Programa de Pós-Graduação em Engenharia Elétrica

Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica

Escola de Engenharia da Universidade Federal de Minas Gerais

Reconstrução de superfície a partir de um conjunto não-organizado de pontos

Autor: Joseane Alves Freire

Orientador: Prof. Renato Cardoso Mesquita

Belo Horizonte, 21 de Outubro de 2005

Dedico aos mes pais,aos meus sobrinhos
Felipe, Tiago e Ana Clara, aos amigos que sempre me
apoiaram e em especial ao meu namorado
Guilherme por não me deixar desistir.

Agradecimentos

Agradeço ao professor Renato Mesquita pela orientação no desenvolvimento deste trabalho, e principalmente por não deixar que eu desistisse. A ajuda do colega João Paulo Gois pelas constantes "dicas"de implementação.
Às colegas de república pela amizade e o constante apoio. Aos antigos colegas de trabalho da Devex, que sempre torceram por mim. E aos atuais colegas da Unitech, por também me incentivarem na conclusão do mestrado.

Resumo

Reconstrução da superfícies é a área que se dedica a obter modelos geométricos complexos a partir de um conjunto finito de pontos não-organizados da superfície de objetos. Esta área vem se tornando cada vez mais importante dentro de Modelagem Geométrica e em outras aplicações como Computação Gráfica, Visão Computacional, Realidade Virtual e Engenharia.

Esta dissertação apresenta uma revisão bibliográfica que aborda os principais métodos de reconstrução de superfícies. Esses métodos foram divididos em quatro categorias: métodos baseados em esculpimento, métodos baseados em funções implícitas, métodos incrementais e métodos baseados em modelos deformáveis.

Uma ênfase maior é dada aos algoritmos baseados em esculpimento, especialmente aos algoritmos da família *Crust*, uma vez que esses possuem garantias teóricas de reconstrução, em função da densidade das amostras.

Em 2D, é demonstrado que as garantias de reconstrução para o algoritmo β-*Skeleton* são melhores que as do algoritmo *raw crust*, no entanto, o primeiro não pode ser estendido para $\Re^3$. Também são apresentadas as versões 3D dos algoritmos *raw crust* e *power crust*, bem como os principais teoremas que definem suas garantias de reconstrução.

Apresentam-se os resultados práticos obtidos na implementação do *raw crust* e *power crust* 3D. Para implementação desses algoritmos foi utilizada a biblioteca *CGAL* (*Computational Geometry Algorithms Library*). Esta biblioteca disponibiliza os principais métodos envolvidos nesses algoritmos, como, por exemplo, as triangulações de Delaunay tradicional e com peso. Além disso, oferece suporte a classes com aritmética de precisão arbitrária, necessária para os cálculos geométricos envolvidos nas triangulações.

Através dessas implementações é mostrado que o algoritmo *raw crust*, tanto na versão bidimensional quanto tridimensional, é bem mais simples que o algoritmo *power crust*, não apresentando maiores dificuldades em sua implementação. O *power crust* é um algoritmo bem mais complexo, que envolve cálculos adicionais como: triangulação de Delaunay com peso, interseção entre bolas polares e cálculo do ortocentro dos tetraedros da triangulação. Por outro lado, apresenta melhores garantias de reconstrução que o *raw crust*. No entanto, os resultados obtidos não foram satisfatórios devido às dificuldades numéricas encontradas em sua implementação.

Assim, este trabalho constitui uma base para o desenvolvimento de uma biblioteca capaz de reconstruir objetos a partir de amostras de pontos, que possa ser integrada ao modelador de sólidos GSM (*GOPAC Solid Modeler*) desenvolvido pelo grupo de pesquisa GOPAC (Grupo de Otimização e Projeto Assistido por Computador) da UFMG. Com esta futura integração, poder-se-á gerar modelos de objetos diretamente no GSM a partir dos dados obtidos em um *scanner* tridimensional.

Abstract

Surface reconstruction is the area dedicated to get complex geometric models from a finite set of unorganized points of object surfaces. This area has become increasingly important in Geometric Modeling and other applications such as Computer Graphics, Computer Vision, Virtual Reality and Engineering.

This Master thesis describes a broad survey of the main reconstruction methods. These methods are divided in four categories: sculpturing methods, implicit function methods, incremental methods and warping methods.

Emphasis is given to sculpturing methods, especially to the Crust family algorithms, because they have theoretical guarantees of reconstruction based on the sample density.

In 2D, it is demonstrated that the reconstruction guarantees for the β-Skeleton algorithm are better than for the raw crust algorithm. However, the first one cannot be extended to $\Re^3$. The three-dimensional versions of the crust and power crust algorithms are presented, together whith the main theorems that define their reconstruction guarantees.

The obtained results for the 3D raw crust and power crust algorithms are presented. The CGAL (Computational Geometry Algorithms Library) library was used to implement these algorithms. This library provides the main methods in these algorithms, as the traditional and weighted Delaunay triangulations. Moreover, it is able to represent a floating point value whith arbitrary precision, which is necessary for geometric calculations in the triangulations.

Through these implementations it is shown that the crust algorithm (2D and 3D versions) is much simpler than power crust, not presenting difficulties in its implementation. Power crust is an algorithm more complex, that involves supplementary calculations such as: weighted Delaunay triangulation, intersecting polar balls and ortocenter calculations. It presents better reconstruction guarantees of that raw crust, however, its output was not satisfactory because numerical difficulties in its implementation have been found.

Thus, this work constitutes a base for developing a library to reconstruct objects from sample points, which can be integrated to the GSM (GOPAC Solid Modeler). With this future integration, the GSM will be able to directly generate object models from three- dimensional points obtained from a 3D scanner.

Sumário

1 Introdução **1**
- 1.1 Motivação e objetivos . . . 2
- 1.2 Organização da dissertação . . . 3

2 Elementos básicos da geometria computacional **4**
- 2.1 Considerações Iniciais . . . 4
- 2.2 Definições . . . 4
- 2.3 Considerações finais . . . 13

3 Métodos para Reconstrução de Superfícies **14**
- 3.1 Considerações iniciais . . . 14
- 3.2 Métodos baseados em esculpimento . . . 15
- 3.3 Métodos baseados em funções implícitas . . . 17
- 3.4 Métodos incrementais . . . 18
- 3.5 Métodos baseados em modelos deformáveis . . . 19
- 3.6 Considerações finais . . . 20

4 Família *Crust* **21**
- 4.1 Considerações Iniciais . . . 21
- 4.2 β*-skeleton* . . . 21
- 4.3 Condições de amostragem para o β*-Skeleton* . . . 22
- 4.4 Crust 2D . . . 27
- 4.5 Crust 3D . . . 30
- 4.6 Power Crust 3D . . . 35
- 4.7 Cocone . . . 40
- 4.8 Tight Cocone . . . 41
- 4.9 Considerações Finais . . . 42

5 Implementações e Resultados **43**
- 5.1 Considerações Iniciais . . . 43
- 5.2 Ferramentas utilizadas na implementação . . . 43
- 5.3 Raw Crust . . . 45
- 5.4 Power Crust . . . 51
  - 5.4.1 Algoritmo prático para rotulação dos pólos . . . 51
  - 5.4.2 Cálculo do ângulo entre duas esferas . . . 54
  - 5.4.3 Resultados do *Power Crust* . . . 56
- 5.5 Considerações Finais . . . 59

6 Conclusões e trabalhos futuros **60**

Lista de Figuras

1.1 Exemplo do processo de reconstrução de superfícies. 2

2.1 Fecho convexo de um conjunto de pontos no plano. 4
2.2 Da esquerda para direita são mostrados respectivamente um *0-simplexo*, *1-simplexo*, *2-simplexo* e *3-simplexo*. 5
2.3 a) Exemplo de um complexo simplicial, b) Não define um complexo simplicial. Note que existem intersecções que não definem uma face. Fonte: [Gois, 2004] 6
2.4 Exemplo do diagrama de Voronoi no plano de um conjunto de pontos S. Cada célula de Voronoi é um polígono formado por um conjunto de segmentos ou semi-retas. As esferas de Voronoi são círculos, o centro de um círculo que passa por três pontos de S é um vértice v de Voronoi. 7
2.5 O eixo medial da curva é dado pelas linha pontilhadas. Fonte: [Amenta and Bern, 1999a] . . 7
2.6 Exemplo do eixo medial de uma superfície $3D$. O eixo medial é mostrado na figura pela superfície $2D$ em destaque dentro da superfície transparente. Fonte: [Amenta and Bern, 1999a]. 8
2.7 Exemplo da triangulação de Delaunay no plano com 100 pontos. Fonte: [Gois, 2004] 9
2.8 Exemplo da relação de dualidade entre a triangulação de Delaunay (pontos $P_1..P_7$) e o diagrama de Voronoi (vértices $V_1..V_6$) no plano. 9
2.9 Representação dos pontos com peso $\hat{p_1}$ e $\hat{p_2}$ em círculos. 10
2.10 Pontos com peso $\hat{p_1}$ e $\hat{p_2}$ ortogonais. 10
2.11 As linhas contínuas formam a triangulação com peso de quatro pontos com peso no plano. As linhas tracejadas representam seu respectivo diagrama de Voronoi. Cada vértice de Voronoi é o centro de um círculo ortogonal aos círculos das regiões que delimitam seu vértice. Fonte: [Edelsbrunner, 2001] 12
2.12 Exemplo da triangulação de Delaunay com peso. Os pontos com peso são representados pelos círculos contínuos. O ponto p é um exemplo de um vértice que foi removido da triangulação. O círculo tracejado é a representação de um ortocírculo. 12
2.13 Exemplo de região proibida. (a) $\beta = 1$, (b) $\beta > 1$. 13
2.14 A aresta definida pelos vértices p_1, p_2 pertence ao β-*Skeleton* considerando $\beta > 1$. 13

3.1 vizinhança natural em $2D$: os vizinhos naturais do ponto x são os pontos $p_1, p_2, p_3, p_4, p_5 e p_6$. 18

4.1 Exemplo de uma reconstrução planar a partir de um conjunto de pontos. 22
4.2 Relação entre a distância $d(p, s)$ e $d(p, p_m)$. p é um ponto da curva F, s é um ponto da amostra e p_m é o ponto do eixo medial E mais próximo de p 23
4.3 Interpretação geométrica do teorema 1. Em linhas contínuas a curva F e em linha tracejadas o eixo medial. 23

4.4 Linha L tangente ao círculo em p. $d(c,p) = d(c,s) = 1$ e $d(s,p) = r$. Fonte: [Amenta et al., 1998] . . . 25
4.5 Definição da região proibida entre dois círculos em termos de ângulos. Fonte: [Gois, 2004] . 25
4.6 Demonstração do teorema 8. Fonte: [Gois, 2004] . . . 26
4.7 A garantia da reconstrução da curva original se dá para valores na região hachurada. Cada uma das curvas é referente a uma equação: 4.1, 4.2 e 4.3. . . . 27
4.8 À esquerda é mostrado o diagrama de Voronoi para uma amostra de pontos de uma curva. Neste caso o diagrama se aproxima do eixo medial da curva. A direita é mostrada a Triangulação de Delaunay com os pontos da amostra (pontos pretos) mais os vértices do Diagrama de Voronoi (pontos cinzas), resultado do algoritmo *Crust-2D*. . . . 28
4.9 Construindo a contradição do teorema 12. . . . 28
4.10 Representação geométrica da contradição do teorema 13. . . . 29
4.11 Nota-se que o ângulo θ é maior que seu correspondente ângulo do outro lado. . . . 30
4.12 Exemplo de um tetraedro *sliver*. . . . 31
4.13 (a)Exemplo de uma célula Voronoi - fina e comprida. Os vértice V_1 - V_6 da célula estão próximos do eixo medial. (b) Polos P^+ e P^- calculados a partir de P . . . 32
4.14 Esfera $(q, LSF(q))$ contém a esfera $(p, LFS(q) - d(p,q))$. . . . 33
4.15 Eixo definido pelo ponto da amostra s. . . . 34
4.16 Caracterização do lema3. . . . 34
4.17 Demonstração do teorema 14. Ponto t de um triângulo de *crust* deve estar mais próximo do ponto q da superfície. . . . 35
4.18 Bolas polares B_{p_1,ρ_1} e B_{p_2,ρ_2} originadas pelo ponto da amostra s. O ângulo α é dado pela interseção entre as duas bolas. . . . 36
4.19 Interseção das bolas polares internas e externas gerando a aproximação da superfície M. . . 37
4.20 Exemplo do *power crust* bidimensional. (a) Curva original, seu eixo medial e uma circunferência que toca a curva em dois pontos e possui centro num ponto do eixo medial. (b) Diagrama de Voronoi. (c) Circunferências que representam as bolas polares. (d) *Power diagrama* do conjunto de bolas polares. (e) Respectivamente o *power crust* e a aproximação do eixo medial denominado *power shape*. Fonte: [Amenta et al., 2001a] . . . 38
4.21 Exemplos da aproximação de objetos através da união de bolas polares internas. Fonte: [Gois, 2004] . . . 38
4.22 *Cocone* (segmentos destacados em negrito) de ápice em p. n é o vetor normal, p^+ e p^- são os pólos de p. Fonte [Gois, 2004] . . . 40
4.23 a) e b) exemplos de superfícies water tight, c) e d) superfícies não water tight. Fonte [Gois, 2004] . . . 41

5.1 Exemplo de um ambiente virtual utilizando VRML. . . . 44
5.2 Exemplo de reconstrução para algoritmo *crust 2D*. (a) Pontos da amostra. (b) Triangulação dos pontos da amostra. (c) Resultado da reconstrução. . . . 46
5.3 Exemplo de reconstrução para algoritmo *crust 2D*. (a) Pontos da amostra. (b) Triangulação dos pontos da amostra. (c) Resultado da reconstrução. . . . 46
5.4 Exemplo de reconstrução para algoritmo *raw crust 2D*. (a) Pontos da amostra. (b) Triangulação dos pontos da amostra. (c) Resultado da reconstrução. . . . 46
5.5 Resultado da reconstrução da superfície para uma amostra com 766 pontos. Em (a) são exibidos os pontos da amostra e em (b) o resultado da reconstrução para o algoritmo *raw crust*. 47
5.6 Resultado da reconstrução da superfície para uma amostra com 1.000 pontos. Em (a) são exibidos os pontos da amostra e em (b) o resultado da reconstrução para o algoritmo *raw crust*. 47

5.7 Resultado da reconstrução da superfície para uma amostra com 1.500 pontos. Em (a) são exibidos os pontos da amostra e em (b) o resultado da reconstrução para o algoritmo *raw crust*. 47
5.8 *Zoom* do resultado da reconstrução da superfície para o exemplo da figura 5.6. Os círculos contínuos exemplificam a relação das falhas da amostra e a presença de buracos em regiões características do objeto. Os círculos pontilhados exemplificam regiões da amostra menos características, onde a densidade dos pontos não provoca problemas na reconstrução. 48
5.9 *Zoom* do resultado da reconstrução da superfície para o exemplo da figura 5.7. Os círculos contínuos exemplificam a relação das falhas da amostra e a presença de buracos em regiões características do objeto. Os círculos pontilhados exemplificam regiões da amostra menos características, onde a densidade dos pontos não provoca problemas na reconstrução. 48
5.10 Amostra de pontos de dois objetos. Em (a) a amostra representada possui 2164 pontos e em (b) a amostra é constituída de 2655 pontos. 49
5.11 Resultado da reconstrução da amostra de pontos apresentada na figura 5.10 (a). É possível identificar a presença de buracos na região em destaque. 49
5.12 Resultado da reconstrução da amostra de pontos apresentada na figura 5.10 (b). Podemos identificar a presença de buracos na região em destaque. 50
5.13 Resultado da reconstrução da superfície para uma amostra com 3.065 pontos. Inicialmente são exibidos os pontos da amostra. Em seguida o resultado da reconstrução para o algoritmo *raw crust*. As regiões destacadas na figura são aquelas em que o algoritmo apresentou problemas na reconstrução. 50
5.14 Em linhas contínuas é mostrado o diagrama de Voronoi. O segmento em negrito representa uma aresta de Delaunay. O ângulo formado pelo ponto da amostra s e seus dois respectivos pólos é dado por β. Ao nomear o pólo p, alteram-se a prioridade do pólo q, pelo valor do ângulo β. 52
5.15 Corte planar da interseção entre duas bolas polares. 55
5.16 Ausência de interseção entre duas bolas polares. 55
5.17 Interseção entre duas bolas polares, onde o centro de uma esfera está no interior de outra. . . 56
5.18 Bola polar totalmente interna a outra bola polar. 56
5.19 Pontos das amostras de duas superfícies a serem reconstruídas. 57
5.20 Em preto, os pólos classificados como internos à superfície, e em cinza os pontos da amostra. 57
5.21 Em preto, pólos classificados como externos à superfície. 58
5.22 Aproximação do objeto através da união das bolas polares internas. Para a primeira amostra, tem-se uma boa aproximação do objeto. No segundo exemplo, bolas incorretamente classificadas fazem com que a união das bolas polares internas não seja uma boa aproximação para a superfície. 58
5.23 Resultados da reconstrução de duas superfícies utilizando o algoritmo *power crust*. 59

Lista de Símbolos

B	Circunferência utilizada nas provas dos teoremas de garantias de reconstrução
B'	Circunferência utilizada nas provas dos teoremas de garantias de reconstrução
B_v	Círculo de Voronoi com centro em v
B'_v	Círculo de Voronoi com centro em v'
c	Ponto que representa o centro de uma circunferência
C	Célula
C_c	Complexo celular
$CONV$	Fecho convexo
C_σ	Complexo simplicial
d	Distância Euclidiana
d_{pow}	*Power* distância
Del	Triangulação de Delaunay
E	Eixo medial de uma curva
EDT	Conjunto de arestas da triangulação de Delaunay
EGH	Hipergrafo de Gabriel Extendido
$EMST$	Árvore geradora mínima Euclidiana
f_d	Função de distância com sinal
f_M	Função de Morse discreta
F	Curva suave em duas dimensões**
GG	Grafo de Gabriel
j	Dimensão de um simplexo
k	Constante utilizada na definição de um k-simplexo, onde $k+1$ é o número de elementos do simplexo.
l	Constante utilizada na definição de um l-simplexo, onde $l+1$ é o número de elementos do simplexo.
L	Lista de prioridade
LFS	*Local Feature Size*
m	Dimensão de um simplexo
M	Superfície do objeto
n	Dimensão de um simplexo. Também é utilizado como constante na definição da complexidade dos algoritmos
NNG	Grafo do vizinho mais próximo
$\hat{o}$	Ortocentro de um tetraedro.
$\hat{p}$	Ponto com peso
p	Ponto sem peso e também pólo calculado nos algoritmos *raw crust* e *power crust*
p_m	Ponto sem peso
q	Pólo calculado nos algoritmos *raw crust* e *power crust*
r	Utilizado na definição de *r-regular shape* e r-amostagem. Também utilizado como raio de um circunferência.

R	Raio de uma circunferência.
S	Conjunto de pontos da amostra de uma superfície
S'	Triangulação de Delaunay para um conjunto de pontos da amostra S
s	Ponto da amostra
t	Ponto da amostra
v	Vértice de um simplexo
V	Conjunto de Vértices de um simplexo
Vor	Diagrama de Voronoi
Vor_w	Diagrama de Voronoi com peso
w	Peso de um ponto com peso, além de raio da bola polar definida por esse ponto com peso
x	Ponto sem peso
X	Conjunto pontos $x_0...x_n$, onde n é o número total de pontos
$Z(f)$	Conjunto zero da função f
α	Constante utilizada na definição do α-*shape*. Também utilizado como ângulo
β	Constante utilizada na definição do β-*Skeleton*. Também utilizado como ângulo
δ	Ângulo utilizado em teoremas e cálculos
ϵ	Ângulo utilizado em teoremas e cálculos
θ	Ângulo utilizado em teoremas e cálculos
ρ	Raio da bola polar B_{p_i,ρ_i}
σ	k-simplexo de um conjunto de k+1 pontos
τ	Face de um k-simplexo

Capítulo 1

Introdução

Dentro da área de Modelagem Geométrica em Computação Gráfica, uma das maneiras de se obter modelos geométricos complexos é reconstruir a superfície do objeto a partir de uma coleção finita de pontos. Este problema é bastante comum em diversas aplicações e nos últimos anos vem se tornando uma área de grande importância em aplicações como Computação Gráfica, Visão Computacional, Cartografia, aplicações para a medicina e outras aplicações industriais, como Engenharia Reversa. Um dos motivos desta evolução é o desenvolvimento de *scanners* a *laser* e outras tecnologias capazes de coletar amostras de conjuntos de pontos cada vez maiores de superfícies de objetos reais. A partir desta evolução e do fato da solução deste problema não ser simples, vários grupos de pesquisa em todo o mundo vêm desenvolvendo diversos métodos para solucionar e adaptar a reconstrução de superfície em suas aplicações específicas. Estes métodos podem ser agrupados de várias formas, sendo a mais importante aquela que classifica os algoritmos segundo a abordagem inicial utilizada no processo.

Os métodos baseados em esculpimento são aqueles que geram uma malha de triângulos a partir de um triângulo inicial. Os métodos baseados em funções implícitas definem uma função de distância baseada nos pontos da amostra (representação volumétrica), segunda a qual a superfície é aproximada. Os métodos incrementais constroem o objeto de forma incremental a partir de um elemento inicial (ponto, aresta ou triângulo). E, por último, os métodos baseados em modelos deformáveis são aqueles que geram a superfície resultante a partir da deformação de uma superfície inicial dada pelos pontos da amostra.

Alguns desses métodos requerem informações adicionais do objeto a ser reconstruído, como, por exemplo, os vetores normais dos pontos da amostra. Estes dados devem ser adquiridos durante o processo de aquisição dos pontos através de algum dispositivo, como, por exemplo, um *scanner* 3D. Neste trabalho, serão abordados apenas algoritmos para reconstrução de superfícies que não contêm nenhum tipo de informação adicional sobre a organização ou estrutura espacial dos pontos do objeto a ser modelado, sendo neste caso, algoritmos mais complexos.

Os algoritmos de reconstrução também podem ser divididos em duas classes: algoritmos de aproximação e algoritmos de interpolação. Na primeira classe, a entrada para o algoritmo é um conjunto de pontos que não pertencem à superfície reconstruída. Neste caso, os pontos são utilizados apenas para guiar o processo de reconstrução. Nos algoritmos de interpolação, os pontos da entrada pertencem necessariamente à superfície reconstruída.

Formalmente, a reconstrução de superfície pode ser definida da seguinte forma: dado um conjunto finito de pontos (amostra S) de uma superfície M, encontrar uma maneira de conectar os pontos da amostra S com o objetivo de gerar uma aproximação poligonal para a superfície M. Este conceito foi inicialmente introduzido por Hoppe et al. em [Hoppe et al., 1992]. O algoritmo proposto em [Hoppe et al., 1992] é baseado numa função de distância definida a partir dos k vizinhos mais próximos de um ponto da amostra (abordagem baseada em função implícita). A partir daí, importantes algoritmos de reconstrução de superfícies foram desenvolvidos, como o algoritmo de Curless e Levoy [Curless and Levoy, 1996], o α-*shape* de Edels-

brunner [Edelsbrunner and Mücke, 1994], o algoritmo de Bernardini [Bernardini et al., 1999], os algoritmos da família *Crust* [Amenta et al., 1998], [Amenta and Bern, 1999b] e [Amenta et al., 2001a], entre outros que serão abordados neste trabalho.

Uma vez definido o problema de reconstrução de superfície, é necessário estabelecer uma relação entre a qualidade da amostra dos pontos e a superfície reconstruída, pois esta relação influencia diretamente na eficiência do processo de reconstrução de superfícies. Segundo [Amenta and Bern, 1999b], uma "boa amostra" é aquela que é densa nas regiões que caracterizam o objeto e esparsa nas regiões menos características. De uma forma geral, as regiões que delimitam o objeto são consideradas mais importantes que suas regiões internas. Isto porque, as fronteiras são as responsáveis pela caracterização da superfície do objeto. Desta forma, em [Amenta and Bern, 1999b] foi introduzido o conceito de *LFS - Local Feature Size* como medida de qualidade da amostra, que será posteriormente definido no Capítulo 2.

No entanto, segundo [Amenta and Bern, 1999b], mesmo que a amostra seja suficientemente densa, é possível encontrar dificuldades no processo de reconstrução. Esta dificuldade pode ocorrer quando a superfície não está totalmente preenchida, quando a amostra contém ruídos ou quando não é densa o suficiente em partes representativas da superfície. Portanto, mesmo dada uma boa amostra, o algoritmo pode não ser tão robusto, podendo apresentar também um alto custo computacional. Na figura 1.1 é mostrado um exemplo deste processo. Neste exemplo, a entrada do algoritmo de reconstrução é um conjunto de pontos $3D$ proveniente de um dispositivo de aquisição de pontos, como por exemplo, um *scanner* 3D. A saída do algoritmo é uma malha de polígonos que representa uma aproximação da superfície.

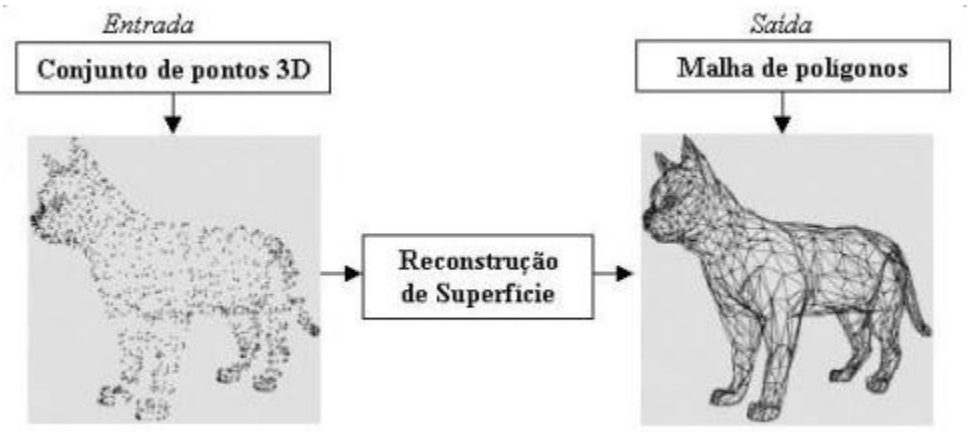

Figura 1.1: Exemplo do processo de reconstrução de superfícies.

1.1 Motivação e objetivos

A principal motivação para realização deste trabalho foi o desenvolvimento de um algoritmo de reconstrução de superfície que pudesse ser integrado ao modelador de sólidos *GSM - GOPAC Solid Modeler*, em desenvolvimento pelo grupo de pesquisa *GOPAC Grupo de Otimização e Projeto Assistido por Computador* do Departamento de Engenharia Elétrica da UFMG.

O *GSM- GOPAC Solid Modeler* é um modelador de sólidos que oferece suporte à criação e manipulação de objetos geométricos tridimensionais, para fins de cálculos eletromagnéticos. Esses objetos podem ser obtidos através de varredura sobre primitivas bidimensionais ou através da construção de modelos por operações booleanas. Além disso, é possível gerar a representação desses modelos em malha volumétrica para análise de simulação eletromagnética, pelo método de elementos finitos.

O objetivo deste trabalho é disponibilizar a construção desses modelos a partir de uma amostra representativa de pontos, sem qualquer informação adicional a respeito do objeto a ser modelado. Esta será uma forma simples de se gerar modelos de objetos diretamente no *GSM*, para que esses possam ser utilizados na obtenção de malhas de elementos finitos, utilizadas no cálculo de campos eletromagnéticos.

