

Wallace Souza Loos

REFORMATAÇÃO CURVILÍNEA BASEADA EM SIMPLIFICAÇÃO E  
COSTURA DE MALHAS

Campinas  
2014



Universidade Estadual de Campinas  
Faculdade de Engenharia Elétrica e de Computação

Wallace Souza Loos

REFORMATAÇÃO CURVILÍNEA BASEADA EM SIMPLIFICAÇÃO E COSTURA DE MALHAS

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Orientadora: Profa. Dra. Wu, Shin-Ting

Este exemplar corresponde à versão final da dissertação defendida pelo aluno, e orientada pela Profa. Dra. Wu, Shin-Ting

---

Campinas  
2014

Ficha catalográfica  
Universidade Estadual de Campinas  
Biblioteca da Área de Engenharia e Arquitetura  
Luciana Pietrosanto Milla - CRB 8/8129

L874r Loos, Wallace Souza, 1989-  
Reformatação curvilínea baseada em simplificação e costura de malhas /  
Wallace Souza Loos. – Campinas, SP : [s.n.], 2014.

Orientador: Wu Shin-Ting.  
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de  
Engenharia Elétrica e de Computação.

1. Computação gráfica. 2. Malhas. 3. Diagnóstico por imagem. I. Wu, Shin-  
Ting, 1958-. II. Universidade Estadual de Campinas. Faculdade de Engenharia  
Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** Curvilinear reformatting based on mesh simplification and zippering

**Palavras-chave em inglês:**

Computer graphics

Meshes

Diagnostic imaging

Simplification algorithm

**Área de concentração:** Engenharia de Computação

**Titulação:** Mestre em Engenharia Elétrica

**Banca examinadora:**

Wu Shin-Ting [Orientador]

Thomas Maurice Lewiner

Jorge Stolfi

**Data de defesa:** 10-10-2014

**Programa de Pós-Graduação:** Engenharia Elétrica





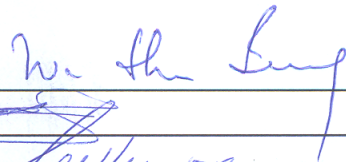
## COMISSÃO JULGADORA - TESE DE MESTRADO

**Candidato:** Wallace Souza Loos

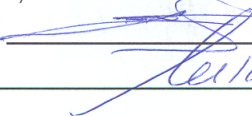
**Data da Defesa:** 10 de outubro de 2014

**Título da Tese:** "Reformatação Curvilínea Baseada em Simplificação e Costura de Malhas"

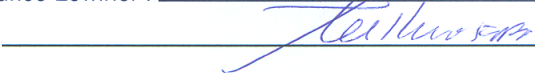
Profa. Dra. Wu Shin-Ting (Presidente):



Prof. Dr. Thomas Maurice Lewiner:



Prof. Dr. Jorge Stolfi:





# Resumo

A reformatação curvilínea é uma técnica computacional de exploração, portanto, não invasiva, que permite realizar cortes curvilíneos sobre as neuroimagens. Ela complementa as reformatações multiplanares na localização de displasia cortical focal, causa comumente associada a epilepsia refratária. Foi desenvolvido um algoritmo de reformatação curvilínea baseado em malhas de *offset* pelo nosso grupo de pesquisa. Dois problemas foram identificados no algoritmo: artefatos visuais e limitação da região a ser reformatada em áreas visíveis. Neste trabalho apresentamos soluções para contornar estes problemas. Mostramos que a causa dos artefatos são as auto-interseções locais e que um algoritmo de simplificação pode removê-las de forma eficiente sem comprometer a geometria de reformatação. Desenvolvemos um algoritmo de costura para juntar malhas obtidas em diferentes ângulos de vista numa única malha de corte. Desta forma, é possível realizar a reformatação simultânea nos dois hemisférios cerebrais propiciando um diagnóstico baseado em análise comparativa dos dois lados. Na implementação dos nossos algoritmos procuramos explorar as vantagens de uma estrutura de dados eficiente e do paralelismo das unidades de processamento gráfico (GPUs) para termos resultados em tempo interativo. Experimentos preliminares com as neuroimagens dos pacientes com crises epiléticas apontam que a ferramenta pode colaborar na identificação de lesões sutis.

Palavras-chave: Reformatação Curvilínea. Malha. *Half-Edge*. Algoritmo de Simplificação. Displasia Cortical Focal.



# Abstract

Curvilinear Reformatting is a computational exploration technique, therefore noninvasive, that allows making curvilinear slices on 3D neuroimaging data. It complements multiplanar reformatting to find lesions of focal cortical dysplasia, a common cause of refractory epilepsy. Our research group has developed an algorithm for curvilinear reformatting based on offset meshes. Two problems were identified: visual artifacts and limitations of the visible area to be reformatted. In this work we presented two solutions to solve these problems. We show that the cause of the artifacts are the local self-intersections and that a simplification algorithm can remove them efficiently without compromising the geometry of reformatting. We developed a sewing algorithm to join meshes from different angles of view on a single mesh. In this way, it is possible to make the curvilinear reformatting in both hemispheres of the brain, providing a diagnosis based on comparative analysis of the two sides. In the implementation of our algorithms we tried to explore the advantages of efficient data structure and parallelism of graphic processor units (GPU) to achieve results at interactive rates. Preliminary results with neuroimages from patients with epileptic seizures indicate that the tool may collaborate in the finding of subtle lesions.

Key-words: Curvilinear Reformatting, Mesh, Half-Edge, Simplification Algorithm . Focal Cortical Dysplasia.



# Sumário

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introdução</b>  | <b>1</b>  |
| 1.1      | Motivação . . . . .  | 3         |
| 1.2      | Proposta . . . . .   | 3         |
| 1.3      | Objetivos . . . . .  | 4         |
| 1.4      | Estrutura do Texto . . . . .   | 4         |
| <b>2</b> | <b>Preliminares</b>  | <b>5</b>  |
| 2.1      | Malhas de <i>Offset</i> . . . . .  | 5         |
| 2.2      | Estrutura de Dados <i>Half-Edge</i> . . . . .                                  | 6         |
| 2.3      | Algoritmo de Simplificação . . . . .   | 7         |
| 2.4      | Mapa de Profundidade . . . . .   | 9         |
| 2.5      | Voxelização . . . . .  | 10        |
| 2.6      | Distância de Hausdorff . . . . .   | 10        |
| 2.7      | <i>Ray Casting</i> . . . . .   | 12        |
| 2.8      | Reformatações . . . . .  | 12        |
| 2.9      | Algoritmo de Reformatação Curvilínea . . . . .                                 | 13        |
| <b>3</b> | <b>Reformatação Curvilínea</b>   | <b>14</b> |
| 3.1      | Geração de Malhas de <i>Offset</i> Usando Algoritmo de Simplificação . . . . . | 14        |
| 3.2      | Trabalhos Relacionados . . . . .   | 15        |
| 3.3      | Hipótese . . . . .   | 16        |
| 3.4      | Proposta . . . . .   | 16        |
| 3.4.1    | Contrações para Simplificar . . . . .  | 19        |
| 3.4.2    | Local ou Global? . . . . .   | 21        |
| 3.4.3    | Algoritmo de Voxelização . . . . .   | 22        |
| 3.4.4    | Geração de Malhas de <i>Offset</i> . . . . .                                   | 26        |
| 3.5      | Testes de Validação . . . . .  | 27        |
| 3.5.1    | Eficiência . . . . .   | 27        |
| 3.5.2    | Qualidade . . . . .  | 28        |
| 3.6      | Discussões . . . . .   | 29        |



|          |  |           |
|----------|--|-----------|
| <b>4</b> | <b>Costura entre Malhas</b>  | <b>33</b> |
| 4.1      | Trabalhos Relacionados . . . . .   | 34        |
| 4.2      | Hipótese . . . . .   | 34        |
| 4.3      | Proposta . . . . .   | 34        |
| 4.3.1    | Remoção da Área Sobreposta . . . . .                                     | 35        |
| 4.3.2    | Ligação da Malha . . . . .   | 40        |
| 4.4      | Testes de Validação . . . . .  | 45        |
| 4.4.1    | Eficiência . . . . .   | 46        |
| 4.4.2    | Qualidade . . . . .  | 47        |
| 4.5      | Discussões . . . . .   | 48        |
| <b>5</b> | <b>Resultados</b>  | <b>52</b> |
| 5.1      | Reformatação Curvilínea em Extensão Arbitrária . . . . .                 | 52        |
| 5.2      | Integração a um Ambiente de Exploração Visual de Neuro-Imagens . . . . . | 52        |
| 5.2.1    | VMTK . . . . .   | 53        |
| 5.3      | Experimentos . . . . .   | 55        |
| 5.4      | Diagnóstico de Displasia Cortical Focal . . . . .                        | 57        |
| 5.5      | Discussões . . . . .   | 60        |
| <b>6</b> | <b>Conclusões e Trabalhos Futuros</b>                                    | <b>63</b> |
|          | <b>Bibliografia</b>  | <b>66</b> |

*Dedico este trabalho a minha mãe,  
Joana Darc.*



# Agradecimentos

Agradeço,

ao Laboratório de Neuroimagem (LNI) da Universidade Estadual de Campinas (UNICAMP) pelos dados cedidos. Em especial ao Prof. Dr. Fernando Cendes, Dra. Clarissa Yasuda e Dra. Ana Carolina Coan por toda paciência e tempo dedicado ao nosso trabalho,

ao pessoal do Centro de Tecnologia da Informação Renato Archer (CTI), Jorge Vicente Lopes da Silva, Marcelo Fernandes de Oliveira e Paulo Henrique Junqueira Amorim por todo suporte oferecido,

a todos os professores que contribuíram com a minha formação, em especial, Prof. Clarimar José Coelho e Prof. Alexandre Ribeiro,

a minha orientadora, Profa. Wu, Shin-Ting,

ao Prof. Richard Frayne, da Universidade de Calgary, pela oportunidade de intercâmbio,

a todos os colegas do Laboratório de Controle e Automação (LCA) da Faculdade de Engenharia Elétrica e de Computação (FEEC) da UNICAMP,

a todos os meus amigos, em especial, Márcio Reis, Marília Reis, Suelen Mapa, Mariana Bento, Roberto Medeiros, Augusto Valente, Tamires Zanão, Paulo Gurgel, Larissa Fleury e Paula Gimenes,

ao CNPq pelo suporte financeiro,

a minha mãe,

e a todos que de alguma forma contribuíram com o meu progresso como aluno e como pessoa.



# Lista de Figuras

|      |   |    |
|------|---|----|
| 1.1  | Suspeita de lesão de displasia cortical focal destacada. . . . .  | 1  |
| 1.2  | Reformatação curvilínea realizada por Bergo e Falcão. . . . .   | 2  |
| 1.3  | Reformatação curvilínea: (a) amostragem, (b) criação da malha, (c) deslocamento da malha e (d) resultado final da reformatação curvilínea. . . . .  | 3  |
| 1.4  | Reformatação curvilínea com artefatos: (a) visualização das malhas deslocadas sobre o volume, (b) e (c) visualização do corte curvilíneo com os artefatos. . . . .  | 4  |
| 2.1  | Exemplos de malhas. . . . .   | 5  |
| 2.2  | Geração das malhas de <i>offset</i> $M_i$ a partir de $M$ . . . . .   | 6  |
| 2.3  | Estruturas de dados (a) <i>winged-edge</i> , (b) <i>half-edge</i> e (c) <i>quad-edge</i> . Na estrutura de dados <i>winged-edge</i> , a aresta $e$ , armazena informações sobre seus vértices extremos, $v_1$ e $v_2$ , a face à esquerda e à direita, $f_1$ e $f_2$ , e as arestas $e_1$ , $e_2$ , $e_3$ e $e_4$ . A estrutura de dados de <i>half-edge</i> , $he_c$ armazena uma referência para o vértice de origem, $v_2$ , a face adjacente, $f_2$ e o próximo, $he4_c$ , anterior, $he2_c$ , e o oposto, $he_{cc}$ <i>half-edge</i> . A estrutura de dados <i>quad-edge</i> possui os mesmos ponteiros da estrutura de dados <i>winged-edge</i> . . . . . | 7  |
| 2.4  | Malha triangular representada numa estrutura de <i>half-edge</i> . (a) As arestas da malha triangular, (b) são representadas por duas arestas direcionadas na estrutura de <i>half-edge</i> . . . . .   | 7  |
| 2.5  | Remoção do vértice $v$ . . . . .  | 8  |
| 2.6  | Contração da aresta entre $u$ e $v$ . A nova posição, $r$ , é escolhida de forma livre. . . . .   | 8  |
| 2.7  | Contração do <i>half-edge</i> entre $u$ e $v$ , $(u, v) \rightarrow u$ . A contração de um <i>half-edge</i> ocasiona a contração de uma aresta. . . . .   | 9  |
| 2.8  | A aresta $e$ é rotacionada para as extremidades as quais não pertencia. . . . .   | 9  |
| 2.9  | Mapa de profundidade ( <a href="http://en.wikipedia.org/wiki/Depth_map">http://en.wikipedia.org/wiki/Depth_map</a> ). . . . .   | 10 |
| 2.10 | Processo de voxelização de um volume geométrico. . . . .  | 11 |
| 2.11 | A distância de Hausdorff do conjunto de $S$ até $S'$ . (a) A maior distância (f) dentre as menores (b–e). . . . .   | 11 |
| 2.12 | <i>Ray casting</i> : Para cada <i>pixel</i> da imagem um raio é lançado. Em seguida o raio é amostrado e as propriedades ópticas do volume são calculadas. . . . .  | 12 |
| 2.13 | Reformatação multiplanar: (a) os planos são definidos para (b) realizar a reformatação multiplanar. . . . .   | 13 |

|      |  |    |
|------|--|----|
| 3.1  | Auto-interseção: (a) global e (b) local (triângulo tendendo a um ponto) . . . . .  | 15 |
| 3.2  | O inraio $\mathbf{r}$ e o circunraio $\mathbf{r}'$ das circunferências inscrita e circunscrita de um triângulo ABC. . . . .  | 17 |
| 3.3  | Maneiras de contrair uma face triangular: (a) contração de AB, (b) contração de BC e (c) contração de CA. . . . .  | 18 |
| 3.4  | Sequência de contrações que prejudicam a qualidade da malha. . . . .   | 18 |
| 3.5  | Sequência de contrações que preservam a qualidade da malha. . . . .  | 18 |
| 3.7  | Contração de um vértice da borda para o interior da malha em (a) não é permitida (b) de $(\mathbf{F}, \mathbf{B}) \rightarrow \mathbf{F}$ , mas é permitido (c) de $(\mathbf{B}, \mathbf{F}) \rightarrow \mathbf{B}$ . . . . .   | 19 |
| 3.8  | Algoritmo de simplificação: (a) 3192 faces, (b) 1206 faces, (c) 364 faces e (d) 206 faces. . . . .   | 20 |
| 3.9  | Algoritmo de geração de malhas de <i>offset</i> utilizando o algoritmo de simplificação com uma estratégia local. As malhas são mostradas com maior detalhe pela Figura 3.11 . . . . .   | 21 |
| 3.10 | Algoritmo de geração de malhas de <i>offset</i> utilizando o algoritmo de simplificação com uma estratégia global. As malhas são mostradas com maior detalhe pela Figura 3.12. . . . .   | 21 |
| 3.11 | Algoritmo de simplificação local: analisando o custo de contração de todas as faces degeneradas, as contrações são realizadas do menor custo para o maior. (a) 450 faces e 13 faces degeneradas. (b) 6 faces removidas e 444 faces restantes. (c) 444 faces e 8 faces degeneradas. (d) 11 faces removidas e 433 faces restantes. (e) 433 faces e 3 faces degeneradas. (f) 6 faces removidas e 427 faces restantes. (g) 427 faces e 5 faces degeneradas. (h) 3 faces removidas e 424 faces restantes. (i) 424 faces e 70 faces degeneradas. (j) 48 faces removidas e 376 faces restantes. (k) 376 faces e 84 face degenerada. (l) 45 faces removidas e 331 faces restantes. . . . . | 23 |
| 3.12 | Algoritmo de simplificação global: analisando o custo de contração de todas as faces, as contrações são realizadas do menor custo para o maior. (a) 450 faces e 13 faces degeneradas. (b) 210 faces removidas e 240 faces restantes. (c) 240 faces e 37 faces degeneradas. (d) 103 faces removidas e 137 faces restantes. (e) 137 faces e 33 faces degeneradas. (f) 54 faces removidas e 83 faces restantes. (g) 83 faces e 12 faces degeneradas. (h) 36 faces removidas e 47 faces restantes. (i) 47 faces e 12 faces degeneradas. (j) 23 faces removidas e 24 faces restantes. (k) 24 faces e 1 face degenerada. (l) 9 faces removidas e 15 faces restantes. . . . .             | 24 |
| 3.13 | Planos ortogonais ao objeto adquirem os mapas de profundidade de diferentes visões. . . . .  | 25 |
| 3.14 | O valor de $x$ é verificado se está entre os valores dos mapas de profundidade $\mathbf{x}_1$ e $\mathbf{x}_2$ . . . . .   | 25 |
| 3.15 | O algoritmo de simplificação é somente aplicado quando alguma face degenerada é encontrada. (a) A malha é deslocada. (b) Quando uma face degenerada é encontrada na malha, (c) o algoritmo de simplificação é o utilizado. . . . .   | 27 |

|      |  |    |
|------|--|----|
| 3.16 | Deslocamento da malha na direção dos vetores normais dos vértices. Três malhas são apresentadas: malha com profundidade 0mm, malha com profundidade 32mm, e malha com profundidade 64,5mm. . . . .   | 28 |
| 3.17 | Reformatação curvilínea sem artefatos. . . . .   | 29 |
| 3.18 | A malha $M_1$ , representa a malha antes da simplificação, e a malha $M_2$ representa a malha depois da simplificação. Algumas regiões se distanciam mais do que outras regiões durante a simplificação. . . . .   | 30 |
| 3.19 | Malhas antes e depois da simplificação. (a) 458 faces. (b) 188 faces removidas e 270 faces restantes. (c) 270 faces. (d) 111 faces removidas e 159 faces restantes. (e) 159 faces. (f) 70 faces removidas e 89 faces restantes. (g) 89 faces. (h) 40 faces removidas e 49 faces restantes. (i) 49 faces. (j) 21 faces removidas e 28 faces restantes. (k) 28 faces. (l) 11 faces removidas e 17 faces restantes. . . . . | 30 |
| 4.1  | Costura de malhas. (a) A primeira região é demarcada, (b) em seguida demarcamos outra região. (c) As duas malhas representam a superfície demarcada. (d) As duas malhas são costuradas, formando uma só malha. . . . .   | 33 |
| 4.2  | Malhas: (a) $M_1$ , (b) $M_2$ e (c) malhas sobrepostas. . . . .  | 35 |
| 4.3  | Mapas de profundidade: (a) da malha $M_1$ , (b) da malha $M_2$ e (c) sobrepostos. . . . .  | 36 |
| 4.4  | Vizinhos do vértice $v$ que estão na borda, $v_{ant}$ e $v_{prox}$ . . . . .   | 37 |
| 4.5  | Vértices de transição externos: (a) $v_{ex1}$ e (b) $v_{ex2}$ . . . . .  | 37 |
| 4.6  | Vértices de transição. (a) Vértices de transição externo da malha $M_1$ , $v_{ex1}$ e $v_{ex2}$ . (b) Vértices de transição interno da malha $M_2$ , $v_{in1}$ e $v_{in2}$ . . . . .   | 38 |
| 4.7  | Vértices de transição internos: (a) $v_{in1}$ e (b) $v_{in2}$ . . . . .  | 38 |
| 4.8  | Faces da malha $M_2$ removidas e frentes de ambas as malhas extraídas. . . . .   | 40 |
| 4.9  | Processo de ligação intercalado. (a) Um vértice da frente da malha $M_1$ é escolhido para ligar com (b) um vértice da malha $M_2$ . (c) Após serem ligados, (d) o próximo vértice da malha $M_1$ é selecionado para ser (e) ligado com o vértice da malha $M_2$ . Esse procedimento continua até que todos os vértices tenham sido ligados. . . . .  | 41 |
| 4.10 | Duas frentes das malhas, $M_1$ e $M_2$ , sendo costuradas como mencionado na Figura 4.9. O processo é interrompido quando ocorre o cruzamento das arestas. (a) Frentes de ambas as malhas $M_1$ e $M_2$ . (b) Cruzamento das arestas durante o processo de costura. . . . .  | 41 |
| 4.11 | Contraindo o vértice $v$ . (a) A distância do vértice $v$ até todos os outros vértices da frente externa é calculada. (b) A menor distância é encontrada. (c) O vértice $v$ é então contraído. . . . .   | 42 |
| 4.12 | Contração de dois vértices. . . . .  | 42 |
| 4.13 | Vértices de um contorno do buraco. O vizinho a esquerda de $v_1$ , é o vértice $v_5$ , e o vizinho a direita, é o vértice $v_2$ . . . . .  | 44 |
| 4.14 | Pontos do contorno do buraco sendo projetados no plano. . . . .  | 44 |
| 4.15 | Processo de preenchimento do buraco formado durante a ligação da malha. . . . .  | 45 |
| 4.16 | Nova malha formada a partir da costura das malhas $M_1$ e $M_2$ . . . . .  | 45 |



