Instituto Nacional de Matemática Pura e Aplicada

DISSERTAÇÃO DE MESTRADO

Algoritmo *Ball-Pivoting*: Contextualização e Estado da Arte

Esdras Soares de Medeiros Filho
Orientação: Luiz Velho

12 de Agosto de 2003

Resumo: A **Fotografia 3D** tem recebido contribuições de diversas áreas da Computação Gráfica. Dentre essas constribuições, destacamos o algoritmo **Ball-Pivoting** para reconstrução de superfícies usando pontos amostrados a partir de objetos do mundo real. Os princípios matemáticos da teoria de **Handlebody** estão incorporados nos principais eventos topológicos do algoritmo *Ball-Pivoting*. Esses princípios justificam a sua efetividade. Eventualmente, por problemas de amostragem, surgem buracos na reconstrução da superfície. Em geral, fechar buracos é um problema difícil de resolver. Nesta dissertação, um algoritmo denominado de **Quick Fill** foi proposto aproveitando-se da estrutura gerada pela malha do *Ball-Pivoting* a fim de fechar buracos de topologia simples.

Abstract: **3D Photography** has received contributions from many Computer Graphics areas. Among these contributions we stand out the **Ball-Pivoting** algorithm for surface reconstruction from sampled points. The mathematical principles of the **Handlebody** theory, which are embedded in the main topological events of the Ball-Pivoting algorithm, explain its effectiveness. Sometimes, because of sampling problems, holes may appear in the reconstructed surface. Generally, closing holes is a difficult problem to solve. In this dissertation, an algorithm called **Quick Fill** was proposed to close topologically simple holes taking advantage of the structure generated by the Ball-Pivoting mesh.

AGRADECIMENTOS

Agradeço primeiramente à Deus pela dom da vida e por Seu amor que sempre me motivou a trabalhar.

Agradeço aos meus pais e familiares que sempre estiveram do meu lado.

Agradeço à Comunidade Católica Shalom do Rio de Janeiro que, através do apoio de membros como João Edson e Sidney Jorge Timbó, deram-me todo apoio nos anos que estive cursando o mestrado.

Agradeço ao meu orientador, Luiz Velho, por sua experiência, paciência e dedicação.

Agradeço ao Prof. Paulo Cézar pelos cursos de Geometria Computacional e Fotografia 3D que me inspiraram na elaboração da dissertação.

Agradeço ao Luiz Henrique pela revisão e sugestões na dissertação.

Agradeço aos meus colegas do Visgraf pelo companheirismo e tempo que passamos juntos convivendo e trabalhando.

Agradeço aos Professores Hélio Lopes e ao Sinésio Pesco pela cortesia em ceder algumas ilustrações e por contribuirem para a melhoria da dissertação.

Por fim, quero agradecer ao IMPA por sua excelente estrutura de pesquisa e ao CNPq por todo apoio financeiro.

Aos meus queridos pais,
Esdras e Rita.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Introduziremos alguns conceitos de Fotografia 3D, área que motivou o estudo da reconstrução de superfícies, e discutiremos os objetivos e as principais contribuições da dissertação.

1.1 Motivação

Fotografia 3D é o processo de captura da geometria de objetos do mundo real por meio de câmeras e fontes de luz (canhão de laser, luz estruturada, etc.). É uma das áreas que atualmente está em crescente desenvolvimento graças à modernização dos dispositivos que capturam a geometria dos objetos (*scanners*). Também envolve várias áreas da Computação Gráfica, atraindo pesquisadores das mais diversas especialidades. Dentre as muitas aplicações da Fotografia 3D, merecem destaque as seguintes:

- **Artes:** Arquivamento de obras de arte feitas por artistas famosos num banco de dados na *internet* para facilitar o acesso às pessoas de todo o mundo por meio digital.
- **Medicina:** Dimensões precisas do corpo humano de um paciente permitem a fabricação de próteses personalizadas com maior exatidão e o melhor planejamento de cirurgias;
- **Entretenimento:** Inserção de objetos reais em ambientes virtuais ou em filmes.

O processo completo da Fotografia 3D contém, em geral, 4 etapas. Pela ordem de execução temos: digitalização, registro de dados, integração de dados e conversão de modelo. Vamos agora descrever resumidamente cada etapa.

A primeira etapa, digitalização, é de nível baixo, ou seja, depende principalmete do dispositivo de leitura. Em geral, o modelo de captura usa uma câmera e um projetor de padrões (cf. fig1.1). Tal modelo é classificado como *stereo ativo*.

A etapa de digitalização compreende as seguintes atividades:

- **Calibração:** determina os parâmetros intrínsecos e extrínsecos da câmera e do projetor;
- **Decodificação:** estabelece correspondências entre pontos da imagem e linhas projetadas no objeto;
- **Triangulação**: reconstrói as coordenadas tridimensionais do objeto na cena.

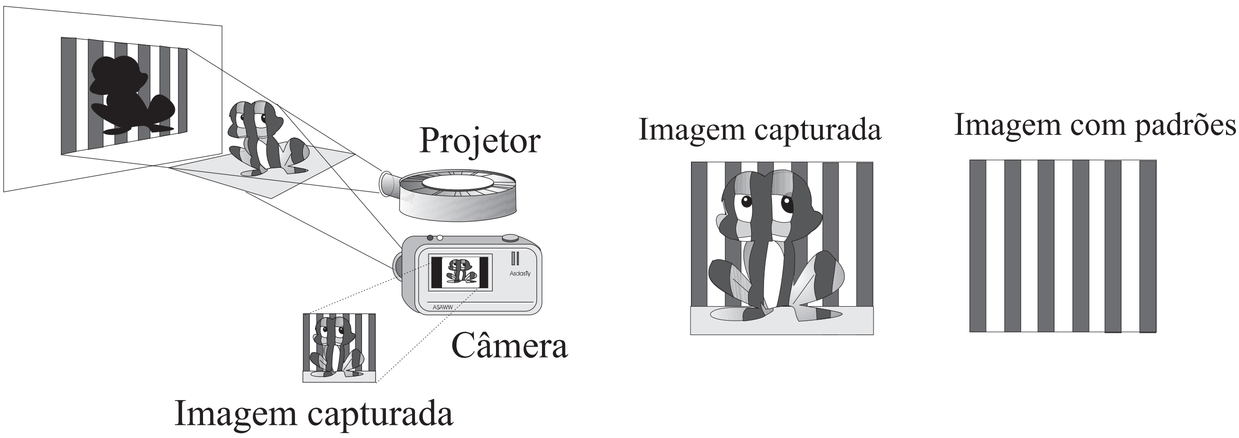


Figura 1.1: Esquema do processo de digitalização usando stereo ativo com luz estruturada. Cortesia de A. Peixoto.

Na etapa de registro de dados, nuvens de pontos das imagens são alinhadas afim de gerar uma só. Para isso, o algoritmo *Iterative Closest Point* (ICP) [6] , que sempre converge para um mínimo local segundo a norma convencional, é o mais adequado. Inclui também o registro global, que é a minimização do erro de todas as nuvens alinhadas.

Na etapa de integração de dados, as amostras são interpoladas em uma malha poligonal, tipo de representação mais comum para os dados amostrados de uma superfície.

Na última etapa, conversão de modelo, a malha é simplificada e otimizada para fins de visualização, modelagem, compressão, etc.

No processo de Fotografia 3D o *Ball-Pivoting algorithm* (BPA)[5] tem-se destacado durante a etapa de integração de dados principalmente por sua elegância e simplicidade. Dentre as mais importantes contribuições para reconstrução de superfícies, podemos dizer que o BPA:

- Gera uma malha que é um subconjunto do *Alpha Shapes*, herdando assim, suas propriedades;
- Provou ser um algoritmo apropriado para dados em grande escala porque ele é bastante eficiente em termos de tempo de execução e em armazenamento;
- É robusto quando se depara com ruídos.

1.2 Objetivos

Nesta dissertação focaremos nosso estudo no BPA. Resumidamente, o nosso objetivo é descrever o BPA e as propriedades da nuvem de pontos capazes de tornar melhor e mais rápido seu funcionamento. Para isso, primeiramente definiremos o problema de reconstrução de superfícies a partir de amostras e falaremos sobre o *Alpha Shapes*[8]. Depois, veremos a parte geométrica do BPA e a fase de construção da malha, essência do BPA. Finalmente, discutiremos as condições ideais da amostragem para o bom funcionamento do BPA.

Em locais onde a amostragem é insuficiente o BPA gera buracos. Nesta dissertação iremos também abordar outro problema que é o de fechar buracos. Apresentaremos uma simples heurística, que chamaremos de *Quick Fill*, capaz de fechar buracos que são topologicamente iguais a um disco.

1.3 Principais Contribuições

Ao implementarmos o BPA passamos por várias versões até conseguirmos gerar todas as poligonais dos buracos e depois fechá-los. Sabíamos que a malha estava sendo gerada visualmente de maneira correta mas havia inconsistências nos índices (cf. seção 4.2) de alguns pontos impossibilitando-nos de gerar tais poligonais dos buracos.

Nesta dissertação fizemos uma diferenciação entre **pontos** e **vértices** que nos ajudou a resolver esses problemas de inconsistências. Nas duas principais rotinas do BPA que são, junção e colagem, explicamos o processo de formação da malha com a ajuda da teoria de *Handlebody*[17]. Na verdade, essa dualidade entre pontos e vértices foi o que nos possibilitou usar a teoria de Handlebody porque conseguimos detectar na junção e colagem os instantes em que os *handles* são associados à malha.

1.4 Esquema da Dissertação

O restante da dissertação está dividida da seguinte forma:

- **Capítulo 2:** Neste capítulo enunciamos o problema de Reconstrução de Superfícies e classificamos os métodos de Reconstrução. Também definimos o *Alpha Shapes* e descrevemos suas propriedades.
- **Capítulo 3:** O BPA é um algoritmo cujo passo-principal é um giro de uma "bolinha" em torno de uma aresta da malha. Descrevemos neste capítulo a geometria desse passo e a estrutura de busca espacial do BPA para efetuá-lo.
- **Capítulo 4:** O pseudo-código do BPA é essencialmente um processo de construção de malha. Descrevemos neste capítulo as estruturas de dados para a malha e as rotinas do BPA. No final, mostramos que a complexidade do BPA é linear.
- **Capítulo 5:** Comentamos condições ideais da amostragem para o BPA e implementações de simplificação de pontos.
- **Capítulo 6:** Abordamos o problema de fechamento de buracos, descrevemos um algoritmo que fecha polígonos simples tridimensionais. Mostramos resultados de desempenho do BPA e do algoritmo de fechar buracos.
- **Capítulo 7:** Neste capítulo comentamos os resultados do BPA, suas contribuições e deixamos algumas propostas para trabalhos futuros.

Capítulo 2

Reconstrução de Superfícies

Neste capítulo abordamos o problema de reconstrução de superfícies a partir de amostras e introduzimos o conceito de *Alpha Shapes*[8]. Faremos também uma introdução intuitiva sobre o BPA.

2.1 Paradigmas de Abstração

Os objetos de estudo da computação gráfica podem ser modelados, em geral, utilizando o paradigma dos quatro universos, conceito introduzido por Gomes & Velho [10]. Veja na figura abaixo um esboço desses universos:

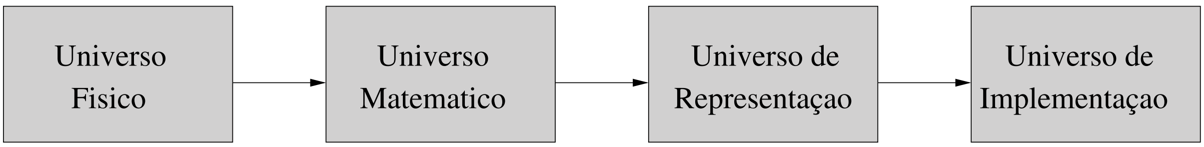


Figura 2.1: Níveis de abstração.

Ao procurarmos representar um objeto do universo matemático para o universo de representação, podemos ter uma perda de informação. Isto acontece porque, em geral, queremos passar de um modelo contínuo para um modelo discreto.

Em Fotografia 3D, por exemplo, desejamos representar um objeto do mundo físico no computador. No universo matemático tal objeto é descrito como uma superfície compacta mergulhada no $\mathbb{R}^3$. No universo de representação o objeto consiste em uma nuvem de pontos. Daí, quando queremos voltar para o modelo matemático de representação, que é o nosso objetivo, damos o nome de reconstrução. Se esta for igual à superfície original, dizemos que a reconstução é exata. Tais representações são, no entanto, muito difíceis de conseguir.

2.2 Definição do Problema

O problema que vamos ressaltar nesta sessão é, de modo geral, enunciado da seguinte maneira: dado um conjunto finito de pontos $P \subset \mathbb{R}^3$ gerar uma superfície S tal que $P \subset S$ ou $max\{\ ||p - S|| : p \in P\ \}$ é suficientemente pequeno.

Esta formulação acima não é muito precisa porque dá espaço a diversas interpretações e é mais classificado como um problema de construção de superfícies. Mas se conhecermos algo sobre a origem da nuvem de pontos P (e.g. o seu modelo real) e nosso objetivo é obter uma descrição exata desta superfície no computador, deparamo-nos com o problema de recontrução de superfícies (fig. 2.2). O conhecimento sobre a superfície é importante na escolha do melhor algoritmo de reconstrução.