O nosso objetivo inicial foi realizar um estudo dos diversos métodos utilizados na reconstrução de superfícies, para que assim fosse possível implementar os algoritmos da família *Crust*: *raw crust 2D*, *raw crust 3D* e *power crust 3D* [Amenta and Bern, 1999b], [Amenta et al., 2001a]. Após a implementação, realizar uma comparação entre o algoritmo *crust* e *power crust*, avaliando as vantagens e desvantagens de cada um deles.

Posteriormente, elaborar uma etapa de pós-processamento para suavização da malha de polígonos resultantes. Essa suavização tem o objetivo de tornar a malha de polígonos o mais regular possível, possibilitando sua utilização na geração de malhas de elementos finitos.

Por fim, a principal contribuição deste trabalho é a elaboração de uma biblioteca robusta, capaz de gerar modelos tridimensionais, utilizando apenas amostras de pontos da superfície a ser modelada. Essa biblioteca estará disponível no modelador de sólidos *GSM*, e será uma forma adicional de se obter modelos de objetos tridimensionais a serem utilizados no cálculo de campos eletromagnéticos.

1.2 Organização da dissertação

Para abordar o desenvolvimento deste trabalho, a dissertação está dividida nas seguintes partes:

No Capítulo 2 são introduzidos alguns conceitos básicos da geometria computacional necessários para o entendimento dos algoritmos tratados posteriormente.

No Capítulo 3 é feito um levantamento bibliográfico dos principais métodos de reconstrução de superfícies. Estes métodos foram divididos em quatro classes: métodos baseados em esculpimento, métodos baseados em funções implícitas, métodos incrementais e métodos baseados em modelos deformáveis.

Os algoritmos de esculpimento são aqueles baseados na triangulação de Delaunay ou no seu dual (diagrama de Voronoi). Neste caso, a superfície é construída através da extração de polígonos gerados inicialmente pela triangulação. O outro grupo de algoritmos são aqueles que utilizam funções implícitas. Neste tipo de algoritmo uma função de distância com sinal f_d é definida e calculada baseada no conjunto de pontos da amostra. A partir daí, a superfície é estimada segundo o cálculo do conjunto zero de f_d, denominado $Z(f_d)$.

Numa abordagem diferente, os métodos incrementais são iniciados a partir da escolha de um triângulo, e através de um processo iterativo, novos triângulos são adicionados ao resultado.

Por último, os algoritmos baseados em modelos deformáveis são aqueles que deformam uma superfície inicial até conseguir uma aproximação da superfície que originou os pontos da amostra.

No Capítulo 4 é feita uma abordagem mais detalhada dos algoritmos de esculpimento da família *Crust*. Neste capítulo são mostrados os resultados teóricos e as garantias de reconstrução para cada algoritmo.

No Capítulo 5 são mostrados os detalhes das implementações realizadas (*raw crust 2D*, *raw crust 3D* e *power crust 3D*), os resultados obtidos, bem como as ferramentas utilizadas para o desenvolvimento e visualização desses resultados.

Baseado nos estudos de cada método de reconstrução e nas implementações realizadas, no Capítulo 6 são apresentadas as conclusões deste trabalho e os principais direcionamentos futuros.

Capítulo 2

Elementos básicos da geometria computacional

2.1 Considerações Iniciais

Neste capítulo são apresentados alguns conceitos de geometria computacional necessários para o entendimento dos algoritmos de reconstrução de superfície abordados nos próximos capítulos.

2.2 Definições

A primeira definição apresentada neste trabalho é a do fecho convexo. O cálculo do fecho convexo é utilizado em algoritmos geométricos com o objetivo de organizar e agrupar um conjunto de elementos (vértices, arestas, faces) numa região com estrutura mais simples. Posteriormente ele será utilizado na definição da triangulação de Delaunay.

Definição 2.1 - Fecho convexo: O fecho convexo $CONV(S)$ de um conjunto finito de pontos $\mathbf{S} \in \Re^{\mathbf{n}}$ é a menor região convexa de $\Re^{\mathbf{n}}$ que contém o conjunto $\mathbf{S}$.

Em $2D$, o fecho convexo é um polígono formado por um conjunto de vértices e arestas. Em $3D$, tem-se um poliedro formado por um conjunto de faces. Em dimensões maiores o fecho convexo deverá ser representado pelo conjunto de faces de cada dimensão. Na figura 2.1 é mostrado o fecho convexo de um conjunto de pontos no plano. Para o conjunto de pontos $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9$, o fecho convexo é representado pelo polígono formado pelos pontos p_8, p_5, p_4, p_2, p_9.

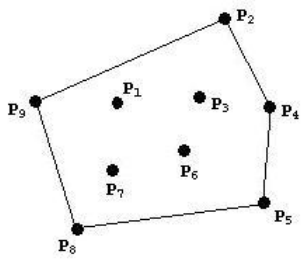

Figura 2.1: Fecho convexo de um conjunto de pontos no plano.

As próximas definições a serem apresentadas estão relacionadas ao conceito de **complexo simplicial** e **complexo celular** que posteriormente serão utilizados para definirmos o diagrama de Voronoi e a triangulação de Delaunay, sendo estes os principais conceitos utilizados no desenvolvimento deste trabalho.

Definição 2.2 - Posição geral: Os pontos de um conjunto S, em três dimensões, estão em posição geral quando não existe nenhuma linha que contenha mais que dois pontos da amostra, nenhum plano que não contenha mais que três pontos e nenhuma esfera que não contenha mais que quatro pontos da amostra.

Definição 2.3 - *k*-simplexo: Seja S um conjunto de $k+1$ pontos pertencentes a $\Re^n$, sendo esses *linearmente independentes*, um k-simplexo σ é o fecho convexo de $S : CONV(S)$.

Na figura 2.2 são apresentados quatro exemplos de simplexos, sendo respectivamente:

0-simplexo: um ponto;

1-simplexo: um segmento;

2-simplexo: um triângulo;

3-simplexo: um tetraedro.

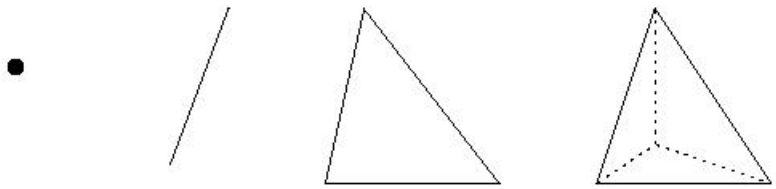

Figura 2.2: Da esquerda para direita são mostrados respectivamente um *0-simplexo*, *1-simplexo*, *2-simplexo* e *3-simplexo*.

Definição 2.4 - Face de um simplexo: Seja σ um k-simplexo formado pelos pontos $X = x_0, ..., x_n$. Qualquer l-simplexo τ dado por um subconjunto de $l + 1$ elementos de X é face de σ, para $l < k$.

Definição 2.5 - Complexo simplicial: Um complexo simplicial C_σ é uma coleção de simplexos que satisfazem as seguintes condições:

1. Dado um simplexo $\sigma \in C_\sigma$, cada face τ que pertence a σ também pertence a C_σ;
2. Se σ e τ pertencem a C_σ então eles não têm nenhuma interseção ou então compartilham uma face em comum.

A dimensão de um complexo simplicial C_σ é dada pela máxima dimensão de seus simplexos. Um exemplo de complexo simplicial é mostrado na figura 2.3 **(a)**. Na mesma figura (2.3 **(b)**) é dado um exemplo de uma estrutura que não constitui um complexo simplicial, uma vez que esta apresenta interseções que não definem uma face.

Definição 2.6 - Célula: Dado um conjunto de pontos $X = x_0, ..., x_n$, uma célula C convexa é formada pelo fecho convexo de X.

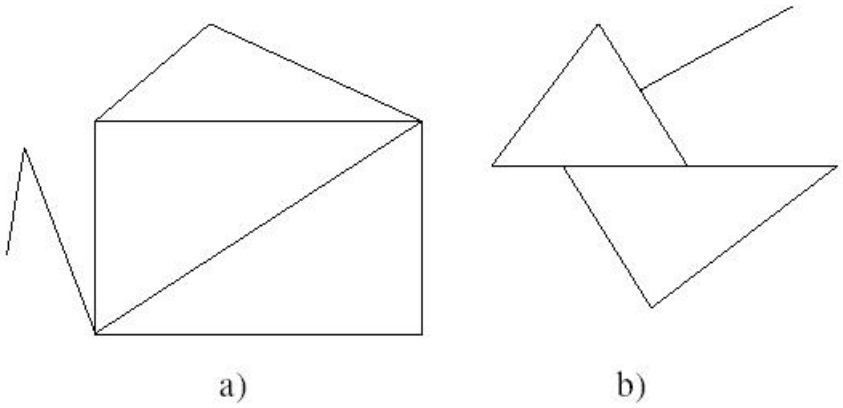

Figura 2.3: a) Exemplo de um complexo simplicial, b) Não define um complexo simplicial. Note que existem intersecções que não definem uma face. Fonte: [Gois, 2004]

A dimensão de uma célula S é o maior número de vetores linearmente independentes $x_1 - x_0, ..., x_n - x_0$. Em particular, as células de dimensão 1 são chamadas de *arestas* e as de dimensão 0 são denominadas *vértices*.

Definição 2.7 - Complexo celular: Seja C_c uma coleção finita de células $C \in \Re^n$. C_c é um complexo celular de um conjunto finito de células $X \in \Re^n$ se forem satisfeitas as seguintes condições:

1. X = $\bigcup C$, onde $C \subset C_c$;
2. Se C_1 e $C_2 \in C_c$, então $C_1 \cap C_2 = \emptyset$, ou $C_1 \cap C_2 \in C_c$ é uma face de C_1 e C_2.

Os resultados dos algoritmos de reconstrução apresentados neste trabalho geram saídas que são complexos celulares ou complexos simpliciais.

Definição 2.8 - Diagrama de Voronoi: Seja S um conjunto de pontos no espaço. O diagrama de Voronoi de S é uma subdivisão do espaço em células, onde cada célula consiste de todos os pontos mais próximos de um ponto específico do que qualquer outro. Podemos considerar $Vor(S_i)$ a célula que define o ponto $S_i \in S$ e $Vor(S)$ o conjunto de todas as células de Voronoi. Em $\Re^n$, o diagrama de Voronoi pode ser definido como:

$$Vor(S_i) = x \in \Re^n : \forall S_k \neq S_i \in S, d(x, S_i) \leq d(x, S_k);$$
$$Vor(S) = \bigcup_{S_i \in S} Vor(S_i)$$

O diagrama de Voronoi em duas dimensões possui uma propriedade importante: cada vértice de Voronoi é o centro de um círculo que passa por três vértices da amostra S e este não contém nenhum outro vértice em seu interior. Esta propriedade pode ser visualizada na figura 2.4 onde é dado um exemplo do diagrama de Voronoi no plano. Observe que em $2D$, o diagrama de Voronoi é um conjunto de polígonos convexos.

A propriedade descrita anteriormente, quando estendida para $3D$ é definida da seguinte forma: cada vértice de Voronoi de um conjunto de pontos $S \in \Re^3$ é o centro de uma esfera vazia de raio máximo que tangencia ao menos quatro pontos de S. Quando os pontos de S estão em posição geral, então esta esfera é definida por quatro pontos exatamente. A esfera correspondente a um vértice de Voronoi é denominada ***esfera polar***. Em $3D$, o diagrama de Voronoi resulta num conjunto de politopos convexos assim como em dimensões superiores.

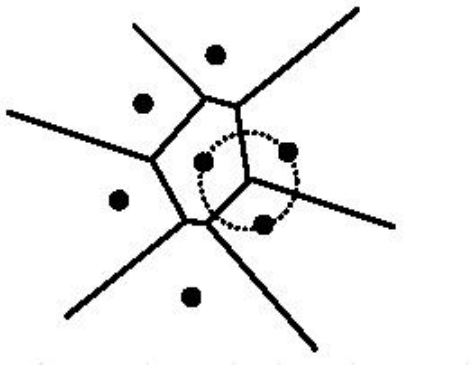

Figura 2.4: Exemplo do diagrama de Voronoi no plano de um conjunto de pontos S. Cada célula de Voronoi é um polígono formado por um conjunto de segmentos ou semi-retas. As esferas de Voronoi são círculos, o centro de um círculo que passa por três pontos de S é um vértice v de Voronoi.

A função de distância utilizada nas definições anteriores é a distância Euclidiana. Maiores detalhes sobre algoritmos para o cálculo do diagrama de Voronoi podem ser encontrados em [Berg et al., 2000].

A partir da definição do diagrama de Voronoi podemos introduzir o conceito de *eixo medial* e *Local Feature Size*. Este último é utilizado para definir as garantias teóricas dos algoritmos da família *Crust* no Capítulo 4.

Definição 2.9 - Eixo medial: O eixo medial de uma superfície fechada é definido como um conjunto de todos os pontos no espaço que são eqüidistantes no mínimo a dois pontos da superfície.

Podemos dizer que o eixo medial é o esqueleto da superfície em questão, ou seja, a estrutura que dá forma ao objeto.

Em duas dimensões, se a amostra S de uma curva é densa, então o eixo medial é formado por todos os vértices de Voronoi de S. Na figura 2.5 é mostrado um exemplo do eixo medial em duas dimensões representado pelas linhas pontilhadas.

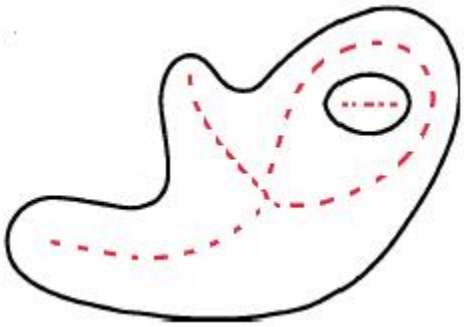

Figura 2.5: O eixo medial da curva é dado pelas linha pontilhadas. Fonte: [Amenta and Bern, 1999a]

Em três dimensões, o eixo medial não pode ser calculado apenas fazendo-se o cálculo do diagrama de Voronoi, pois alguns vértices do diagrama podem aparecer muito próximos à superfície, sendo estes não pertencentes ao eixo medial. No entanto, segundo [Amenta and Bern, 1999b], a maioria dos vértices de Voronoi encontra-se próxima ao eixo medial. Na figura 2.6, é mostrado um exemplo do eixo medial de uma superfície em três dimensões. Observe que o eixo medial neste caso é uma superfície, e não uma curva.

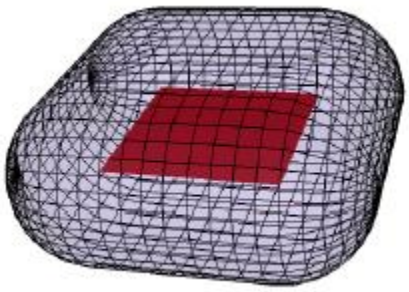

Figura 2.6: Exemplo do eixo medial de uma superfície $3D$. O eixo medial é mostrado na figura pela superfície $2D$ em destaque dentro da superfície transparente. Fonte: [Amenta and Bern, 1999a].

Definição 2.10 - *Local Feature Size*: Seja F uma curva suave e p um ponto pertencente a F. O *Local Feature Size* de p - $LFS(p)$ é a distância Euclidiana de p ao ponto p_m mais próximo pertencente ao eixo medial de F. $LFS(p) = d(p, p_m)$.

Outro importante conceito relacionado a este trabalho é a triangulação de Delaunay. Inicialmente será introduzido o conceito de triangulações e em seguida serão definidas as condições que fazem com que uma triangulação qualquer seja uma triangulação de Delaunay.

Definição 2.11 - Triangulação: Seja um conjunto de pontos $P \in \Re^n$. Uma triangulação de P de dimensão $j \leq n$ é um complexo simplicial onde todo simplexo de dimensão $m < j$ está contido em algum simplexo de dimensão j e os vértices da triangulação são pontos de P.

Definição 2.12 - Triangulação de Delaunay: Seja $S = S_0, ..., S_k \in \Re^n$, $n \leq k$, dizemos que uma triangulação de S é de Delaunay ($DEL(S)$) se a circunsfera de todo n-simplexo não contém nenhum outro ponto de S em seu interior.

A figura 2.7 mostra um exemplo da triangulação de Delaunay no plano.

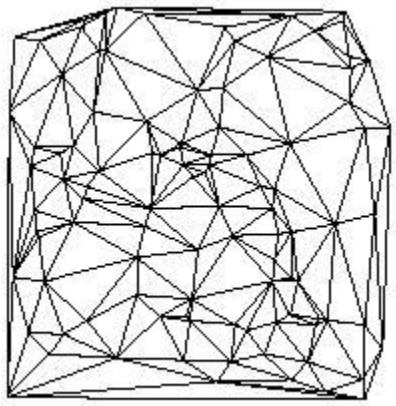

Figura 2.7: Exemplo da triangulação de Delaunay no plano com 100 pontos. Fonte: [Gois, 2004]

Dado um conjunto de pontos S, a triangulação de Delaunay de S possui as seguintes propriedades:

1. Todo simplexo da triangulação deve estar contido no fecho convexo de S;

2. A circunsfera de cada simplexo não deve conter nenhum outro vértice da triangulação;

3. É dual do diagrama de Voronoi;

4. A Triangulação de Delaunay será única se e somente se o conjunto de pontos, para o qual desejamos obter a triangulação, estiver em posição geral.

Em duas dimensões, a triangulação de Delaunay maximiza o ângulo mínimo de cada triângulo e pode ser obtida em $O(n \log n)$.

Podemos estabelecer algumas relações de dualidade entre a triangulação de Delaunay e o diagrama de Voronoi:

1. Cada vértice do diagrama de Voronoi corresponde a um k-simplexo da triangulação de Delaunay. A figura 2.8 mostra um exemplo desta relação no plano. A triangulação de Delaunay é dada pelos triângulos formados pelos vértices $P_1..P_7$, e o diagrama de Voronoi é dado pelos polígonos formados pelos vértices $V_1..V_6$. Observe que cada triângulo está associado a um vértice V_i do diagrama de Voronoi. Por exemplo, o triângulo em negrito está relacionado ao vértice de Voronoi V_1. Em três dimensões cada vértice de Voronoi está associado a um tetraedro da triangulação de Delaunay, em $\Re^n$ cada vértice de Voronoi está associado a um n-simplexo da triangulação de Delaunay.

2. Cada aresta do diagrama de Voronoi corresponde a uma aresta da triangulação em $2D$, e a um triângulo da triangulação em $3D$. Por exemplo, na figura 2.8 a aresta formada pelos vértices de Voronoi V_6, V_1 é dual da aresta formada pelos pontos P_1, P_7 da triangulação de Delaunay.

3. Em $3D$, cada face do diagrama de Voronoi corresponde a uma aresta da triangulação de Delaunay.

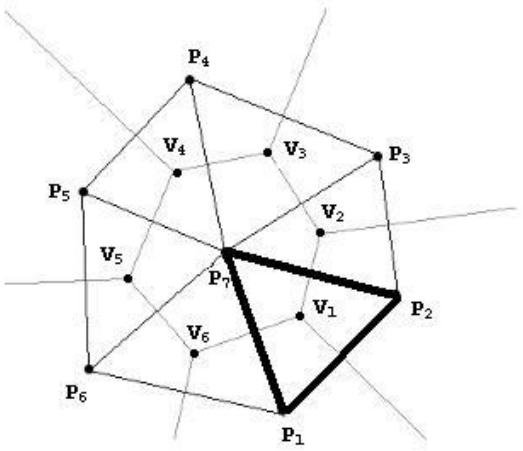

Figura 2.8: Exemplo da relação de dualidade entre a triangulação de Delaunay (pontos $P_1..P_7$) e o diagrama de Voronoi (vértices $V_1..V_6$) no plano.

As próximas definições estão relacionadas à triangulação com peso e ao diagrama de Voronoi com peso.

Definição 2.13 - Ponto com peso: Um ponto com peso $\hat{p}$ é definido pela sua localização no espaço p e um peso w^2 associado: $\hat{p} = (p, w^2)$.

Definição 2.14 - Distância com peso: A distância com peso entre dois pontos $\hat{p_1} = (p_1, w_1^2)$, $\hat{p_2} = (p_2, w_2^2)$ é dada por $\mid \hat{p_1} - \hat{p_2} \mid_w = \mid p_1 - p_2 \mid^2 - w_1^2 - w_2^2$, onde $\mid p_1 - p_2 \mid^2$ é o quadrado da distância Euclidiana entre os pontos p_1 e p_2.

Em $2D$, podemos interpretar um ponto com peso $\hat{p} = (p, w^2)$ como sendo um círculo de centro em p e raio w. Esta interpretação é mostrada na figura 2.9 pelos pontos com peso $\hat{p_1}$ e $\hat{p_2}$. Em $3D$, esta interpretação pode ser feita através de esferas.

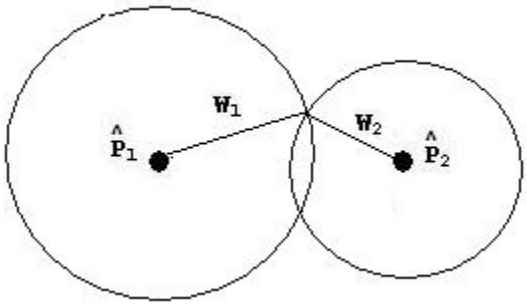

Figura 2.9: Representação dos pontos com peso $\hat{p_1}$ e $\hat{p_2}$ em círculos.

Quando a distância com peso entre dois pontos $\hat{p_1}$ e $\hat{p_2}$ é zero, dizemos que os dois pontos são ortogonais. Verifica-se facilmente, em duas dimensões, que $||\hat{p_1} - \hat{p_2}|| = 0$, se e somente se, os dois círculos se encontram em pontos onde suas respectivas tangentes formam um ângulo reto. Na figura 2.10 é mostrado um exemplo em que os pontos com peso $\hat{p_1}$ e $\hat{p_2}$ são ortogonais. Observe que o quadrado da distância Euclidiana d^2 entre $\hat{p_1}$ e $\hat{p_2}$ é dada por $d^2 = w_1^2 + w_2^2$. Como a distância com peso é definida por $\mid \hat{p_1} - \hat{p_2} \mid_w = \mid p_1 - p_2 \mid^2 - w_1^2 - w_2^2$, decorre que, $\mid \hat{p_1} - \hat{p_2} \mid_w = 0$.

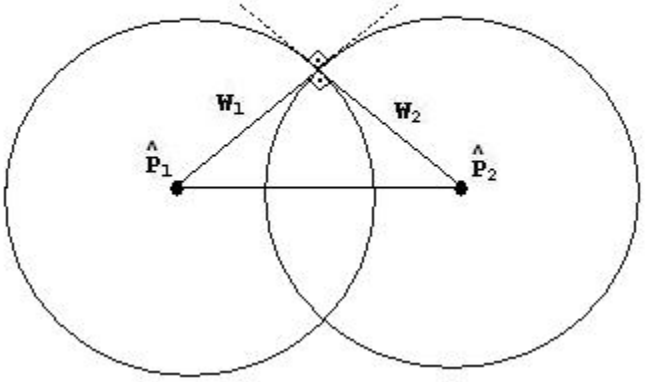

Figura 2.10: Pontos com peso $\hat{p_1}$ e $\hat{p_2}$ ortogonais.

Em $3D$, $\mid \hat{p_1} - \hat{p_2} \mid_w = 0$, se e somente se, as duas esferas se encontram em um círculo e os dois planos tangentes às esferas em cada ponto deste círculo formam um ângulo reto. A ortogonalidade é o conceito chave na generalização da triangulação de Delaunay para a triangulação de Delaunay com peso.

A partir da definição de distância com peso também podemos introduzir o conceito de *power* distância.

Definição 2.15 - *Power* distância: Dado um ponto sem peso $x \in \Re^3$ e um ponto com peso $\hat{P} = (p, w^2)$, onde p é a localização do ponto no espaço e w^2 o peso associado, a *power* distância d_{pow} entre eles é:

$$d_{pow}(x, \hat{P}_{p,w}) = d(p, x)^2 - w^2,$$

onde d representa a distância Euclidiana.

Desta forma, quando x estiver dentro da esfera $\hat{P}_{p,w}$, d_{pow} será negativa, e quando x estiver fora, d_{pow} será positiva. Podemos utilizar a *power* distância para definir um tipo particular de diagrama de Voronoi, o diagrama de Voronoi com peso, também conhecido por *power* diagrama.

Definição 2.16 - *Power* diagrama: Seja S um conjunto de pontos com peso no espaço. O *power* diagrama ou diagrama de Voronoi com peso $Vor_w(S)$ é a subdivisão do espaço em células, onde cada célula consiste de todos os pontos $x \in S$ mais próximos de um ponto $p \in Vor_w(S)$ específico do que qualquer outro, considerando a *power* distância para o cálculo desta distância.

Assim como o dual do diagrama de Voronoi tradicional é a triangulação de Delaunay, o dual do *power* diagrama é a triangulação de Delaunay com peso, também conhecida como triangulação regular.

Definição 2.17 - Triangulação de Delaunay com peso: Define-se por triangulação de Delaunay com peso ou triangulação regular o dual do *power* diagrama.

Considerando pontos sem peso, um triângulo pertence à triangulação de Delaunay, se e somente se, a circunferência que passa por três vértices da triangulação não contém nenhum outro ponto da triangulação. Para pontos com peso em três dimensões, a circunferência é substituída pelo conceito de *ortoesfera*. Uma *ortoesfera* é uma esfera ortogonal a todas as quatro esferas cujos centros são vértices de um tetraedro. Cada centro de uma esfera é um vértice do diagrama de Voronoi e seu respectivo peso é a distância com peso entre o vértice e os centros das quatro esferas. Este conceito é ilustrado na figura 2.11, onde a triangulação de Delaunay com peso no plano é representada pelas linhas contínuas e o *power* diagrama pelas linhas tracejadas. Observe que destacamos apenas um círculo em pontilhado cujo centro é um vértice de Voronoi. Este círculo intercepta três regiões do diagrama de Voronoi com peso representadas pelas circunferências com centros em negrito.

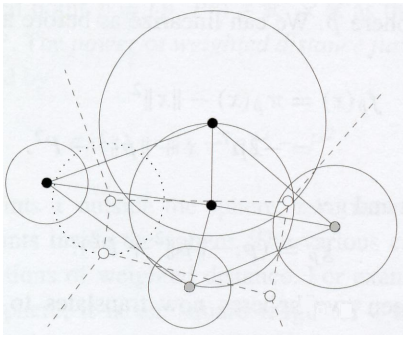

Figura 2.11: As linhas contínuas formam a triangulação com peso de quatro pontos com peso no plano. As linhas tracejadas representam seu respectivo diagrama de Voronoi. Cada vértice de Voronoi é o centro de um círculo ortogonal aos círculos das regiões que delimitam seu vértice. Fonte: [Edelsbrunner, 2001]

Na figura 2.12 é mostrada a triangulação de Delaunay com peso em duas dimensões. Podemos obervar que neste tipo de triangulação podem existir vértices sem arestas conectadas a ele (não existe o seu respectivo dual), vértice p, por exemplo.

As próximas definições a serem apresentadas constituem subconjuntos da triangulação de Delaunay utilizados como heurísticas para reconstrução de superfícies a partir de nuvens de pontos, descritas no capítulo 3.

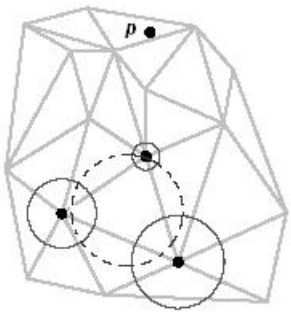

Figura 2.12: Exemplo da triangulação de Delaunay com peso. Os pontos com peso são representados pelos círculos contínuos. O ponto p é um exemplo de um vértice que foi removido da triangulação. O círculo tracejado é a representação de um ortocírculo.

