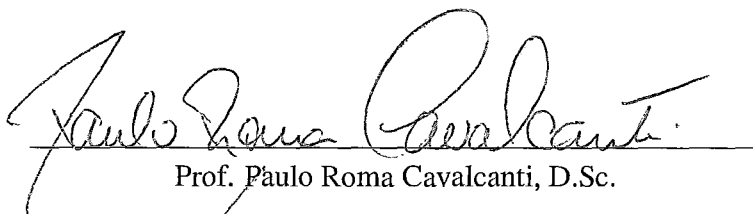


# TRIANGULAÇÕES ADAPTATIVAS EM MULTI-RESOLUÇÃO PARA OBJETOS HETEROGÊNEOS

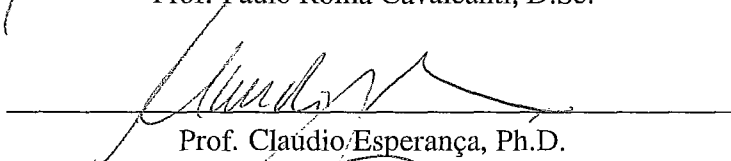
Ricardo Guerra Marroquim

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS  
PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE  
FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS  
NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS  
EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

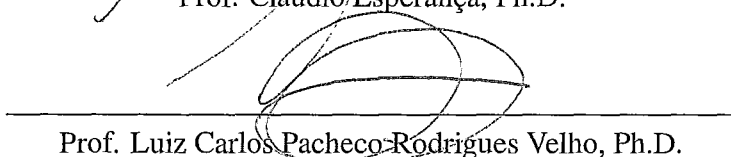
Aprovada por:



Prof. Paulo Roma Cavalcanti, D.Sc.



Prof. Claudio Esperança, Ph.D.



Prof. Luiz Carlos Pacheco Rodrigues Velho, Ph.D.

RIO DE JANEIRO, RJ - BRASIL  
ABRIL DE 2005

MARROQUIM, RICARDO GUERRA

Triangulações Adaptativas em Multi-resolução para Objetos Heterogêneos [Rio de Janeiro] 2005

XII, 56 p., 29,7 cm, (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2005)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 – Meshes 2 – Multi-resolution

3 – Triangulations

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

## TRIANGULAÇÕES ADAPTATIVAS EM MULTI-RESOLUÇÃO PARA OBJETOS HETEROGÊNEOS

Ricardo Guerra Marroquim

Abr/2005

Orientador: Paulo Roma Cavalcanti

Programa: Engenharia de Sistemas e Computação

Esta tese apresenta um método para gerar triangulações adaptativas em multi-resolução para objetos 3D compostos de várias regiões e geometria arbitrária. O processo primeiro cria uma malha tetraedral semi-regular adaptativa, chamada de triangulação BMT, em volta da fronteira original do modelo. Depois, os elementos da malha são empurrados em direção da fronteira utilizando um sistema de compressão baseado em física, mais especificamente um sistema de massa-mola. A triangulação final não contém tetraedros degenerados e oferece uma aproximação da fronteira baseado em um nível de resolução escolhido.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

## ADAPTIVE MULTI-RESOLUTION TRIANGULATIONS FOR HETEROGENEOUS OBJECTS

Ricardo Guerra Marroquim

Apr/2005

Advisor: Paulo Roma Cavalcanti

Department: Computing and Systems Engineering

This these presents a method for generating multi-resolution adaptive triangulations of non-manifold 3D objects composed of several regions with arbitrary geometry. The process first immerses the input boundary elements inside a semi-regular adaptive tetrahedral mesh known as a BMT Triangulation. The mesh elements are then pushed towards the boundary by means of a physically based compression scheme, more specifically, a mass-spring system. The final triangulation has no degenerated triangles and provides an approximation of the boundary based on a chosen resolution.



# Dedicatória

Aos meus pais.  
Em memória da minha grande amiga Angela Couto.

# Agradecimentos

Gostaria de agradecer a todas essas pessoas que durante a realização deste trabalho ajudaram tanto direta ou indiretamente.

Primeiro aos professores que apoiaram e contribuíram diretamente para a produção desta dissertação, ao meu orientador Prof. Paulo Roma, ao Prof. Claudio Esperança, e ao Prof. Luiz Velho. Ao Vinícius Mello por ter fornecido a sua estrutura de dados e ter dado todo suporte necessário para sua utilização. Ao Prof. Antônio Oliveira, pelas caronas e papos descontraídos no refeitório do LCG, e ao xará e amigo Prof. Ricardo Farias.

Ao Cenpes/Petrobras pelo apoio financeiro através do projeto Pesc 3678 da COPPE-TEC.

Aos companheiros do LCG pela união, seja no descanso da EBA, no refresco do Gremio na hora do almoço, nas chopadas, nos sambas das sextas-feiras, e sem esquecer aqueles momentos de trabalho também. Ao Vitor que se tornou um grande companheiro, e ao Leandro com quem dividi o têtto em paz nos últimos meses. À legião estrangeira, O Kapi (Álvaro), Carlitos (Karl) e Gil (Yalmar). Seguindo pelos nem tão estrangeiros assim Saulo, Disney, César, e Marcelo. Finalmente os Cariocas ... não sobrou ninguém!?!?! Mas o mineiro chega lá. E a galera nova que vem chegando com força, grande Caique, Max, André, André e Léo. E ao companheiro que não é do LCG mas que vem acompanhando dêes da graduação, Luciano.

A Alice que esteve tão presente na minha vida durante todo esse tempo.

À nata dos companheiros de trilha Helena e Marquinhos, pela união e fraternidade durante todas aquelas viagens maravilhosas pelas matas e montanhas do país, e todas que estão por vir.

A toda galera de Santa Tereza, juntando também o pessoal que não mora mas que está sempre por lá. Helena, Shaun, Alice, Marquinhos, Papi, Pedro, Vivi, Piu e Flávia, Jonas, Rafinha e Camila, Vi, Mari, Léo, Zé e Clarisse, Julio e Julia, Junior e Camila, Sergio e Sandra, Sensa, Janaína, Simone, Lili e André, Jon, Tânia e Denise, Xavier e Lia, Elisa, Nezinho, Tiago, Paulinho, Jorge Dalai, Diana, Mel, Pan, Aline, Maíra, Aurora, Bête, Mabel, Elena, Beta e Jim.

Galera do Projeto Quebra-Côco e futuras aventuras Henrique e Mari.

À toda galera do Sana, com quem convivi durante todos aqueles feriados e fins de semana. Luiz, Glaucia, Jack, Tati, Flôr, Claudia, Rodrigo, Charles, Celsão, Zezinho, Sana, e aos amigos do Tenar, Laercio, Vilma, Ciely, Laerty e Vylmara.

À minha irmã Aninha e minha sobrinha peste Bia.

À toda família e amigos de Recife. Meus tios, tias, primos e primas, Vó Célia, Vó Carmem e Vô Izaltino. Meus primos que na verdade são irmãos Guga e Carol.

Aos meus professores culturais João Cantiber, Isidoro e Simone.

Aos meus queridos que sempre estão de bom humor e são mais humanos que tanta

gente, Jimmy, Tuti, Kika, Chamito, Peninha, Nenem, Odin, Ruby e Pingo. Em memória do Saci e da Bala.

Ao CCJF pelas quartas instrumentais que equilibram tão bem a semana.

A Mauá, o que seria de nós sem você, sempre trazendo a alegria da forma mais simples!

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Trabalhos Relacionados</b>	<b>3</b>
2.1	Refinamento de Delaunay . . . . .	3
2.1.1	O Algoritmo de Ruppert . . . . .	4
2.1.2	O Algoritmo de Chew . . . . .	8
2.1.3	O Algoritmo de Shewchuk . . . . .	8
2.2	Refinamento de Delaunay para Malhas Tetraedrais . . . . .	10
2.3	Avanço de Fronteira . . . . .	11
2.4	Empacotamento de Círculos . . . . .	16
2.5	Empacotamento de Esferas . . . . .	17
<b>3</b>	<b>Subdivisão Espacial</b>	<b>21</b>
3.1	Triangulações 4-8 . . . . .	21
3.2	Multi-Triangulação Binária . . . . .	23
<b>4</b>	<b>Modelos Tratados</b>	<b>25</b>
4.1	Representação de Polígonos em 2D . . . . .	25
4.2	Representação de Modelos em 3D . . . . .	27
<b>5</b>	<b>Função Distância</b>	<b>28</b>
5.1	Função Distância em 2D . . . . .	28
5.2	Função Distância em 3D . . . . .	31
<b>6</b>	<b>Malha de Deslocamento</b>	<b>32</b>
6.1	Malha de Deslocamento . . . . .	33
6.2	Modelos com Multi-Região . . . . .	34
<b>7</b>	<b>Sistema Físico</b>	<b>38</b>
7.1	Configuração das Molas . . . . .	38
7.2	Sistema de Partículas . . . . .	40
7.3	Acumuladores de Forças . . . . .	41
7.4	Valores Derivados . . . . .	41
7.5	Atualização das Coordenadas e Velocidades . . . . .	41
7.6	Crítérios de Compressão . . . . .	41
7.7	Limitando a Compressão . . . . .	42

<b>8 Resultados</b>	<b>43</b>
<b>9 Conclusões e Trabalhos Futuros</b>	<b>52</b>
9.1 Conclusões . . . . .	52
9.2 Trabalhos Futuros . . . . .	52
<b>Referências Bibliográficas</b>	<b>54</b>

# Lista de Figuras

2.1	(a) O circuncírculo com centro em $d$ de um triângulo $\triangle abc$ , onde $\theta$ é o ângulo mínimo do triângulo, $l$ seu menor lado, e $r$ o raio do círculo. (b) Os dois triângulos isósceles $\triangle adb$ e $\triangle adc$ . . . . .	4
2.2	O ponto $C$ marca o circuncentro do triângulo <i>magro</i> destacado. Ao inserir um novo vértice no lugar de $C$ , nenhuma aresta nova terá comprimento menor do que o circunraio $r$ . . . . .	5
2.3	O segmento $s$ é “envolto” pelo vértice $v$ que se encontra dentro do seu círculo <i>diametral</i> . . . . .	5
2.4	Problema gerado por ângulos pequenos do polígono. Como todo vértice inserido torna algum segmento <i>envolto</i> , o algoritmo continua subdividindo as aresta indefinidamente. . . . .	6
2.5	O “escudo” criado em volta do vértice $v$ . A zona de proteção é previamente triangulada (linhas pontilhadas). As linhas escuras representam os segmentos do polígono que formam ângulos pequenos. . . . .	7
2.6	Quando o segmento $pr$ é subdividido, ao invés de utilizar o ponto médio $m_1$ , o ponto $v_1$ no círculo mais próximo é utilizado. Analogamente o segmento $pq$ é subdividido no ponto $v_2$ . Os segmentos subdivididos $pv_1$ e $pv_2$ possuem os mesmo tamanho, e portanto $v_1$ não envolve $pv_2$ e $v_2$ não envolve $pv_1$ . . . . .	7
2.7	(a) O triângulo <i>magro</i> $t$ e seu circuncentro $c$ estão de lados opostos ao segmento de entrada $s$ . (b) Vértices removidos marcados por um $X$ , e o novo vértice inserido no ponto $m$ . . . . .	8
2.8	Os segmentos subdivididos do conjunto formado por aqueles incidentes ao vértice $a$ que formam ângulos pequenos. . . . .	9
2.9	Os circuncírculos vazios dos segmentos incidentes à $a$ . . . . .	9
2.10	Um tetraedro chato apesar de não possuir uma razão do circunraio-menor-aresta superior ao limite $L$ . . . . .	11
2.11	O modelo em vários passos durante a triangulação utilizando “Advancing Front”. (a) passo 1; (b) passo 500; (c) passo 2000; (d) passo 4000; (e) malha final após 4721 passos. . . . .	12
2.12	Escolha do ponto $p$ como extensão da frente. . . . .	13
2.13	$p$ não é ideal porque existem outros vértices dentro do círculo de raio $\delta$ e centro em $p$ . . . . .	13
2.14	Os centros $c_1$ , $c_2$ e $c_3$ dos círculos formados por $a$ , $b$ e os vértices $q_1$ , $q_2$ e $q_3$ respectivamente. . . . .	14
2.15	Seqüência de pontos iguais em espaços de $p$ . O primeiro ponto válido será escolhido para avançar a fronteira. . . . .	14

2.16	Escolha do ponto ideal $p$ para o avanço da face definida por $a$ , $b$ e $c$ e centróide $m$ . . . . .	15
2.17	Escolha do ponto de junção quando existem vértices da malha dentro da esfera de raio $\delta$ e centro em $p$ . . . . .	15
2.18	(a) Colocação dos círculos nos vértices do polígono. (b) Preenchimento do domínio com círculos. (c) Inserção das arestas. (d) Triangulação final do polígono. . . . .	16
2.19	Discos adicionados nos cantos convexos (a), e nos côncavos (b). . . . .	17
2.20	Triangulação dos cantos convexos (a), e dos cantos côncavos (b). . . . .	17
2.21	Triangulação das regiões com três lados. . . . .	18
2.22	Triangulação das regiões com quatro lados. . . . .	18
2.23	Distribuição dos círculos em uma <i>quadtrees</i> . . . . .	19
2.24	Forças entre esferas. (a) Forças de repulsão; (b) forças estáveis; (c) forças de atração. . . . .	19
2.25	Qualidade da malha. (a) Triangulação de Delaunay; (b) Diagrama de Voronoy; (c) Distribuição das esferas. . . . .	20
3.1	Uma malha 4-8 gerada sobre um modelo de uma guitarra. O refinamento local gera triângulos menores perto dos detalhes da malha. . . . .	22
3.2	Uma esfera com elementos menores perto da fronteira e maiores no interior. A malha exibida já passou pelo processo de compressão. . . . .	24
4.1	Modelo em 3 dimensões da Bacia do Golfo do México. . . . .	25
4.2	Um corte bidimensional do modelo do Golfo do México. . . . .	26
4.3	Árvore binária representando uma curva poligonal com oito vértices. . . . .	26
5.1	A região envolvente de um nó, definida pela <i>distância máxima</i> (linha pontilhada). O <i>segmento central</i> é definida pelos vértices $v_0$ e $v_n$ . . . . .	28
5.2	O ponto $p$ é interior à região, e o ponto $q$ exterior. O segmento $a$ é definido pelo ponto $p$ e o ponto mais próximo no segmento $s$ . O segmento $b$ é definido da mesma forma para o ponto $q$ . . . . .	29
5.3	Os pontos $p$ e $q$ são exteriores à região mas o sinal da função distância será diferente por estarem mais próximos de um vértice e os dois segmentos mais próximos, $s$ e $u$ , formarem um ângulo interno menor do que $90^\circ$ . . . . .	30
5.4	O ponto $r$ é definido como interior ao triângulo formado pelos dois segmentos $s$ e $u$ . O segmento $c$ é calculado pela função distância aplicada a $r$ . . . . .	30
5.5	Pseudo-código para a função distância da octree. . . . .	31
6.1	A <i>malha-d</i> de uma esfera antes e depois da compressão em três níveis de resolução diferentes. . . . .	32
6.2	Uma representação em duas dimensões da <i>malha de deslocamento</i> . Os vértices do <i>envelope interior</i> estão desenhados em branco. Em preto, os vértices da fronteira da <i>malha de deslocamento</i> , ligados pela linha preta grossa. A curva representa a fronteira original do modelo. As linhas pontilhadas são as arestas descartadas da malha inicial. . . . .	33

6.3	As arestas apontadas representam as interiores da <i>malha-d</i> conectando dois vértices marcados. A fronteira da <i>malha-d</i> está desenhada com segmentos grossos, e a linha curva representa a fronteira do modelo. Para facilitar a visualização a <i>malha-d</i> está representada em 2 dimensões, mas o caso para 3 dimensões é análogo. . . . .	34
6.4	Na esquerda, um visão do modelo completo de um domo de sal. Na direita, as regiões separadas. . . . .	35
6.5	Uma visão em duas dimensões da superfície delimitadora (linha grossa) entre as regiões branca e cinza. Os vértices na superfície delimitadora são os vértices marcados para serem empurrados em direção à fronteira, representada pela curva. . . . .	36
6.6	Um canto de um cubo com uma face interior com os três vértices fronteira.	36
6.7	Tetraedro exterior com os quatro vértices marcados: $V_0$ , $V_1$ , $V_2$ , e $V_3$ . As faces de fronteira são as hachuradas ( $F_0$ e $F_1$ ), e as setas apontam para o interior do modelo. . . . .	37
7.1	Seis esquemas diferentes para configuração do sistema de massa-mola na malha: (a) somente molas nas arestas da malha, (b) somente molas dos vértices ao centróide do triângulo, (c) molas nas arestas e nos centróides, (d) somente molas nas medianas do triângulo, (e) molas nas arestas e nas medianas, (f) molas nas arestas e nas projeções dos vértices às arestas opostas. Em preto os vértices reais da malha, em branco os vértices virtuais.	39
7.2	Molas incidentes no vértice $v_0$ . Três molas de aresta e uma mola de projeção. As outras molas do tetraedro, três de projeção e três de aresta, não estão ilustradas nesta figura. . . . .	39
8.1	Modelo do Lago Superior triangulado. . . . .	44
8.2	Detalhe da aproximação da malha nas ilhas do Lago Superior. As fronteiras originais são representadas por contornos de linha, e a malha final por regiões preenchidas. . . . .	44
8.3	Modelo de um domo de sal. . . . .	45
8.4	Fatia do modelo do Golfo do México. . . . .	45
8.5	(a) malha com as regiões pintadas. (b) <i>serrilhamento</i> no local onde várias regiões se encontram. (c) e (e) detalhes das regiões onde a aproximação foi quase exata. (d) e (f) as mesmas regiões trianguladas. . . . .	46
8.6	Aproximação grosseira do Coelho de Stanford com 30K tetraedros. Na seqüência a <i>malha-d</i> antes, durante e ao final (com e sem arestas) da compressão. . . . .	47
8.7	Triangulação do Tigre com aproximadamente 160K tetraedros. . . . .	48
8.8	Um corte do modelo do Tigre. . . . .	48
8.9	Modelo do tigre exposto em diversas resoluções consecutivas. . . . .	49
8.10	Três visões da malha final e uma da <i>malha-d</i> . . . . .	50
8.11	Corte da malha do Domo de Sal. . . . .	51



# Lista de Tabelas

2.1	Ângulo Mínimo . . . . .	3
-----	-------------------------	---

# Capítulo 1

## Introdução

As simulações numéricas se tornaram ferramentas importantes no desenvolvimento de produtos de engenharia e predição do comportamento de fenômenos físicos, como condições climáticas, geração e migração de óleo, movimento das ondas, terremotos, etc.

Para fazer uso de uma simulação numérica, é necessário que o domínio em questão seja discretizado. Nesta discretização, geralmente conhecida como triangulação, uma série de equações descrevendo leis físicas devem ser resolvidas.

Nas últimas três décadas vários métodos de triangulação foram propostos. Porém, estes métodos foram criados para trabalhar principalmente com modelos exatos, comumente gerados por sistemas CAD. Nestas situações a triangulação gerada precisa ser totalmente fiel aos contornos do modelo original, pois pequenos detalhes não podem ser perdidos.

No entanto, existem outras aplicações onde uma malha que aproxime o modelo original é aceitável, como os modelos geológicos por exemplo. Estes modelos são gerados a partir de uma interpretação de dados sísmicos por geólogos. Como consequência, o processo de criação dos modelos não é exato e está sujeito a erros.

Modelos como estes contêm várias propriedades que dificultam sua discretização. Geralmente possuem bordas complexas e irregulares, pois devem representar falhas, cavidades e diferentes materiais nas camadas dos terrenos. Os algoritmos tradicionais, quando conseguem, encontram muitas dificuldades para triangular estes modelos, e normalmente geram muitos elementos de má qualidade na malha. Estes elementos geram problemas para a simulação numérica, e podem impedi-la de chegar a um término.

É apresentado nesta dissertação um método para triangular modelos em duas ou três dimensões, com bordas complexas e irregulares, contendo várias regiões e em multi-resolução. O esquema prioriza a geração de uma malha com qualidade, e para tanto não é necessariamente fiel à borda original do polígono.

O algoritmo pode ser dividido em dois passos principais. Primeiramente uma malha adaptativa é gerada sobre o modelo original para subdividir o espaço em questão. No Capítulo 3 detalhes sobre a subdivisão espacial são expostos. A representação dos modelos de entrada é detalhada no Capítulo 4. O Capítulo 5 descreve uma função distância de ponto no espaço ao modelo, utilizada para acelerar o algoritmo.

No segundo passo a malha gerada é comprimida utilizando um sistema físico para adequar-se à fronteira do modelo. O Capítulo 6 descreve o processo de preparação da malha para compressão. O processo de compressão utilizando um sistema de molas é descrito no Capítulo 7.