Figura 2.2: Superfície amostrada e recosntruída. Cortesia de A. Peixoto.

A maioria dos métodos que propõem uma solução requerem pontos "suficientemente densos" para gerar a superfície desejada. Tal suficiência, conhecida como condição de amostragem, deve garantir uma boa[1] aproximação em relação à superfície original. Por exemplo, em Amenta et. al [1] é proposto o *power crust*, método baseado numa aproximação do MAT (Medial Axis Transform). É definida a característica local (LFS) de um ponto $p \in S$ como a distância ao ponto mais próximo do eixo medial de S. Verifica-se então que se $P \subseteq S$ é uma r-amostra com $(r \leq 0.1)$, ou seja, a distância de qualquer ponto $p \in S$ ao ponto mais próximo da amostra P é no máximo $r \cdot LFS(p)$ então o método nos dá uma boa reconstrução. Intuitivamente, para curvas planares, isso exige que, em locais onde a curvatura é grande, deve haver um maior coeficiente de amostragem. No capítulo 5 faremos um estudo semelhante para impor condições ideais para o BPA gerar uma boa aproximação.

Um dos desafios do problema de reconstrução de superfícies está na busca de um método que cubra, de acordo com a necessidade da aplicação a ser considerada, o maior número de formas possível além de ter que resolver dados ruidosos.

[1] Quando a superfície reconstruída é homeomorfa e suficiente próxima.

2.3 Modelo de Representação

Sabemos que a superfície reconstruída deve estar armazenada de alguma forma no computador. A representação estrutural mais comum é a representação triangular porque nos permite o uso de métodos simpliciais e de combinatória, principalmente em sub-problemas de reconstrução como, por exemplo, o fechamento de curvas de bordo e simplificação.

2.4 Classes de Algoritmos

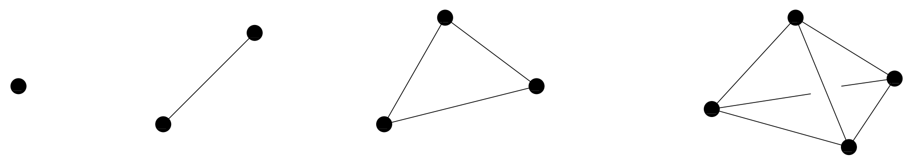
Mencl & Muller [16] descrevem a existência de quatro classes de algoritmos de reconstrução. São elas: algoritmos baseados em subdivisão espacial (cf. Boissonant [7]) , funcões distância (cf. Hope [11]), deformação (cf. Osher et al. [18]) , e crescimento incremental de superfícies, sendo esta última a classe em que o BPA está incluído. A seguir, afim de entendermos melhor a malha gerada pelo BPA, falaremos sobre um algoritmo de subdivisão espacial que gera o *Alpha Shapes*[8] de um conjunto de pontos no espaço a partir da triangulação de Delaunay.

2.5 *Alpha Shapes*

Façamos algumas definições prévias. Como antes, considere $P \subset \mathbb{R}^3$ um conjunto finito de pontos. Seja $T \subset P$ um subconjunto tal que $\mid T \mid= k+1$ com $0 \leq k \leq 3$, em posição geral [2]

Definição 1 *k-simplexo é o fecho convexo de T ou $conv(T)$. Representaremos um k-simplexo por σ_T. Ver figura 2.3.*

Definição 2 *Uma α-bola é uma bola aberta de raio α. Dizemos que um k-simplexo σ_T $1 \leq k \leq 2$ está α-exposto se existir uma α-bola vazia e circunscrita a T.*



(a) (b) (c) (d)

Figura 2.3: Tipos de k-simplexo. (a) 0-simplexo é um ponto (b) 1-simplexo é uma aresta (c) 2-simplexo é um triângulo (d) 3-simplexo é um tetraedro.

[2] n+1 pontos no $\mathbb{R}^n$ que não pertencem ao mesmo (n-1)-dimensional hiperplano.

Definição 3 *Seja* $F_{k,\alpha} = \{k\text{-simplexos de } P\ \alpha\text{-expostos}\}$*. O* α*-shape de* P*, denotado por* $S_\alpha(P)$ *é o politopo cujo bordo é formado por todos os triângulos em* $F_{2,\alpha}$*, arestas em* $F_{1,\alpha}$ *e vértices em* $F_{0,\alpha}$*, ou seja,* $\partial S_\alpha(P) = F_{2,\alpha} \cup F_{1,\alpha} \cup F_{0,\alpha}$*.*

Não é fácil ver que existe um politopo Q tal que $\partial Q = \partial S_\alpha(P)$. Por isso, vamos agora dar uma outra definição para α-shapes envolvendo complexos simpliciais e triangulação de Delaunay.

Definição 4 *Complexo simplicial* C *de um conjunto de pontos* P *é uma coleção finita de* k*-simplexos que satisfazem as seguintes propriedades:*

(a) *Se* $\sigma_T \in C$ *então para todo* $T' \subseteq T$ *temos que* $\sigma_{T'} \in C$*. Dizemos então que* $\sigma_{T'}$ *é uma face de* σ_T *quando* $T' \subseteq T$

(b) *Se* $\sigma_T, \sigma_{T'} \in C$ *então* $\sigma_T \cap \sigma_{T'} = \emptyset$ *ou* $\sigma_T \cap \sigma_{T'} = \sigma_{T \cap T'} = conv(T \cap T')$*.*

Definição 5 *Para* $0 \leq k \leq 3$ *seja* F_k *o conjunto dos* k*-simplexos* $\sigma_T = conv(T)$*,* $T \subset P$ *e* $|T| = k + 1$ *para os quais existem bolas abertas* b *tais que* $\partial b \cap P = T$*. A triangulação de Delaunay de* P*, denotada por* $DT(P)$ *é o complexo simplicial definido por* $DT(P) = F_3 \cup F_2 \cup F_1 \cup F_0$*.*

Proposição 1 *Para* $0 \leq k \leq 2$ *é válida a seguinte relação entre* $DT(P)$ *e o bordo de* $S_\alpha(P)$*:*

$$F_k = \bigcup_{0 \leq \alpha \leq \infty} F_{k,\alpha}$$

Demonstração. (Inclusão) Por definição, para cada simplexo $\sigma_T \in DT(P)$ existem valores de $\alpha \geq 0$ tais que σ_T-exposto. (Continência) Cada face de $S_\alpha(P)$ é um simplexo de $DT(P)$ ■

Sejam, b_T e μ_T a menor esfera que contém os pontos de T e seu raio respectivamente.

Definição 6 *Seja* $0 \leq \alpha \leq \infty$*. O* α*-complexo* $C_\alpha(P)$ *é o sub-complexo simplicial de* *DT(P) onde cada k-simplexo* σ_T *está em* $C_\alpha(P)$ *se satifizer as seguintes condições:*

(a) *Se* $\mu_T \leq \alpha$ *e* $b_T \cap P = \emptyset$*.*

(b) σ_T *é face de algum k-simplexo que satisfizer (a).*

Note que a condição (b) da definição 6 nos garante que $C_\alpha(P)$ é um complexo-simplicial.

Finalmente chegamos a uma relação entre *Alpha Shapes* e *Alpha Complexos* que enunciamos aqui sem demonstração:

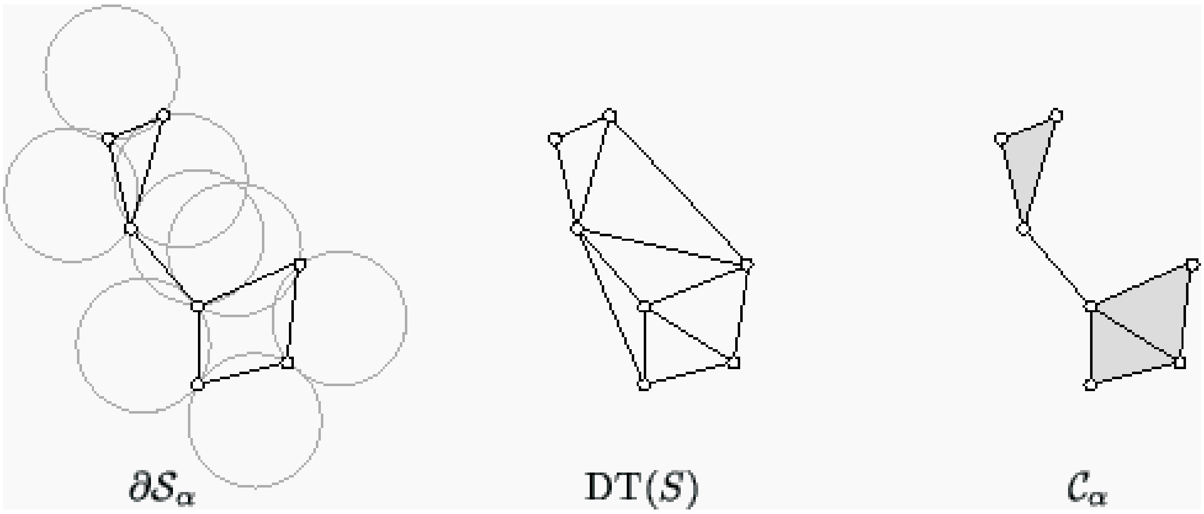


Figura 2.4: Alpha Shapes, Triangulação de Delaunay e Alpha Complexo

Proposição 2 $\partial C_\alpha(P) = \partial S_\alpha(P)$.

Sabendo que o complexo simplicial é um politopo, agora fica claro porque que definimos *Alpha Shapes* como o bordo de um politopo. Temos então outra definição para *Alpha Shapes* que é $S_\alpha(P) := ||C_\alpha(P)||$, onde $||X||$ é o sub-espaço do $\mathbb{R}^3$ coberto pelo complexo simplicial X.

Usando as definições e resultados descritos anteriormente formularemos um algoritmo para encontrar o *Alpha Shapes* de P.

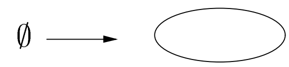
- Calcular $DT(P)$;
- Encontrar $C_\alpha(P)$ incluindo todos os simplexos $\sigma_T \in DT(P)$ tais que $\mu_T < \alpha$ juntamente com suas faces;
- Retornar todos os simplexos de $\partial C_\alpha(P)$.

Em [8] Edelsbrunner faz mais que isso: para cada k-simplexo $\sigma_T \subset DT(P)$ determina um intervalo para o qual σ_T é face de $S_\alpha(P)$ se e somente se α estiver contido nesse intervalo. Isso tem a vantagem de permitir variar α interativamente sem ter que recalcular $C_\alpha(P)$. Não obstante, a triangulação de Delaunay é cara (para $\mathbb{R}^n, n \geq 3$) tanto em custo de memória como em instabilidades numéricas principalmente quando envolve grande quantidade de dados.

2.6 O *Ball-Pivoting* Intuitivo

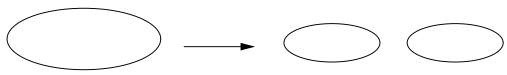
O BPA é um algoritmo de expansão de frente (*advancing-front* [14, 9]). Esta frente é formada essencialmente por curvas poligonais de fronteira. Inicialmente a frente é um ciclo composto apenas pelas arestas de um triângulo que pertence à malha e que foi gerado por um algoritmo que denominaremos de primeira-escolha. Novos triângulos são associados à frente por um passo chamado de passo-principal. Este passo consiste em girar uma bolinha de raio fixo, que está circunscrita a um triângulo da malha, em torno de uma das arestas da frente até tocar um novo ponto da amostragem (cf. fig 2.5). A frente muda de configuração acrescentando-se duas novas arestas. Estas novas arestas são "coladas" às suas respectivas arestas opostas, mudando a topologia da frente. Em geral, as mudanças de topologia são as seguintes:

1. Criação de uma curva:

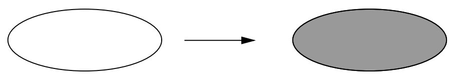


2. Duas curvas se juntam formando uma só:

3. Uma curva divide-se em duas:



4. Uma curva é fechada:



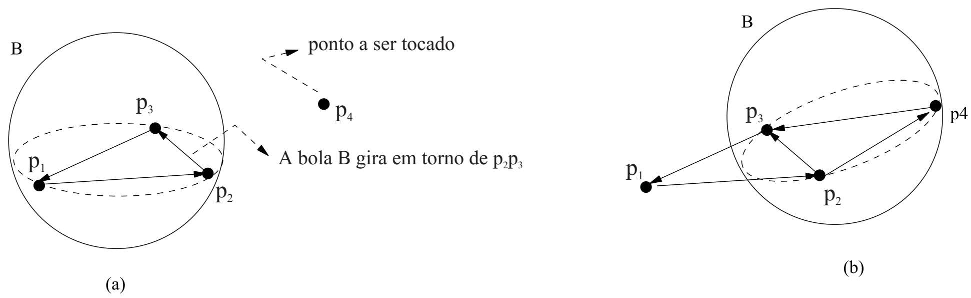


Figura 2.5: *Ball-Pivoting* intuitivo. Inicialmente a frente é formada pela poligonal $p_1p_2p_3$ (a). Após o passo-principal a frente muda para a poligonal $p_1p_2p_3p_4$ (b).

Capítulo 3

A Geometria

Neste capítulo vamos descrever a parte geométrica do funcionamento do algoritmo Ball-Pivoting. Discutiremos o passo principal e a estrutura de dados de busca feita com uma subdivisão espacial (voxels).