Definição 2.18 - Grafo do Vizinho mais Próximo: Seja um conjunto de pontos P, o grafo do vizinho mais próximo (*Nearest Neighbour Graph, NNG*) de P é o grafo $NNG(P) = (P, E)$ tal que $E \subseteq P \times P$ e $E = e_i = (p_i, p_j), i = 1, ..., n : p_j$ é o ponto de P mais próximo de p_i.

Definição 2.19 - Árvore Geradora Mínima Euclidiana: A árvore geradora mínima Euclidiana (*Euclidean Minimun Spanning Tree, EMST*) é a árvore $EMST(P, E) = (P, E)$, tal que $E \subseteq P \times P$ e $E = e_i = (p_i, p_j), i = 1, ..., n : \sum \|pi - pj\|$ é mínima.

Definição 2.20 - Grafo de Gabriel: O grafo de Gabriel de P é o grafo $GG(P) = (P, E)$, definido por $E \subseteq P \times P$ e $E = e_i = (p_i, p_j), i = 1, ..., n :$ a menor esfera tocando p_i e p_j não contém qualquer outro ponto de P.

A partir das definições anteriores e considerando $EDT(P)$ o conjunto das arestas da triangulação de Delaunay dos pontos pertencentes a P, tem-se a seguinte relação de inclusão:

$$NNG(P) \subseteq EMST(P) \subseteq GG(P) \subseteq EDT(P)$$

Definição 2.21 - Hipergrafo de Gabriel Estendido (EGH): Sejam P, E e T respectivamente um conjunto de vértices, arestas e triângulos de um grafo. E seja $GG(P) = (P, E_{GG})$, e $EGH(P) = (P, E_{EGH}, T)$, onde P, E e T são, respectivamente o conjunto de vértices, de arestas e de triângulos do grafo. Seja o conjunto de arestas iniciais de EGH o mesmo de GG, o conjunto de triângulos é construído da seguinte maneira:

1. Se $\forall e_1, e_2 \in E_{GG}$ não colineares, sendo $e_1 = (v_1, v_2)$, $e_2 = (v_2, v_3)$, e se a menor esfera que passa por v_1, v_2 e v_3 não contém quaisquer outros pontos de P, então $e_3 = (v_1, v_3) \in E_{EGH}$;
2. Qualquer ciclo de três arestas em E_{EGH} é um triângulo em T. Em $\Re^3$, EGH também é um subconjunto da triangulação de Delaunay.

Em [Attene and Spagnuolo, 2000], são encontrados maiores detalhes destas estruturas e definições.

As duas últimas definições dadas em seguida serão utilizadas nos capítulos 3 e 4, e estão relacionadas ao algoritmo β-*Skeleton* proposto por Nina Amenta em [Amenta et al., 1998]. O β-*Skeleton* baseia-se no conceito de região proibida, considerando uma constante $\beta \geq 1$.

Definição 2.22 - Região Proibida: Seja S um conjunto finito de pontos no plano, com pontos p_1 e p_2 pertencentes a S, sendo $d(p_1, p_2)$ a distância entre eles. A região proibida de p_1, p_2 é a união dos dois discos de raio $\frac{\beta * d(p_1, p_2)}{2}$ tocando p_1 e p_2.

Na figura 2.13 são mostrados dois exemplos de região proibida, considerando os valores do parâmetro β igual a 1 e maior que 1 respectivamente.

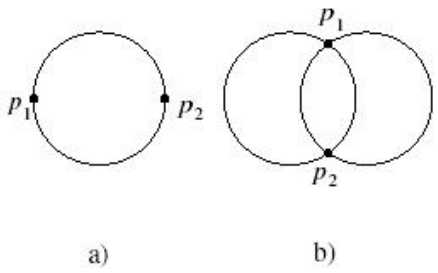

Figura 2.13: Exemplo de região proibida. (a) $\beta = 1$, (b) $\beta > 1$.

Definição 2.23 - β-*Skeleton*: Seja S um conjunto finito de pontos no plano, com pontos p_1 e p_2 pertencentes a S. A aresta p_1, p_2 pertence ao β-*Skeleton* de S se a região proibida de p_1, p_2 estiver vazia.

Na figura 2.14 é mostrado um exemplo do β-*Skeleton* em duas dimensões, considerando $\beta > 1$. A aresta definida pelos vértices p_1, p_2 pertence ao β-*Skeleton*, uma vez que a região proibida entre p_1, p_2 não contém nenhum outro vértice da triangulação.

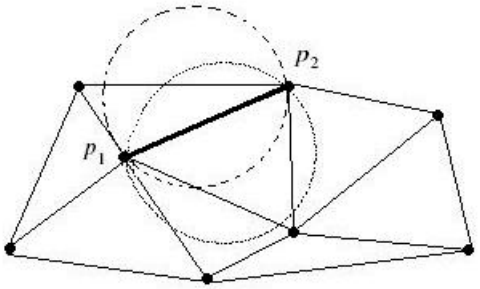

Figura 2.14: A aresta definida pelos vértices p_1, p_2 pertence ao β-*Skeleton* considerando $\beta > 1$.

2.3 Considerações finais

Neste capítulo foram apresentadas as principais definições relacionadas à geometria computacional que serão empregadas no decorrer do trabalho. Demais definições serão apresentadas ao longo do texto de acordo com a necessidade. No próximo capítulo será apresentada a primeira parte do levantamento bibliográfico sobre reconstrução de superfícies.

Capítulo 3

Métodos para Reconstrução de Superfícies

3.1 Considerações iniciais

O problema de reconstrução de superfície pode ser definido da seguinte forma: seja uma superfície S contida em $\Re^3$, dada uma amostra de pontos da superfície de S, construir uma superfície topologicamente próxima ou equivalente a S.

Neste capítulo será apresentada uma descrição geral dos principais métodos utilizados na reconstrução de superfície, e, no próximo capítulo serão detalhados os algoritmos da família *Crust*, uma vez que estes possuem resultados práticos satisfatórios e garantias teóricas de reconstrução.

Os algoritmos de reconstrução em geral podem ser divididos em duas classes: algoritmos de aproximação e algoritmos de interpolação. Algoritmos de aproximação são aqueles que utilizam os pontos da amostra apenas para guiar o processo de reconstrução. Esses pontos não serão necessariamente os pontos da superfície reconstruída, mas serão próximos, daí a idéia de "aproximação". No segundo caso, algoritmos de interpolação, os pontos de entrada pertencem necessariamente à superfície reconstruída.

Outra forma de agrupar os algoritmos de reconstrução de superfície é através de seus métodos utilizados. Desta forma, podemos distinguir quatro tipos de abordagens utilizadas na reconstrução de superfície:

1. Métodos baseados em esculpimento;
2. Métodos baseados em funções implícitas;
3. Métodos incrementais;
4. Métodos baseados em modelos deformáveis;

Os métodos de esculpimento são aqueles baseados na triangulação de Delaunay ou no diagrama de Voronoi. Esse tipo de abordagem se baseia em dois passos: primeiramente é feita a construção de uma estrutura geométrica a partir de um conjunto finito de pontos da amostra utilizando a triangulação de Delaunay ou o diagrama de Voronoi. Em seguida é feita uma extração de um conjunto de faces da estrutura construída no passo 1, a fim de promover a aproximação da superfície dada.

Uma outra abordagem comum na reconstrução de superfície é a utilização de pontos da amostra para definir uma função de distância com sinal f_d, e, a partir da função zero $Z(f_d)$, gerar os polígonos resultantes. Esses tipos de algoritmo, baseados em funções implícitas, baseiam-se em aproximação e não em interpolação, como é o caso dos algoritmos de esculpimento. Por se tratar de uma aproximação, esses tipos de algoritmo podem provocar uma perda de informação durante o processo de reconstrução, sendo necessária uma etapa de pós-processamento para filtragem dos buracos resultantes.

Numa diferente abordagem, os métodos incrementais são aqueles inicializados a partir da escolha de um triângulo, e, através de um processo iterativo, novos triângulos são adicionados ao resultado. A principal etapa deste processo é a escolha do triângulo inicial, a partir do qual a superfície é reconstruída.

Os métodos baseados em modelos deformáveis começam com uma superfície inicial, em seguida é aplicada uma série de deformações até se conseguir uma aproximação para a nuvem de pontos.

3.2 Métodos baseados em esculpimento

Os algoritmos de esculpimento são aqueles que utilizam como base para a reconstrução a triangulação de Delaunay e o diagrama de Voronoi. Em seguida, alguma heurística é utilizada para se remover os tetraedros e fazer com que a superfície reconstruída se aproxime da original.

Rodriguez em [Rodriguez et al., 1994] propõe um algoritmo simples baseado no tamanho das faces dos tetraedros medido através do perímetro. Inicialmente é calculada a triangulação de Delaunay do conjunto de pontos da amostra. Em seguida os tetraedros são removidos da triangulação caso alguma de suas faces tenha o perímetro maior que a média dos perímetros de todas as demais faces. A utilização do perímetro é bem eficiente, uma vez que sendo a superfície bem amostrada, existem grandes chances dos triângulos pequenos estarem presentes na superfície do objeto.

Edelsbrunner e Mücke (1994), [Edelsbrunner and Mücke, 1994], [Edelsbrunner et al., 1983] propuseram um algoritmo conhecido como α-*shape*. Neste algoritmo, um simplexo (aresta, triangulo ou tetraedro) é incluído no α-*shape* se existe uma circunsfera vazia, ou seja, que não contenha nenhum ponto da amostra com raio maior ou igual a α. Desta forma, o *0-shape* se iguala ao conjunto de pontos iniciais da mostra, já o ∞-*shape* é equivalente a triangulação de Delaunay. O α-*shape* funciona bem para um conjunto de pontos uniformes, sendo freqüentemente problemático para a reconstrução de superfícies. Isso acontece porque é necessário encontrar o melhor valor do parâmetro α que funcione bem para todas as regiões da superfície. Bajaj, Bernadini e Xu, [Bernardini and Bajaj, 1997] utilizaram o α-*shape* como o primeiro passo de seus algoritmos, onde a seleção do parâmetro α é automática. No entanto, este algoritmo não foi estendido para $\Re^3$.

Techmann e Capps 1998 [Teichmann and Capps, 1998] apresentam generalizações do α-*shape*. Neste trabalho as esferas são substituídas por elipsóides (α-elipsóides). Além disso, o parâmetro α pode variar conforme a densidade local da superfície. Essas duas estratégias têm o objetivo principal de contornar a deficiência do α-*shape*, que apresenta bons resultados apenas para amostragens regularmente espaçadas. A desvantagem da utilização dos algoritmos baseado no α-*shape* é que o parâmetro α deve ser escolhido experimentalmente, e muitas vezes não se consegue determinar uma relação entre a variação de α e a variação da densidade da amostra.

Em [Attali, 1998], Attali apresenta um algoritmo $2D$ com garantias de reconstrução, baseado em malhas normalizadas, que são subconjuntos de arestas da triangulação de Delaunay e de *r-regular shape*. Esses dois conceitos são apresentados em seguida.

Definição 3.1 - *r-regular shape*: Seja $S \in \Re^2$ e F a fronteira de S. S é dito *r-regular shape* se o círculo vazio que passa por três pontos de F tem raio maior que r.

Definição 3.2 - Malha normalizada: Malha normalizada é o grafo definido pelos simplexos cujos duais interceptam a curva F.

Para determinados valores de r tem-se a garantia da reconstrução correta de curvas no plano, no entanto este algoritmo não pôde ser estendido para $\Re^3$.

Attene e Spagnuolo 2000 [Attene and Spagnuolo, 2000], propõem uma heurística baseada na árvore geradora mínima e no hipergrafo de Gabriel estendido para localizar na triangulação de Delaunay aqueles triângulos que têm maior probabilidade de pertencer à superfície original.

Adamy 2002 [Adamy et al., 2000] apresenta um algoritmo de esculpimento baseado no filtro guarda-chuva.

Definição 3.3 - Filtro guarda-chuva: A vizinhança de um vértice v de um complexo simplicial de dimensão n é a união dos interiores de todos os simplexos incidentes a v mais o vértice v. Se a vizinhança é homeomorfa a uma bola aberta n-dimensional então é dita guarda-chuva.

Em $2D$, duas arestas que compartilham um vértice v sem seus outros dois vértices definem um guarda-chuva. Para este algoritmo são apresentadas garantias de reconstrução apenas para o caso bidimensional. Em três dimensões é necessário executar uma etapa de pós-processamento baseada em técnicas de programação linear. O trabalho de Adamy [Adamy et al., 2000] tenta preservar as boas características do α-*shape* [Edelsbrunner and Mücke, 1994] e do $\beta - Skeleton$ (definição 2.22), evitando as características indesejáveis de ambos. A saída desse algoritmo não é tão restritiva quanto o $\beta - Skeleton$ e também não é limitada às variações de densidade da amostragem como o α-*shape*.

Em [Edelsbrunner, 2002], Edelsbrunner propôs um algoritmo baseado na Teoria de Morse definida em seguida.

Definição 3.4 - Teoria de Morse discreta: Uma função $f_M : C_c \rightarrow \Re$ que associa a cada célula de um complexo celular C_c a um valor real é uma função de Morse discreta quando satisfaz as seguintes condições para cada célula $C^{(p)} \in C_c$:

1. $\{T^{(p+1)} \succ C^{(p)} : f(T) \leq f(C)\} \leq 1$;
2. $\{U^{(p-1)} \prec C^{(p)} : f(U) \geq f(C)\} \leq 1$.

Ou seja, para cada célula C, f atribui a uma face de C no máximo um valor maior do que $f(C)$, e a uma célula na qual C pertence ao bordo , no máximo um valor menor que $f(C)$.

Maiores detalhes sobre a Teoria de Morse podem ser encontradas no trabalho de Milnor [Milnor, 1963]. Edelsbrunner utiliza a Teoria de Morse para definir uma função sobre a triangulação de Delaunay. A partir desta função é definido um campo gradiente nos simplexos da triangulação de Delaunay que define quais triângulos serão removidos.

Geiesen e John [Giesen and John, 2002],[Giesen and John, 2003] apresentam trabalhos baseados na teoria de sistemas dinâmicos. Inicialmente é definida uma função de distância dos pontos da amostra. Baseado nesta função é gerado um sistema dinâmico, a partir do qual um conjunto de simplexos é utilizado na reconstrução da superfície. Neste algoritmo também é proposta uma etapa de pós-processamento para melhorar o resultado da reconstrução eliminando alguns simplexos.

A vantagem destes algoritmos baseados na abordagem de esculpimento é que as características estruturais da triangulação de Delaunay e do diagrama de Voronoi dão um bom complemento à falta de informação sobre a estrutura da superfície a ser reconstruída. No entanto, o custo computacional da triangulação de Delaunay pode ser considerado uma das principais desvantagens desses métodos. Além disso, o processo de extração baseado na estrutura da triangulação de Delaunay e no diagrama de Voronoi, na maioria das vezes, não é simples e nem trivial.

Esses algoritmos em muitos casos requerem não somente o cálculo da triangulação de Delaunay, mas também do diagrama de Voronoi, fazendo com que a triangulação de Delaunay seja calculada mais de uma vez, aumentando consideravelmente o tempo de processamento desses algoritmos.

No capítulo 4, serão abordados com maiores detalhes os algoritmos de esculpimento da família *Crust*.

3.3 Métodos baseados em funções implícitas

Para os métodos baseados em funções implícitas, inicialmente uma função de distância com sinal é definida e calculada a partir do conjunto de pontos da amostra. Em seguida o conjunto zero $Z(f_d)$ da função calculada é utilizado para reconstruir a superfície. A função de distância pode ser definida como: f_d: $D \rightarrow \Re$, $D \subset \Re^3$, onde o conjunto zero de $f_d - Z(f_d)$ é a estimativa para a superfície S procurada. Este tipo de abordagem foi proposto por Hoppe et al. [Hoppe et al., 1992] e mais recentemente por Curless e Levoly [Curless and Levoy, 1996].

Hoppe et al. [Hoppe et al., 1992] determinam uma aproximação do plano tangente de cada ponto da amostra utilizando os quadrados dos n vizinhos mais próximos para encontrar a função de distância com sinal. Para obter o conjunto zero da função encontrada, foi utilizada uma variação do Algoritmo *Marching Cubes* no qual cada cubóide é decomposto em tetraedros. No entanto, este algoritmo não é capaz de recuperar detalhes finos de um objeto. Outra desvantagem está na grande quantidade de cálculos geométricos necessários.

O algoritmo de Curless e Levoly [Curless and Levoy, 1996] é mais efetivo. Ele utiliza uma combinação acumulativa de pesos para construir sua função de distância. Este algoritmo é útil para conjuntos de dados muito grande de onde ele deriva ruídos e a informação de tangência do plano. Apesar de sua implementação ser rápida e robusta, é necessária uma etapa de pós-processamento para preencher os buracos da superfície. Além disso, esse algoritmo utiliza não somente a informação da amostra, mas também do *scanner* utilizado para obter a amostra.

Bajaj et al. [Bajaj et al., 1995] propõem um método em que o sinal da função de distância é baseado na classificação dos tetraedros da triangulação de Delaunay em internos ou externos. Essa classificação é feita através do α-*shape* calculado sobre a amostragem. A distância da função é obtida através do diagrama de Voronoi. A partir da função de distância encontrada é feita a aproximação da superfície, sendo necessário ainda uma etapa de suavização. Assim como o algoritmo citado anteriormente, este algoritmo utiliza uma grande quantidade de operações geométricas para obter a aproximação final, o que torna o algoritmo bastante vulnerável a erros de ordem numérica.

Boissonnat e Cazals [Boissonnat and Cazals, 2002] utilizam a combinação do diagrama de Voronoi e funções implícitas. Esta aproximação utiliza vizinhança natural. Dada uma amostra S, a vizinhança natural de um ponto x no espaço é dada pelo conjunto de vértices de Delaunay mais próximos de x. Formalmente, pode-se dizer que eles representam as células de Voronoi que seriam destruídas caso o ponto x fosse acrescentado ao diagrama de Voronoi. A figura 3.1 exemplifica a vizinhança natural em $2D$.

Baseado neste conjunto de vizinhos naturais uma função global f é construída de forma a sempre ser diferenciável, evitando assim pontos irregulares. Este função f é aplicada em todos os vértice de Voronoi. O conjunto de triângulos duais das arestas de Voronoi que possuem sinais diferentes em seus dois vértices definem os triângulos resultantes da aproximação da superfície.

Carr et al. [Carr et al., 2001] utilizam funções de base radial (***RBF-radial basis function***) para aproximar a função de distância. Neste algoritmo, a função base sofre ajustes de acordo com a amostra durante a evolução da **RBF** sendo capaz de tratar falhas nas amostras. No entanto, segundo Carr et al. [Carr et al., 2001], o custo computacional deste algoritmo é alto.

Ohtake et al. [Ohtake et al., 2003] utilizam métodos baseados na partição da unidade para encontrar resultados semelhantes. Este algoritmo é capaz de trabalhar com amostragens da ordem de milhões de pontos utilizando pouca memória, sendo razoavelmente rápido. No entanto, segundo Ohtake et al. [Ohtake et al., 2003], apresenta problemas com ruídos na amostra.

De uma forma geral, os algoritmos baseados na abordagem implícita conseguem manipular grandes conjuntos de dados. Porém, como são baseados em aproximações, podem não ser adequados em aplicações onde a precisão do modelo é importante. Além disso, a maioria dos algoritmos não apresentam garantias teóricas de reconstrução.

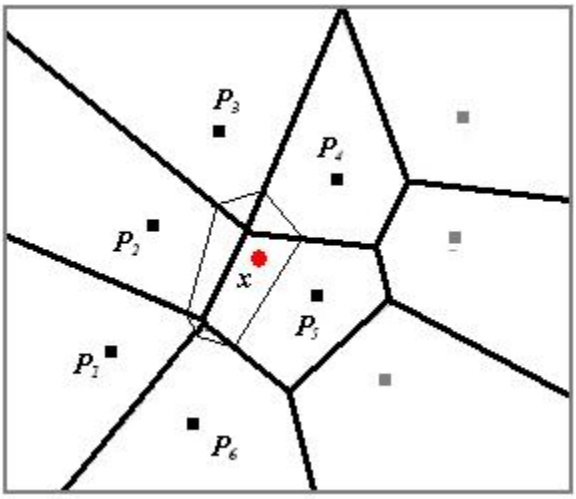

Figura 3.1: vizinhança natural em $2D$: os vizinhos naturais do ponto x são os pontos $p_1, p_2, p_3, p_4, p_5 ep_6$.

3.4 Métodos incrementais

Numa abordagem diferente, os métodos incrementais são inicializados a partir da escolha de um ponto, aresta ou um triângulo, e através de um processo iterativo, novos pontos, arestas ou triângulos são adicionados ao resultado. Esses métodos, também conhecidos como métodos de avanço de fronteira, exigem que a amostra de pontos seja uniforme, uma vez que nenhuma estrutura de dados é dada inicialmente (por exemplo, triangulação de Delaunay nos métodos de esculpimento).

Esta abordagem inclui o algoritmo de Bernadini et al. [Bernardini et al., 1999] conhecido como *Ball-Pivotting Algorithm (BPA)*. Considera-se uma amostra de pontos *S* suficientemente densa e uma bola de raio w que não pode atravessar a variedade sem tocar algum ponto da amostra. O algoritmo é iniciado colocando-se a bola de raio w em contato com três pontos da amostra. Permanecendo em contato com dois dos três pontos, a bola é "pivoteada"até tocar um terceiro ponto. Este processo é feito em torno de cada aresta da fronteira atual e cada três pontos tocados pela bola definem um triangulo na reconstrução. A saída deste algoritmo é um subconjunto do α*-shape* [Edelsbrunner and Mücke, 1994] e segundo os autores, algumas garantias de reconstrução do α*-shape* são mantidas no *BPA*. A principal vantagem deste algoritmo é que sua saída é um subconjunto da triangulação de Delaunay sem a necessidade do cálculo da triangulação, cujo processo é custoso. No entanto, este algoritmo exige uma amostragem com densidade uniforme. Além disso, é necessário selecionar o valor do raio w da bola.

Gopi e Krishman [Gopi et al., 2000] apresentam um algoritmo onde a triangulação de Delaunay é calculada em partes isoladas da superfície. Inicialmente é feito o cálculo da normal para todos os pontos da amostra. Em seguida, é feita a seleção dos pontos candidatos, encontrando os possíveis vizinhos de cada vértice na triangulação. Posteriormente é calculada a triangulação de Delaunay bidimensional para cada conjunto de pontos vizinhos. Este algoritmo possui algumas garantias de reconstrução e é capaz de reconstruir superfícies com bordo e sem bordo.

Huang e Menq [Huang and Menq, 2002] propõem um algoritmo baseado na estimativa da curvatura da superfície. Para definir esta curvatura são utilizados conceitos de Geometria Diferencial. Este algoritmo também possui um bom desempenho, no entanto utiliza estimativas de normais dos pontos e necessita da configuração correta de parâmetros.

Mederos et al. [Mederos et al., 2003] apresentam um trabalho onde inicialmente é feita a redução da nuvem de pontos para em seguida iniciar a reconstrução da superfície. Este técnica é baseada em *clusterização*,

ou seja após dividir o conjunto de pontos em *clusters*, apenas os pontos representativos de cada *cluster* são utilizados na reconstrução. Para este algoritmo não é apresentada nenhuma garantia teórica de reconstrução.

Embora os métodos incrementais tenham em comum a vantagem de não ser necessário o cálculo da triangulação de Delaunay, eles possuem uma desvantagem em comum: a qualidade da reconstrução depende necessariamente da definição de parâmetros. Esses parâmetros variam de acordo com a densidade da amostra e, portanto não podem ser encontrados facilmente. Outra desvantagem está no fato de que esses algoritmos podem deixar pequenos buracos na superfície reconstruída quando a amostra, por exemplo, contiver ruídos. Assim, é necessário efetuar uma rotina de pós-processamento, a fim de garantir a reconstrução de uma superfície fechada.

Recentemente no trabalho de Kuo e Yau [Kuo and Yau, 2005] é apresentado um algoritmo ***DBRG*** (***Delaunay-basead region-growing***) que utiliza tanto técnicas de esculpimento (triangulação de Delaunay) quanto a abordagem incremental. Inicialmente a triangulação de Delaunay é calculada para os pontos da amostra. Em seguida é escolhido um triangulo inicial, e a partir deste, novos triângulos são adicionados apenas nas regiões limite das arestas. A saída do algoritmo pode conter buracos caso a amostra contenha ruídos. Desta forma é necessário fazer um pós-processamento para filtragem desses buracos. A vantagem desse algoritmo em relação aos métodos de esculpimento, é que este calcula apenas a triangulação de Delaunay, não sendo necessário o cálculo do diagrama de Voronoi. Por outro lado, em comparação aos métodos incrementais, o **DBRG** é mais sistemático, não sendo necessária a configuração de parâmetros para o seu correto funcionamento. Segundo os autores, experimentalmente este algoritmo se demonstrou bastante eficiente em relação aos demais algoritmos citados.

3.5 Métodos baseados em modelos deformáveis

Os métodos baseados em modelos deformáveis são métodos de reconstrução que deformam uma superfície inicial até conseguir uma aproximação da superfície que originou os pontos da amostra. Estes métodos são considerados uma técnica de aproximação, uma vez que superfícies intermediárias são geradas pelas deformações que ocorrem durante o processo.

Em geral, esses métodos possuem um bom desempenho computacional, no entanto para se obter resultados satisfatórios é necessário que a superfície inicial esteja próxima da forma original. Por esse motivo, muitas vezes esses métodos são utilizados em etapas de pós-processamento de outros algoritmos.

Na literatura podemos encontrar dois subconjuntos de métodos baseados em modelos deformáveis: o primeiro é baseado em *Equações Diferenciais Parciais (E.D.P.s)* para deformações de modelos iniciais [Zhao et al., 2000]. No entanto, estas técnicas necessitam das discretização das *E.D.P.s* e resolução de grandes sistemas lineares.

O segundo conjunto de métodos é baseado em funções paramétricas [Bardinet et al., 1998]. Dada uma função paramétrica inicial, são aplicadas alterações nos parâmetros iniciais até se obter valores satisfatórios que melhor enquadrem a função à nuvem de pontos. O problema das funções paramétricas é que elas são limitadas quando existe muita variação de topologia e de detalhes da superfície.

Bardinet et al [Bardinet et al., 1998] apresenta um trabalho em superquádricas, cuja a principal vantagem é o número reduzido de parâmetros necessários para representar o objeto. No entanto, esta abordagem não apresenta bons resultados para superfícies mais complexas, uma vez que é necessário conhecimento global sobre a estrutura do objeto. Neste caso é necessário aplicar um pós-processamento denominado deformação de forma livre (*free-form-deformation*) com o objetivo de fazer alguns ajustes na superfície reconstruída.

Neste contexto também podemos citar o algoritmo de Algorri e Schmitt [Algorri and Schmitt, 1996] que é baseado na deformação de uma malha de triângulos inicial utilizando o modelo de molas (*spring-model*). O espaço definido pelo conjunto de pontos iniciais é subdividido em cubos; em seguida, são extraídos apenas os cubos que possuem no mínimo um ponto da amostra. Este conjunto de cubos é considerado

uma aproximação inicial para a superfície. Esses quadriláteros então são subdivididos em triângulos e seus vértices são deformados para que estes se aproximem da superfície original. Assim como o algoritmo de Bardinet et. al. [Bardinet et al., 1998], este método também apresenta dificuldades na reconstrução de superfícies complexas. Além disso, não apresenta garantias teóricas de reconstrução. Esses algoritmos são aplicados em nuvens de pontos adquiridas a partir de imagens médicas, como por exemplo, imagens do coração.