No Capítulo 2 são apresentados alguns trabalhos anteriores realizados na área de geração de malhas. Finalmente, resultados são expostos no Capítulo 8, e conclusões e trabalhos futuros no Capítulo 9.

O algoritmo proposto é baseado na idéia de Molino et al. [19], onde um esquema de triangulação baseado em simulação física também é descrito. Porém, Molino concentra seus esforços em criar malhas especificamente para simulações onde grande deformações são necessárias. Como extensão do seu trabalho, o método apresentado gera malhas em multi-resolução para modelos contendo várias regiões sem gerar nenhum, ou praticamente nenhum, elemento mal-formado.

# Capítulo 2

## Trabalhos Relacionados

Neste capítulo são apresentados alguns algoritmos clássicos de geração de malhas.

### 2.1 Refinamento de Delaunay

A idéia geral dos algoritmos de *refinamento de Delaunay* é gerar uma triangulação de Delaunay sobre o domínio, e a partir desta, inserir novos pontos para melhorar a qualidade da malha. Em uma triangulação de Delaunay todo triângulo deve permanecer com seu circuncírculo vazio. O circuncírculo, por sua vez, é o círculo único que passa pelos três vértices do triângulo.

A qualidade de um triângulo gerado é medida pela razão entre seu circunraio e a aresta mais curta, ou seja, o raio do circuncírculo  $r$  dividido pelo comprimento da menor aresta  $l$ , como pode ser visto na Figura 2.1(a). A razão  $\frac{r}{l}$  está diretamente associada ao ângulo mínimo  $\theta$  do triângulo. Quanto menor esta razão, maior o ângulo mínimo. Alguns valores são apresentados na Tabela 2.1.

Tabela 2.1: Ângulo Mínimo

$\theta$	60°	45°	30°	10°	5°	1°
$r/l$	0.57	0.70	1.0	2.88	5.74	28.65

Pela Figura 2.1(b), percebe-se que  $\angle bdc = 2\theta$ . Seja  $\alpha = \angle cad$ , tem-se:

$$\angle adb = 180^\circ - 2(\theta + \alpha) \quad (2.1)$$

e

$$\angle adc = 180^\circ - 2\alpha, \quad (2.2)$$

peço fato de que  $\triangle adb$  e  $\triangle adc$  são isósceles. Subtraindo a equação 2.1 de 2.2, encontra-se:

$$\angle bdc = 2\theta. \quad (2.3)$$

Ainda pela Figura 2.1(b), nota-se que  $\text{sen}\theta = \frac{l}{2r}$ . Logo, para  $\frac{r}{l} \leq \mathbf{L}$  o ângulo mínimo  $\theta$  não pode ser superior à  $\text{arcsen}\frac{1}{2\mathbf{L}}$ . Um triângulo com razão superior ao limite  $\mathbf{L}$  é chamado de triângulo *magro*.

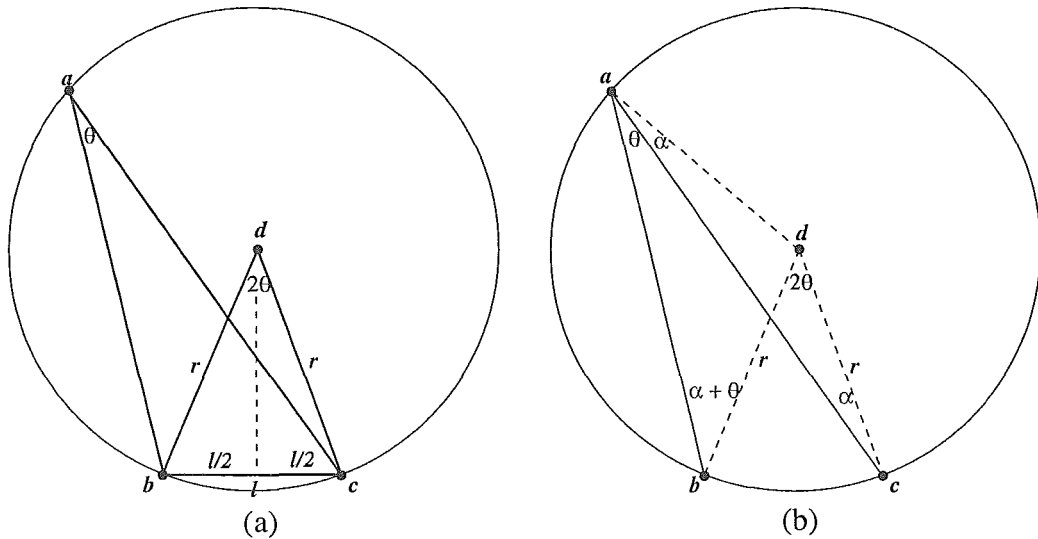


Figura 2.1: (a) O circuncírculo com centro em  $d$  de um triângulo  $\triangle abc$ , onde  $\theta$  é o ângulo mínimo do triângulo,  $l$  seu menor lado, e  $r$  o raio do círculo. (b) Os dois triângulos isósceles  $\triangle adb$  e  $\triangle adc$ .

A operação principal no *refinamento de Delaunay* é a inserção de um ponto no circuncentro de um triângulo para melhorar a qualidade da malha. Para remover os triângulos *magros* da malha, um novo vértice é inserido no seu circuncentro. Após cada operação a malha deve ser recomputada. Como o circuncírculo de qualquer triângulo deve permanecer sempre vazio numa triangulação de Delaunay, inserindo o novo vértice o triângulo é eliminado. Além disso, nenhuma aresta nova terá comprimento menor do que o circunraio, como pode ser observado na Figura 2.2. O algoritmo continua inserindo novos vértices até que nenhum triângulo possua razão maior do que  $L$ .

### 2.1.1 O Algoritmo de Ruppert

Um dos algoritmos de triangulação mais conhecido é o criado por Jim Ruppert [22]. Este foi um dos primeiros algoritmos de *refinamento de Delaunay* com provas teóricas para suportá-lo.

Ruppert chama de *envolto*, um segmento da triangulação cujo *círculo diametral* contém algum vértice que não seja um dos vértices que definem o próprio segmento. O *círculo diametral*, por sua vez, é o menor círculo possível que contém o segmento. A Figura 2.3 ilustra um exemplo de um segmento *envolto*  $s$  pelo vértice  $v$ .

Para gerar uma triangulação respeitando o limite  $L$  é preciso tratar os casos de segmentos *envoltos* e triângulos *magros*. Um segmento *envolto* é subdividido inserindo um novo vértice no seu ponto médio. Por outro lado, um triângulo *magro* é geralmente eliminado inserindo um novo vértice no seu circuncentro. Porém, se este novo vértice gerar um ou mais segmentos *envoltos*, ao invés de inserí-lo, todos esses segmentos que se tornariam *envoltos* são subdivididos.

Ruppert provou que para um limite  $L = \sqrt{2}$ , o algoritmo possui garantia de convergência. O limite  $L \geq \sqrt{2}$  gera uma triangulação com ângulos entre  $20.7^\circ$  e  $138.6^\circ$ . Em contrapartida, Ruppert afirma que esta garantia só é válida para polígonos com ângulo

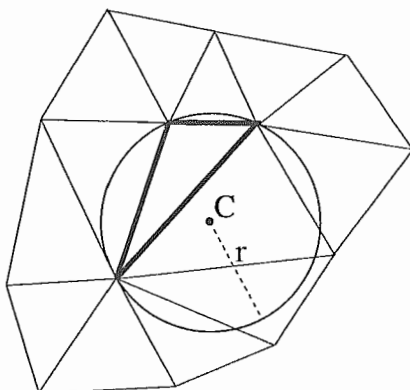


Figura 2.2: O ponto  $C$  marca o circuncentro do triângulo *magro* destacado. Ao inserir um novo vértice no lugar de  $C$ , nenhuma aresta nova terá comprimento menor do que o circunraio  $r$ .

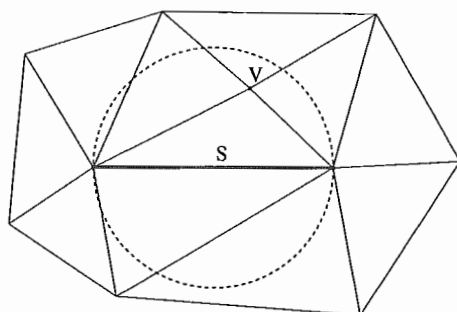


Figura 2.3: O segmento  $s$  é “envolto” pelo vértice  $v$  que se encontra dentro do seu *círculo diametral*.

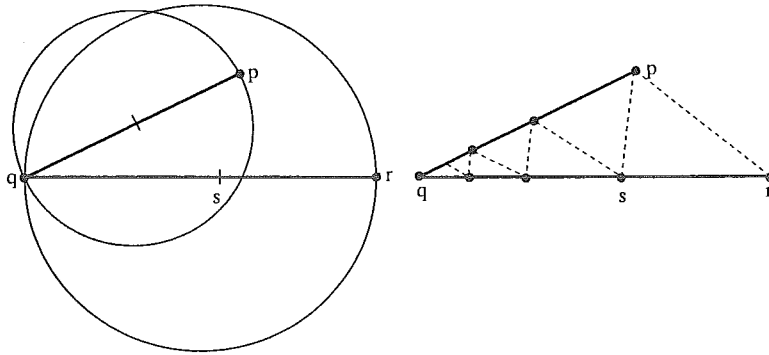


Figura 2.4: Problema gerado por ângulos pequenos do polígono. Como todo vértice inserido torna algum segmento *envolto*, o algoritmo continua subdividindo as aresta indefinidamente.

mínimo de pelo menos  $90^\circ$ . Shewchuk [24] prova que na verdade esta restrição pode ser reduzida para  $60^\circ$ .

Para ângulos menores do que  $60^\circ$  o algoritmo tende a gerar triângulos mal formados, especialmente perto dos ângulos pequenos de entrada. Ainda mais, Shewchuk [24] mostra que em alguns casos, os algoritmos de *Refinamento de Delaunay* podem não terminar quando existem ângulos pequenos no polígono. A Figura 2.4 ilustra um dos problemas gerados por ângulos pequenos do polígono, particularmente quando existe algum menor do que  $45^\circ$ . O segmento  $qr$  está envolto pelo vértice  $p$ . Logo,  $qr$  é dividido inserindo um novo vértice  $s$  no seu ponto médio. Porém, o segmento  $qp$  será envolto pelo novo vértice  $s$ , e também deve ser subdividido. Seguindo, o segmento  $qs$  será envolto pelo novo vértice e será dividido. Este procedimento pode continuar indefinidamente, e portanto, o algoritmo não chega a um término.

Para sanar este problema, Ruppert inicialmente sugere a idéia introduzida por Bern et al. [3], chamada de “*corner looping*”. É criado um “escudo” em volta dos vértices de entrada com ângulos pequenos. Este “escudo” funciona como uma zona de proteção, pois é triangulado previamente para evitar que ângulos pequenos sejam inseridos durante o refinamento. Na Figura 2.5 um exemplo de um “escudo” é ilustrado.

No entanto, na prática Ruppert não utilizou o algoritmo de Bern, alegando que sua implementação é demasiada complicada, e que o esquema tende a gerar mais triângulos do que o necessário. Ele criou uma nova técnica mas não obteve provas teóricas de que seja ótimo.

O esquema de Ruppert cria círculos concêntricos em volta de cada vértice do polígono de entrada. Cada círculo possui o dobro do raio do círculo mais interno, onde todos raios são potências de dois. Ao dividir um segmento, cuja uma das extremidades é um vértice de entrada, o segmento é dividido na intersecção com o círculo mais próximo, e não no seu ponto médio. A Figura 2.6 ilustra o método de Ruppert. Eventualmente, os segmentos separados por um ângulo pequeno serão subdivididos a um mesmo tamanho. Como segmentos de tamanhos iguais não envolvem um ao outro, o ciclo indefinido de subdivisões é impedido.

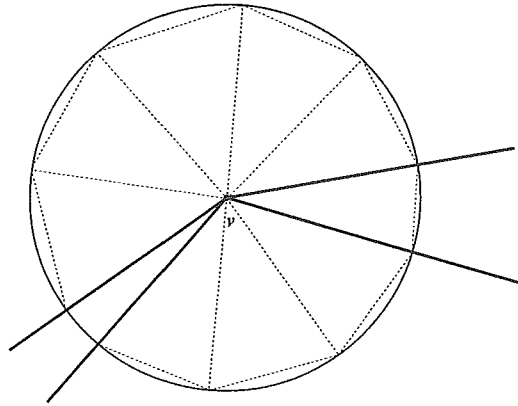


Figura 2.5: O “escudo” criado em volta do vértice  $v$ . A zona de proteção é previamente triangulada (linhas pontilhadas). As linhas escuras representam os segmentos do polígono que formam ângulos pequenos.

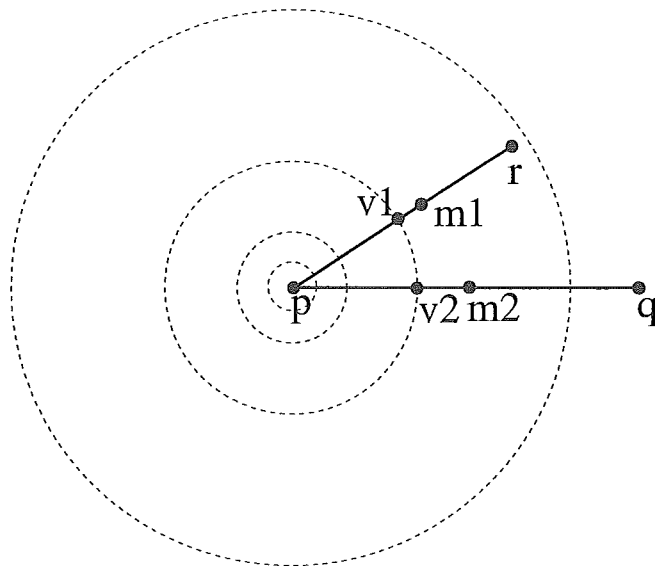


Figura 2.6: Quando o segmento  $pr$  é subdividido, ao invés de utilizar o ponto médio  $m_1$ , o ponto  $v_1$  no círculo mais próximo é utilizado. Analogamente o segmento  $pq$  é subdividido no ponto  $v_2$ . Os segmentos subdivididos  $pv_1$  e  $pv_2$  possuem o mesmo tamanho, e portanto  $v_1$  não envolve  $pv_2$  e  $v_2$  não envolve  $pv_1$ .



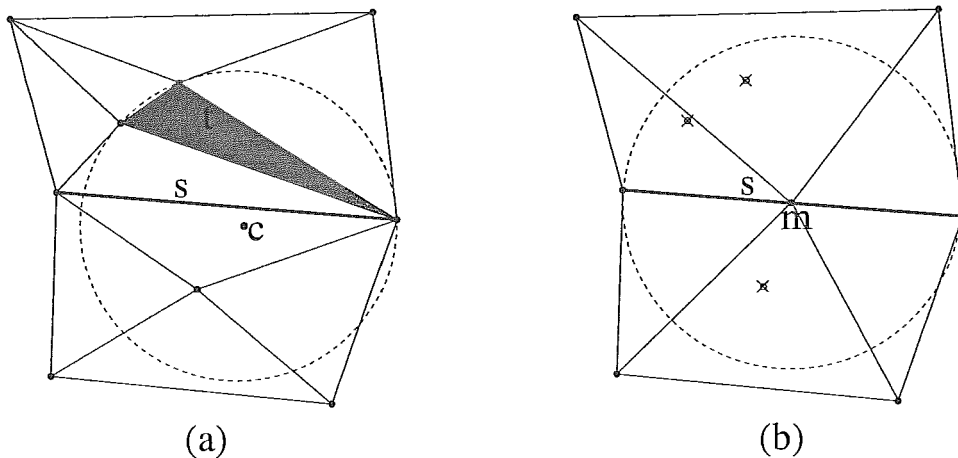


Figura 2.7: (a) O triângulo *magro*  $t$  e seu circuncentro  $c$  estão de lados opostos ao segmento de entrada  $s$ . (b) Vértices removidos marcados por um  $X$ , e o novo vértice inserido no ponto  $m$ .

### 2.1.2 O Algoritmo de Chew

O algoritmo de Chew [7], comparado com o de Ruppert, oferece uma melhoria no limite do ângulo mínimo. No seu algoritmo a triangulação é gerada com ângulo mínimo de  $26.5^\circ$ , ao invés de  $20.7^\circ$ . Ainda mais, o primeiro algoritmo de Chew [6] possui limite de  $30.0^\circ$ , porém gera apenas malhas uniformes.

A diferença principal para o algoritmo de Ruppert está no fato de Chew não utilizar *círculos diametraes* para subdividir os segmentos *envoltos*. Um ponto não é mais inserido no circuncentro  $c$  de um triângulo *magro*  $t$  quando existe algum segmento de entrada separando os dois, como ilustrado na Figura 2.7 (a). Ao inserir um novo vértice em  $c$ , o triângulo  $t$  não será eliminado ao refazer a triangulação. Neste caso, todos vértices (exceto os de entrada) contidos pelo *círculo diametral* de  $s$  e visíveis de  $m$  (ponto médio de  $s$ ) são retirados da triangulação. Após, um novo vértice é inserido em  $m$  e a triangulação recomputada, como pode ser visto na Figura 2.7 (b).

Se não houverem ângulos de entrada menores do que  $60^\circ$ , é provado que o algoritmo de Chew termina para um limite  $L \geq \frac{\sqrt{5}}{2} = 1.12$ . Este limite garante um ângulo mínimo  $\theta = \arcsen \frac{1}{\sqrt{5}} = 26.56^\circ$ . O seu algoritmo para malhas uniformes possui limite mínimo  $L = 1$

### 2.1.3 O Algoritmo de Shewchuk

O resultado desejado para os algoritmos de triangulação é gerar uma malha onde os únicos ângulos pequenos sejam aqueles já impostos pelo polígono de entrada. Porém, nenhum algoritmo obteve esta garantia ainda, e pior, Shewchuk [26] afirma que esta propriedade é impossível de ser alcançada. No entanto, ele criou alternativas melhores para tratar estes ângulos pequenos do polígono de entrada [24, 23, 26].

O seu algoritmo, conhecido como *The Terminator* em [26] e *The Quitter* em [24], é baseado nos *círculos concêntricos* de Ruppert. No entanto, Shewchuk oferece garantias de término e de ângulo mínimo sem restrição aos ângulos do polígono. Um problema no

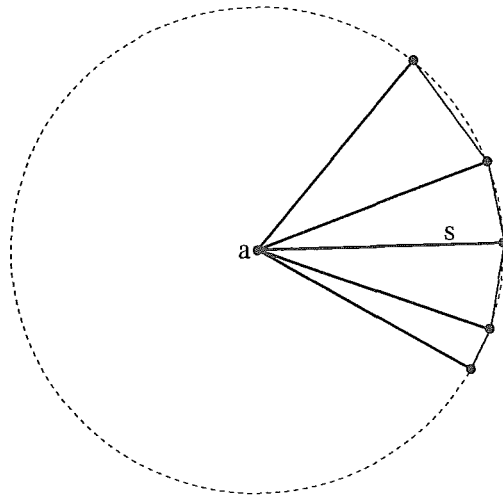


Figura 2.8: Os segmentos subdivididos do conjunto formado por aqueles incidentes ao vértice  $a$  que formam ângulos pequenos.

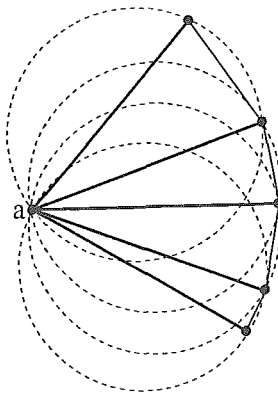


Figura 2.9: Os circuncírculos vazios dos segmentos incidentes à  $a$ .

algoritmo de Ruppert é a geração de arestas pequenas e alguns ângulos muito grandes. Para minimizar este problema Shewchuk criou um processo de decisão mais elaborado para a subdivisão de um segmento *envolto*.

Em particular, esse processo ocorre quando a inserção de um novo vértice no circuncentro de um triângulo *magro* torna algum outro segmento *envolto*  $s$ . Neste caso o circuncentro é rejeitado e um processo para decidir se  $s$  deve ser ou não subdividido é iniciado como detalhado a seguir.

Se nenhum vértice de  $s$  possui um ângulo de entrada menor do que  $60^\circ$ , ou se os dois possuem,  $s$  é subdividido. Caso contrário, seja  $a$  o vértice do ângulo pequeno de entrada. Um conjunto de segmentos é definido contendo todos segmentos incidentes ao vértice  $a$  que estão separados de  $s$ , ou um outro membro do conjunto, por  $60^\circ$  ou menos. No vértice  $a$  são criados círculos concêntricos para subdividir os segmentos do conjunto. Se um segmento for subdividido todos outros seguirão na subdivisão. A Figura 2.8 ilustra o conjunto de segmentos incidentes ao vértice  $a$ , incluindo o segmento *envolto*  $s$ , após o processo de subdivisão. Os circuncírculos vazios dos triângulos formados pelo conjunto de segmentos são ilustrados na Figura 2.9.