|      |   |    |
|------|---|----|
| 4.17 | Resultado final de uma sequência de costura de malhas. (a) Malha sobre a cabeça é o resultado de uma sequência de costuras. (b) Visualização somente da malha sobre a cabeça. . . . .   | 46 |
| 4.18 | Sequência de costura entre malhas. . . . .  | 47 |
| 4.19 | Costura entre duas malhas. Malha $M_1$ , em vermelho, Malha $M_2$ , em azul. . . . .  | 48 |
| 4.20 | Extração da superfície da cabeça. . . . .   | 49 |
| 4.21 | Impressão 3D da superfície obtida através do algoritmo de costura. (a) Modelo não aramado. (b) Modelo aramado. . . . .  | 49 |
| 4.22 | Superfície extraída, (imagem à esquerda), a partir da cabeça, (imagem à direita). . . . .   | 49 |
| 4.23 | Demarcações: (a) com dois pontos de interseção e (b) com mais de dois pontos de interseção. . . . .   | 50 |
| 4.25 | Cruzamento de arestas durante o preenchimento do buraco. O triângulo com vértices, $(v_{k-1}, v_k, v_{k+1})$ , é formado pois o menor ângulo interno encontrado foi entre as arestas $(v_k, v_{k-1})$ e $(v_k, v_{k+1})$ . Contudo a formação desse triângulo, ocasiona o cruzamento de arestas. . . . .                        | 50 |
| 4.26 | Faces degeneradas formadas durante a costura das malhas. . . . .  | 51 |
| 5.1  | Interface de VMTK. . . . .  | 54 |
| 5.2  | Arquitetura de VMTK. . . . .  | 54 |
| 5.3  | Nova arquitetura de VMTK após as alterações na reformatação curvilínea e inserção do módulo de interação costura. . . . .   | 55 |
| 5.4  | Procedimento da reformatação curvilínea: (a) importação da neuroimagem, (b) remoção de ruídos, (c) demarcação da região de interesse e (d) criação da malha. (e) Em seguida uma nova região é demarcada. (f) Novamente a malha é criada. (g) As duas malhas são costuradas e (h) a reformatação curvilínea é realizada. . . . . | 56 |
| 5.5  | Reformatação curvilínea sendo realizada em toda parte superior da cabeça. Profundidade: (a) 16 mm, (b) 23 mm e (c) 33 mm. . . . .   | 57 |
| 5.6  | Reformatação curvilínea sendo realizada em toda parte superior da cabeça. Profundidade: (a) e (d) 0mm, (b) e (e) 25mm e (c) e (f) 35mm. . . . .   | 57 |
| 5.7  | Reformatação curvilínea sendo realizada na parte posterior da cabeça. Profundidade: (a) 15mm, (b) 23 mm e (c) 31mm. . . . .   | 58 |
| 5.8  | Reformatação curvilínea sendo realizada na parte frontal da cabeça. Profundidade: (a) 12mm, (b) 18mm, (c) 24mm, (d) 22mm, (e) 30mm e (f) 38mm. . . . .  | 58 |
| 5.9  | Suspeita de lesão de displasia cortical focal: (a) vista do topo do cérebro usando a reformatação curvilínea, (b) em corte sagital, (c) em corte coronal e (d) em corte axial. . . . .  | 59 |
| 5.10 | Suspeita de lesão de displasia cortical focal: (a) vista do lado direito, (b) lado esquerdo, (c) e topo da cabeça após a reformatação curvilínea. (d) Reformatação multiplanar . . . . .  | 60 |
| 5.11 | Suspeita de lesão de displasia cortical focal: (a) vista do lado direito, (b) lado esquerdo, (d) e topo da cabeça após a reformatação curvilínea. (c) Reformatação multiplanar . . . . .  | 61 |

|  |    |
|--|----|
| 5.12 Superfície da cabeça que apresenta alguns picos devido a não suavidade da superfície. . . . .   | 61 |
| 5.13 A geometria do escalpo se diferencia do envoltório do córtex cerebral. (a) A malha, uma aproximação da geometria do escalpo, é deslocada. (b) Quando a malha tangência o envoltório do córtex cerebral, existem algumas regiões da malha que ainda não tangenciaram o envoltório. . . . . | 62 |



# Lista de Tabelas

|     |  |    |
|-----|--|----|
| 3.1 | Tempo gasto para simplificar as 6 malhas. Tempo total: 0,379s. . . . .   | 27 |
| 3.2 | Distância das malhas apresentadas pela Figura 3.19. . . . .  | 29 |
| 3.3 | Tempos gastos para voxelizar as malhas de <i>offset</i> da Figura 3.17. Tempo total da voxelização: 6,685s. Tempo total do algoritmo de geração de malhas de <i>offset</i> : 7,064s. . . . . | 32 |
| 4.1 | Tempo gasto para costurar as Malhas #1 e #2. . . . .   | 47 |
| 5.1 | Tempo gasto para realizar a reformatação curvilínea. Tempo médio: 24,5s. . . .   | 55 |



## Introdução

A displasia cortical focal é uma malformação no desenvolvimento cortical, sendo esta a causa mais comum de epilepsia refratária (Kabat & Król 2012). A epilepsia refratária é resistente ao tratamento medicamentoso e na maioria dos casos o procedimento recomendado é a remoção da área com a displasia. A epilepsia afeta de 3% a 5% da população em geral e aproximadamente 25% a 30% apresentam quadro refratário (Meneses, Hertz, Gruetzmacher, Blattes, Silva, Vosgerau, Laroça & Kowacs 2006). O avanço das técnicas de neuroimagem fornece uma avaliação mais precisa das lesões patológicas. As imagens de ressonância magnética são uma importante ferramenta para a localizações dessas lesões, que são caracterizadas principalmente pela presença de neurônios anormais no córtex (Abdel Razek, Kandell, Elsorogy, Elmongy & Basett 2009). Visualmente essa área é associada a um borramento na transição córtico-subcortical, como mostra a Figura 1.1. Contudo, existem casos em que as lesões são sutis e não são facilmente visíveis através das clássicas reformatações multiplanares (Lee & Kim 2013)(Kabat & Król 2012). Em decorrência disso, a extensão da área atingida fica difícil de ser delineada, o que pode resultar na remoção incompleta de um foco epileptogênico. Isso é uma das causas do não sucesso cirúrgico (Bastos, Comeau, Andermann, Melanson, Cendes, Dubeau, Fontaine, Tampieri & Olivier 1999)(Kabat & Król 2012). A ferramenta de reformatação curvilínea é uma solução

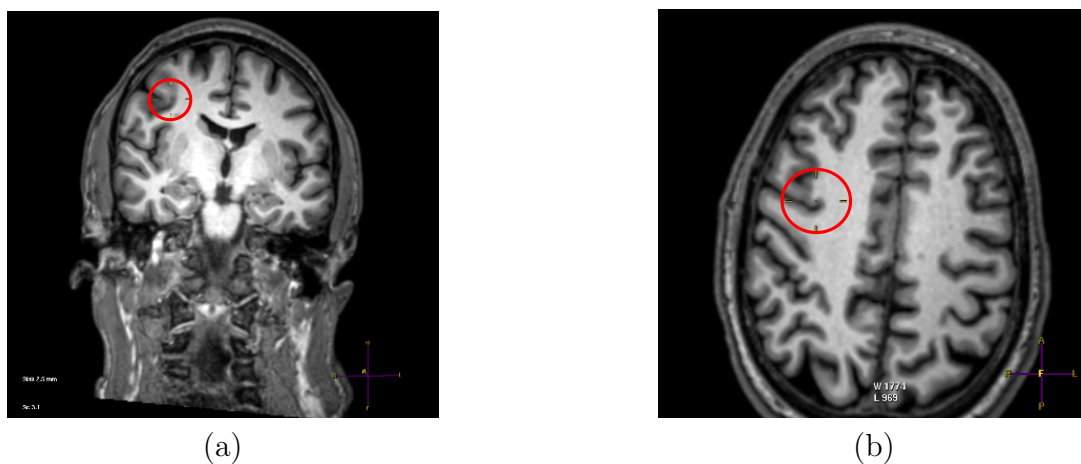


Figura 1.1: Suspeita de lesão de displasia cortical focal destacada.

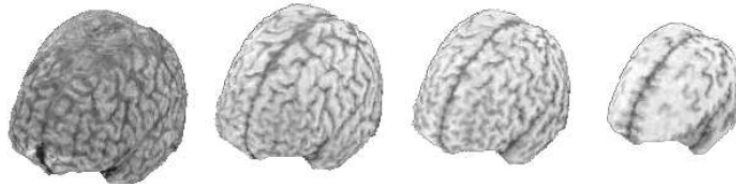


Figura 1.2: Reformatação curvilínea realizada por Bergo e Falcão (Bergo & Falcao 2006).

computacional que permite uma análise não invasiva das camadas paralelas ao escalpo, podendo auxiliar na detecção de tais lesões sutis (Bastos et al. 1999). Ela consiste em uma série de superfícies curvilíneas equidistantes, que são criadas com o deslocamento do envoltório da superfície cortical na direção dos seus vetores normais. A reformatação curvilínea complementa as reformatações multiplanares já existentes.

O *BrainSight* (Bastos et al. 1999) é um *software* que fornece uma ferramenta de reformatação curvilínea interativa. Curvas são obtidas delimitando interativamente o envoltório do córtex numa sequência de planos coronais. Após a demarcação, a reformatação curvilínea é obtida através da interpolação dessas curvas. O tempo requerido para realizar manualmente as demarcações impede o uso dessa ferramenta na rotina clínica (Bastos et al. 1999) (Huppertz, Kassubek, Altenmüller, Breyer & Fauser 2008). Partindo de uma abordagem de processamento de imagens, Bergo e Falcão (Bergo & Falcao 2006), e Huppertz *et al.* (Huppertz et al. 2008) propuseram automatizar o procedimento de reformatação curvilínea. A motivação destes dois trabalhos está no fato de que a reformatação manual disponível em *BrainSight* consome bastante tempo. Bergo e Falcão segmentaram o cérebro aplicando a transformada imagem-floresta, removendo assim a caixa craniana. Depois o seu envelope é extraído e utilizando a transformada de distância euclidiana é calculada uma série de isosuperfícies de várias profundidades a partir deste envelope. A Figura 1.2 apresenta o resultado da reformatação proposta por Bergo e Falcão. Huppertz *et al.* utilizaram máscaras axiais pré-definidas, distanciadas de 2mm entre si, para remover as regiões da caixa craniana do cérebro no espaço estereotáxico antes de realizar a reformatação curvilínea com Matlab-SPM (Friston, Ashburner, Kiebel, Nichols & Penny 2007). A vantagem de ambas as técnicas está na realização automática da reformatação do cérebro como todo, o que facilita um diagnóstico baseado na análise comparativa dos dois hemisférios. Diferentemente das propostas de reformatação curvilínea interativa, o procedimento realizado por estes dois trabalhos aplica técnicas computacionais bem distintas das clássicas reformatações multiplanares no espaço nativo. Isso dificulta a unificação dos dois tipos de reformatação em um mesmo ambiente, de forma a propiciar atenção visual (Ware 2004). As técnicas utilizadas na reformatação automatizada apresentam distintos parâmetros de controle, como o tamanho do elemento estruturante no caso da proposta de (Bergo & Falcao 2006), e o tamanho da máscara utilizada no trabalho de (Huppertz et al. 2008). Apesar delas garantirem a realização da reformatação sem artefatos, em decorrência da baixa resolução das demarcações nos procedimentos manuais, é importante que as reformatações multiplanares e curvilíneas sejam integradas em um mesmo ambiente para facilitar a exploração comparativa das estruturas cerebrais.

Como mostrado em (Wu, Yasuda & Cendes 2012), com uma interface apropriada em cima da tecnologia GPU é possível diminuir o tempo consumido no procedimento manual de demarcação

das amostras. Partindo da premissa de que a geometria do escalpo da cabeça se aproxima da geometria do envoltório cortical, Wu *et al.* (Wu et al. 2012) propuseram uma ferramenta de reformatação curvilínea interativa que, diferente do *BrainSight*, proporciona ao usuário uma visão 3D da neuroimagem, permitindo que este realize a demarcação com a pintura da área do escalpo, como ilustra a Figura 1.3(a). Com isso, evita-se o pré-processamento de remoção da caixa craniana. Após a demarcação, a superfície é amostrada e a partir dessas amostras uma malha triangular é criada (Figura 1.3(b)). A malha é, então, deslocada na direção dos vetores normais dos seus vértices (Figura 1.3(c)) gerando uma série de malhas de *offset*. Com estas malhas de *offset* pode-se reformatar uma neuroimagem 3D em uma série de cortes curvilíneos. A Figura 1.3(d) mostra um corte curvilíneo obtido com o procedimento.

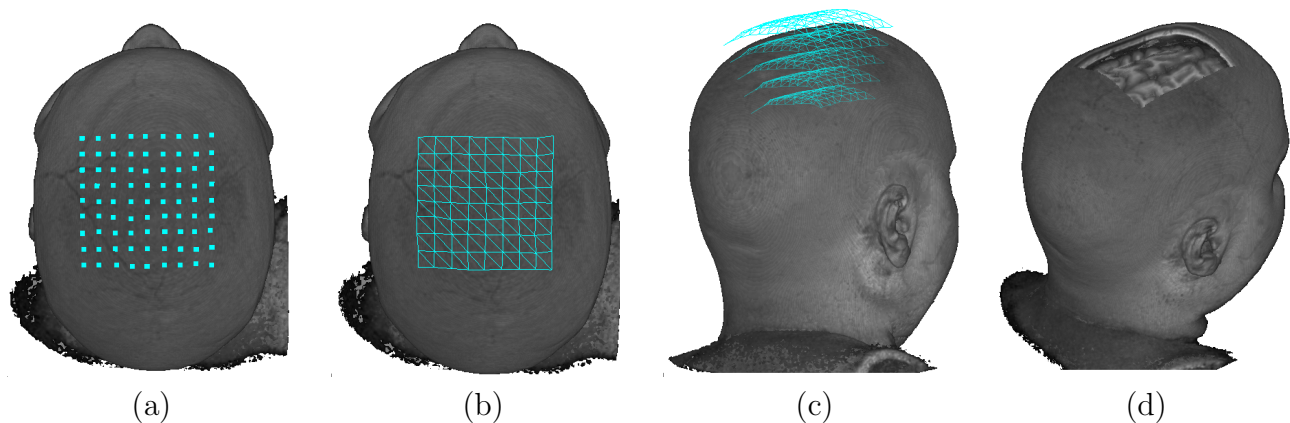


Figura 1.3: Reformatação curvilínea: (a) amostragem, (b) criação da malha, (c) deslocamento da malha e (d) resultado final da reformatação curvilínea.

## 1.1 Motivação

A principal motivação deste trabalho é desenvolver um ambiente de exploração visual interativo, em que as reformatações, multiplanares e curvilínea, sejam integradas em um mesmo ambiente. Conjeturou-se que, mantendo sempre o mesmo contexto seja possível proporcionar ao usuário uma forma mais natural de interações para ajudar na localização de lesões sutis de displasia cortical focal. Além disso, as reformatações curvilíneas são realizadas de forma local, preservando a caixa craniana como referência do espaço nativo do paciente. Isso abriria a perspectiva das potenciais aplicações da nossa ferramenta, como por exemplo o diagnóstico visual através de neuroimagens que é complementado com EEG<sup>1</sup> e a preservação da caixa craniana que pode facilitar o procedimento de posicionamento dos eletrodos sobre o escalpo de um paciente. Outras aplicações que foram vislumbradas são o neuroplanejamento e a neuronavegação.

---

<sup>1</sup>Eletroencefalografia



## 1.2 Proposta

A proposta desse trabalho é desenvolver um algoritmo de reformatação curvilínea, livre de artefatos e com suporte a cortes de maior extensão, baseada na reformatação curvilínea proposta por Wu *et al.* (Wu et al. 2012). Diferente da reformatação interativa proposta por Bastos *et al.* (Bastos et al. 1999), as demarcações não são feitas nas fatias e sim sobre o volume. As demarcações são realizadas somente na região de interesse e a reformatação é realizada somente na região demarcada, mantendo a caixa craniana como referência do espaço nativo. Assim o contexto da reformatação curvilínea pode ser mantido o mesmo em que são realizadas as reformatações multiplanares.

Contudo, a reformatação proposta por (Wu et al. 2012) pode, durante a reformatação, apresentar artefatos (Figura 1.4). Esses artefatos surgem devido à degeneração da geometria da malha durante a sua geração. O algoritmo não consegue tratar de forma satisfatória as faces que se degeneram nas malhas de *offset*. Além disso, o algoritmo permite a demarcação somente de uma região sobre o volume, não permitindo cortes de maior extensão e uma análise comparativa entre os dois hemisférios.

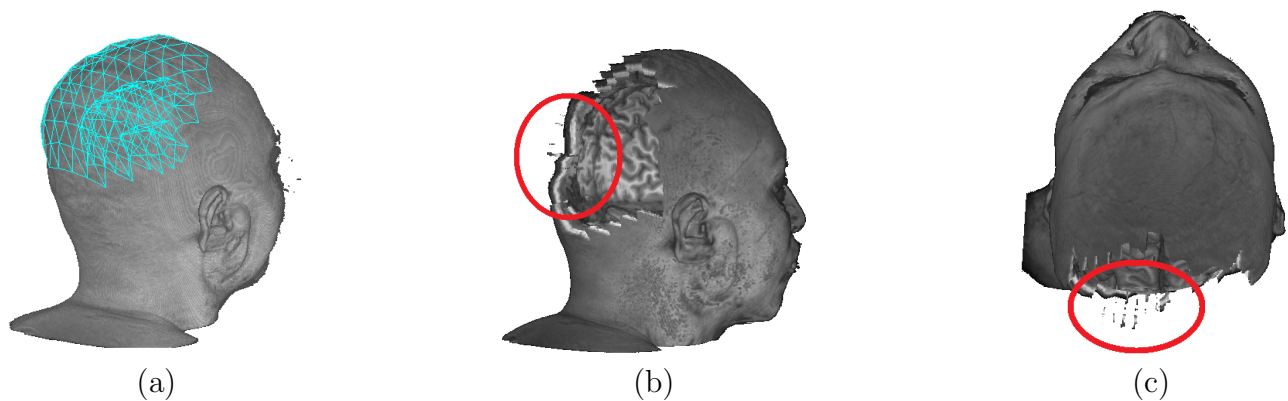


Figura 1.4: Reformatação curvilínea com artefatos: (a) visualização das malhas deslocadas sobre o volume, (b) e (c) visualização do corte curvilíneo com os artefatos.