3.1 Triângulos orientados e consistentes

Vamos acrescentar atributos ao conjunto $P \subset \mathbb{R}^3$. Como estes pontos pertencem a uma superfície S vamos considerar agora como dados de entrada $\tilde{P} = \{(p,n) \in \mathbb{R}^3 \times \mathbb{R}^3 \ \mid \ p \in P \text{ e } n \text{ é normal a } S \text{ em } p\}$. No entanto, um vetor n que aponte "fora" da superfície S é suficiente.

Seja $T = \{p_1, p_2, p_3\}, p_i \in \tilde{P}$. O 2-simplexo σ_T tem duas orientações, que são $p_1p_2p_3$ ou $p_3p_2p_1$ e que são associadas às duas normais opostas $(p_2-p_1)\wedge(p_3-p_2)$ e $(p_2-p_3)\wedge(p_1-p_2)$ respectivamente. Vamos definir triângulo orientado Δ_T como o 2-simplexo σ_T com uma orientação definida. Dizemos que Δ_T é consistente quando $\langle n_i, N\rangle > 0, \forall i$, onde N é a normal de Δ_T.

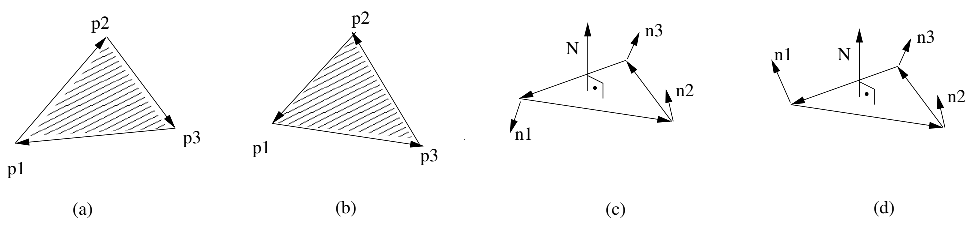


Figura 3.1: (a) e (b) são orientações possíveis, (c) e (d) triângulo não-consistente e consistente respectivamente.

3.2 Triângulo α-exposto

Como vimos no capítulo anterior, um triângulo Δ_T α-exposto é aquele que possui uma α-bola circunscrita e vazia. Sabemos que todo triângulo possui no máximo duas α-bolas circunscritas. Mas como o triângulo está orientado então podemos escolher a α-bola, que iremos denominar por $B_\alpha(\Delta_T)$, cujo centro está no semi-espaço positivo determinado pelo plano suporte do triângulo. Veremos mais adiante que a malha construída pelo algoritmo é formada por triângulos orientados com suas respectivas α-bolas circunscritas.

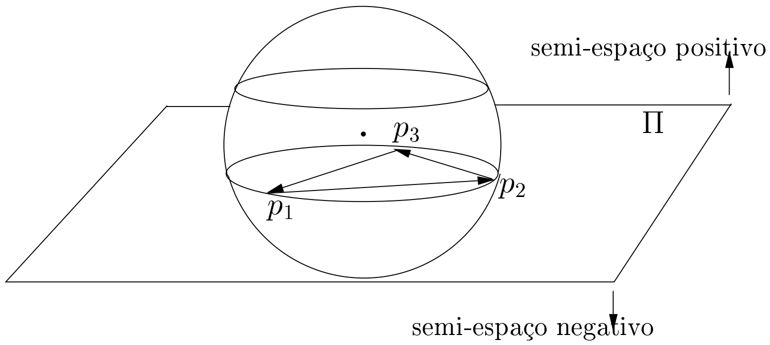


Figura 3.2: semi-espaços positivo e negativo

3.3 Determinando o centro de $B_\alpha(\Delta_T)$

Usaremos geometria euclidiana para encontrarmos o centro de $B_\alpha(\sigma_T)$.

Lema 1 *Sejam $\Delta_T = p_1p_2p_3$ um triângulo orientado e Π seu plano suporte, então $\forall p \in \Pi$ existem coeficientes $\lambda_1, \lambda_2, \lambda_3 \in R$ tais que $\lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3 = p$ com $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Além disso, esses coeficientes são invariantes por transformaçoes afins e $\lambda_1 = \frac{S_{pp_2p_3}}{S_{p_1p_2p_3}}, \lambda_2 = \frac{S_{pp_3p_1}}{S_{p_1p_2p_3}}$ e $\lambda_3 = \frac{S_{pp_1p_2}}{S_{p_1p_2p_3}}$, onde S_{abc} é a área do triângulo orientado abc.*

Demonstração: Π é parametrizado como $p = p_1 + \lambda_2(p_2 - p_1) + \lambda_3(p_3 - p_1) = (1 - \lambda_2 - \lambda_3)p_1 + \lambda_2(p_2) + \lambda_3(p_3)$. Fazendo $\lambda_1 = 1 - \lambda_2 - \lambda_3$, temos que $\lambda_1 + \lambda_2 + \lambda_3 = 1$ e provamos a primeira parte.

Seja $T(p) = A(p) + t_0$ uma tranformção afim, com A linear e t_0 um vetor fixo. Logo:

$$\begin{aligned}
T(\lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3) &= A(\lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3) + t_0 \\
&= \lambda_1 A(p_1) + \lambda_2 A(p_2) + \lambda_3 A(p_3) + t_0 \\
&= \lambda_1(A(p_1) + t_0) + \lambda_2(A(p_2) + t_0) + \lambda_3(A(p_3) + t_0) \\
&= \lambda_1 T(p_1) + \lambda_2 T(p_2) + \lambda_3 T(p_3)
\end{aligned}$$

provando a segunda parte. Considerando uma isometria $I(p) = p'$ (rotação + translação) que leva Π em $z = 1$ e utilizando o resultado da primeira parte, encontramos λ_i solucionando o seguinte sistema linear,

$$\begin{pmatrix} p'_1 & p'_2 & p'_3 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = p'$$

e cuja solução é $\lambda_1 = \frac{|p'p'_2p'_3|}{|p'_1p'_2p'_3|} = \frac{S_{pp_2p_3}}{S_{p_1p_2p_3}}$. Obtemos analogamente λ_2 e λ_3 conforme o enunciado ■

Seja C o centro de $B_\alpha(\Delta_T)$ e sejam d, O e $\vec{N}$ a distância de C ao plano suporte de Δ_T , o seu circuncentro e sua normal unitária, respectivamente (fig 3.3).

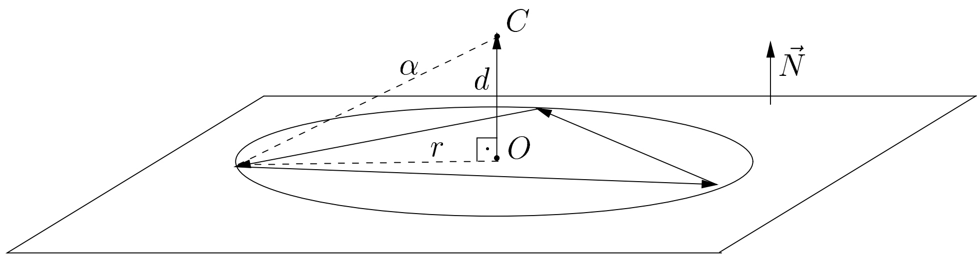


Figura 3.3: teorema de pitágoras

- Pelo Teorema de Pitágoras $d = \sqrt{r^2 - \alpha^2}$, onde $r = \frac{|a||b||c|}{4S_{p_1p_2p_3}}$ é o raio do círculo circunscrito de Δ_T e $a = (p_2 - p_1), b = (p_3 - p_2), c = (p_1 - p_3)$ são os seus lados;
- $\vec{N} = \frac{a \times b}{|a \times b|}$;
- Pela proposição 1, $O = \lambda_1 p_1 + \lambda_2 p_2 + \lambda_3 p_3$ e $\lambda_1 = \langle \frac{a \times (O - p_3)}{|a \times b|}, \vec{N} \rangle$, $\lambda_2 = \langle \frac{a \times (O - p_1)}{|a \times b|}, \vec{N} \rangle$, $\lambda_3 = \langle \frac{a \times (O - p_2)}{|a \times b|}, \vec{N} \rangle$;

Tendo $d, \vec{N}$ e O determinados, concluímos então que $C = O + d\vec{N}$ e obtemos a seguinte proposição.

Proposição 3 *As coordenadas do centro de $B_\alpha(\Delta_T)$ são completamente determinadas pelas coordenadas dos pontos de Δ_T e sua orientação.*

3.4 Subdivisão Espacial Uniforme

Uma das características do algoritmo é o fato de ser local e para tanto, fazemos uma subdivisão uniforme do espaço em voxels de lado 2α. Em termos de implementação estes voxels são armazenados numa matriz tridimensional porque é bastante cômodo identificar os índices da matriz de um ponto $p \in \tilde{P}$ fazendo uma conta simples. A seguir faremos em detalhes este cálculo para o caso 2D que é análogo ao 3D.

Sejam $(x_0, y_0), (x_1, y_1)$ o canto inferior esquerdo e o canto superior direito da caixa envoltória B dos pontos de $\tilde{P}$, respectivamente. O espaço que queremos dividir de maneira uniforme é um retângulo R que tem as seguintes propriedades:

- O canto inferior esquerdo é (x_0, y_0)
- $B \subset R$
- O comprimento é $n(2\alpha)$ e a altura é $m(2\alpha)$

onde $m = \lfloor \frac{y_1 - y_0}{2\alpha} \rfloor + 1$ e $n = \lfloor \frac{x_1 - x_0}{2\alpha} \rfloor + 1$. R é interpretado como uma matriz de m linhas e n colunas e seus elementos são identificados por R_{ij}.

Seja $p = (x, y) \in R$. Então $p \in R_{ij}$ onde $i = \lfloor \frac{y - y_0}{2\alpha} \rfloor + 1$ e $j = \lfloor \frac{x - x_0}{2\alpha} \rfloor + 1$. Este cálculo que identifica a célula R_{ij} a que p pertence é a principal operação que usaremos na matriz R.

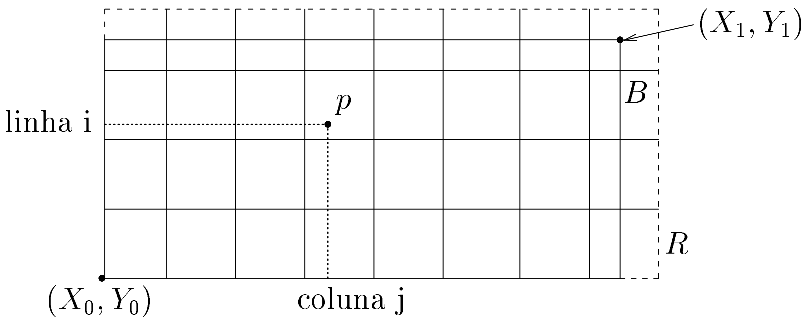


Figura 3.4: Representação da matriz R em que os pontos estarão organizados.

Esta é a estrutura em que os dados de entrada, $\tilde{P}$, estão configurados. Note que é bastante simples de implementar e o tempo gasto para sua montagem é $O(|\tilde{P}|)$.

Definição 7 *M é a estrutura de matriz de voxels que armazena os pontos de $\tilde{P}$ e é capaz de identificar em tempo constante o elemento M_{ijk} a que pertence um ponto do espaço.*

Definição 8 *Seja $p \in |M|$, onde $|M| \subset \mathbb{R}^3$ é o espaço coberto pelos elementos de M. Então $V(p)$ é uma vizinhança de p definida como a união do voxel a qual p pertence e de seus voxels adjacentes.*

3.5 O Passo-principal

No código do BPA (cf. seção 4.2) existe uma rotina de caráter geométrico que denominamos de passo-principal e que vamos descrevê-la agora. Sejam $\Delta_T = p_1p_2p_3$ e C_{123} o centro de $B_\alpha(\Delta_T)$. O passo-principal consiste em girar a esfera $B_\alpha(\Delta_T)$ no sentido do vetor normal de Δ_T em torno de um de seus lados, digamos p_1p_3, até tocar algum outro ponto de $\tilde{P}$.

O primeiro ponto a ser tocado tem o menor ângulo de rotação. Logo, precisamos calcular para todos os pontos de $V(m)$, onde m é o ponto médio de p_3p_1, o menor ângulo formado entre os segmentos mC_{123} e mC_{i13}, sendo C_{i13} o centro de $B_\alpha(\Delta_{p_ip_1p_3})$.

Seja $p_i \in \tilde{P}$. A trajetória descrita por C_{123} é um círculo de centro $m = \frac{p_3+p_1}{2}$ e raio $|C_{123} - m|$. Usando a proposição 1, determinamos o centro C_{i13} do triângulo $p_ip_2p_3$ e através de produto vetorial encontramos o ângulo orientado $C_{123}\hat{m}C_{i13}$ tendo o vetor $\overrightarrow{p_3p_1}$ como referência.

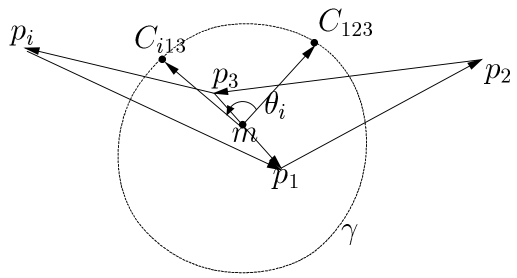


Figura 3.5: Trajetória γ de $B_\alpha(\Delta_T)$ no passo principal.

