubos_aresta[2][0]];
            q32 = q3 → p[cubos_aresta[2][1]];
        }else {
            q31 = q32 = q3;
        }

        if(q4 → p0 != NULL) {
            q41 = q4 → p[cubos_aresta[3][0]];
            q42 = q4 → p[cubos_aresta[3][1]];
        }else {
            q41 = q42 = q4;
        }

        Procura_aresta(q11, q21, q31, q41);
        Procura_aresta(q12, q22, q32, q42);
    }
}
```

5.3 Problemas ao Utilizar a Definição Clássica de Arestas Mínimas na Grade Adaptativa

Nesta seção são mostrados três exemplos de dificuldades que surgem ao tentar gerar a malha utilizando a subdivisão proposta nesta tese em conjunto

com a definição original da Aresta Mínima.

A figura 45 mostra um octante de uma esfera poligonalizada com dois níveis de subdivisão, ambas geradas com a subdivisão da *octree* proposta nesta tese. A figura 45 a) exibe a conexão dos pontos da malha com a definição original de Aresta Mínima. É possível ver, na figura 45, que as conexões não são feitas corretamente pois a superfície resultante apresenta buracos. Com a conexão dos pontos efetuada de acordo com a modificação de Aresta Mínima apresentada nessa tese é possível ver na figura 45 b) que os vértices da malha são conectados corretamente, mantendo a topologia da superfície.

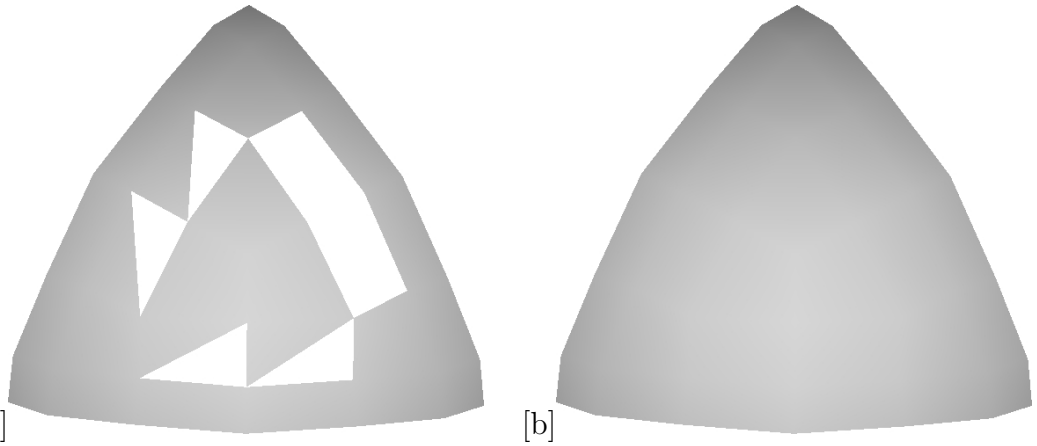
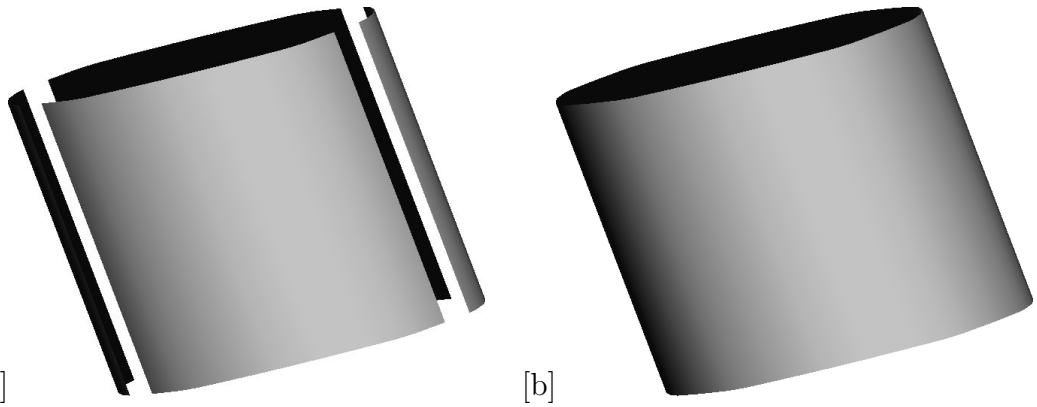


Figura 45: Octante de uma esfera. a) Conexão dos vértices da malha com a definição original de Aresta Mínima. b) Conexão com a definição de Aresta Mínima apresentada nessa tese.

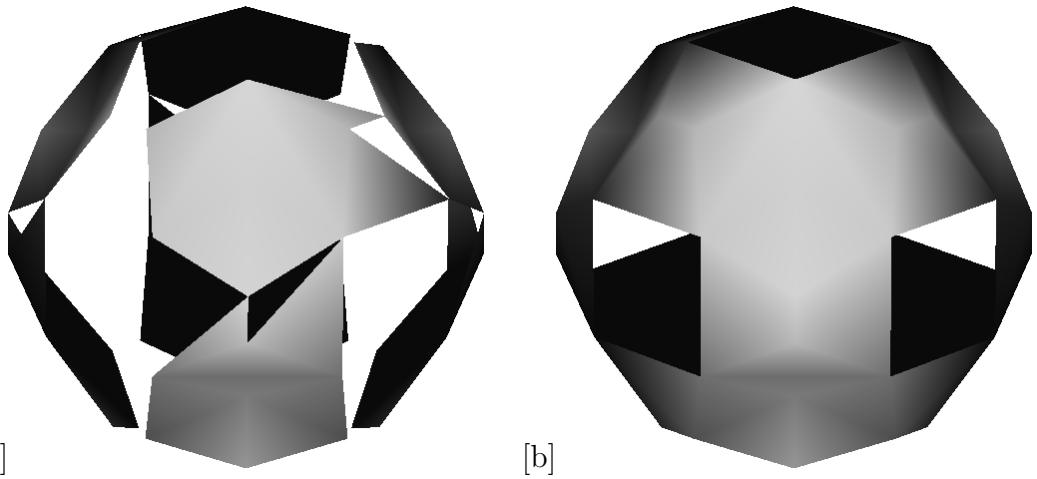
O cilindro de base elíptica mostrado na figura 46 é gerado com três níveis de subdivisão da *octree*. A figura 46 a) mostra a conexão dos pontos da malha utilizando a definição original de Aresta Mínima. Foram gerados quatro componentes conexas e não um cilindro. Na figura 46 b) a conexão dos pontos da malha é feita com a definição de Aresta Mínima apresentada nesta tese, e o resultado é um cilindro sem nenhuma alteração na sua topologia, pois todas as arestas mínimas foram corretamente identificadas e portanto os pontos da malha estão corretamente conectados.

Para finalizar esta seção é mostrada uma superfície não compacta gerada com dois níveis na subdivisão da *octree* com seis buracos, onde cada buraco está na direção de um dos eixos: x, y ou z. Com a conexão dos pontos da malha utilizando a definição original de Aresta Mínima vários pontos da malha não são conectados, como pode ser visto na figura 47 a). A figura 47 b) mostra a conexão feita com a definição de Aresta Mínima apresentada nessa tese, com os pontos da malha conectados corretamente.



[a] [b]

Figura 46: Cilindro de base elíptica. a) As quatro componentes conexas resultantes da poligonalização com a definição original de Aresta Mínima. b) Conexão com a definição de Aresta Mínima apresentada nessa tese, resultando em um cilindro.



[a] [b]

Figura 47: Superfície não compacta. a) Conexão dos vértices da malha com a definição original de Aresta Mínima. b) Conexão com a definição de Aresta Mínima apresentada nessa tese.

Com os exemplos mostrados nas figuras 45, 46 e 47 a) é possível ver que a definição original de Aresta Mínima não consegue conectar corretamente os pontos da malha. Na seção seguinte é mostrada a definição de Aresta Mínima proposta nesta tese, que conecta corretamente os pontos da malha para a *octree* gerada com os pontos da subdivisão próximos à superfície.

5.4 Arestas Mínimas na Grade Adaptativa

Como foi visto na descrição das Arestas Mínimas no *Dual Contouring*, elas são encontradas utilizando o fato de que a subdivisão sempre é efetuada no centro do cubo. Portanto, dados três ou quatro cubos que se encontram em uma aresta, e correspondem a folhas de mesmo nível na *octree*, se eles

forem subdivididos o ponto de subdivisão desses cubos se encontra na mesma posição, com relação à aresta, gerando duas novas arestas.

Com o algoritmo proposto nesta tese, como os pontos de subdivisão podem estar em posições diferentes, com relação à aresta em que eles se encontram, a subdivisão desses cubos pode gerar de uma a cinco novas arestas, como é mostrado nesta seção.

Como o ponto de subdivisão não está sempre no centro, a subdivisão de um cubo usualmente gera paralelepípedos, e não cubos, como pode ser visto na figura 48. Nesta seção são feitas referências a cubos, mesmo que da subdivisão resultem paralelepípedos. Assim como na definição clássica de Aresta Mínima, no algoritmo proposto nessa tese uma nova aresta é gerada de três formas:

- No interior da subdivisão de um cubo.
- Na subdivisão de pelo menos um dos dois cubos que se encontram em uma face.
- Na subdivisão de pelo menos um dos três ou quatro cubos que se encontram em uma aresta.

5.4.1
Arestas Geradas a Partir da Subdivisão de um Cubo

Quando um cubo é subdividido, mesmo que a subdivisão não seja definida a partir do centro, o número de cubos, encontros de faces e encontros de arestas são mantidos, oito cubos, doze encontros de faces e seis encontros de arestas, como pode ser visto na figura 48.

5.4.2
Arestas Geradas a Partir da Subdivisão de Faces

As mudanças na definição de Aresta Mínima iniciam nas arestas geradas quando pelo menos um dos dois cubos que se encontram em uma face é subdividido. Nesse caso, essa diferença pode ocorrer porque os pontos de subdivisão podem estar em posições distintas com relação à face, como mostra a figura 49. A figura 49 a) mostra dois cubos que se encontram em uma face. Esses cubos são resultantes de outros cubos, subdivididos fora do centro. Na figura 49 b) eles são subdivididos, cada um deles em posições distintas com relação à face. Na figura 49 c) é possível ver somente os cubos resultantes da subdivisão que se encontram nessa face, e na figura 49 d) são mostradas somente as faces pertencentes a esses cubos que se encontram nessa face.

A figura 50 mostra as faces e arestas apresentadas na figura 49. Nela é possível ver, figura 50 a) que são geradas nove novas faces e não quatro,