3.6 Considerações finais

Neste capítulo foram apresentados os trabalhos mais recentes sobre reconstrução de superfícies a partir de pontos não-organizados, classificando os diversos algoritmos nas respectivas abordagens: esculpimento, incremental, modelo deformável e funções implícitas. No próximo capítulo serão abordados os algoritmos que pertencem à família *Crust* bem como as garantias teóricas de reconstrução destes algoritmos.

Capítulo 4

Família *Crust*

4.1 Considerações Iniciais

Neste capítulo serão abordados os algoritmos desenvolvidos por Nina Amenta e demais autores para a reconstrução de superfície a partir de nuvens de pontos, bem como resultados teóricos e as garantias de reconstrução para cada algoritmo. Primeiramente será apresentado o algoritmo denominado β-*Skeleton* [Amenta et al., 1998] e o *crust* bidimensional, denominado *raw crust* [Amenta et al., 1998]. Em seguida será mostrada a variação deste algoritmo em $3D$, que com a adição de uma etapa de pós-processamento, passa a se chamar *crust* [Amenta and Bern, 1999a], [Amenta and Bern, 1999b].

Em [Amenta et al., 2001a] e [Amenta et al., 2001b], Nina Amenta introduziu o algoritmo denominado *power crust*. Este algoritmo é mais sofisticado e oferece garantias de reconstrução mais fortes que o *crust*. O *power Crust* tem como saída além da reconstrução da superfície, uma aproximação para o eixo medial denominado *power shape*. Ainda serão abordados os algoritmos *cocone* [Amenta et al., 2002] e o *tight cocone* [Dey and Goswami, 2002a], que é uma etapa de pós-processamento do *cocone*. Para os algoritmos β-*Skeleton*, *raw crust* e *power crust* serão apresentados os principais teoremas e as garantias de reconstrução.

4.2 β-*skeleton*

O β-*Skeleton* introduzido por Kirkpatrick e Radke [Kirkpatric and Radke, 1985] define um grafo como medida de proximidade de vizinhança entre pontos. Assim como o α-*shape* [Edelsbrunner and Mücke, 1994], o β-*Skeleton* de um conjunto de pontos varia de acordo com o parâmetro β, gerando estruturas diferentes para o mesmo conjunto de pontos. Kirkpatrick em seu trabalho não relacionou o β-*Skeleton* à reconstrução de curvas planares a partir de um conjunto de pontos. Somente em 1998, Nina Amenta [Amenta et al., 1998] utilizou o β-*Skeleton* na reconstrução de curvas planares, estabelecendo garantias teóricas de reconstrução. Em [Amenta et al., 1998], Amenta demonstra que sob certas condições de amostragem pode-se garantir a correta reconstrução da curva de forma suave, contínua, fechada e sem auto-interseções. O objetivo final é obter uma aproximação poligonal que conecte exatamente todas as amostras adjacentes ao longo da curva. Na figura 4.1 é mostrado um exemplo de reconstrução planar a partir de um conjunto de pontos.

O β-*Skeleton* anteriormente introduzido no capítulo 2 baseia-se na definição de uma região proibida, considerando $\beta \geq 1$. Segundo esta definição, se uma aresta pertence ao β-*Skeleton*, então há pelo menos um disco que contém apenas os pontos s_1 e s_2 na fronteira e nenhum outro no interior. Logo o β-*Skeleton* de S é um subconjunto da triangulação de Delaunay de S.

Desta forma, podemos descrever o algoritmo β-*Skeleton* para um conjunto de pontos S, com uma constante β da seguinte forma:

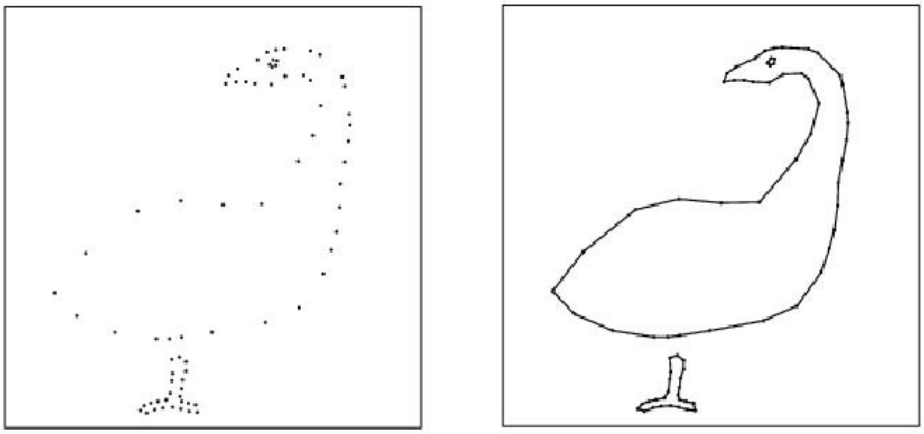

Figura 4.1: Exemplo de uma reconstrução planar a partir de um conjunto de pontos.

ALGORITMO [4.1]: β-Skeleton(s,b)

1: Obter a triangulação de Delaunay de S: $Del(S)$.
2: Para cada aresta **e** pertencente a $Del(S)$ faça
3: Se os circuncírculos dos triângulos adjacentes a **e** têm raios menores que $\frac{\beta}{2} * \|e\|$ então
4: **e** pertencente ao *β-Skeleton*

A triangulação de Delaunay calculada no passo 1 do algoritmo pode ser obtida em $O(n * \log n)$. Como o número de arestas é linear em relação ao número de triângulos, o passo 2 pode ser obtido em $O(n)$. Desta forma, o algoritmo tem complexidade $O(n * logn)$.

Em seguida, serão expostos alguns teoremas e definições relativos ao β-*Skeleton*, extraídos de [Amenta et al., 1998].

4.3 Condições de amostragem para o β-*Skeleton*

Dadas as definições do grafos na seção anterior, é necessário determinar as condições de amostragem que garantem que a curva reconstruída seja próxima à curva original. Para isso, em [Amenta and Bern, 1999b] é utilizado o conceito de *Local Feature Size* (definição 2.10) que permite calcular a qualidade da amostra de pontos para a reconstrução.

A partir do eixo medial (definição 2.9) podemos descrever as condições de amostragem. Em [Amenta and Bern, 1999b] é definido que uma "boa amostra" é aquela cuja densidade é, no mínimo, inversamente proporcional à distância ao eixo medial. Uma amostra de pontos S é uma r-amostra de uma superfície M quando a distância Euclidiana entre qualquer ponto $p \in M$ a seu ponto s mais próximo da amostra é r vezes a distância de p ao ponto mais próximo do eixo medial p_m de M. Ou seja,

$$d(p,s) \leq r * d(p,p_m) \Rightarrow r = \frac{d(p,s)}{d(p,p_m)}$$

Através da figura 4.2 pode-se visualizar esta relação em duas dimensões. Seja p um ponto da curva F, s um ponto da amostra e E o eixo medial da curva F. Considere $p_m \in E$, o ponto mais próximo de p. Para um valor de r fixo, quanto mais distante se estiver do eixo medial, maior poderá ser a distância entre os pontos

da amostra. Por outro lado, quando o eixo medial estiver próximo da superfície, a amostra deverá ter pontos mais próximos entre si, para manter uma boa amostragem da superfície.

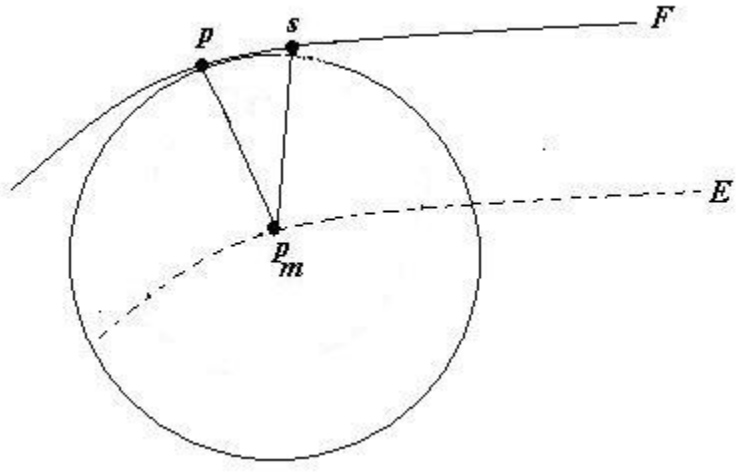

Figura 4.2: Relação entre a distância $d(p, s)$ e $d(p, p_m)$. p é um ponto da curva F, s é um ponto da amostra e p_m é o ponto do eixo medial E mais próximo de p

Os teoremas a seguir serão utilizados nas garantias de reconstrução do β-*Skeleton*. As provas correspondentes a eles são encontradas em [Amenta et al., 1998].

Teorema 1: Qualquer circunferência que contenha no mínimo dois pontos de uma curva suave F no plano: intercepta a curva num segmento de curva, ou contém um ponto do eixo medial, ou ambos.

Na figura 4.3 é mostrada a interpretação geométrica para este teorema. Em linhas contínuas temos a curva F, em tracejado temos o eixo medial. Como podemos observar, a circunferência mostrada na figura intercepta tanto a curva como o eixo medial.

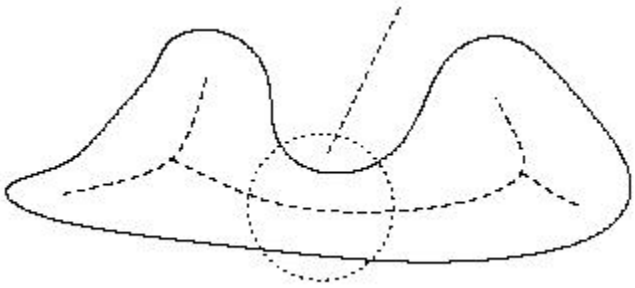

Figura 4.3: Interpretação geométrica do teorema 1. Em linhas contínuas a curva F e em linha tracejadas o eixo medial.

O próximo teorema a ser apresentado é baseado nos circuncírculos dos triângulos da triangulação de Delaunay, também conhecidos como discos de Voronoi. Os vértices das células de Voronoi são os centros desses discos e cada disco de Voronoi contém três pontos da amostragem em sua fronteira e não em seu interior (para pontos em posição geral).

Teorema 2: Seja $S \subset F$, onde F é uma curva suave no plano. Então qualquer disco de Voronoi de S deve conter um ponto do eixo medial de F.

Este teorema não pode ser generalizado para $\Re^n$, onde $n > 2$. Isso acontece porque uma amostra qualquer S de uma superfície suave M, em três dimensões, pode conter pequenos discos de Voronoi na superfície de M muito distantes do eixo medial. Esse é um dos principais motivos que não se permite generalizar diretamente alguns algoritmos $2D$ em $3D$ para os métodos de esculpimento.

Os teoremas 3 e 4 garantem que para uma curva r-amostrada, a triangulação de Delaunay contém o subconjunto de arestas que determinam a reconstrução da curva.

Teorema 3: Seja F uma curva suave no plano r-amostrada com $r \leq 1$. Existe um disco de Voronoi tocando cada par de pontos adjacentes.

Teorema 4: Seja $S \subset F$, S uma r-amostragem de uma curva suave F no plano, com $r < 1$. A triangulação de Delaunay de S contém uma aresta entre cada par de pontos adjacentes.

Em [Amenta and Bern, 1999b] é observado que, para uma curva r-amostrada, se $r \geq 1$, então pode não haver reconstrução única para a curva.

Para valores pequenos de r, é possível mostrar que existe uma única reconstrução em que o β-*skeleton* está enquadrado, conforme explicitado nos próximos teoremas.

Teorema 5 : Uma circunferência tangente a uma curva suave F em um ponto p com raio no máximo $LSF(p)$ não contém nenhum outro ponto de F em seu interior.

Teorema 6 : Para uma curva r-amostrada no plano, com $r < 1$, o ângulo formado no vértice de Voronoi entre dois pontos adjacentes da amostragem é, no mínimo, $\pi - 2\arcsin(r/2)$ (figura 4.4).

A partir da figura 4.4 podemos extrair as relações trigonométricas utilizadas nos próximos teoremas. Seja s um ponto da amostra, p um ponto da amostra por onde passa o disco de Voronoi com centro c, e r a distância de s a p. Sem perda de generalidade, considere as distâncias $d(s,c) = LFS(s)$ e $d(p,c) = LFS(p)$ iguais a 1. Desta forma, podemos verificar as seguintes relações:

1. O ângulo $\sin(\gamma)$ é dado pelo tamanho do segmento (s,x);
2. O ângulo $\sin(\alpha) = \frac{d(s,x)}{d(s,p)}$. Considerede $r = d(s,p)$, então $\frac{r}{2} = \sin(\gamma/2) \Rightarrow r = 2\sin(\gamma/2) \Rightarrow \gamma = 2\arcsin(r/2)$;
3. O ângulo $\beta = \frac{\pi-\gamma}{2} \Rightarrow \beta = \pi/2 - \arcsin(r/2)$;
4. O ângulo α formando entre a tangente da linha L em p e o segmento (s,p) é é dado por $\alpha = \pi/2 - \beta$, então $\alpha = \arcsin(r/2)$. Logo $\alpha = \gamma/2 = \arcsin(r/2)$

De maneira similar pode-se demonstrar o seguinte resultado:

Teorema 7 : Seja F uma curva r-amostrada no plano, com $r < 1$, o ângulo gerado por três pontos adjacentes da amostra, é no mínimo $\pi - 4\arcsin(r/2)$.

Os teoremas a seguir são utilizados para definir os valores adequados de r e β para obtenção da forma original da curva no algoritmo do β-*skeleton*.

Primeiramente iremos definir as regiões proibidas das arestas em termos dos ângulos entre os dois círculos (figura 4.5).

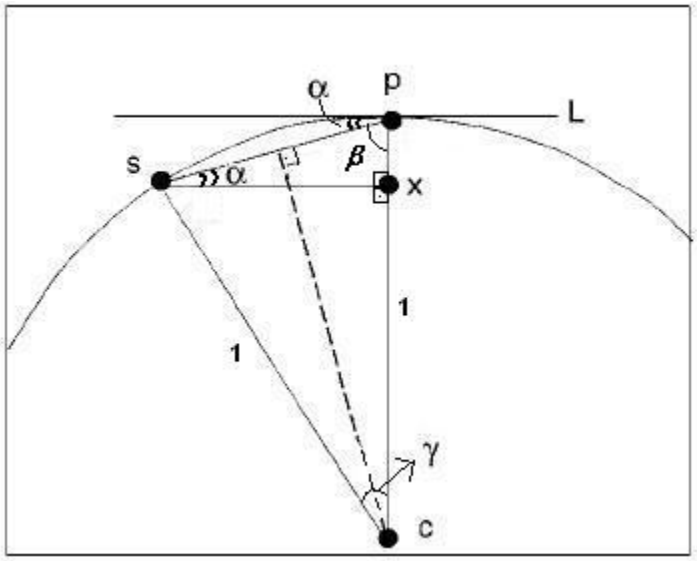

Figura 4.4: Linha L tangente ao círculo em p. $d(c,p) = d(c,s) = 1$ e $d(s,p) = r$. Fonte: [Amenta et al., 1998]

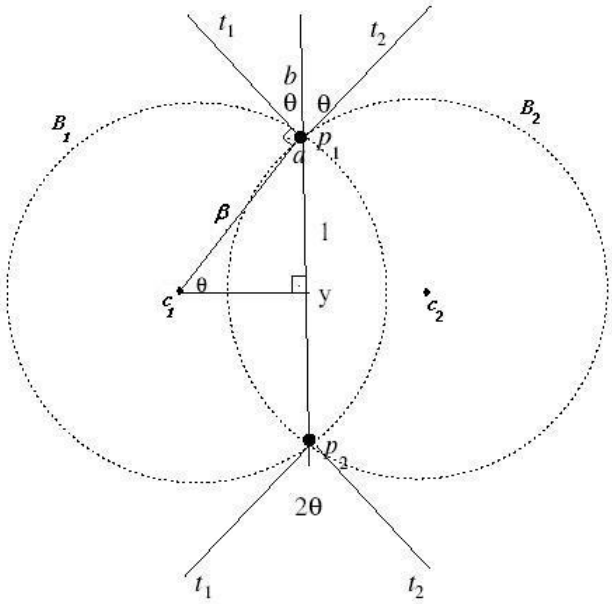

Figura 4.5: Definição da região proibida entre dois círculos em termos de ângulos. Fonte: [Gois, 2004]

Sejam p_1 e $p_2 \in \Re^2$, $\beta \geq 1$, $\theta = \arcsin(1/\beta)$. Seja a região proibida definida pela interseção das duas circunferências B_1 e B_2 de raios $||p2 - p1||\beta/2$ (figura 4.5). As tangentes t_1 e t_2 em p_1 e p_2 definem um ângulo de 2θ em p_1 e p_2.

Esta relação pode ser verificada através da figura 4.5. Dado o triângulo $t = c_1 y p_1$, o ângulo interno a t em p_1 e os ângulos a e b:

1. $a + b + \pi/2 = \pi$;
2. $\theta + \pi/2 + a = \pi$.

Subtraindo as duas equações acima, temos que $b = \theta$.

Em seguida iremos utilizar a figura 4.6 para demonstrar o teorema 8.

Teorema 8 : Sejam p_1, p_2, p_3 pontos sucessivos sobre F. Quando $\theta > 4\arcsin(r/2)$, p_3 não pertencerá a região proibida definida pela aresta p_1p_2.

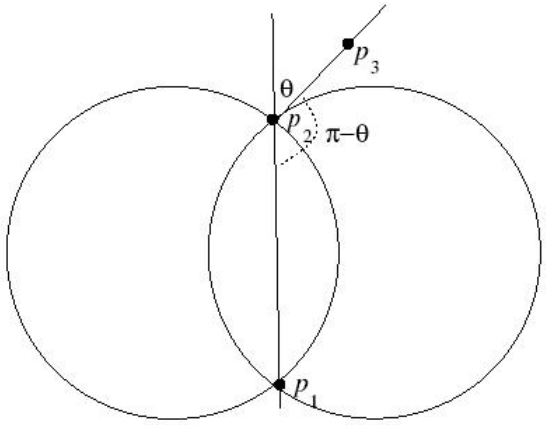

Figura 4.6: Demonstração do teorema 8. Fonte: [Gois, 2004]

Prova: Pelo teorema 7 temos que o ângulo mínimo definido por $p_1\hat{p_2}p_3$ é $\pi - 4\arcsin(r/2)$. Observe na figura 4.6 que se $p_1\hat{p_2}p_3$ for maior que $\pi - \theta$, então p_3 não pertencerá à região proibida definida por p_1 e p_2; logo, p_3 estará fora da região. Desta forma, podemos definir que $p_1\hat{p_2}p_3 \geq \pi - (\pi - 4\arcsin(r/2))$. ■

Os teoremas 9 e 10 serão introduzidos para que possamos usar na prova do teorema 11. Suas respectivas provas encontram-se em [Amenta and Bern, 1999b].

Teorema 9 : A região proibida de uma aresta entre dois pontos adjacentes sobre uma curva F r-amostrada não pode conter um ponto do eixo medial, quando $\theta > \arcsin(2\sin(2\arcsin(r/2)))$.

Teorema 10 : O β-*skeleton* de uma curva suave F r-amostrada não contém arestas entre qualquer par de pontos não adjacentes, quando $\theta < \arccos(2r) - 2\arcsin(r/2)$.

Teorema 11 : Seja F uma curva suave r-amostrada, com $r < 0.297$. O β-*skeleton* de F contém exatamente as arestas entres vértices adjacentes, para $\beta = 1.70$ (figura 4.7).

Prova: Segundo o teorema 10, não existem arestas entre vértices não adjacentes para

$$\theta < \arccos(2r) - 2\arcsin(r/2). \tag{4.1}$$

Sejam p_1 e p_2 dois pontos adjacentes e sejam p_0 e p_3 adjacentes a p_1 e p_2 respectivamente. Pelo teorema 8, nem p_0 e p_3 pertencem às regiões proibidas quando

$$\theta > 4\arcsin(r/2). \tag{4.2}$$

Se existir algum p_i pertencente à região proibida, não sendo p_0 e p_3, isto implicaria que um dos círculos da região proibida intercepta na curva F em no mínimo duas componentes conexas (uma contendo p_1 e outra

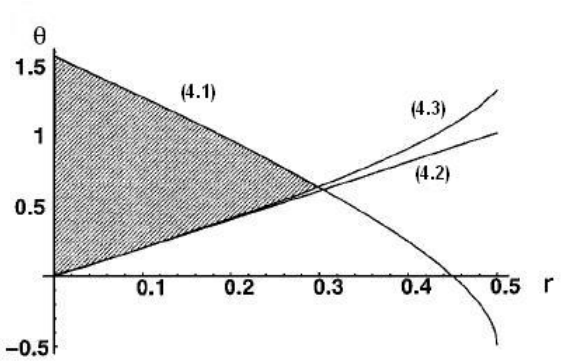

Figura 4.7: A garantia da reconstrução da curva original se dá para valores na região hachurada. Cada uma das curvas é referente a uma equação: 4.1, 4.2 e 4.3.

contendo p_i) e, portanto, pelo teorema 1, contém um ponto do eixo medial. No entanto, pelo teorema 9 isto não ocorre quando:

$$\theta > \arcsin(2 \sin 2(\arcsin(r/2))). \tag{4.3}$$

Fazendo a interseção da região definida pelas inequações, 4.1, 4.2 e 4.3, existe uma escolha adequada para $r < 0.297$. O valor de θ que permite pontos mais esparsos é obtido quando se maximiza r, que equivale a aproximadamente $\theta = 0.637$, equivalente a $1/\beta = \sin(0.637) \Rightarrow \beta = 1.70$. ■

4.4 Crust 2D

O crust 2D pode ser definido da seguida forma: dada uma amostra de pontos S, e sejam V os vértices do diagrama de Voronoi de S. Considere a triangulação de Delaunay de S'= $S \cup V$. Uma aresta da triangulação de S' pertence ao crust de S se os dois vértices da aresta pertencem a S.

Resumidamente, o algoritmo *crust 2D* consta dos seguintes passos:

ALGORITMO [4.2]: *Crust 2D*

1: Compute o diagrama de Voronoi da amostra de pontos S,
   e seja V os vértices desse diagrama,
2: Compute a triangulação de Delaunay de $S \bigcup V$,
3: O resultado do algoritmo *crust* consiste apenas das
   arestas com todos os vértices contidos em S.

O passo (3) do algoritmo é conhecido como filtragem de Voronoi, ou seja, aplica-se um filtro às arestas geradas pela triangulação de Delaunay. O resultado do algoritmo é exemplificado na figura 4.8.

A principal rotina do algoritmo é o cálculo do diagrama de Voronoi. Se a implementação do cálculo deste diagrama for eficiente, então o algoritmo como um todo será rápido.

Em [Amenta and Bern, 1999a] é provado, que dada uma boa amostra, a saída do algoritmo *crust* é topologicamente equivalente à curva desejada. À medida que a densidade da amostra aumenta, a saída converge para a curva original. Além disso Amenta [Amenta and Bern, 1999a] demonstra que, para uma curva r-amostrada com valor de r suficientemente pequeno, todas as arestas desejadas pertencem ao *crust* e as indesejáveis não pertencem.

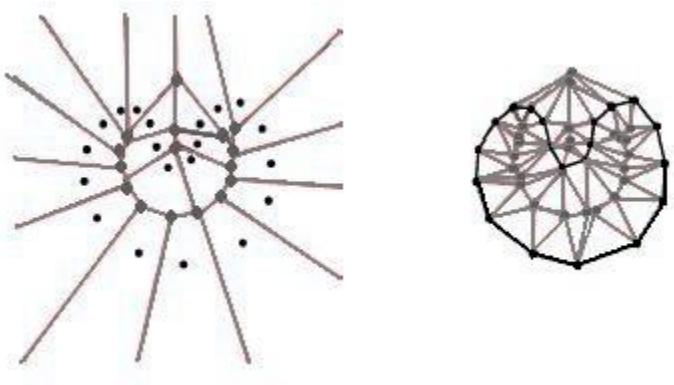

Figura 4.8: À esquerda é mostrado o diagrama de Voronoi para uma amostra de pontos de uma curva. Neste caso o diagrama se aproxima do eixo medial da curva. A direita é mostrada a Triangulação de Delaunay com os pontos da amostra (pontos pretos) mais os vértices do Diagrama de Voronoi (pontos cinzas), resultado do algoritmo *Crust-2D*.

Teorema 12: O *crust* de uma curva suave r-amostrada, com $r < 0.4$, contém uma aresta entre cada par de amostras adjacentes em F.

Prova: Segundo a definição, uma aresta pertence ao *crust* se e somente se existe um círculo vazio tocando os extremos da aresta de outros pontos da amostra e vértices de Voronoi. Esta afirmação é válida para cada disco de Voronoi sobre uma curva r-amostrada, pois um disco de Voronoi toca cada par de vértices adjacentes (teorema 3).

Seja B um disco de Voronoi centrado em p. Por definição, B não contém pontos da amostragem. Na figura 4.9 o ponto v é um vértice de Voronoi que assumimos, por contradição, estar no interior de B. Assuma-se que, sem perda de generalidade, $LFS(p) = 1$. Como v é um vértice de Voronoi, o raio R do círculo de Voronoi B_v de centro em v é no máximo a distância das amostras induzidas por p. Este círculo de Voronoi deve conter um ponto do eixo medial (teorema 2).

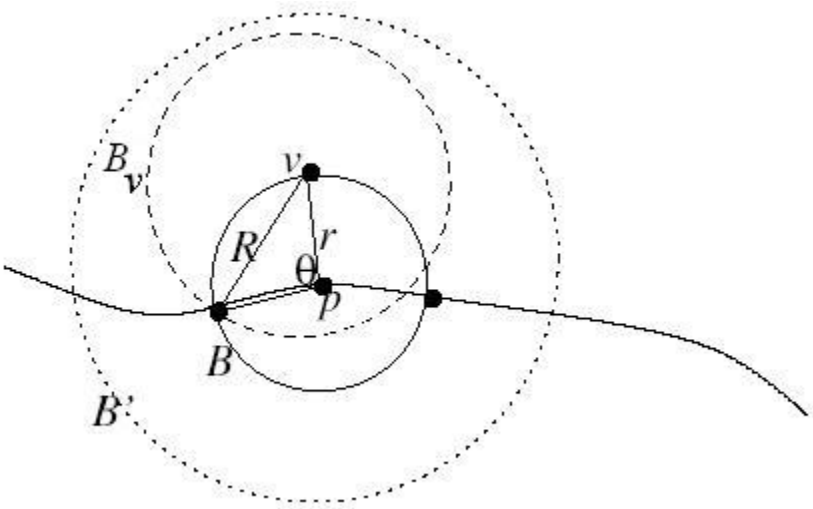

Figura 4.9: Construindo a contradição do teorema 12.

Por outro lado, a circunferência B' de raio $LFS(p) = 1$ de centro p não contém pontos do eixo me-

dial, pela definição de *Local Feature Size*. Tomemos r tal que V esteja contido inteiramente dentro de B', definindo a contradição. Qualquer ponto em B_v está a uma distância no máximo $r + R$ de p, e R é maximizado quando v está na fronteira de B. Neste caso, R é o comprimento da base de um triângulo isósceles cujas outras arestas têm comprimento r. Como a curva é suave em p (teorema 6), o ângulo θ em p é, no máximo, $0.5(\pi + 2\arcsin(r/2))$ e $R/2 \leq r\sin(\theta/2)$. A partir destas duas condições sobre os ângulos temos que:

$$r + 2r\sin(\frac{\pi + 2\arcsin(r/2)}{4}) \leq 1. \tag{4.4}$$

Caso $r \in [0, 1]$, o valor da expressão acima é no máximo $\pi/2$, e tal expressão é crescente neste intervalo. Então, para $r \leq 0.40$ temos a inequação satisfeita. ■

Teorema 13: O *crust* de uma curva r-amostrada não contém nenhuma aresta entre vértices não-adjacentes, para $r < 0.252$.