## 1.3 Objetivos

O principal objetivo desse trabalho é proporcionar a exploração das neuroimagens de forma a dar melhor suporte aos médicos para identificar lesões corticais sutis. Para proporcionar a realização desse objetivo propõe-se:

1. aprimorar o algoritmo de geração de malhas de *offset*, proposto em (Wu et al. 2012), e
2. estender a aquisição de amostras de um ponto de vista à aquisição de múltiplos pontos de vista.

## 1.4 Estrutura do Texto

Este trabalho está organizado da seguinte forma: no Capítulo 2 descreveu-se conceitos preliminares que serão utilizados no decorrer do texto; nos Capítulos 3 e 4 foram apresentados os algoritmos de geração de malhas de *offset* e o algoritmo de costura de malhas. Ao final de cada capítulo, foram discutidos os algoritmos implementados e os resultados obtidos; no Capítulo 5 os resultados da reformatação curvilínea foram apresentados e, finalmente, no Capítulo 6 conclui-se o trabalho com uma discussão dos resultados obtidos e sugestões de trabalhos futuros.

## Preliminares

Esse capítulo tem como objetivo introduzir alguns conceitos que serão utilizados no decorrer do texto desta dissertação.

### 2.1 Malhas de *Offset*

A malha é uma coleção de vértices, arestas e faces que define uma forma geométrica. Uma malha triangular  $M$  consiste de componentes geométricos e topológicos, representados pelo conjunto de vértices:

$$V = \{v_1, \dots, v_V\},$$

e um conjunto de faces triangulares conectadas

$$F = \{f_1, \dots, f_F\}, f_i \in V \times V \times V, \quad (2.1)$$

onde cada triângulo é especificado pelos seus vértices de  $V$  (Botsch, Pauly, Rossl, Bischoff & Kobbelt 2006). Uma malha de *offset*  $M$  é obtida pelo deslocamento de  $M$  em relação ao vetor

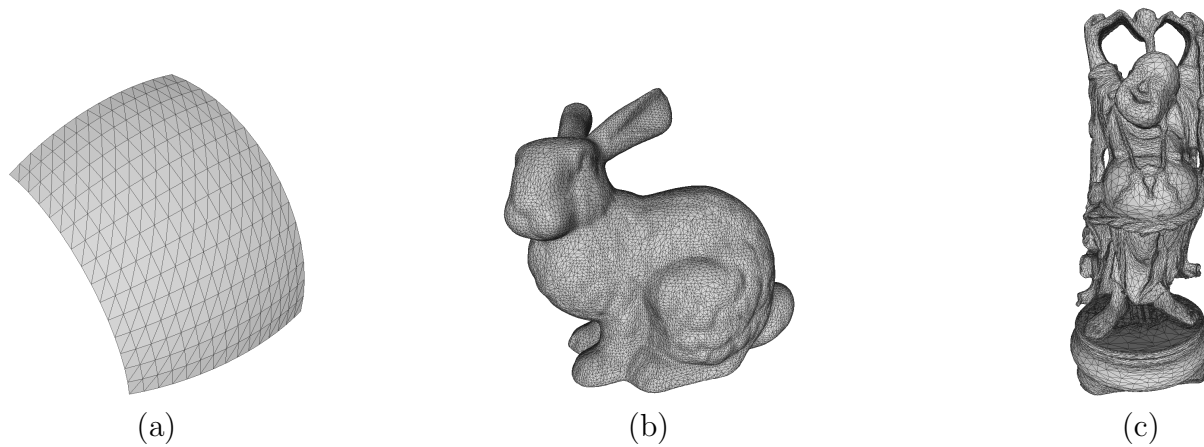


Figura 2.1: Exemplos de malhas.

normal de seus vértices, ou seja, para cada vértice  $v_j$  de  $M$  determinamos o vértice  $v'_j \in M_i$  por

$$v'_j = v_j + t \vec{n}_j, \quad (2.2)$$

onde  $\vec{n}_j$  é o vetor normal no vértice em  $v_j$ . A Figura 2.2 ilustra um conjunto de malhas de *offset*  $M_i$  da malha  $M$ .

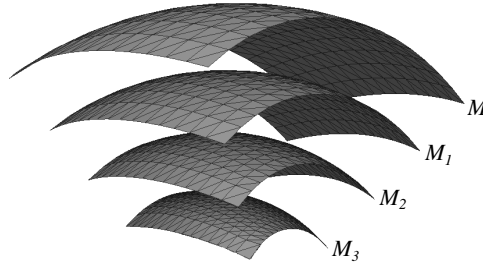


Figura 2.2: Geração das malhas de *offset*  $M_i$  a partir de  $M$ .

## 2.2 Estrutura de Dados *Half-Edge*

Uma malha pode ser representada utilizando listas de adjacência de: vértices com faces indexadas, faces ou de arestas. Contudo essas representações não introduzem informações topológicas, ou seja, não fornecem informações sobre a conectividade entre os elementos geométricos necessários para reconstruir um modelo 3D. Uma outra forma de representar uma malha é utilizando um modelo de representação de fronteira. O modelo B-Rep, acrônimo de *Boundary Representation*, modela um objeto geométrico através de sua fronteira (Stroud 2006). O modelo separa dois tipos de informação: geométrica e topológica. Posição espacial, vetor normal ou curvamento são exemplos de informações geométricas, enquanto a conectividade entre as partes, como a construção de um triângulo a partir dos vértices e a construção de uma malha a partir dos triângulos, é conhecida por topologia. Em princípio toda informação topológica pode ser inferida a partir da informação geométrica se esta estiver completa. A redundância facilita, no entanto, acessos aos mais diversos dados a partir de qualquer elemento, propiciando manipulações interativas a custo de uma ocupação maior de memória. As principais classes topológicas são: sólidos, faces, arestas e vértices. Elas guardam entre si a relação de fronteira. Os sólidos são delimitados por um conjunto de faces que formam uma superfície fechada. As faces tem como borda uma sequência fechada de arestas que, por sua vez, tem dois vértices como pontos extremos. O *winged-edge*, proposto por Baumgart (Baumgart 1972), *half-edge*, proposto por Mantyla (Mantyla 1988) e *quad-edge*, proposto por (Guibas & Stolfi 1985) são as mais conhecidas estruturas de dados para armazenar de forma eficiente essas relações. A estrutura de dados *half-edge* e *quad-edge* é uma variação da *winged-edge*. A Figura 2.3 apresenta as estrutura de dados *winged-edge*, *half-edge* e *quad-edge*.

Diferente do *winged-edge*, a estrutura de dados *half-edge* divide uma aresta em duas arestas orientadas, fazendo algumas operações serem aplicada com mais eficiência. Cada aresta é dividida em dois *half-edges* com direções opostas, tal que todo *half-edge* dentro da face da malha

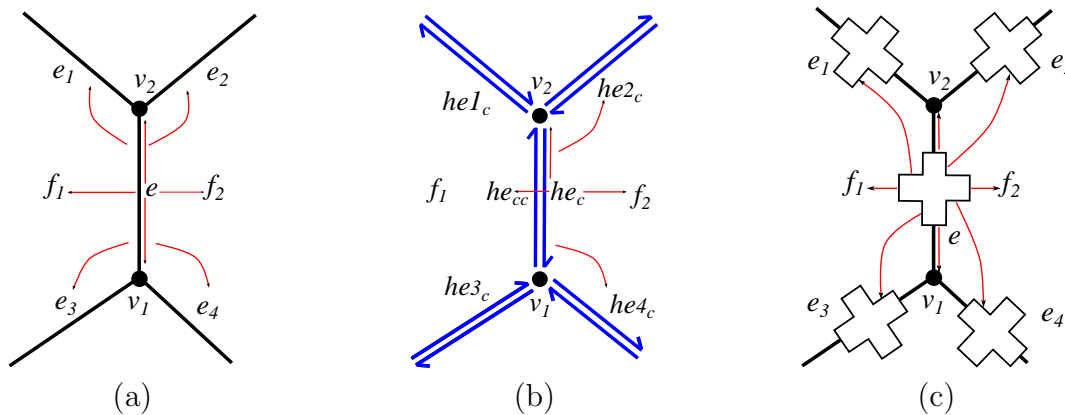


Figura 2.3: Estruturas de dados (a) *winged-edge*, (b) *half-edge* e (c) *quad-edge*. Na estrutura de dados *winged-edge*, a aresta  $e$ , armazena informações sobre seus vértices extremos,  $v_1$  e  $v_2$ , a face à esquerda e à direita,  $f_1$  e  $f_2$ , e as arestas  $e_1, e_2, e_3$  e  $e_4$ . A estrutura de dados de *half-edge*,  $he_c$  armazena uma referência para o vértice de origem,  $v_2$ , a face adjacente,  $f_2$  e o próximo,  $he4_c$ , anterior,  $he2_c$ , e o oposto,  $he_{cc}$  *half-edge*. A estrutura de dados *quad-edge* possui os mesmos ponteiros da estrutura de dados *winged-edge*.

tenha uma orientação anti-horária (Botsch et al. 2006). A Figura 2.4 ilustra como uma malha triangular é representada numa estrutura de *half-edge*. O *quad-edge* possui os mesmos ponteiros que *winged-edge*, porém, ele fornece a vantagem de manipular os dados de forma uniforme e de extrair o modelo dual utilizando poucas operações (Guibas & Stolfi 1985). Contudo, como o *winged-edge*, essa estrutura de dados armazena muitas referências, sendo mais custosa a sua manutenção. Optou-se então por utilizar a estrutura de dados *half-edge*.



Figura 2.4: Malha triangular representada numa estrutura de *half-edge*. (a) As arestas da malha triangular, (b) são representadas por duas arestas direcionadas na estrutura de *half-edge*.

## 2.3 Algoritmo de Simplificação

Algoritmos de simplificação são utilizados para diminuir o número de faces de uma malha. O procedimento de remoção das faces é definido pelo critério de qualidade, que procura preservar propriedades específicas da malha original. Exemplos de critérios de qualidade são as distâncias geométricas (por exemplo, distância de Hausdorff) ou aparência visual (por exemplo, preservação

de características ou cores). Os métodos de simplificação podem ser classificados em (Botsch et al. 2006):

1. algoritmos de agrupamento de vértices;
2. algoritmos incrementais;
3. algoritmos de reamostragem.

A primeira classe de algoritmos consiste em agrupar os vértices em um número menor de vértices representantes. Eles são eficientes e robustos, contudo a qualidade da malha não é sempre satisfatória. Algoritmos incrementais, por sua vez, removem um vértice ou uma aresta incrementalmente. Na maioria dos casos produzem malhas com qualidade, porém sua complexidade computacional pode chegar a  $O(n^2)$ . Técnicas de reamostragem distribuem novas amostras sobre a geometria da superfície e, em seguida, elas são conectadas. Problemas como *aliasing* podem ocorrer caso as amostras não sejam perfeitamente distribuídas de acordo com as características da geometria original. No presente trabalho foi utilizado um algoritmo incremental para realizar a simplificação da malha. A cada passo o melhor candidato (vértice) a ser removido é escolhido baseado no critério de qualidade estabelecido. As operações topológicas para a remoção de vértices podem ser:

1. Remoção do vértice: O vértice  $v$  é removido e a área é retriangulada novamente (Figura 2.5).



Figura 2.5: Remoção do vértice  $v$ .

2. Contração de aresta: A aresta entre  $u$  e  $v$  é contraída e ambos os vértices são movidos para uma nova posição  $r$ . A nova posição do vértice é escolhida de forma livre (Figura 2.6). Para que a topologia da malha seja preservada, esta contração deve satisfazer a



Figura 2.6: Contração da aresta entre  $u$  e  $v$ . A nova posição,  $r$ , é escolhida de forma livre.

condição de conexão (*link condition*). No caso de uma malha triangular  $M$ , definimos

como conexão (*link*) de uma aresta  $ab$ , o conjunto de vértices que geram triângulos com a aresta  $ab$ ,

$$Link_{ab} = \{x \in M \mid abx \in M\},$$

e a conexão de um vértice  $a$ , como o conjunto de vértices que geram arestas com  $a$  e de arestas que geram triângulos com  $a$ ,

$$Link_a = \{x, xy \in M \mid ax, axy \in M\}.$$

A condição de conexão é satisfeita se  $Link_{ab} = Link_a \cap Link_b$  (Dey, Edelsbrunner, Guha & Nekhayev 1998).

3. Contração de *half-edge*: A contração de *half-edge* é um caso particular de contração de aresta (Botsch et al. 2006), em que o vértice  $v$  é deslocado para a posição do vértice  $u$  (Figura 2.7).



Figura 2.7: Contração do *half-edge* entre  $u$  e  $v$ ,  $(u, v) \rightarrow u$ . A contração de um *half-edge* ocasiona a contração de uma aresta.

Adicionalmente, para preservar a condição de Delaunay que consiste em manter os dois ângulos opostos de dois triângulos adjacentes menor ou igual a  $180^\circ$ , a técnica de giro de aresta (*flip*) pode ser utilizada. No giro de aresta a aresta  $e$ , que tem extremidades nos vértices  $v$  e  $w$ , depois do giro, passa a ter suas extremidades nos vértices  $u$  e  $t$  (Figura 2.8).



Figura 2.8: A aresta  $e$  é rotacionada para as extremidades as quais não pertencia.

## 2.4 Mapa de Profundidade

Na computação gráfica um mapa de profundidade é uma imagem que informa a distância relativa do objeto da cena até a tela onde ele é projetado. O mapa de profundidade pode ser adquirido utilizando o algoritmo de *Z-Buffer*. Quando um objeto é renderizado a profundidade

do objeto projetado no *pixel* é armazenado no espaço de memória denominado *z-buffer*. Assim quando outro objeto é renderizado, a profundidade desse novo objeto é comparado com a profundidade do objeto renderizado anteriormente no *z-buffer*. Caso o valor de profundidade do *pixel* esteja mais próximo do observador, ou seja, o seu valor é menor do que o valor armazenado do *pixel* anterior, o conteúdo do *z-buffer* é substituído pelo valor de profundidade do *pixel* atual. Vale ressaltar que o valor da profundidade é, usualmente, normalizado entre  $[0,1]$  e a sua precisão depende da razão  $z_{far}/z_{near}$ , onde  $z_{far}$  e  $z_{near}$  são, respectivamente, distâncias dos planos de recorte posterior e frontal do volume de visão (OpenGL 2014). Quanto mais próximo o objeto estiver do observador, o valor da profundidade será mais próximo de zero, e quanto mais distante, o valor da profundidade será mais próximo de um.



Figura 2.9: Mapa de profundidade ([http://en.wikipedia.org/wiki/Depth\\_map](http://en.wikipedia.org/wiki/Depth_map)).

A Figura 2.9(b) é o mapa de profundidade da Figura 2.9(a) ao mapear os valores de profundidade normalizados em níveis de cinza. Quanto mais próxima a amostra está do observador, o seu valor de profundidade fica próximo de zero. Portanto, o *pixel* em que ela está mapeada é colorido com um nível de cinza escuro. Por outro lado, quando a amostra está distante do observador, o seu *pixel* recebe uma cor mais clara (Figura 2.9(b)).

## 2.5 Voxelização

A voxelização consiste na geração de uma representação de enumeração espacial (Foley, van Dam, Feiner & Hughes 1990) a partir de uma forma geométrica e ela é uma discretização volumétrica de um objeto (Dong, Chen, Bao, Zhang & Peng 2004). Cada elemento do volume discretizado é chamado de *voxel*. Em vários casos, a voxelização é necessária a fim de se aplicar os algoritmos orientados a representações de enumerações espacial (Karabassi, Papaioannou & Theoharis 1999)(Eisemann & Décoret 2006). A principal ideia da maioria dos algoritmos de voxelização é tomar como ponto de partida um espaço discretizado em *voxels* e examinar se cada *voxel* pertence ou não ao objeto de interesse, e em seguida rotulá-lo (Karabassi et al. 1999). A Figura 2.10 apresenta a voxelização de um objeto. Ele é imerso num reticulado regular de dimensão pré-definido para descobrir quais dos seus *voxels* pertencem ao objeto, como mostra a Figura 2.10(b). A Figura 2.10(c) ilustra o resultado da seleção dos *voxels* que pertencem ao objeto. Assim, temos um objeto discretizado na Figura 2.10(d).



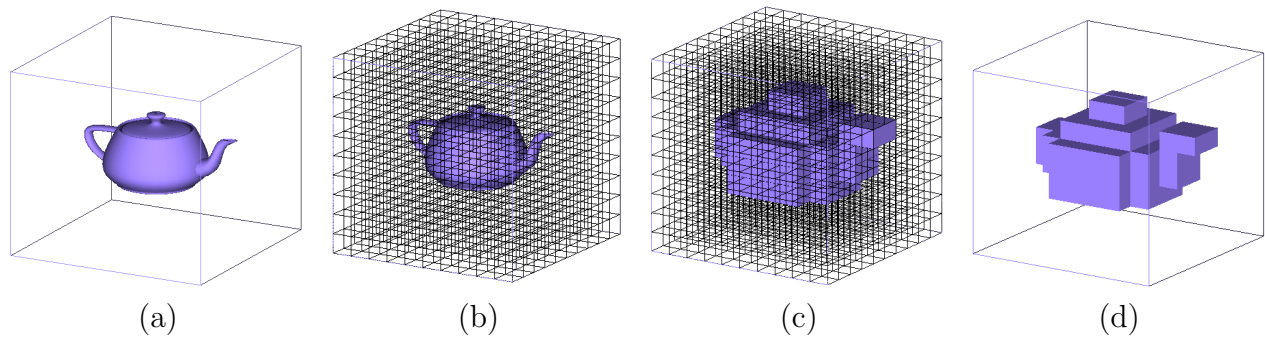


Figura 2.10: Processo de voxelização de um volume geométrico.

## 2.6 Distância de Hausdorff

A distância de Hausdorff é a máxima distância de um conjunto até o ponto mais próximo de um outro conjunto. Dados dois conjuntos  $S$  e  $S'$  na Figura 2.11(a), a distância de Hausdorff é a máxima distância de um conjunto até o ponto mais próximo de um outro conjunto. Isso pode ser obtido nos seguintes passos: 1) para o primeiro ponto de  $S$  é calculada a distância entre ele até todos os pontos do conjunto  $S'$  (Figura 2.11(b)) e a menor distância é escolhida (Figura 2.11(c)); 2) o mesmo é realizado para o outro ponto, sua distância é calculada e depois a menor distância é escolhida (Figuras 2.11(d) e (e)); 3) finalmente, a maior distância entre as menores distâncias dos pontos do conjunto de  $S$  até  $S'$ , ou seja, a distância de Hausdorff de  $S$  até  $S'$ , é determinada. O mesmo procedimento é realizado do conjunto de  $S'$  até  $S$ . A distância de Hausdorff entre os dois conjuntos,  $S$  e  $S'$ , é a maior das duas distâncias computadas.