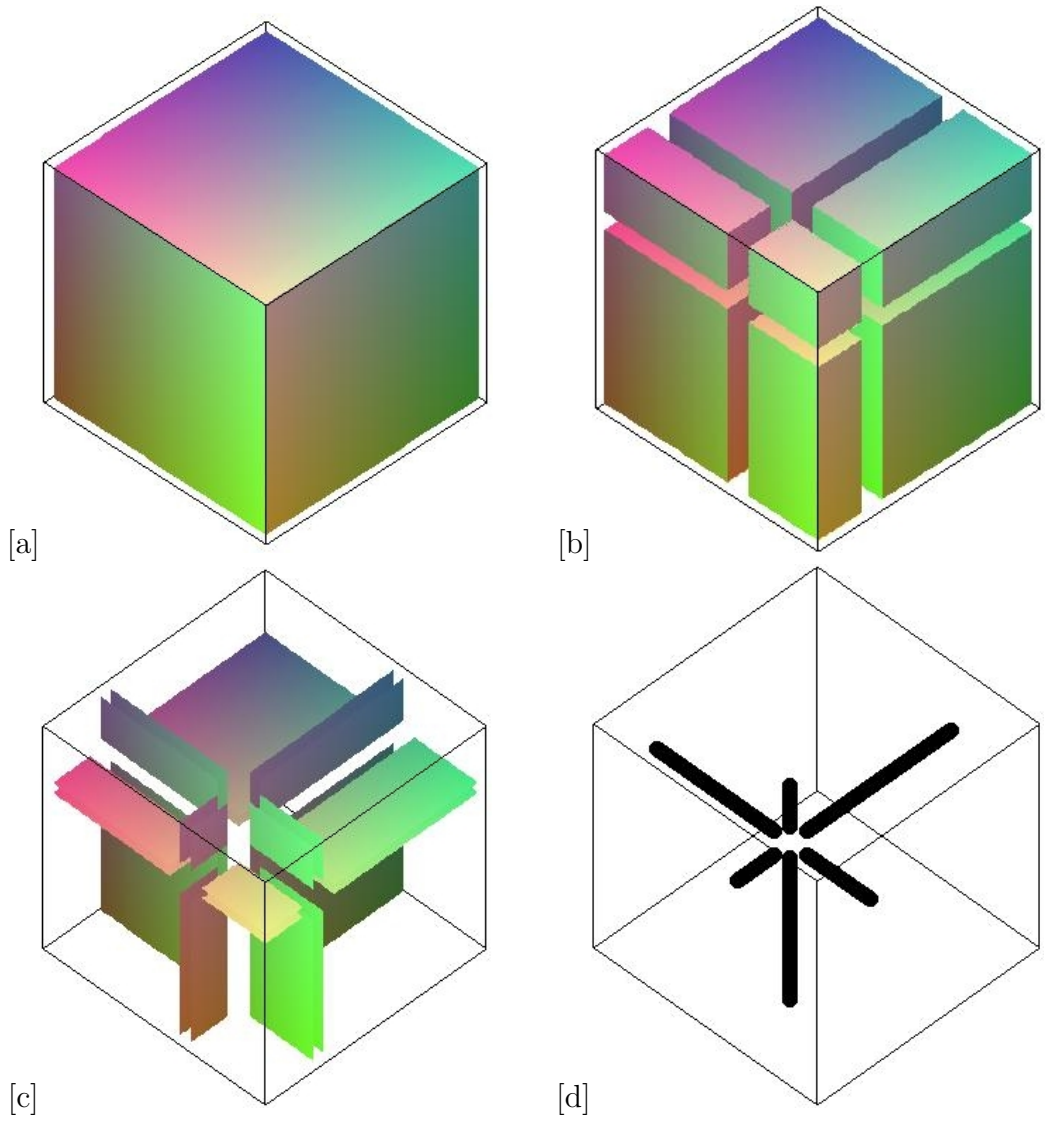


Figura 48: Subdivisão de um cubo. a) Cubo original. b) Cubos resultantes da subdivisão. c) Faces, definidas como o encontro de cubos dois a dois. d) Arestas, definidas como o encontro de cubos quatro a quatro.

como acontece na definição clássica de *octree*. Além disso, o número de novas arestas também é alterado, pois são geradas doze novas arestas, ao contrário da definição clássica que gerava sempre quatro novas arestas, como pode ser visto na figura 50 b).

Como a subdivisão apresentada neste trabalho não é efetuada sempre no centro do cubo, ela pode gerar cubos que se encontram em uma face, mas cuja interseção não está totalmente contida em nenhum dos dois cubos, ver figura 51 a). Nesse caso, os encontros de faces e arestas gerados na subdivisão desses dois cubos estão na região de sua interseção, conforme exibe a figura 51 b). A subdivisão dos dois cubos fora da região de interseção gera retas fora dessa região, que não são consideradas para esse passo da poligonalização, figura 51

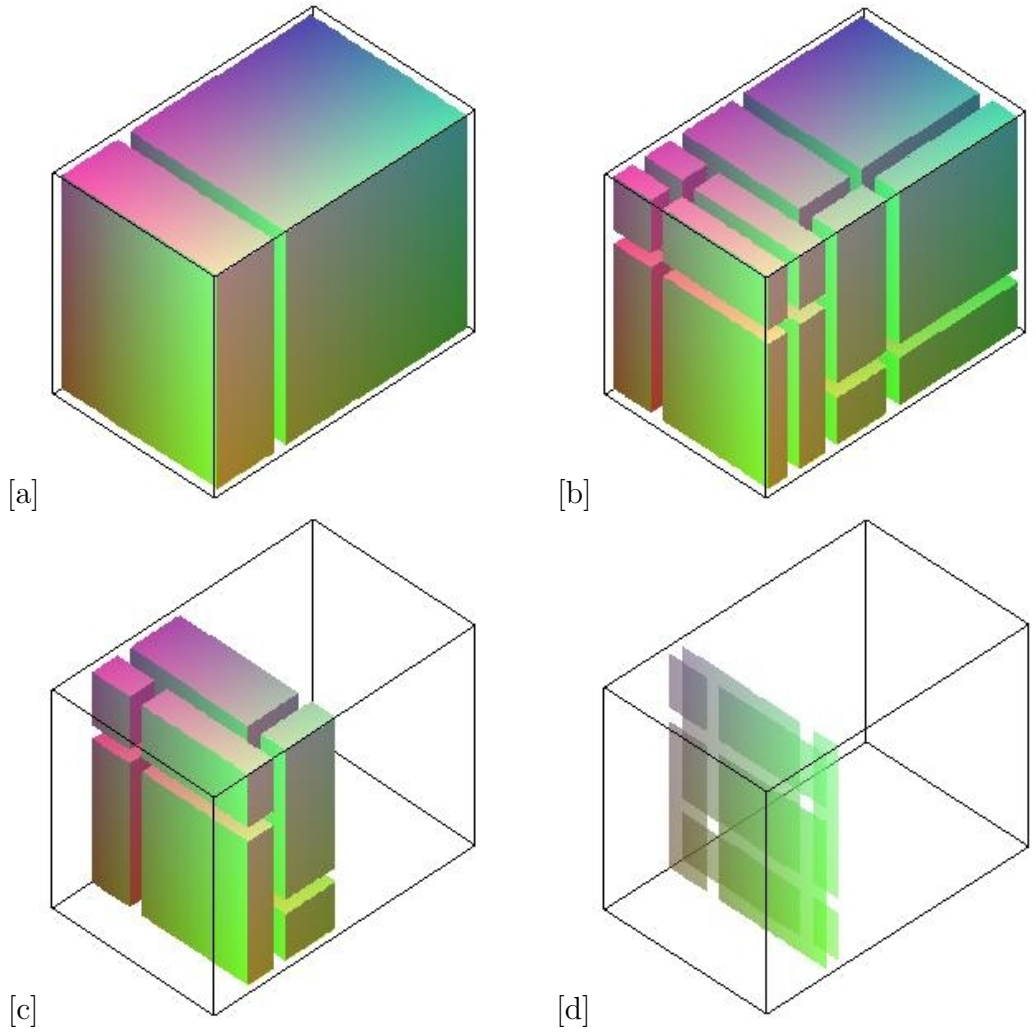


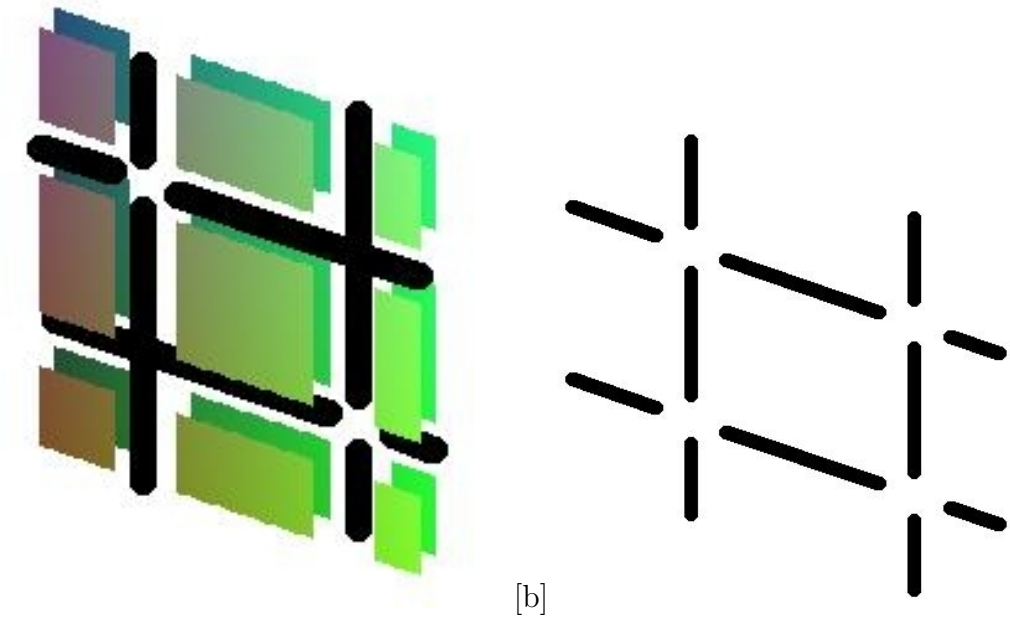
Figura 49: Subdivisão de dois cubos que se encontram em uma face. a) Dois cubos que se encontram em uma face. b) Subdivisão dos dois cubos. c) Cubos resultantes da subdivisão, que se encontram nessa face. d) Faces pertencentes a esses cubos que se encontram nessa face.

c). Na figura 51 d) é possível ver que esta subdivisão resulta em somente uma nova face e nenhuma nova aresta.

As chamadas às arestas e faces fora da região de interseção são tratadas na interseção dos encontros de faces a que elas pertencem, em outras chamadas do algoritmo de poligonalização.

A subdivisão de dois cubos que se encontram em uma aresta pode resultar desde apenas uma nova face e nenhuma nova aresta, como foi visto na figura 51, até nove novas faces e doze novas arestas, vistas na figura 50.

Na figura 52 é mostrada uma situação intermediária, em que duas das arestas das subdivisões estão na região de interseção das faces e as outras não. A figura 52 a) mostra os cubos antes da subdivisão, na figura 52 b) são mostrados os cubos resultantes da subdivisão. A figura 52 c) mostra, com transparência, a



[a] [b]

Figura 50: Faces e arestas resultantes da subdivisão de um cubo. a) As faces duas a duas. b) As arestas resultantes da subdivisão.

vista frontal dos cubos, com as faces e as arestas resultantes dessa subdivisão. Na figura 52 d) a região mais clara indica a região de interseção dos cubos, que é utilizada para a geração de novas faces e arestas.

Como foi visto acima, quando dois cubos se encontram em uma face, inicialmente deve ser encontrada a região de interseção dessas faces sendo determinadas então as arestas e faces nessa região de interseção.

5.4.3 Arestas Geradas a Partir da Subdivisão de uma Aresta

Assim como no caso do encontro de dois cubos em uma face, no caso de uma aresta, que é resultante do encontro de três ou quatro cubos, também há diferença entre a definição original de Aresta Mínima e o algoritmo proposto nessa tese. A aresta está na região de interseção desses cubos, seja no eixo x, y ou z, e as subdivisões que geram novas arestas são aquelas que ocorrem nessa região de interseção. Com a subdivisão dos cubos podem ser geradas de uma a cinco novas arestas. A figura 53 a) mostra quatro cubos que se encontram em uma aresta, todos os cubos são subdivididos, mas com todos os pontos de subdivisão fora da região de interseção, como pode ser visto na figura 53 b). A figura 53 c) mostra a única aresta gerada pela subdivisão com os outros cubos, onde a transparência foi usada a fim de melhor visualizar a aresta.

O caso em que são geradas cinco novas arestas é mostrado na figura 54. Em 54 a) são mostrados os cubos que se encontram em uma aresta no eixo

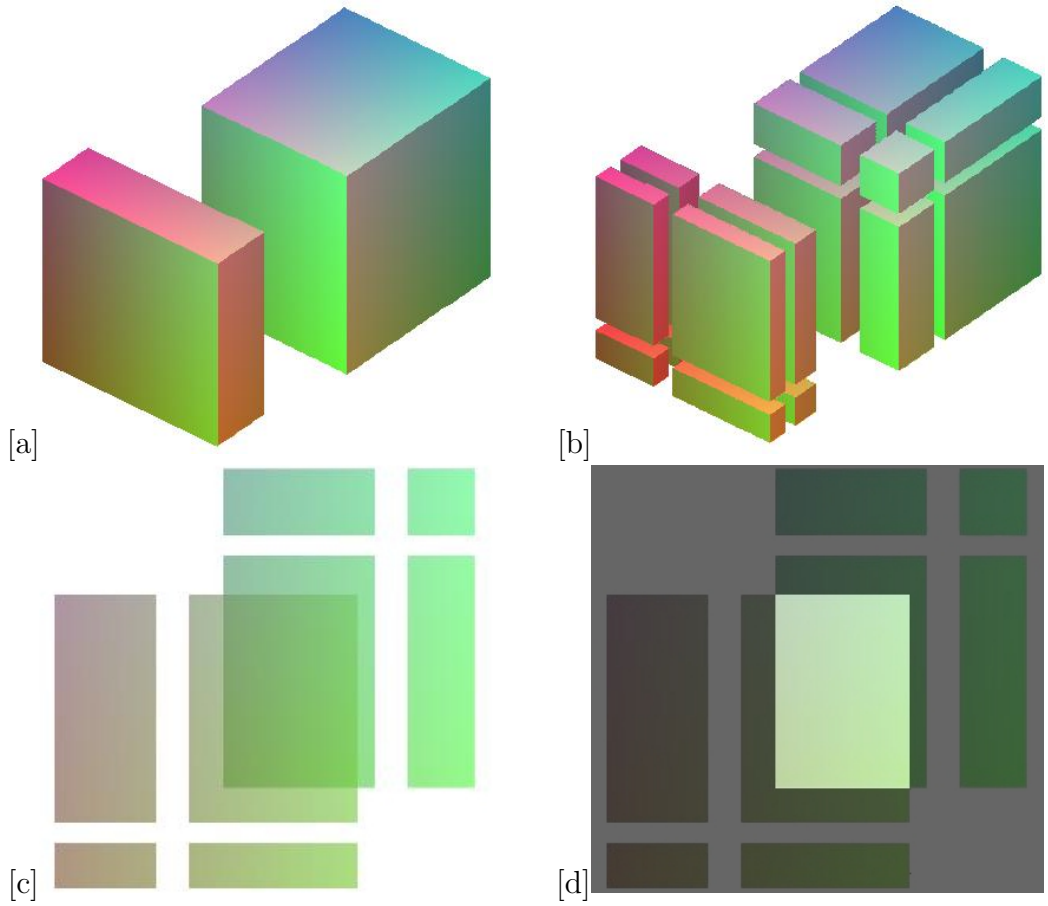


Figura 51: Uma nova face gerada na subdivisão. a) Os dois cubos originais. b) Subdivisão dos cubos. c) Vista de frente dos cubos. d) A região mais clara mostra a nova face resultante da subdivisão.

z. Todos os cubos são subdivididos com os pontos de subdivisão na região de interseção, figura 54 b), o que gera as cinco novas arestas mostradas na figura 54 c), e de novo utilizando transparência nos cubos.

5.5 Pseudo-código de Arestas Mínimas para a Grade Adaptativa

O algoritmo proposto nessa tese para encontrar as Arestas Mínimas, assim como o algoritmo original, é um algoritmo recursivo. Ele procura as arestas resultantes da subdivisão de um cubo, as arestas resultantes da subdivisão de pelo menos um dos dois cubos que se encontram em uma face, e as arestas resultantes da subdivisão de pelo menos um dos cubos que se encontram em uma aresta.

O algoritmo para percorrer a *octree* e encontrar as arestas mínimas é composto de três funções: $Procura\_cubo\_GA(q1)$, $Procura\_face\_GA(q1, q2)$ e $Procura\_aresta\_GA(q1, q2, q3, q4)$. Nas seções abaixo são descritas cada uma

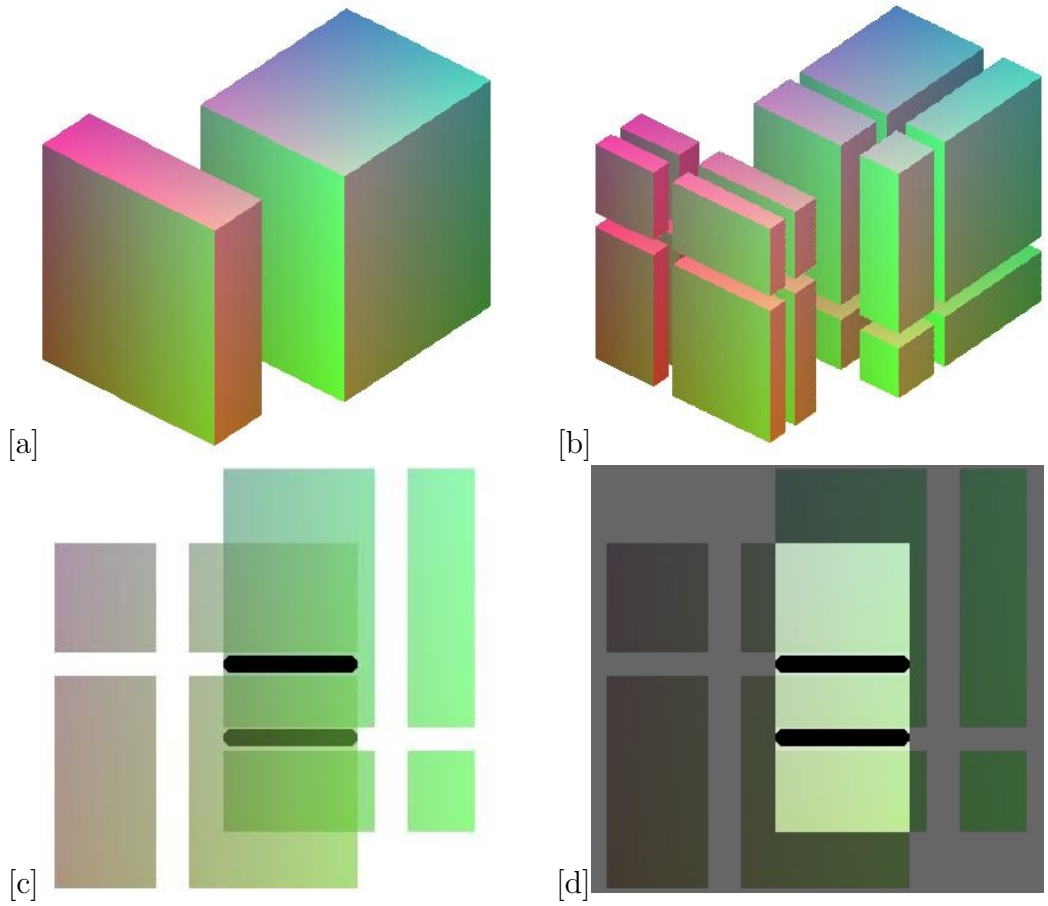


Figura 52: Três novas face e duas arestas geradas pela subdivisão. a) Os dois cubos originais. b) Subdivisão dos cubos. c) Vista de frente dos cubos. d) A região mais clara mostra as três novas faces e as duas novas arestas resultantes da subdivisão.

dessas funções.

5.5.1 Procura_cubo_GA

Quando é feita a subdivisão de um cubo, independentemente de onde é posicionado o ponto de subdivisão, o resultado é o mesmo: oito cubos, doze faces e seis arestas.

A função *Procura_cubo_GA* tem o mesmo funcionamento da função *Procura_cubo* do *Dual Contouring*, onde $q1$ é o cubo a ser subdividido.

A função *Procura_cubo_GA* tem:

- Oito chamadas à *Procura_cubo_GA*.
- Doze chamadas à $Procura\_face\_GA$.
- Seis chamadas à $Procura\_aresta\_GA$.

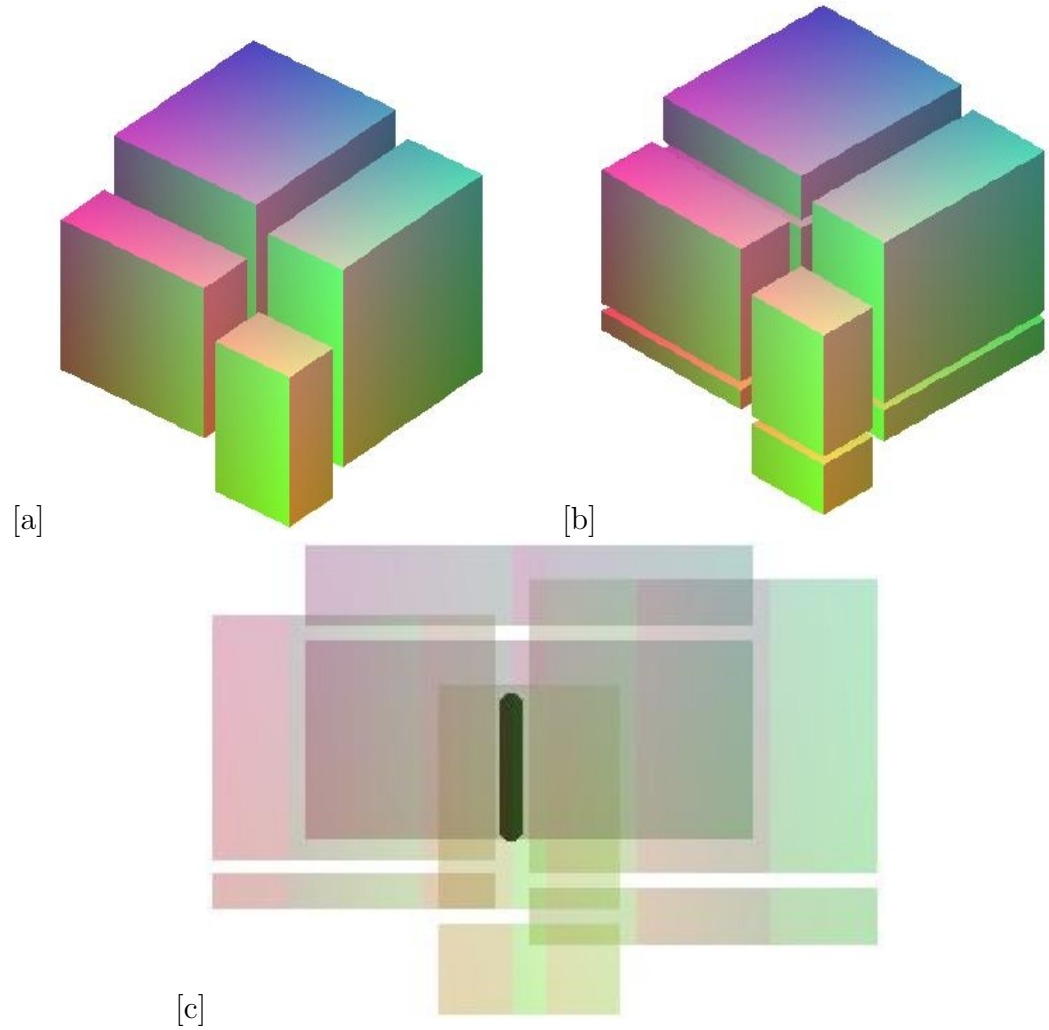


Figura 53: Uma nova aresta gerada pela subdivisão. a) Os quatro cubos originais. b) Subdivisão dos cubos. c) Vista de frente dos cubos com transparência mostrando a nova aresta.