Para decidir se  $s$  deve ou não ser subdividido, Shewchuk se baseia no *raio de inserção* e em uma relação de ancestrais dos vértices. O *raio de inserção*  $r_v$  de um vértice  $v$  é o tamanho da menor aresta incidente logo após sua inserção. Por outro lado,  $p(v)$  (o pai de  $v$ ) é o vértice responsável pela inserção de  $v$ . O pai de um vértice é definido para três situações diferentes:

- se  $v$  é um vértice de entrada não existe  $p(v)$ ;
- se  $v$  é um vértice inserido no ponto médio de um segmento,  $p(v)$  é o vértice que tornou o segmento *envolto*;
- se  $v$  é um vértice inserido (ou rejeitado) no circuncentro de um triângulo,  $p(v)$  é o vértice incidente na menor aresta do triângulo inserido mais recentemente.

Seja  $v$  o vértice que será inserido se o segmento  $s$  for subdividido, e  $g$  o avô de  $v$ . O algoritmo determina  $r_g$ , o *raio de inserção* de  $g$ , e  $r_{min}$ , o *raio de inserção mínimo* detalhado a seguir.

Se  $s$  for subdividido, todos outros segmentos pertencentes ao conjunto incidente em  $a$  (como definido acima), que tiverem tamanho maior ou igual a  $|s|$ , serão subdivididos até que cheguem ao mesmo tamanho  $|s|/2$ . O *raio de inserção mínimo* é definido como o menor *raio de inserção* de todos vértices inseridos nas subdivisões do conjunto, incluindo o próprio  $v$ . Se  $\phi_{min}$  é o menor ângulo separando dois segmentos do conjunto então  $r_{min} = \frac{|s|}{2} \text{sen}(\phi_{min})$ .

Com estas definições, Shewchuk coloca que  $r_{min} \geq r_g$  é a condição principal para determinar se a subdivisão deve ocorrer. Caso esta condição não seja satisfeita, a subdivisão de  $s$  pode gerar arestas pequenas e ângulos grandes desnecessários. Outras condições também são propostas para permitir a subdivisão de  $s$ , mesmo se  $r_{min} < r_g$ , em casos especiais que não geram problemas.

Para malhas sem ângulos menores do que  $60^\circ$  o algoritmo de Shewchuk prossegue igualmente ao de Ruppert. Caso contrário, Shewchuk garante que o algoritmo terminará sempre e que os únicos ângulos pequenos criados serão em torno dos ângulos de entrada pequenos. Em particular, Shewchuk [26] mostra que, teoricamente, seu algoritmo não produz nenhum ângulo menor do que  $\frac{\phi}{2\sqrt{2}}$ , onde  $\phi$  é o menor ângulo de entrada. Porém, relata que na prática não foi observado nenhum ângulo menor do que  $\phi$ .

## 2.2 Refinamento de Delaunay para Malhas Tetraedrais

Shewchuk estendeu seu trabalho para gerar malhas tetraedrais, [25]. Os conceitos utilizados no algoritmo para 2 dimensões são expandidos para 3 dimensões. Uma *esfera diametral* de um segmento é definida como a menor esfera que contém o próprio. Um segmento é *envolto* quando outros vértices estão contidos por sua *esfera diametral*, e são subdivididos inserindo um novo vértice no seu ponto médio. Por sua vez, um triângulo *envolto* é aquele que não possui sua *esfera equatorial* vazia, a menor esfera que passa pelos seus três vértices. Um tetraedro *magro*, com razão  $\frac{r}{l} \geq \mathbb{L}$ , é eliminado inserindo um novo ponto no centro da sua *esfera equatorial*. No entanto, as provas teóricas para ângulo mínimo e garantia de término são válidas apenas para modelos com menor ângulo de pelo menos  $90^\circ$ .

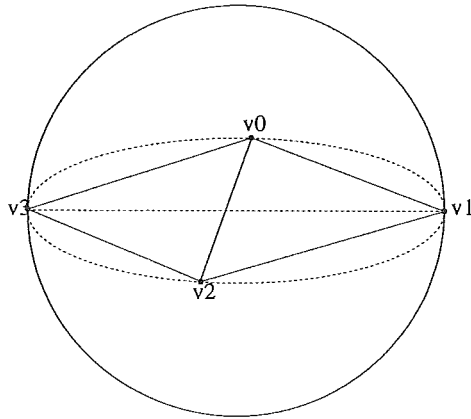


Figura 2.10: Um tetraedro chato apesar de não possuir uma razão do circunraio-menor-aresta superior ao limite  $L$ .

A razão principal pela qual o *Refinamento de Delaunay* não obteve grande sucesso em três dimensões, se deve ao fato de ocorrerem tetraedros chatos na triangulação final, conhecidos como “sliver”. Estes tetraedros, como ilustrado na Figura 2.10, podem ter volumes desprezíveis, ou até mesmo igual a zero. Porém, alguns sobrevivem ao processo de eliminação de tetraedros mal formados pois geralmente não possuem razão de qualidade inferior ao limite  $L$ . Isto se deve ao fato da menor aresta de um tetraedro chato não ser necessariamente muito menor do que o circunraio do mesmo. Mesmo existindo métodos para remover tetraedros chatos, nenhum possui garantia de remover todos.

## 2.3 Avanço de Fronteira

O método de “*Avanço de Fronteira*” (Advancing Front) gera uma malha inserindo elementos a partir da fronteira do domínio em direção ao interior. Uma frente inicial é criada discretizando a fronteira do domínio. Em duas dimensões um polígono é gerado, enquanto que em três dimensões uma superfície triangulada é gerada. Os elementos da malha são inseridos um por vez atualizando a frente. A cada passo uma aresta ou face da frente é escolhida, e um novo ponto criado gerando um novo triângulo ou tetraedro. A aresta ou face escolhida é chamada de *aresta base* ou *face base* respectivamente.

A dificuldade encontrada neste método reside na junção dos elementos da frente. A medida que ela avança, o algoritmo de inserção de pontos pode gerar elementos intersecantes. Neste momento é preciso “costurar” a malha para que todo domínio seja triangulado.

Vários esquemas de triangulação utilizando esta técnica foram propostos no passado [17, 16, 20, 15, 11, 10]. A Figura 2.11 ilustra uma seqüência da geração de uma malha utilizando esta técnica.

A escolha da *aresta base* e do local de inserção do próximo vértice são os dois passos mais importantes do algoritmo. Estes passos devem garantir que a malha continue válida e com boa qualidade.

Como critério de escolha da próxima *aresta base*, geralmente é utilizada a menor aresta da frente. Em três dimensões o triângulo com menor altura da frente é escolhido

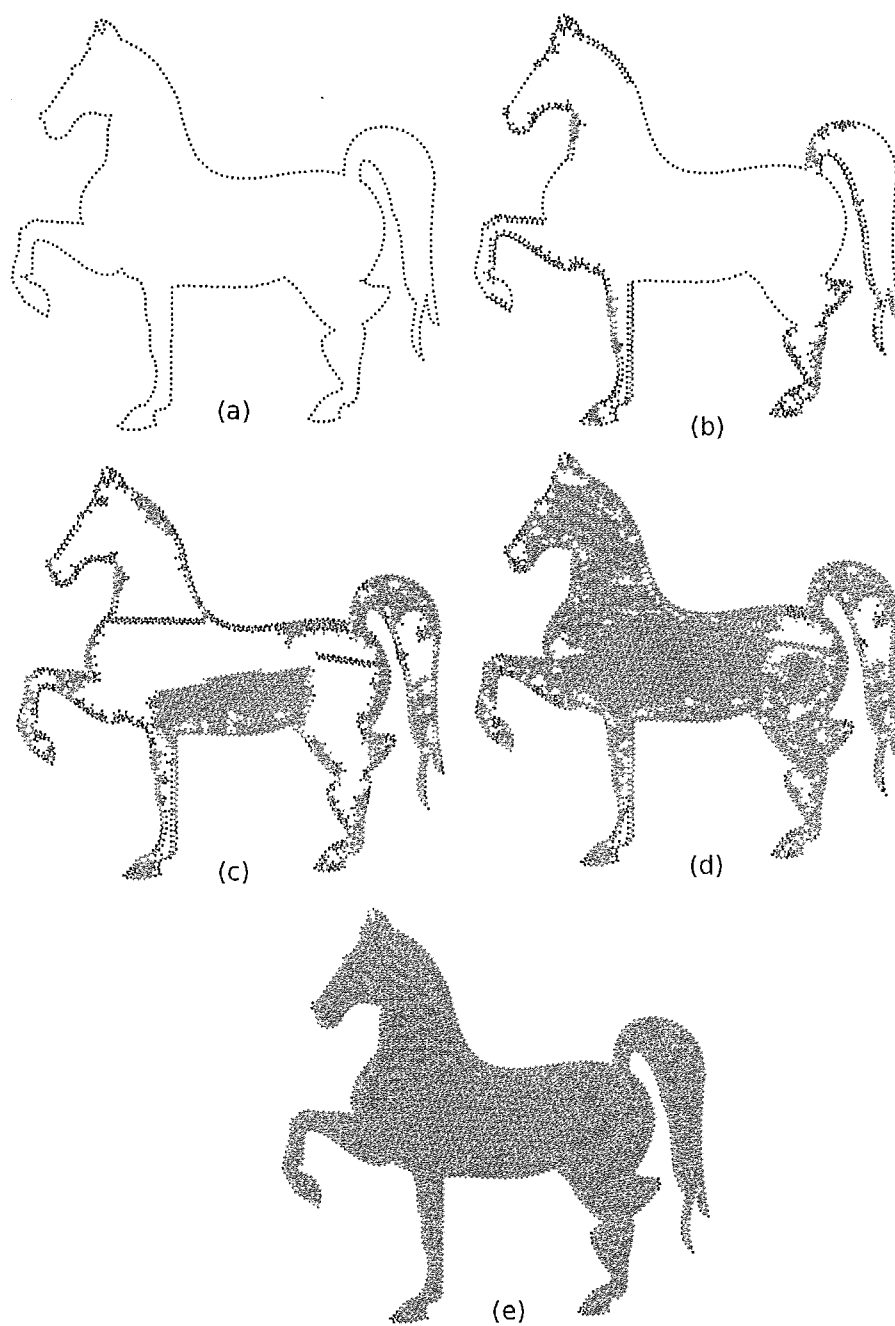


Figura 2.11: O modelo em vários passos durante a triangulação utilizando “Advancing Front”. (a) passo 1; (b) passo 500; (c) passo 2000; (d) passo 4000; (e) malha final após 4721 passos.

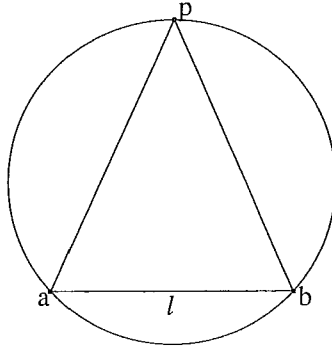


Figura 2.12: Escolha do ponto  $p$  como extensão da frente.

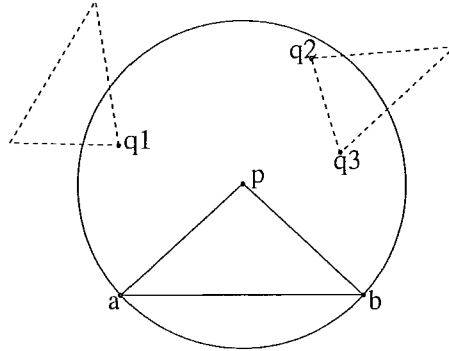


Figura 2.13:  $p$  não é ideal porque existem outros vértices dentro do círculo de raio  $\delta$  e centro em  $p$ .

como *face base*.

A seguir, será descrito como Peraire et al. [20] elaborou o processo de decisão do local de inserção do ponto  $p$  para avançar a frente.

Como ilustrado na Figura 2.12,  $\overline{ab}$  é a aresta da frente escolhida e  $l$  o comprimento da aresta. O novo ponto  $p$  deve ser inserido a uma distância  $\delta$  dos pontos  $a$  e  $b$  da *aresta base*. Esta distância é calculada da seguinte forma:

$$\delta = \begin{cases} 1, & \text{se } 0.55l < 1 < 2l; \\ 0.55l, & \text{se } 0.55l > 1; \\ 2l, & \text{se } 1 > 2l. \end{cases}$$

O ponto  $p$  será ideal se não houverem outros vértices da triangulação nas proximidades. Neste caso a frente avança sem problemas pois ainda está afastada das outras. Porém, se existirem vértices dentro do círculo de raio  $\delta$  e centro em  $p$ , como mostra a Figura 2.13, um novo ponto deve ser definido.

Os pontos contidos pelo círculo são ordenados pelos tamanhos dos raios dos circuncírculos dos triângulos  $\Delta abq_i$ , como mostrado na Figura 2.14. A ordem de preferência neste caso é  $\{q_3, q_1, q_2\}$ . O primeiro vértice válido desta lista é considerado como terceiro vértice do novo triângulo. Um vértice é válido quando o triângulo  $\Delta abq$  não intersecta nenhum outro triângulo da malha. Neste caso há uma junção das frentes, pois um vértice já existente da malha é utilizado.

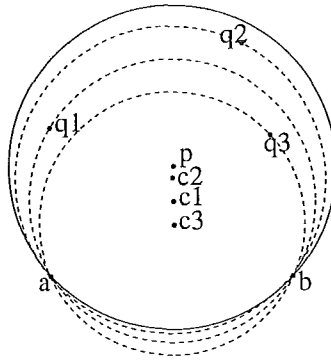


Figura 2.14: Os centros  $c_1$ ,  $c_2$  e  $c_3$  dos círculos formados por  $a$ ,  $b$  e os vértices  $q_1$ ,  $q_2$  e  $q_3$  respectivamente.

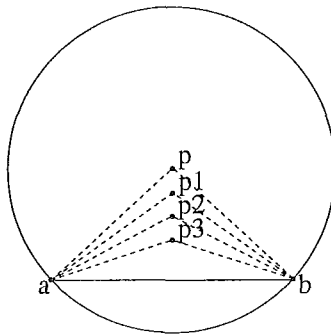


Figura 2.15: Seqüência de pontos igualmente espaçados de  $p$ . O primeiro ponto válido será escolhido para avançar a fronteira.

Se nenhum vértice da lista for válido o próprio ponto  $p$  é utilizado. Porém, se o ponto  $p$  também for inválido, Péraire sugere utilizar o primeiro ponto válido de uma seqüência igualmente espaçada, como ilustrado na Figura 2.15.

Em três dimensões o esquema é similar. Seja  $\triangle abc$  a face base escolhida, e seja  $m$  o seu centróide. O ponto ideal  $p$ , como mostra a Figura 2.16, é criado respeitando a seguinte equação:

$$|\overline{ap}| + |\overline{bp}| + |\overline{cp}| = 3,$$

Desta forma, o comprimento médio é mantido igual à 1. Seja  $\delta$  a aresta de maior comprimento entre  $|\overline{ap}|$ ,  $|\overline{bp}|$  e  $|\overline{cp}|$ . Para que o ponto  $p$  seja realmente ideal, é necessário que não hajam outros vértices da malha dentro da esfera de raio  $\delta$  e centro em  $p$ .

Caso contrário, para cada vértice  $q_i$  dentro da esfera é definido um ponto  $c_i$  equidistante de  $q_i$  e  $d$ . O ponto  $d$ , por sua vez, é aquele entre  $a$ ,  $b$  e  $c$  mais distante de  $p$ . A ordem de preferência, no caso da Figura 2.17 é  $\{q_3, q_1, q_2\}$ . O primeiro vértice válido da lista é utilizado como ponto ideal para criar o novo tetraedro. Caso nenhum seja válido, novamente o ponto  $p$  é utilizado. Caso este também não seja válido o processo utilizado em duas dimensões é repetido até que um vértice válido seja encontrado.

Outros métodos utilizam alternativas para escolha do novo ponto da frente procurando garantir a qualidade da malha. Por exemplo, Li et al. [13, 14] faz uso da técnica de *empacotamento de esferas*, enquanto outros aproveitam as vantagens de Delaunay [17,

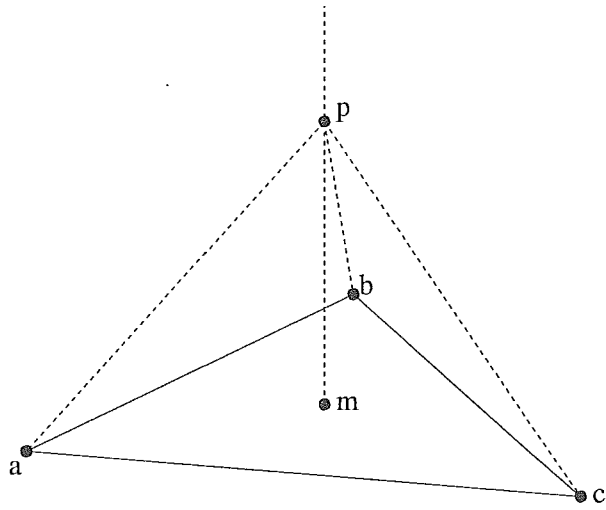


Figura 2.16: Escolha do ponto ideal  $p$  para o avanço da face definida por  $a$ ,  $b$  e  $c$  e centróide  $m$ .

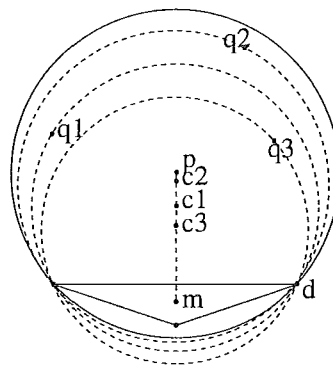


Figura 2.17: Escolha do ponto de junção quando existem vértices da malha dentro da esfera de raio  $\delta$  e centro em  $p$ .



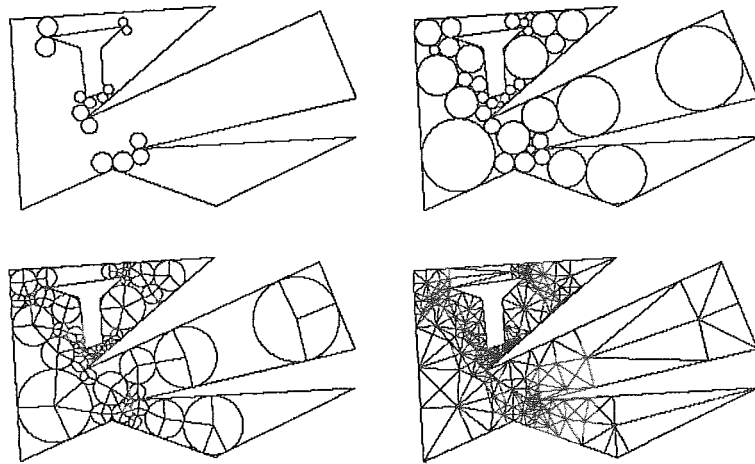


Figura 2.18: (a) Colocação dos círculos nos vértices do polígono. (b) Preenchimento do domínio com círculos. (c) Inserção das arestas. (d) Triangulação final do polígono.

10, 11].

No entanto, todos métodos são fortemente dependentes da poligonização inicial da fronteira. Se houverem triângulos mal formados na frente inicial, as chances de tetraedros mal formados serem gerados são grandes.

Diferentemente do *refinamento de Delaunay*, este método não possui fortes garantias teóricas para suportá-lo. Particularmente, Bern limita o ângulo máximo da malha em  $120^\circ$  mas não apresenta conclusões sobre o ângulo mínimo.

## 2.4 Empacotamento de Círculos

Outra técnica que obteve resultados expressivos na área de triangulação de malhas, foi a de “*Empacotamento de Círculos*” (Circle Packing), introduzida por Bern [2]. Este esquema faz uso de uma subdivisão recursiva de discos para triangular o domínio.

O algoritmo é dividido em duas etapas. Na primeira, o domínio é dividido em círculos não intersectantes. Na segunda etapa, arestas são inseridas entre os centros dos círculos e os pontos de tangência entre os círculos e a fronteira do polígono. Depois, vértices são inseridos para triangular o polígono particionado. A Figura 2.18 ilustra uma seqüência de passos do algoritmo.

A primeira etapa, por sua vez, é subdividida em três passos. No primeiro passo, discos são adicionados nas quinas do polígono. Para todo ângulo convexo do polígono, um círculo é posicionado, e para todo ângulo côncavo, dois círculos são adicionados. A Figura 2.19 ilustra os dois casos. Todos os cantos do polígono ficam isolados por regiões com três ou quatro lados, sendo estes lados arcos ou segmentos retos.