Algoritmo 1: passo-principal

Input : $(M, B_\alpha(p_1p_2p_3), m, C_{123})$
Output: $p_0 \in V(m)$ tal que $C_{123}\hat{m}C_{031}$ é mínimo

$p_0 \leftarrow NULO$;
$minang \leftarrow 2\pi$;
for *each* $p_i \in V(m)$ **do**
  calcule o centro C_{i13} de $B_\alpha(p_ip_1p_3)$;
  calcule o angulo θ_i entre $\overrightarrow{mC_{123}}$ e $\overrightarrow{mC_{i23}}$;
  if $\theta_i < minang$ **then**
    $p_0 \leftarrow p_i$;
  end
end
return p_0;

Note que a vizinhança é grande o suficiente para conter todas os possíveis pontos, isto é, se $p_0 \notin V(m)$ então $|m - p_0| > 2\alpha$ e não existe $B_\alpha(p_0p_1p_2)$. Assim, o espaçamento 2α do grid M está bem definido. Veja o pseudo-algoritmo 1.

Observação 1 *O ponto p_0 retornado pelo algoritmo acima é tal que $B_\alpha(pp_3p_1)$ é α-exposto.*

3.6 A Primeira-escolha

É necessária uma rotina que encontre algum triângulo Δ_T α-exposto afim de iniciar o processo descrito na seção anterior. O problema que queremos resolver é: dado $\tilde{P}$ em uma matriz M (como descrito na seção 3.4), achar $p_1, p_2, p_3 \in \tilde{P}$ tais que $\Delta_{p_1p_2p_3}$ é α-exposto e consistente.

A nossa estratégia é usar a matriz M testando possíveis triângulos em cada voxel M_{ijk}. Escolhemos um ponto $p \in (\tilde{P} \cap M_{ijk})$ e criamos uma lista de segmentos $\overrightarrow{pp_l}, p_l \in (\tilde{P} \cap M_{ijk})$. Daí, para cada $\overrightarrow{pp_l}, \overrightarrow{pp_m}$ testamos a existência de $B_\alpha(pp_lp_m)$ a consistência de pp_lp_m e verificamos se $B_\alpha(pp_lp_m)$ é α-exposto. Veja abaixo como fica esse algoritmo que denominamos de <u>primeira-escolha</u>:

Algoritmo 2: primeira-escolha

```
for each Mijk do
    p ← primeiro ponto em Mijk ;
    L ← gerar lista de segmentos pp_l, p_l ∈ (P̃ ∩ M_ijk);
    for each pp_l, pp_m ∈ L; l ≠ m do
        if (existe B_α(pp_i p_j) e pp_i p_j é consistente e B_α(pp_i p_j) é α-exposto)
        then
            return pp_i p_j;
        end
    end
end
return NULO;
```

Capítulo 4

Construção de Malha

Neste capítulo discutiremos como a estrutura da malha está definida. Descrevemos o BPA que constrói a malha e as principais rotinas colagem e junção que tratam os eventos topológicos no seu decorrer. Baseado na teoria de *Handlebody*[17] damos um novo enfoque para tais eventos.

4.1 A Malha Topológica

A malha é um grafo $\mathcal{G}(E, V, F)$ direcionado onde E e V são respectivamente arestas e vértices que possuem informações de vizinhança baseada na estrutura *half winged edge*[3]. Também existe o conjunto de faces F, que são triangulares.

Os eventos que acontecem na malha são de natureza topológica / geométrica(fig 4.1). Para evitar ambiguidades e facilitar o entendimento do algoritmo criamos duas estruturas que refletem essa dualidade: ponto (natureza geométrica) e vértice (natureza topológica). Existe uma relação entre pontos e vértices $1 : n$ (fig 4.2). Isto nos dá uma boa representação da topologia das curvas de bordo porque poderão existir, eventualmente durante o decorrer do algoritmo, pontos que pertencem a mais de uma curva de bordo.

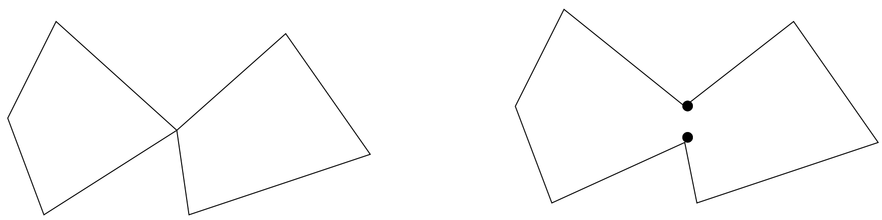


Figura 4.1: Tipos de representações de curvas: à esquerda geométrica e à direita topológica.

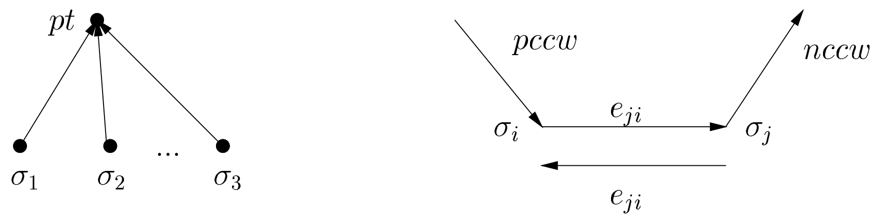


Figura 4.2: Relação ponto e vértice; estrutura *half winged edge*

Definição 9 *Dado $\sigma \in V$, o conjunto das arestas que "entram" e "saem" de um vértice de σ são respectivamente:*

$$i^*(\sigma) = \{e \in E : e_j = \sigma\};\ o^*(\sigma) = \{e \in E : e_i = \sigma\}$$

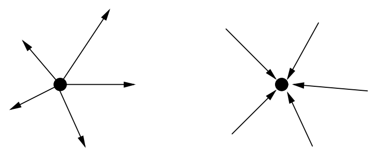


Figura 4.3: i^* e o^*

A **definição 9** é utilizada nas estruturas de dados e tem um importante papel na rotina de "colagem" que será vista mais adiante. A seguir listamos as principais estruturas de dados escritas em código C.

```
struct _POINT {
    vector              p, n; //coordenadas e normal
    int                index; //índice
    list                   v; //lista de vértices
} point;

struct _VERTEX{
    struct point         *pt; //ponto atribuído ao vértice
    list                   i; //lista de arestas que entram
    list                   o; //lista de arestas que saem
    struct edge     *ne, *pe; //arestas próxima e anterior
                              //da curva de bordo
} vertex;
```

```
struct _EDGE  {
    struct vertex   *pi, *pj; //vértices da aresta
    struct point         *go; //ponto oposto
    vector              cijo; //centro da alpha bola
    struct edge         *eji; //aresta oposta
    struct edge *nccw, *pccw; //poxima e anterior
    bool              status; //bordo=true or ativo=false
}  edge;
```

4.2 Algoritmo Ball-Pivoting

Em Bernadini et al. [5] é proposto o seguinte algoritmo:

Algoritmo 3: Ball-Pivoting

while *verdadeiro* **do**
  while ($e_{ij} \leftarrow$ *primeiro elemento de* $\mathcal{F}) \neq NULO$ **do**
    if $p_k \leftarrow$ *passo-principal*(e_{ij}) *e* (*nao-usado*(p_k) *or indice*(p_k)) **then**
      saida$(\sigma_i, \sigma_j, \sigma_k)$;
      juntar$(e_{ij}, \sigma_k, \mathcal{F})$;
      if $eki \in \mathcal{F}$ **then**
        colar(e_{ik}, e_{ki});
      end
      if $ejk \in \mathcal{F}$ **then**
        colar(e_{kj}, e_{jk});
      end
    end
    marque-como-bordo(e_{ij});
  end
  if $\sigma_i, \sigma_j, \sigma_k \leftarrow$ *primeira-escolha()* **then**
    saida$(\sigma_i, \sigma_j, \sigma_k)$;
    adicionar$(e_{ij}, \mathcal{F})$;
    adicionar$(e_{jk}, \mathcal{F})$;
    adicionar$(e_{ki}, \mathcal{F})$;
  end
end

Processo de formação da malha

A construção da malha ocorre da seguinte forma: inicialmente $\mathcal{G}$ é apenas um triângulo que foi gerado a partir do algoritmo primeira-escolha e cuja fronteira são seus 3 lados (seção 3.6). Criamos então uma fila de arestas, que chamaremos de frente $\mathcal{F}$, e conterá todas as arestas de fronteira da malha num determinado instante. Retiramos o primeiro elemento de $\mathcal{F}$ e executamos o passo-principal. Caso o passo-principal retorne um ponto inexistente então a aresta é de bordo e é acrescentada em uma lista de bordos $\mathcal{B}$, caso contrário criamos mais três arestas, duas das quais passam por duas operações topológicas denominadas junção e colagem. Observe que quando $\mathcal{F}$ é esvaziada a rotina primeira-escolha é chamada novamente criando em $\mathcal{G}$ uma nova componente conexa .

Estão implícitos na lista $\mathcal{B}$ vários ciclos de arestas que também são denominados poligonais de buracos. Como veremos no capítulo 6, um dos nossos principais objetivos é dado $e_{ij} \in \mathcal{F}$, construir um polígono que tenha e_{ij} como aresta inicial e percorrer um ciclo até retornar novamente a aresta e_{ij}.

Consultas

Duas consultas são fundamentais para o algoritmo. A primeira consulta é saber se um ponto é da fronteira ou do interior da malha. Para isso, atribuímos um índice a cada ponto (veja na estrutura POINT) que indicará o número de arestas vizinhas sem um par oposto. Vejamos, então, as propriedades do índice que respondem a nossa consulta:

- Um ponto p do interior da malha tem índice zero e é único, isto é, não existe outro vértice de bordo que aponte para p.
- Um ponto p com índice $n \geq 2$ tem $\frac{n}{2}$ vértices de bordo distintos apontando para p.

No algoritmo 3, essa consulta é representada pela rotina *indice.*

Na segunda consulta, dado e_{ij}, queremos verificar se $e_{ji} \in \mathcal{F}$. Uma das características da estrutura *half winged edge* é a de que permite resolvermos consultas simples como a de encontrar todos os vértices vizinhos de um dado vértice. Logo, a solução natural para a segunda consulta é procurar na vizinhança de $\sigma_j \rightarrow pt$ (aproveitando-se das estruturas i* e o*) por um vértice σ_k tal que $\sigma_k \rightarrow pt = \sigma_i \rightarrow pt$. Note que a busca é de natureza geométrica.

Junção

A operação de junção é efetuada logo após o passo-principal sobre a aresta e_{ij} que retorna um ponto p_k. Os passos de junção são:

- Criar um vértice σ_k apontando para p_k;

- Criar e_{ik}, e_{kj}, e_{ji} e inserir e_{ik}, e_{kj} no final da fila $\mathcal{F}$;
- Atualizar as informações topológicas (vizinhança) da malha[1];

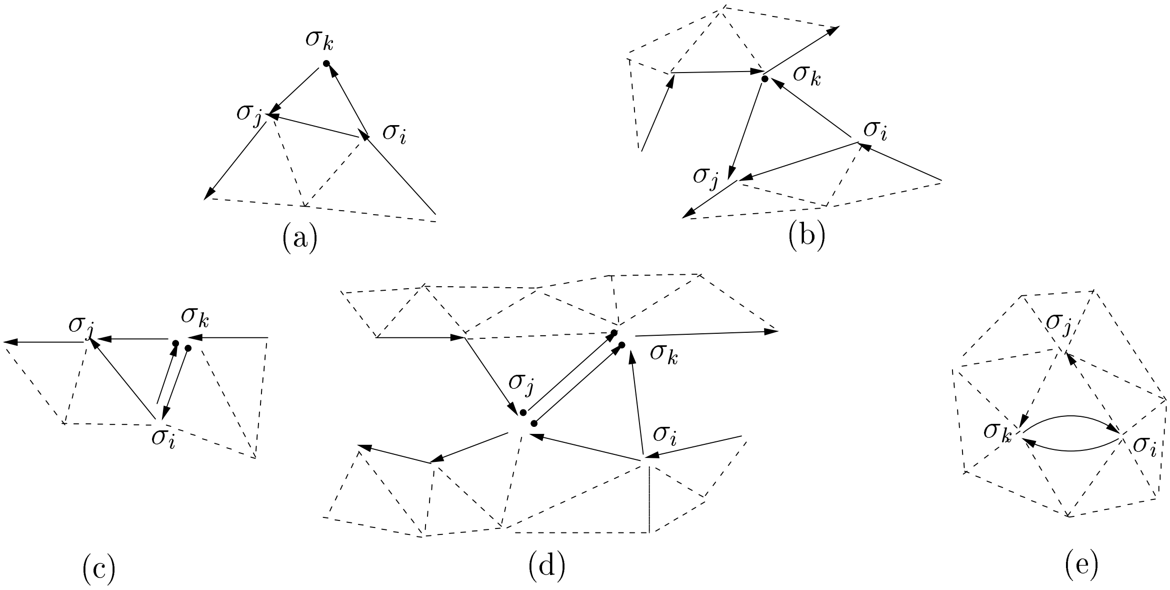


Figura 4.4: A bola gira em torno do segmento $\sigma_i\sigma_j$ e toca o ponto p_k. Em cada item é representado um caso de evento topológico. Apenas nos itens (c), (d) e (e) ocorrem junções e colagens. Nos itens (a) e (b) acontecem apenas junções.