```
Procura_cubo_GA(q1)  {
    If(q1 → p0 != NULL) {
        Procura_cubo_GA(q1 → p0);
        Procura_cubo_GA(q1 → p1);
        Procura_cubo_GA(q1 → p2);
        Procura_cubo_GA(q1 → p3);
        Procura_cubo_GA(q1 → p4);
        Procura_cubo_GA(q1 → p5);
        Procura_cubo_GA(q1 → p6);
```

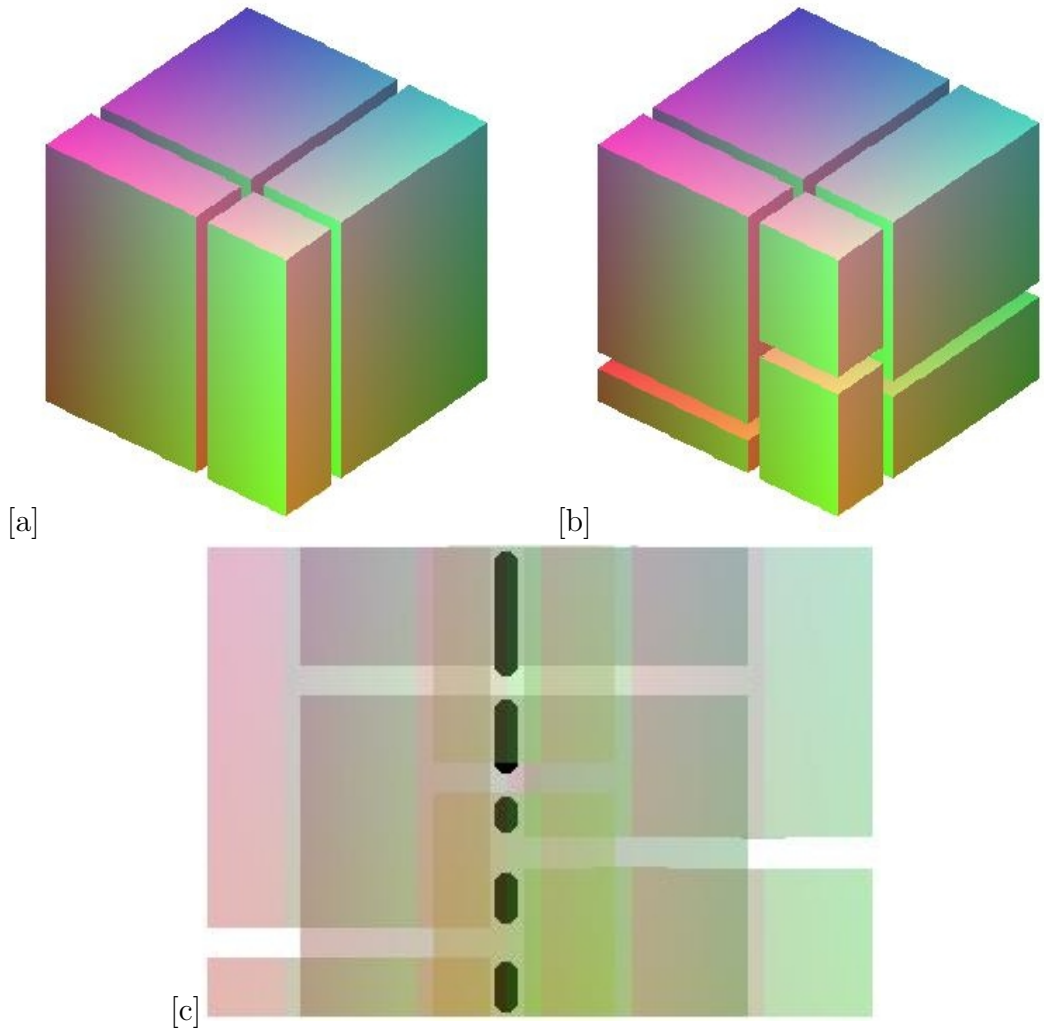


Figura 54: Cinco novas arestas geradas pela subdivisão. a) Os quatro cubos originais. b) Subdivisão dos cubos. c) Vista de frente dos cubos, com transparência, mostrando as cinco novas arestas.

$$Procura\_cubo\_GA(q1 \rightarrow p7);$$

$$Procura\_face\_GA(q1 \rightarrow p0, q1 \rightarrow p4);$$
$$Procura\_face\_GA(q1 \rightarrow p1, q1 \rightarrow p5);$$
$$Procura\_face\_GA(q1 \rightarrow p2, q1 \rightarrow p6);$$
$$Procura\_face\_GA(q1 \rightarrow p3, q1 \rightarrow p7);$$
$$Procura\_face\_GA(q1 \rightarrow p0, q1 \rightarrow p3);$$
$$Procura\_face\_GA(q1 \rightarrow p1, q1 \rightarrow p2);$$
$$Procura\_face\_GA(q1 \rightarrow p4, q1 \rightarrow p7);$$
$$Procura\_face\_GA(q1 \rightarrow p5, q1 \rightarrow p6);$$
$$Procura\_face\_GA(q1 \rightarrow p0, q1 \rightarrow p1);$$
$$Procura\_face\_GA(q1 \rightarrow p2, q1 \rightarrow p3);$$
$$Procura\_face\_GA(q1 \rightarrow p4, q1 \rightarrow p5);$$

$Procura\_face\_GA(q1 \rightarrow p6, q1 \rightarrow p7);$

$Procura\_aresta\_GA(q1 \rightarrow p0, q1 \rightarrow p1, q1 \rightarrow p4, q1 \rightarrow p5);$
$Procura\_aresta\_GA(q1 \rightarrow p2, q1 \rightarrow p3, q1 \rightarrow p6, q1 \rightarrow p7);$
$Procura\_aresta\_GA(q1 \rightarrow p0, q1 \rightarrow p3, q1 \rightarrow p4, q1 \rightarrow p7);$
$Procura\_aresta\_GA(q1 \rightarrow p1, q1 \rightarrow p2, q1 \rightarrow p5, q1 \rightarrow p6);$
$Procura\_aresta\_GA(q1 \rightarrow p0, q1 \rightarrow p1, q1 \rightarrow p2, q1 \rightarrow p3);$
$Procura\_aresta\_GA(q1 \rightarrow p4, q1 \rightarrow p5, q1 \rightarrow p6, q1 \rightarrow p7);$
}
}

5.5.2 Procura_Face_GA

Na definição original do algoritmo para encontrar arestas mínimas, quando pelo menos um dos dois cubos que se encontram em uma face é subdividido, o número de chamadas e os cubos utilizados para fazer as chamadas às funções *Procura_Face* e *Procura_Aresta* já estão definidas. No caso do algoritmo apresentado nessa tese, a quantidade de chamadas à função $Procura\_Face\_GA$ varia de um a nove, e à $Procura\_Aresta\_GA$ varia de zero a doze.

Para encontrar a quantidade e quais são as faces resultantes do encontro de cubos dois a dois, é utilizada a função auxiliar $Encontra\_Face\_Face\_GA$. Nesta função $q1$ e $q2$ são os cubos que se encontram nessa face, $cubos\_face$ e $cubos\_aresta$ guardam os cubos resultantes dessa subdivisão que se encontram nessa face e nessa aresta, $sequencia\_face$ e $sequencia\_aresta$ guarda a ordem em que esses cubos se encontram e finalmente num_faces e $num\_arestas$ guardam o número de cubos que se encotram nessa face e arestas.

Procura_face_GA(q1,q2) {
$If((q1 \rightarrow p0 != NULL)||(q2 \rightarrow p0 != NULL))$ {
$Encontra\_Face\_Face\_GA(q1, q2, cubos\_face, cubos\_aresta,$
$sequencia\_face, sequencia\_aresta, num\_faces, num\_arestas);$
$if(q1 \rightarrow p0 != NULL)$ {
$cubo[0][0] = q1 \rightarrow p[cubos\_face[0][0]];$
$cubo[0][1] = q1 \rightarrow p[cubos\_face[0][1]];$