Prova: A figura 4.10 é utilizada na demonstração deste teorema. O objetivo é mostrar que não existem círculos vazios de amostras e vértices de Voronoi tocando quaisquer dois pontos não adjacentes.

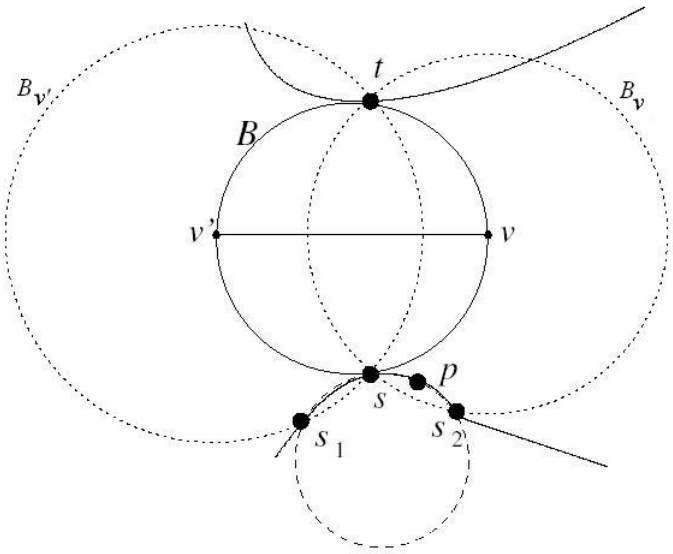

Figura 4.10: Representação geométrica da contradição do teorema 13.

Assume-se, por contradição, que existe um círculo B que contém algum ponto em seu interior. Considere os dois círculos B_v e B'_v tocando s e t e centrados respectivamente em v e v' e B intercepta o bissetor perpendicular de s e t. Afirma-se que se B não contém vértices de Voronoi, então B_v e B'_v são vazios de pontos de P. Caso algum deles fosse não vazio, poderia conter um ponto $s' \in P$, determinando um círculo mínimo vazio de todos outros pontos tocando s, t e s' e, portanto induzindo a ter um vértice de Voronoi dentro de B. Considere a figura 4.11 como sendo a parte da figura 4.10 na região do ponto s com maiores detalhes. O ângulo θ entre os círculos tangentes aos círculos B_v e B'_v em s é igual a $\pi/2$. Isso se deve ao fato de que o semi-círculo inferior de B, contendo s, ser um conjunto dos pontos que formam um ângulo raso com B_v e B'_v. Além disso, as tangentes são perpendiculares a vs e $v's$. Assim o ângulo $s_1\hat{s}s_2$ é, no mínimo, $\pi - 4\arcsin(r/2)$ (teorema 7). Sem perda de generalidade, seja B_v o círculo tal que o ângulo θ

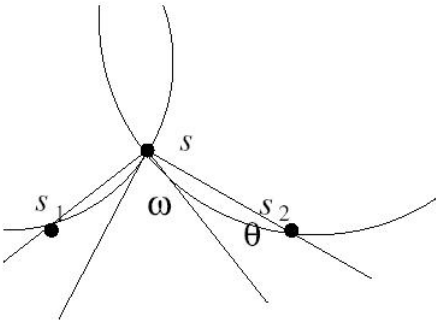

Figura 4.11: Nota-se que o ângulo θ é maior que seu correspondente ângulo do outro lado.

entre a tangente de B_v e s e a corda (s, s_1) ou (s, s_2) é maior que o ângulo correspondente do outro lado. Então:

$$\theta \geq 1/2(\pi - 4\arcsin(r/2) - p/2) = \pi/4 - 2\arcsin(r/2). \tag{4.5}$$

Assumindo que o raio de B_v é um, sem perda de generalidade, o limite de θ, então

$$||s - s_2|| \geq 2\sin(pi/4 - 2\arcsin(r/2)). \tag{4.6}$$

Conforme o teorema 3, existe um disco de Voronoi p entre s e s_2. Substituindo a equação 4.6 na definição de *Local Feature Size* temos:

$$\sin(\pi/4 - 2\arcsin(r/2)) \leq r.LFS(p). \tag{4.7}$$

Assim tem-se um limite superior para $LSF(p)$, o que estabelece a contradição. Os pontos s e t pertencem a duas diferentes componentes conexas da intersecção $B \cap B_v$, portanto B_v contém um ponto do eixo medial (teorema 1). O ponto p está no interior de B_v, com raio um, então $LFS(p) \leq 2$. Portanto:

$$\sin(\pi/4 - 2\arcsin(r/2)) \leq 2r. \tag{4.8}$$

A equação 4.8 é crescente para $r \in [0, 1]$. Escolhendo $0 \leq r \leq 0.252$ temos a desigualdade violada, implicando numa contradição. ■

Desta forma, podemos concluir que o β-*Skeleton* possui uma garantia melhor de reconstrução, uma vez que suas garantias são feitas para $r < 0.297$, enquanto que para o *crust* $r < 0.252$. No entanto [Gois, 2004] observou que o *crust* tende a adicionar arestas que podem ser úteis para a reconstrução. Outro ponto a ser destacado é que não se conhece trabalhos que descrevam a tentiva de implementação do β-*Skeleton* para $\Re^3$, já o *crust* foi extendido para $\Re^3$, como veremos na próxima seção.

4.5 Crust 3D

A versão $3D$ do algoritmo *crust* foi apresentada em dois artigos. No primeiro [Amenta and Bern, 1999b] são apresentados os resultados teóricos que garantem a reconstrução e no segundo [Amenta and Bern, 1999a] é discutida a complexidade do algoritmo, sua eficiência e detalhes de implementação. Este algoritmo possui garantias fracas de reconstrução da superfície original, sendo necessária a utilização de amostragens muito

densas. Desta forma, é adicionada uma etapa de pós-processamento para corrigir os problemas. Com esta etapa adicional o algoritmo inicialmente chamado de *raw crust*, passa a se chamar *crust*.

Para o algoritmo $3D$ o eixo medial não pode ser calculado apenas fazendo-se o cálculo do diagrama de Voronoi, pois alguns vértices do diagrama podem aparecer arbitrariamente sobre a superfície. Isso acontece quando tetraedros *slivers* são gerados pela triangulação. Um tetraedro *sliver* é caracterizado por possuir seus vértices próximos ao equador de sua circunferência, conforme exemplificado na figura 4.12. Observe que o ortocentro deste tetraedro, respectivo vértice do diagrama de Voronoi, estará praticamente no mesmo plano definido pelos seus vértices, ou seja, o ortocentro estará sobre a superfície, e portanto não deve ser considerado no eixo medial.

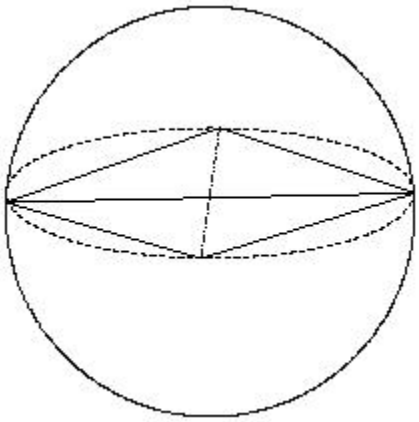

Figura 4.12: Exemplo de um tetraedro *sliver*.

Segundo [Amenta et al., 2001a], o cálculo do eixo medial em $3D$ é computacionalmente caro. Assim o que se tenta fazer é encontrar um algoritmo mais simples e rápido que possibilite uma boa aproximação do eixo medial.

Em [Amenta99] é citada uma maneira de se estimar o eixo medial sem prejudicar a eficiência do algoritmo. A idéia básica consiste em computar dois pólos: um pólo positivo e outro negativo. Dada uma célula do diagrama de Voronoi, o pólo positivo de um sítio pode ser obtido verificando-se nesta célula qual o vértice de Voronoi mais distante em relação ao sítio em questão. Um vetor denominado $sp+$ é formado pelos pontos do sítio e do vértice mais distante. O pólo negativo $p-$ é obtido calculando-se o vértice mais distante em relação ao vetor $sp+$ e que faz um ângulo maior que 90° com este vetor.

Assim pode-se definir o *crust* $3D$ da seguinte forma: seja P o conjunto de pontos em $\Re^3$ e V_p o conjunto de pólos - respectivos vértices de Voronoi. O *crust* de P são os triângulos da triangulação de Delaunay de $P \cup V_p$, onde todos os vértices pertencem a P. O algoritmo resumido é mostrado a seguir:

ALGORITMO [4.3]: *Crust 3D*

1: Compute o diagrama de Voronoi a partir da amostra S,
2: Para cada ponto de S compute os pólos:
   positivo $p+$ e negativo $p-$,
3: Seja P o conjunto de pontos $p+$ e $p-$.
   Calcule a triangulação de Delaunay de $S \cup P$.
4: Por fim deixe apenas os triângulos formados pelos
   pontos contidos na amostra S.

Observe que no passo (3) do algoritmo, a triangulação de Delaunay é feita apenas com os pontos da amostra inicial mais os pontos do conjunto P, que neste caso é formado pelos pólos positivo e negativo. É neste passo que o algoritmo $3D$ difere do $2D$, onde antes todos os vértices do diagrama de Voronoi eram utilizados no cálculo da triangulação. Esta melhoria tem o intuito de evitar o cálculo da triangulação de Delaunay com vértices do diagrama de Voronoi que estão próximos à superfície, ou seja, distantes do eixo medial, tornando a reconstrução do objeto mais eficiente. O algoritmo para o cálculo dos pólos é mostrado em seguida:

ALGORITMO [4.4]: *Cálculo do pólos*

1: Considere o diagrama de Voronoi do conjunto de pontos $P = Vor(p)$
2: Para todo $p \in P$ faça
3: Se p não pertence ao fecho convexo de P então
4: $p+$ é o vértice de Vor(p) mais distante de p.
5: fim do se
6: Se p pertence ao fecho convexo de P então
7: $p+$ é um ponto no infinito fora do fecho convexo e sua direção $pp+$ será dada pelo valor médio da direção normal das faces finitas da célula de Voronoi de p.
8: fim do se
9: $p-$ é dado pelo vértice de Vp mais distante de p, tal que o ângulo formado entre os vetores $p\vec{p}+$ e $p\vec{p}-$ seja maior que $\pi/2$.
10: fim do para

Assim como no algoritmo *Power Crust* descrito posteriormente, na implementação deste algoritmo foi utilizada a estratégia do *bounding box*. Essa estratégia permite obter uma estimativa para os vértices das células de Voronoi infinitas. Desta forma, são adicionados oito vértices na primeira triangulação que formam uma "caixa" que contém todos os pontos da amostra em seu interior. Conforme aumenta-se o tamanho do *bounding Box* obtém-se estimativas mais precisas dos pólos. Assim, tanto os pontos pertencentes ao fecho convexo quanto aqueles que não pertencem, são tratados da mesma maneira.

Na figura 4.13 é mostrado um exemplo do cálculo dos pólos para um ponto da amostra p. Pode-se observar que as células de Voronoi, geralmente são finas e compridas, podendo ser utilizadas como uma aproximação para os vetores normais da superfície, figura 4.13.

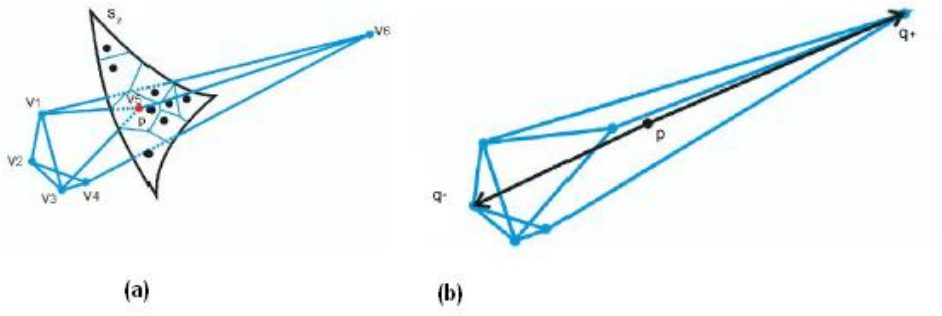

Figura 4.13: (a)Exemplo de uma célula Voronoi - fina e comprida. Os vértice V_1 - V_6 da célula estão próximos do eixo medial. (b) Polos P^+ e P^- calculados a partir de P

A seguir serão apresentados os lemas utilizados na prova do teorema de garantia de reconstrução do *raw crust*:

Lema 1: Dados dois pontos p e q pertencentes à superfície M, $|LFS(p) - LFS(q)| \leq d(p,q)$.

Prova: Primeiramente será provado que $LFS(p) \geq LFS(q) - d(p,q)$. Se $LFS(q) <= d(p,q)$ então não há nada a ser provado, desde que $LFS(p) \geq 0$. No entanto, se $LFS(q) > d(p,q)$, então a esfera com centro q e raio $LFS(q)$ contém p. Esta esfera também contém a esfera de centro p e raio $LFS(q) - d(p,q)$, conforme mostrado na figura 4.14.

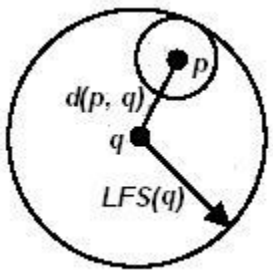

Figura 4.14: Esfera $(q, LSF(q))$ contém a esfera $(p, LFS(q) - d(p,q))$.

Uma vez que a esfera de centro p está contida na esfera de ponto q, esta última não contém nenhum ponto do eixo medial, portanto $LFS(p) \geq LFS(q) - d(p,q)$. Analogamente também pode-se verificar que $LFS(q) \geq LFS(p) - d(p,q)$. ■

Lema 2: Seja s um ponto de uma r-amostragem S:

1. Existe um vértice de $Vor(s)$ em um dos lados da superfície M com distância no mínimo $LFS(s)$ de s.
2. A interseção de $Vor(s)$ e M está contida em uma bola com raio $\frac{r}{1-r}LFS(s)$ em torno de s.

Prova: A primeira parte do teorema se refere ao primeiro pólo calculado para amosta s. Este pólo é o vértice da $Vor(s)$ mais distante da amostra s. Essa distância é no mínimo a distância de s em relação ao ponto mais próximo do eixo medial, ou seja, $LFS(s)$. Para a segunda parte, seja $p \in Vor(s) \cap F$. Como s é o ponto da amostra mais próximo de p, então $d(p,s) \leq r(LFS(s) + d(p,s))$, segundo o lema 1. Desta forma, $d(p,s) \leq \frac{r}{1-r}LFS(s)$. ■

Em seguida será introduzida a definição de reflexão e eixo de um ponto. Esses conceitos são utilizados nos próximos teoremas.

Considere a figua 4.15, onde s é um ponto da amostra que pertence a curva F. Seja v um dos pólos de s e B_v a bola polar com raio em v. Em [Amenta and Bern, 1999b], Amenta define uma região proibida no interior da bola polar B_v, a qual não pode ser penetrada por nenhum triângulo pertencente ao *crust*.

Definição 4.2 - Reflexão: Seja t um vértice que pertence a um triângulo resultante do crust. A reflexão do vértice t através da bola polar B_v é o ponto t' ao longo da semi-reta vt que intercepta a bola B_v e divide o segmento de reta tt' exatamente ao meio.

Definição 4.3 - Eixo: Seja tt' um segmento de reta que intercepta B_v. O eixo de um ponto da amostra s é dado por todos os pontos que pertencem a B_v, cuja a reflexão está sobre B_v ou em seu exterior.

Na figura 4.15 o eixo do ponto da amostra s é mostrado em hachurado.

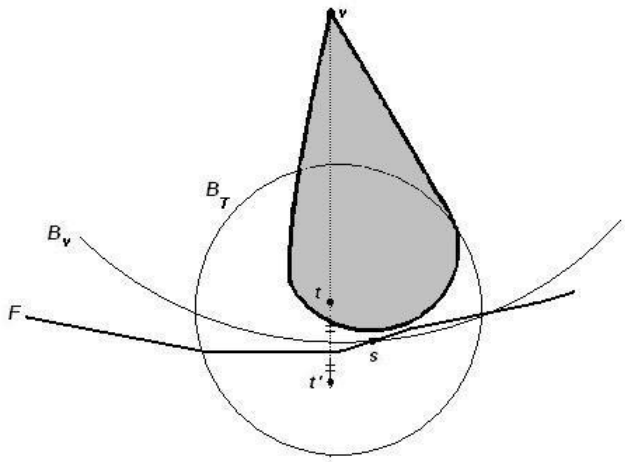

Figura 4.15: Eixo definido pelo ponto da amostra s.

Lema 3: Considere a distância $\delta \leq 0.06$ entre as duas bolas B e B_v. (figura 4.16). Seja t um vértice de um triângulo resultante do *crust* fora da bola B e fora do eixo induzido por B em B_v. Seja p o ponto de B mais próximo de t. Se $|\angle omp|$ é menor que 0.20 radianos, então $d(t,p) \leq \delta + |\angle omp|$.

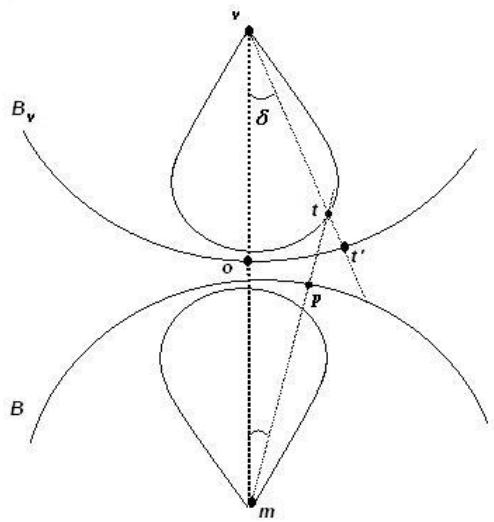

Figura 4.16: Caracterização do lema3.

A prova deste lema pode ser encontrada em [Amenta and Bern, 1999b]. Este será utilizado na prova do teorema relativo às garantias de reconstrução do *crust 3D*.

Teorema 14: Seja P um conjunto de pontos de uma superfície suave r-amostrada S. Para $r < 0.06$ o *raw crust* de P contém um conjunto de triângulos topologicamente equivalente a S.

Prova: Seja B_v a bola polar de s, B_m^- a bola polar de raio $LFS(s)$ tangente a s no lado oposto de F, e B a bola concêntrica à B_m^- com raio reduzido para $rLFS(s)$, conforme mostrado na figura 4.17. Seja o e o'

os pontos de $B_m^- \cap B_v$ mais próximos dos centros de B_m^- e B_v, respectivamente. Se a superfície F pudesse passar pelo ponto o', então s seria necessariamente o ponto da amostra mais próximo de o', desde que B_m^- e B_v fossem ambas vazias. Assim, de acordo com o lema 2, $d(s, o') \leq rLFS(s)/(1-r)$. Considerando que o raio de B_v é no mínimo o raio de B_m^-, então $d(s, o) \leq d(s, o')$.

Seja t é um ponto que pertence a um triângulo do *crust*, e p e p' os dois pontos mais próximos de t em B e B_m^-, respectivamente. Seja q o ponto de F na linha pt mais próximo de t, então $d(t, q) \leq d(p, t)$.

De acordo com o mesmo argumento utilizado para o', $d(s, p') \leq rLFS(s)/(1-r)$, e de acordo com a inequação triangular, $d(o, p') \leq 2rLFS(s)/(1-r)$. Sendo m o ponto central da bola B_m^-, então $d(o, m) \leq LFS(s)$. Assim o ângulo $\angle omp' \leq 2\arcsin(r/(1-r))$, o qual tem valor menor que .20 radianos para valores de $r \leq .06$, o que satisfaz as condições do teorema 3. ■

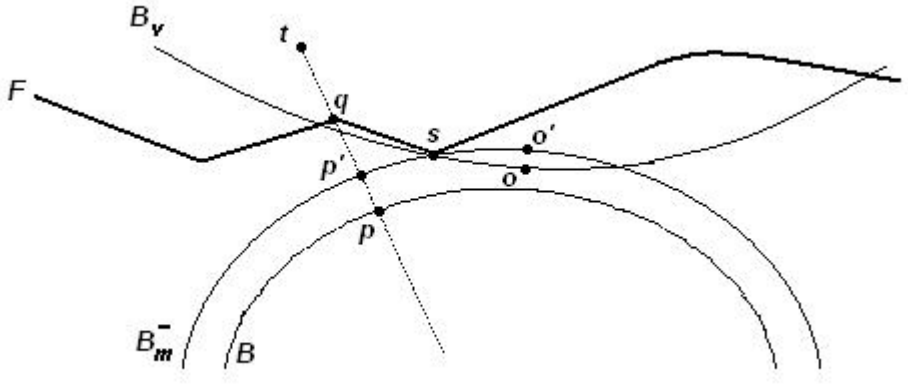

Figura 4.17: Demonstração do teorema 14. Ponto t de um triângulo de *crust* deve estar mais próximo do ponto q da superfície.

Como pode-se notar, as garantias do *raw crust* são fracas ($r < 0.06$), o que implica na necessidade de uma amostragem de pontos extremamente densa para a correta reconstrução da superfície original. Em [Amenta and Bern, 1999a] são propostas duas etapas de pós-processamento para tornar o processo mais eficiente:

***Filtering by Normal*:** Remova todos os triângulos em que o ângulo definido entre sua normal e o vetor formado pelo pólo mais distante e um vértice do triângulo for muito grande (próximo de $\pi/2$).

***Trimming*:** Primeiramente os triângulos e seus pólos são orientados consistentemente, isto é, todos os triângulos possuirão orientação no sentido anti-horário e os pólos que pertencem ao exterior da superfície serão marcados como externos e os que pertencem ao interior como internos. O resultado do algoritmo deverá retornar apenas aqueles triângulos classificados como externos à superfície.

4.6 Power Crust 3D

Em [Amenta et al., 2001a], [Amenta et al., 2001b] é proposta uma nova aproximação para o eixo medial, onde a eficiência do algoritmo não é afetada pela qualidade da amostra de pontos, ou seja, o algoritmo é robusto. Este algoritmo é conhecido como *power crust*. A principal idéia deste algoritmo é produzir a aproximação do eixo medial utilizando a triangulação de Delaunay com peso e o power diagrama para a obtenção da aproximação da superficie.

Em [Amenta et al., 2001b] são apresentados os fundamentos teóricos, resultados de garantias e um algoritmo teórico para rotulação de bolas polares. Em [Amenta et al., 2001a] são apresentadas extensões do

algoritmo para obter melhores resultados para objetos pontiagudos, amostras com pontos esparsos, buracos e ruídos. Além disso é apresentado um algoritmo prático para a nomeação de bolas polares que será detalhado no próximo capítulo.

O algoritmo $3D$ não apresenta diferenças em relação ao $2D$, ou seja, a generalização do algoritmo $2D$ é feita diretamente para $3D$. Na prática existem algumas diferenças de implementação, uma vez que na versão tridimensional foram encontrados alguns problemas nos cálculos de duais e problemas de ordem numérica.

Assim como no algoritmo *crust*, apresentado na seção anterior, o *power crust* também calcula os pólos para cada ponto da amostra. Esses pólos são então utilizados para definir o conceito de bolas polares mostrado em seguida.

Definição 4.4 - Bolas polares: Sejam os pólos p_1 e p_2 de uma amostra $s \in S$ os dois vértices de $Vor(s)$ mais distantes de s, cada um de um lado da superfície. As esferas B_{p_1,ρ_1} e B_{p_2,ρ_2} que passam respectivamente pelos vértices de Voronoi p_1 e p_2 são denominadas bolas polares, com raio $\rho_i = d(p_i, s)$.

A figura 4.18 ilustra as duas bolas polares B_{p_1,ρ_1} e B_{p_2,ρ_2} originadas pelo ponto da amostra s. Como mostrado na seção anterior, o ângulo $\beta \geq \pi/2$.

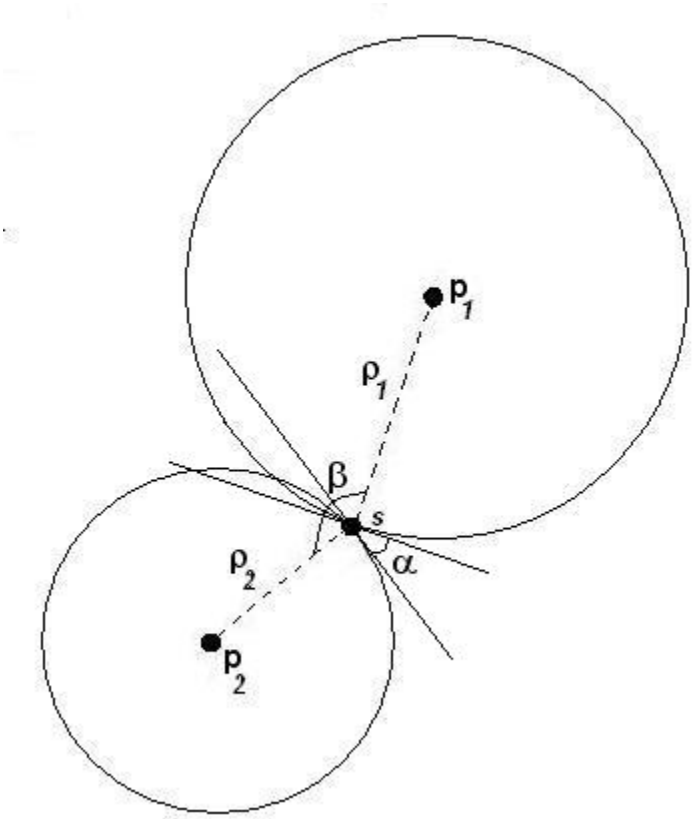

Figura 4.18: Bolas polares B_{p_1,ρ_1} e B_{p_2,ρ_2} originadas pelo ponto da amostra s. O ângulo α é dado pela interseção entre as duas bolas.

Intuitivamente pode-se perceber que dado um conjunto de pólos ρ gerados a partir da amostra de uma superfície M, esta divide o conjunto de pólos em dois: os que estão no interior da superfície: ρ_I; e os que que estão no exterior da superfície: ρ_O. De forma semelhante, tem-se os conjuntos de bolas polares B_I e B_O. Na figura 4.19 é mostrado o conjunto de bolas polares em duas dimensões para uma curva F.

Seja $U_I = \bigcup B_I$, a união das bolas polares internas e $U_O = \bigcup B_O$, a união das bolas externas. Considerando o *power* diagrama $Pow(B_I \cup B_O)$ (definição 2.15), pode-se definir o conceito de *Power crust*.

Definição 4.5 - *Power Crust*: O *power crust* de S é o conjunto de faces poligonais de $Pow(B_I \cup B_O)$ que separam as células pertencentes às bolas polares internas das células que pertencem às bolas polares externas.

Na figura 4.19 é mostrada uma representação da curva F aproximada pela interseção da bolas polares internas e externas.

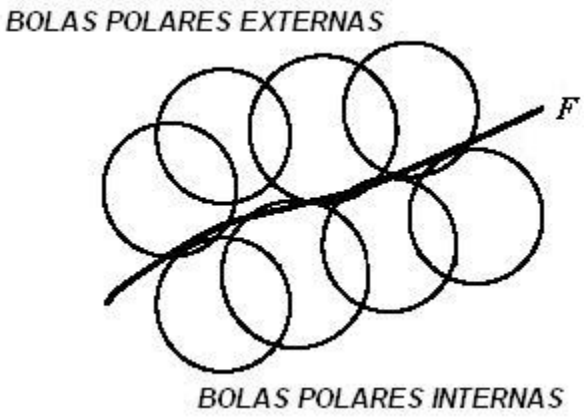

Figura 4.19: Interseção das bolas polares internas e externas gerando a aproximação da superfície M.