Colagem

Esta rotina é executada depois da junção e trata de verificar os pares opostos correspondentes da fronteira e retirá-los de $\mathcal{F}$. Basicamente colagem executa dois passos:

- Se $e_{ki} \in \mathcal{F}$ então retire e_{ik}, e_{ki} de $\mathcal{F}$;
- Cole e_{ki} com e_{ik} caso não aconteça uma das situações especiais;

As situações especiais são: 1) Quando e_{ki} e e_{ik} tem a mesma direção (superfície não orientável) e 2) Quando e_{ki} é uma aresta interna [2]. O segundo caso é mais difícil de identificar mas uma reconstrução pictórica nos ajuda a visualizar (cf. fig 4.5).

Embora a palavra "cole" tenha um sentido conotativo e genérico ela é uma função complexa porque deve diagnosticar e tratar três casos que são classificados de acordo com a mudança da topologia das curvas de fronteira.

[1] aqui não explicitamos por sua irrelevância no contexto do algoritmo

[2] Em Bernardini[5] este caso não está claro

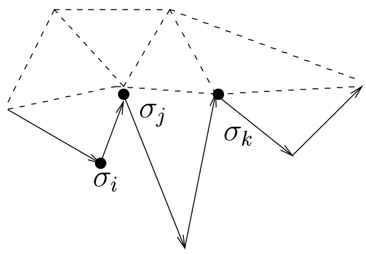


Figura 4.5: A bola gira em torno de $\sigma_i\sigma_j$ até tocar σ_k. A aresta $\sigma_k\sigma_j$ já existe e o triângulo $\sigma_i\sigma_j\sigma_k$ não pode ser criado.

Primeiro Caso: $e_{ik} \to \sigma_i = e_{ki} \to \sigma_j$ e $e_{ik} \to \sigma_j = e_{ki} \to \sigma_i$. Acontece quando existe um ciclo entre e_{ik} e e_{ki}. Uma curva deixa de existir (fig. 4.4e).

Segundo Caso: $e_{ik} \to \sigma_i = e_{ki} \to \sigma_j$ e $e_{ik} \to \sigma_j \neq e_{ki} \to \sigma_i$. Este é o caso mais comum e nada acontece com topologia das curvas. Um dos vértices herda as propriedades do outro que é removido (fig.4.4c).

Terceiro Caso: $e_{ik} \to \sigma_i \neq e_{ki} \to \sigma_j$ e $e_{ik} \to \sigma_j \neq e_{ki} \to \sigma_i$. Uma curva pode se dividir em duas ou duas formarem uma só. Dois vértices herdam as propriedades de seus pares correspondentes que são removidos (fig. 4.4d).

Algoritmo 4: cole

Input : $(e, \hat{e})$

if $e_i \neq \hat{e}_j$ **then**
  $ne(e_i) \leftarrow ne(\hat{e}_j)$;
  $i^*(e_i) \leftarrow i^*(e_i) + i^*(\hat{e}_j)$;
  $o^*(e_i) \leftarrow o^*(e_i) + o^*(\hat{e}_j))$;
end
if $e_j \neq \hat{e}_i$ **then**
  $pe(e_j) \leftarrow pe(\hat{e}_i)$;
  $i^*(e_j) \leftarrow i^*(e_j) + i^*(\hat{e}_i)$;
  $o^*(e_j) \leftarrow o^*(e_j) + o^*(\hat{e}_i)$;
end

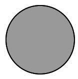
4.3 Operadores *Handlebody*

A teoria de *Handlebody*[3][17] nos ajuda a compreender melhor as mudanças topológicas na malha durante a execução do algoritmo. Sejam D^i um disco de dimensão $i = 0, 1, 2$ e ∂D^i seu respectivo bordo.

Definição 10 *$H_\lambda = (A_\lambda, B_\lambda)$ é um handle de índice $\lambda = 0, 1, 2$ tal que $B_\lambda \subseteq \partial A_\lambda$ onde $A_\lambda = D^\lambda \times D^{2-\lambda}$ e $B_\lambda = (\partial D^\lambda) \times D^{2-\lambda}$*

De acordo com a definição acima, só existem três tipos de *handle*:
Tipo-0, $\lambda = 0$:

$$A_0 = D^0 \times D^2 =$$



$$B_0 = (\partial D^0) \times D^2 = \emptyset$$

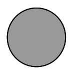
Tipo-1, $\lambda = 1$:

$$A_1 = D^1 \times D^1 =$$

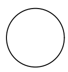
$$B_1 = (\partial D^1) \times D^1 =$$

Tipo-2, $\lambda = 2$;

$$A_2 = D^2 \times D^0 =$$



$$B_2 = (\partial D^2) \times D^0 =$$



Anexar um *handle* $H_\lambda = (A_\lambda, B_\lambda)$ ao bordo de uma superfície S consiste em associar por um homeomorfismo o conjunto $B_\lambda \subseteq \partial A_\lambda$ com um subconjunto de ∂S. Temos então o seguinte teorema:

Teorema 1 *(Decomposição de Superfícies por Handlebodies) Para toda superfície S, existe uma sequência finita de superfícies $S_{i=1\ldots N}$ tal que $S_0 = \emptyset$ e a superfície S_i é obtida anexando um handle $H_\lambda = (A_\lambda, B_\lambda)$ ao bordo de S_{i-1}.*

Um exemplo clássico é a decomposição do toro com a sequência $S_4 = (((S_0 + H_0) + H_1) + H_1) + H_2$ (fig.4.6)

[3]estudaremos somente o caso bidimensional

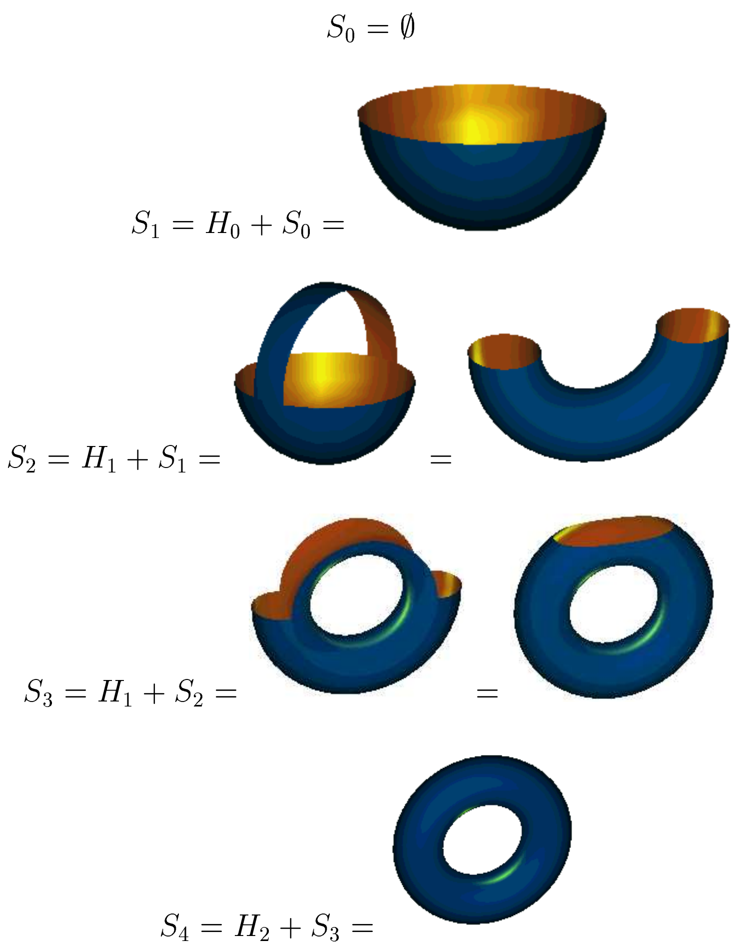


Figura 4.6: Decomposição do toro por Handlebodies. Cortesia de Sinésio Pesco.

O *handle* do tipo-0 cria sempre uma nova componente conexa homeomorfa a um disco e uma curva de bordo. Quando um *handle* do tipo-1 é anexado a uma superfície, podem ocorrer três situações:

- B_1 pode ser anexado a dois intervalos disjuntos na mesma curva de bordo. Nesse caso, uma curva se divide em duas.
- B_1 pode ser associado a intervalos de diferente curvas de bordo em uma superfície. A mudança topológica é caracterizada pelo aumento do gênero da superfície.
- B_1 é associado a diferentes componentes conexas.

No caso do *handle* de tipo-2, uma curva de bordo é fechada.

4.3.1 Aplicação no BPA

Esses *handles* estão relacionados diretamente com as operações de colagem vistas anteriormente e nos ajudam a compreender melhor o que está por trás do BPA.

- Associado ao *handle* do tipo-0 está a operação primeira-escolha (vista na seção 3.6) que gera uma nova componente conexa;
- O *handle* do tipo-1 acontece quando os dois vértices são diferentes durante a colagem de arestas opostas;
- Por fim, um *handle* do tipo-2 também é acrescentado na colagem quando ambos os vértices são coincidentes.

Por não haver nehuma mudança topológica no segundo caso da colagem, consideramos esta operação equivalente a um homeomorfismo.

4.3.2 Aplicação em Compressão de Malhas

Em Rossignac et al. [15] temos uma aplicação de Handlebodies para compressão de malhas em superfícies de genus maior que zero[4]. O algoritmo de compressão, *Edgebreaker*, é capaz de, na prática, comprimir os dados da malha com um gasto de menos de dois bits por triângulo. Este resultado pode contribuir para mais um melhoramento para o MPEG-4, padrão de compressão de dados 3D.

4.4 Análise de Complexidade

Nesta seção vamos mostrar que o BPA é $O(n)$ em relação ao número de vértices, assumindo que a densidade de uma amostragem P de uma superfície S é limitada, isto é, numa vizinhança 2α de um ponto $p \in S$ temos que $A < (B_{2\alpha}(p) \cap P) < B$, onde A e B são fixos.

Observe que a maioria dos passos são alterações ou verificações simples de listas encadeadas, filas e outras estruturas de dados que tomam um tempo constante pois lidam com um número limitado de vizinhos, pelo fato da densidade ser limitada. Dentre estas operações estão a junção e a colagem. As únicas rotinas que devem ser verificadas com mais cuidado são o passo-principal e a primeira-escolha.

A função passo-principal faz uma busca local. Como assumimos a densidade uniforme, concluímos que esta função também gasta um tempo $O(27B) = O(1)$.

A operação primeira-escolha testa em cada voxel todos os pares de segmentos incidentes num ponto não usado a existência de algum triângulo incidente que seja válido ou seja, verifica se existe algum ponto vizinho que esteja dentro de uma bola circunscrita

[4] Anteriormente resolvido somente para genus igual a zero

de raio α , o que leva um tempo $O(B^3) = O(1)$. No total o algoritmo gasta um tempo $O(n + m)$, onde m é o número de células no grid.

No capítulo 7 colocamos uma tabela que mostra a linearidade do BPA na prática.

Capítulo 5

A Amostragem

Neste capítulo vamos descrever condições ideais de amostragem para o BPA. Discutiremos também um algoritmo que amostra pontos a partir de uma poligonização.

5.1 A Amostragem

Vejamos inicialmente a relação entre BPA e o alpha shapes.

Proposição 4 *Seja $\mathcal{T}$ o conjunto das faces triangulares construídas pelo BPA e P o conjunto de amostras. Então $\mathcal{T} \subset S_\alpha(P)$.*

Prova: Dado $F \in \mathcal{T}$ uma face triangular. Sabemos que F é α-exposto pela *Observação 1* da seção 3.5, logo, toda componente conexa que o construída pelo BPA é subconjunto de $S_\alpha(P)$■

Voltemos ao Capítulo 2 e consideremos novamente o problema de reconstrução: dada uma superfície S, queremos computar uma superfície $\mathcal{T}$ de faces triangulares tal que dado $\epsilon > 0$ tenhamos $d(\mathcal{T}, S) < \epsilon$.

Escrito de maneira formal, temos o seguinte problema:

Problema 1 *Seja S uma superficie compacta sem bordo e $P \subset S$ um conjunto finito de pontos. Construa um complexo simplicial $\mathcal{C}(P)$ homeomorfo a S tal que $d(\mathcal{C}(P), S) < \epsilon$, fixado $\epsilon > 0$.*

A formulação do problema 1 nos remete a seguinte pergunta: Quais as condições suficientes da amostra que nos garantem a existência de um α tal que a malha construída pelo BPA satisfaça os requerimentos problema 1?

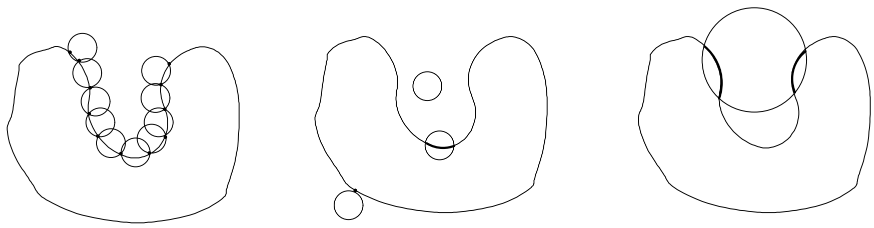
Essas condições estão respondidas em Bernardini et al. [4] para o *Alpha Shapes* de curvas planares, podendo também ser estendidas para superfícies. Além disso, com a Proposição 4, os mesmos argumentos funcionam para o BPA.