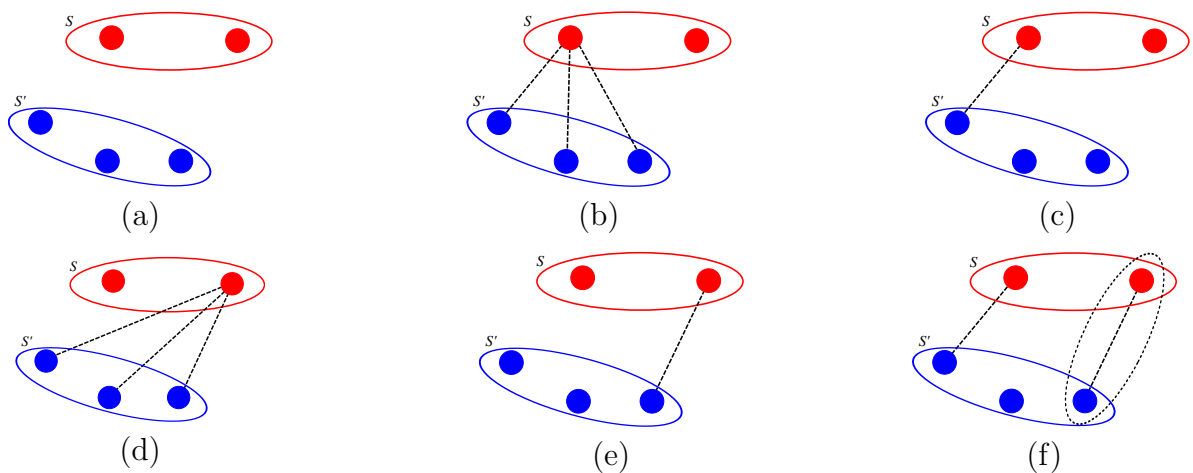


Figura 2.11: A distância de Hausdorff do conjunto de  $S$  até  $S'$ . (a) A maior distância (f) dentre as menores (b–e).

Defina a distância  $d(p, S')$  entre o ponto  $p$  que pertence a superfície  $S$ , até uma superfície  $S'$  como:

$$d(p, S') = \min_{p' \in S'} \|p - p'\|, \tag{2.3}$$

onde  $\| \cdot \|$  denota a norma Euclidiana. A partir dessa definição, a distância de Hausdorff entre  $S$

e  $S'$ , denotada por  $d(S, S')$  é dada por:

$$d(S, S') = \max_{p \in S} [d(p, S')]. \quad (2.4)$$

Contudo note que essa distância não é simétrica,  $d(S, S') \neq d(S', S)$ . Assim, a distância simétrica de Hausdorff é definida como:

$$d_s(S, S') = \max[d(S, S'), d(S', S)]. \quad (2.5)$$

A distância de Hausdorff entre duas malhas triangulares,  $M = (V, T)$  e  $M' = (V', T')$  onde  $V$  e  $V'$  são os conjuntos de vértices e  $T$  e  $T'$  são os conjuntos de triângulos, é denotada como:

$$d_s(M, M') = \max[d(M, M'), d(M', M)]. \quad (2.6)$$

A métrica de Hausdorff não precisa que o tamanho dos conjuntos sejam iguais, ou seja, não precisa satisfazer a correspondência 1:1 entre os elementos dos conjuntos (Aspert, Santa-Cruz & Ebrahimi 2002).

## 2.7 Ray Casting

O *ray casting* é uma técnica de renderização volumétrica orientada a imagem (Hadwiger, Kniss, Rezk-salama, Weiskopf & Engel 2006). Para cada *pixel* da imagem, um raio é lançado para dentro do volume e o volume de dados é amostrado ao longo deste raio. E, para cada amostra, são calculadas suas propriedades ópticas; mais especificamente a cor e a opacidade da amostra. Estas propriedades são combinadas para gerar a cor final do *pixel*. O processo básico de *ray casting* é ilustrado na Figura 2.12. Diversas variações deste algoritmo básico foram propostas; uma delas é o uso de um volume de controle adicional capaz de modificar a forma como as amostras são consideradas e/ou como as propriedades ópticas são combinadas (Hadwiger et al. 2006).

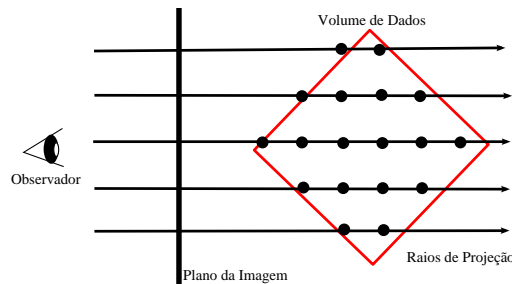


Figura 2.12: *Ray casting*: Para cada *pixel* da imagem um raio é lançado. Em seguida o raio é amostrado e as propriedades ópticas do volume são calculadas.

## 2.8 Reformatações

As imagens médicas 3D são, de fato, um volume de fatias 2D orientadas na direção da sua aquisição. Usualmente a direção de aquisição é perpendicular ao plano sagital, axial ou coronal

do paciente. A reformatação consiste, então, em refatiar o volume reconstruído por estas fatias em outras direções. Pode-se dividi-la em dois grupos: multiplanares e curvilíneas. As reformatações multiplanares refatiam o volume reconstruído em planos. Os planos refatiados podem ser paralelos (Figura 2.13(b)) ou oblíquos (Figura 2.13(c)) ao volume reconstruído (Figura 2.13(a)). As reformatações curvilíneas permitem reformatar o volume reconstruído em superfícies curvas.

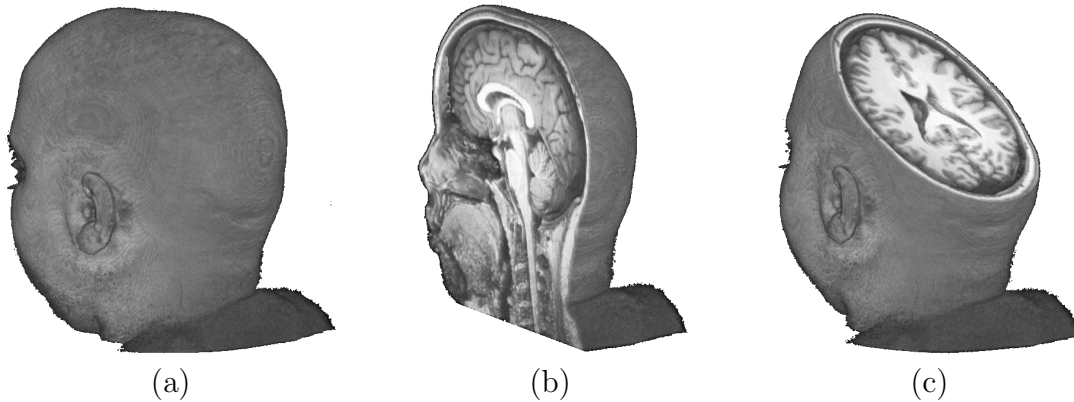


Figura 2.13: Reformatação multiplanar: (a) os planos são definidos para (b) realizar a reformatação multiplanar.

As Figuras 1.3 e 1.2 apresentam, respectivamente, uma reformatação curvilínea de uma região do cérebro e uma de todo o cérebro.

## 2.9 Algoritmo de Reformatação Curvilínea

Nessa seção uma versão reduzida do algoritmo de reformatação curvilínea proposto por (Wu et al. 2012) será descrita. Utilizando a técnica de renderização, *ray casting*, obtêm-se o mapa

---

### Algoritmo 1: Algoritmo de Reformatação Curvilínea

---

- 1 **início**
  - 2     Gerar mapa de profundidade (*Ray Casting*)
  - 3     Demarcar região visível
  - 4     Amostrar a região demarcada
  - 5     Construir malha a partir das amostras
  - 6     Gerar e voxelizar malhas de *offset*
  - 7 **fim**
- 

de profundidade do volume (linha 2). Em seguida a região a ser reformataada, utilizando o *mouse*, é demarcada e amostrada para formar a malha triangular (linha 3-5). A demarcação é realizada no espaço de tela, assim o mapa de profundidade é utilizado para levar as amostras para o espaço de mundo, deste modo as amostras ficam sobre o escalpo. Para cada nova malha de *offset* gerada, ela é voxelizada (linha 6). Um volume de controle de visibilidade, onde cada *voxel* contém o nível de profundidade em relação ao escalpo, é utilizado para que seja decidida de forma interativa quais *voxels* serão visíveis. A Figura 1.3 ilustra esse procedimento.

## Reformatação Curvilínea

Análogo às reformatações multiplanares, uma reformatação curvilínea é uma técnica de pós-processamento que permite o refatiamento de uma imagem 3D em superfícies curvilíneas. Para preservar a referência no espaço nativo do paciente foi introduzido o que chamamos de reformatação curvilínea local, que consiste em reformatar parcialmente uma imagem 3D em superfícies curvilíneas (Wu et al. 2012). O algoritmo proposto é baseado na geração de malhas de *offset*. Ele utiliza contrações locais para evitar as faces degeneradas durante a geração de malhas de *offset*. Contudo, essas contrações podem gerar artefatos, como mostrado na Figura 1.4. Neste capítulo será apresentado uma alternativa de remoção de faces degeneradas em malhas de *offset* que evitem tais artefatos.

### 3.1 Geração de Malhas de *Offset* Usando Algoritmo de Simplificação

A geração de malhas de *offset* é bastante utilizada nas áreas de CAM<sup>1</sup>/CAD<sup>2</sup> (Saeed, de Pennington & Dodsworth 1988) (Pham 1992). Contudo Wu *et al.* (Wu et al. 2012) apresentou uma aplicação de um algoritmo de geração de malhas de *offset* na área médica utilizando-as como ponte entre uma interface com o usuário e um volume de dados discretos. Do lado do usuário, uma malha proporciona uma visão contínua dos dados discretos e do lado de processamento é associada cada amostra manipulada pelo usuário com os *voxels* através da voxelização das malhas.

O principal problema quando se desloca uma malha são as auto-interseções. Durante o deslocamento de uma malha em relação ao seu vetor normal auto-interseções podem ocorrer. Elas podem ser classificadas como globais ou locais. As globais ocorrem entre faces diferentes (Figura 3.1(a)) e as locais quando a face se torna degenerada, ou seja, quando uma face tende a um ponto ou a uma aresta (Figura 3.1(b)). Para que a reformatação curvilínea seja bem sucedida, precisa-se evitar essas auto-interseções ao longo do deslocamento, a fim de manter a consistência geométrica da malha. Como as auto-interseções locais ocorrem quando as áreas das

---

<sup>1</sup> *Computer Aided Manufacturing*

<sup>2</sup> *Computer Aided Design*

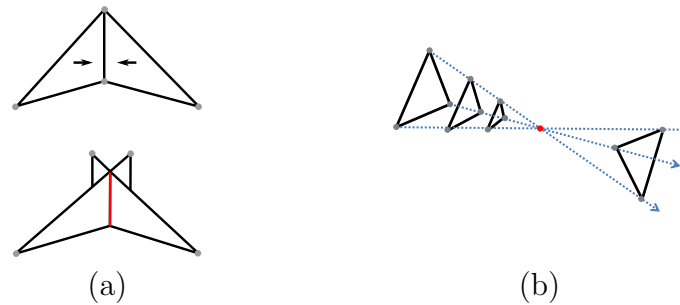


Figura 3.1: Auto-interseção: (a) global e (b) local (triângulo tendendo a um ponto).

faces deslocadas ficam muito pequenas, conjecturou-se que se continuarmos a deslocar uma malha simplificada na direção do vetor normal, as auto-interseções podem ser evitadas. Nesse trabalho propomos criar um algoritmo que explora essa característica. Essencialmente, propomos aplicar um algoritmo de simplificação quando as faces da malha de *offset* não satisfizerem o critério de qualidade pré-estabelecido, a fim de aumentar as suas áreas sem comprometer a geometria da malha.

## 3.2 Trabalhos Relacionados

Jung *et al.* (Jung, Shin & Choi 2004) removem as auto-interseções e as faces degeneradas de uma dada malha triangular fechada através de um algoritmo dividido em 4 passos. No primeiro passo as faces degeneradas são encontradas e removidas. Os operadores topológicos, contração de aresta (*edge collapse*) e giro de aresta (*flip edge*) são utilizados para remover as faces degeneradas. Triângulos com área quase nula são consideradas faces degeneradas. O segundo passo é definir uma região válida. Uma região válida é um conjunto de triângulos que define uma fronteira externa dada pelo deslocamento do volume a partir da malha. Em seguida os triângulos são classificados em 3 grupos: válidos, inválidos e parcialmente válidos. Triângulos válidos não fazem interseção com outros triângulos e estão dentro da região válida, triângulos parcialmente válidos fazem interseção com outros triângulos e estão dentro da região válida e inválida, já os triângulos não válidos são triângulos que estão na região inválida. O terceiro passo é encontrar um triângulo semente. Um triângulo semente pertence a uma região válida que irá formar a região válida inicial. O quarto passo consiste em expandir essa região: os triângulos inválidos são removidos, os triângulos parcialmente válidos são retriangulados e os novos triângulos formados são classificados como válidos e inválidos. Os triângulos classificados como inválidos são ainda removidos antes de terminar o procedimento. As auto-interseções são tratadas em uma etapa de pós-processamento

Yi *et al.* (Yi, Lee & Shin 2008) monitoram a topologia da malha deslocada associando um erro a cada vértice. Esse erro é o somatório das distâncias ao quadrado do vértice deslocado  $\mathbf{v}$  aos planos adjacentes a ele antes do deslocamento, menos o deslocamento  $d$  realizado. Caso esse erro seja maior que um limiar  $t$ , esse vértice é adicionado a uma fila de prioridades, do maior para o menor. Para cada vértice  $\mathbf{v}$  colocado na fila de prioridades, é realizada uma operação de divisão, resultando em dois novos vértices  $\mathbf{v}_0$  e  $\mathbf{v}_1$ . Para cada novo vértice criado, o seu erro é

comparado com o limiar  $t$ . Caso o erro seja maior que  $t$ , tal vértice é também adicionado na fila de prioridades. Essa operação é realizada para cada vértice da fila. Liu *et al.* (Liu & Wang 2011) propuseram um método de deslocamento de superfície que amostra a malha em um *grid* regular. Em seguida, usando a transformada de distância o deslocamento da malha é calculado. Após o deslocamento da malha em relação ao vetor normal de seus vértices, é verificado quais são as partes do *grid* que fazem interseção com a malha. A nova malha é construída encontrando as interseções da superfície deslocada com as arestas do *grid*. Esse método evita as auto-interseções locais e globais, porém o custo para calcular a transformada de distância e para extrair o novo contorno são altos.

Wu *et al.* (Wu et al. 2012), previne as auto-interseções removendo as faces degeneradas através de operações locais. Faces com a área próximo de zero são classificadas como faces degeneradas e são removidas. A operação topológica, contração de arestas, foi utilizada para remover as faces. Contudo essas operações locais geraram problemas na geometria da malha, pois a ordem como essas contrações são realizadas proporcionam diferentes impactos na geometria da malha.

### 3.3 Hipótese

A hipótese do presente trabalho é que, utilizando um algoritmo de simplificação durante o deslocamento da malha e adicionando algumas restrições de contração discutidas na Seção 3.4, possa-se evitar as auto-interseções locais. Em vista de diversos algoritmos eficientes disponíveis para simplificação de malhas o problema poderá ser então reduzido ao problema de aplicação de um algoritmo de simplificação. Tendo as malhas de *offset* será necessário rotular os *voxels* correspondentes para que eles sejam removidos da imagem final de acordo com a profundidade de corte selecionada pelo usuário.

### 3.4 Proposta

A proposta é realizar a geração de malhas de *offset* com um algoritmo de simplificação de malhas para evitar as auto-interseções locais. Ao combinar com um eficiente algoritmo de voxelização que identifica quais *voxels* são ocupados pela malha durante o deslocamento, pode-se gerar um volume de controle com os *voxels* pré-rotulados de valores de profundidade. Isso propicia o desenvolvimento de um ambiente de exploração em tempo interativo, como em (Wu et al. 2012). O algoritmo de simplificação só é utilizado quando uma face degenerada é encontrada. Para encontrar as faces degeneradas a qualidade de cada face é avaliada em cada iteração. A qualidade de uma face é medida pela sua forma. Há diversas alternativas para quantizar este conceito. Algumas delas são:

- *área*;
- *área/menor lado*;
- *inraio/circunraio*.

Essencialmente, elas se diferem nas grandezas geométricas utilizadas. A primeira forma calcula a *área* do triângulo e compara o valor obtido a um limiar  $\delta$ . Caso a área esteja abaixo desse limiar, a face é considerada degenerada. A segunda forma de classificar as faces degeneradas é calculando a razão *área/maior lado*. Da mesma forma um limiar pode ser utilizado para classificar se a face é degenerada ou não. A terceira forma calcula a razão *inraio/circunraio*. O inraio de um triângulo ABC é o raio  $r$  da circunferência inscrita no triângulo ABC e o circunraio de raio  $r'$  é o raio da circunferência que circunscrita o triângulo ABC como ilustra a Figura 3.2. A razão, entre o raio  $r$  e o raio  $r'$ , é comparada com o limiar  $\delta$  para definir se uma face é degenerada ou não. Como deve-se privilegiar triângulos agudos escolheu-se a segunda métrica

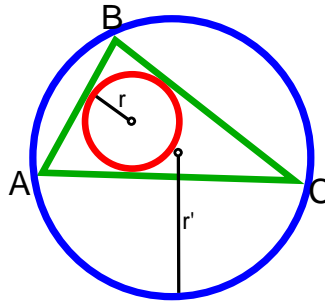


Figura 3.2: O inraio  $r$  e o circunraio  $r'$  das circunferências inscrita e circunscrita de um triângulo ABC.

no lugar da primeira adotada em (Wu et al. 2012)

$$\frac{\text{área da face}}{\text{maior lado}} \quad (3.1)$$

A métrica *inraio/circunraio* apresentou resultados similares, contudo a métrica *área/maior lado* é mais simples de ser calculada. Quando o valor da razão *área/maior lado* começa a diminuir a diferença entre os lados do triângulo começam a aumentar. Quanto menor a razão, maior é a diferença. Uma face é considerada degenerada caso a razão da Equação 3.1 fique abaixo de  $\delta$ . O limiar  $\delta$  é um valor pré-definido: quanto maior for o  $\delta$ , mais faces serão consideradas degeneradas.