Definição 4.6 - *Power Shape*: O *power shape* é um complexo simplicial obtido através da triangulação de Delaunay com peso das bolas polares classificadas como internas à superfície M, constituindo uma aproximação para o eixo medial. O *power shape* é o dual do *power crust*.

Enquanto o eixo medial em três dimensões é formado por uma superfície, o *power shape* é constituído por um conjunto de tetraedros, em geral muito planos.

A figura 4.20 exemplifica o *power crust* bidimensional. Em (a) temos a curva original, seu eixo medial e uma circunferência que toca a curva em dois pontos e possui centro num ponto do eixo medial. Em (b) é apresentado o diagrama de Voronoi e em (c) as circunferências que representam as bolas polares. Em (d) o power diagrama do conjunto de bolas polares e em (e) respectivamente o *power crust* e a aproximação do eixo medial denominada *power shape*.

Em seguida serão apresentados os principais teoremas que compõem as garantias de reconstrução para o *power crust*. Assim como no algoritmo *crust*, esses teoremas são baseados no conceito de *local feature size - LFS*, que define a qualidade da amostra. Segundo Amenta, as amostragens utilizadas nas demonstrações das garantias são sempre r-amostragens para $r \leq 0.1$ sendo as variedades sem bordo.

Teorema 15: Dada uma superfície r-amostrada para r suficientemente pequeno, o ângulo α entre uma bola polar interna e uma externa é no máximo $r' = r/(1 - r)$.

Pode-se notar que a interseção entre duas bolas polares internas e externas, caso exista, é muito pequena. Na figura 4.18 pode-se verificar que o ângulo α formado pela interseção entre as duas bolas polares é muito pequeno, ou seja, a interseção entre as bolas é rasa. Este ângulo será utilizado posteriormente no algoritmo

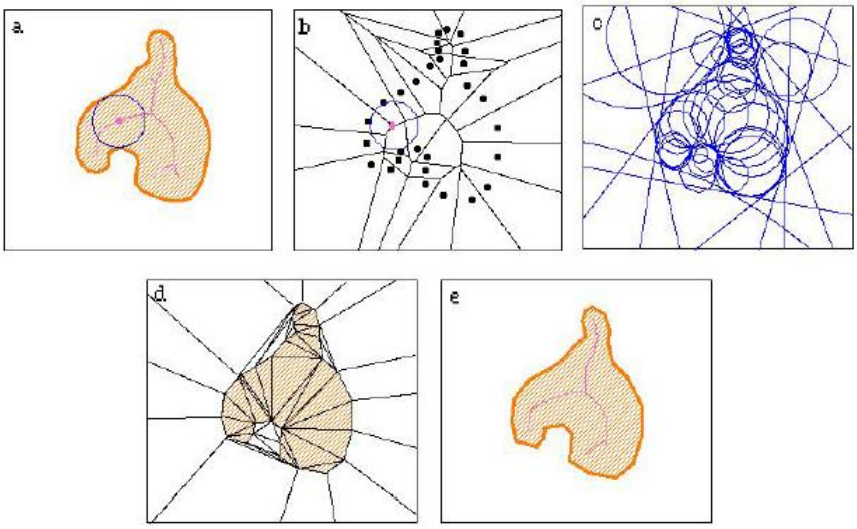

Figura 4.20: Exemplo do *power crust* bidimensional. (a) Curva original, seu eixo medial e uma circunferência que toca a curva em dois pontos e possui centro num ponto do eixo medial. (b) Diagrama de Voronoi. (c) Circunferências que representam as bolas polares. (d) *Power diagrama* do conjunto de bolas polares. (e) Respectivamente o *power crust* e a aproximação do eixo medial denominado *power shape*. Fonte: [Amenta et al., 2001a]

de classificação das bolas polares entre internas e externas. Uma outra idéia por trás deste teorema é que as bolas polares internas estão quase inteiramente contidas no interior do objeto. Em seu trabalho, Amenta mostra que a união das bolas polares internas é uma aproximação para o objeto. A figura 4.21 apresenta alguns exemplos desta aproximação.

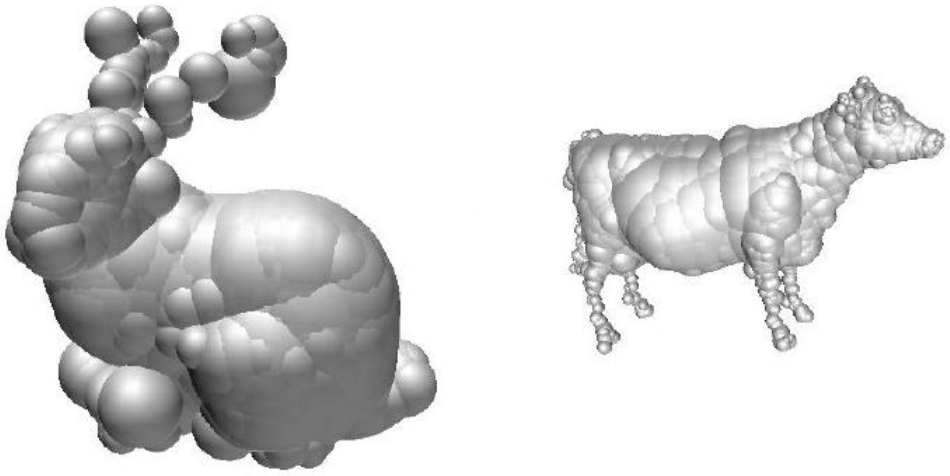

Figura 4.21: Exemplos da aproximação de objetos através da união de bolas polares internas. Fonte: [Gois, 2004]

Os teoremas 16 e 17, demonstrados por Amenta garantem que dada uma r-amostragem com r pequeno, o *power crust* é próximo e homeomorfo à superfície original.

Observação: Amenta assume que as amostragens utilizadas nas demonstrações das garantias são sempre

r-amostragens, com $r \leq 0.1$ e que as variedades são sem bordo.

Teorema 17: proximidade Qualquer ponto u de uma face do *power crust* está a uma distância de no máximo $O(r)LFS(x)$ de algum x da superfície.

Teorema 18: Homeomorfismo Existe uma deformação contínua que transforma o power crust na superfície.

O algoritmo *power crust* resumido é mostrado a seguir:

ALGORITMO [4.5]: *Power Crust*

1. Compute o diagrama de Voronoi da amostra S.
2. Para cada ponto de S compute os pólos:
   positivo $p+$ e negativo $p-$,
3. Compute o *power diagrama* das bolas polares.
5. Classifique as bolas polares como internas ou externas à superfície.
6. Retorne as faces do *power* diagrama que separam uma bola interna de uma externa (*power crust*).
7. Calcule o dual do *power diagrama* (triangulação regular) apenas para as faces formadas pelas bolas polares internas - *power shape*.

No primeiro passo do algoritmo o diagrama de Voronoi é calculado para todos os pontos da amostra S. Já o *power* diagrama calculado no passo 3, leva em conta apenas os pólos e não os pontos da amostra. Observe que nesse passo os pólos passam a ser chamados de bolas polares, onde seus respectivos raios são calculados pela distância do ponto da amostra s_i ao pólo p_i (definição 4.2). O peso de cada bola polar no diagrama de Voronoi com peso é dado por esse raio ao quadrado.

Após a classificação das bolas polares, o *power crust* é formado pelas faces do *power* diagrama que separam as bolas internas das externas. Para calcular o *power shape*, último passo do algoritmo, ainda se faz necessário o cálculo de um novo diagrama de Voronoi com peso apenas para as bolas polares internas. O dual deste novo diagrama (triangulação de Delaunay com peso) irá retornar um conjunto de simplexos que formam o *power shape*.

O próximo algoritmo resume o procedimento teórico de nomeação dos pólos:

ALGORITMO [4.6]: *Classificação dos pólos (teórico)*

1: Selecione um ponto do fecho convexo $CONV(s)$ da amostra S.
2: Marque sua bola polar com vértice no infinito como externa e a oposta como interna e insira ambas numa fila;
3: enquanto a fila não estiver vazia faça
4: retire uma bola polar da fila e examine cada vizinho q não marcado do *power* diagrama
5: se a bola polar de q intercepta a bola polar p num ângulo maior que $\pi/4$ então
6: atribua a q o mesmo rótulo de p e insira-o na fila;
7: para cada ponto $s \in P$ tal que q é uma bola polar de s faça
8: se a bola polar q' oposta a q em s não está marcada, atribua a q' o rótulo oposto de q e insira q' na fila;
9: fim do para
10: fim do se
11:fim do enquanto

Observe que a classificação da bolas polares como interna ou externa é baseada no ângulo formado pela interseção entre suas bolas bolares vizinhas. No passo 5 do algoritmo apenas interseções profundas, (ângulo maior que $\pi/4$) são levadas em consideração, uma vez que quanto mais próximas umas das outras, maior será a chance de duas bolas vizinhas possuirem o mesmo rótulo.

Por outro lado, no passo 8, consideramos apenas a interserção entre bolas polares opostas. Como mostrado no teorema 15, a interseção entre duas bolas polares opostas, quando existir é pequena, e neste caso o algoritmo atribui rótulos opostos entre elas. Ou seja, quanto mais distante uma bola estiver se sua oposta, maior será a chance de ambas possuirem rótulo opostos.

Em [Amenta et al., 2001b] Amenta garante que todas as bolas polares recebem uma classificação através deste algoritmo e além disso, todas as bolas são corretamente rotuladas, isto é, bolas polares internas são classificadas como internas e bolas polares externas como externas.

No próximo capítulo será apresentada um versão prática deste algoritmo, uma vez que esta classificação das bolas polares consiste na etapa principal do algoritmo *power crust*.

4.7 Cocone

Como mostrado no item 4.5 o *crust*, apesar de suas garantias teóricas, necessita executar duas triangulações de Delaunay: a primeira com o conjunto de pontos iniciais da amostra, e a segunda com o conjunto de pontos da mostra mais os pólos calculados.

O algoritmo *Cocone* foi desenvolvido por Amenta et, al. [Amenta et al., 2002] a partir do algoritmo *crust* com o objetivo de evitar o cálculo das duas triangulações. Um *cocone* C_p de um ponto p de uma amostragem é definido como um duplo cone, sendo p seu ápice e o ângulo entre a normal do ponto p e a lateral do cone igual $3\pi/8$ (figura 4.22).

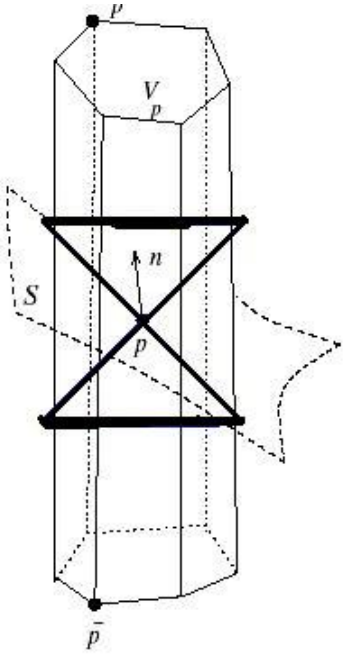

Figura 4.22: *Cocone* (segmentos destacados em negrito) de ápice em p. n é o vetor normal, p^+ e p^- são os pólos de p. Fonte [Gois, 2004]

Como o pólo p^+ de p aproxima o vetor normal n em p, o *cocone* aproxima uma vizinhança do plano tangente em p. Para cada ponto p, o algoritmo determina todas as arestas de Voronoi de V_p que são interceptadas pelo cocone C_p. Os triângulos duais destas arestas de Voronoi constituem o conjunto de triângulos que definem a reconstrução.

O *Cocone* possui as mesmas garantias de reconstrução do algoritmo *crust*, ou seja a correta aproximação da superfície é garantida para uma r-amostra, onde $r \leq 0.06$. A vantagem do *cocone* é que este executa apenas um cálculo da triangulação de Delaunay, diminuindo assim a complexidade do algoritmo e conseqüentemente o tempo de processamento. Outra vantagem é que o *cocone* não necessita da etapa de pós-processamento denominada *Filtering by Normal*, proposta por Amenta [Amenta and Bern, 1999a], pois esta já é alcançada diretamente pelo *cocone*. Apenas a segunda etapa de pós-processamento *trimming* é aplicada ao cocone.

4.8 Tight Cocone

Uma importante extensão do algoritmo *cocone* é apresentada por Dey e Goswami [Dey and Goswami, 2002a], o *tight cocone*. Este constitui uma etapa de pós-processamento do *cocone*. A saída deste algoritmo é uma superfície denominada *water tigh*.

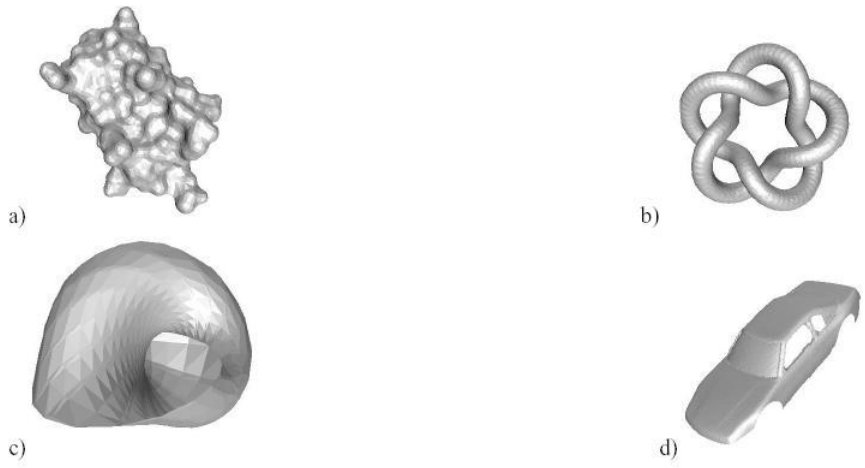

Figura 4.23: a) e b) exemplos de superfícies water tight, c) e d) superfícies não water tight. Fonte [Gois, 2004]

Definição - 4.4 *Water Tight*: Segundo [Dey and Goswami, 2002a], uma superfície é denominada *water tight* se é um 2-complexo simplicial em $\Re^3$ cujo espaço gerado é o mesmo que a fronteira de uma 3-variedade em $\Re^3$ (figura 4.23).

A idéia geral deste algoritmo é nomear os tetraedros de Delaunay como internos e externos, baseado na superfície obtida pelo *cocone*. Após todos os tetraedros serem nomeados, apenas os tetraedros considerados como internos serão utilizados para definir a fronteira da superfície denominada *water tight*. Em [Dey et al., 2001] e [Dey and Goswami, 2002b] são apresentadas discussões sobre as implementações do *cocone* e do *tight cocone*.

4.9 Considerações Finais

Neste capítulo foram apresentados os algoritmos da família *Crust*, as principais definições e teoremas relacionados às suas garantias de reconstrução. No próximo capítulo serão abordados os detalhes de implementação dos algoritmos tratados neste trabalho.

Capítulo 5

Implementações e Resultados

5.1 Considerações Iniciais

Neste capítulo serão apresentados os detalhes das implementações dos seguintes algoritmos: *raw crust 2D*, *raw crust 3D* e *power crust 3D*. Também serão discutidos os detalhes de implementação e resultados de cada um destes algoritmos, bem como as ferramentas utilizadas para a execução e visualização dos resultados.

5.2 Ferramentas utilizadas na implementação

Como descrito no capitulo anterior, a triangulação de Delaunay é a etapa mais importante dos processos de reconstrução que se baseiam na abordagem de esculpimento. Esta é sem dúvida a etapa mais custosa dos algoritmos apresentados. No melhor caso a triangulação é feita em $O(nlogn)$ e no pior caso pode assumir a complexidade quadrática (n^2) [Berg et al., 2000]. Assim, a implementação da triangulação deve ser robusta e possuir estruturas de dados eficientes. Por esse motivo utilizamos a biblioteca *CGAL- Computational Geometry Algorithms Library* versão 3.0 [CGAL, 2005]. Esta biblioteca oferece diversas triangulações implementadas de maneira eficiente e robusta. Além disso, a *CGAL* é uma biblioteca escrita em C++ padrão, podendo ser compilada em diversos sistemas operacionais. Outro motivo da escolha da *CGAL* é o fato desta ser muito bem documentada e disponível *on-line* no *site* `http://www.cgal.org`. Neste *site* é possível encontrar vários manuais, e ainda participar da lista de discussões que permite solucionar dúvidas sobre a utilização da biblioteca.

A característica mais importante desta biblioteca para o nosso trabalho é o suporte a classes com aritmética de precisão arbitrária necessária para os cálculos geométricos envolvidos nas triangulações de Delaunay tradicional e com peso. Para reconstrução de superfície é inviável trabalhar com tipos numéricos convencionais, por exemplo, *double*, já que erros de precisão numérica influenciam diretamente nos resultados das triangulações.

Um dos algoritmos que a *CGAL* disponibiliza é a triangulação de Delaunay. Sua implementação é feita de forma incremental e no pior caso sua complexidade é $O(n^2)$, sendo o caso médio $O(n \log n)$, quando os pontos são inseridos de forma aleatória. Neste caso, n é o número de pontos a serem inseridos na triangulação. Esta implementação oferece várias operações geométricas, como por exemplo, localização e inserção de pontos na triangulação. Sua estrutura de dados armazena índices que fazem acesso rápido aos elementos da triangulação, sendo facilmente possível percorrer os vértices, arestas, faces e em n-dimensão, $n > 2$, as células que compõem sua estrutura.

No pior caso, a localização de pontos na triangulação é feita em $O(n)$. No entanto, para o caso médio, tem-se $O(\sqrt{n})$. Neste trabalho é utilizada a classe *Hierarchy Triangulation* para o cálculo das triangulações.

Esta classe otimiza a busca de elementos na triangulação, tornando os algoritmos implementados mais eficientes.

A otimização da busca é baseada no armazenamento da triangulação em camadas. A última camada é responsável pela pesquisa de elementos na triangulação. O que esta triangulação faz em especial é armazenar em camadas superiores pequenas amostras de vértices da triangulação, tendo estes uma relação de vizinhança. Desta forma, a busca é iniciada na camada superior, tentando buscar o vizinho mais próximo do elemento procurado. Esta busca é feita de forma linear através de cada camada até que o elemento seja encontrado. Como cada camada armazena apenas um pequeno número de vértices da camada imediatamente inferior, a localização de pontos é feita de forma rápida já que não se faz necessário percorrer todos os pontos da triangulação. Como provado em [Devillers, 1998], esta estrutura tem um comportamento ótimo quando é utilizada na triangulação de Delaunay. Já para a triangulação com peso, a otimização da busca não encontra-se disponível na versão mais atual da *CGAL*.

A entrada do algoritmo é feita a partir da escolha de um arquivo de pontos contendo as coordenadas X e Y para o caso bidimensional; e X, Y e Z para o caso tridimensional. A saída é um conjunto de arestas em 2D, e para a versão 3D uma malha de polígonos. Para a visualização dos resultados é utilizada a linguagem *VRML (Virtual Reality Modeling Language)* [VRML, 2005]. Esta linguagem é independente de plataforma e permite a criação de ambientes virtuais através de elementos geométricos por onde se pode navegar, visualizar e interagir com os objetos criados. Para tanto é necessário apenas utilizar um *browser* que ofereça suporte à linguagem ou então instalar um *plug-in* gratuito. A figura 5.1 mostra um exemplo do ambiente virtual utilizando o *plug-in Cosmo Player* [Cosmo, 2005] no *browser*.

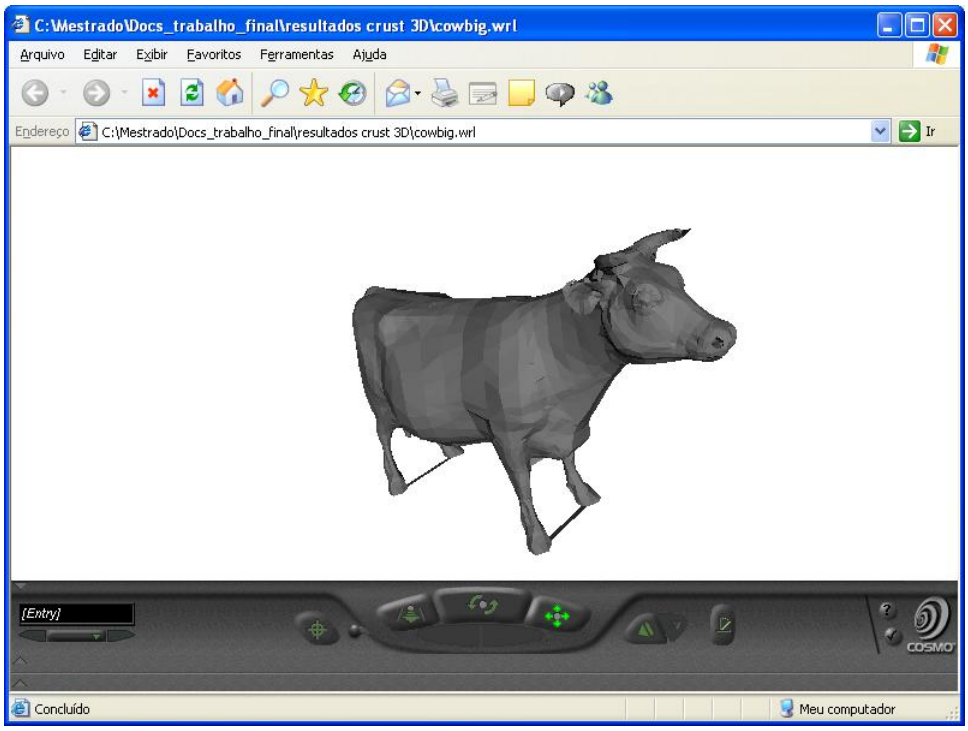

Figura 5.1: Exemplo de um ambiente virtual utilizando VRML.

5.3 Raw Crust

Neste trabalho foram implementadas tanto a versão bidimensional quanto a tridimensional do algoritmo *raw crust*. A principal diferença entre as duas versões está no fato de que na versão em três dimensões é necessário o cálculo dos pólos. Para este algoritmo não foram encontradas maiores dificuldades, uma vez que a *CGAL* oferece todos os cálculos necessários para sua implementação: triangulação e cálculo de duais. Tanto na versão bidimensional quanto na versão tridimensional, a triangulação de Delaunay é calculada duas vezes, a primeira com os n pontos da amostra e a segunda com os pontos da amostra mais os vértices de Voronoi. Em $3D$, a triangulação não é feita com todos os vértices de Voronoi, neste caso selecionam-se dois vértices por amostra (pólos). Assim, a segunda triangulação é feita com $3n$ pontos, n da amostra inicial e mais dois pólos para cada ponto da amostra.

Na figura 5.2 é mostrado um exemplo da reconstrução de uma curva utilizando o algoritmo *crust 2D*. Em (a) temos os pontos da amostra, em (b) a triangulação de Delaunay dos pontos da amostra e em (c) a reconstrução. Como pode-se observar, para esta amostra, a reconstrução foi correta, uma vez que a curva dada é bastante suave. O mesmo pode ser observado no exemplo ilustrado na figura 5.3.

Observe que no próximo exemplo (figura 5.4), a curva possui pontos que se interceptam em seu interior. Alguns desses pontos não foram considerados no resultado da reconstrução. Isso ocorreu pelo fato do algoritmo ter considerado que esses pontos eram internos à superfície, fazendo com que a reconstrução não fosse totalmente satisfeita.

Os exemplos apresentados em seguida são relativos à versão em três dimensões do algoritmo *raw crust*. A primeira figura (5.5) mostra a reconstrução de uma superfície referente a uma amostra de 766 pontos. Observe que neste caso a reconstrução foi satisfatória, isso porque esta é uma superfície bem suave. Além disso, pode-se perceber que a amostra de pontos tem uma boa distribuição sobre a superfície, através da qual o algoritmo conseguiu reconstruir todas as regiões de forma correta.

As duas próximas figuras (5.6 e 5.7) exibem dois exemplos da reconstrução para amostras de 1.000 e 1.500 pontos respectivamente. Em ambos os casos pode-se notar que a amostra possui falhas, principalmente em regiões que caracterizam o objeto.

Nas figuras (5.8 e 5.9) é feito um *zoom* das duas superfícies anteriores com o objetivo de evidênciar a relação da densidade de determinadas regiões da amostra com a correta reconstrução do objeto. Em ambas as figuras algumas das regiões mais características dos dois objetos são identificadas pelos círculos contínuos. Observe que nestas regiões a amostra possui algumas falhas, fazendo com que buracos apareçam na superfície. Já para as regiões menos características (círculos pontilhados), uma densidade menor de pontos não ocasiona problemas na reconstrução. Outra observação a ser feita, é com relação às regiões pontiagudas dos objetos. Nessas regiões a densidade dos pontos deve ser maior, pois neste caso, a distância de um ponto da superfície ao eixo medial é muito pequena. Uma vez que as garantias do *raw crust* são fracas ($LSF(r) < 0.06$), essas regiões só seriam reconstruídas corretamente se a densidade de pontos nessas áreas fosse suficientemente grande, o que não acontece nos dois exemplos anteriores.

A figura 5.10 mostra o exemplo de duas amostras com 2164 e 2655 pontos respectivamente.

Os resultados de ambas as reconstruções são mostrados na figura 5.11 e 5.12. Na primeira, é possível notar que a reconstrução está bem próxima do esperado, uma vez que esta superfície é bem mais suave. Ainda assim, pode-se identificar alguns buracos na região em destaque. Na figura 5.12, observa-se que a superfície, apesar de possuir bastante detalhes, também teve um bom resultado, devido à maior densidade de pontos da amostra. Ainda assim, pode-se visualizar a presença de pequenos buracos na superfície.

Por último, a figura 5.13 apresenta o resultado para uma amostra com 3.065 pontos. Observe que, em comparação aos exemplos anteriores, esta superfície apresentou melhores resultados. Observe que a amostra deste exemplo é consideravelmente mais densa que as anteriores, inclusive em regiões que caracterizam o objeto, como por exemplo, a região dos olhos e chifres da vaca. No entanto, mesmo a amostra sendo densa, esta também apresenta algumas falhas nas regiões em destaque na figura 5.13.

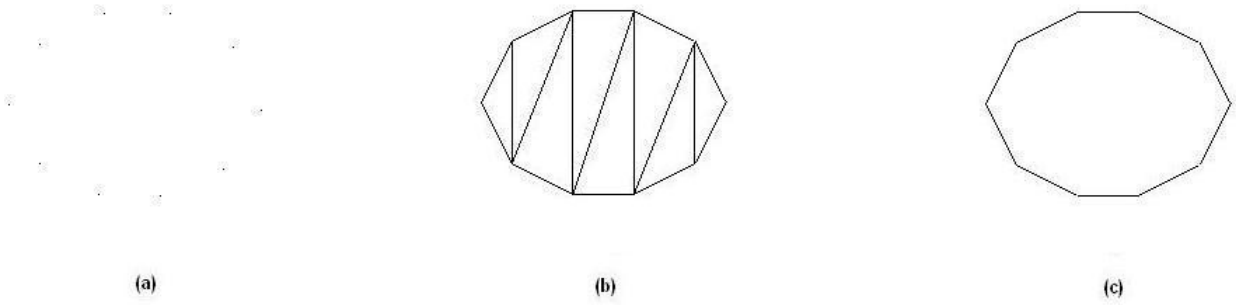

Figura 5.2: Exemplo de reconstrução para algoritmo *crust 2D*. (a) Pontos da amostra. (b) Triangulação dos pontos da amostra. (c) Resultado da reconstrução.