```
            cubo[0][2] = q1 → p[cubos_face[0][2]];
            cubo[0][3] = q1 → p[cubos_face[0][3]];
        }else {
            cubo[0][0] = cubo[0][1] = cubo[0][2] = cubo[0][3] = q1;
        }

        if(q2 → p[0] != NULL) {
            cubo[1][0] = q2 → p[cubos_face[1][0]];
            cubo[1][1] = q2 → p[cubos_face[1][1]];
            cubo[1][2] = q2 → p[cubos_face[1][2]];
            cubo[1][3] = q2 → p[cubos_face[1][3]];
        }else {
            cubo[1][0] = cubo[1][1] = cubo[1][2] = cubo[1][3] = q2;
        }

        for(i = 0; i < num_faces; i++)
            Procura_Face_GA(cubo[0][sequencia_face[0][i]],
                            cubo[1][sequencia_face[1][i]]);

        for(i = 0; i < num_arestas; i++)
            Procura_Aresta_GA(cubo[0][sequencia_aresta[0][i]][0],
                              cubo[0][sequencia_aresta[0][i]][1],
                              cubo[1][sequencia_aresta[1][i]][0],
                              cubo[1][sequencia_aresta[1][i]][1]);

    }
}
```


5.5.3 Procura_Aresta_GA

Na definição original do algoritmo para encontrar arestas mínimas, quando pelo menos um dos dois cubos que se encontram em uma aresta é subdividido, são geradas duas chamadas à função *Procura_Aresta* e os cubos para cada uma dessas duas chamadas já estão definidas. No caso do algoritmo apresentado nesta tese, o número de chamadas para *Procura_Aresta_GA* varia de um a cinco. Para encontrar a quantidade e quais os cubos

que estão em cada uma das novas arestas é utilizada a função auxiliar $Encontra\_Aresta\_Aresta\_GA$.

A função $Encontra\_Aresta\_Aresta\_GA$ tem como entrada os quatro cubos que se encontram em uma aresta, $q1$, $q2$, $q3$ e $q4$, retorna os cubos resultantes da subdivisão que se encontram na região de interseção desses cubos ($cubos\_aresta$), a sequência ordenada das subdivisões dos cubos em relação a essa aresta ($sequencia\_trocas$), o primeiro cubo resultante da subdivisão na interseção ($cubo\_inicial$) e o número de subdivisões que tem a aresta ($num\_arestas$).

```
Procura_Aresta_GA(q1, q2, q3, q4) {
  if((q1 → p0 != NULL)||(q2 → p0 != NULL)
      ||(q3 → p0 != NULL)||(q4 → p0 != NULL)) {
    Encontra_Aresta_Aresta_GA(q1, q2, q3, q4, cubos_aresta,
        sequencia_trocas, cubo_inicial, num_arestas);
    if(q1 → p0 != NULL) {
      q[0][0] = q1 → p[cubos_aresta[0][0]];
      q[0][1] = q1 → p[cubos_aresta[0][1]];
    }else {
      q[0][0] = q[0][1] = q1;
    }

    if(q2 → p0 != NULL) {
      q[1][0] = q2 → p[cubos_aresta[1][0]];
      q[1][1] = q2 → p[cubos_aresta[1][1]];
    }else {
      q[1][0] = q[1][1] = q2;
    }

    if(q3 → p0 != NULL) {
      q[2][0] = q3 → p[cubos_aresta[2][0]];
      q[2][1] = q3 → p[cubos_aresta[2][1]];
    }else {
      q[2][0] = q[2][1] = q3;
    }

    if(q4 → p0 != NULL) {
```

```
            q[3][0] = q4 → p[cubos_aresta[3][0]];
            q[3][1] = q4 → p[cubos_aresta[3][1]];
        }else {
            q[3][0] = q[3][1] = q4;
        }

        pont[0] = q[0][cubo_inicial[0]];
        pont[1] = q[1][cubo_inicial[1]];
        pont[2] = q[2][cubo_inicial[2]];
        pont[3] = q[3][cubo_inicial[3]];
        for(i = 0; i < num_arestas; i++)
            Procura_aresta_GA(pont[0], pont[1], pont[2], pont[3]);
            pont[sequencia_trocas] = q[sequencia_trocas][1];