Por que remover faces durante a geração de malhas de *offset*? Como já foi mencionado, quando a malha é deslocada suas faces começam a ficar cada vez menores. Essas faces podem causar auto-interseções locais se elas não forem removidas. A realização da reformatação curvilínea pode ficar comprometida, pois artefatos podem aparecer em cortes feitos pelas malhas com auto-interseções como vimos na Figura 1.4. Contudo, quais cuidados são necessários ao se remover uma face? A remoção das faces degeneradas pode ocorrer contraindo seus *half-edges*. Existem três maneiras de se contrair uma face triangular. Dado um triângulo ABC podemos contrair AB (Figura 3.3(a)), BC (Figura 3.3(b)) ou CA (Figura 3.3(c)). A ordem de remoção das faces pode, no entanto, ocasionar diferentes impactos na malha. As Figuras 3.4 e 3.5 mostram duas sequências de contrações usadas para remover as faces degeneradas. A segunda sequência de contrações mostrada na Figura 3.5 preserva mais a qualidade das faces e a geometria do que a primeira sequência de contrações (Figura 3.4). Porém, nem toda contração é permitida. A Figura 3.6 apresenta um exemplo em que a contração do *half-edge* entre os vértices **A** e **B** ocasiona na formação de duas faces com os mesmos vértices, pois a condição de conexão não é

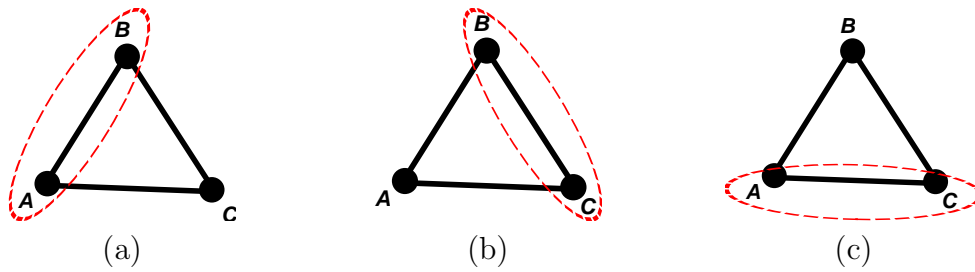


Figura 3.3: Maneiras de contrair uma face triangular: (a) contração de AB, (b) contração de BC e (c) contração de CA.

satisfeita. Além disso, a geometria das contrações de vértices que estão na borda para o interior da malha pode causar um maior impacto na geometria da malha como exemplifica a Figura 3.7.

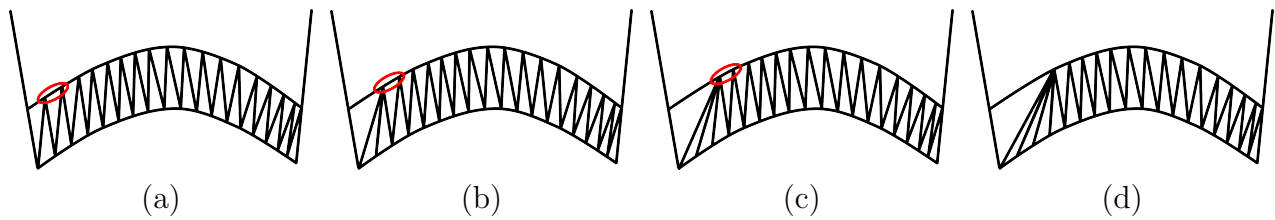


Figura 3.4: Sequência de contrações que prejudicam a qualidade da malha.

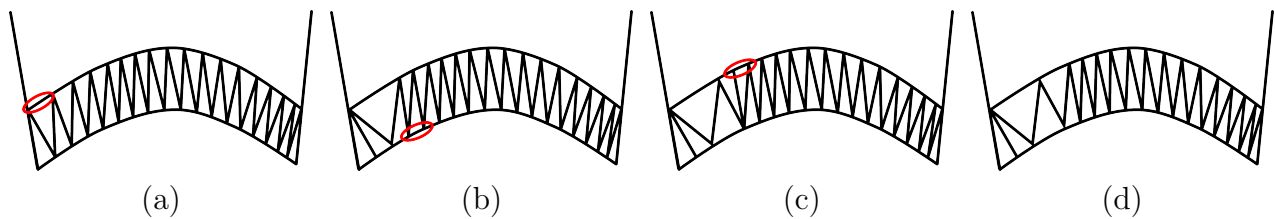


Figura 3.5: Sequência de contrações que preservam a qualidade da malha.

Finalmente, pergunta-se: quais faces devem ser retiradas para impactar o menos possível a geometria da malha? A hipótese do presente trabalho sugere que a utilização dos critérios de qualidade conhecidos em algoritmos de simplificação juntamente com as restrições mencionadas anteriormente torne possível escolher um conjunto de faces que impacte o menos possível a geometria da malha. Propõe-se utilizar como base do trabalho o algoritmo incremental de sim-



Figura 3.6: Operação de contração de aresta proibida.

plicação proposto por Garland e Heckbert (Garland & Heckbert 1997). O critério de qualidade



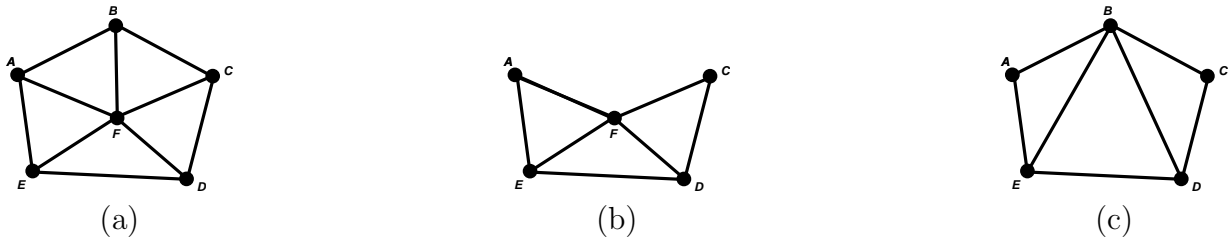


Figura 3.7: Contração de um vértice da borda para o interior da malha em (a) não é permitida (b) de  $(\mathbf{F}, \mathbf{B}) \rightarrow \mathbf{F}$ , mas é permitido (c) de  $(\mathbf{B}, \mathbf{F}) \rightarrow \mathbf{B}$ .

adotado por Garland e Heckbert prioriza a geometria da malha, ou seja, minimizar o erro entre o modelo original e o modelo simplificado. Para contemplar as restrições de contrações discutidas nesta seção, introduz-se uma etapa de filtragem excluindo-as do processo de simplificação.

### 3.4.1 Contrações para Simplificar

O algoritmo proposto é baseado na decimação via contração de pares de vértices para remover uma face. As contrações dos pares de vértices são realizados contraindo as arestas. Seja uma aresta formada pelo par de vértices  $(\mathbf{v}_i, \mathbf{v}_j)$  a contração desse par,  $(\mathbf{v}_i, \mathbf{v}_j) \rightarrow \bar{\mathbf{v}}$ , resulta mover o vértice  $\mathbf{v}_i$  e  $\mathbf{v}_j$  para uma nova posição  $\bar{\mathbf{v}}$ . A ordem para realizar as contrações dos pares em cada iteração é definida de acordo com os custos da contração. Portanto, antes de realizar alguma contração, os custos de contração de todos os vértices de uma malha são calculados, filtrados e ordenados. O custo da contração de um par  $(\mathbf{v}_i, \mathbf{v}_j) \rightarrow \mathbf{v}_i$  é o somatório das distâncias ao quadrado de  $\mathbf{v}_i$  até todos os planos  $\mathbf{p}$  adjacentes a  $\mathbf{v}_j$ . Dado  $\epsilon$  um limiar pré-estabelecido que garante que as contrações com um custo muito alto não sejam aceitas. Se o custo da contração for maior do que  $\epsilon$  ou se a contração não for permitida essa contração é descartada. Somente são ordenados na ordem crescente os custos de contração dos vértices restantes. As contrações cujos custos são menores são realizadas primeiro. Quanto menor o custo, menor o impacto na geometria.

Uma matriz  $4 \times 4$  simétrica  $\mathbf{Q}$  é associada a cada vértice. A matriz  $\mathbf{Q}$  é o somatório da matriz fundamental quadrática  $\mathbf{K}_p$ ,

$$\mathbf{Q} = \sum_{\mathbf{p} \in \text{planos}(\mathbf{v})} \mathbf{K}_p \quad (3.2)$$

e representa o somatório dos coeficientes dos planos  $\mathbf{p} = [a \ b \ c \ d]^T$  adjacentes ao vértice  $\mathbf{v} = [x \ y \ z \ 1]^T$ . Cada plano  $\mathbf{p}$  é definido pela equação  $ax + by + cz + d = 0$  com  $a^2 + b^2 + c^2 = 1$ .

A matriz  $\mathbf{K}_p$  pode ser usada para encontrar a distância ao quadrado de qualquer ponto no espaço até o plano  $\mathbf{p}$

$$\mathbf{K}_p = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}.$$

Assim, calcular o custo da contração de um par  $(\mathbf{v}_i, \mathbf{v}_j) \rightarrow \mathbf{v}_i$  se resume em calcular

$$C(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i^T (\mathbf{Q}_i + \mathbf{Q}_j) \mathbf{v}_i, \quad (3.3)$$

onde  $\mathbf{Q}_i$  é o somatório dos coeficientes dos planos adjacentes a  $\mathbf{v}_i$  e  $\mathbf{Q}_j$  é o somatório dos coeficientes dos planos adjacentes a  $\mathbf{v}_j$ .

O **Algoritmo 2** resume o algoritmo de simplificação usado na reformatação curvilínea. Para cada vértice a matriz  $\mathbf{Q}$  é calculada (linha 2). Em seguida o custo da contração é calculado usando a Equação 3.3 (linha 3). Após os custos das contrações serem calculados, todos os custos que estiverem abaixo do erro  $\epsilon$  e que forem permitidos são adicionados à fila de prioridades (linha 4). O primeiro par da fila é retirado e a contração é realizada. Após a contração do par  $(\mathbf{v}_i, \mathbf{v}_j)$ , o vértice  $\mathbf{v}_j$  é removido, restando apenas o vértice  $\mathbf{v}_i$ . Todos os pares que contenham o vértice retirado  $\mathbf{v}_j$  são removidos da fila e a matriz  $\mathbf{Q}$  dos vértices que eram adjacentes ao vértice  $\mathbf{v}_j$  são atualizados (linhas 5-9). Vale ressaltar que o erro  $\epsilon$  é estimado em cada iteração de acordo com os custos de contração das faces degeneradas. Para evitar excessivas contrações sempre é setado como  $\epsilon$  o maior custo de contração.

---

**Algoritmo 2:** Algoritmo de Simplificação

---

**Entrada:** Malha  $M$  com  $n$  faces e erro  $\epsilon$

**Saída:** Malha  $M$  com  $m$  faces,  $m < n$

```

1 início
2   Iniciar a matriz  $\mathbf{Q}$  para todos os vértices
3   Para cada vértice  $\mathbf{v}_i$  calcule o custo de contração dele aos seus vértices adjacentes  $\mathbf{v}_j$ ,
   e escolha o menor
4   Adicione na fila de prioridades todas as contrações permitidas com custos abaixo de  $\epsilon$ ,
   do menor para o maior
5   enquanto Fila não vazia faça
6     Realize a contração  $(\mathbf{v}_i, \mathbf{v}_j)$ 
7     Remova o par  $(\mathbf{v}_i, \mathbf{v}_j)$  da fila
8     Atualize o custo de cada par da fila que era adjacente ao vértice removido
9   fim enquanto
10 fim
```

---

A Figura 3.8 ilustra a diminuição do número de faces de um modelo usando o algoritmo de simplificação descrito. Repare que as áreas das faces aumentam depois da simplificação. As imagens foram geradas pelo aplicativo *MeshLab* (MeshLab 2014).

### 3.4.2 Local ou Global?

Deve-se considerar todas as faces da malha ou somente as faces degeneradas para decidir quais contrações devem ser feitas? Há duas formas de se aplicar o algoritmo de simplificação: local ou global. No paradigma local somente as faces degeneradas são candidatas às contrações, enquanto na global, todas as faces são sujeitas às contrações. Inicialmente os custos das contrações são calculados somente para as faces degeneradas e depois as contrações na ordem crescente de custos são realizadas. A Figura 3.9 apresenta o resultado da simplificação da malha utilizando uma estratégia local. As malhas simplificadas são ilustradas em maior escala na Figura 3.11. No paradigma global, ao invés de olhar apenas para as faces degeneradas, analisa-se todas as faces. Os custos das contrações são calculados não só das faces degeneradas, mas de todas as

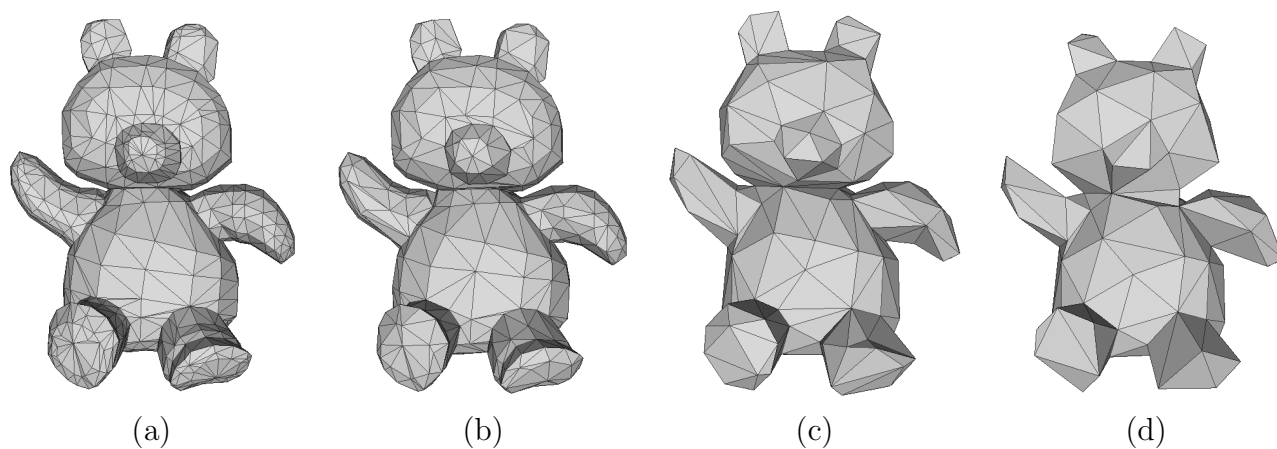


Figura 3.8: Algoritmo de simplificação: (a) 3192 faces, (b) 1206 faces, (c) 364 faces e (d) 206 faces.

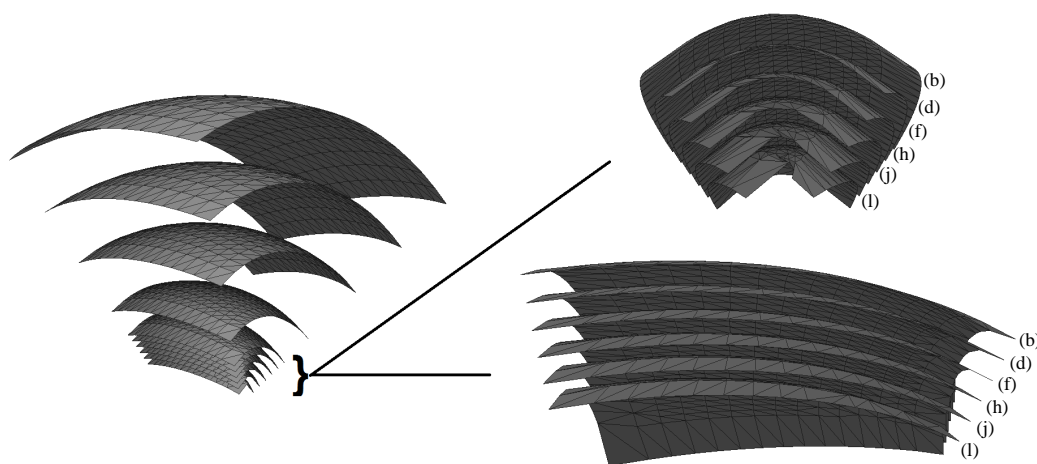


Figura 3.9: Algoritmo de geração de malhas de *offset* utilizando o algoritmo de simplificação com uma estratégia local. As malhas são mostradas com maior detalhe pela Figura 3.11

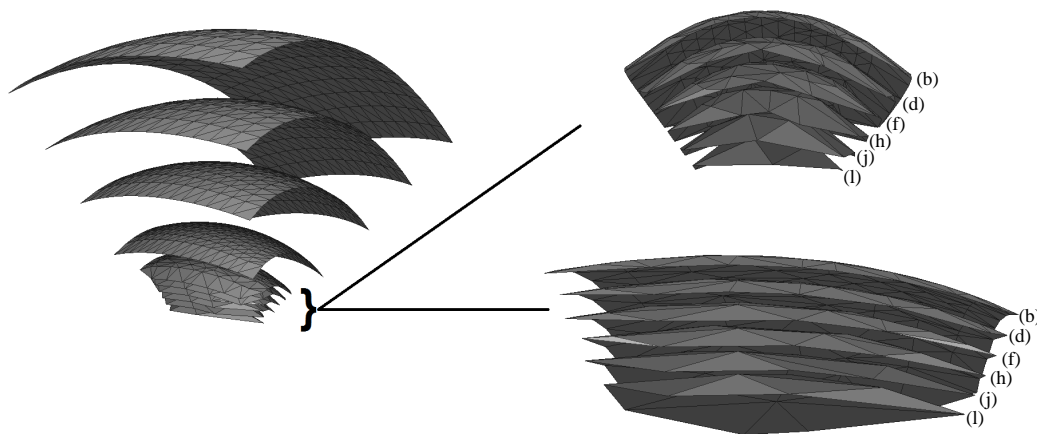


Figura 3.10: Algoritmo de geração de malhas de *offset* utilizando o algoritmo de simplificação com uma estratégia global. As malhas são mostradas com maior detalhe pela Figura 3.12.

faces. A Figura 3.10 apresenta o resultado da simplificação da malha utilizando uma estratégia global. As malhas simplificadas são ilustradas em maior escala pela Figura 3.12. A malha que considerou uma abordagem local teve menos faces removidas permanecendo mais suave do que a malha que utilizou uma abordagem global. Porém, ela não conseguiu evitar as auto-interseções locais para o mesmo deslocamento definido para a estratégia global, pois as contrações permitidas não foram suficientes para remover as faces degeneradas. Apesar da abordagem global ser menos suave, devido a remoção de mais faces, ela conseguiu evitar as auto-interseções locais, como mostrado na Figura 3.11. Por esse motivo optou-se por uma abordagem global.

### 3.4.3 Algoritmo de Voxelização

Sob o ponto de vista geométrico, obter a reformatação curvilínea com uma série de malhas de *offset* é análogo a obter reformatações planares com uma série de planos: basta visualizar somente os *voxels* de um dos semi-espacos separados por cada malha ou por cada plano. Porém, sob o ponto de vista computacional, determinar um semi-espaco dividido por uma malha de geometria arbitrária tem um custo bem maior do que determinar um semi-espaco dividido por um plano. Em (Wu et al. 2012) foi proposta uma alternativa para que a exploração do volume de interesse ocorra em tempo interativo: pré-processar os *voxels* do volume de interesse, rotulando-os com os valores de profundidade da malha de *offset* que os intersectam e utilizar este volume de rótulos como volume de controle do avanço de raios do algoritmo de *ray-casting* na etapa de renderização.