O segundo passo da primeira etapa consiste em conectar as cavidades do polígono à fronteira externa inserindo novos círculos. O terceiro por sua vez, reduz todas as regiões não cobertas por círculos, para regiões com três ou quatro lados.

Na segunda etapa do algoritmo, o polígono particionado é triangulado. Esta etapa é tratada com três casos diferentes. O primeiro caso trata das regiões contendo vértices

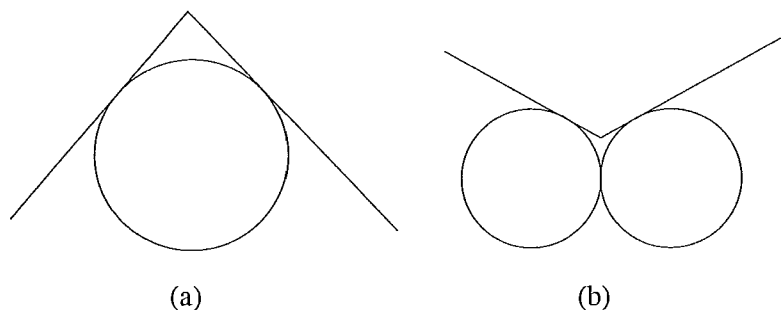


Figura 2.19: Discos adicionados nos cantos convexos (a), e nos côncavos (b).

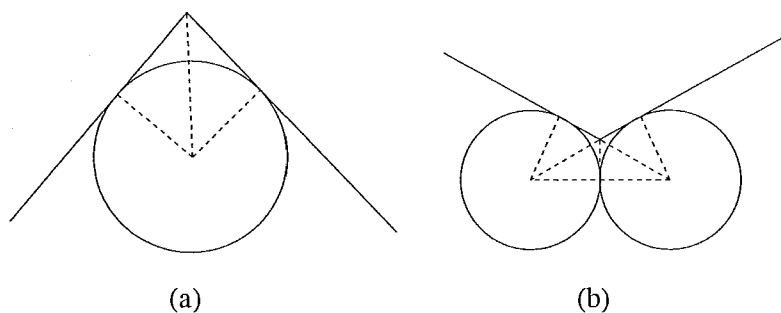


Figura 2.20: Triangulação dos cantos convexos (a), e dos cantos côncavos (b).

do polígono. Este caso corresponde a triangular as regiões tratadas no primeiro passo da primeira etapa. Ele é facilmente tratado como ilustra a Figura 2.20.

Os casos de regiões com três lados são tratados de duas maneiras distintas. No caso da região conter um lado reto, ou arco de ângulo infinito como apresentado por Bern, a configuração (a) da Figura 2.21 é utilizada. Os centros dos círculos e os pontos que tangenciam o segmento são ligados formando uma quadrilátero. Os centros dos círculos, mais o ponto tangente entre os dois círculos, são ligados ao centro do segmento reto. Quatro triângulos são formados neste processo. Quando a região possui três arcos como lados, um círculo  $C$  inscrito pelos pontos tangenciais entre os três círculos é definido. O centro de  $C$  é ligado aos pontos tangenciais e aos centros dos círculos, como pode ser visto na Figura 2.21 (b). Neste caso, seis triângulos são criados.

As regiões com quatro lados são tratadas de forma análoga na maioria das vezes. Todos pontos tangenciais, e os centros dos círculos, são ligados ao centro do círculo  $C$ , novamente definido entre os quatro círculos. Este caso é ilustrado na Figura 2.22, e 16 triângulos são criados. Bern explica que em algumas configurações não é possível triangular regiões com quatro lados desta forma. Neste caso, novos círculos são inseridos para completar a malha triangular.

## 2.5 Empacotamento de Esferas

Shimada [27] estendeu a idéia do “Empacotamento de Círculos” para três dimensões. Ele criou um método chamado “*Bubble Mesh*”. O algoritmo é dividido em duas etapas: colocação das esferas e triangulação a partir dos seus centros.

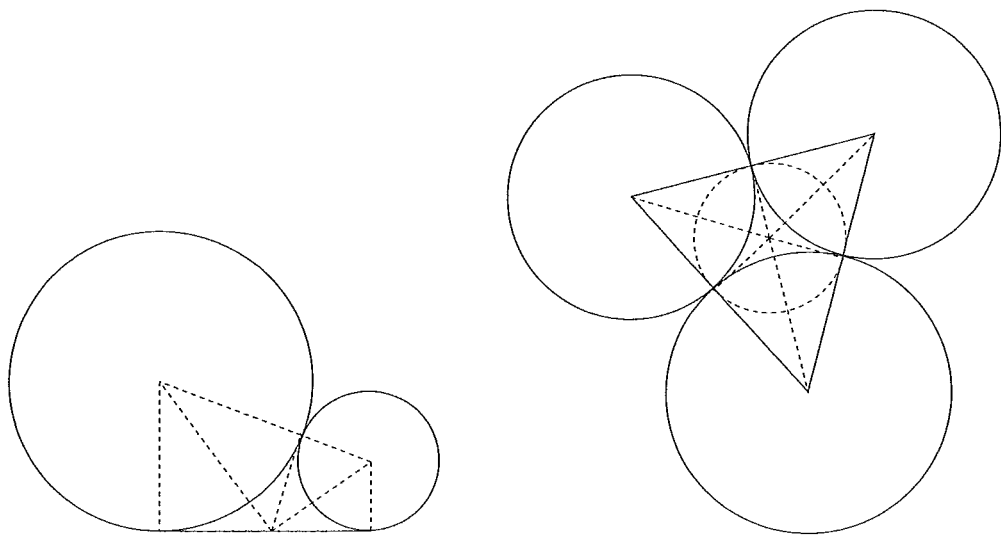


Figura 2.21: Triangulação das regiões com três lados.

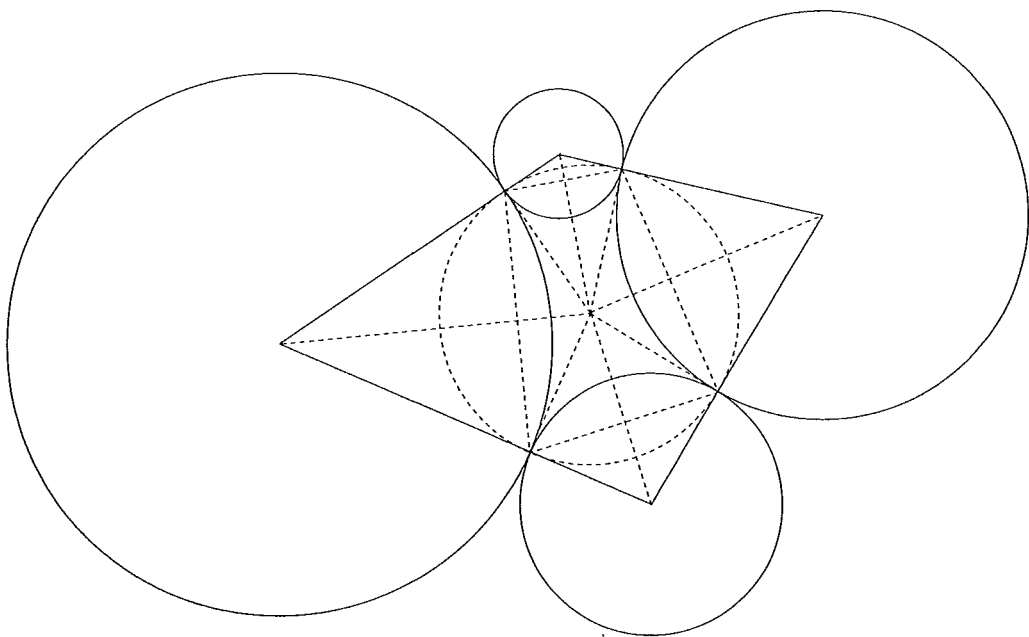


Figura 2.22: Triangulação das regiões com quatro lados.

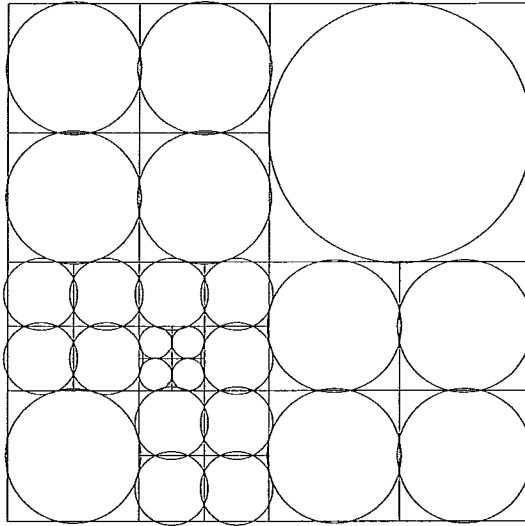


Figura 2.23: Distribuição dos círculos em uma *quadtree*.

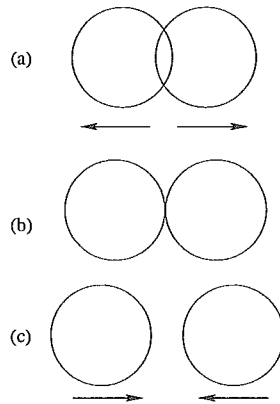


Figura 2.24: Forças entre esferas. (a) Forças de repulsão; (b) forças estáveis; (c) forças de atração.

No primeiro passo esferas são colocadas no domínio. O local inicial é determinado por uma subdivisão espacial, como uma *octree* por exemplo. A Figura 2.23 ilustra um exemplo da distribuição inicial em duas dimensões utilizando uma *quadtree*.

Para melhorar a distribuição inicial, Shimada utilizou um sistema físico para rearrumar as esferas. São utilizadas forças de atração e repulsão para empacotar as esferas de forma mais eficiente, como ilustrado na Figura 2.24.

No segundo passo do algoritmo os centros das esferas são interligados utilizando uma triangulação de Delaunay Restrita. O autor coloca que a inovação do seu método está no fato das esferas imitarem um padrão de Voronoy que corresponde a uma triangulação de Delaunay com elementos bem formados. Este fato é ilustrado na Figura 2.25.

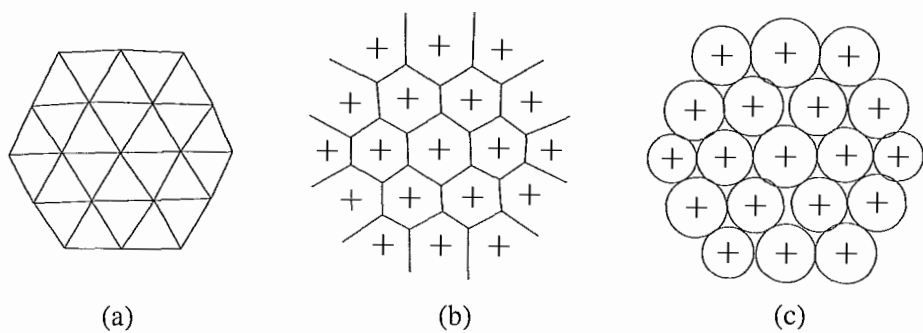


Figura 2.25: Qualidade da malha. (a) Triangulação de Delaunay; (b) Diagrama de Voronoy; (c) Distribuição das esferas.

# Capítulo 3

## Subdivisão Espacial

Para gerar os elementos da triangulação o primeiro passo do algoritmo consiste em subdividir o espaço que envolve o modelo. Em duas dimensões é gerada uma malha triangular e em três dimensões uma malha tetraedral. Esta malha contém os elementos do complexo simplicial da triangulação final. Como será mostrado nos próximos capítulos, os elementos desta subdivisão espacial serão rearrumados de forma a se adequarem à fronteira do modelo. No entanto, nenhum elemento novo é inserido após a fase de subdivisão.

Existem alguns pontos importantes na escolha da estrutura da malha. Em primeiro lugar, ela deve ser adaptativa. Em segundo, deve suportar multi-resolução, e em terceiro, deve gerar elementos com ângulos bons.

Uma estrutura adaptativa permite que coexistam elementos de tamanhos e formas diferentes na malha. Esta propriedade é necessária para que se obtenham elementos menores somente nos locais onde uma aproximação melhor é mais importante como, por exemplo, nas imediações da fronteira do modelo. Desta forma, obtém-se uma estrutura menos densa, economizando elementos onde possível.

A segunda propriedade importante é o suporte a multi-resolução. Deve ser possível gerar a subdivisão espacial de um mesmo modelo em diferentes níveis de resolução. A escolha do nível implica na quantidade de elementos gerados na malha e, conseqüentemente, na qualidade final da aproximação do modelo. Resoluções altas geram malhas densas e aproximações melhores, enquanto resoluções baixas geram malhas mais leves porém aproximações mais grosseiras.

A terceira característica desejada é que os elementos gerados na malha não possuam ângulos pequenos. Quanto menores os ângulos da subdivisão espacial mais chances de obter elementos com formas ruins ao final da compressão. Como será visto no Capítulo 7, um dos objetivos durante a compressão é manter, o máximo possível, a forma inicial dos elementos. Para tanto, é importante que a malha inicial os forneça numa configuração “arredondada”, isto é, que possuam somente ângulos “gordos”, entre  $45^\circ$  e  $90^\circ$ .

### 3.1 Triangulações 4-8

Para modelos de 2 dimensões são utilizadas triangulações 4-8 [32, 29, 28, 30, 31]. Esta estrutura possui a vantagem de combinar propriedades estruturais de malhas triangulares e quadrilaterais. Os triângulos gerados são metades ou quartos de quadrados, implicando que todos os ângulos são de  $45^\circ$  e  $90^\circ$ .

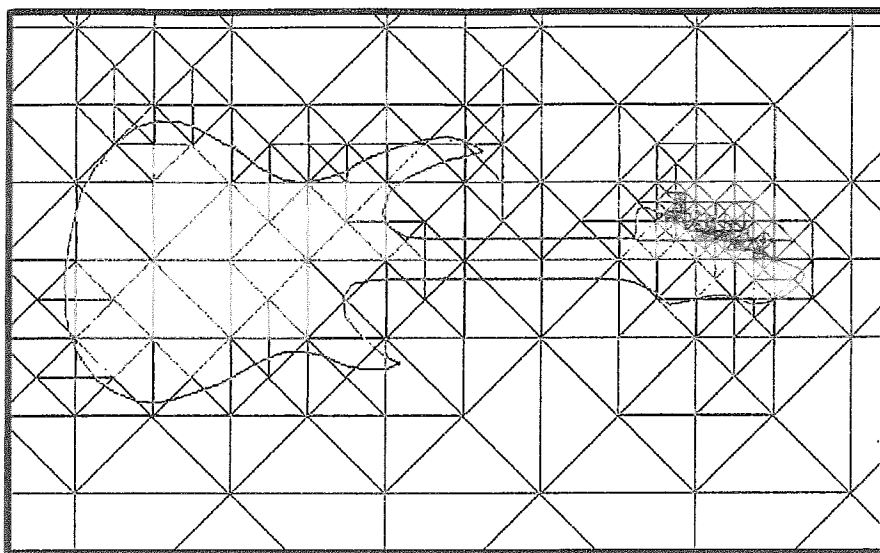


Figura 3.1: Uma malha 4-8 gerada sobre um modelo de uma guitarra. O refinamento local gera triângulos menores perto dos detalhes da malha.

Em sua mais baixa resolução, a triangulação 4-8 possui apenas dois triângulos formando um quadrado. O critério utilizado para subdividir uma aresta é proporcional a sua distância à fronteira do modelo. Isto provoca um refinamento mais denso perto da fronteira, onde a aproximação ocorre:

Para subdividir uma aresta em particular, os seguintes critérios devem ser respeitados:

- o nível máximo de subdivisão não ter sido alcançado,
- a distância do ponto médio da aresta à fronteira deve ser maior do que uma determinada tolerância, e
- seu comprimento deve ser maior do que a distância do seu ponto médio à fronteira.

O nível máximo de subdivisão e a tolerância são definidos pelo usuário. Estes parâmetros possuem uma grande influência na resolução da malha, e conseqüentemente no número de triângulos gerados. O nível máximo impõe um limite à resolução da malha, enquanto que a tolerância evita que arestas muito próximas à fronteira sejam subdivididas muitas vezes sem necessidade.

Para cobrir melhor os detalhes da fronteira, um esquema de subdivisão local é utilizado. Este esquema é realizado em duas partes distintas. Na primeira, todo triângulo contendo mais do que um ponto do modelo é subdividido. Na segunda, os triângulos que são intersectados por mais de um segmento do modelo também são subdivididos. Estes refinamentos devem respeitar um limite máximo para a subdivisão local, que, por sua vez, deve ser maior do que o limite máximo imposto na subdivisão global, definido anteriormente.

A Figura 3.1 mostra uma malha 4-8 gerada a partir de um modelo de uma guitarra. Perto das tarraxas, onde existe uma concentração maior de pontos do modelo, o refinamento local gerou triângulos menores.

## 3.2 Multi-Triangulação Binária

Em 3 dimensões é utilizada uma Multi-Triangulação Binária (BMT) [18] para suportar a triangulação. O conceito da BMT é uma extensão das estruturas de multi-resolução de 2 dimensões, como a 4-8 citada acima, para 3 dimensões.

A estrutura inicial de uma BMT é um cubo dividido em seis tetraedros com ângulos diedrais de  $45^\circ$ . Os critérios utilizados na subdivisão são diferentes da estrutura em 2 dimensões, mas permanece o propósito de uma subdivisão mais refinada perto da fronteira do modelo. Para subdividir um tetraedro em particular a distância do seu centróide à fronteira deve ser menor do que a média dos comprimentos de suas arestas. Assim como antes, um número máximo de subdivisões é imposto pelo usuário.

Este tipo de estrutura também possui a vantagem de produzir uma transição gradual entre os elementos de níveis de refinamento diferentes. Um exemplo desta transição entre os diferentes níveis de refinamento é apresentado na Figura 3.2.



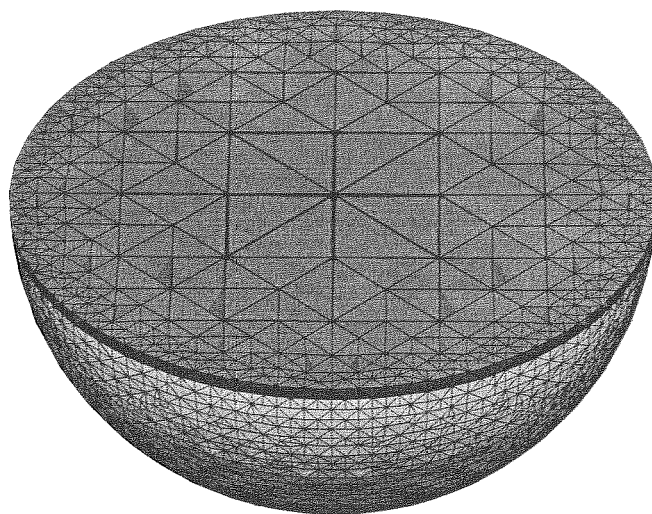
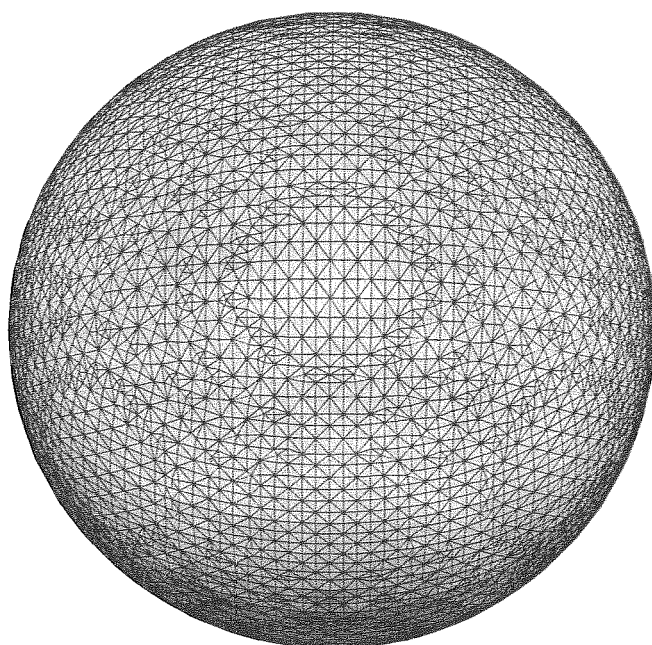


Figura 3.2: Uma esfera com elementos menores perto da fronteira e maiores no interior. A malha exibida já passou pelo processo de compressão.