    }
}
```

6
Resultados

Nesse capítulo são apresentados os resultados da poligonalização de superfícies implícitas com o método proposto nesta tese. São comparados o número de polígonos que formam a superfície e o número de cubos que a intersectam obtidos com este método e com o *Dual Contouring*. Como a Grade Adaptativa faz a subdivisão mais próxima à superfície, um número maior de cubos intersectam a superfície gerando mais pontos da malha e mais polígonos do que o *Dual Contouring*, que posiciona o ponto de subdivisão sempre no centro do cubo, independente da posição da superfície.

6.1
Octante de uma Esfera

Nesta seção é acompanhada a poligonalização do octante de uma esfera, desde uma *octree* com um nível até uma *octree* com três níveis de subdivisão. Como esse octante engloba somente um dos vértices do cubo de poligonalização, dos oito vértices desse cubo sete deles estão na região exterior à superfície, gerando três arestas com mudança de sinal e com isso permitindo que o posicionamento do ponto de subdivisão seja próximo à superfície desde o primeiro nível de subdivisão.

6.1.1
Octree com um nível de subdivisão

Na figura 55 é mostrado o octante da esfera gerada por uma *octree* com dois níveis. Nas figuras 55 a) e b) o ponto branco é utilizado para marcar o centro do cubo de poligonalização e as grades em vermelho, azul e verde são os planos xy, xz e yz, que passam na origem e estão no primeiro octante. As linhas sobre a superfície limitam os polígonos que geram essa superfície. A figura 55 b) mostra a superfície poligonalizada com a Grade Adaptativa e a figura 55 a) mostra a poligonalização com o *Dual Contouring*. Em ambos os casos a superfície é poligonalizada com três polígonos. Apesar de os dois métodos de poligonalização gerarem uma superfície com o mesmo número de polígonos, é

possível ver na figura 55 d) que a subdivisão da *octree* não está no seu centro. A superfície gerada pela Grade Adaptativa, mesmo tendo somente um nível de subdivisão, possui um formato bem mais próximo do octante de uma esfera do que a poligonalização efetuada com o *Dual Contouring*, mostrada na figura 55 c). É possível observar na figura 55 c) e d) que em ambas as poligonalizações somente um dos cubos não intersecta a superfície.

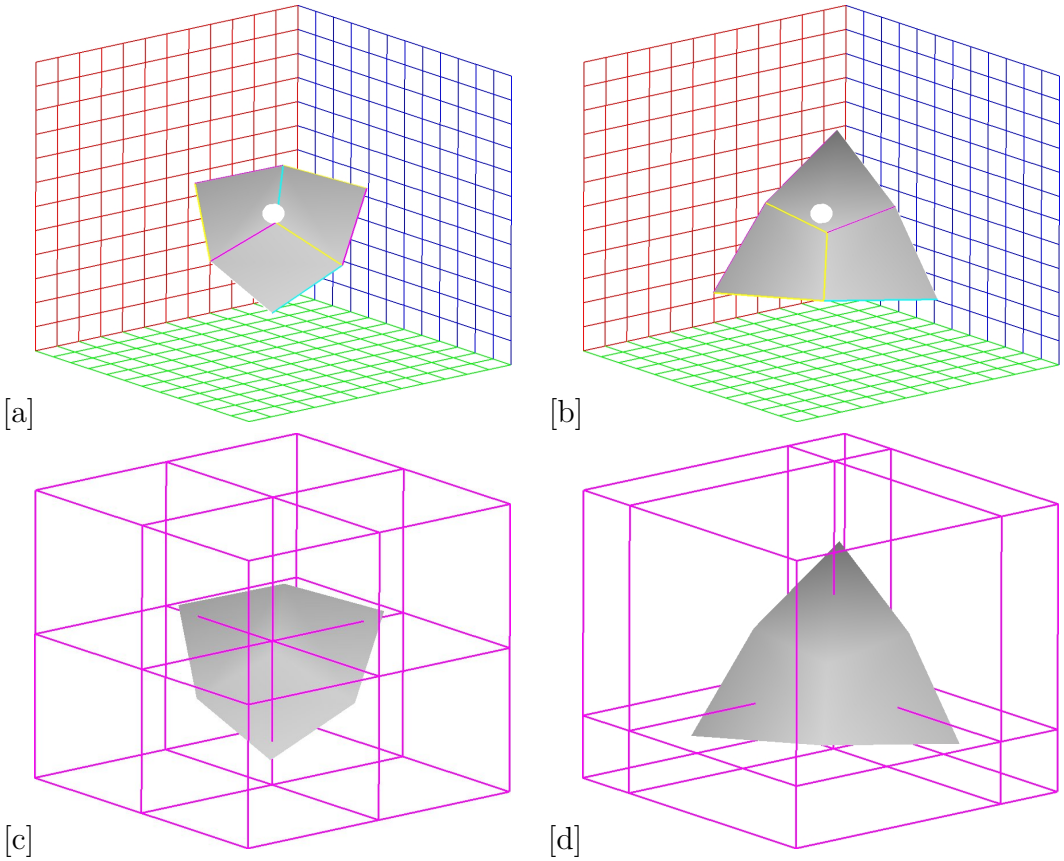


Figura 55: Octante de uma esfera poligonalizada por uma *octree* com um nível de subdivisão. a) *Dual Contouring* com os polígonos. b) Grade Adaptativa com os polígonos. c) *Dual Contouring* com a *octree*. d) Grade Adaptativa com a *octree*.

6.1.2
Octree com Dois Níveis de Subdivisão

Na figura 56 b) é possível ver o octante da esfera poligonalizada pela Grade Adaptativa e na figura 56 a) está a superfície poligonalizada pelo *Dual Contouring*. É possível ver que o bordo da superfície poligonalizada pela Grade Adaptativa é mais regular, e portanto mais próximo do octante de uma esfera do que a superfície poligonalizada pelo *Dual Contouring*.

Em 57 b) e d) é mostrada a superfície poligonalizada com a Grade Adaptativa e em 57 a) e c) é mostrada a poligonalização com o *Dual Contouring*.

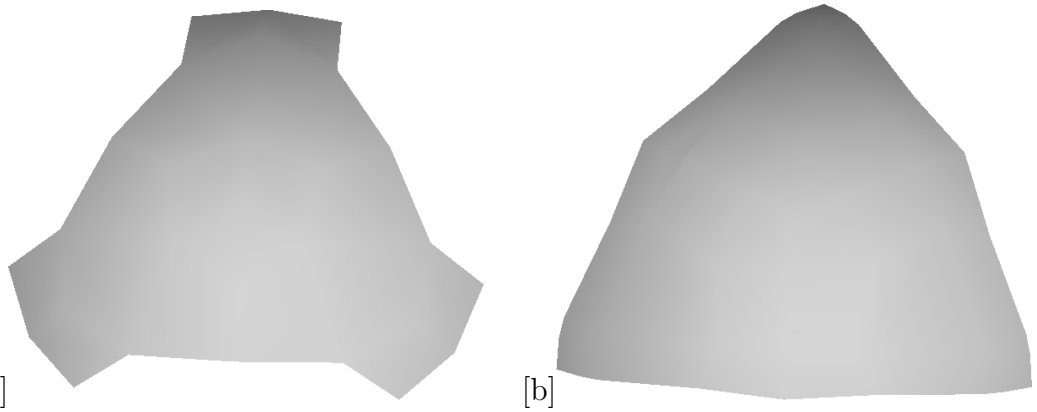


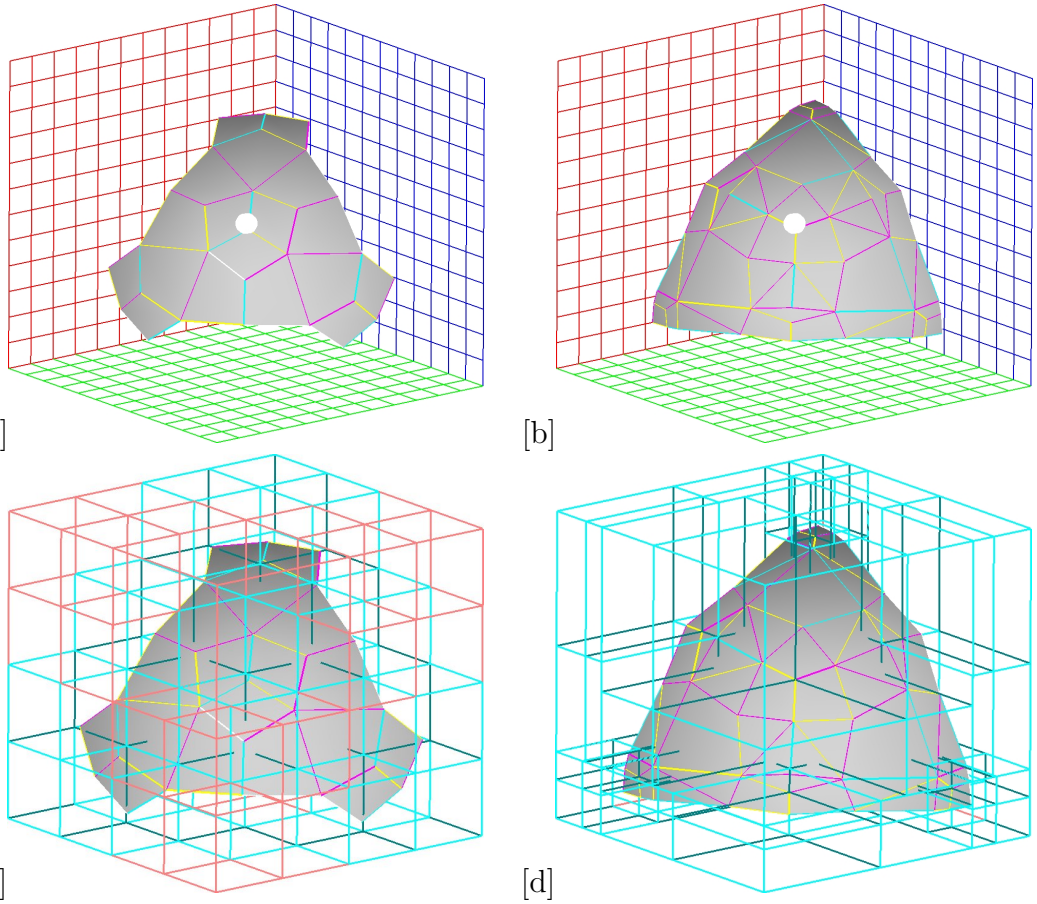
Figura 56: Octante de uma esfera poligonalizada por uma *octree* com dois níveis de subdivisão. a) *Dual Contouring*. b) Grade Adaptativa.

É possível ver o posicionamento dos cubos da subdivisão e os polígonos que geram essas superfície. Nesse caso, a Grade Adaptativa gera a superfície com 48 polígonos, enquanto a superfície poligonalizada com o *Dual Contouring* tem 18 polígonos. Em dois níveis de subdivisão é possível analisar alguns aspectos da *octree* gerada pela Grade Adaptativa, como por exemplo quantos, e quais, cubos intersectam a superfície. Nas figuras 57 c) e d) os cubos em verde intersectam a superfície e os de cor de abóbora não intersectam. Ao poligonalizar a superfície com a Grade Adaptativa 49 cubos a intersectam, enquanto no *Dual Contouring* apenas há interseção para 28 cubos.

6.1.3
Octree com Três Níveis de Subdivisão

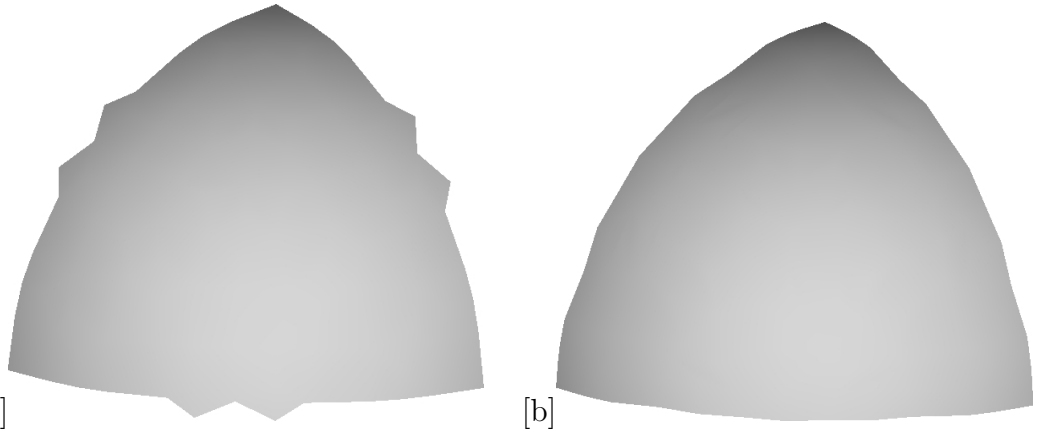
As figuras 58 a) e b) mostram um octante de uma esfera. Em a) a poligonalização é feita com o *Dual Contouring* e em b) o método utilizado é a Grade Adaptativa. O bordo da superfície poligonalizada com a Grade Adaptativa é mais regular do que o bordo correspondente à poligonalização com o *Dual Contouring*, como é possível ver em 58 a) que mostra a existência de bicos.

O bordo da superfície poligonalizada com a Grade Adaptativa é mais regular pois, além do melhor posicionamento dos cubos da *octree*, são gerados mais polígonos. A figura 59 b) mostra os polígonos que formam a superfície poligonalizada pela Grade Adaptativa. Nesse caso são gerados 282 polígonos contra os 84 polígonos da superfície poligonalizada com o *Dual Contouring*, que podem ser vistos em 59 a). As figuras 59 c) e d) mostram os cubos que intersectam a superfície, em verde, e as que não intersectam, em cor de abóbora. Na figura 59 d), que foi poligonalizada com a Grade Adaptativa, 217 cubos intersectam a superfície, enquanto a figura 59 c) mostra os cubos gerados com



[a] [b] [c] [d]

Figura 57: Octante de uma esfera poligonalizada por uma *octree* com dois níveis de subdivisão. a) *Dual Contouring* com os polígonos. b) Grade Adaptativa com os polígonos. c) *Dual Contouring* com a *octree*. d) Grade Adaptativa com a *octree*.



[a] [b]

Figura 58: Octante de uma esfera poligonalizada por uma *octree* com três níveis de subdivisão. a) Poligonalização com o *Dual Contouring*. b) Poligonalização com a Grade Adaptativa.

o *Dual Contouring* e somente 103 desses cubos intersectam a superfície.

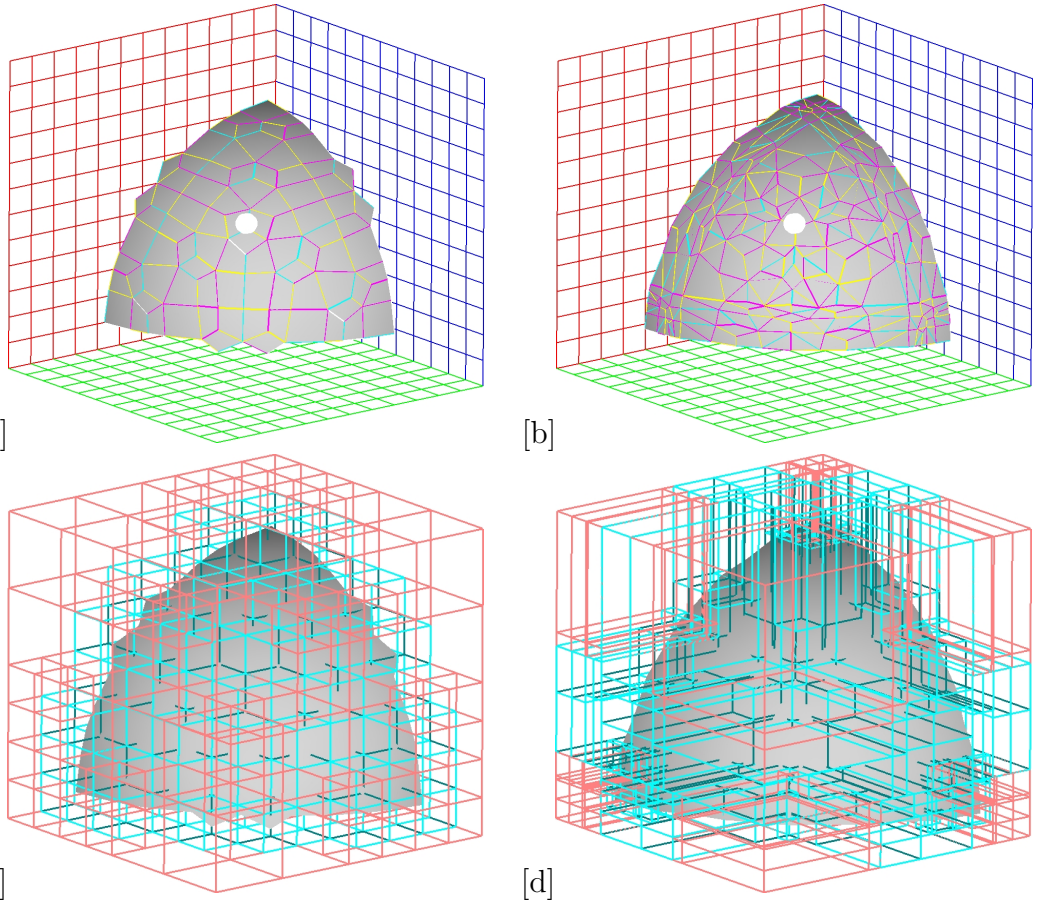


Figura 59: Octante de uma esfera poligonalizada por uma *octree* com três níveis de subdivisão. a) *Dual Contouring* com os polígonos. b) Grade Adaptativa com os polígonos. c) *Dual Contouring* com a *octree*. d) Grade Adaptativa com a *octree*.

6.2
Esfera com Seis Níveis de Subdivisão

As figuras 60 a) e b) mostram uma esfera poligonalizada com seis níveis de poligonalização com os métodos *Dual Contouring* e Grade Adaptativa respectivamente. A esfera está totalmente contida no cubo de poligonalização, por isso os oito vértices do cubo estão no exterior da esfera. Portanto não existem vértices com sinais distintos, e como consequência o posicionamento do primeiro ponto de subdivisão ocupa o centro do cubo. Nas figuras 60 c) e d) são mostrados os polígonos que geram as esferas: 40206 polígonos geram a superfície com a Grade Adaptativa, figura 60 d), enquanto 13232 polígonos geram a superfície com o *Dual Contouring*, figura 60 c).