Diferente da implementação em CPU proposta em (Wu et al. 2012), foi utilizado no presente trabalho um algoritmo de voxelização baseado na proposta de (Karabassi et al. 1999) por ser facilmente implementado com os recursos da GPU. Karabassi *et. al.* utilizou seis mapas de profundidade para classificar qual *voxel* pertence ou não ao objeto. Esses mapas,  $\mathbf{x}_1$ ,  $\mathbf{x}_2$ ,  $\mathbf{y}_1$ ,  $\mathbf{y}_2$ ,  $\mathbf{z}_1$  e  $\mathbf{z}_2$ , são obtidos através dos planos ortogonais ao objeto (Figura 3.13). Os mapas de profundidade tem tamanho  $N \times N$ , assim a resolução do grid será de  $N^3$ . A posição do *voxel*  $v(i, j, k)$  é normalizada para o intervalo dos valores do mapa de profundidade

$$(x, y, z) = (i, j, k) \cdot \frac{\text{profundidade}_{max} - \text{profundidade}_{min}}{N - 1},$$

onde  $(x, y, z)$  é a posição do *voxel* normalizada,  $(i, j, k)$  é a posição do *voxel* não normalizada e  $\text{profundidade}_{max}$  e  $\text{profundidade}_{min}$  são o maior e o menor valor do mapa de profundidade. Para cada *voxel* do *grid*,  $v(i, j, k)$ , é verificada a sua pertinência ao objeto. O *voxel* só pertence ao objeto se e somente se:

$$\begin{aligned} \mathbf{x}_1(i, j) &\leq x \leq \mathbf{x}_2(j, k) \text{ e} \\ \mathbf{y}_1(k, i) &\leq y \leq \mathbf{y}_2(k, i) \text{ e} \\ \mathbf{z}_1(j, i) &\leq z \leq \mathbf{z}_2(j, i). \end{aligned}$$

O valor de  $x$  é verificado se está dentro do intervalo dos mapas de profundidade  $\mathbf{x}_1$  e  $\mathbf{x}_2$  (Figura 3.14). Depois o mesmo acontece para o valor de  $y$  e  $z$ , mas ao invés dos mapas  $\mathbf{x}_1$  e  $\mathbf{x}_2$  são usados os mapas  $\mathbf{y}_1$  e  $\mathbf{y}_2$ , e  $\mathbf{z}_1$  e  $\mathbf{z}_2$ .

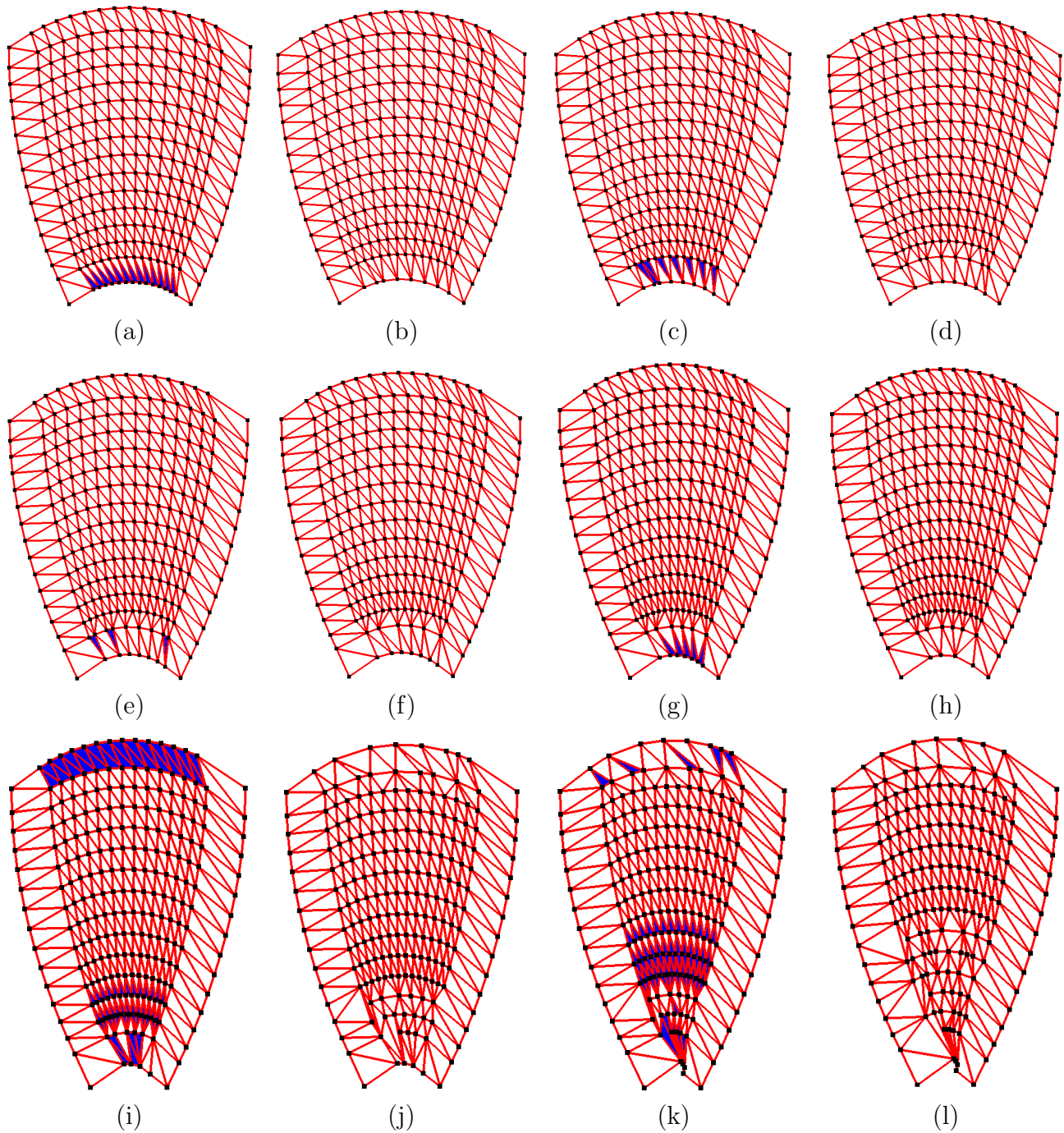


Figura 3.11: Algoritmo de simplificação local: analisando o custo de contração de todas as faces degeneradas, as contrações são realizadas do menor custo para o maior. (a) 450 faces e 13 faces degeneradas. (b) 6 faces removidas e 444 faces restantes. (c) 444 faces e 8 faces degeneradas. (d) 11 faces removidas e 433 faces restantes. (e) 433 faces e 3 faces degeneradas. (f) 6 faces removidas e 427 faces restantes. (g) 427 faces e 5 faces degeneradas. (h) 3 faces removidas e 424 faces restantes. (i) 424 faces e 70 faces degeneradas. (j) 48 faces removidas e 376 faces restantes. (k) 376 faces e 84 face degenerada. (l) 45 faces removidas e 331 faces restantes.

Duas modificações foram realizadas no algoritmo original de Karabassi *et. al.*. A primeira modificação foi no número de mapas de profundidade. Foram utilizados cinco mapas:  $\mathbf{x}_1$ ,  $\mathbf{x}_2$ ,

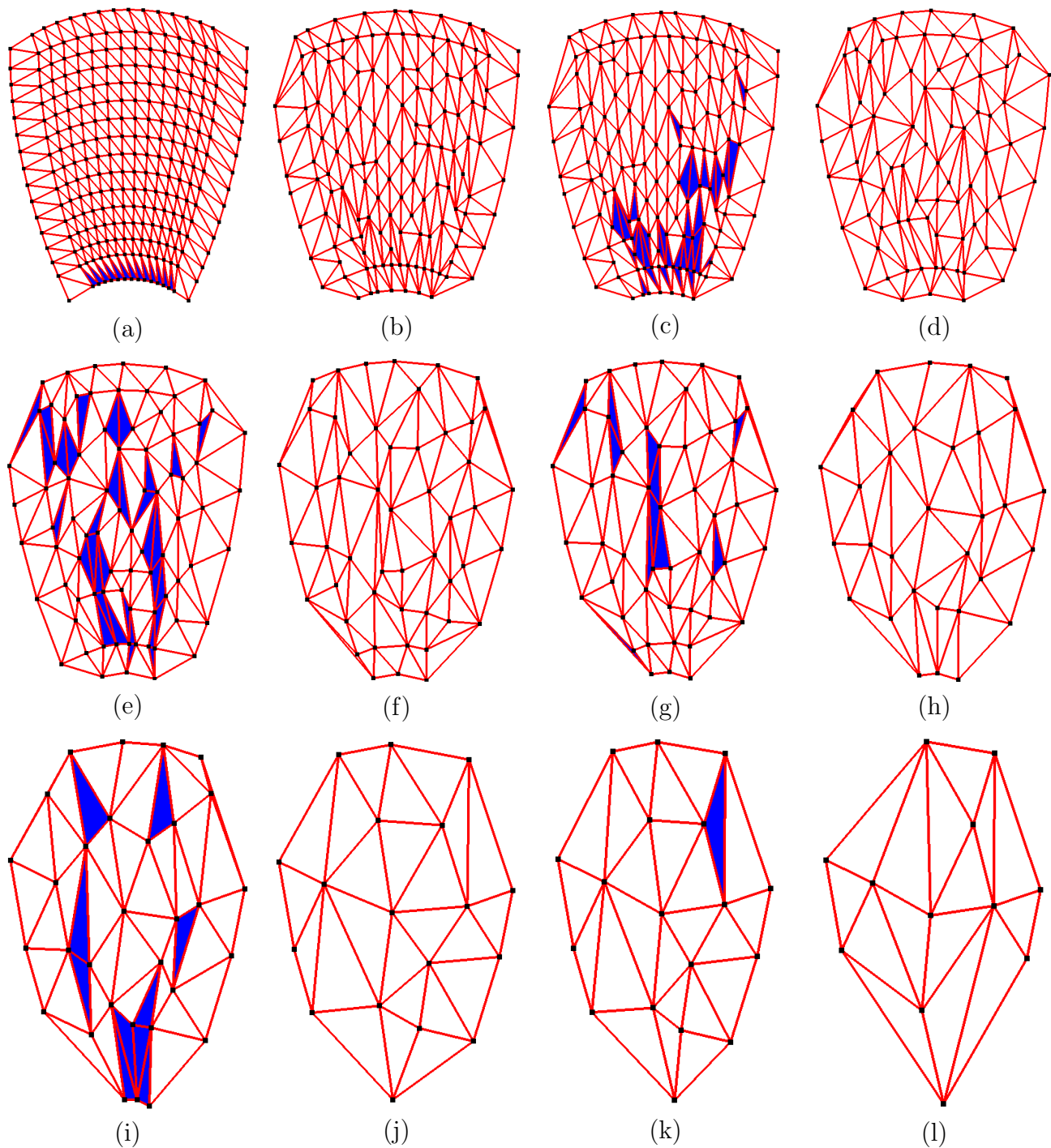


Figura 3.12: Algoritmo de simplificação global: analisando o custo de contração de todas as faces, as contrações são realizadas do menor custo para o maior. (a) 450 faces e 13 faces degeneradas. (b) 210 faces removidas e 240 faces restantes. (c) 240 faces e 37 faces degeneradas. (d) 103 faces removidas e 137 faces restantes. (e) 137 faces e 33 faces degeneradas. (f) 54 faces removidas e 83 faces restantes. (g) 83 faces e 12 faces degeneradas. (h) 36 faces removidas e 47 faces restantes. (i) 47 faces e 12 faces degeneradas. (j) 23 faces removidas e 24 faces restantes. (k) 24 faces e 1 face degenerada. (l) 9 faces removidas e 15 faces restantes.

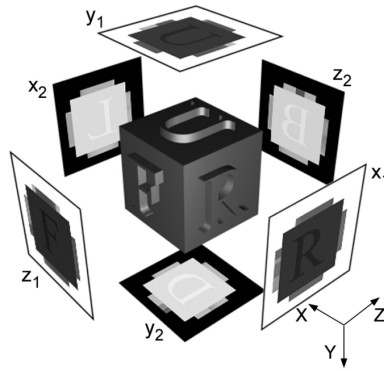


Figura 3.13: Planos ortogonais ao objeto adquirem os mapas de profundidade de diferentes visões (Karabassi et al. 1999).

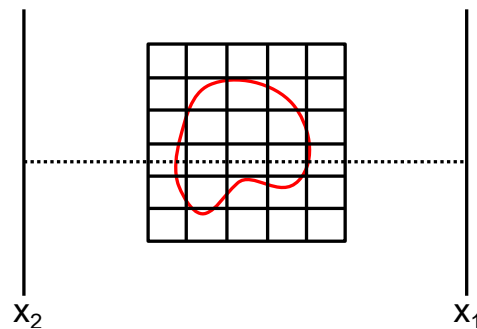


Figura 3.14: O valor de  $x$  é verificado se está entre os valores dos mapas de profundidade  $x_1$  e  $x_2$ .

$y_1$ ,  $z_1$  e  $z_2$ . Reduziu-se o número de mapas, pois essa quantidade já era o suficiente para voxelizar a superfície que queríamos. A segunda modificação foi na maneira de como classificar os *voxels*. Ao invés de percorrer todos os *voxels* do *grid*,  $N^3$  posições, percorram-se os mapas de profundidade,  $5N^2$  posições. Cada *pixel* que pertença ao objeto é projetado no *grid*. Os valores das coordenadas projetadas são truncados e a pertinência do *voxel* ao objeto classificada. O

---

### Algoritmo 3: Algoritmo de Voxelização

---

**Entrada:** Malha  $M$ , *grid*  $V$ , mapa de profundidade  $P$  e rótulo  $r$

**Saída:** *Grid* com os *voxels* que pertencem ao objeto com o rótulo  $r$

```

1 início
2   para cada valor da profundidade no  $pixel(i,j)$  de  $P$  faça
3     se valor da profundidade no  $pixel(i,j) < 1.0$  então
4        $(x, y, z) = \text{ProjeçãoTelaMundo}(i, j, P(i,j))$ 
5        $(x, y, z) = \text{trunca}(x, y, z)$ 
6        $V(x, y, z) = r$ 
7     fim se
8   fim
9 fim
```

---

**Algoritmo 3** descreve o algoritmo utilizado para voxelizar a malha. O algoritmo é chamado cinco vezes, um para cada mapa de profundidade. Para cada valor da profundidade no  $pixel(i,j)$

do mapa de profundidade  $P$  é verificado se ele pertence ou não à malha (linha 3). Para descobrir isso é verificado se o valor do mapa de profundidade na posição  $(i, j)$  é menor do que 1.0, pois como o mapa foi inicializado com 1.0 em todos os seus elementos e só a malha foi renderizada na tela, qualquer valor igual a 1.0 significa que aquele *pixel* não pertence a malha. Caso o *pixel* pertença a malha, realiza-se a projeção da coordenada de tela para o mundo (linha 4). Essa projeção foi realizada com a função de *OpenGL*, `gluUnProject`. Os valores de  $x$ ,  $y$  e  $z$  encontrados na projeção são trucados (linha 5). Em seguida, o *voxel* na posição  $(x, y, z)$  do *grid* é marcado com o rótulo  $r$  (linha 6).

### 3.4.4 Geração de Malhas de *Offset*

O nosso algoritmo de geração de malhas utiliza o algoritmo de simplificação, **Algoritmo 2**, para evitar as faces degeneradas durante o deslocamento e o algoritmo de voxelização, **Algoritmo 3**, para rotular os *voxels* de forma que o algoritmo de renderização implementado no nosso protótipo VMTK, a ser apresentado no Capítulo 5, possa gerar o efeito de reformatação curvilínea em tempo interativo. Ele é descrito no **Algoritmo 4**. Enquanto a malha tiver um

---

#### Algoritmo 4: Algoritmo de Geração de Malhas

---

**Entrada:** Malha  $M$   
**Saída:** Malha  $M$  deslocada

```

1 início
2   Grid  $V = 0$ 
3   enquanto Número de faces > Número mínimo de faces e Profundidade máxima não
   alcançada faça
4     Gerar mapas de profundidade  $P$ 
5     Algoritmo de Voxelização( $M, V, P, profundidade\ atual$ )
6     se Existe face degenerada então
7       Calcular o custo das contrações das faces degeneradas e escolher o maior custo
       como  $\epsilon$ 
8       Algoritmo de Simplificação ( $M, \epsilon$ )
9     fim se
10    Deslocar a malha na direção dos seus vetores normais
11  fim enquanto
12 fim
```

---

número mínimo de faces e não tiver alcançado a profundidade máxima definida, a malha continua sendo deslocada (linhas 3-11). O *grid*  $V$  de volume de rótulos é inicializado com 0 (linha 2). Os mapas de profundidade são gerados (linha 4), como mencionados na seção anterior, e a cada deslocamento a malha é voxelizada (linha 5). O algoritmo de voxelização é chamado para cada mapa de profundidade  $P$  gerado. O rótulo atribuído ao *grid* de volume de rótulos é a profundidade atual. Caso alguma face degenerada seja encontrada, o custo para contrair esta face é calculado e atribuído a  $\epsilon$ . Caso exista mais de uma face, o maior custo de contração é atribuído, ou seja, o maior custo de contração dentre as faces degeneradas é passado como custo máximo permitido no algoritmo de simplificação naquela iteração (linha 7 e 8).



## 3.5 Testes de Validação

A plataforma utilizada para os experimentos foi um *desktop* Intel ®Core i7 2.8 GHz com 8GB de RAM e placa de vídeo NVIDIA GeForce GTX 650 Ti com 2GB de VRAM, com sistema operacional *Windows 7* e *Ubuntu 14.02*. A linguagem utilizada foi C++ e as bibliotecas usadas foram *wxWidgets* e *OpenGL*. A estrutura de dados *half-edge* foi utilizada para representar a malha.

### 3.5.1 Eficiência

O algoritmo de geração de malhas de *offset* consiste essencialmente no deslocamento da malha ao longo dos vetores normais dos seus vértices. O critério de parada é pelo número mínimo de faces ou pela profundidade. Nos experimentos realizados, o número mínimo de faces foi de 10 faces e a profundidade foi de 64,5mm. A cada passo do deslocamento é gerada uma malha de *offset* a partir dos novos vértices deslocados. A topologia da malha só é preservada se o critério de razão *área* pelo *maior lado* for satisfeito por todas as novas faces. Faces que se encontrarem abaixo do limiar  $\delta$  são consideradas degeneradas. Quanto maior o limiar, mais faces serão consideradas degeneradas. Fixamos o limiar em  $\delta = 0.2$  a fim de considerar faces de áreas pequenas que podem ocorrer nas regiões de maior curvatura. Vale ressaltar que aplicamos o algoritmo de simplificação somente na situação em que surgem as faces degeneradas, como ilustrado pela Figura 3.15. Portanto, apesar do seu alto custo, a sua aplicação ocorre a uma

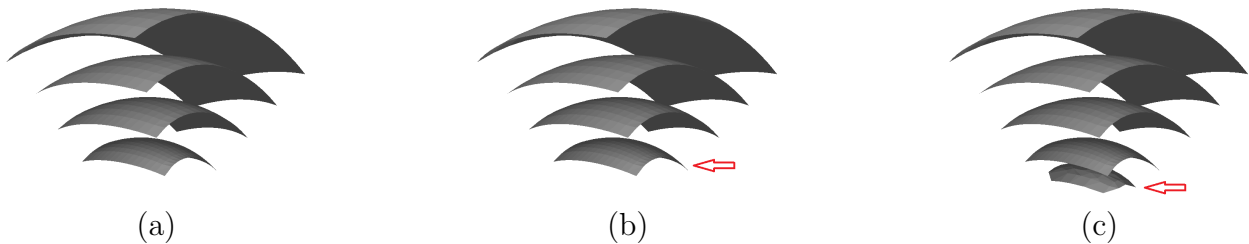


Figura 3.15: O algoritmo de simplificação é somente aplicado quando alguma face degenerada é encontrada. (a) A malha é deslocada. (b) Quando uma face degenerada é encontrada na malha, (c) o algoritmo de simplificação é o utilizado.

| Número de Faces | Número de Faces Removidas | Profundidade(mm) | Tempo(s) |
|-----------------|---------------------------|------------------|----------|
| 458             | 188                       | 26,0             | 0,242    |
| 270             | 111                       | 26,5             | 0,086    |
| 159             | 70                        | 27,0             | 0,033    |
| 89              | 40                        | 27,5             | 0,012    |
| 49              | 21                        | 44,5             | 0,004    |
| 28              | 11                        | 49,0             | 0,002    |

Tabela 3.1: Tempo gasto para simplificar as 6 malhas. Tempo total: 0,379s.

baixa taxa de frequência. Por exemplo, para o caso mostrado na Figura 3.17 foram geradas 130 malhas e o algoritmo de simplificação foi aplicado somente em 6 malhas.