# Capítulo 4

## Modelos Tratados

Modelos geológicos geralmente possuem características que só podem ser representadas por estruturas de dados complexas. Estes modelos podem conter várias regiões, falhas, cavidades, e normalmente não são variedades. As falhas geológicas representam descontinuidades na terra e dividem o modelo em várias regiões. Grupos de regiões com atributos similares formam camadas geológicas. Além disso, modelos que representam bacias sedimentárias podem se estender por vários quilômetros nos eixos da superfície mas ter uma espessura relativamente pequena. Um exemplo de um modelo geológico pode ser visto na Figura 4.1. Um corte bidimensional deste modelo é mostrado na Figura 4.2.

### 4.1 Representação de Polígonos em 2D

Uma curva poligonal é representada por duas estruturas: uma lista de vértices e uma árvore binária contendo seus segmentos. Como um modelo pode ser dividido em várias regiões uma árvore é necessária para cada região ou curva diferente. Cada nó da árvore contém um ou mais segmentos adjacentes de uma região, representando uma parte da curva total. Em um nó são armazenados ponteiros para o primeiro e o último vértices do seu fragmento da curva. O nó raiz, por exemplo, cobre toda região, por isso possui

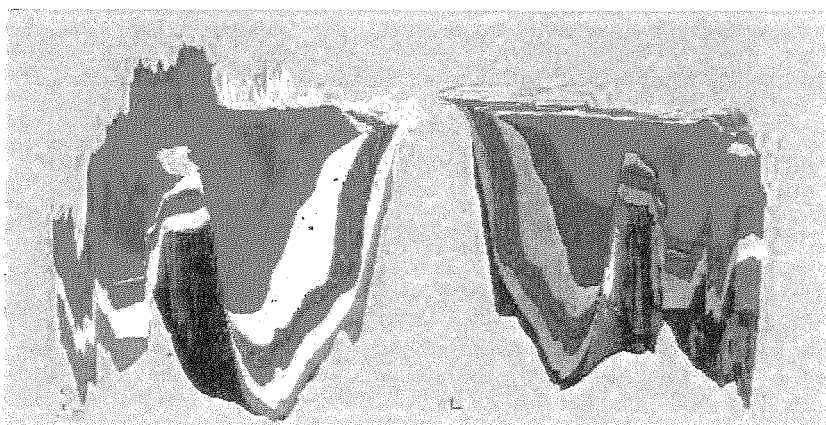


Figura 4.1: Modelo em 3 dimensões da Bacia do Golfo do México.

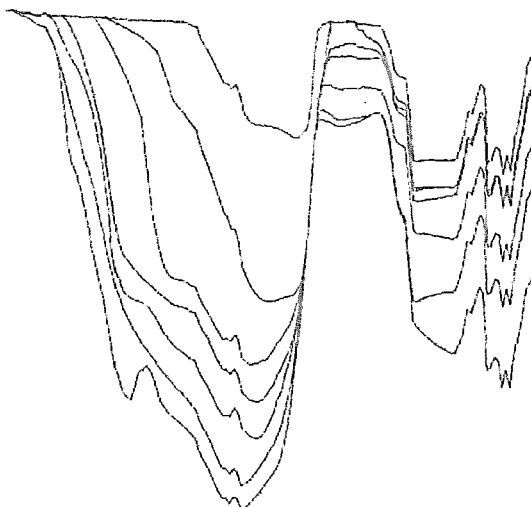


Figura 4.2: Um corte bidimensional do modelo do Golfo do México.

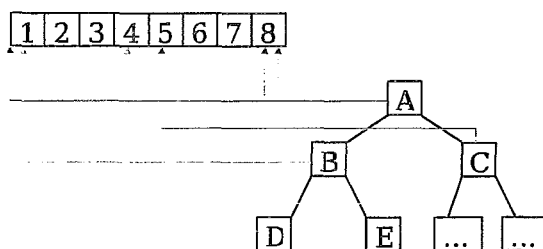


Figura 4.3: Árvore binária representando uma curva poligonal com oito vértices.

ponteiros para o primeiro e último nós da curva poligonal. Caso a curva seja fechada, o nó raiz apontará duas vezes para um mesmo vértice. Nós filhos representam as metades de seu nó pai.

Considerando uma curva aberta com oito vértices, por exemplo, o nó raiz possui dois ponteiros para o primeiro vértice e o oitavo. O filho da esquerda apontará para os vértices 1 e 4, e o da direita para os vértices 5 e 8. Descendo na árvore, os netos do nó raiz possuem quartos de seu segmento total, e assim por diante, como ilustra a Figura 4.3. A união de todos nós em um determinado nível da árvore produz a curva poligonal inteira.

Para otimizar a função distância, descrita no Capítulo 5, também é armazenada a *distância máxima* dentro de um nó. Este valor refere-se à distância máxima de todos os vértices dos segmentos de um nó ao seu *segmento central*. O *segmento central*, por sua vez, é aquele definido pelo primeiro e último vértices do nó.

A curva poligonal deve ser orientada, mas são aceitas tanto curvas no sentido horário quanto no anti-horário. Um teste utilizando a área do polígono é efetuado para testar sua orientação. Se a área for positiva, o polígono está no sentido anti-horário, no caso contrário ele está no sentido horário. Esta orientação também se fará necessária nos próximos passos do algoritmo.

## 4.2 Representação de Modelos em 3D

Para representar adequadamente modelos em 3 dimensões, é utilizada a estrutura de dados *radial – edge* (RED) [33]. Ela permite obter qualquer relação de adjacência em tempo constante. As estruturas de dados dos modelos são criadas utilizando a biblioteca do sistema *CGC* [5]. Esta biblioteca permite efetuar todas operações necessárias com o modelo original.

# Capítulo 5

## Função Distância

Durante toda execução do algoritmo a distância de um ponto no espaço à fronteira do modelo é computada diversas vezes. Portanto se faz necessária uma função distância rápida.

### 5.1 Função Distância em 2D

A função distância escolhida para buscas em 2 dimensões é baseada em [9, 12], e utiliza a árvore binária da representação poligonal.

Para computar a menor distância de um ponto  $p$  a uma curva poligonal, a árvore binária é percorrida. A *distância máxima* armazenada em cada nó, define sua caixa envolvente, como ilustrado na figura 5.1.

Uma estrutura auxiliar é criada para armazenar os nós já visitados da árvore. Particularmente, esta estrutura é uma pilha onde os elementos são inseridos de forma ordenada. O critério de ordenação é a distância do ponto  $p$  à caixa envolvente do nó. Esta distância é computada subtraindo a *distância máxima* do nó da distância de  $p$  ao *segmento central* do nó.

Quando um nó é visitado, ele é removido da pilha. O critério de ordenação é computado para seus dois filhos, que são inseridos na pilha. O algoritmo continua visitando o nó com menor distância, situado no topo da pilha, até que um nó sem filhos seja encontrado. Um nó sem filhos, ou nó folha, é aquele que contém apenas um segmento e, portanto, a distância exata do ponto ao segmento mais próximo é encontrada. Como a distância utilizada como critério de ordenação é uma super estimativa da distância exata do ponto aos segmentos, é garantido que nenhum outro segmento na pilha terá distância menor do que a encontrada.

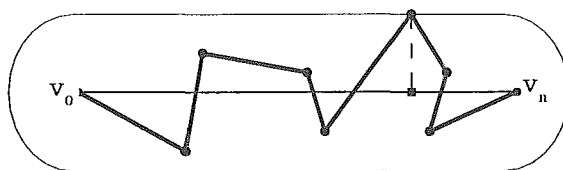


Figura 5.1: A região envolvente de um nó, definida pela *distância máxima* (linha pontilhada). O *segmento central* é definida pelos vértices  $v_0$  e  $v_n$ .

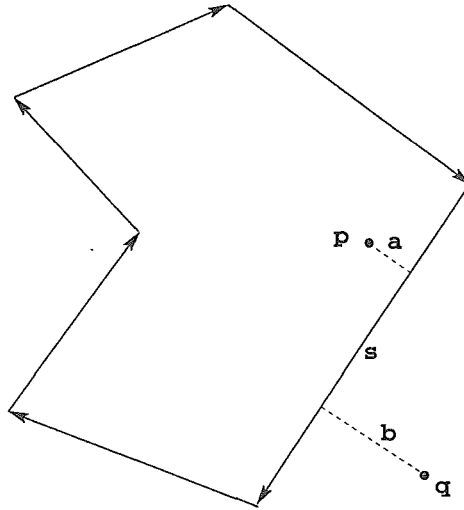


Figura 5.2: O ponto  $p$  é interior à região, e o ponto  $q$  exterior. O segmento  $a$  é definido pelo ponto  $p$  e o ponto mais próximo no segmento  $s$ . O segmento  $b$  é definido da mesma forma para o ponto  $q$ .

No entanto, é necessário que a função distância compute valores positivos no interior das regiões e valores negativos no exterior das mesmas. Para isso, um teste de ponto dentro de região é realizado utilizando a orientação do polígono. O teste consiste em comparar o resultado do produto vetorial do segmento encontrado,  $s$ , e do segmento  $a$  definido pela menor distância de  $p$  a  $s$ . A Figura 5.2 mostra dois resultados possíveis. Os produtos vetoriais calculados são os seguintes:

$$s \times a = (s_x * a_y) - (s_y * a_x) < 0 \quad (5.1)$$

e

$$s \times b = (s_x * b_y) - (s_y * b_x) > 0. \quad (5.2)$$

Quando o ponto mais próximo no segmento é um vértice do polígono, podem ocorrer inconsistências, especialmente quando o ponto mais próximo for um vértice entre dois segmentos formando um ângulo interno menor que  $90^\circ$ . A Figura 5.3 mostra um caso onde dois pontos,  $p$  e  $q$ , fora da região possuem classificações diferentes utilizando o teste descrito acima.

Para resolver este caso particular, um ponto  $r$  é definido dentro do triângulo formado pelos dois segmentos mais próximos,  $s$  e  $u$ . A Figura 5.4, mostra o triângulo formado pelos dois segmentos, o ponto interior  $r$ , e o segmento da distância mínima  $c$ . A função distância é calculada para este ponto interior que terá sinal inverso ao dos pontos  $p$  e  $q$ . Portanto, se  $r$  for interior,  $p$  e  $q$  são exteriores, e vice-versa.

Ao lidar com modelos contendo várias regiões, a distância a um ponto é computada para cada região separadamente. Cavidades em regiões são determinadas utilizando a função distância. Pontos dentro de cavidades são considerados exteriores à região.

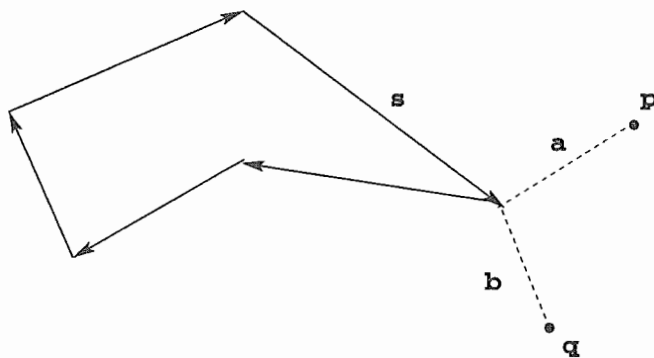


Figura 5.3: Os pontos  $p$  e  $q$  são exteriores à região mas o sinal da função distância será diferente por estarem mais próximos de um vértice e os dois segmentos mais próximos,  $s$  e  $u$ , formarem um ângulo interno menor do que  $90^\circ$ .

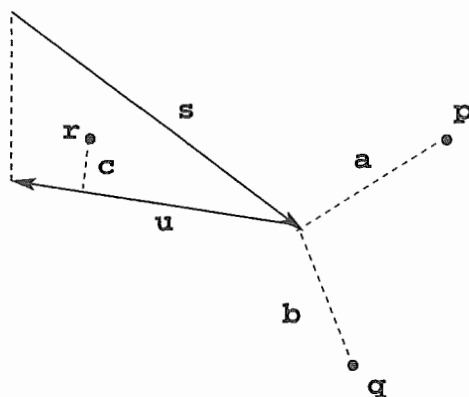


Figura 5.4: O ponto  $r$  é definido como interior ao triângulo formado pelos dois segmentos  $s$  e  $u$ . O segmento  $c$  é calculado pela função distância aplicada a  $r$ .

## 5.2 Função Distância em 3D

Em 3 dimensões a representação das regiões e a função distância são realizadas de forma diferente. Uma estrutura auxiliar é utilizada mais uma vez, visando tornar esta função mais rápida. Neste caso uma *octree*.

O nó raiz da *octree* é definido pelas coordenadas da caixa envolvente do modelo em questão. Para criar um estrutura compacta, as coordenadas dos nós internos e folhas não são armazenadas, mas computadas em tempo real ao atravessar a árvore. Os únicos atributos mantidos em um nó interno são ponteiros para seus oito filhos, enquanto um nó folha armazena somente uma lista de objetos intersectados pela sua caixa.

Especificamente, a lista de itens contém todas as faces do modelo que intersectam ou estão contidas na caixa do nó folha. Quando o número de elementos desta lista ultrapassar um limite máximo pré-determinado, o nó é subdividido. No entanto, para que um nó não seja repartido indefinidamente, a altura da árvore também é limitada. Note que se existisse um vértice com mais faces incidentes do que o tamanho limite da lista, o nó contendo este vértice seria subdividido eternamente.

Para encontrar a menor distância de um ponto  $p$  à superfície do modelo, a árvore é atravessada começando pelo nó raiz. Durante a execução do algoritmo uma estimativa do melhor resultado é armazenada. Quando um nó folha é visitado a distância para a face mais próxima é retornada. Quando um nó interno é visitado, os seus filhos são ordenados em ordem ascendente da distância de  $p$  aos seus octantes. Cada octante filho é visitado recursivamente, mas somente se a sua distância correspondente for menor do que a melhor estimativa no momento. Se algum filho retornar um valor menor do que a melhor estimativa, esta é atualizada. A Figura 5.5 expõe um pseudo-código para este algoritmo.

```
global float melhor ← ∞
procedimento distancia (ponto  $p$ , nóOctree  $T$ )
  se  $T$  é um nó folha então
    retorna a distância entre  $p$  e a face mais perto em  $T$ 
  se não
    ordena os filhos de  $T$  em ordem ascendente de distância à  $p$ 
    para cada filho  $S$  (ordenados) faça
      seja  $d$  a distância entre  $p$  e a caixa de  $S$ 
      se  $d < melhor$  então
         $D \leftarrow$  distância ( $p$ ,  $S$ )
        se  $D < melhor$  então  $melhor \leftarrow D$ 
    retorna melhor
```

Figura 5.5: Pseudo-código para a função distância da octree.



## Capítulo 6

### Malha de Deslocamento

Um sistema físico de massa-mola é utilizado para adaptar a malha refinada à fronteira do modelo. O propósito é selecionar um sub-conjunto de elementos da malha, chamado de *malha-d* ou *malha de deslocamento*, e comprimi-lo para que se adeque à fronteira do modelo. Os vértices na fronteira da *malha-d* são marcados para serem projetados na direção da fronteira. Ao movimentar estes vértices marcados as forças geradas são propagadas para os outros vértices, rearrumando todos os elementos interiores.

Na Figura 6.1 é mostrada a *malha-d* e a triangulação final de uma esfera em três resoluções diferentes.

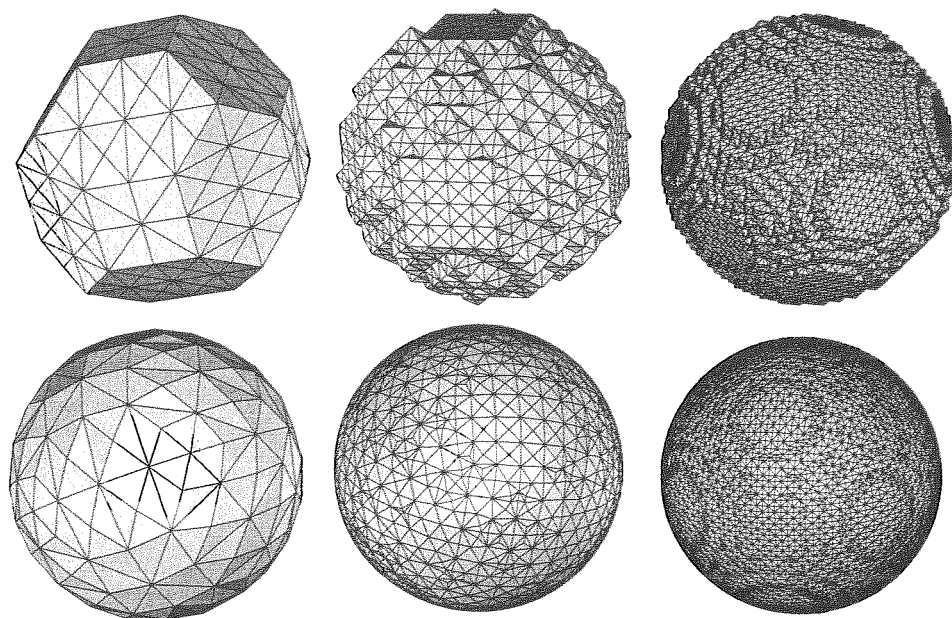


Figura 6.1: A *malha-d* de uma esfera antes e depois da compressão em três níveis de resolução diferentes.

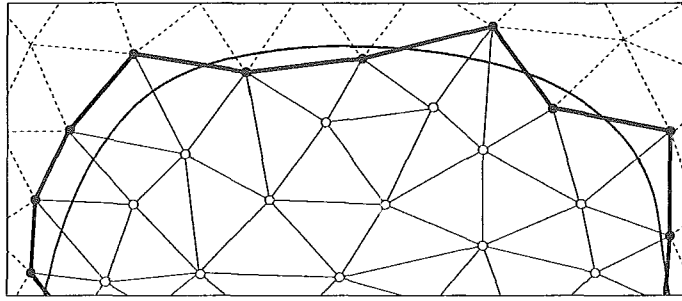


Figura 6.2: Uma representação em duas dimensões da *malha de deslocamento*. Os vértices do *envelope interior* estão desenhados em branco. Em preto, os vértices da fronteira da *malha de deslocamento*, ligados pela linha preta grossa. A curva representa a fronteira original do modelo. As linhas pontilhadas são as arestas descartadas da malha inicial.

## 6.1 Malha de Deslocamento

Alguns vértices da malha devem ser selecionados, ou marcados, para serem empurrados na direção da fronteira do modelo. A solução escolhida em duas dimensões foi a de classificar todos os triângulos como *interior*, *exterior* ou *intersectando a fronteira*. Depois disso são marcados todos os vértices exteriores dos triângulos intersectando a fronteira. Analogamente, todos vértices no interior podem ser escolhidos. Entretanto, alguns vértices marcados podem estar longe da fronteira e, quando empurrados, provocam grande perturbações na malha física. Além disso, alguns triângulos podem ter todos os três vértices marcados, resultando, na maioria dos casos, em triângulos com área zero após a compressão. Por isto é necessário utilizar rotinas para reparar os casos de colapsos de triângulos.

Idealmente, a *malha-d* deve estar o mais próxima possível da fronteira. Seguindo o esquema de Molino [19] para 3 dimensões, a *malha de deslocamento* é composta por todos os tetraedros incidentes em um conjunto de vértices suficientemente interiores ao modelo, chamado de *envelope interior*. Este conjunto contém todos os vértices cujas arestas incidentes estão pelo menos 25% no interior do modelo. Esta condição implica em que todos os tetraedros da *malha-d* possuam no mínimo um vértice no interior da fronteira do modelo.

Todos os vértices da *malha-d* que não pertencem ao *envelope interior* são marcados para serem empurrados em direção à fronteira. Este procedimento garante que nenhum tetraedro terá seus quatro vértices marcados. O conjunto de vértices marcados é uma aproximação grosseira da fronteira da malha. Todos os tetraedros que não pertencem a *malha de deslocamento* são descartados.

Alguns vértices marcados são exteriores à fronteira, e outros interiores mas todos são suficientemente próximos, como pode ser visto na figura 6.2. Quanto maior o deslocamento de um vértice para alcançar a fronteira, maior as forças criadas na malha física. Forças de compressão muito altas causam perturbações na malha, podendo inclusive colapsar alguns elementos.

Para garantir uma compressão suave, dois casos particulares devem ser tratados. Primeiro, arestas internas da *malha-d* com os dois vértices incidentes marcados podem ser achatadas durante a compressão. Uma aresta interna, como mostrado na Figura 6.3, é

### Interior Edges

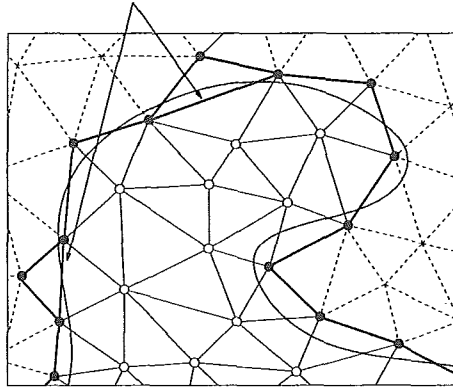


Figura 6.3: As arestas apontadas representam as interiores da *malha-d* conectando dois vértices marcados. A fronteira da *malha-d* está desenhada com segmentos grossos, e a linha curva representa a fronteira do modelo. Para facilitar a visualização a *malha-d* está representada em 2 dimensões, mas o caso para 3 dimensões é análogo.

identificada quando todos os tetraedros do seu operador *estrela*, i.e. os tetraedros incidentes na aresta, pertencem à *malha-d*. O segundo caso refere-se às arestas na fronteira da *malha-d*, com mais de duas faces na fronteira incidentes, ou seja, arestas que não são variedades.