A figura 61 mostra os cubos da *octree*, em lilás estão todos os cubos da *octree*, figuras a) e b). A figura 61 b) foi poligonalizada com a Grade Adaptativa. É possível ver pela densidade das arestas que formam os cubos que foi gerado um número muito maior de cubos do que com o *Dual Contouring*,

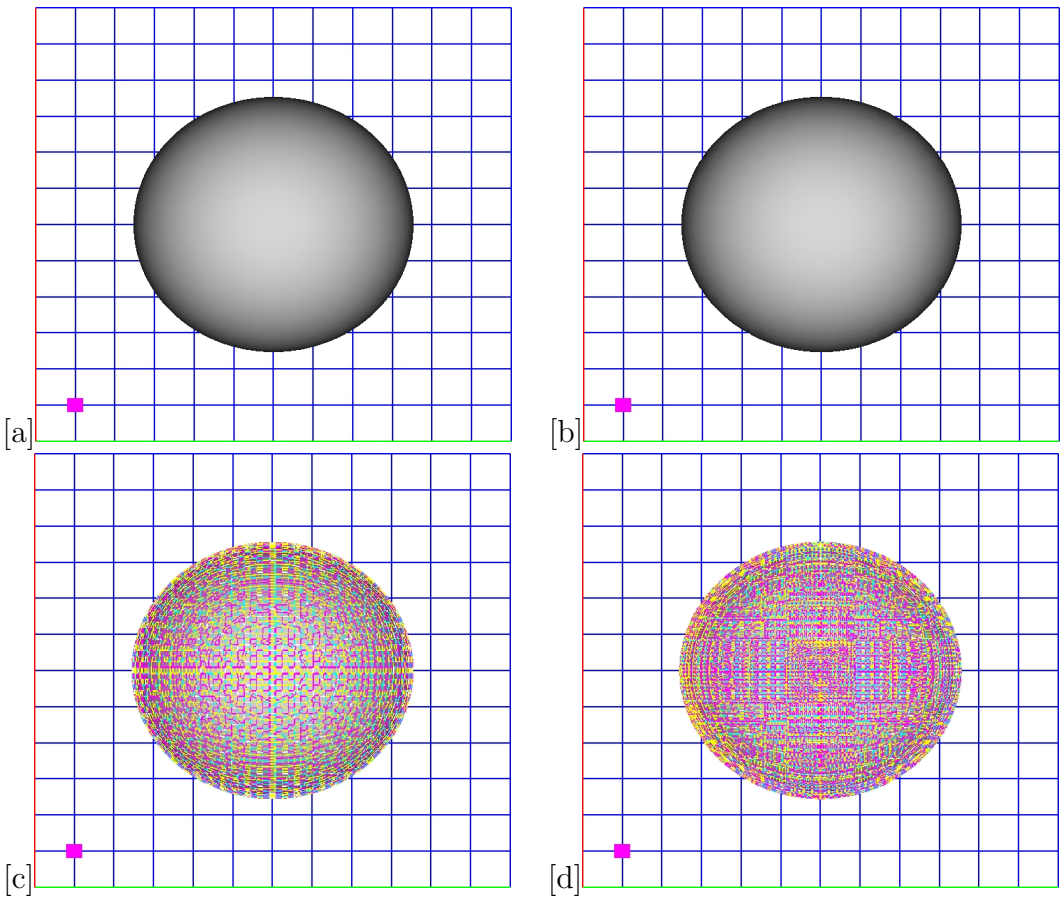


Figura 60: Esfera poligonalizada por uma *octree* com seis níveis de subdivisão. a) Poligonalização com o *Dual Contouring*. b) Poligonalização com a Grade Adaptativa. c) *Dual Contouring* com os polígonos que formam a superfície. d) Grade Adaptativa com os polígonos que formam a superfície.

figura 61 a). Isso se deve ao melhor posicionamento dos cubos da *octree*.

As figuras 61 c) e d) mostram os mesmos cubos das figuras a) e b), respectivamente. Os cubos estão pintados de acordo com a sua situação em relação à superfície: em cor de abóbora, os que não intersectam a superfície e em verde os que a intersectam. Na figura 61 d), poligonalizada com a Grade Adaptativa, 33296 cubos intersectam a superfície, enquanto na figura 61 c), poligonalizada com o *Dual Contouring*, 13232 cubos intersectam a superfície.

6.3 Toro com Seis Níveis de Poligonalização

Nesse exemplo é mostrada poligonalização de um toro com seis níveis de subdivisão. A poligonalização com a Grade Adaptativa é mostrada nas figuras 62 b) e d), enquanto as figuras 62 a) e c) mostram a poligonalização com o *Dual Contouring*.

Apesar de serem muito parecidas as superfícies em a) e b), a superfície

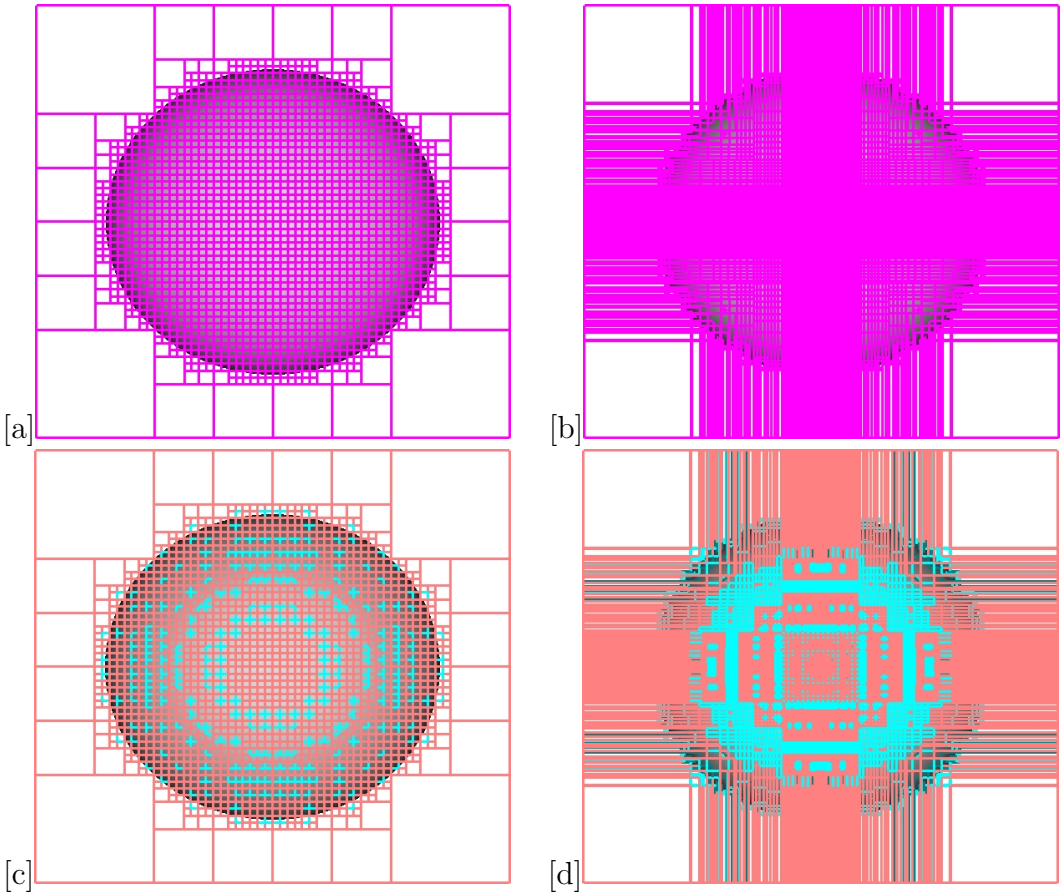


Figura 61: Esfera poligonalizada por uma *octree* com seis níveis de subdivisão. a) *Dual Contouring* – Todos os cubos . b) Grade Adaptativa – Todos os cubos. c) *Dual Contouring* com os cubos que intersectam e os que não intersectam a superfície. d) Grade Adaptativa com os cubos que intersectam e os que não intersectam a superfície.

associada à Grade Adaptativa foi gerada com 45042 polígonos, enquanto aquela associada ao *Dual Contouring* tem 13232. Portanto, com o mesmo número de subdivisões da *octree*, o método apresentado nessa tese gerou mais de três vezes o número de polígonos gerado com o *Dual Contouring*, conforme exibido nas figuras c) e d).

Essa diferença no número de polígonos que formam a superfície se deve ao fato de que mais cubos da *octree* intersectam a superfície utilizando a Grade Adaptativa, que neste caso são 30688, contra os 13232 que intersectam a superfície com o *Dual Contouring*.

6.4 Rampa Senoidal

Outro aspecto interessante da Grade Adaptativa é que por ser ela mais próxima à superfície e haver mais cubos que intersectam a superfície, os

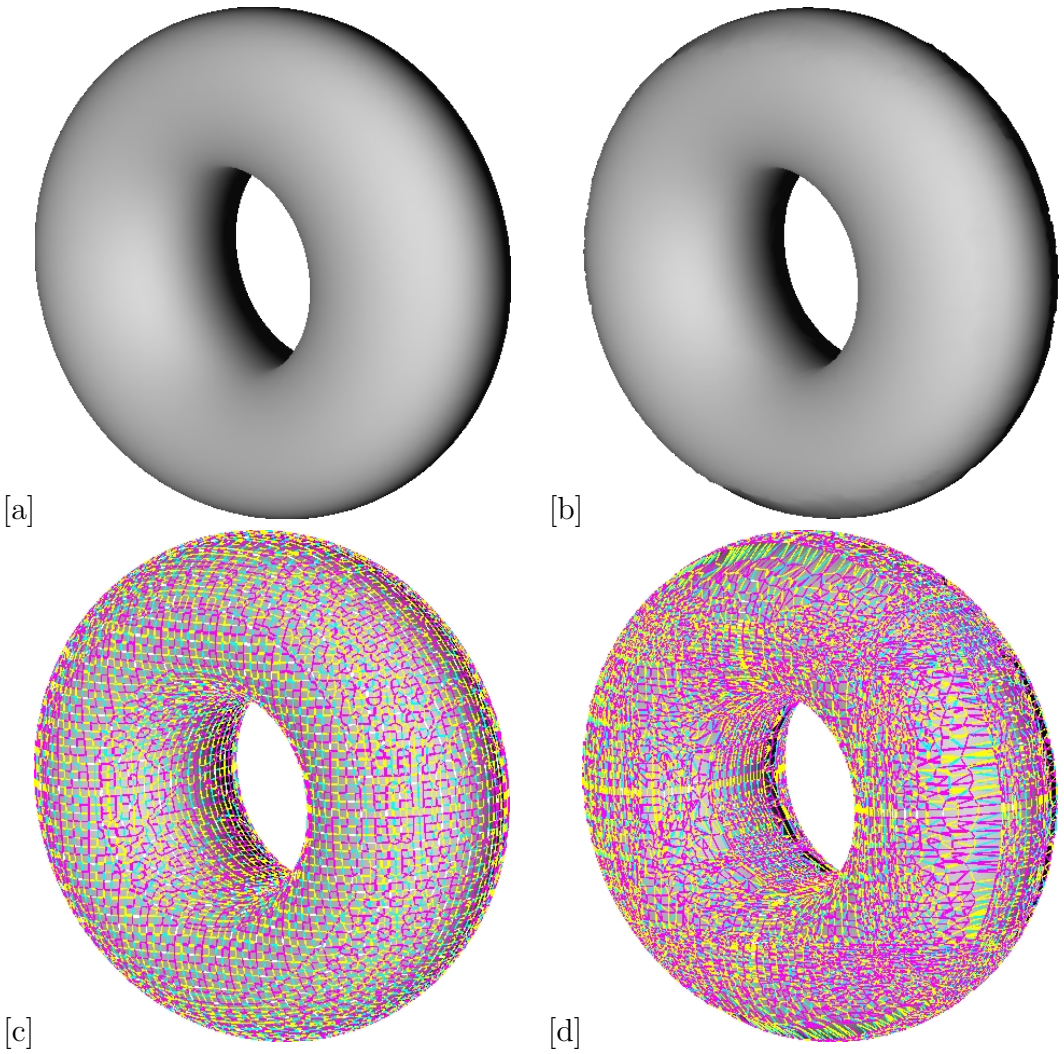


Figura 62: Toro poligonalizado por uma *octree* com seis níveis de subdivisão. a) Poligonalização com o *Dual Contouring*. b) Poligonalização com a Grade Adaptativa. c) *Dual Contouring* com os polígonos que formam a superfície. d) Grade Adaptativa com os polígonos que formam a superfície.

detalhes são reconstruídos com mais precisão, mesmo para um número pequeno de subdivisões. A seguir é mostrada uma superfície com a forma de uma rampa com ondulações. Inicialmente ela é poligonalizada com três níveis de subdivisão e a seguir com cinco níveis.

6.4.1
Rampa Senoidal com Três Níveis de Subdivisão

Neste caso a poligonalização é claramente distinta, como pode ser visto na figura 63 b), poligonalizada com a Grade Adaptativa e que, mesmo para um número muito pequeno de subdivisões, mostra o comportamento ondulatório da superfície. Na poligonalização com o *Dual Contouring*, figura 63 a), as ondulações da superfície não são tão visíveis como na figura 63 b).

As figuras 63 c) e d) mostram vistas laterais da superfície poligonalizada com a Grade Adaptativa e o *Dual Contouring* respectivamente. Em d) é possível ver as ondulações da superfície, enquanto a figura c) tem o comportamento mais próximo de uma parábola, sem as ondulações que essa superfície possui.

A Grade Adaptativa gera 160 polígonos, contra as 98 geradas com o *Dual Contouring*. Além disso, 168 cubos da Grade Adaptativa intersectam a superfície, enquanto para o *Dual Contouring* são 120 cubos.

Este exemplo ilustra que, mesmo com um número pequeno de subdivisões, a Grade Adaptativa capta mais detalhes da superfície ao passo que o *Dual Contouring* necessita de mais níveis para revelar tais detalhes.

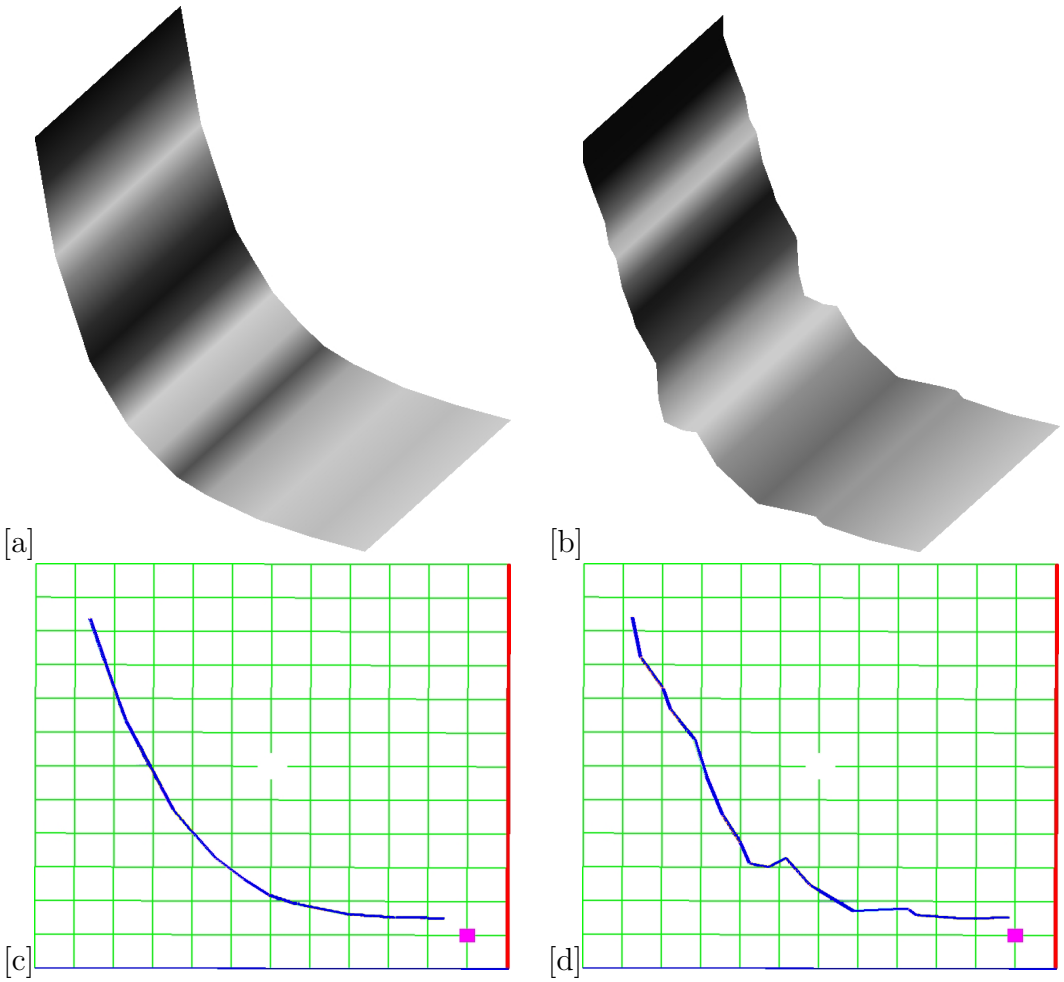


Figura 63: Rampa senoidal poligonalizada por uma *octree* com três níveis de subdivisão. a) Poligonalização com o *Dual Contouring*. b) Poligonalização com a Grade Adaptativa. c) *Dual Contouring*, vista lateral. d) Grade Adaptativa, vista lateral.

6.4.2
Rampa Senoidal com Cinco Níveis de Subdivisão

Com cinco níveis de subdivisão, tanto a superfície gerada pela Grade Adaptativa, figuras 64 b) e d), quanto a gerada pelo *Dual Contouring*, figuras 64 a) e c), captam as ondulações da superfície. Em c) e d) é mostrada a vista lateral dessa superfície, onde é possível ver que nesse caso elas estão muito próximas, pois o número maior de subdivisões permite que o *Dual Contouring* capte as ondulações dessa superfície.

Com essas cinco subdivisões a Grade Adaptativa gera 4285 polígonos, contra os 2201 gerados pelo *Dual Contouring*, e o número de cubos da *octree* que intersectam a superfície também é distinto, 3840 da Grade Adaptativa contra os 2304 do *Dual Contouring*.

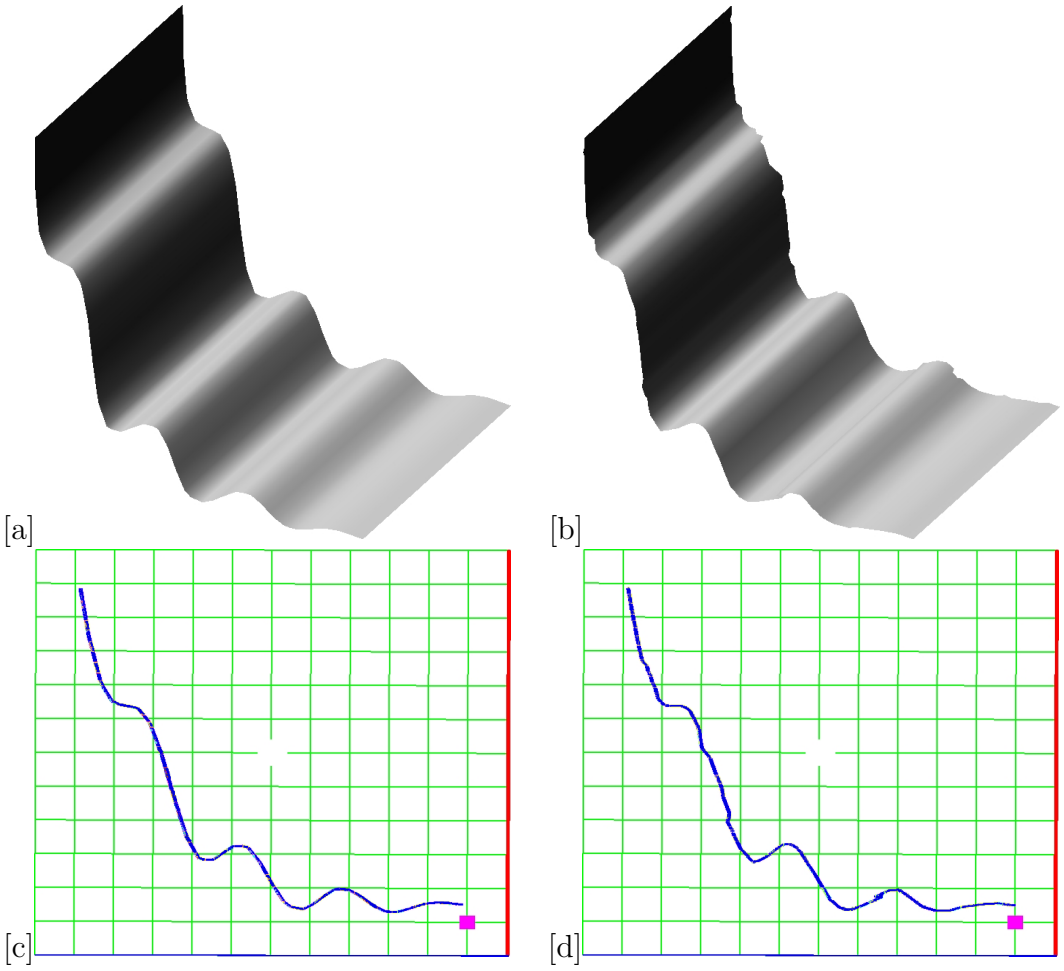


Figura 64: Rampa senoidal poligonalizada por uma *octree* com cinco níveis de subdivisão. a) Poligonalização com o *Dual Contouring*. b) Poligonalização com a Grade Adaptativa. c) *Dual Contouring*, vista lateral. d) Grade Adaptativa, vista lateral.

6.5 Superfície Senoidal

Este capítulo se encerra com um exemplo que mostra mais claramente como a Grade Adaptativa pode captar detalhes da superfície que o *Dual Contouring* só consegue com um número bem maior de subdivisões.

A figura 65 apresenta o resultado da poligonalização de uma superfície definida implicitamente a partir de uma função seno, com três níveis de subdivisão da *octree*. Essa superfície está situada na porção inferior do cubo de poligonalização. Ao gerar a superfície com o *Dual Contouring* ela não é reconstruída corretamente, gerando um plano, como pode ser visto nas figuras 65 a) e b). Ao utilizar a Grade Adaptativa a posição da superfície é levada em conta nas subdivisões dos cubos, o que resulta em um maior número de cubos que intersectam a superfície e consequentemente a uma melhor reconstrução da superfície: são geradas as ondas características da modelagem de uma superfície senoidal, conforme exibido nas figuras 65 c) e d).

Para a superfície mostrada na figura 65, a Grade Adaptativa gera 140 polígonos contra os 49 polígonos gerados pelo *Dual Contouring*. Além disso, na Grade Adaptativa 192 cubos intersectam a superfície, enquanto no *Dual Contouring* somente 64 cubos a intersectam, gerando um resultado mais preciso com o mesmo número de subdivisões.