A Tabela 3.1 apresenta o tempo gasto para realizar essas simplificações, quantas faces foram removidas e em qual profundidade. O tempo da simplificação diminui com a diminuição do número de faces da malha. Na Figura 3.16 são destacadas três dessas malhas. Quando a malha é criada, nenhuma face é degenerada (profundidade 0mm). Ela começa a ser deslocada e as áreas das faces começam a diminuir até algumas delas ficarem degeneradas. O algoritmo de simplificação só é aplicado quando isso ocorre. Por isso a simplificação sempre acontece depois do deslocamento ter atingido uma certa profundidade. A profundidade máxima definida foi de 64mm. Com essa profundidade se consegue um corte profundo. A Figura 3.17 mostra o

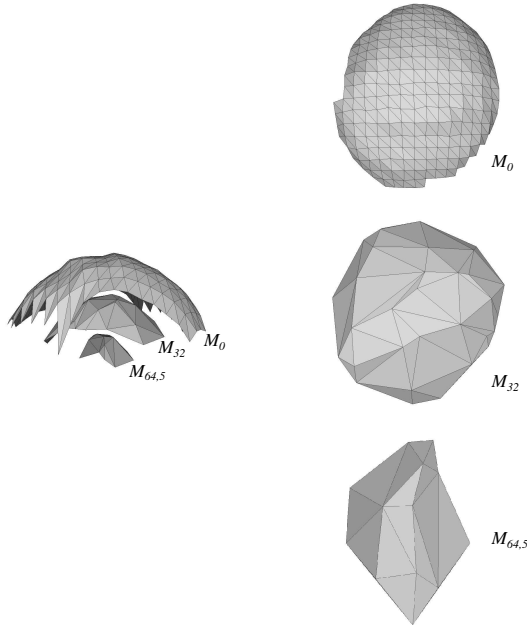


Figura 3.16: Deslocamento da malha na direção dos vetores normais dos vértices. Três malhas são apresentadas: malha com profundidade 0mm, malha com profundidade 32mm, e malha com profundidade 64,5mm.

resultado destas malhas na reformatação local e de forma curvilínea do cérebro de um paciente. O tempo gasto para realizar a reformatação foi de aproximadamente 7 segundos, incluindo a etapa de voxelização. A Tabela 3.3 apresenta detalhadamente os tempos gastos para voxelizar cada uma das 130 malhas de *offset* geradas para a cabeça do paciente da Figura 3.17. O tempo de voxelização diminui quando a profundidade aumenta, pois as projeções realizadas, como mostrado no algoritmo de voxelização (**Algoritmo 3**), diminuem.

### 3.5.2 Qualidade

A distância de Hausdorff, Seção 2.6, foi utilizada para medir o quanto a geometria de uma malha está distante da geometria da malha original depois de ser simplificada. Foi utilizada a ferramenta **Metro** para calcular a distância de Hausdorff. Esta ferramenta foi desenvolvida por (Cignoni, Rocchini & Scopigno 1996) para comparar quantitativamente a forma geométrica de duas malhas triangulares  $M_1$  e  $M_2$ . Calculamos as distâncias de Hausdorff entre as malhas antes e após cada um dos seis passos de simplificação que ocorreram ao longo da geração de malhas

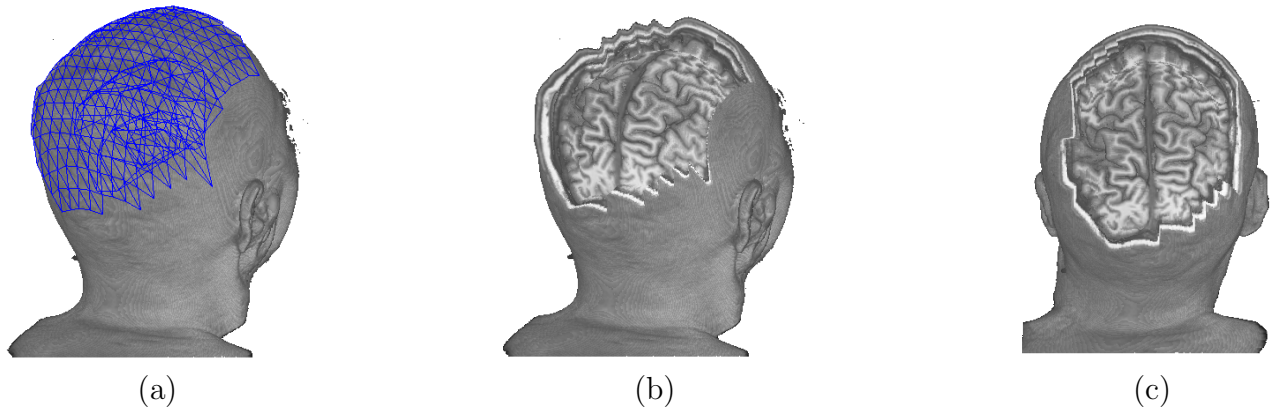


Figura 3.17: Reformatação curvilínea sem artefatos.

| Malha #1 | Malha #2 | Distância de Hausdorff |
|----------|----------|------------------------|
| (a)      | (b)      | 9,691101               |
| (c)      | (d)      | 17,948236              |
| (e)      | (f)      | 8,176524               |
| (g)      | (h)      | 25,396851              |
| (i)      | (j)      | 13,433139              |
| (k)      | (l)      | 13,145451              |

Tabela 3.2: Distância das malhas apresentadas pela Figura 3.19.

de *offset* para o paciente mostrado na Figura 3.17. As seis malhas que foram simplificadas são ilustradas na Figura 3.19. A Tabela 3.2 apresenta os valores das distâncias de Hausdorff entre as malhas: Malha #1; malha antes da simplificação e Malha #2; a malha depois da simplificação. Considerando (1) o grau de liberdade que os vértices das malhas tem para se deslocarem, e portanto, a grande variação na geometria das bordas da malha e (2) a resolução das imagens 3D, pode-se dizer que a qualidade de simplificação é aceitável. Vale comentar a aparente discrepância na segunda linha da Tabela 3.2. Isso decorre do fato de que após uma simplificação, a nova malha pode ter algumas regiões mais distantes da malha original. A Figura 3.18 ilustra a superfície de uma malha,  $M_1$ , antes da simplificação e a superfície da malha  $M_1$ , depois da simplificação,  $M_2$ . Como a distância de Hausdorff calcula a maior distância entre as menores, tais desajustes nas malhas podem causar um aumento na distância.

### 3.6 Discussões

O algoritmo de voxelização é utilizado a cada vez que uma malha é gerada. Ele consome mais de 50% do tempo total de execução do algoritmo de geração de malhas de *offset*, mesmo que tenha-se explorado os recursos de paralelismo em processamento de dados da GPU. Esse tempo alto decorre do fato dele ser aplicado em todas as malhas de *offset* criadas. Embora o custo computacional do algoritmo de simplificação seja maior, a parcela de tempo que ele consome é pequeno devido a baixa frequência em que ele é aplicado.

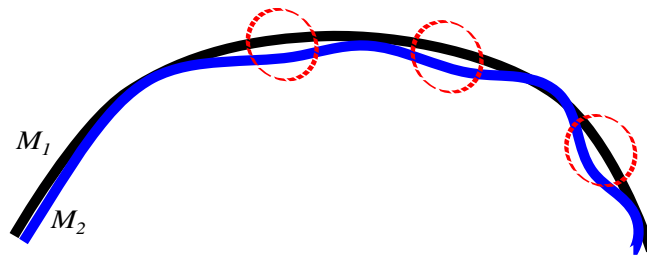


Figura 3.18: A malha  $M_1$ , representa a malha antes da simplificação, e a malha  $M_2$  representa a malha depois da simplificação. Algumas regiões se distanciaram mais do que outras regiões durante a simplificação.

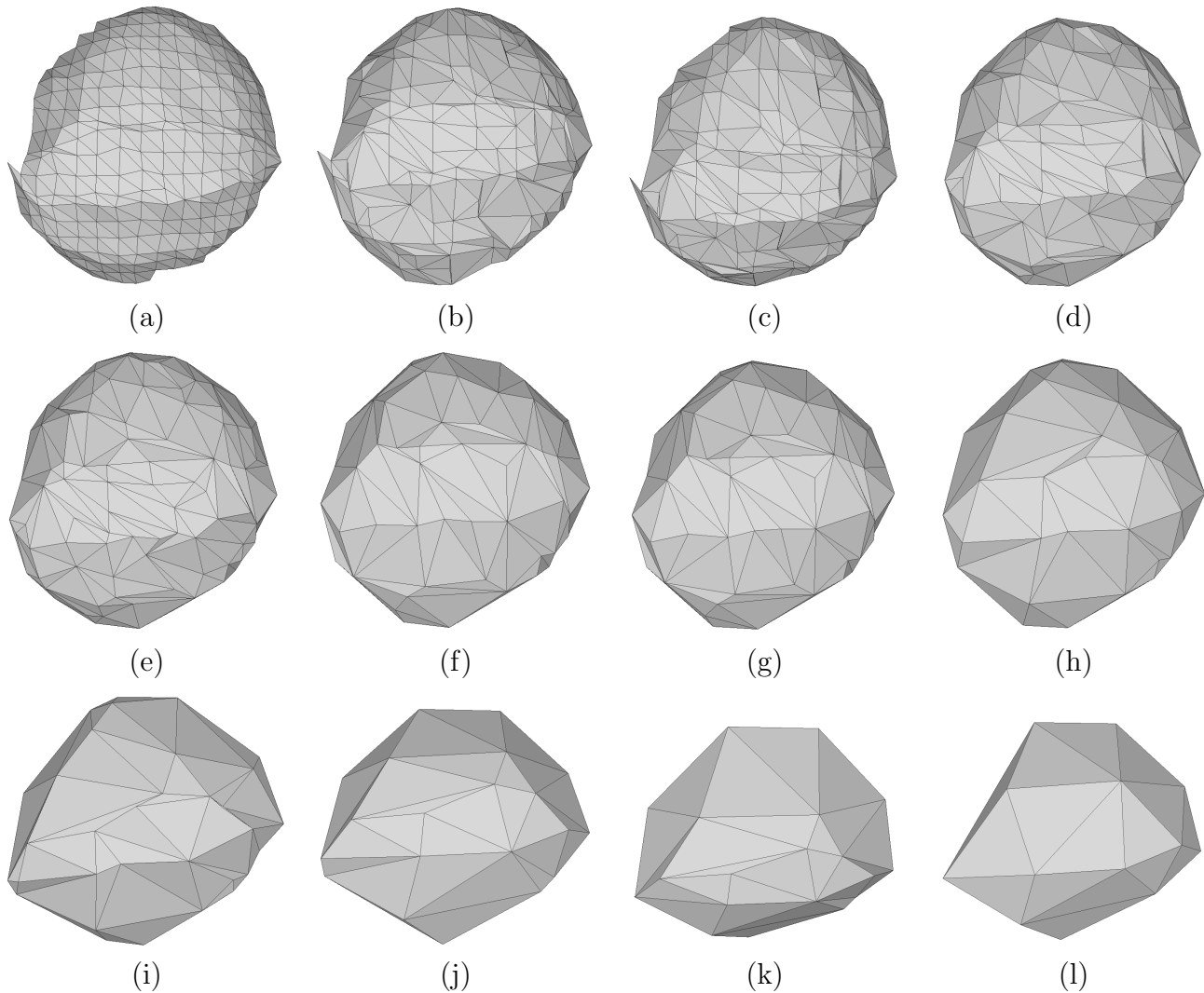


Figura 3.19: Malhas antes e depois da simplificação. (a) 458 faces. (b) 188 faces removidas e 270 faces restantes. (c) 270 faces. (d) 111 faces removidas e 159 faces restantes. (e) 159 faces. (f) 70 faces removidas e 89 faces restantes. (g) 89 faces. (h) 40 faces removidas e 49 faces restantes. (i) 49 faces. (j) 21 faces removidas e 28 faces restantes. (k) 28 faces. (l) 11 faces removidas e 17 faces restantes.

As distâncias entre as malhas e suas simplificações, apresentadas pela Tabela 3.2, mostraram uma pequena distância entre o modelo original e o modelo simplificado. Para que a suavidade

da malha não seja alterada de forma abrupta, a distância entre a malha original e a malha simplificada deve ser pequena. Quando o limiar  $\delta$  for muito grande haverá remoção de muitas faces em cada simplificação e isso pode ocasionar cortes com pouca profundidade. Para evitar essa situação, nosso algoritmo de simplificação não permite que sejam removidos mais do que 15% do total de faces da malha a cada deslocamento. Caso após as remoções a malha ainda permaneça com faces degeneradas, o algoritmo de simplificação será aplicado novamente. A desvantagem desta estratégia está no seu impacto direto no tempo total do algoritmo de geração de malhas de *offset*.

Em relação ao procedimento de contração em nosso trabalho, a nova posição do vértice  $\mathbf{v}$ , após a contração da aresta entre  $\mathbf{v}_i$  e  $\mathbf{v}_j$ , será sempre um dos extremos,  $\mathbf{v}_i$  ou  $\mathbf{v}_j$ . Contudo, pode-se tentar encontrar uma posição  $\mathbf{v}$  que minimize a função custo  $C(v_i, v_j)$  da Equação 3.3 fazendo  $\delta C/\delta x = 0$ ,  $\delta C/\delta y = 0$  e  $\delta C/\delta z = 0$ . Isso equivale a resolver (Garland & Heckbert 1997),

$$\mathbf{Q} \mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{v} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Caso a matriz  $\mathbf{Q}$  não seja inversível um dos pontos extremos ou o ponto médio é escolhido.

Finalmente, vale observar que trata-se nesse trabalho apenas as auto-interseções locais, pois as auto-interseções globais não ocorrem. Isso decorre do fato de que as superfícies curvilíneas são abertas e suaves, como ilustra a Figura 3.15.

| N. de Faces | Prof.(mm) | Tempo(s) | N. de Faces | Prof.(mm) | Tempo(s) | N. de Faces | Prof.(mm) | Tempo(s) |
|-------------|-----------|----------|-------------|-----------|----------|-------------|-----------|----------|
| 458         | 0,0       | 0,102    | 458         | 13,0      | 0,086    | 458         | 26,0      | 0,057    |
| 458         | 0,5       | 0,096    | 458         | 13,5      | 0,076    | 270         | 26,5      | 0,056    |
| 458         | 1,0       | 0,097    | 458         | 14,0      | 0,073    | 159         | 27,0      | 0,052    |
| 458         | 1,5       | 0,097    | 458         | 14,5      | 0,079    | 89          | 27,5      | 0,049    |
| 458         | 2,0       | 0,091    | 458         | 15,0      | 0,069    | 49          | 28,0      | 0,049    |
| 458         | 2,5       | 0,094    | 458         | 15,5      | 0,071    | 49          | 28,5      | 0,049    |
| 458         | 3,0       | 0,09     | 458         | 16,0      | 0,07     | 49          | 29,0      | 0,049    |
| 458         | 3,5       | 0,087    | 458         | 16,5      | 0,085    | 49          | 29,5      | 0,047    |
| 458         | 4,0       | 0,096    | 458         | 17,0      | 0,071    | 49          | 30,0      | 0,048    |
| 458         | 4,5       | 0,087    | 458         | 17,5      | 0,076    | 49          | 30,5      | 0,045    |
| 458         | 5,0       | 0,094    | 458         | 18,0      | 0,07     | 49          | 31,0      | 0,047    |
| 458         | 5,5       | 0,09     | 458         | 18,5      | 0,065    | 49          | 31,5      | 0,048    |
| 458         | 6,0       | 0,084    | 458         | 19,0      | 0,065    | 49          | 32,0      | 0,047    |
| 458         | 6,5       | 0,088    | 458         | 19,5      | 0,067    | 49          | 32,5      | 0,044    |
| 458         | 7,0       | 0,082    | 458         | 20,0      | 0,07     | 49          | 33,0      | 0,044    |
| 458         | 7,5       | 0,083    | 458         | 20,5      | 0,065    | 49          | 33,5      | 0,048    |
| 458         | 8,0       | 0,091    | 458         | 21,0      | 0,063    | 49          | 34,0      | 0,042    |
| 458         | 8,5       | 0,08     | 458         | 21,5      | 0,063    | 49          | 34,5      | 0,042    |
| 458         | 9,0       | 0,075    | 458         | 22,0      | 0,064    | 49          | 35,0      | 0,043    |
| 458         | 9,5       | 0,077    | 458         | 22,5      | 0,062    | 49          | 35,5      | 0,04     |
| 458         | 10,0      | 0,077    | 458         | 23,0      | 0,062    | 49          | 36,0      | 0,041    |
| 458         | 10,5      | 0,075    | 458         | 23,5      | 0,062    | 49          | 36,5      | 0,041    |
| 458         | 11,0      | 0,085    | 458         | 24,0      | 0,059    | 49          | 37,0      | 0,041    |
| 458         | 11,5      | 0,079    | 458         | 24,5      | 0,06     | 49          | 37,5      | 0,047    |
| 458         | 12,0      | 0,083    | 458         | 25,0      | 0,064    | 49          | 38,0      | 0,04     |
| 458         | 12,5      | 0,095    | 458         | 25,5      | 0,07     | 49          | 38,5      | 0,043    |

| N. de Faces | Prof.(mm) | Tempo(s) | N. de Faces | Prof.(mm) | Tempo(s) |
|-------------|-----------|----------|-------------|-----------|----------|
| 49          | 39,0      | 0,039    | 17          | 52,0      | 0,024    |
| 49          | 39,5      | 0,036    | 17          | 52,5      | 0,024    |
| 49          | 40,0      | 0,036    | 17          | 53,0      | 0,025    |
| 49          | 40,5      | 0,037    | 17          | 53,5      | 0,023    |
| 49          | 41,0      | 0,037    | 17          | 54,0      | 0,024    |
| 49          | 41,5      | 0,036    | 17          | 54,5      | 0,023    |
| 49          | 42,0      | 0,035    | 17          | 55,0      | 0,023    |
| 49          | 42,5      | 0,036    | 17          | 55,5      | 0,022    |
| 49          | 43,0      | 0,036    | 17          | 56,0      | 0,025    |
| 49          | 43,5      | 0,034    | 17          | 56,5      | 0,025    |
| 49          | 44,0      | 0,037    | 17          | 57,0      | 0,02     |
| 49          | 44,5      | 0,03     | 17          | 57,5      | 0,021    |
| 28          | 45,0      | 0,031    | 17          | 58,0      | 0,046    |
| 28          | 45,5      | 0,034    | 17          | 58,5      | 0,022    |
| 28          | 46,0      | 0,032    | 17          | 59,0      | 0,021    |
| 28          | 46,5      | 0,03     | 17          | 59,5      | 0,02     |
| 28          | 47,0      | 0,029    | 17          | 60,0      | 0,02     |
| 28          | 47,5      | 0,029    | 17          | 60,5      | 0,029    |
| 28          | 48,0      | 0,029    | 17          | 61,0      | 0,019    |
| 28          | 48,5      | 0,028    | 17          | 61,5      | 0,02     |
| 28          | 49,0      | 0,027    | 17          | 62,0      | 0,019    |
| 17          | 49,5      | 0,025    | 17          | 62,5      | 0,02     |
| 17          | 50,0      | 0,026    | 17          | 63,0      | 0,023    |
| 17          | 50,5      | 0,025    | 17          | 63,5      | 0,021    |
| 17          | 51        | 0,026    | 17          | 64,0      | 0,02     |
| 17          | 51,5      | 0,026    | 17          | 64,5      | 0,019    |

Tabela 3.3: Tempos gastos para voxelizar as malhas de *offset* da Figura 3.17. Tempo total da voxelização: 6,685s. Tempo total do algoritmo de geração de malhas de *offset*: 7,064s.