Em um primeiro passo, todas as arestas que se enquadram em um desses casos são subdivididas, e a *malha-d* é computada novamente. Quando uma aresta interna é subdividida, o vértice criado é automaticamente inserido no *envelope interior*. Como a aresta está no interior da malha é certo que o novo vértice também estará.

Em passadas subsequentes, a *malha-d* é novamente verificada. Cada vez que um dos dois casos é identificado o vértice da aresta em questão com menor distância à fronteira da malha é inserido no *envelope interior*. Como a distância é negativa dentro da fronteira, se os dois vértices forem interiores o mais interno é escolhido. Após cada passada a *malha-d* é recomputada, e o processo repetido até que nenhum outro caso seja identificado.

## 6.2 Modelos com Multi-Região

Ao trabalhar com modelos com multi-região, *envelopes interiores* são criados separadamente para cada uma. A Figura 6.4 expõe um exemplo de um modelo contendo mais de uma região. Como os *envelopes interiores* possuem apenas vértices internos das regiões, é garantido que não há intersecção entre envelopes diferentes. Porém, um tetraedro intersectando uma fronteira do modelo pode ser incidente em mais de um *envelope interior*, e ao computar a *malha-d* pode ter os quatro vértices marcados.

Para solucionar este problema é preciso criar uma superfície delimitadora entre regiões adjacentes. Esta superfície deve conter os vértices marcados, e, após a compressão, se constituirá na aproximação da fronteira entre as regiões.

Ao computar os *envelopes interiores* de cada região, as mesmas regras são aplicadas como descrito na Sessão 6.1. Um vértice deve ter todas arestas incidentes pelo menos 25% no interior da fronteira para pertencer ao envelope. Porém, para encontrar a super-

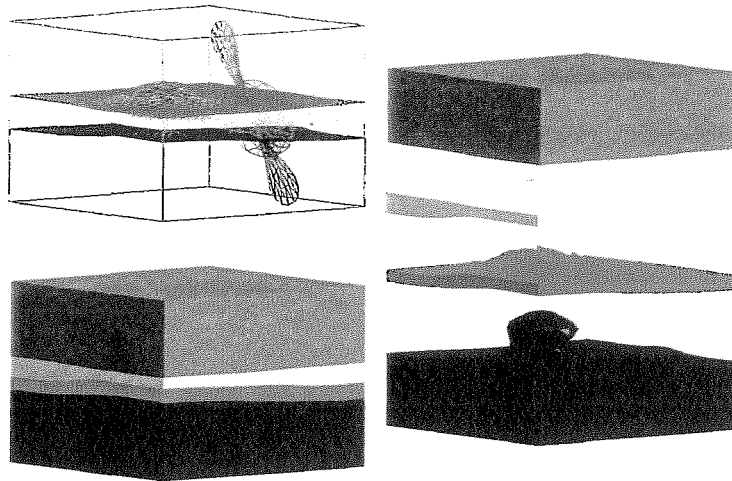


Figura 6.4: Na esquerda, um viso do modelo completo de um domo de sal. Na direita, as regies separadas.

fcie delimitadora, a *malha-d*  computada somente para algumas regies. Para isso  utilizado um critrio simples, baseado na identificao da regio. Ao selecionar os vrtices marcados na fronteira entre duas regies, somente a regio com menor nmero de identificao  considerada. Ao computar a superfcie delimitadora da *malha-d* para uma regio, ela  automaticamente definida para a regio adjacente, como mostrado na Figura 6.5.

Em modelos com multi-regies  importante identificar as faces na fronteira da *malha de deslocamento* para que as arestas internas sejam removidas. Estas arestas so facilmente identificadas como aquelas que possuem os dois vrtices incidentes marcados, e sem incidncia de faces na fronteira. O processo descrito na Sesso 6.1  utilizado para remover essas arestas.

No entanto, no  trivial identificar uma face na fronteira da *malha-d*, ou seja, na regio delimitadora entre duas regies. Para uma face estar na fronteira algumas condies devem ser atendidas. A primeira condio estabelece que todos os seus trs vrtices estejam tambm na fronteira da *malha-d* e, portanto, marcados para serem empurrados para a fronteira do modelo. No entanto, esta condio no  suficiente para garantir que a face esteja na fronteira, como pode ser visto na Figura 6.6. Para garantir que uma face esteja realmente na fronteira  preciso analisar a classificao dos dois tetraedros incidentes nela.

A classificao dos tetraedros  realizada de acordo com as regies que contm os seus vrtices. O principal fato que permite classificar os tetraedros de acordo com a classificao de seus vrtices  que, exceto aqueles marcados (na fronteira da *malha-d*, todos os remanescentes devem pertencer a uma mesma regio. Na Figura 6.5 nota-se que ao remover os vrtices marcados nenhum tringulo cinza possui vrtices na regio branca e vice-versa. Esta mesma propriedade  vlida para modelos em trs dimenses. Portanto, um tetraedro pode ser classificado como pertencente  mesma regio de qualquer um de seus vrtices no marcados.

Aps classificar os tetraedros uma face de fronteira pode ser facilmente identificada

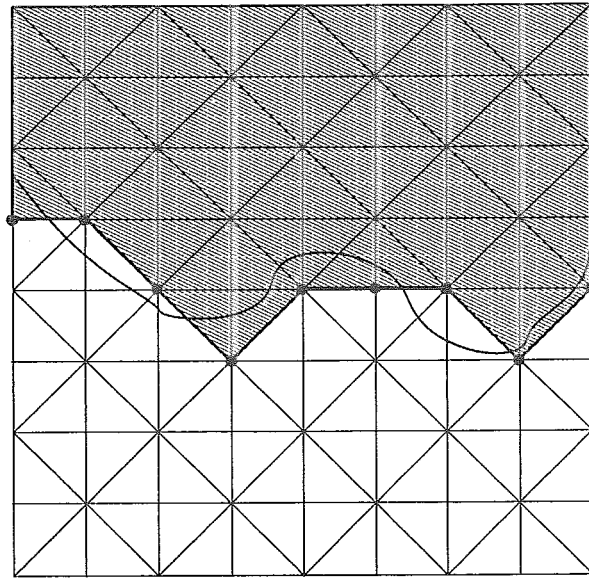


Figura 6.5: Uma visão em duas dimensões da superfície delimitadora (linha grossa) entre as regiões branca e cinza. Os vértices na superfície delimitadora são os vértices marcados para serem empurrados em direção à fronteira, representada pela curva.

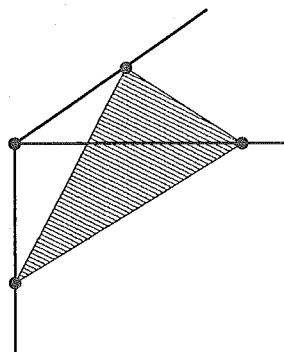


Figura 6.6: Um canto de um cubo com uma face interior com os três vértices fronteira.

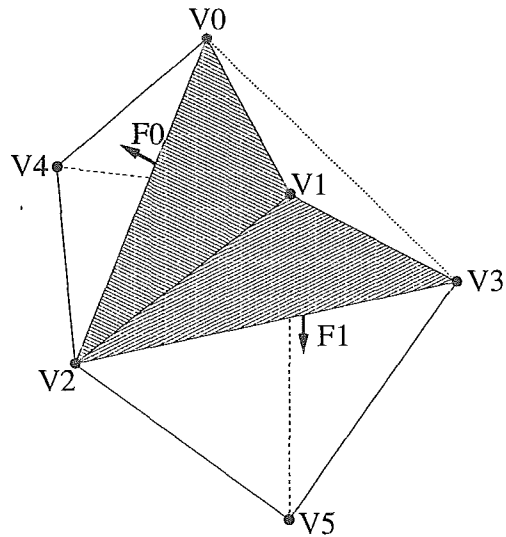


Figura 6.7: Tetraedro exterior com os quatro vértices marcados: V0, V1, V2, e V3. As faces de fronteira são as hachuradas (F0 e F1), e as setas apontam para o interior do modelo.

quando seus dois tetraedros incidentes estão em regiões diferentes.

O esquema de classificação de tetraedros pode falhar para aqueles no exterior do modelo. Estes tetraedros podem ter todos os quatro vértices marcados e, portanto, nenhum vértice livre para associar a classificação. Embora seja uma condição que deve ser evitada, como explicado na Sessão 6.1, estes tetraedros não fazem parte da *malha de deslocamento*, e portanto não são considerados na triangulação final. A Figura 6.7 ilustra um exemplo de um tetraedro exterior com todos os vértices marcados. Para evitar falhas no teste sempre que este caso é encontrado o tetraedro é classificado como exterior, e portanto não pertencente a nenhuma região.

# Capítulo 7

## Sistema Físico

O sistema físico escolhido para comprimir a malha é baseado em um esquema de massas e molas. Esta compressão realiza a aproximação da *malha-d* em relação à fronteira do modelo. Os elementos da *malha-d* são utilizados para representar as molas e partículas do sistema.

### 7.1 Configuração das Molas

Uma questão muito importante é como configurar as molas na malha triangular ou tetraedral para realizar a compressão. Molas mal posicionadas podem ser incapazes de manter a forma dos elementos, ou pior, não impedir que sejam colapsados durante a compressão.

Para o sistema em duas dimensões diversos esquemas diferentes foram testados. Alguns exemplos explorados são exibidos na Figura 7.1.

Porém, é fácil perceber que uma configuração com molas somente nas arestas dos triângulos, por exemplo, não é suficiente para segurar a forma dos elementos. Um vértice pode se deslocar, sem muita resistência, ao centro da sua aresta oposta colapsando o triângulo.

A configuração que apresentou melhores resultados consiste em uma mola por aresta, mais três molas de projeção por triângulo, como ilustrado no item (f) da Figura 7.1. Cada mola de projeção é posicionada entre um vértice da malha e a sua projeção ortogonal na aresta oposta do triângulo. Para isto, é necessário criar alguns vértices auxiliares que não fazem parte da *malha-d*, desenhados em branco na Figura 7.1.

Esta configuração não só apresenta melhores resultados na prática, mas também possui prova teórica de que evita os colapsos com maior eficiência, como demonstrado por Cooper & Maddock [8].

Cada aresta da malha possui uma referência para sua mola correspondente, enquanto cada face possui referências para as suas três molas de projeção correspondentes. Uma mola de aresta é ligada por dois vértices da malha. No entanto, molas de projeção conectam um vértice real da malha, a um vértice virtual.

Para o algoritmo em três dimensões foi utilizado uma configuração análoga aquela escolhida em duas dimensões. As arestas da malha referenciam suas molas correspondentes, enquanto que cada face referencia suas duas molas de projeção, como mostrado na Figura 7.2.

Para computar as forças nas molas, duas equações distintas são utilizadas: uma para as

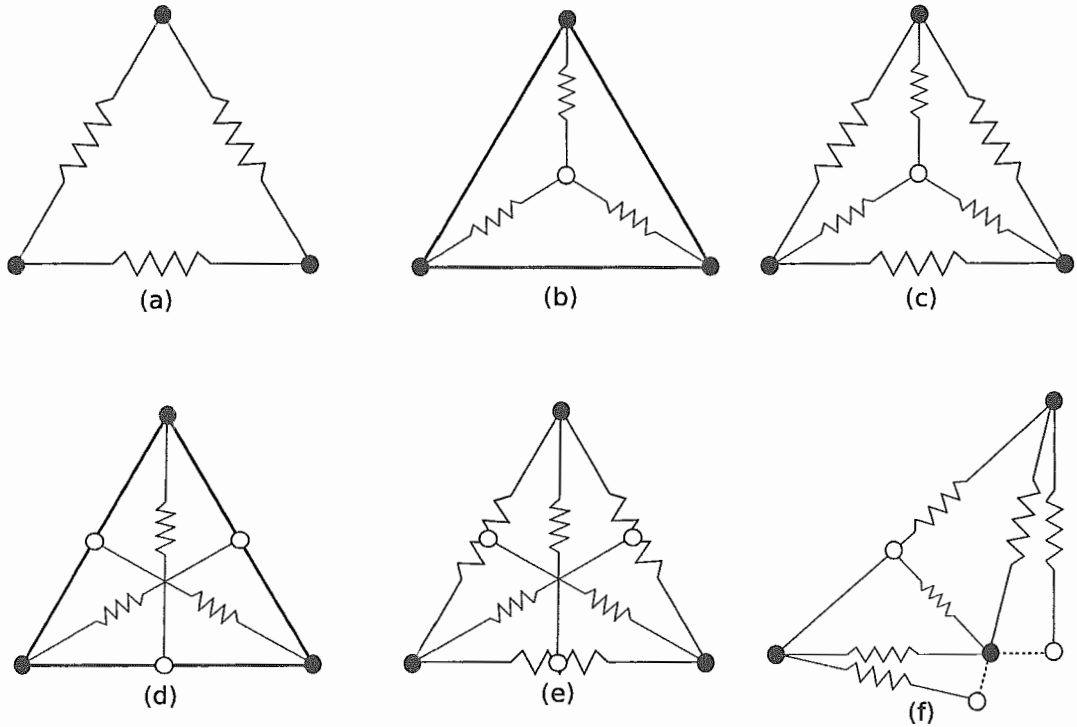


Figura 7.1: Seis esquemas diferentes para configuração do sistema de massa-mola na malha: (a) somente molas nas arestas da malha, (b) somente molas dos vértices ao centróide do triângulo, (c) molas nas arestas e nos centróides, (d) somente molas nas medianas do triângulo, (e) molas nas arestas e nas medianas, (f) molas nas arestas e nas projeções dos vértices às arestas opostas. Em preto os vértices reais da malha, em branco os vértices virtuais.

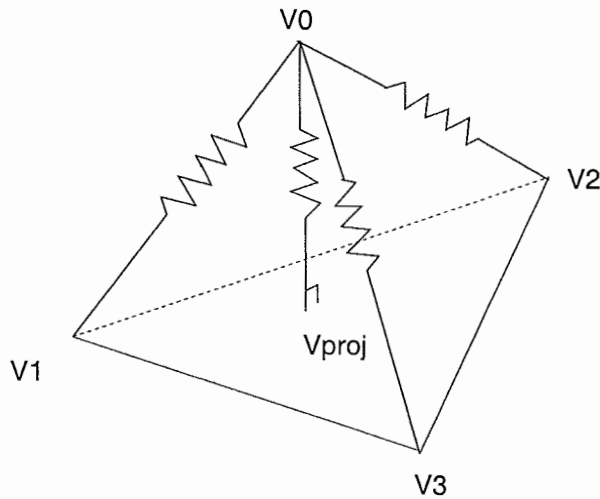


Figura 7.2: Molas incidentes no vértice  $v_0$ . Três molas de aresta e uma mola de projeção. As outras molas do tetraedro, três de projeção e três de aresta, não estão ilustradas nesta figura.



molhas de aresta, outra para as molhas de projeção. A equação das molhas de aresta seguem a lei de Hooke:

$$F = \left[ k_s * (|\Delta x| - x_o) + k_d \left( \frac{\Delta v \cdot \Delta x}{|\Delta x|} \right) \right] * \frac{\Delta x}{|\Delta x|}, \quad (7.1)$$

onde  $k_s$  e  $k_d$  são os coeficientes de elasticidade e amortecimento,  $x_o$  é o comprimento de descanso da mola, e  $\Delta x$  e  $\Delta v$  são as diferenças de posição e velocidade dos vértices.

Para as molhas de projeção é aplicada uma equação não linear, como descrito em [8] e mostrado na Equação 7.2. Esta mola não linear exerce uma força infinita nos vértices quando seu comprimento tende à zero.

$$F = k_s * \left( |\Delta x| - \frac{x_o^2}{|\Delta x|} \right) * \frac{\Delta x}{|\Delta x|}. \quad (7.2)$$

## 7.2 Sistema de Partículas

Para simular o movimento oscilatório das massas-molhas, a malha é tratada como um sistema de partículas [34]. Cada vértice representa uma partícula no sistema, e partículas são associadas às extremidades das molhas. Algumas partículas temporárias podem ser introduzidas para representar os vértices virtuais. Estas partículas existem apenas ao computar as forças associadas as suas molhas de projeção, e portanto influenciam apenas localmente o sistema.

Cada partícula possui os seguintes atributos: posição, velocidade, força, e massa. O sistema itera em passos de tempo, atualizando a cada ciclo a posição e a velocidade de cada partícula. A cada passo de tempo uma iteração de todas aresta é realizada, computando as forças nas suas partículas. A força em cada vértice é a soma de todas as forças geradas por suas molhas incidentes.

O valor da massa é calculado diferentemente para cada partícula. Aquelas nas proximidades da fronteira do modelo são mais pesadas do que aquelas mais afastadas. Isto implica que os tetraedros menores perto da fronteira sejam mais robustos, e menos suscetíveis a perder a forma original. Por outro lado, os tetraedros maiores no interior do modelo são mais leves e podem ser rearrumados com maior facilidade. Desta forma, quando a resistência dos tetraedros interiores é muita grande, evita-se que aqueles na fronteira sejam muito deformados, ou que seus vértices marcados não alcancem a fronteira. A massa de um tetraedro é computada como a razão entre seu nível de refinamento e o nível máximo de refinamento da subdivisão espacial, detalhados no Capítulo 3. A massa de cada partícula é calculada como a média de todas massas dos tetraedros incidentes.

A duração de um passo de tempo é definida pelo usuário. Valores pequenos são melhores pois criam menos oscilações nas molhas. Porém, valores pequenos também resultam em mais iterações, e conseqüentemente em tempos de simulação maiores.

Cada iteração pode ser dividida em três fases distintas:

1. cálculo da força acumulada em cada vértice;
2. cálculo dos valores derivados;
3. cálculo das novas posições e velocidades.

## 7.3 Acumuladores de Forças

A força acumulada em cada vértice é a soma das contribuições de todas molas incidentes. Vértices virtuais não possuem acumuladores, mas contribuem para a força acumulada no vértice real na outra extremidade da mola.

O primeiro passo em cada iteração de tempo é inicializar as forças de todas partículas com valores nulos. Após, todas arestas e faces são iteradas para calcular os valores correspondentes nas suas partículas utilizando as Equações 7.1 e 7.2, respectivamente. Para cada mola, uma das equações é computada, e o seu valor somado ao acumulador de força de um vértice. Ao acumulador de força do vértice na outra extremidade, caso seja um vértice real, é somado o mesmo valor com sinal contrário. No caso de um vértice virtual, não é necessário acumular sua força incidente, já que sua posição é computada dinamicamente de acordo com a aresta de projeção.

## 7.4 Valores Derivados

Neste passo, as derivadas dos valores da posição e velocidade de cada partícula são calculadas, ou seja, sua velocidade e aceleração, respectivamente. Um vetor auxiliar é utilizado para armazenar estes valores. O tamanho do vetor é  $6n$ , onde  $n$  é o número de partículas no sistema. Este vetor de derivadas é escalado para o tempo de uma iteração.

## 7.5 Atualização das Coordenadas e Velocidades

Outro vetor é criado para armazenar os valores correntes das coordenadas e velocidade de cada partícula. O tamanho do vetor é o mesmo do que o vetor de derivadas. Para calcular os valores das novas coordenadas e velocidades das partículas, os dois vetores descritos acima são somados.

## 7.6 Critérios de Compressão

Os comprimentos de repouso das molas são definidos como seus comprimentos iniciais, ou seja, inicialmente o sistema está completamente estável. Portanto, se nenhuma perturbação for introduzida ao sistema, ele não mudará seu estado. Para iniciar a compressão os vértices marcados, descritos na Sessão 6.1, são empurrados em direção à fronteira.

No esquema em 2 dimensões os vértices são empurrados na direção ortogonal à fronteira. Porém, desta forma, alguns vértices chegam na fronteira muito próximos. Ainda mais, algumas vezes, vértices diferentes da malha podem chegar em um mesmo ponto da fronteira do modelo. Como consequência, são gerados triângulos com ângulos pequenos, ou até mesmo totalmente achatados.

Para corrigir estes triângulos uma etapa de pós-processamento foi criada. Após a chegada de todos os vértices marcados na fronteira, estes são inseridos para interagir no sistema físico. Porém, o único grau de liberdade atribuído a estes vértices é o movimento ao longo da fronteira. Sempre que um vértice marcado é movimentado nesta etapa, ele é novamente projetado na fronteira ao fim da iteração. Pode-se pensar como se as partículas

estivessem correndo em um trilho, enquanto as arestas entre elas produzem as forças de repulsão necessárias para espalhá-las.