Teorema 2 *(Bernardini et al. [4]) Seja S uma superfície compacta sem bordo e $P \subset S$ um conjunto finito de pontos. Se*

1. *Para qualquer bola B_α de raio α, $S \cap B_\alpha$ pode ser uma das três situações seguintes: (a) vazio; (b) um ponto p (neste caso $p \in \partial B_\alpha$); (c)é homeomorfo a um disco no plano;*

2. *Toda bola aberta de raio α centrada em S contém no mínimo um ponto de S;*

então o α-shape $S_\alpha(P)$ é homeomorfo a S. Além disso $d(S_\alpha(P), S) < \alpha$.



(a) (b) (c)

Figura 5.1: *Alpha Shapes* dando a intuição do BPA (a); as três condições do **Teorema 2.1**(b); um contra-exemplo em que a condição **Teorema 2.1** não é satisfeita.

Corolário 1 *Seja $\mathcal{G}$ a malha gerada pelo BPA. Então $d(\mathcal{G}, S) < \alpha$.*

Prova: Decorre diretamente da Proposição 4 e do Teorema 2 vistos acima.

A primeira condição garante que o raio de curvatura de uma superfície é maior que α, permitindo que uma α-bola passe pelas cavidades e concavidades de S sem ter múltiplas interseções. A segunda condição nos diz que a amostragem P deve ser densa o suficiente para que uma bola possa caminhar sobre toda a amostragem sem deixar buracos.

Na verdade, sabemos que tal tipo de amostragem é quase sempre impossível por causa dos diversos tipos de problemas nos sistemas de captura: falta de dados, densidade não unifome, imperfeições no registro de dados, dentre outros.

No geral, o BPA comporta-se bem com ruídos. Quando uma região está faltando pontos surgem buracos. Nada acontece se existem muitos pontos. Se em uma região a curvatura é maior que o raio da bola então perdem-se alguns detalhes da superfície.

5.1.1 Simplificação

Vamos descrever um algoritmo que recebe como entrada uma malha $\mathcal{M}$, onde estão definidas faces, arestas e vértices e retorna um conjunto de pontos no espaço e suas respectivas normais.

Sejam F e S, o número de faces e a área superficial de $\mathcal{M}$ respectivamente e N o número de amostras que queremos. A densidade de pontos por unidade de área é dada por $D = \frac{N}{A}$. Seja $\mathcal{P}$ uma função boleana definida por $\mathcal{P}(f_0) = (Area(f) * D \geq random([0,1])$, onde f é uma face e $random(0,1)$ é uma função que retorna um valor pseudo-randômico entre entre 0 e 1 [1]. O procedimento consiste em percorrer todas as faces da malha e, dependendo da resposta positiva ou negativa de $\mathcal{P}$, ele gera um ponto no baricentro da face cuja normal é a normal de f. A função $\mathcal{P}$ nos permite concluir que:

- $\mathcal{X} = \{f \in F \,|\, Area(f) > 1/D\} = \{f \in F \,|\, \mathcal{P}(f) \geq 1\}$;
- Se $f_1, f_2 \in F - \mathcal{X}$ e $Area(f_1) > Area(f_2)$ então $\mathcal{P}(f_1) = 1$ é mais provável do que $\mathcal{P}(f_2) = 1$;
- Se $\mathcal{P}$‘ é o número de pontos da reamostragem, então $\mathcal{P}$‘ $\approx F$ quando $N = F$.

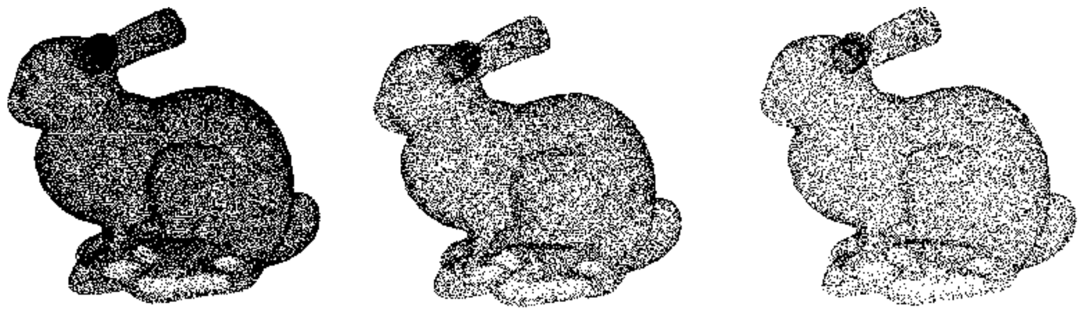
Intuitivamete podemos dizer que se a amostra está uniformentente distribuída então as faces têm pequenas variações de área. Se a amostra não está uniformente distribuída então existe $f \in \mathcal{X}$ tal que $Area(f) * D >> 1$ geraria apenas um ponto, ficando um grande espaço vazio naquela região. Isto pode ser contornado subdividindo f em triângulos menores.

Algoritmo 5: Simplificação

Input : $\mathcal{G}(V, E, F)$, número de pontos de saída N.
Output: N Pontos simplificados com normais.

```
S ← área superficial de G;
D = N/S;
for each F ∈ G do
    P ← Area(F)D;
    if P > random(0,1) then
        output(baricentro de F);
        output(normal de F);
    end
end
```

[1] Os números estão distribuídos uniformemente entre 0 e 1



(a) (b) (c)

Figura 5.2: Sequência de simplicações. Da esquerda para a direita 35933, 17782, 11890 pontos.

5.1.2 O Trabalho de Kobbelt et al.

Diferentemente do nosso método de simplificação que necessita de uma poligonização, o trabalho recente de Kobbelt et al. [12] faz a reamostragem a partir da nuvem de pontos. Essencialmente o algoritmo é implementado usando *clustering* hierárquico e incremental, simplificação iterativa e simulação de partículas. Outra boa característica é a simplificação adaptativa concentrando mais amostras nas regiões de alta curvatura. Esse algoritmo seria ideal para o BPA não fosse a ausência das normais.

Deixaremos como sugestão para trabalhos futuros uma adaptação desse algoritmo a fim de que ele gere também as normais.

5.2 Uma Estimativa Inicial para α

A motivação para esta seção é encontrar um valor para α como estimativa inicial e próximo de um valor ideal admitindo-se que a superfície está uniformemente amostrada. Apesar de existirem técnicas que calculam o raio baseado nas propriedades da luz estruturada ou do laser dos scanners, queremos estimá-lo de duas maneiras: uma conhecendo a área da superfície e outra desconhecendo-a.

Quando a área é conhecida a nossa idéia é imaginar um reticulado uniforme com a mesma quantidade de pontos e mesma área da superfície a ser considerada com o lado da célula do reticulado igual a α para que satisfaça a condição 2 do Teorema 2.

Quando não é conhecida a área da superfície então precisamos de uma aproximação. Para isso, tomamos a caixa envoltória da amostra e calculamos a área superficial de uma esfera que está contida num cubo de mesmo volume. O cálculo do raio procede-se da mesma forma que o caso anterior.

Analisando bem o último caso, percebemos que a caixa envoltória pode não ter volume (por exemplo uma superfície planar). Adaptamos então para um cubo que tenha a mesma área superficial da caixa envoltória. Apesar de ser uma aproximação grosseira, conseguimos bons resultados.

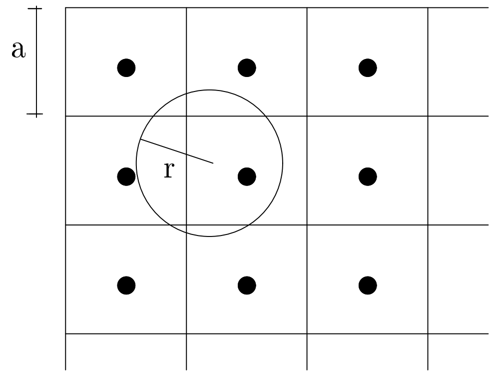


Figura 5.3: reticulado com um círculo que contém pelo menos uma amostra

Capítulo 6

Fechamento de Curvas de Bordo

Neste capítulo faremos uma descrição do problema de fechamento de buracos. Discutiremos também o algoritmo que chamaremos de *Quick Fill* e que foi implementado para casos em que o buraco tem a topologia de um disco.

6.1 Introdução

Os buracos são classificados em dois tipos: digitalização e algorítmico. O primeiro tipo decorre dos dispositivos atuais de captura de pontos que apesar de serem de alta resolução e de grande exatidão não conseguem exergar toda superfície por causa de oclusões da linha de visada, regiões de baixa reflectância, restrições de posicionamento do scanner, etc. O segundo, como o próprio nome diz, decorre de limitações do algoritmo ou de sua implementação.

Apesar de ser fácil detectar a existência de um buraco, há uma grande dificuldade de reconhecer qual a real topologia (fig. 6.1).

Figura 6.1: Topologias de buracos distintas

Em casos cuja topologia é a de um disco caímos no problema de fechar polígonos 3D. Levantamos então a seguinte pergunta: Dado um polígono 3D é possível triangulá-lo de tal modo que os triângulos não se interseptem? Não obstante, esse é um problema que já foi discutido em Barequet et. al [2]. É o que vamos ver a seguir.

6.1.1 O Trabalho de Barequet et al.

No trabalho de Barequet et. al [2] demonstra-se que o problema de decidir se um polígono 3D é triangulável, é NP-Completo. A prova é para um problema ainda mais geral.

Definição 11 *Dada uma coleção de polígonos, determinar se podemos simultaneamente triangular todos os polígonos com uma coleção de triângulos que não se interceptam, é definido como Problema de Triangulação Tridimensional Generalizado (PTTG).*

Teorema 3 *Existe uma redução de 3-SAT para PTTG em tempo polinomial. Assim, PTTG é NP-Completo.*

Este teorema nos diz que o problema é intratável em dados de grande escala (como em fotografia 3D por exemplo). No entanto, mostra-se que a existência de uma projeção esférica do polígono sem que haja interseções entre as arestas na superfície da esfera é uma condição suficiente para que exista uma triangulação. Tal triangulação é feita a partir da projeção gnomônica do bordo da esfera ao plano, reduzindo o problema original ao problema de triangulação planar.

6.1.2 O Trabalho de Levoy et al.

Atualmente tem ocorrido muita pesquisa sobre fechamento de buracos. Num trabalho recente de Levoy et al. [13] é apresentado um algoritmo que utiliza um método pioneiro de difusão volumétrica para reconstrução de superfícies, bastante conhecido na comunidade de pesquisadores em processamento de imagens. O método consiste em aplicar uma sequência de operações iterativamente afim de propafar informação de regiões conhecidas para regiões desconhecidas.

Neste trabalho, também são descritas as características de um algoritmo ideal de preenchimento de buracos, dentre elas, destacamos:

- Gerar uma variedade, isto é, não conter auto-interseções;
- Escolher a topologia apropriadas para buracos que possam ter múltiplas componentes de bordo;
- Usar todas as ferramentas disponíveis tais como linhas de visada dos *scanners*, normais, etc.

- Ser eficiente e escalável para grande quantidade de dados;

Também são classificados os tipos de algoritmos quanto a construção da malha. Basicamente são dois tipos: preenchimento em pós-processamento e preenchimento durante a construção da malha.

6.2 O Método *Quick Fill*

Feitas então, uma introdução sobre fechamento de buracos e caracterizações desejáveis dos seus diversos tipos de algoritmo, veremos que o nosso método, denominado de *Quick Fill*, permite auto-interseções e funciona somente para buracos que têm a topologia de um disco. Quanto à classificação, ele é de pós-processamento e divide-se em duas etapas:

1. Reconhecimento de Curvas de Bordo;
2. Triangulação.

A primeira etapa consiste em, a partir da lista $\mathcal{B}$ de arestas de bordo, gerar uma lista de polígonos simples. Para isso, basta tomar alguma aresta $e \in \mathcal{B}$, percorrer a poligonal do bordo em que e está contida marcando todas as arestas que for passando como já visitadas até retornar novamente à aresta e. O algoritmo segue tomando a próxima aresta de $\mathcal{B}$ que não esteja marcada.

Note que o algoritmo deve ter a capacidade de subdividir a poligonal que é composta de várias poligonais porque há casos em que o ponto contém mais de um vértice (fig. 7.2). Basicamente esse problema é reduzido a um algoritmo que detecta ciclos em grafos.

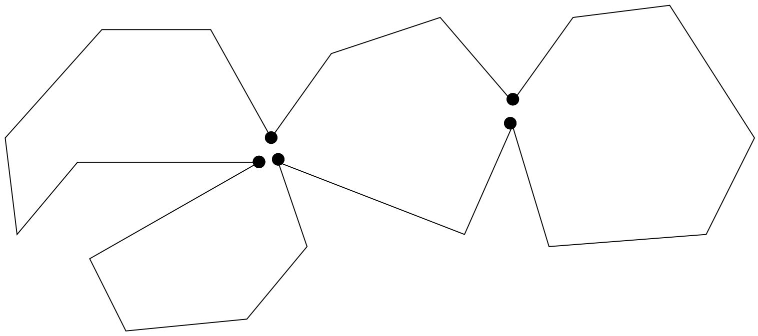


Figura 6.2: Uma poligonal que contém 4 buracos

O algoritmo encontra-polígonos, que reconhece as curvas de bordo, foi implementado utilizando duas pilhas. Uma que armazena as sub-poligonais e outra que guarda os vértices que têm índice de fronteira maior que dois. Denominemos as pilhas de Pp e Pv

respectivamente (cf. algoritmo 6). Em encontra-polígonos, quando o algoritmo chega em um vértice v cujo ponto tem índice maior que dois, ele empilha um novo polígono p em Pp e empilha v em Pv. Ao alcançar um vértice cujo ponto é igual ao do vértice que está no topo da pilha, atualizamos ambas as pilhas removendo o topo de Pv e também o topo de Pp fechando assim, um polígono. Devemos manter as propriedades do ponto, isto é, reduzir seu índice de dois e remover um de seus vértices.