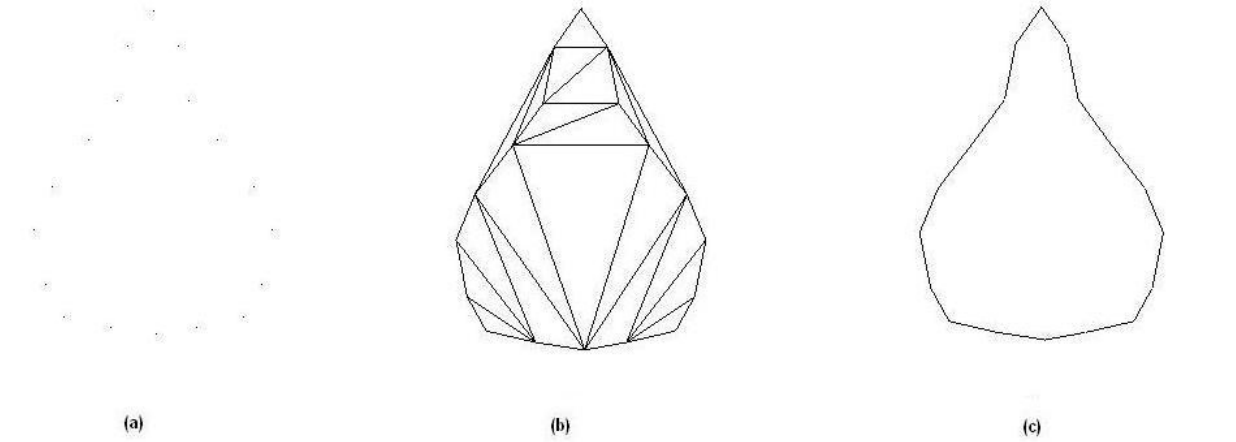

Figura 5.3: Exemplo de reconstrução para algoritmo *crust 2D*. (a) Pontos da amostra. (b) Triangulação dos pontos da amostra. (c) Resultado da reconstrução.

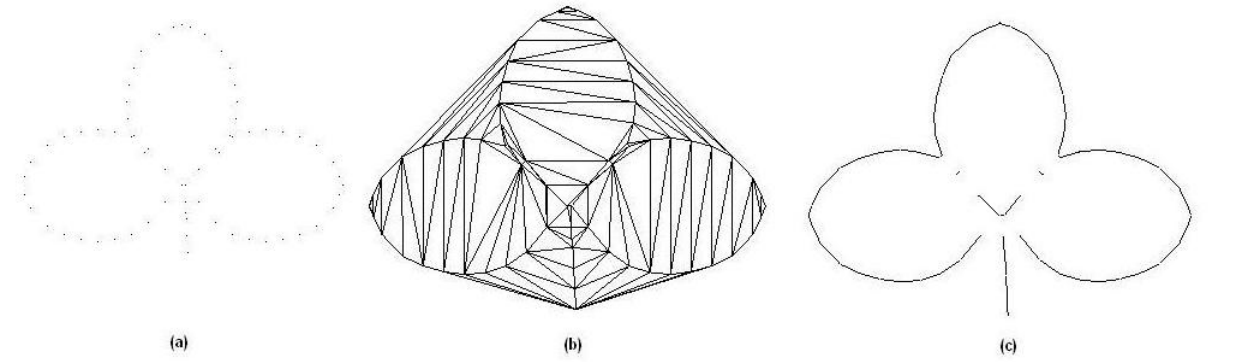

Figura 5.4: Exemplo de reconstrução para algoritmo *raw crust 2D*. (a) Pontos da amostra. (b) Triangulação dos pontos da amostra. (c) Resultado da reconstrução.

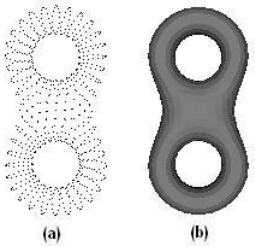

Figura 5.5: Resultado da reconstrução da superfície para uma amostra com 766 pontos. Em (a) são exibidos os pontos da amostra e em (b) o resultado da reconstrução para o algoritmo *raw crust*.

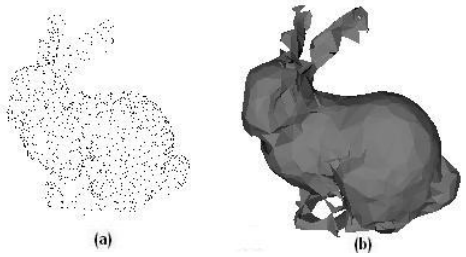

Figura 5.6: Resultado da reconstrução da superfície para uma amostra com 1.000 pontos. Em (a) são exibidos os pontos da amostra e em (b) o resultado da reconstrução para o algoritmo *raw crust*.

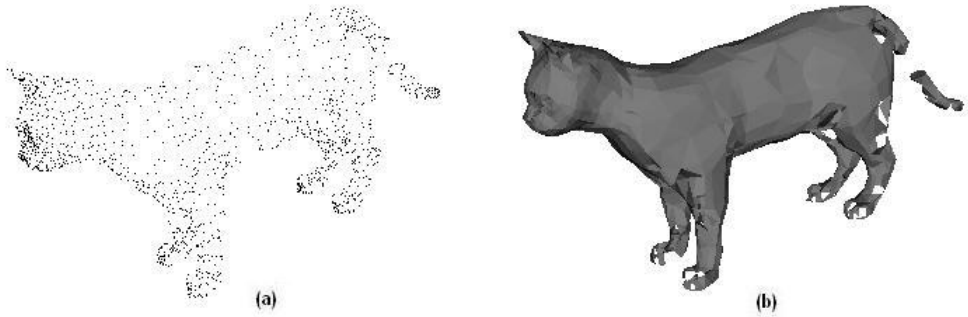

Figura 5.7: Resultado da reconstrução da superfície para uma amostra com 1.500 pontos. Em (a) são exibidos os pontos da amostra e em (b) o resultado da reconstrução para o algoritmo *raw crust*.

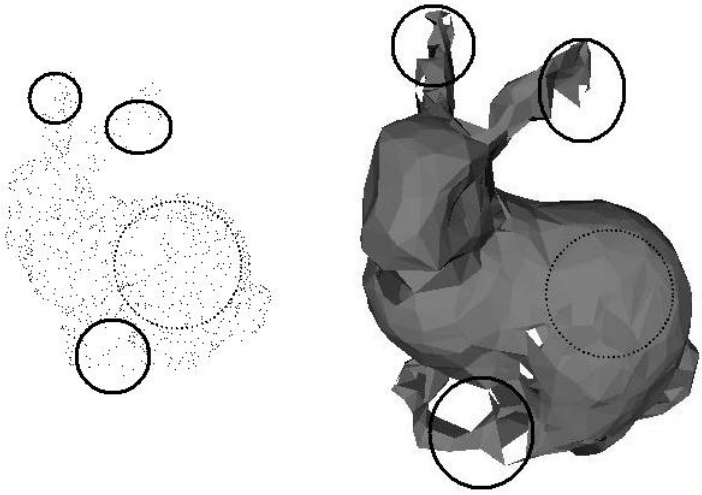

Figura 5.8: *Zoom* do resultado da reconstrução da superfície para o exemplo da figura 5.6. Os círculos contínuos exemplificam a relação das falhas da amostra e a presença de buracos em regiões características do objeto. Os círculos pontilhados exemplificam regiões da amostra menos características, onde a densidade dos pontos não provoca problemas na reconstrução.

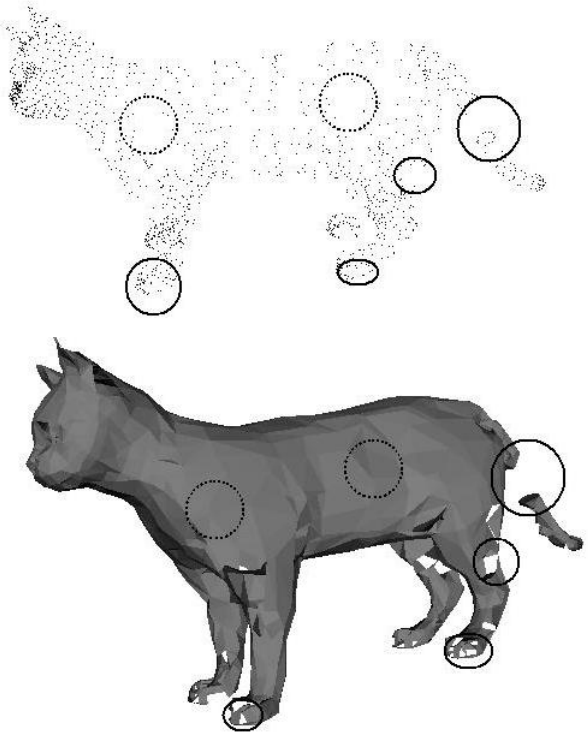

Figura 5.9: *Zoom* do resultado da reconstrução da superfície para o exemplo da figura 5.7. Os círculos contínuos exemplificam a relação das falhas da amostra e a presença de buracos em regiões características do objeto. Os círculos pontilhados exemplificam regiões da amostra menos características, onde a densidade dos pontos não provoca problemas na reconstrução.

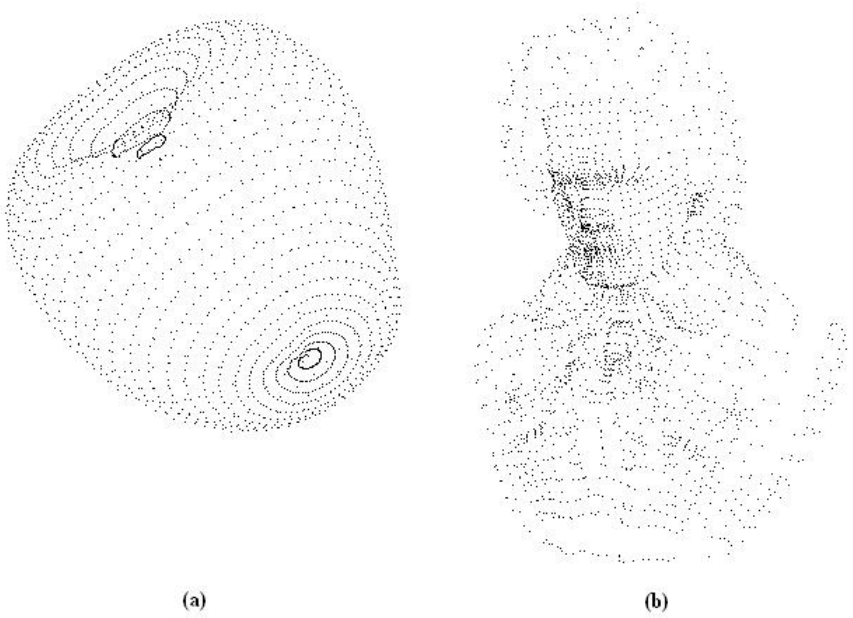

Figura 5.10: Amostra de pontos de dois objetos. Em (a) a amostra representada possui 2164 pontos e em (b) a amostra é constituída de 2655 pontos.

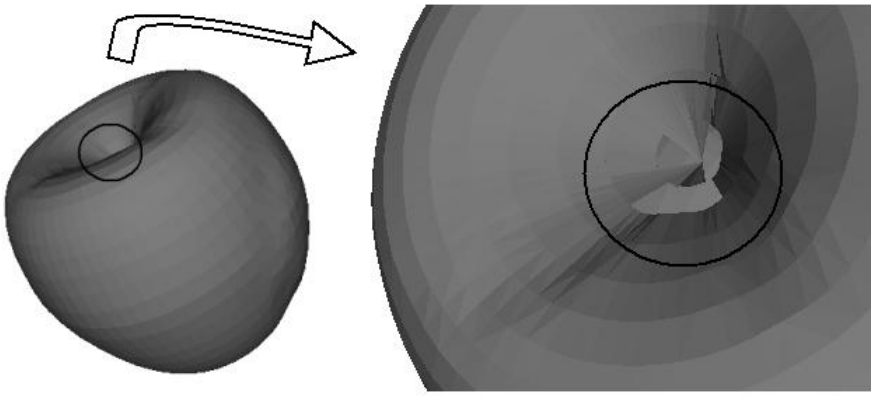

Figura 5.11: Resultado da reconstrução da amostra de pontos apresentada na figura 5.10 (a). É possível identificar a presença de buracos na região em destaque.

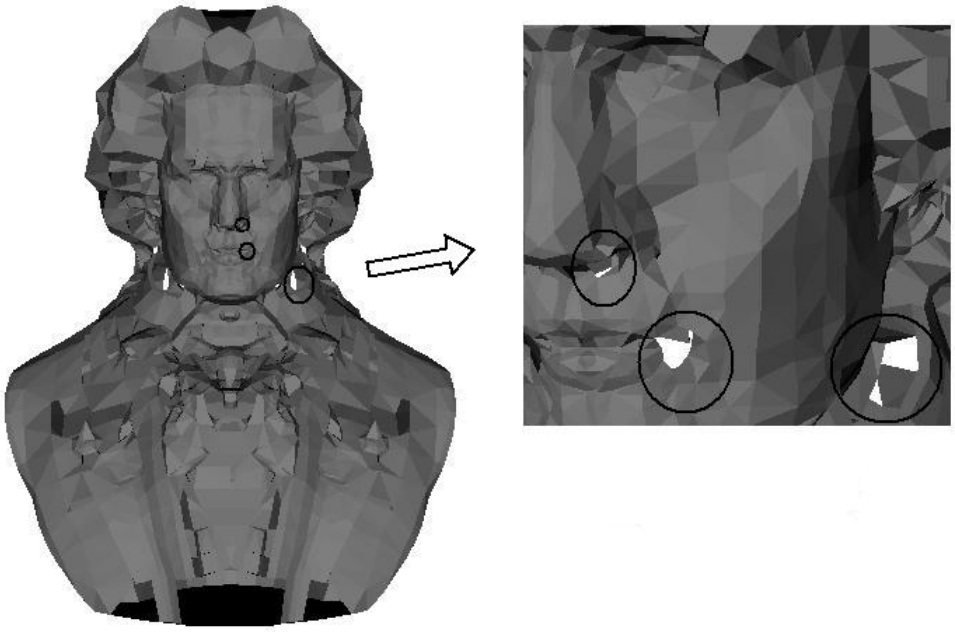

Figura 5.12: Resultado da reconstrução da amostra de pontos apresentada na figura 5.10 (b). Podemos identificar a presença de buracos na região em destaque.

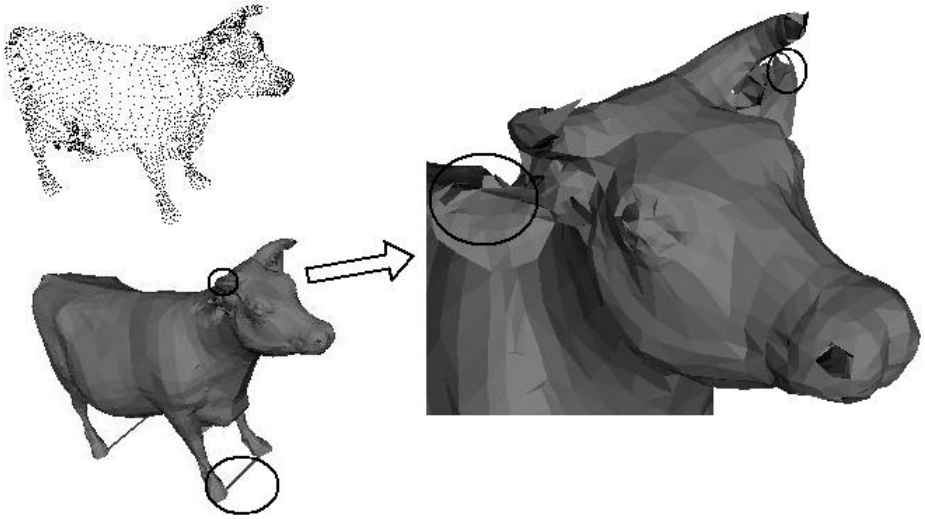

Figura 5.13: Resultado da reconstrução da superfície para uma amostra com 3.065 pontos. Inicialmente são exibidos os pontos da amostra. Em seguida o resultado da reconstrução para o algoritmo *raw crust*. As regiões destacadas na figura são aquelas em que o algoritmo apresentou problemas na reconstrução.

5.4 Power Crust

Conforme apresentado no capítulo anterior, o algoritmo *power crust* possui maiores garantias de reconstrução que o *raw crust*, sendo esta a motivação para sua implementação. No entanto, este algoritmo é mais complexo, e na prática não apresentou os resultados esperados.

Assim como na versão $3D$ do algoritmo *raw crust*, na implementação do *power crust* também utilizou-se a estratégia do *bounding box*. Conforme indicado por [Gois, 2004], em nossos testes colocamos *bounding Box* de 5 a 10 vezes o tamanho do menor *bounding Box* para obter-se resultados satisfatórios.

Após o cálculo dos pólos, a principal rotina envolvida no algoritmo do *power crust* é a classificação dos pólos como internos ou externos à superfície. Caso esta rotina não funcione corretamente, então o algoritmo como um todo ficará comprometido. Na próxima seção será apresentada uma versão prática do algoritmo de rotulação dos pólos proposta por Amenta em [Amenta et al., 2001a].

5.4.1 Algoritmo prático para rotulação dos pólos

No capítulo 4 foi apresentado um algoritmo teórico para a nomeação dos pólos. Em [Amenta et al., 2001b] é provado que este algoritmo é correto uma vez que a interseção entre duas bolas polares internas e externas, quando existe, é pequena e além disso uma bola polar é interna e a outra sempre externa. No entanto, este algoritmo é muito sensível a perturbações nas amostras. Caso alguma bola seja rotulada de forma incorreta, esse erro é propagado, podendo comprometer todo o processo de reconstrução.

Em [Amenta et al., 2001a] Nina Amenta apresenta um algoritmo baseado numa lista de prioridade. A idéia principal desta heurística é rotular as bolas polares mais confiáveis primeiramente, e por último aquelas onde existe dúvida em relação a sua classificação. Esse grau de confiança é medido através de duas variáveis denominadas **in** e **out**. A variável **in** indica o quão interna ao objeto a bola polar é, e **out** indica o quão externa esta bola pode estar, onde 0 é incerteza e 1 é certeza.

O processo é iniciado atribuindo às bolas polares adjacentes ao *bouding Box* **out** = 1 e **in** =0. Os demais pólos são inicializados com **in=out**=0. Todas as bolas polares são colocadas numa lista de prioridade determinada pelos valores **in** e **out**. Se um dos valores de **in** e **out** for nulo, então a prioridade é determinada pelo valor não-nulo. Se os dois valores forem não-nulo a prioridade é calculada da seguinte forma: $|in - out| - 1$. Observe que a prioridade neste caso pode variar entre 0 e 1. O algoritmo para atualização da prioridade é detalhado em seguida.

ALGORITMO [4.7]: *Atualização da prioridade*

```
1:  se p.in > 0 e p.out > 0 então
2:    p.priority = | p.in − p.out | − 1
3:  caso contrário
4:    p.priority = max(p.in, p.out)
5:  fim do se
```

A cada iteração o elemento com maior prioridade é rotulado e removido da lista. Os valores **in** e **out** do pólo atual são propagados para os demais pólos segundo duas estratégias. A primeira é baseada no ângulo formado entre as duas bolas polares calculadas para cada amostra. Conforme o cálculo dos pólos apresentado na seção 4.5, o ângulo β entre dois pólos opostos deve estar entre $\pi/2$ e π. A figura 5.14 apresenta um exemplo bidimensional onde são apresentados dois pontos da amostra s e seus respectivos pólos opostos p e q. Note que neste exemplo um mesmo pólo foi calculado para dois pontos da amostra diferentes.

De acordo com o ângulo β, podemos estabelecer uma relação entre o rótulo de dois pólos opostos. Quanto mais distante um pólo estiver do outro, maior será o ângulo β e maior será a chance de ambos

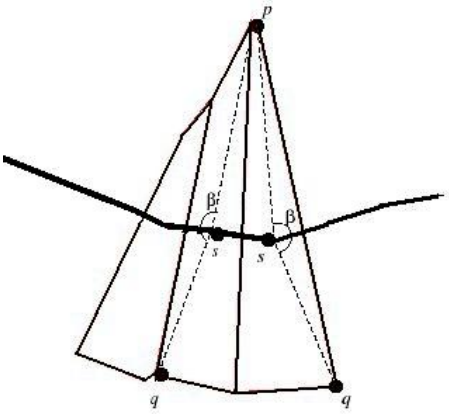

Figura 5.14: Em linhas contínuas é mostrado o diagrama de Voronoi. O segmento em negrito representa uma aresta de Delaunay. O ângulo formado pelo ponto da amostra s e seus dois respectivos pólos é dado por β. Ao nomear o pólo p, alteram-se a prioridade do pólo q, pelo valor do ângulo β.

possuírem rótulos diferentes. Assim utiliza-se o coseno do ângulo de β para medir essa proporção. Observe que à medida que o ângulo β se aproxima de π, o $\cos(\beta)$ se aproxima de -1. Esse valor, multiplicado por (-1) indica a chance de dois pólos opostos possuirem rótulos diferentes. Por outro lado, quando o ângulo β se aproxima de $\pi/2$, o $\cos(\beta)$ se aproxima de 0, indicando que os dois pólos opostos possuem o mesmo rótulo. Desta forma, o valor do $\cos(\beta)$ é utilizado para medir a conexão entre os pólos p e q, sendo $0 \leq -\cos(\beta) \leq 1$.

A segunda estratégia é baseada na interseção entre as bolas polares vizinhas. Conforme Amenta em [Amenta et al., 2001a], a interseção entre uma bola polar interna e outra externa deve ser rasa. Caso esta interseção seja profunda, então é provável que ambas tenham o mesmo rótulo. Em [Amenta et al., 2001a], a interseção entre duas bolas polares é considerada profunda quando o ângulo α formado por esta interseção está entre $\pi/2$ e π (figura 4.19). Assim, à medida que o ângulo α se aproxima de π (interseção profunda), o $\cos(\alpha)$ se aproxima de -1. Este valor multiplicado por (-1) indica que as duas bolas possuem o mesmo rótulo. O contrário acontece quando o ângulo α se aproxima de $\pi/2$. Portanto, como medida de conexão entre as bolas polares vizinhas utiliza-se $0 \leq -\cos(\alpha) \leq 1$. O cálculo do ângulo α será detalhado na próxima seção. Todo esse processo é repetido até que todas as bolas polares sejam devidamente rotuladas. Este algoritmo é resumido em seguida.

ALGORITMO [4.8]: ***Classificação dos pólos (prático)***

```
1:  Para todo pólo p ∈ P faça
2:    p.in = p.out = 0.0;
3:    insira p na lista de prioridade L
4:  fim do para

5:  Para todo p adjacente ao bounding box faça
6:    p.out = 1;
7:    L.atualize(p);
8:  fim do para

9:  Enquanto L ≠ ∅  faça
10:   L.remove_topo(p)
```

```
11:   se p.in > p.out então
12:     p.label = IN e p.temp = p.in
13:   caso contrário
14:     p.label = OUT e p.temp = p.out
15:   fim do se

16:   Para todo s tal que p é pólo faça
17:     Tome q como sendo o outro pólo de s
18:     q.labeloposto(p.label) = max(p.temp * ( −cos(p̂sq) ), q.labeloposto(p.label)
19:     L.atualize(q)
20:   fim do para

21:   Para toda intersecção α entre p e seus pólos vizinhos q faça
22:     q.(p.label) = max(p.temp * (-cos(α)), q.(p.label)
23:   fim do para
24: fim do enquanto
```

Neste trabalho foi implementada apenas a versão $3D$ do *power crust* utilizando o algoritmo prático, uma vez que nosso objetivo é reconstruir modelos geométricos $3D$. Uma das dificuldades encontradas neste trabalho foi relativa ao cálculo de duais com peso. A versão 3.1 da biblioteca *CGAL* não possui uma implementação do cálculo de duais de tetraedros para a triangulação de Delaunay com peso, conhecido como ortocentro. Esta rotina foi implementada segundo os predicados disponíveis em `http://www.cs.utexas.edu/users/amenta/powercrust/`. A teoria envolvida no cálculo do ortocentro foi retirada do trabalho de [Gois, 2004] onde o cálculo do ortocentro é feito através de determinantes, conforme mostrado em seguida.

Seja o tetraedro t com vértices com peso $p_1 = (x_1, y_1, z_1, w_1^2), p_2 = (x_2, y_2, z_2, w_2^2), p_3 = (x_3, y_3, z_3, w_3^2)$ e $p_4 = (x_4, y_4, z_4, w_4^2)$ e com ortocentro $o = (o_x, o_y, o_z, o_w^2)$. A expressão do ortocentro pode ser determinada a partir dos seguintes cálculos:

$$a = \begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{pmatrix} \tag{5.1}$$

$$b_x = + \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 - w_1^2 & y_1 & z_1 & 1 \\ x_2^2 + y_2^2 + z_2^2 - w_2^2 & y_1 & z_2 & 1 \\ x_3^2 + y_3^2 + z_3^2 - w_3^2 & y_3 & z_3 & 1 \\ x_4^2 + y_4^2 + z_4^2 - w_4^2 & y_4 & z_4 & 1 \end{vmatrix} \tag{5.2}$$

$$b_y = - \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 - w_1^2 & x_1 & z_1 & 1 \\ x_2^2 + y_2^2 + z_2^2 - w_2^2 & x_1 & z_2 & 1 \\ x_3^2 + y_3^2 + z_3^2 - w_3^2 & x_3 & z_3 & 1 \end{vmatrix} \tag{5.3}$$

$$b_z = + \begin{vmatrix} x_1^2 + y_1^2 + z_1^2 - w_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 + z_2^2 - w_2^2 & x_1 & y_2 & 1 \\ x_3^2 + y_3^2 + z_3^2 - w_3^2 & x_3 & y_3 & 1 \end{vmatrix} \tag{5.4}$$

Consideremos $o_x = \frac{b_x}{2a}$, $o_y = \frac{b_y}{2a}$ e $o_z = \frac{b_z}{2a}$, então o ortocentro é dado por:

$$\hat{o} = (o_x, o_y, o_z, (o_x - x_1)^2 + (o_y - y_1)^2 + (o_z - z_1)^2 - w_1^2) \tag{5.5}$$

O principal problema encontrado no cálculo do ortocentro foi que o determinante calculado em 5.1 pode ser muito próximo de zero, caso em que os pontos são praticamente coplanares. Na CGAL, a classe numérica utilizada para representar os números flutuantes com precisão arbritárias não suporta divisão exata e converte numerador e denominador para *double* padrão da linguagem C++, fazendo com que o denominador se anule.

Para contornar esse problema [Gois, 2004] sugere que seja feita uma perturbação em uma das coordenadas dos vértices do tetraedro quando se identificar que a divisão será feita por zero. Outra estratégia é verificar a ortogonalidade do ortocentro com seu tetraedro. Este teste verifica se o ortocentro calculado é ortogonal aos quatro vértices do tetraedro. Essas estratégias não são citadas no trabalho original de Amenta, uma vez que a triangulação e os predicados utilizados na implementação do *power crust* fornecem resultados satisfatórios [Amenta et al., 2001a]. Neste trabalho não foram implementadas essas estratégias, devendo estas duas alternativas serem melhor estudadas e implementadas futuramente.

Em [Amenta et al., 2001a] ainda são apresentados tratamentos específicos para identificação de buracos, ruídos e células de Voronoi finas e pontiagudas. Esses tratamentos também não foram implementados, sendo necessário maior estudo para uma implementação futura.