Para se obter uma aproximação perfeita da fronteira do modelo, é preciso ter um vértice da malha posicionado em cada ponto do polígono. Por este motivo, nesta última etapa, quando um vértice passa por um ponto do polígono, ele é restrito de qualquer movimento até o final da simulação. No entanto, nem sempre é possível associar um vértice da malha com ponto do modelo, principalmente em resoluções baixas.

Para evitar esta etapa de pós-processamento no sistema em 3 dimensões, um novo esquema de compressão foi criado. Ao invés de projetar as partículas na direção ortogonal, a direção é definida pela média das normais das faces de fronteira incidentes. Se o vértice estiver no exterior da fronteira do modelo, as normais apontando para dentro da fronteira são utilizadas na média. Caso contrário, se o vértice estiver no interior, as normais contrárias são utilizadas.

Ainda mais, para os vértices marcados, é apenas considerada a componente da força perpendicular à velocidade. Após cada passo de tempo a velocidade é recomputada utilizando as normais das faces incidentes. Assim, os vértices marcados caminham em direção à fronteira enquanto se rearrumam no plano perpendicular à sua trajetória. Esta direção impede que vértices vizinhos cheguem muito próximos na fronteira, espalhando-os durante o trajeto. Se estes vértices interagissem livremente no sistema, as forças internas da malha eventualmente retornaria-os a seus estados iniciais.

## 7.7 Limitando a Compressão

Para melhor controlar a *malha-d* durante a compressão, algumas restrições são impostas ao movimento das partículas. Estas restrições são motivadas pelas técnicas utilizadas em simulação de tecidos [4, 1, 21]. São limitadas por esta técnica a compressão máxima das molas, e a taxa de compressão por passo de tempo. Primeiro, nenhuma mola pode sofrer uma compressão maior do que 60% do seu comprimento de repouso. Segundo, durante um passo de tempo, nenhuma partícula pode se movimentar de tal sorte que alguma de suas molas incidentes sofra uma compressão maior do que 10% da sua altura corrente.

Quando um vértice infringe uma dessas condições, ele é congelado durante o passo de tempo atual. Todos vértices que não foram parados, têm suas posições e velocidades recomputadas iterativamente como em um passo de tempo comum. Este procedimento é repetido até que nenhuma partícula precise ser congelada. Estes limites artificiais geram uma robustez maior durante a compressão da malha, e ajudam a obter uma triangulação com maior qualidade.

Os vértices marcados, que estão sendo empurrados em direção a fronteira, também podem ser congelados quando infringem uma das condições acima. Em alguns casos, podem ser continuamente parados de forma que nunca cheguem a fronteira. Isto acontece mais comumente com malhas em baixa resolução, onde os vértices devem caminhar distâncias maiores, e a variação de tamanho, entre os elementos perto da fronteira e aqueles no interior, é pequena. Este procedimento é válido somente onde uma justaposição exata da fronteira do modelo não é necessária. A simulação termina quando todos os vértices marcados chegam na fronteira ou estão congelados.

# Capítulo 8

## Resultados

O modelo do *Lago Superior* no Canadá, mostrado na Figura 8.1, foi triangulado com ângulo mínimo de  $11.5^\circ$ . A malha final contém 1906 triângulos, e 87.4% dos ângulos estão entre  $30^\circ$  e  $60^\circ$ . Na Figura 8.2, um detalhe da malha é mostrado, ilustrando que a fronteira é ligeiramente diferente da original, porém bem aproximada.

Outro exemplo é o Domo de Sal exibido na Figura 8.3. Este modelo possui seis regiões diferentes e a malha foi gerada com 6183 triângulos. O menor ângulo é de  $10.4^\circ$ , e 89.4% dos ângulos estão na faixa  $[30^\circ, 60^\circ]$ .

Na Figura 8.4 é mostrada uma triangulação de uma fatia em duas dimensões do modelo do Golfo do México. Esta fatia é composta de 15 regiões diferentes, e possui 11015 triângulos. O menor ângulo é de  $5.25^\circ$ , e 75% dos ângulos estão na faixa  $[30^\circ, 60^\circ]$ . Alguns detalhes da malha são expostos na Figura 8.5. Problemas de *serrilhamento* ocorrem nas áreas de afunilamento, e como conseqüência, a fronteira não é bem aproximada,

Na Figura 8.6 é exposta uma aproximação grosseira do modelo do Coelho de Stanford. Esta malha contém 30K tetraedros, com ângulo diedral mínimo de aproximadamente  $3.5^\circ$ , onde 99.4% dos ângulos estão acima de  $18^\circ$ .

A triangulação do Tigre, na Figura 8.7, foi gerada com 160K tetraedros e ângulo diedral mínimo de  $4.6^\circ$ . A Figura 8.8 expõe o interior do Tigre. Na Figura 8.9 são mostrados 21 níveis de resolução diferentes para o modelo.

Na Figura 8.10, três visões (duas com arestas e uma sem) de um modelo de uma cabeça são mostradas, e sua respectiva *malhas-d*. Esta malha contém 242K tetraedros, seu ângulo mínimo é de  $3.1^\circ$  e 99% dos ângulos estão acima dos  $25^\circ$ . O tempo de execução foi de aproximadamente uma hora.

Dois cortes da malha do Domo de Sal, apresentada na Sessão 6.2, são mostrados na Figura 8.11. A triangulação final contém aproximadamente 900K tetraedros, ângulo mínimo de  $4.9^\circ$  e com 99.7% dos ângulos acima de  $18^\circ$ .

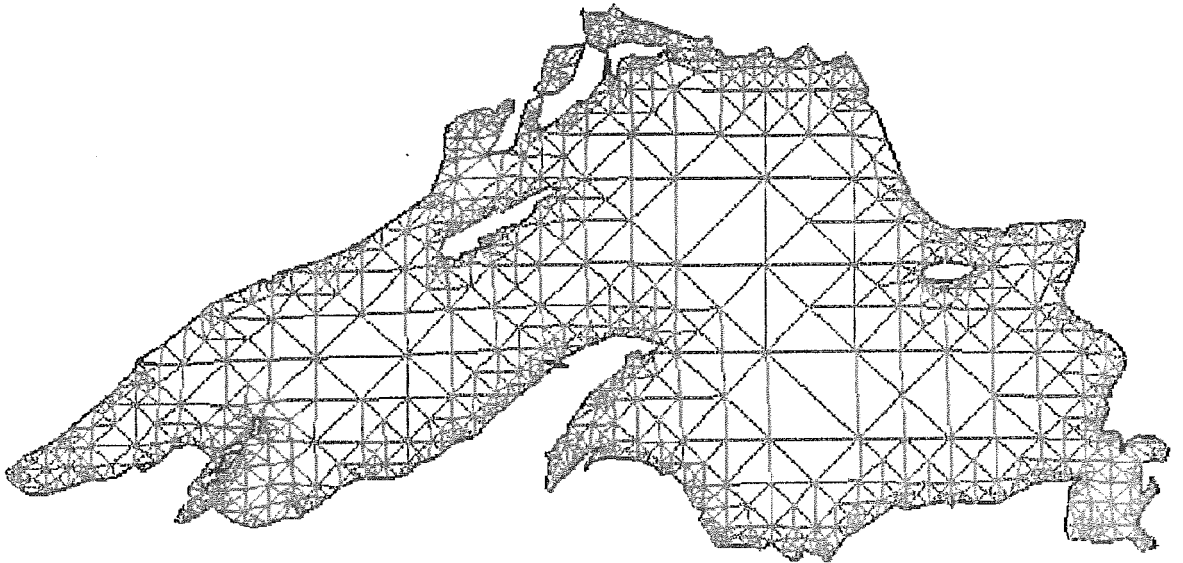


Figura 8.1: Modelo do Lago Superior triangulado.

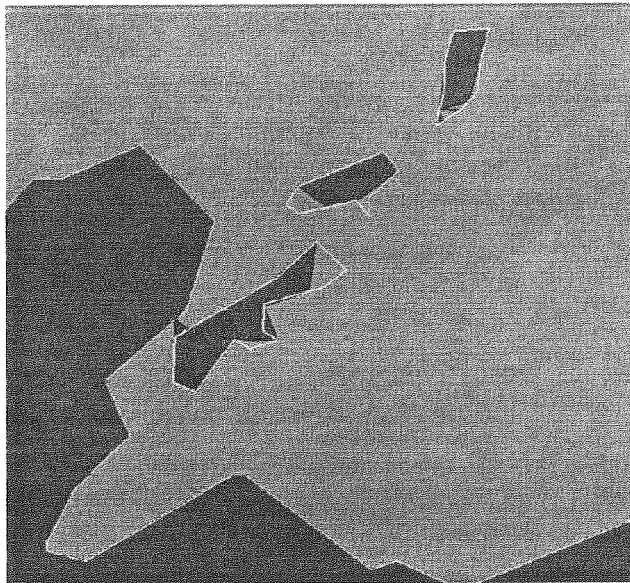


Figura 8.2: Detalhe da aproximação da malha nas ilhas do Lago Superior. As fronteiras originais são representadas por contornos de linha, e a malha final por regiões preenchidas.

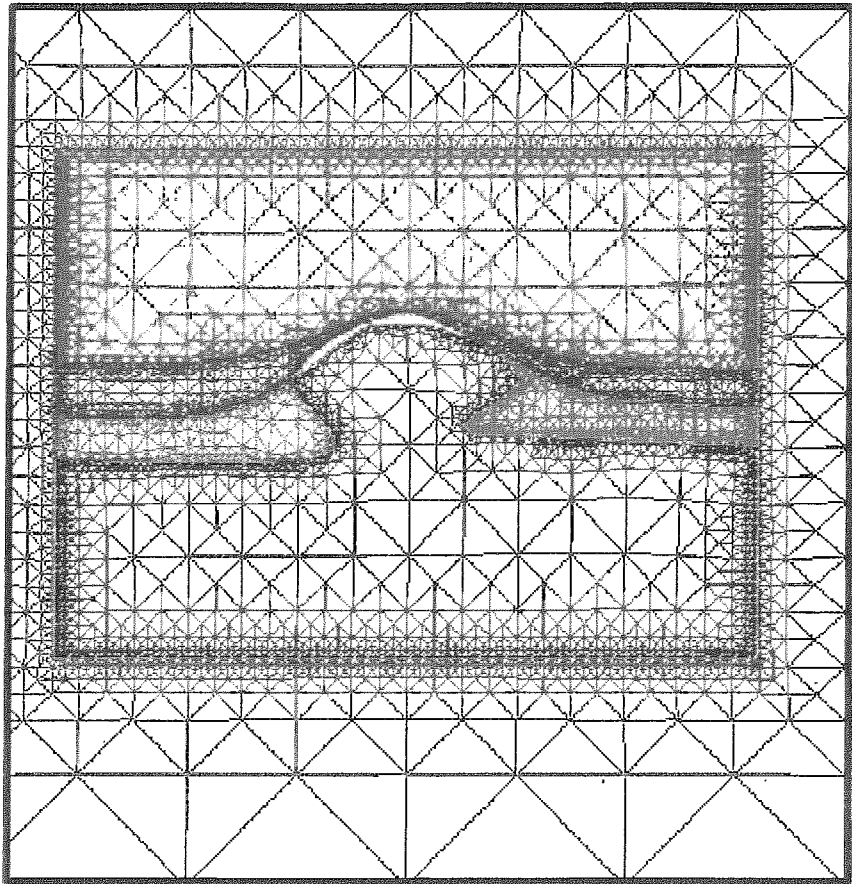


Figura 8.3: Modelo de um domo de sal.

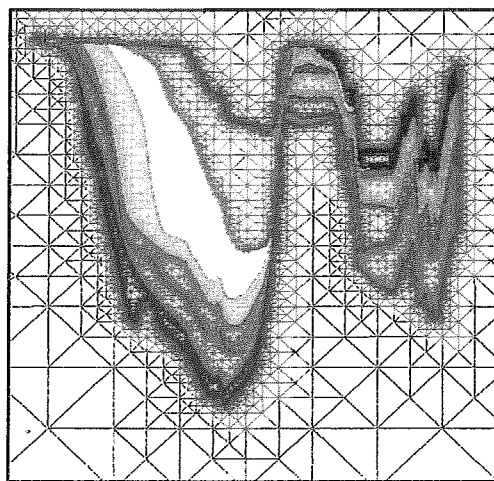


Figura 8.4: Fatia do modelo do Golfo do México.

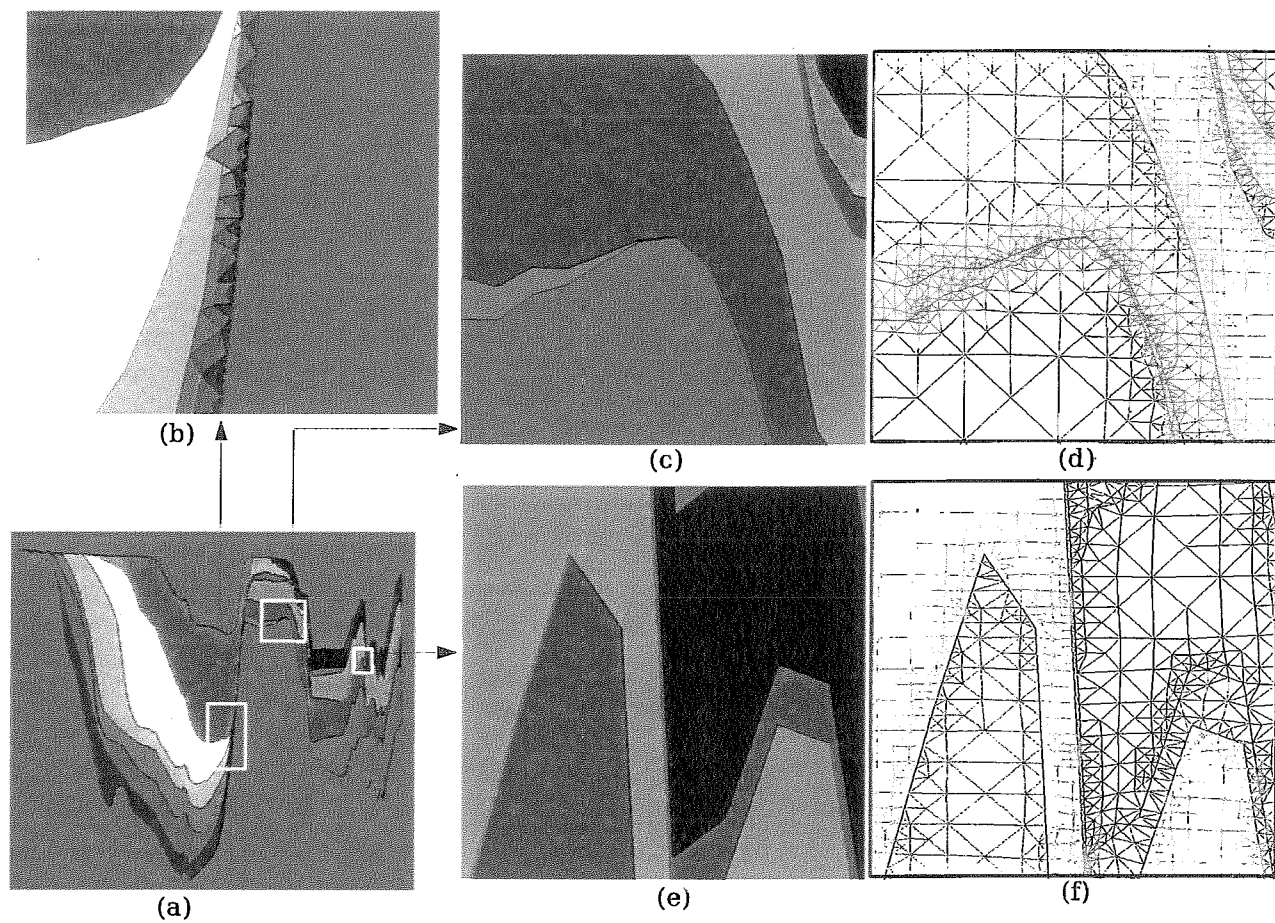


Figura 8.5: (a) malha com as regiões pintadas. (b) *serrilhamento* no local onde várias regiões se encontram. (c) e (e) detalhes das regiões onde a aproximação foi quase exata. (d) e (f) as mesmas regiões trianguladas.

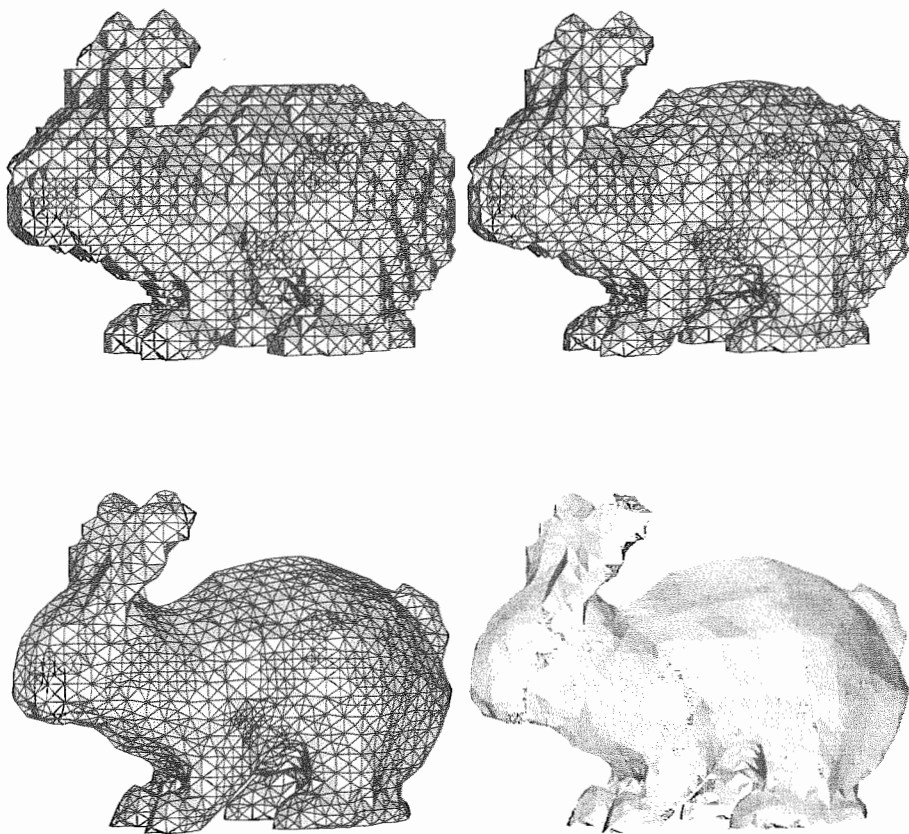


Figura 8.6: Aproximação grosseira do Coelho de Stanford com 30K tetraedros. Na seqüência a *malha-d* antes, durante e ao final (com e sem arestas) da compressão.



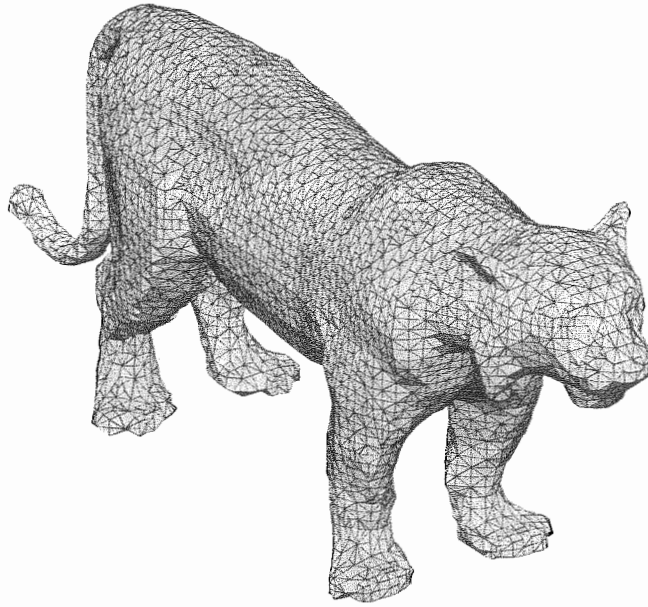


Figura 8.7: Triangulação do Tigre com aproximadamente 160K tetraedros.