Na segunda etapa a idéia da triangulação é criar uma nova face a partir do menor ângulo do polígono (cf. algoritmo 7, triangula-polígonos). No entanto, como disse antes, esta heurística não garante que não ocorram interseções entre as faces, até mesmo no caso bidimensional. Analisando os casos em que ocorrem interseções entre as faces, observamos uma irregularidade na amostragem, o que nos leva a crer que a poligonal deve ter lados "aproximadamente" iguais (fig 6.3). Isto é garantido pelo BPA porque os lados das poligonais estão limitados pelo diâmetro das α-bolas, que é 2α. Esse argumento é apenas uma justificativa que não está totalmente esgotada, o que deixa campo aberto para futura pesquisa. Contudo, apresentamos nos resultados o número de interseções detectadas no algoritmo com variações de raio e densidade de pontos da amostra.

A vantagem do *Quick Fill* é sua rapidez e simplicidade. A complexidade é $O(n)$ em encontra-poligonos e $O(n \log n)$ em triangula-polígono usando uma pilha de prioridades através da estrutura de dados *heap*.

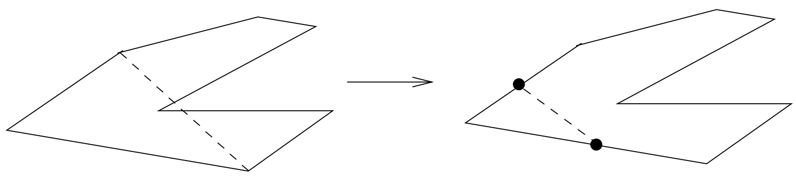


Figura 6.3: Contra-exemplo para o *Quick Fill*

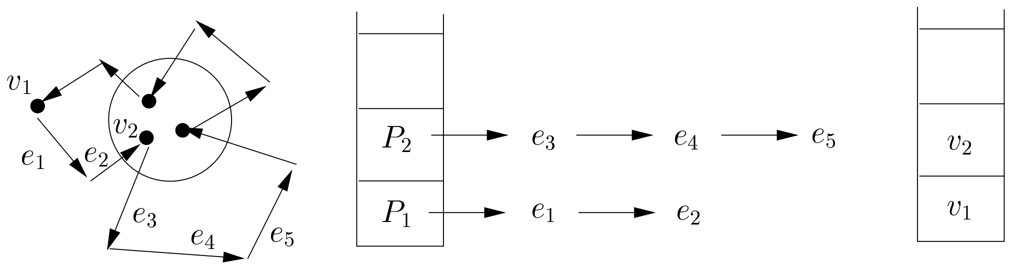


Figura 6.4: Estado das pilhas no algoritmo encontra-polígonos

Algoritmo 6: encontra-polígonos

```
for each e ∈ B do
    adicionar(topo(Pp), e);
    push(Pp, novo poligono);
    push(Pv, e_i);
    ê ← ne(e);
    while ê ≠ e do
        if indice(ê_i) > 2 then
            if pt(ê_i) ≠ pt(topo(Pv)) then
                push(Pp, novo poligono);
                push(Pv, ê_i);
            end
            else
                cole(ê_i, topo(Pv))^1;
                saida(pop(Pp));
                if indice(pt(topo(Pv))) > 2 then
                    push(Pp, novo poligono);
                end
                else
                    pop(Pv);
                end
            end
        end
        adicionar(topo(Pp), ê);
        ê ← ne(ê)
    end
    saida(topo(Pp), ê);
end
```

Observação 2 *A função em (1) no algoritmo acima "cola" dois vértices que apontam para o mesmo ponto, isto é, remove um e o outro herda as suas propriedades. O índice do ponto é reduzido de duas unidades.*

Algoritmo 7: triangula-polígono

Input : $P = p_1 p_2 ... p_n, \mathcal{G}$

$H = ConstruaHeap(P)$;
while $Tamanho(H) > 3$ **do**
  $p_i = pop(H)$;
  $juntar(\mathcal{G}, p_{i-1}, p_i, p_{i+1})$;
  $atualiza(H, p_{i-1})$;
  $atualiza(H, p_{i+1})$;
end
$juntar(\mathcal{G}, H[1], H[2], H[3])$;

6.3 Resultados

Para fazer os testes com o BPA, todas as aresta foram retiradas do *3D scanning repository* numa página *web* da Universdade de Stanford[1], exceto a amostra da cabeça. As primeiras amostras estavam originalmente em formato *.ply*, um dos padrões de descrição de malhas para fins de pesquisa, e foram tratadas no processo de simplificação (cf. seção 5.1.1). A amostra da cabeça foi retirada de um curso de fotografia 3D ministrado na *Caltech*[2]. O programa foi implementado em C++ e executado em plataforma Linux. O processador é de 500 MHz e a memória RAM é de 64 MB.

Os nomes que aparecerão nas tabelas de testes são:

r Raio proposto (cf. seção 5.2).

npts Número de pontos da amostra.

t(s) Tempo em segundos gasto na triangulação do BPA.

tps(s) Tempo gasto em segundos na rotina primeira-escolha.

nbur Número de buracos gerados pelo BPA.

ncomp Número de comparações feitas entre todos os triângulos de cada buraco para detectar interseções.

nint Número de interseções detectadas.

[1] http://graphics.stanford.edu/data/3Dscanrep/
[2] http://www.ugcs.caltech.edu/dragos/3DP/

% ncomp/nint Percentual do número de interseções detectadas sobre o número de comparações.

ntg Número de triângulos gerados para fechar os buracos.

ntg/nbur Média de triângulos gerados por buracos.

Na tabela da figura 6.5 percebe-se que o BPA é bastante eficiente com o raio sugerido r. Os números de buracos bem como os tempos de fechamento crescem à medida que aumenta os números de pontos. Apesar de existirem algumas interseções entre os triângulos que fecgam os buracos (**nint**), note que este número é pequeno em relação ao número de comparações (**% ncomp/nint**). Percebe-se também que o número de buracos é grande principalmente pela amostragem não estar bem uniforme. Veja nas figuras 6.9b e 6.9c as curvas de bordo e a triangulação respectivamente para 3501 pontos. Nessas figuras a bolinha que aparece em cor vermelhatem raio r.

Na tabela da figura 6.6 acrescentamos 50% do raio sugerido e percebe-se que o BPA praticamente dobrou o seu tempo de execução. O número de buracos bem como os tempos de fechamento têm poucas alterações por que são algorítmicos na sua maioria. O número de interseções entre os triângulos que fecham os buracos (**nint**) é quase sempre zero. Veja nas figuras 6.9d e 6.9e as curvas de bordo e a triangulação para 3501 pontos. A bolinha que aparece em cor vermelha tem raio $r + r/2$. Percebe-se nas figuras que existem buracos grandes não por problemas de amostragem mas porque no modelo do *coelho* os buracos estão presentes. Apesar desse raio $r + r/2$ ter melhores resultados em relação ao número de buracos, o modelo reconstruído perde muitas características do modelo original, isto é, a curvatura é suavizada em regiões de alta curvatura.

Na tabela da figura 6.7 acrescentamos 100% do raio sugerido e percebe-se que o BPA quase triplicou os seu tempo. O número de buracos bem como os tempos de fechamento também têm poucas alterações. O número de interseções entre triângulos que fecham os buracos (**nint**) é quase zero, exceto na última amostra. Veja nas figuras 6.9f e 6.9g as curvas de bordo e a triangulação respectivamente para 3501 pontos. A bolinha aparece em cor vermelha e tem raio $2r$. Nesse caso, o raio $2r$ já é indesejável porque o *coelho* está "pescoçudo".

Na figura 6.10 temos a a mostra da mão com 65222 pontos e a triangulação após a execução do BPA com raio r e o fechamento de buracos. No item 6.10c temos um detalhe do punho recontruído. Observe que o raio foi suficientemente pequeno para triangular uma pequena "caverna". Na tabela 6.11 temos os desempenhos com raio r, $r + r/2$ e $2r$. Essa amostra demonstroualguns casos de curvas de bordo indesejáveis que causaram problemas na hora de reconhecê-las. Existem triângulos que ligam os dedos da mão porque o raio da bola é menor do que a *vizinhança tubular* naquele local. Estes triângulos "pontes" são causadores de tais buracos porque os consideramos sem topologia definida.

Na figura 6.12 temos a amostra da cabeça com 38220 pontos e a triangulação após a execução do BPA com raio r eo fechamento de alguns buracos. Observe que existem várias componentes conexas, aumentando assim, o tempo gasto no algoritmo primeira-escolha por ser chamado várias vezes. Na tabela 6.13 vemos esses resultados para os raios r, $r + r/2$, e $2r$. No programa que implementamos é possível selecionar um buraco e depois triangulá-lo.

n pts	t(s)	tps(s)	nbur	tfb	ncomp	nint	% ncomp/nint	ntg	ntg/nbur
3501	1.23	0	185	0.16	32127	11	0.0342391	2204	11.9135
7092	2.32	0.01	347	0.58	54723	6	0.0109643	4124	11.8847
10695	3.48	0.03	521	1.27	77862	7	0.00899026	6107	11.7217
14398	4.65	0.04	678	2.12	92028	13	0.0141261	7614	11.2301
18068	5.87	0.09	763	2.68	117885	26	0.0220554	8347	10.9397
21686	7.04	0.13	912	3.65	122514	13	0.010611	9429	10.3388
25283	8.17	0.19	1017	4.61	130083	9	0.00691866	10254	10.0826
28815	9.35	0.27	1138	5.49	125088	37	0.0295792	10726	9.42531
32391	10.41	0.44	1227	6.99	119766	42	0.0350684	10602	8.64059

Figura 6.5: Desempenho do BPA (raio r) e do algoritmo de fechamento de buracos com as amostras do *coelho*.

n pts	t(s)	tps(s)	nbur	tfb	ncomp	nint	% ncomp/nint	ntg	ntg/nbur
3501	2.34	0.01	21	0	1050	0	0	141	6.71429
7092	4.62	0.02	28	0.01	2415	0	0	214	7.64286
10695	6.91	0.03	32	0.01	4146	0	0	278	8.6875
14398	9.26	0.04	22	0.01	8568	2	0.0233427	346	15.7273
18068	11.55	0.04	30	0.01	13008	13	0.0999385	396	13.2
21686	13.97	0.05	32	0.01	16536	3	0.0181422	461	14.4062
25283	16.36	0.07	28	0.02	19671	0	0	517	18.4643
28815	18.54	0.09	26	0.01	21690	0	0	512	19.6923
32391	20.94	0.12	16	0.01	25062	0	0	496	31

Figura 6.6: Desempenho do BPA (raio $r + r/2$) e do algoritmo de fechamento de buracos com as amostras do *coelho*.

n pts	t(s)	tps(s)	nbur	tfb	ncomp	nint	% ncomp/nint	ntg	ntg/nbur
3501	3.78	0.08	22	0	243	0	0	76	3.45455
7092	7.67	0.23	20	0	870	0	0	86	4.3
10695	11.5	0.32	21	0.01	1428	0	0	123	5.85714
14398	15.62	0.45	27	0	2019	0	0	132	4.88889
18068	19.51	0.53	20	0.01	4443	0	0	221	11.05
21686	23.61	0.61	19	0	6861	0	0	277	14.5789
25283	27.53	0.7	17	0	14250	0	0	347	20.4118
28815	31.23	0.79	14	0	16074	0	0	374	26.7143
32391	35.35	0.85	14	0.01	18327	2	0.0109129	407	29.0714

Figura 6.7: Desempenho do BPA (raio $2r$) e do algoritmo de fechamento de buracos com as amostras do *coelho*.

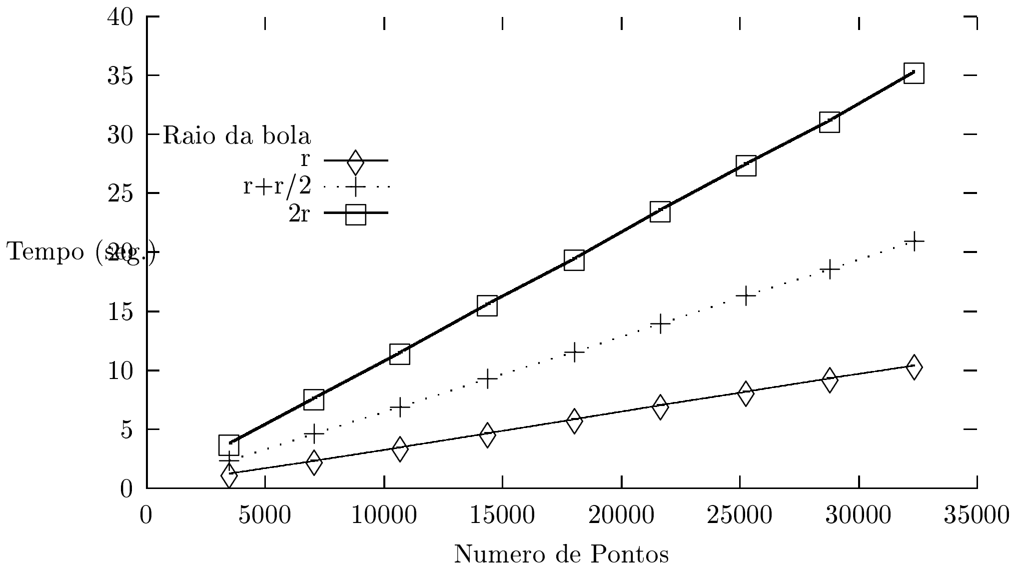
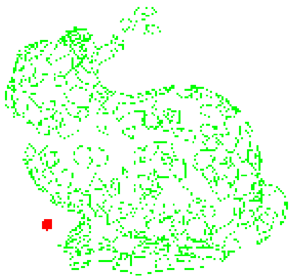


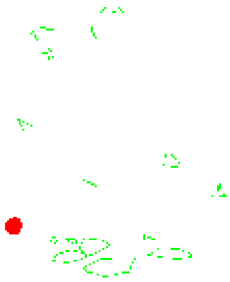
Figura 6.8: Gráfico que mostra a linearidade do BPA em relação ao raio. No gráfico r é o raio proposto (cf seção 5.2).