6.6 Métrica

Para comparar os resultados apresentados nesta tese foi definida uma métrica. Essa métrica mede a distância entre os pontos aproximados da superfície implícita e o ponto exato da superfície. Os pontos aproximados são gerados pelo método apresentado nesta tese e pelo *Dual Contouring*.

Para medir a distância foi utilizado o vetor gradiente. Esse vetor tem a característica de apontar para a direção de maior crescimento da função, portanto, ao seguir a direção desse vetor um ponto aproximado da superfície alcança mais rapidamente o ponto exato da superfície. Caso o ponto aproximado esteja no interior da superfície, o vetor gradiente é percorrido na sua direção crescente, caso contrário, a direção a ser seguida é a decrescente.

Na figura 66 é possível ver, em vermelho, os pontos que geram a aproximação do octante de uma esfera. Nesse caso os pontos estão no interior da esfera, e os vetores gradientes, que estão em azul, apontam para os pontos exatos da esfera, que estão no seu exterior. É possível observar que esses vetores gradientes são normais à superfície.

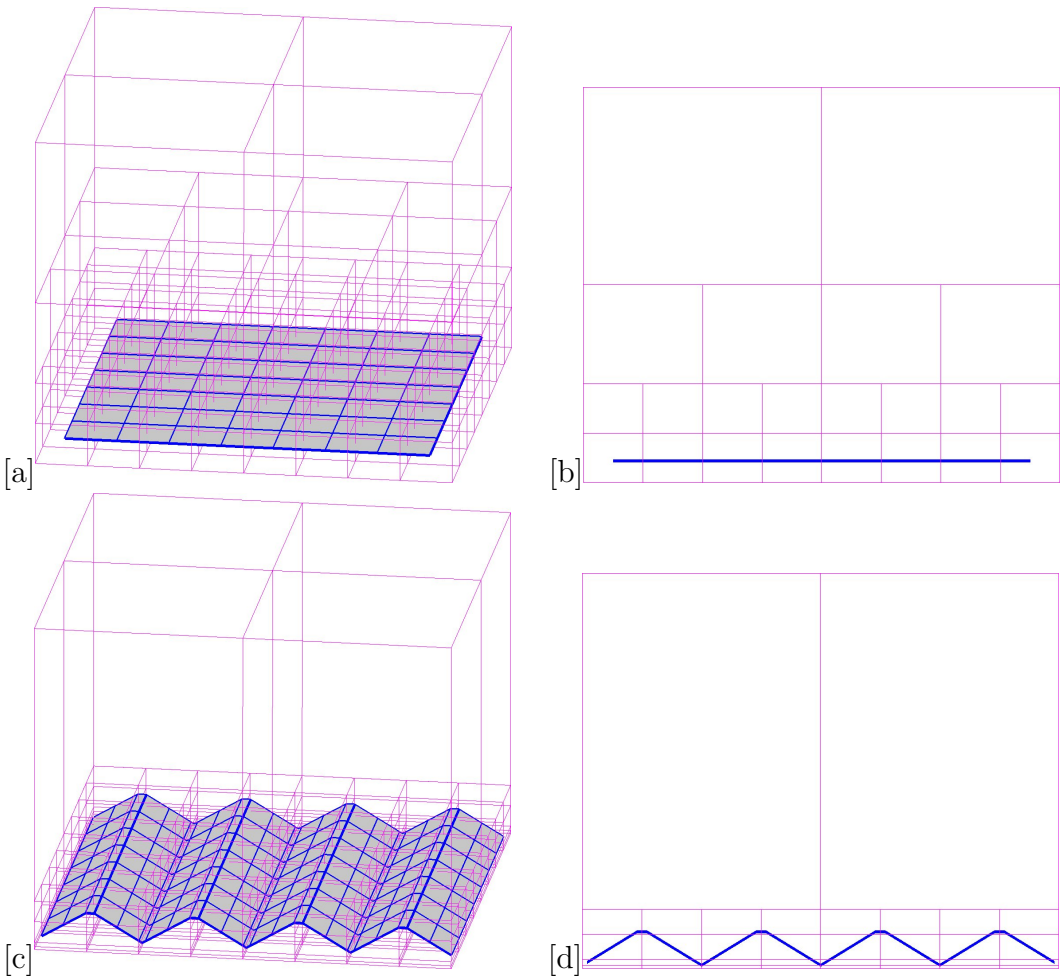
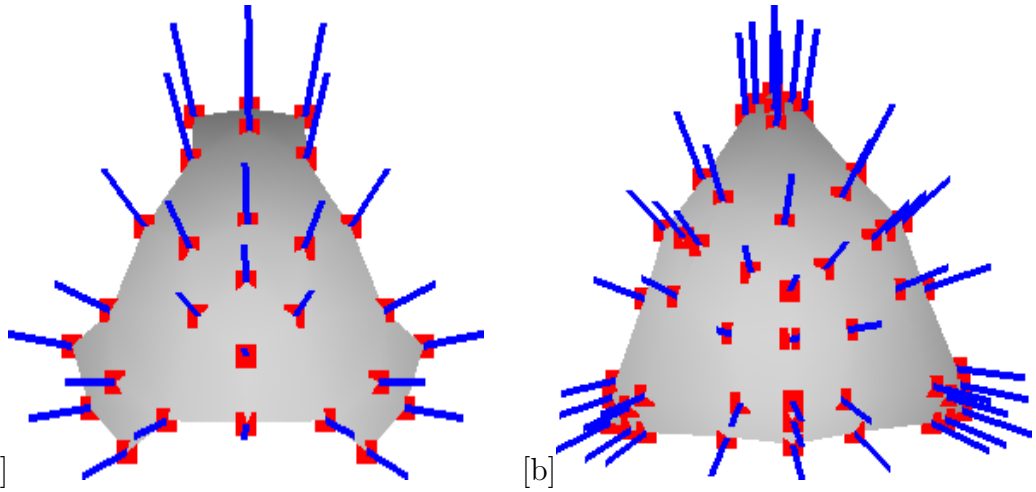


Figura 65: Poligonalização de uma superfície senoidal. a)*Dual Contouring*, vista em perspectiva. b) *Dual Contouring*, vista frontal. c) Grade Adaptativa, vista em perspectiva. d) Grade Adaptativa, vista frontal.

Inicialmente, o vetor gradiente é percorrido para encontrar um ponto que esteja no lado oposto ao do ponto aproximado que gera a superfície. A partir destes dois pontos, o aproximado e o que está no lado oposto, é feita uma aproximação dividindo o segmento de reta ao meio. É calculado o valor deste ponto pela função implícita. Este ponto passa a ser o novo extremo do segmento de reta, se ele for positivo toma o lugar do extremo positivo anterior, e caso o valor seja negativo toma o lugar do extremo negativo. Com isso, o novo segmento tem metade do comprimento do segmento anterior e os pontos do extremo estão convergindo para o ponto exato.

Este processo pode ser repetido até que o valor da função no ponto médio seja menor do que um erro em torno do valor procurado da função, ou a distância entre os pontos seja menor do que um valor dado, ou até um número máximo de iterações.

Nas figuras 66 a) e b) é possível ver um octante de esfera gerado com duas



[a] [b]

Figura 66: Octante de uma esfera poligonalizada com dois níveis de recursão. Em vermelho estão os pontos aproximados, que geram a superfície, e em azul estão os segmentos de reta que ligam estes pontos aos pontos exatos mais próximos. a) *Dual Contouring*. b) Grade Adaptativa.

subdivisões da *octree*, pelo método *Dual Contouring* e pelo método apresentado nesta tese, respectivamente. É possível observar que os vetores gradientes são perpendiculares à esfera. Ao fazer os pontos percorrerem esses vetores, são alcançadas mais rapidamente outras esferas com raios maiores, ou menores, do que a esfera original.

A métrica é definida como a soma das distâncias entre o ponto da superfície aproximada e o seu respectivo ponto exato, e esse valor é dividido pelo número de vértices da malha. Essa divisão é necessária, pois o método apresentado nesta tese pode gerar mais vértices da malha do que o método *Dual Contouring*.

6.7 Análise dos Resultados

A Grade Adaptativa utiliza uma *octree*, e por fazer um melhor posicionamento do ponto de subdivisão aproveita melhor os cubos que podem ser gerados nesta *octree*. O número máximo de operações para posicionar o ponto de subdivisão é o número de iterações para o reposicionamento deste ponto utilizando o vetor gradiente, este número de iterações pode ser fixo ou definido pelo usuário. Portanto o número de operações para o posicionamento do ponto de subdivisão é um valor constante k.

O número máximo de folhas de uma *octree* com profundidade n é 8^n, que ocorre no caso de uma *octree* cheia. Como é posicionado um vértice da malha em cada folha que intersecta a superfície, então podem ser gerados no

máximo 8^n vértices da malha. A geração da *octree* pela Grade Adaptativa ou pelo *Dual Contouring* tem a mesma complexidade computacional, pois o custo de geração da Grade Adaptativa é a constante k vezes o custo de geração da *octree* pelo *Dual Contouring*.

Como a Aresta Mínima é definida pelo encontro de três ou quatro cubos, o número máximo de Arestas Mínimas desta *octree* é dado pelo número de combinações de cubos três a três, $C_4^{8^n}$. A diferença entre a Grade Adaptativa e o *Dual Contouring* é que o método apresentado nesta tese tem um melhor aproveitamento dos cubos gerados pela *octree*, se aproximando mais de uma *octree* cheia do que o *Dual Contouring*, mas a complexidade do algoritmo para encontrar as arestas mínimas é a mesma. Portanto a Grade Adaptativa tem a mesma complexidade computacional do *Dual Contouring*.

Para encerrar este capítulo é mostrada a tabela 1, que compara o número de vértices, polígonos e a métrica das superfícies mostradas nos exemplos anteriores. Nesta tabela, DC se refere ao *Dual Contouring* e GA se refere a Grade Adaptativa.

Pela tabela 1 é possível ver que para um número pequeno de subdivisões a diferença do número de vértices e polígonos gerados pelo *Dual Contouring* e pela Grade Adaptativa não é muito grande, mas conforme essa diferença cresce o número de subdivisões aumenta.

	Vértices		Polígonos		Métrica		
Superfície	DC	GA	DC	GA	DC	GA	GA/DC
Octante 1 sub	7	7	3	3	0.06455	0.13469	2.087
Octante 2 sub	28	49	18	48	0.01773	0.01696	0.956
Octante 3 sub	103	217	84	282	0.00480	0.00435	0.906
Esfera 6 sub	13232	33296	13232	40206	0.00018	0.00006	0.323
Toro 6 sub	13232	30688	13232	45042	0.00029	0.00028	0.952
Rampa 3 sub	120	168	98	160	0.03527	0.02604	0.738
Rampa 5 sub	2304	3840	2201	4285	0.00859	0.00513	0.597

Tabela 1: Tabela com o número de vértices, polígonos e a métrica das superfícies dos exemplos anteriores

Os valores obtidos para os dois métodos indicam um desempenho sensivelmente melhor para o método desenvolvido nessa tese, com exceção para o caso do octante gerado com uma única subdivisão da *octree*, o qual não tem interesse prático.

7
Conclusões e Trabalhos Futuros

Neste trabalho é apresentado um método de poligonalização adaptativa de superfícies que utiliza a estrutura de dados *octree*. Este método, ao contrário dos métodos clássicos, leva em conta a posição da superfície para gerar a poligonalização, resultando uma grade mais próxima à superfície.