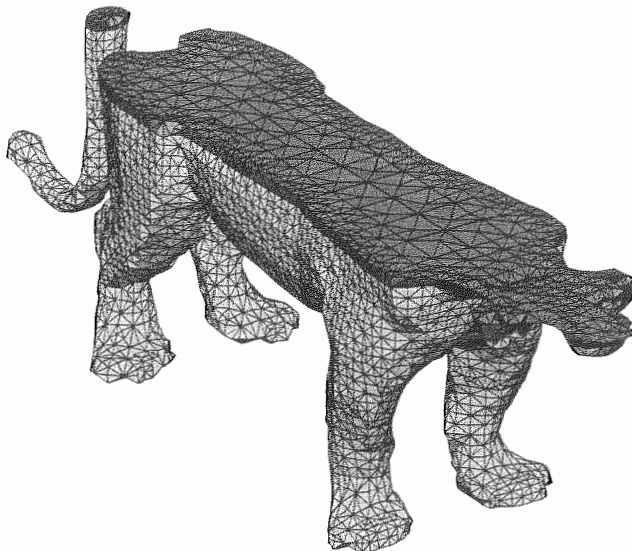


Figura 8.8: Um corte do modelo do Tigre.

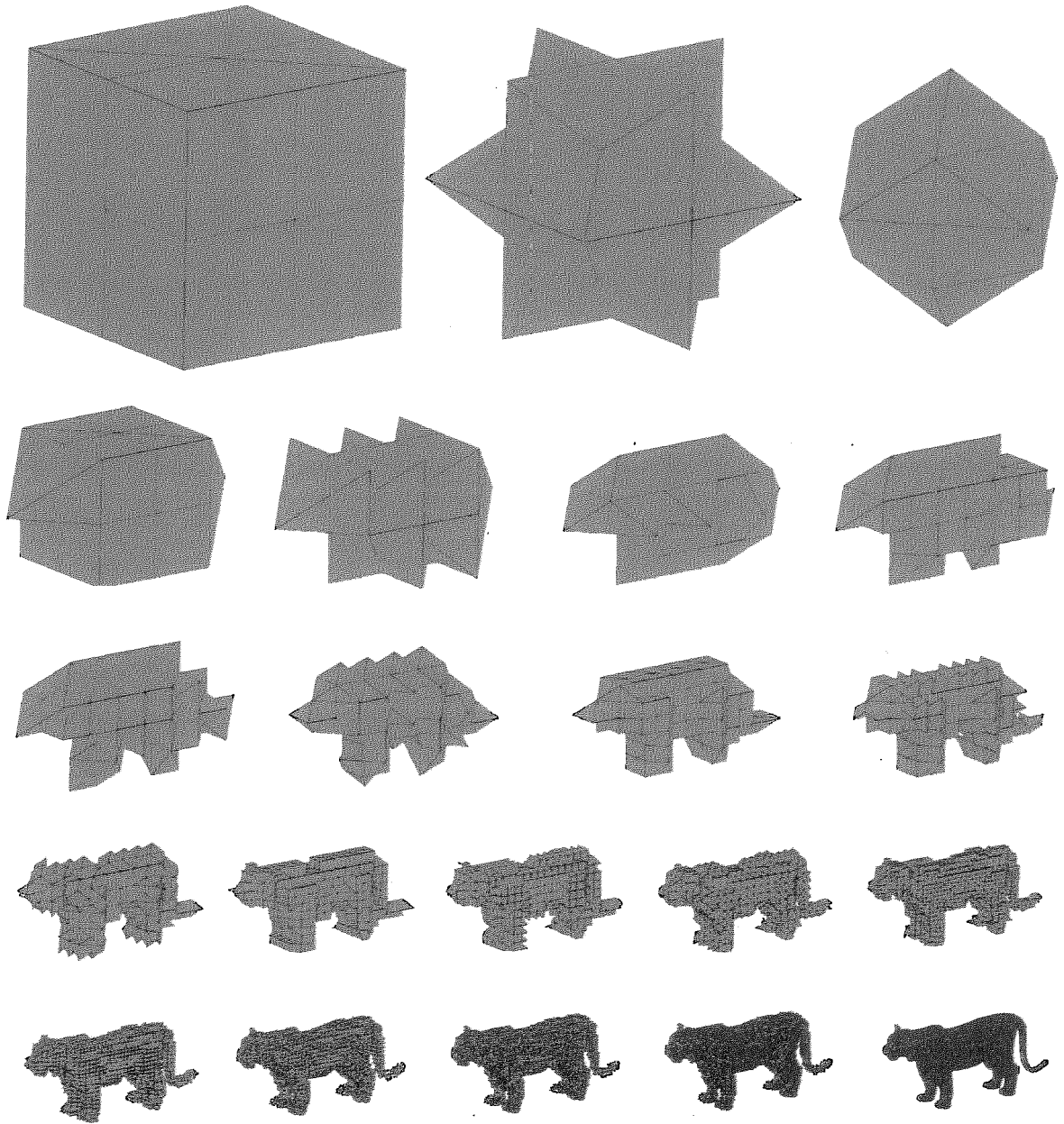


Figura 8.9: Modelo do tigre exposto em diversas resoluções consecutivas.

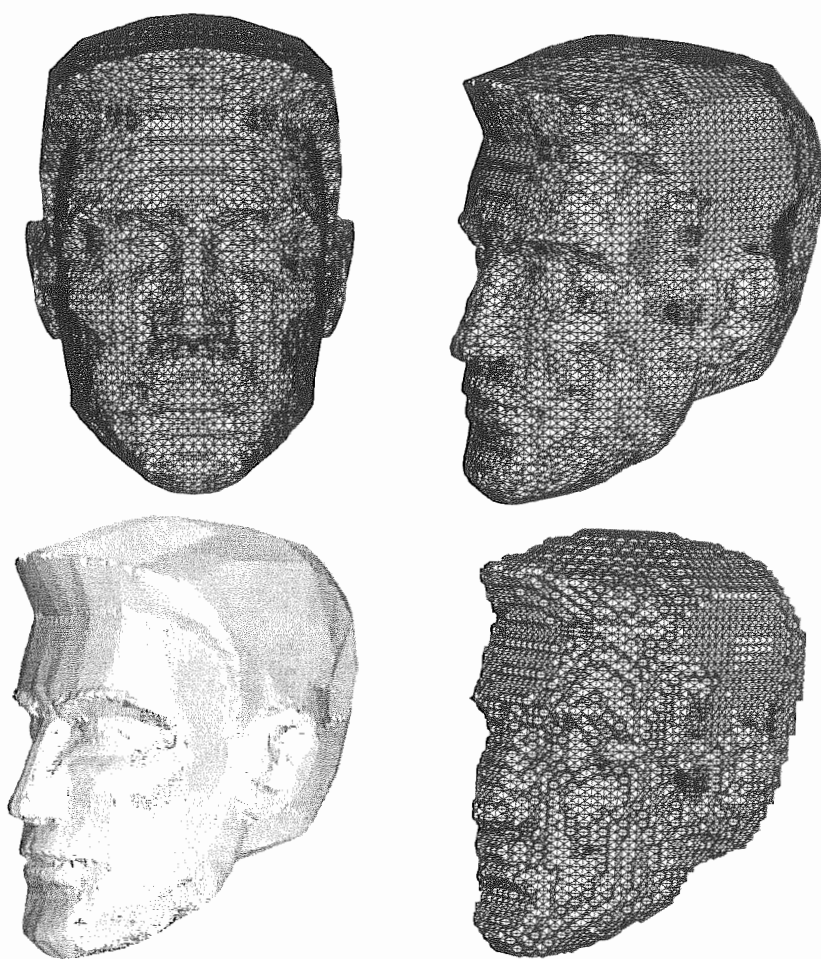


Figura 8.10: Três visões da malha final e uma da *malha-d*.

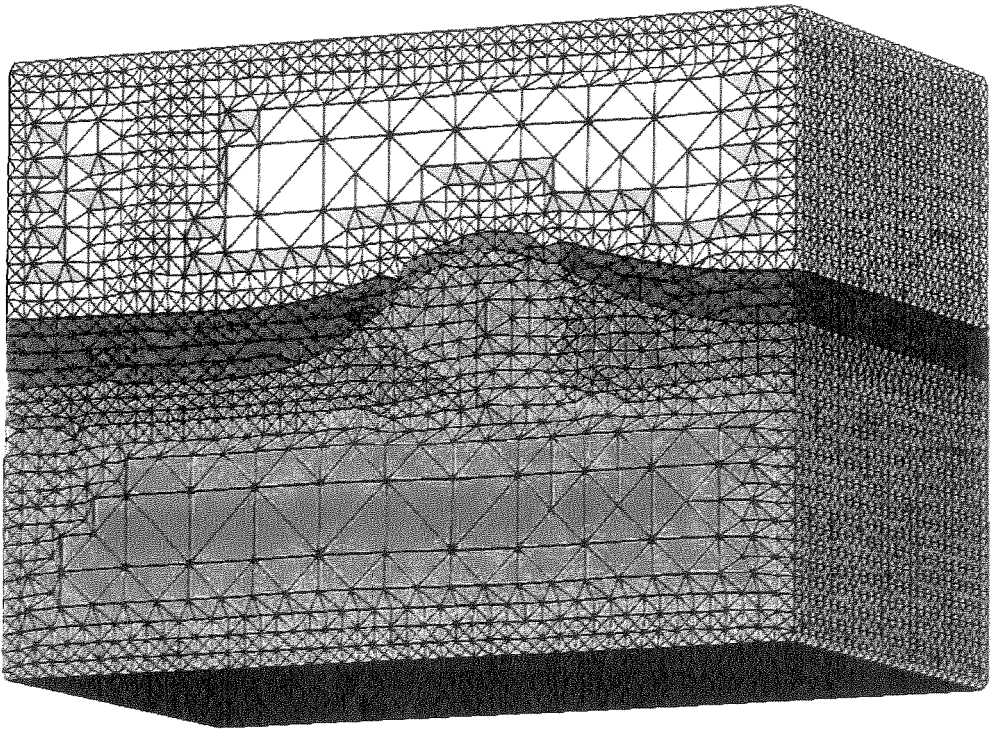
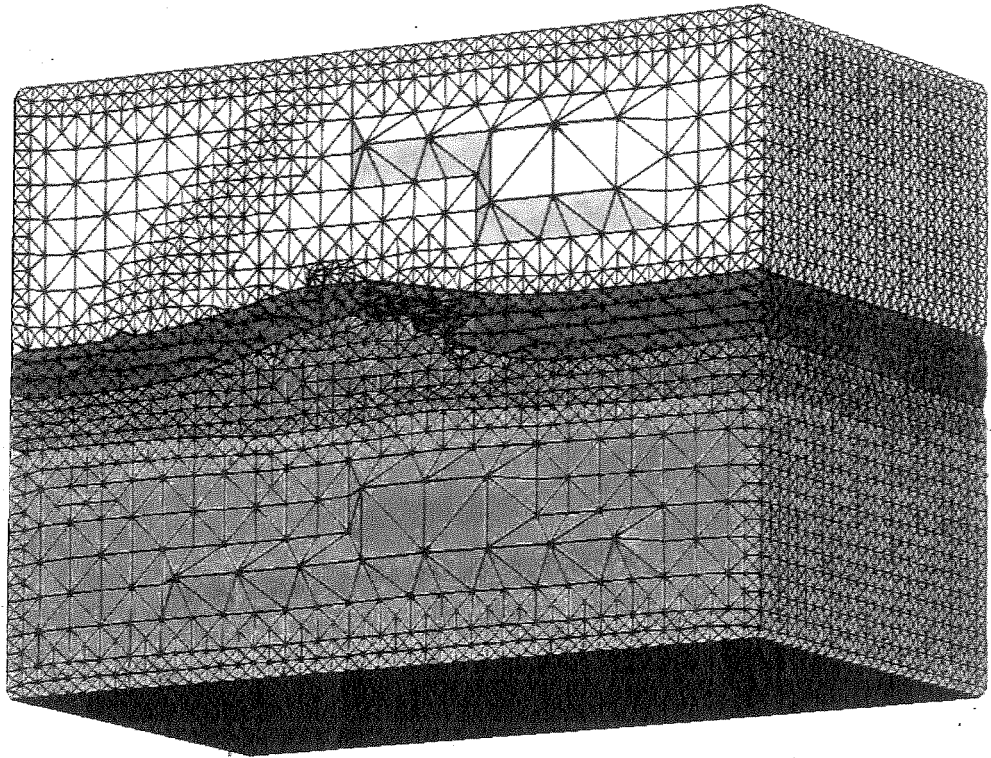


Figura 8.11: Corte da malha do Domo de Sal.

# Capítulo 9

## Conclusões e Trabalhos Futuros

### 9.1 Conclusões

Apresentamos um algoritmo para triangulação de modelos em duas ou três dimensões com bordas complexas e irregulares. Por mais, o algoritmo é capaz de gerar a malha em diversas resoluções e não produz elementos de área ou volume zero. Estas são as principais vantagens oferecidas como extensões dos algoritmos de triangulação tradicionais.

Todavia, uma das maiores diferenças reside no fato de não haver uma fidelidade quanto à fronteira da malha gerada em relação à fronteira original do modelo. Os algoritmos tradicionais, como aqueles apresentados no Capítulo 2, são sempre fiéis à fronteira original. Apesar desta propriedade ser importante em vários casos, existem aplicações onde uma aproximação inexata é aceitável, como por exemplo os modelos geológicos ressaltados nesta dissertação. Estes modelos provêm de interpretações humanas de dados sísmicos e, por esta razão, estão sujeitos à erros. Assim, pequenos desvios nos detalhes da malha não provocam problemas na sua utilização.

Ainda mais, modelos geológicos são utilizados em simulações numéricas, onde a qualidade dos elementos da malha é de maior importância do que a fidelidade à fronteira. Elementos achatados e ângulos muito pequenos podem impedir que a simulação chegue ao seu término.

Contudo, alguns algoritmos de triangulação tradicionais obtiveram resultados expressivos com provas teóricas de ângulo mínimo, e garantia de termino, especialmente em duas dimensões. No entanto, apesar dos maiores esforços relacionados a estes algoritmos serem de, justamente, minimizar os danos causados nas vizinhanças destes ângulos pequenos herdados, eles sempre estarão presentes na malha final.

O algoritmo proposto também oferece um esquema de multi-resolução sem esforço adicional. A malha após gerada pode ser visualizada em qualquer resolução inferior à máxima. Isto se deve ao fato da estrutura utilizada para gerar a grade já possuir esta propriedade de multi-resolução, e da topologia da malha não ser alterada durante todo processo.

### 9.2 Trabalhos Futuros

Primeiramente, um esquema de subdivisão espacial baseado na curvatura da fronteira, e não na distância à mesma, é de grande interesse. A malha obtém desta forma, tetraedros

menores em locais onde a fronteira é mais detalhada e complexa. Nas regiões onde a fronteira é mais homogênea é possível economizar muitos elementos.

Outra proposta é a criação de um métrica para avaliar a aproximação da malha quanto à fronteira original. Com isto é possível saber quais malhas apresentaram uma aproximação melhor e onde ocorrem os maiores desvios.

O problema do *serrilhamento* é outro ponto a ser melhorado. Eliminar o problema por completo não é viável nesta implementação. Isto se deve ao fato de que a única maneira de eliminá-lo, é aceitando a herança dos ângulos pequenos originais, onde o problema geralmente ocorre. Porém, como ressaltado anteriormente, o algoritmo têm como prioridade a geração de elementos de qualidade sobre a exatidão da fronteira. Refinar mais a malha onde ocorre o *serrilhamento*, pode melhorar a aproximação, mas ainda não elimina o problema por completo.

Uma investigação maior para encontrar valores ideais para as constantes elásticas e de amortecimento das molas deve trazer grandes melhorias aos resultados gerados. Os testes mostram que variações nas constantes melhoram os ângulos mínimos gerados em alguns modelos consideravelmente. Para um mesmo modelo de uma esfera, por exemplo, foi possível melhorar o ângulo mínimo de  $19^\circ$  para  $41^\circ$  apenas aumentando os valores das constantes, tornando as molas mais flexíveis. Porém, está relação ainda não está clara até o presente momento. Para outros casos, modificando os valores da mesma maneira, não foram notadas melhorias.

# Referências Bibliográficas

- [1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM Press, 1998.
- [2] Marshall Bern, Scott Mitchell, and Jim Ruppert. Linear-size nonobtuse triangulation of polygons. In *Proc. 10th Symp. Computational Geometry, ACM*, pages 221–230, June 1994.
- [3] Marshall W. Bern, David Eppstein, and John R. Gilbert. Provably good mesh generation. In *IEEE Symposium on Foundations of Computer Science*, pages 231–241, 1990.
- [4] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 594–603. ACM Press, 2002.
- [5] Paulo Roma Cavalcanti, Paulo C.P. Carvalho, and Luiz F. Martha. Nonmanifold modeling: An approach based on spatial subdivisions. *Computer-Aided Design*, 29(3):209–220, March 1997.
- [6] Paul Chew. Guaranteed-quality triangular meshes. Technical report, Department of Computer Science, Cornell University, April 1989.
- [7] Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Ninth Annual Symposium on Computational Geometry*, pages 274–280, San Diego, California, May 1993.
- [8] Lee Cooper and Steve Maddock. Preventing collapse within mass-spring-damper models of deformable objects. In *The Fifth International Conference in Central Europe on Computer Graphics and Visualization*, pages 70–78, Plzen, Czech Republic, February 1997.
- [9] Luiz Henrique de Figueiredo, Jorge Stolfi, and Luiz Velho. Approximating parametric curves with strip trees using affine arithmetic. *Computer Graphics Forum*, 22(2):171–179, 2003.
- [10] Peter Fleischmann and Siegfried Selberherr. Three-dimensional delaunay mesh generation using a modified advancing front approach. In *6th International Meshing Roundtable*, pages 267–278, Sandia National Laboratories, October 1997.

- [11] Pascal J. Frey, Houman Borouchaki, and Paul-Louis George. Delaunay tetrahedralization using an advancing-front approach. In *5th International Meshing Roundtable*, pages 21–46, Sandia National Laboratories, October 1996.
- [12] André Guézic. Meshsweeper: Dynamic point-to-polygonal-mesh distance and applications. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):47–61, 2001.
- [13] Xiang-Yang Li, Shang-Hua Teng, and Alper Üngör. Biting spheres in 3d. In *Proc. 8th Int. Meshing Roundtable*, pages 85–95, South Lake Tahoe, CA, October 1999.
- [14] Xiang-Yang Li, Shang-Hua Teng, and Alper Üngör. Biting: Advancing front meets sphere packing. *International Journal for Numerical Methods in Engineering*, 49(1):61–91, September 2000.
- [15] Rainald Löhner. Progress in grid generation via the advancing front technique. *Engineering with Computers*, Springer-Verlag, 12:186–210, December 1996.
- [16] Rainald Löhner and Paresh Parikh. Three-dimensional grid generation by the advancing front method. *International Journal of Numerical Methods in Fluids*, 8:1135–1149, 1988.
- [17] Dimitri J. Mavriplis. An advancing front delaunay triangulation algorithm designed for robustness. *Journal of Computational Physics*, 117(1):90–101, 1995.
- [18] Vinicius Mello, Luiz Velho, Paulo Roma Cavalcanti, and Claudio Silva. Bmt: A generic programming approach to multiresolution spatial decompositions. In H. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 337–360. Springer-Verlag, Berlin, 2003.
- [19] Neil Molino, Robert Bridson, Joseph Teran, and Ronald Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable*, pages 103–114, Santa Fe, New Mexico, September 2003.
- [20] Jaime Peraire, Joaquim Peiró, and Ken Morgan. Advancing front grid generation. In Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherill, editors, *Handbook of Grid Generation*, chapter 17. CRC Press, 1999.
- [21] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In Wayne A. Davis and Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Human-Computer Communications Society, 1995.
- [22] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.
- [23] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.



- [24] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997.
- [25] Jonathan Richard Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95. ACM Press, 1998.
- [26] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry : Theory and Application*, 22(1-3):21–74, May 2002.
- [27] Kenji Shimada and David C. Gossard. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In *SMA '95: Proceedings of the Third Symposium on Solid Modeling and Applications*, pages 409–419, 1995.
- [28] Luiz Velho. Quasi 4-8 subdivision. *Computer-Aided Geometric Design*, 18(4):345–357, May 2001.
- [29] Luiz Velho. Using semi-regular 4-8 meshes for subdivision surfaces. *Journal of Graphics Tools*, 5(3):35–47, 2001.
- [30] Luiz Velho. Dynamic adaptive meshes. In *Dagstuhl Seminar on Hierarchical Methods in Computer Graphics.*, 2003.
- [31] Luiz Velho. A dynamic adaptive mesh library based on stellar operators. *Journal of Graphics Tools*, 9(2):1–29, 2004.
- [32] Luiz Velho and Denis Zorin. 4-8 subdivision. *Computer-Aided Geometric Design*, 18(5):397–427, 2001. Special Issue on Subdivision Techniques.
- [33] Kevin Weiler. The radial-edge structure: A topological representation for non-manifold geometric boundary representations. In *Geometric Modeling for CAD Applications*, pages 3–36. North-Holland, 1988.
- [34] Andrew Witkin. An introduction to physically based modeling: Particle system dynamics. ACM Siggraph Course Notes, 1994.