(a)



(b)

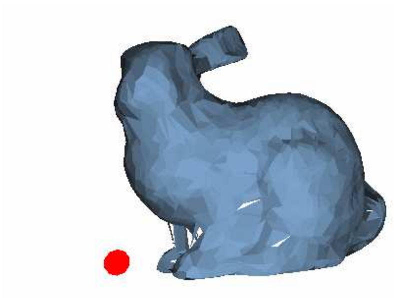
(c)



(d)

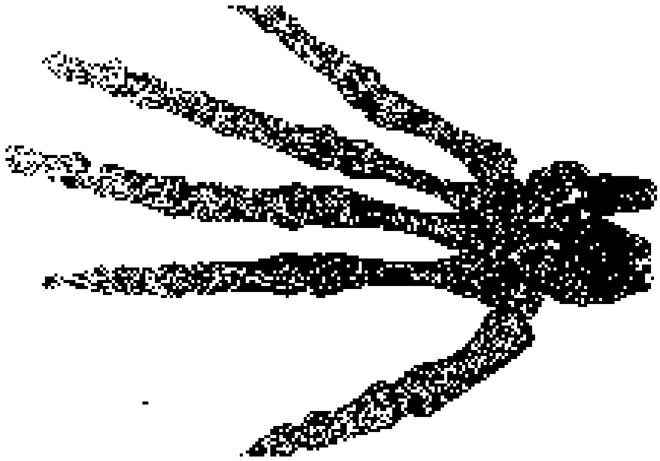
(e)

(f)

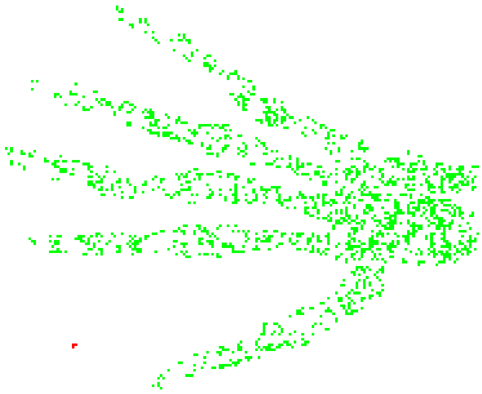


(g)

Figura 6.9: 30501 amostras do coelho (a) e com essas amostras buracos abertos (b), (d) e (f) e fechados (c), (e) e (g) gerados pelo BPA com raios r, $r + r/2$ e $2r$ respectivmente.



(a)



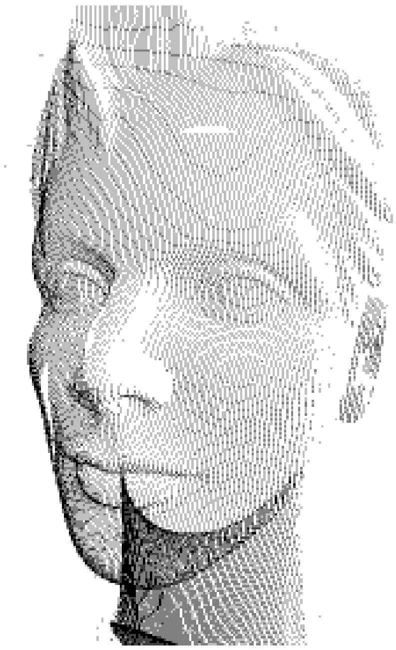
(b)

(c)

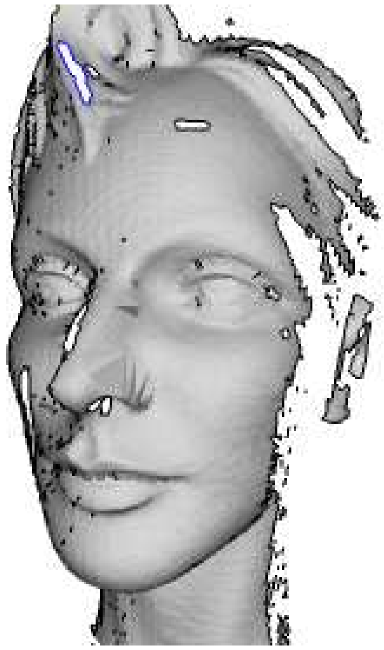
Figura 6.10: Amostras da mão (a), buracos (b) e dos buracos fechados (c). Em (c) há uma região côoncava que a bolinha conseguiu triangular.

n pts	t(s)	tps(s)	nbur	tfb	ncomp	nint	% ncomp/nint	ntg	ntg/nbur
65222	28.92	1.43	1455	9.42	69330	29	0.0418289	10859	7.46323
65222	62.75	8.9	260	0.31	5280	1	0.0189393	1023	3.93462
65222	102.3	36.18	179	0.06	1701	2	0.117578	413	2.30726

Figura 6.11: Desempenho do BPA (raio $r + r/2$) e do algoritmo de fechamento de buracos com as amostras do *mão*.



(a)



(b)

Figura 6.12: (a) Amostragem; (b) alguns buracos fechados. No programa é possível escolher a curva para fechar.

n pts	t(s)	tps(s)
38220	19.91	10.44
38220	29.41	31.1
38220	33.05	153.74

Figura 6.13: Desempenho do BPA (raio r, $r + r/2$ e $2r$) com as amostras da *cabeça*.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste capítulo discutiremos as contribuições da dissertação e descreveremos propostas para trabalhos futuros que são frutos dessa dissertação.

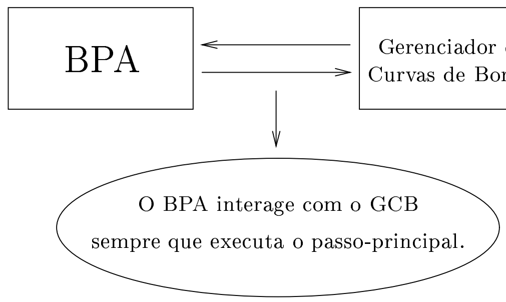
7.1 Conclusões

Em síntese, nos principais pontos da dissertação:

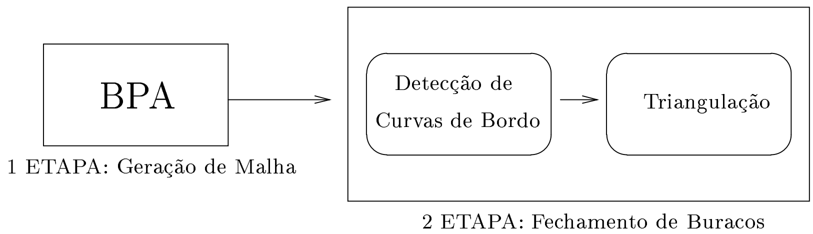
- Apresentamos o BPA e falamos de sua geometria, complexidade, vantagens, etc.;
- Descrevemos as características de uma amostragem ideal para o BPA;
- Explicamos a estrutura da malha e utilizamos a teoria de Handlebody para formalizar os eventos topológicos ocorridos no decorrer do algoritmo;
- Propomos um algoritmo para fechar as curvas de bordo;

Apesar de no trabalho de Bernardini et al. [5] existir uma proposta para o fechamento de buracos, optamos por fazer um novo algortimo aproveitando-nos da estrutura das curvas de bordo, recaindo no problema de triangulação de polígonos 3D. Graças a essa decisão descobrimos, como mencionado na introdução, algumas inconsistências nos índices dos pontos localizados nas arestas de bordo que nos levaram a criar uma diferenciação entre pontos e vértices. Tal distinção deu-nos uma nova implementação que se combinou com a base teórica dos operadores de Handlebody. Assim, as contribuições da teoria Handlebody[17] foram de suma importância porque não há um detalhamento dos eventos topológicos no trabalho de Bernardini da forma que foram descritos no capítulo 4 dessa dissertação.

Existe uma importante diferença que é sutil mas pode ser percebida na nossa estrutura de dados em relação ao trabalho de Bernardini. No nosso trabalho não mantivemos uma lista de curvas de bordo assim como foi proposto na implementação original. Neste capítulo, gostaríamos de enfatizar que a lista foi criada, como descrito no capítulo anterior, depois do BPA, separando assim o algoritmo em duas fases bem distintas que são triangulação e fechamento de buracos (detecção e triangulação). Isso, de fato, contribui efetivamente no decaimento do tempo de processamento do BPA porque não perdemos tempo para manter a lista. Veja uma ilustração dos esquemas propostos na figura abaixo.



(a)



(b)

Figura 7.1: Comparando abordagens de reconhecimento de Curvas de Bordo. (a) Abordagem de Benardini et al. - uma lista de curvas é mantida durante o BPA (b) Abordagem da dissertação - a lista de curvas é criada após o BPA.

7.2 Trabalhos Futuros

De imediato podemos melhorar o BPA implementando uma matriz de voxels esparsa porque existem muitos deles que são vazios. Isto dará uma economia de memória. Também podemos dividir a caixa envoltória dos pontos em fatias igualmente espaçadas onde cada uma delas seria carregada do disco para a memória principal afim de efetuar o BPA. As arestas de bordo que ficam entre as fatias são "congeladas". Ao iniciar-se o processo na próxima fatia as arestas congeladas são reutilizadas (descongeladas). Estes dois melhoramentos também são discutidos em Bernardini et al. [5].

Podemos implementar operadores Handlebody em processos de construção mais gerais. Por exemplo, dado um arquivo que armanzena uma triangulação com pontos e faces com os índices dos pontos o processo de construção da malha é semelhante ao processo de construção da malha do BPA.

Outro resultado interessante que pode ser extraído no BPA é o gênero da superfície. Como foi comentado no Capítulo 4, fica claro com o auxílio da teoria de Handlebody como detectar o instante em que é incrementado o gênero da superfície.

Bibliografia

[1] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, union of balls, and medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, March 2001.

[2] G. Barequet, M. Dickerson, and D. Eppstein. On triangulating three-dimensional polygons. *Computational Geometry: Theory and Applications*, 10(3), June 1998.

[3] B. Baumgart. A polyhedron representation for computer vision. In *AFIPS Conf. Proc.*, pages 589–596, 2002.

[4] F. Bernardini and C. Bajaj. Sampling and reconstructiong manifolds using alpha-shapes. In *Ninth Canadian Canadian Conf. Computational Geometry*, pages 193–198, August 1997.

[5] F. Bernardini, J. Mittlelman, H. Rushmeir, and C. Silva. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Visual. Comput. Graphics*, 5(4):349–359, 1999.

[6] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), February 1992.

[7] J. D. Boissonant. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics.*

[8] H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 10(1):43–72, Jan. 1994.

[9] P.L. George and E. Seveno. The advancing front mesh generation method revisited), journal= International Journal for Numerical Methods in Engineering, volume =37, pages = 3605-3619, year = 1994,.

[10] J. Gomes and L. Velho. Graphical objects. *The Visual Computer.*

[11] H. Hoppe, T. DeRose, T. Duchanp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *ACM Computer Graphics*, volume 26, pages 71–78, 1992.

[12] L. P. Kobbelt, M. Pauly, and M. Gross. Efficient simplifications of point sampled surfaces. In *IEEE Visualization 2002*, October.

[13] M. Levoy, J. Davis, S. R. Marshner, and M. Garr. Filling holes in complex surfaces using volumetric diffusion. In *13th International Symposium on 3D Data Processing Visualization and Transmission*, Padova, Italy, June 2002.

[14] S.H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[15] H. Lopes, J. Rossignac, A. Szymczack, A. Safanova, and G. Tavares. Edgebreaker: A simple algorithm for surfaces with handles. *Proceedings of the seventh ACM SIGGRAPH Symposium on Solid Modeling and Its Applications*, pages pp. 108–117, 2002.

[16] E. Mencl and H. Muller. Interpolation and approximation of surfaces from three-dimensional scattered data points. In *State of the Art Reports, ACM Eurographics 98*, pages 63–69, 1998.

[17] C. P. Rourke and Sanderson B. J. Introduction to piecewise linear topology. *Springer Study Edition*, 1982.

[18] H. K. Zhao, S. Osher, B Merriman, and M. Kang. Implicit, non-parametric shape re-contruction from unorganized points using variational levelset method. In *Computer Vision and Image Processing*, 2002.