5.4.2 Cálculo do ângulo entre duas esferas

Para a correta nomeação dos pólos, processo principal na rotina do *power crust* é necessário calcular a interseção entre duas bolas polares vizinhas na triangulação com peso. Na figura 5.15 é mostrado o corte planar de duas esferas que se interceptam.

Dada as duas esferas $e1$ e $e2$ com centros e raios respectivamente c_1, c_2 e r_1, r_2. Seja $d = ||c_1 - c_2||$, $d_1 = ||c_1 - c_3||$ e $d_2 = ||c_2 - c_3||$.

Assim tem-se o seguinte sistema de equações:

$$\begin{cases} d_1 + d_2 = d \\ d_1^2 + r_3^2 = r_1^2 \\ d_2^2 + r_3^2 = r_2^2 \end{cases} \tag{5.6}$$

Através de substituições obtém-se as seguintes equações:

$$d_1 = \frac{r_1^2 - r_2^2 + d^2}{2d} \tag{5.7}$$

$$d_2 = d - d_1 \tag{5.8}$$

$$r_3 = \sqrt{(r_1^2 - d_1^2)} \tag{5.9}$$

Os ângulos γ e ϵ na figura 5.15 podem ser encontrados através do cálculo dos cosenos:

$$\cos(\delta) = \frac{r_3}{r_1} \tag{5.10}$$

$$\cos(\epsilon) = \frac{r_3}{r_2} \tag{5.11}$$

Seja $\beta = \angle c_1 k c_2 = \delta + \epsilon$. O ângulo entre as esferas é dado por $\alpha = \pi - \beta$, uma vez que os planos tangentes às duas esferas são perpendiculares aos segmentos $\overline{c_1 k}$ e $\overline{c_2 k}$ em k. Por substituição tem-se que $\alpha = \pi - (\gamma + \epsilon)$.

No entanto, o cálculo deste ângulo pode não funcionar em alguns casos. Primeiramente, imagine que não exista interseção entre as duas bolas polares. Observe que neste caso, a distância d entre os centros c_1 c_2 é maior que a soma dos raios r_1, r_2 (figura 5.16).

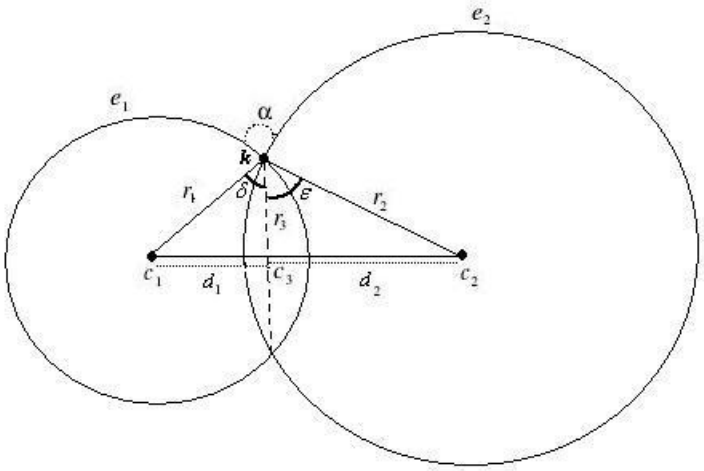

Figura 5.15: Corte planar da interseção entre duas bolas polares.

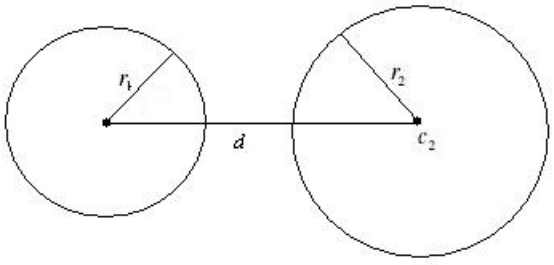

Figura 5.16: Ausência de interseção entre duas bolas polares.

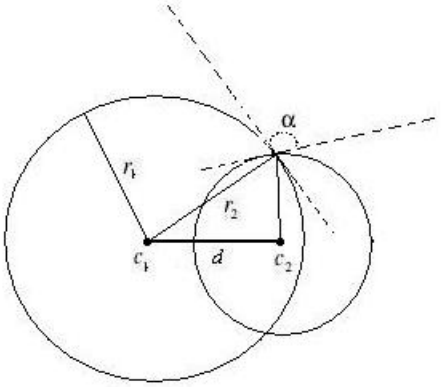

Figura 5.17: Interseção entre duas bolas polares, onde o centro de uma esfera está no interior de outra.

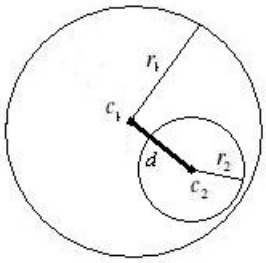

Figura 5.18: Bola polar totalmente interna a outra bola polar.

É possível verificar também a existência de uma outra exceção: o centro de uma esfera pode estar dentro da outra (figura 5.17). Neste caso a interseção entre as esferas é profunda (bolas polares possuem o mesmo rótulo), sendo o ângulo α necessariamente maior que $\pi/2$. Esta exceção ocorre quando:

$$r_1 > r_2 \text{ quando } r_1 > d, \text{ ou}$$

$$r_2 > r_1 \text{ quando } r_2 > d.$$

A figura 5.18 mostra uma situação em que não há interseção, mas que uma bola está completamente no interior de outra. Neste caso pode-se considerar que as duas bolas possuem o mesmo rótulo. Note que esta exceção ocorre quando:

$$d + r_2 < r_1, \text{ caso em que } r_1 \geq r_2, \text{ ou}$$

$$d + r_1 < r_2, \text{ caso em que } r_2 \geq r_1.$$

5.4.3 Resultados do *Power Crust*

Os resultados obtidos com o *power crust* serão apresentados passo a passo, conforme o algoritmo apresentado no capítulo 4. Na figura 5.19 temos os pontos de duas amostras. Após o cálculos dos pólos positivo e negativo para cada amostra, esses pólos são utilizados no cálculo do *power* diagrama, e posteriormente

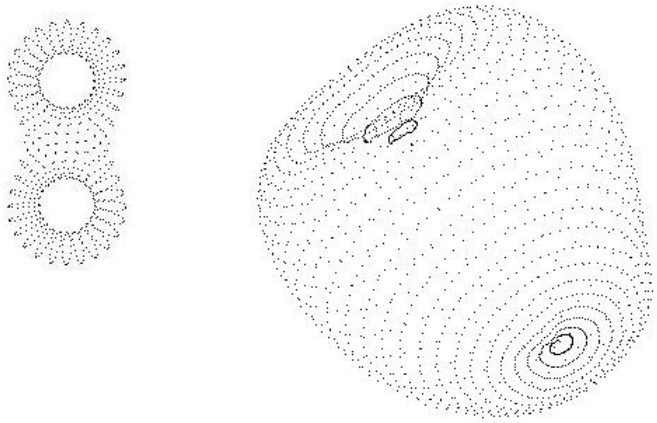

Figura 5.19: Pontos das amostras de duas superfícies a serem reconstruídas.

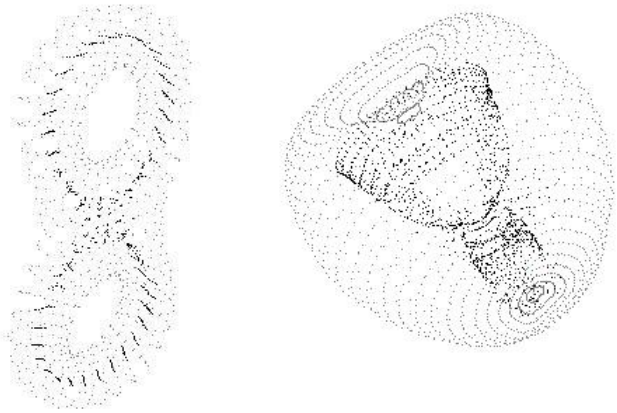

Figura 5.20: Em preto, os pólos classificados como internos à superfície, e em cinza os pontos da amostra.

classificados como internos ou externos à superfície. Na figura 5.20 são mostrados os pontos das duas amostras em cinza, e em preto os pólos classificados como internos.

Na figura 5.21 temos para as mesmas amostras, os pólos classificados como externos (pontos pretos).

A próxima figura (5.22) apresenta a aproximação da superfície através da união das bolas polares, classificadas como internas ao objeto. Note que, para a primeira amostra, a união das bolas polares internas dá uma boa aproximação para a superfície. Já para o segundo exemplo, a aproximação não foi muito próxima. Esse fato pode ocorrer quando algumas bolas polares recebem rótulos incorretos.

Por último, a figura 5.23 apresenta a reconstrução das duas superfícies. Como pode-se notar, mesmo com a correta classificação dos pólos mostrada nas figuras anteriores, os resultados da reconstrução para o *power crust* não foram os esperados, a superfície reconstruída em ambos os casos apresenta muitos buracos. Um dos motivos das presença desses buracos é ocasionado por problemas de ordem numérica durante o cálculo do ortocentro. A classe numérica com precisão arbritária utilizada nesta implementação não suporta a operação de divisão, sendo necessário transformar os vértices da triangulação de Delaunay em *double*. Assim, perde-se precisão justamente no cálculo dos vértices dos polígonos da superfície resultante. Futuramente, será necessário revisar a última parte do algoritmo com o objetivo de encontrar possíveis problemas na implementação que possam estar gerando o resultado da reconstrução de forma incorreta.

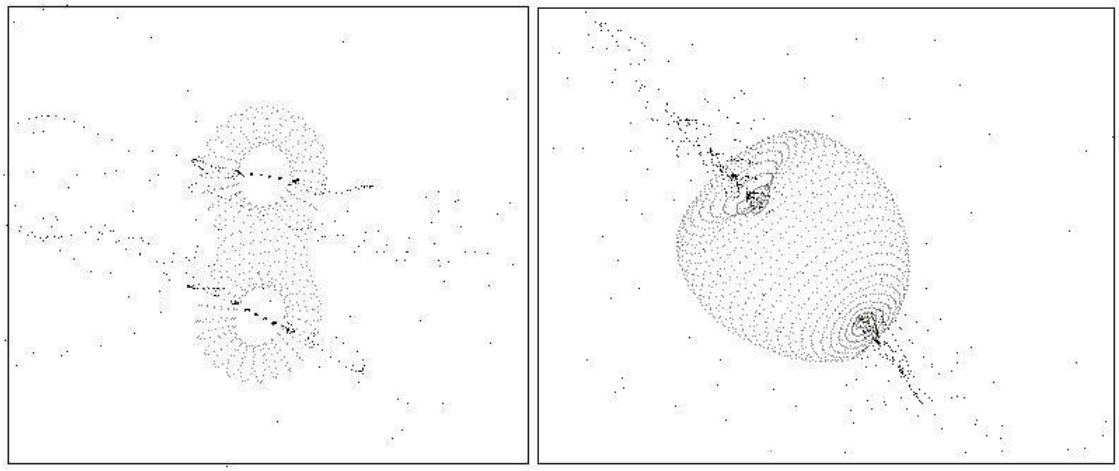

Figura 5.21: Em preto, pólos classificados como externos à superfície.

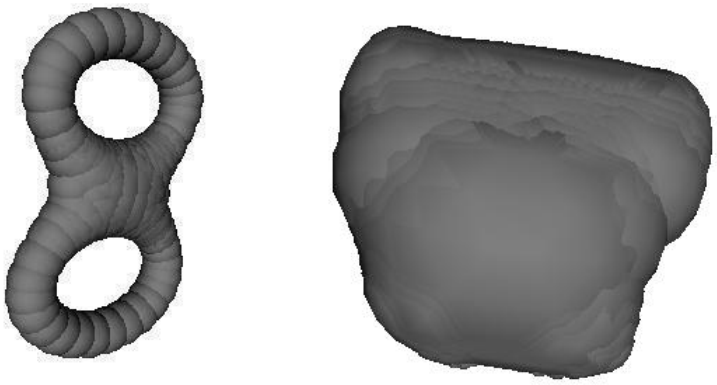

Figura 5.22: Aproximação do objeto através da união das bolas polares internas. Para a primeira amostra, tem-se uma boa aproximação do objeto. No segundo exemplo, bolas incorretamente classificadas fazem com que a união das bolas polares internas não seja uma boa aproximação para a superfície.

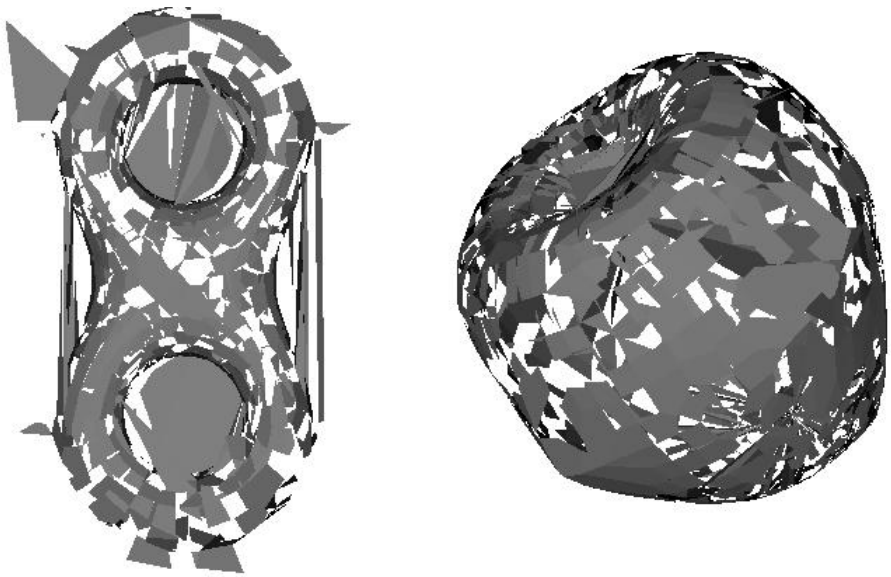

Figura 5.23: Resultados da reconstrução de duas superfícies utilizando o algoritmo *power crust*.

5.5 Considerações Finais

Neste capítulo foram descritas as implementações realizadas, bem como os resultados obtidos e alguns exemplos de reconstrução. As principais rotinas dos algoritmos *crust* e *power crust* foram detalhadas e evidenciadas as principais dificuldades encontradas na implementação do algoritmo *power crust*.

Capítulo 6

Conclusões e trabalhos futuros

Neste trabalho foi apresentada uma revisão bibliográfica que abordou os principais métodos utilizados no processo de reconstrução de superfícies: métodos baseados em esculpimento, em funções implícitas, incrementais e modelos deformáveis. Descrevemos as principais vantagens e dificuldades de cada método, dando maior ênfase aos algoritmos de esculpimento. Observamos que a maioria dos métodos apresentados não possuem garantias teóricas de reconstrução. Por esse motivo destacamos os algoritmos da família *Crust*, pois estes apresentam um embasamento teórico a respeito das garantias de reconstrução.

Como mostrado no capítulo 4, as garantias teóricas dos algoritmos da família *Crust* são baseadas na qualidade das amostras. Esta qualidade é medida através da relação entre a densidade da amostra e da distância de seus pontos ao eixo medial. Segundo Amenta [Amenta and Bern, 1999b], uma r-amostragem é aquela em que a distância Euclidiana entre qualquer ponto da superfície ao ponto mais próximo da amostra é r vezes a distância do ponto da superfície ao ponto mais próximo do eixo medial. A constante r é utilizada para definir as garantias teóricas dos algoritmos. Em $2D$ monstramos que as garantias de reconstrução para algoritmo β-*Skeleton* são melhores que para o algoritmo *raw crust*. Enquanto o primeiro possui garantias de reconstrução para $r < 0.297$, o segundo garante a reconstrução apenas para $r < 0.252$. Porém, não se conhecem trabalhos que descrevam a tentiva de implementação do β-*Skeleton* em $\Re^3$, diferentemente do *raw crust*. Para o *raw crust* 3D mostramos que o conjunto de triângulos resultante deste algoritmo é topologicamente equivalente à superfície para $r < 0.06$.

Ainda no capítulo 4 abordamos as definições e garantias teóricas relacionadas ao algoritmo *power crust*. Este algoritmo, apesar de apresentar melhores garantias de reconstrução que o *raw crust* ($r < 0.01$), é bem mais complexo, envolvendo cálculos adicionais como: triangulação de Delaunay com peso, interseção entre bolas polares e cálculo do ortocentro dos tetraedros da triangulação.

No capítulo 5, apresentamos os detalhes das implementações dos algoritmos *crust 2D*, *raw crust 3D* e *power crust 3D*. Para o algoritmo *raw crust 2D* mostramos que para curvas suaves, este aproxima a curva reconstruída da original. No entanto, para curvas que se interceptam, o *raw crust 2D* considera que os pontos desta interseção não pertencem à curva original, apresentando falhas nestas regiões. Para a versão tridimensional deste algoritmo foram apresentados alguns exemplos, através dos quais pudemos evidenciar a relação da densidade da amostra com a correta reconstrução da superfície. Mostramos que regiões da amostra com baixa densidade de pontos podem gerar buracos na superfície reconstruída. Além disso, também destacamos a dificuldade de reconstrução em áreas finas ou pontiagudas dos objetos. Nessas regiões, a correta reconstrução só é garantida quando a densidade de pontos da amostra é muito grande. Em ambas as versões 2D e 3D do algoritmo raw crust, notamos que suas implementações não apresentaram maiores dificuldades, uma vez que, comparado ao *power crust*, seu desenvolvimento é bem mais simples.

Para o algoritmo *power crust*, apresentamos passo a passo os resultados obtidos em sua implementação, pois este apresenta cálculos adicionais em relação ao *raw crust*, sendo portanto, mais complexo. O *power crust*, além de retornar a malha de polígonos que representa a superfície reconstruída, também possibilita

a aproximação do objeto através da união de suas bolas polares. Além disso, este também retorna uma aproximação para o eixo medial, a partir da qual a superfície é reconstruída. No entanto, no desenvolvimento deste algoritmo foram encontradas algumas dificuldades de implementação não referenciadas nos trabalhos originais, como por exemplo, o problema numérico encontrado no cálculo de duais da triangulação com peso.

A principal diferença deste trabalho em relação ao trabalho original de Amenta [Amenta et al., 2001a], foi a utilização da biblioteca CGAL. Esta biblioteca apresenta a maioria dos cálculos utilizados nestes algoritmos, como as triangulações de Delaunay tradicional e com peso. No entanto, esta biblioteca não apresenta o cálculo de duais para a triangulação com peso, constituindo a principal dificuldade para o algoritmo *power crust*. Este cálculo de duais é utilizado para se obter o power diagrama, através do qual são extraídas as faces poligonais que compõem a superfície do objeto reconstruído. Por problemas de implementação, mostramos que mesmo classificando corretamente os pólos, gerados a partir das amostras consideradas, ainda assim não conseguimos obter os resultados esperados para o power crust.

Baseado nos estudos apresentados e nas implementações realizadas, podemos definir um direcionamento futuro deste trabalho. O nosso objetivo futuro será implementar as melhorias e correções necessárias para contornar os problemas numéricos do algoritmo *power crust*. Em seguida realizar comparações de ordem prática entre o *power crust* e *crust*, levando em consideração o tempo de processamento e a qualidade da reconstrução.

Além disso, deveremos estudar e propor uma etapa de pós-processamento para suavização da malha de polígonos resultantes. Este pós-processamento tem o objetivo de tornar a malha de polígonos resultantes o mais regular possível. Por exemplo, no caso do crust, em que a saída é um conjunto de triângulos, o objetivo seria tornar os triângulos o mais eqüiláteros quanto possível.

Após a avaliação dos resultados, poderemos verificar as demais necessidades de adaptações do algoritmo e então gerar uma ferramenta robusta que será integrada ao modelador de sólidos em desenvolvimento pelo **GOPAC (GSM)**, sendo esta a principal motivação deste trabalho.

Referências Bibliográficas

[Adamy et al., 2000] Adamy, U., Giesen, J., and John, M. (2000). Surface reconstruction using umbrella filters. *Computational Geometry Theory and Applications*, 21:63–86.

[Algorri and Schmitt, 1996] Algorri, M.-E. and Schmitt, F. (1996). Surface reconstruction from unstructured 3d points. *Computer Graphics Forum*. 15(1):47-60.

[Amenta and Bern, 1999a] Amenta, N. and Bern, M. (1999a). A new voronoi based surface reconstruction algorithm. *Symposium on Computation Geometry*. 19(23):127-153.

[Amenta and Bern, 1999b] Amenta, N. and Bern, M. (1999b). Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry*, 22:481–504.

[Amenta et al., 1998] Amenta, N., Bern, M., and Eppstein, D. (1998). The crust and the $\beta-$skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135.

[Amenta et al., 2002] Amenta, N., Choi, S., Dey, T. K., and Leekha, N. (2002). A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*. 12(12):125-141.

[Amenta et al., 2001a] Amenta, N., Choi, S., and K., K. R. (2001a). The power crust. *Em 6th ACM Symposium on Solid Modeling*. pag. 249-260.

[Amenta et al., 2001b] Amenta, N., Choi, S., and Kolluri, R. (2001b). The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*. 19(23):127-153.

[Attali, 1998] Attali, D. (1998). *r-regular* shape reconstruction from unorganized points. *Computational Geometry Theory and Applications*. 10:239-249.Elsevier.

[Attene and Spagnuolo, 2000] Attene, M. and Spagnuolo, M. (2000). Automatic surface reconstruction from point sets in space. *Computer Graphics Forum*. 19(3):9.

[Bajaj et al., 1995] Bajaj, C., Bernardini, F., and Xu, G. (1995). Automatic reconstruction of surfaces and scalar fields from 3d scans. *SIGGRAPH 95 Proceedings*. pag. 109-118.

[Bardinet et al., 1998] Bardinet, E., Cohen, L. D., and Auache, N. (1998). A parametric deformable model to fit unstructured 3d data. *Computer Vision and Image Understanding*. 71(1):39-54.

[Berg et al., 2000] Berg, M., Kreveld, M., and Pvermars, M. (2000). *Computational Geometry, algorithms and applications*. Springer.

[Bernardini and Bajaj, 1997] Bernardini, F. and Bajaj, C. L. (1997). Sampling and reconstructing manifolds using alphashapes. *Em Proc. 9th Canadian Conf. Computational Geometry*. pag. 193-198.

[Bernardini et al., 1999] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., and Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization andComputer Graphics*. 5(4), pág. 349–359.

[Boissonnat and Cazals, 2002] Boissonnat, J.-D. and Cazals, F. (2002). Smooth surface reconstruction via natural neighbour interpolations of distance functions. *Computational Geometry Theory and Applications*. 22:185-203.

[Carr et al., 2001] Carr, J., Beatson, R., Cherrie, J., Mitchell, T., Fright, W., McCallum, B., and et al. (2001). Reconstruction and representation of 3d objects with radial basis functions. *SIGGRAPH01*. pag. 67-76.

[CGAL, 2005] CGAL (2005). The CGAL home page. `http://www.cgal.org`. Acessado em Setembro, 2005.

[Cosmo, 2005] Cosmo (2005). The cosmo player home page. `http://cic.nist.gov/vrml/cosmoplayer.html`. Acessado em Setembro, 2005.

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. *Em In Proceedings of ACM Siggraph*. pag. 303-312.

[Devillers, 1998] Devillers, O. (1998). Improved incremental randomized delaunay triangulation. *In Proc. 14th Annu. ACM Sympos. Comput. Geom.* pages 106-115.

[Dey et al., 2001] Dey, T. K., Giesen, J., and Zhao, W. (2001). Robustness issues in surface reconstruction. *Lecture Notes in Computer Science*. 2073:658-663.

[Dey and Goswami, 2002a] Dey, T. K. and Goswami, S. (2002a). Tight cocone : A water-tight surface reconstruction. *Relatório Técnico OSU-CISRC-12/02-TR31*. The Ohio State University.

[Dey and Goswami, 2002b] Dey, T. K. and Goswami, S. (2002b). Tight cocone and cgal. *Em CGAL workshop, Barcelona,Espanha.*

[Edelsbrunner, 2001] Edelsbrunner, H. (2001). *Geometry and Topology for Mesh Generation*. Cambridge.

[Edelsbrunner, 2002] Edelsbrunner, H. (2002). Surface recosntruction by wrapping finite point set in space. *Rick Pollack e Eli Goodman Festschrift, ed. A. Aronov, S. Basu, J.Pach e M. Sharir.Springer- Verlag, submetido*.

[Edelsbrunner et al., 1983] Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*. 29:551–559.

[Edelsbrunner and Mücke, 1994] Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*. 13:43-72.

[Giesen and John, 2002] Giesen, J. and John, M. (2002). Surface reconstruction based on a dynamical system. *Em Eurographics, volume Drettakins, G. e Siedel, H. ditores*. pag. 21–30.

[Giesen and John, 2003] Giesen, J. and John, M. (2003). The flow complex: a data structure for geometric modeling. *Em Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. pag. 285-294. Society for Industrial and Applied Mathematics.

[Gois, 2004] Gois, J. P. (2004). Reconstrução de superfícies a partir de nuvens de pontos. *Dissertação de Mestrado, Universidade de São Paulo - Instituto de Ciências Matemáticas e Computação*.

[Gopi et al., 2000] Gopi, M., Krishnan, S., and Silva, C. T. (2000). Surface reconstruction based on lower dimensional localized delaunay triangulation. *Em Gross, M. e Hopgood, F. R. A., editores,Computer Graphics Forum (Eurographics 2000)*. volume 19(3).

[Hoppe et al., 1992] Hoppe, H., Derose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized point clouds. *Em Proceedings of ACM Siggraph*. pag. 71-78.

[Huang and Menq, 2002] Huang, J. and Menq, C.-H. (2002). Combinatorial manifold mesh reconstruction and optimizationfrom unorganized points with arbitrary topology. *Computer Aided Design*. 34(2), pág. 149–165.

[Kirkpatric and Radke, 1985] Kirkpatric, D. G. and Radke, J. D. (1985). Framework for computational morphology - computational geometry. *Computational Methods Applied Mechannical Enginnering*. pág. 217-245.

[Kuo and Yau, 2005] Kuo, C. and Yau, H. (2005). A delaunay-based region-growing approach to surface reconstruction from unorganized points. *Computer-Aided Design*. Volume 37, edição 8, pág. 825–835.

[Mederos et al., 2003] Mederos, B., Velho, L., and de Figueiredo, L. H. (2003). Moving least squares multiresolution surface approximation. *Em SIBGRAPI 2003*. 1:1926, São Carlos,Brasil.

[Milnor, 1963] Milnor, J. (1963). Morse theory. *Princenton University Press. Annals Mathematics Studies.*

[Ohtake et al., 2003] Ohtake, Y. end Belyaev, A., Alexa, M., Turk, G., and Seidel, H. (2003). Multi-level partition of unity implicits. *ACM Trans. Graph.* 22(3) pág. 463–470.

[Rodriguez et al., 1994] Rodriguez, A., Espadero, J., and López, D. (1994). Delaunay surface reconstruction from scattered points. *Em 9th International Conference Discrete Geometry forComputer Imagery, DGCI*.

[Teichmann and Capps, 1998] Teichmann, M. and Capps, M. (1998). Surface reconstruction with anisotropic density-scaled alpha shapes. *Em Ebert D., Hagen H., e Rushmeier H., IEEE Visualization 98*. pag. 67-72.

[VRML, 2005] VRML (2005). The vrml specification. `http://www.graphcomp.com/info/specs/sgi/vrml/spec/`. Acessado em Setembro, 2005.

[Zhao et al., 2000] Zhao, H., Merriman, B., Osher, S., and Kang, M. (2000). Implicit nonparametric shape reconstruction from unorganized points using a variational level set method. *Computer Visionand Image Understanding*. 60:295-313.