A poligonalização mais próxima à superfície é obtida graças a um melhor posicionamento dos pontos de subdivisão da *octree*, utilizando o valor da função na região de poligonalização e o vetor gradiente. Este posicionamento deve ser de tal forma que o ponto de subdivisão não esteja nem sobre a superfície nem demasiado próximo de uma das faces ou arestas do cubo.

Ao utilizar uma grade mais próxima à superfície, mais cubos são utilizados na poligonalização, gerando mais pontos da malha para uma dada profundidade da *octree* quando comparado ao método *Dual Contouring*, que utiliza uma subdivisão com os pontos da subdivisão sempre no centro do cubo. Outra consequência de mais cubos estarem intersectando a superfície é que mais arestas a intersectam, gerando assim mais polígonos da malha que irão aproximar a superfície.

Para conectar os pontos da malha é necessário redefinir o conceito de Aresta Mínima, que é utilizado para conectar os vértices da malha. Na Grade Adaptativa o número de arestas que intersectam a superfície na subdivisão de um cubo não é predefinido, como no caso do *Dual Contouring*. Essa redefinição encontra as Arestas Mínimas da região de interseção dos cubos que se encontram em uma face ou aresta.

Os resultados apresentados nesta tese mostram que para duas poligonalizações, com a Grade Adaptativa e o *Dual Contouring*, mesmo para resultados visualmente não muito distintos, a Grade Adaptativa é mais precisa. Este aspecto é quantificado com a métrica introduzida. Isso se deve ao fato de serem geradas superfícies com mais polígonos e vértices da malha, pois as subdivisões da *octree* são mais bem aproveitadas. Outro aspecto importante apresentado pelos resultados é que a Grade Adaptativa reconstrói melhor variações na superfície, como por exemplo ondulações.

Como trabalhos futuros considera-se estudar, em resultados obtidos originalmente para o *Dual Contouring*, a possibilidade de aplicá-los à Grade Adaptativa e pesquisar as correspondentes adaptações. Neste caso, apresentam-se os trabalhos de simplificação da superfície mantendo suas características topológicas, como (20) e (21). Além disso, pretende-se estudar as implicações topológicas e geométricas de inserir mais de um ponto por cubo e como deve ser efetuada a conexão desses pontos com os pontos pertencentes aos cubos vizinhos.

Pode-se também trabalhar no ajuste da malha a condições específicas, como para o Método de Elementos Finitos. Este método requer uma malha cujos triângulos sejam mais próximos de triângulos equiláteros. Além disso, podem ser pensadas em condições para a geração da *octree* que levem em conta suas características geométricas e topológicas, como superfícies com buracos e com a curvatura mais acentuada em uma determinada região.

Outros trabalhos incluem desenvolver um algoritmo que una duas ou mais grades distintas que estão conectadas por faces ou arestas, costurando assim as superfícies poligonalizadas em cada uma dessas grades. O resultado desse trabalho pode ser utilizado para fazer um processamento paralelo, dividindo a grade original em grades menores, poligonalizando cada uma delas em paralelo e fazendo a conexão final das porções de superfície obtidas.

8
Referências Bibliográficas

[1] RUGGIERO, M.; LOPES, V. **Cálculo Numérico: Aspectos Teóricos e Computacionais**. São Paulo: Makron Books, 1996. 406 p.

[2] SüLI, E.; MAYERS, D. **An Introduction to Numerical Analysis**. Cambridge: Cambridge University Press, 2003. 443 p.

[3] MANTYLA, M. **Introduction to Solid Modeling**. New York: W.H. Freeman & Company, 1988. 144 p.

[4] JU, T. et al. Dual contouring of hermite data. In: **Proceedings of SIGGRAPH**. San Antonio: ACM SIGGRAPH Computer Graphics, 2002. p. 339–346.

[5] VELHO, L. Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints. In: **Proceedings of Eurographics**. Montreux: Elsevier Science, 1990. p. 125–136.

[6] VELHO, L.; GOMES, J.; FIGUEIREDO, L. H. **Implicit Objects in Computer Graphics**. New York: Springer-Verlag, 2002. 190 p.

[7] LORENSEN, W. E.; CLINE, H. E. Marching cubes: a high resolution 3d surface reconstruction algorithm. In: **Proceedings of SIGGRAPH**. Anaheim: ACM SIGGRAPH Computer Graphics, 1987. p. 163–169.

[8] KOBBELT, L. P. et al. Feature sensitive surface extraction from volume data. In: **Proceedings of SIGGRAPH**. Los Angeles: ACM SIGGRAPH Computer Graphics, 2001. p. 57–66.

[9] GIBSON, S. F. F. Using distance maps for accurate surface reconstruction in sampled volumes. In: **Proceedings of the IEEE Symposium on Volume Visualization**. North Carolina: Association for Computing Machinery, 1998. p. 23–30.

[10] NIELSON, G. M.; HAMANN, B. The asymptotic decider: Resolving the ambiguity in marching cubes. In: **Proceedings of IEEE Visualization**. San Diego: IEEE Computer Society Press, 1991. p. 83–93.

[11] BENTLEY, J. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, v. 8, p. 509 – 517, 1975.

[12] MEAGHER, D. **Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer**. Troy, NY, 1980.

[13] LEWINER, T. et al. Efficient implementation of marching cubes cases with topological guarantees. **Journal of Graphics Tools**, v. 8, p. 1–15, 2003.

[14] CHERNYAEV, E. **Marching Cubes 33: Construction of Topologically Correct Isosurfaces**. Genève, Switzerland, 1995.

[15] LEAL, T. et al. Generation of triangular meshes through vertices connection modification of the marching cubes algorithm. In: **11th USNCCM**. Minneapolis: US Association for Computational Mechanics, 2011. p. 120.

[16] LEAL, T. et al. A global change for mesh improvements in the marching cubes algorithm. In: **10th WCCM**. São Paulo: ABMEC, 2012. p. 210.

[17] EASTMAN, C.; WEISS, S. Tree structures for high dimensionality nearest neighbor searching. **Information Systems**, v. 7, p. 115–122, 1982.

[18] BRUIN, P. W. de et al. Improving triangle mesh quality with surfacenets. In: **MICCAI**. Pittsburgh: Springer LNCS, 2000. p. 804–813.

[19] SCHAEFER, S.; WARREN, J. **Dual Contouring: "The Secret Sauce"**. Houston, TX, 2002.

[20] PEIXOTO, A.; FARIAS, R.; VELHO, L. Simplificação de superfícies implicitas não-compactas com preservação de topologia. In: **SIBGRAPI Workshop of Thesis and Dissertations**. Curitiba: IEEE Press, 2004. p. 202–210.

[21] SCHAEFER, S.; JU, T.; WARREN, J. Manifold dual contouring. In: **Proceedings of IEEE Transactions on Visualization and Computer Graphics**. Piscataway: IEEE Educational Activities Department, 2007. p. 610–619.

[22] VARADHAN, G. et al. Feature- sensitive subdivision and isosurface reconstruction. In: **Proceedings of IEEE Visualization**. Seattle: IEEE Computer Society Press, 2003. p. 99–106.

[23] ZHANG, N.; HONG, W.; KAUFMAN, A. Dual contouring with topology-preserving simplification using enhanced cell representation. In: **Proceedings of the conference on Visualization**. Washington: IEEE Computer Society, 2004. p. 505–512.