## Costura entre Malhas

Devido à geometria da cabeça não é possível ter uma visão simultânea dos dois hemisférios. A visualização dos dois hemisférios ajuda no diagnóstico comparativo. Como o procedimento utilizado para selecionar a região de interesse é baseado na sua visibilidade, uma reformatação curvilínea cobrindo a extensão de dois hemisférios requer a seleção de mais de uma área. A necessidade de realizar mais de uma seleção é uma limitação ao nosso algoritmo de reformatação curvilínea apresentado na Seção 2.9, que é baseado na visibilidade. Para contornar esta restrição, propõe-se um algoritmo de costura de malhas adquiridas em diferentes pontos de vista antes da efetivação de uma reformatação curvilínea. A demarcação da primeira área é realizada, em seguida a cabeça é mudada de posição e a segunda demarcação é realizada. Para cada posição da cabeça do paciente, uma malha é construída. Em seguida elas são costuradas formando assim uma só malha para assegurar a suavidade na região de sobreposição das duas malhas durante a geração de malhas de *offset*. A Figura 4.1 ilustra o procedimento. A costura sempre é realizada par a par. Caso as malhas não fossem costuradas elas iriam se sobrepor durante o deslocamento e artefatos indesejáveis surgiriam. Considera-se que as superfícies das malhas da cabeça sejam convexas.

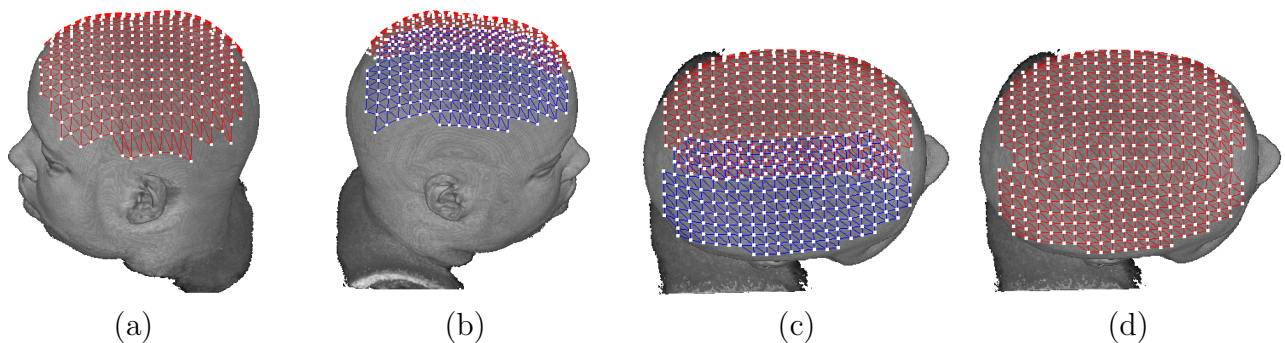


Figura 4.1: Costura de malhas. (a) A primeira região é demarcada, (b) em seguida demarcamos outra região. (c) As duas malhas representam a superfície demarcada. (d) As duas malhas são costuradas, formando uma só malha.

## 4.1 Trabalhos Relacionados

A maioria dos trabalhos relacionados encontra a interseção entre as malhas e depois utilizam os pontos de interseção para ligá-las. Turk e Levoy (Turk & Levoy 1994) apresentaram um método que combina malhas a partir de imagens geradas por um sensor. O objetivo era reconstruir um modelo físico. O método é dividido em três passos. O primeiro passo é encontrar uma transformação rígida que minimize a distância entre as duas malhas. O próximo passo é realizar a remoção da parte sobreposta de uma das malhas sobre a outra. Somente faces que possuem todos os vértices sobre a outra malha são removidas. Para cada ponto de interseção um novo vértice é criado. Em seguida novos triângulos são criados para juntar as duas malhas. O último passo consiste na reintrodução de informações sobre detalhes da superfície que foram perdidas no momento da remoção da parte sobreposta. Em nosso trabalho as malhas são alinhadas, assim não há a necessidade de realizar a etapa de encontrar uma transformação rígida.

Shostko *et al.* (Shostko, Hner & Sandberg 1999) combina superfícies de geometrias primitivas (esferas, cubos, cilindros) por meio de operações booleanas para realizar construções interativas na área de CAD/CAM<sup>1</sup>. Para combinar as duas superfícies a curva de interseção entre elas é encontrada. Em seguida a área onde a interseção ocorre é retirada e ambas as superfícies são ligadas. Figueiredo *et al.* (Figueiredo 1999) propuseram um algoritmo que combina pequenas superfícies. O objetivo é projetar estruturas que transportam petróleo no mar. Dadas duas superfícies seus pontos de interseção são encontrados e armazenados. Em seguida esses pontos de interseção são ligados. As faces sobre essa região são removidas. Novos triângulos são formados utilizando os pontos da curva de interseção das superfícies. Os algoritmos apresentados realizam a costura da malha de forma robusta. Contudo, optou-se em propor uma outra maneira de resolver o problema de costurar malhas, pois o nosso problema envolve geometrias bem específicas, quase coplanares na região de interseção.

## 4.2 Hipótese

O maior custo computacional dos algoritmos mencionados na Seção(4.1) está na fase de remoção da área de sobreposição, pois ela requer que sejam determinados os pontos de interseção entre as duas malhas. A hipótese do presente trabalho é que explorando devidamente os recursos disponíveis nas GPU's e as características específicas do nosso problema, pode-se realizar a costura das malhas em tempo interativo. Nossa proposta explora o recurso de mapa de profundidade oferecido pelas modernas placas gráficas para encontrar as faces sobrepostas e removê-las.

## 4.3 Proposta

Nossa proposta consiste em dois passos: remoção da área sobreposta e ligação das malhas. O passo de remoção da área sobreposta foi dividida em:

---

<sup>1</sup>CAD (*Computer-Aided Design*) e CAM (*Computer-Aided Manufacturing*)



1. adquirir o mapa de profundidade de ambas as malhas;
2. encontrar vértices de transição;
3. extrair as frentes das malhas e
4. remover a parte sobreposta.

Enquanto o passo da ligação das malhas foi dividida em:

1. contrair os vértices mais próximos e
2. preencher os buracos que se formem durante o passo anterior.

### 4.3.1 Remoção da Área Sobreposta

Quando se costura duas malhas com sobreposição parcial é necessário remover as faces de uma das malhas para evitar duplicidade na malha final. Decidir quais faces devem ser removidas pode ser uma tarefa não trivial quando se trata de malhas com geometria de interseção muito complexa. Para o presente caso, em que as malhas desenhadas sobre o escalpo de um paciente tem as partes sobrepostas quase coplanares, pode-se escolher aleatoriamente uma das malhas e remover as suas faces que sobrepõem a outra malha. Isso reduz o nosso problema de remoção da área de sobreposição de suas malhas num problema de determinação da área de sobreposição destas duas malhas quase coplanares. Partindo desta observação chega-se a um eficiente algoritmo que evita o cômputo explícito de interseções.

#### Adquirir o mapa de profundidade de ambas as malhas

Para costurar as duas malhas o primeiro passo do algoritmo é adquirir o mapa de profundidade de ambas as malhas para realizar a identificação de quais faces estão na área de sobreposição. Toda face que sobrepõe uma outra face é considerada uma face sobreposta. A área de sobreposição é formada por todas as faces sobrepostas. Como as malhas são quase coplanares, os seus mapas de profundidade na região de sobreposição, colocada em paralelo à tela de projeção, devem se sobrepor. A Figura 4.2 e a Figura 4.3 ilustram a disposição entre duas malhas e seus respectivos mapas de profundidade.

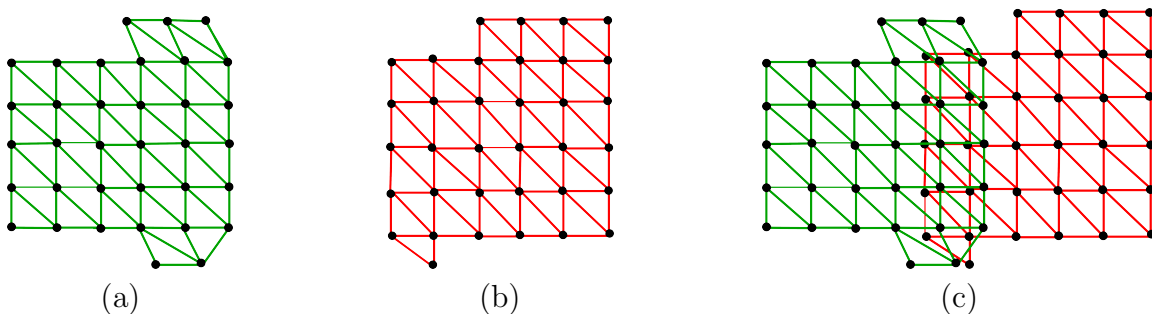


Figura 4.2: Malhas: (a)  $M_1$ , (b)  $M_2$  e (c) malhas sobrepostas.

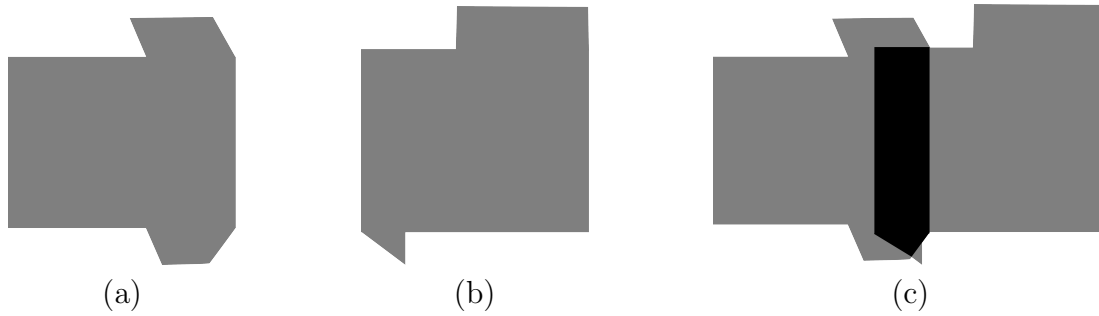


Figura 4.3: Mapas de profundidade: (a) da malha  $M_1$ , (b) da malha  $M_2$  e (c) sobrepostos.

Dadas duas malhas,  $M_1$  e  $M_2$ , os mapas de profundidade de  $M_1$  (Figura 4.3(a)) e da malha  $M_2$  (Figura 4.3(b)) são sobrepostos e a região em comum aos dois mapas demarca a área de sobreposição indicada pela região em preto na Figura 4.3(c). Disposto dos mapas de profundidade pode-se determinar as faces sobrepostas. Para isso é verificado para cada face se alguma aresta está dentro da região de sobreposição dos mapas de profundidade. Cada aresta do triângulo é amostrada e projetada do espaço de coordenadas do mundo para o espaço de coordenadas de tela, usando a função `gluProject`<sup>2</sup>. O **Algoritmo 5** sintetiza esse procedimento. Para cada

---

**Algoritmo 5:** Algoritmo de Classificação de Faces Sobrepostas

---

**Entrada:** Malha  $M_1$  e mapa de profundidade  $M_{p2}$  da malha  $M_2$

**Saída:** Conjunto de faces sobrepostas da malha  $M_1$

```

1 início
2   para cada Triângulo  $t$  de  $M_1$  faça
3     para cada Aresta  $e$  de  $t$  faça
4       se Aresta  $e$  estiver no mapa de profundidade  $M_{p2}$  então
5         classifica o triângulo  $t$  como sobreposto
6       fim se
7     fim
8   fim
9 fim
```

---

triângulo  $t$  da malha  $M_1$  é verificado se alguma aresta de  $t$  está no mapa de profundidade da malha  $M_2$  (linhas 2-4). Para realizar essa verificação foi utilizada a equação paramétrica da reta. A aresta a um passo de discretização  $\lambda = 0,01$ ,  $\lambda \in [0,1]$ , é amostrada e para cada amostra é verificado se ela se encontra no mapa de profundidade de  $M_2$ . Caso esteja, a face é marcada como sobreposta (linha 5). Ao final desse processo temos as faces da malha  $M_1$  que sobrepõem as faces da malha  $M_2$ . De maneira análoga as faces da malha  $M_2$  que sobrepõem as faces da malha  $M_1$  são encontradas.

---

<sup>2</sup>Uma função de *OpenGL* que realiza a projeção do espaço de coordenadas do mundo para o espaço de coordenadas de tela.

### Encontrar vértices de transição

Após a marcação das faces sobrepostas o segundo passo é encontrar os vértices de transição. Os vértices de transição são classificados em dois tipos: externo e interno. Eles são usados para encontrar quais vértices serão utilizados para costurar as duas malhas. Um vértice na borda tem dois vizinhos adjacentes que estão na borda,  $v_{ant}$  e  $v_{prox}$ , como ilustra a Figura 4.4. Para

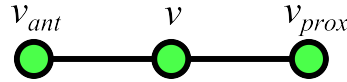


Figura 4.4: Vizinhos do vértice  $v$  que estão na borda,  $v_{ant}$  e  $v_{prox}$ .

identificar os vértices de transição externos a borda da malha é percorrida no sentido horário. Se o vértice  $v$  estiver fora da área de sobreposição mas algum vértice adjacente a ele e que pertença a borda,  $v_{ant}$  ou  $v_{prox}$ , estiver dentro da área de sobreposição este vértice é classificado como vértice de transição externo. Existem dois vértices de transição externos: um vértice que tem o seu próximo vizinho na região de sobreposição, e um vértice que tem o seu vizinho anterior na área de sobreposição. A Figura 4.5 mostra os dois vértices de transição externos encontrado,  $v_{ex1}$  e  $v_{ex2}$ . O vértice  $v_{ex1}$  tem seu próximo vizinho dentro da área de sobreposição, e o vértice  $v_{ex2}$  tem seu vizinho anterior dentro da área de sobreposição.

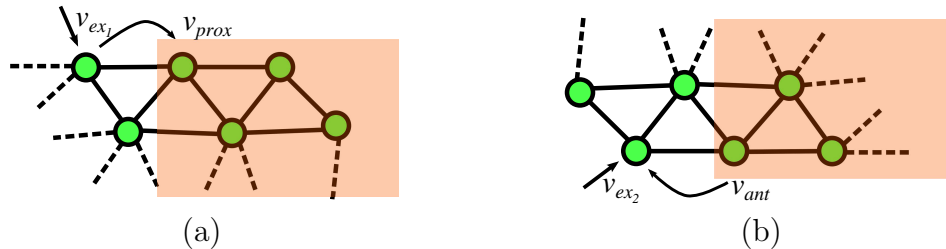


Figura 4.5: Vértices de transição externos: (a)  $v_{ex1}$  e (b)  $v_{ex2}$ .

O **Algoritmo 6** descreve os passos para encontrar os vértices de transição externos. Para cada vértice  $v$  da borda da malha  $M_1$  (linha 2) é verificado se ele não está na área de sobreposição e se ele possui algum vizinho,  $v_{ant}$  ou  $v_{prox}$ , que esteja dentro da região de sobreposição (linhas 3 e 7). Caso isso aconteça o vértice  $v$  é marcado como vértice de transição externo (linhas 4 e 8). A saída do algoritmo consiste de dois vértices de transições externos,  $v_{ex1}$  e  $v_{ex2}$ , um quando  $v_{prox}$  está dentro da região de sobreposição e outro quando  $v_{ant}$  está dentro da região de sobreposição (linha 5 e 9). A Figura 4.6(a) mostra os vértices de transição externos da malha  $M_1$ .

Os vértice de transição internos são vértices que pertencem a faces sobrepostas, porém possui algum vizinho na borda que não esteja em uma face sobreposta. Como nos vértices de transição externos, existem dois vértices de transição internos: um vértice quando o vizinho  $v_{prox}$  está na borda e não está numa face sobreposta e outro quando  $v_{ant}$  está na borda e não está numa face sobreposta. Esses vértices são ilustrados, respectivamente, nas Figuras 4.7(a) e (b). O **Algoritmo 7** descreve os passos para encontrar os vértices de transição internos. Para cada vértice  $v$  da face sobreposta da malha  $M_2$  (linha 2) que estiver na borda e algum vizinho,  $v_{prox}$  ou

---

**Algoritmo 6:** Algoritmo Vértices de Transição Externos

---

**Entrada:** Malha  $M_1$  e mapa de profundidade  $M_{p_2}$  da malha  $M_2$ .

**Saída:** Vértices de transição externo  $v_{ex_1}$  e  $v_{ex_2}$  da malha  $M_1$ .

```

1 início
2   para cada vértice  $v$  da borda de  $M_1$  faça
3     se  $v$  não está em  $M_{p_2}$  e  $v_{prox}$  está em  $M_{p_2}$  então
4        $v$  é um vértice de transição externo
5        $v_{ex_1} = v$ 
6     fim se
7     se  $v$  não está em  $M_{p_2}$  e  $v_{ant}$  está em  $M_{p_2}$  então
8        $v$  é um vértice de transição externo
9        $v_{ex_2} = v$ 
10    fim se
11  fim
12 fim
    
```

---

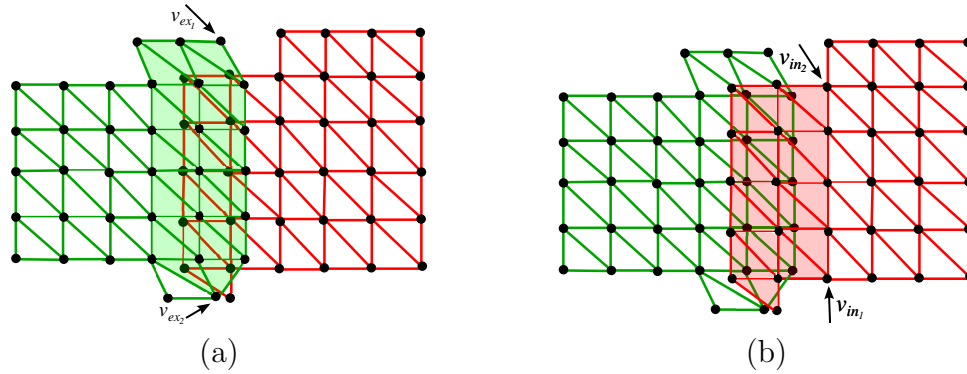


Figura 4.6: Vértices de transição. (a) Vértices de transição externo da malha  $M_1$ ,  $v_{ex_1}$  e  $v_{ex_2}$ . (b) Vértices de transição interno da malha  $M_2$ ,  $v_{in_1}$  e  $v_{in_2}$ .

$v_{ant}$ , não estiver numa face sobreposta, então  $v$  é classificado como vértice de transição interno (linha 3-12). A saída do algoritmo consiste de dois vértices de transição internos:  $v_{in_1}$  e  $v_{in_2}$  (linha 6 e 10). A Figura 4.6(b) mostra os vértices de transição interno da malha  $M_2$ .

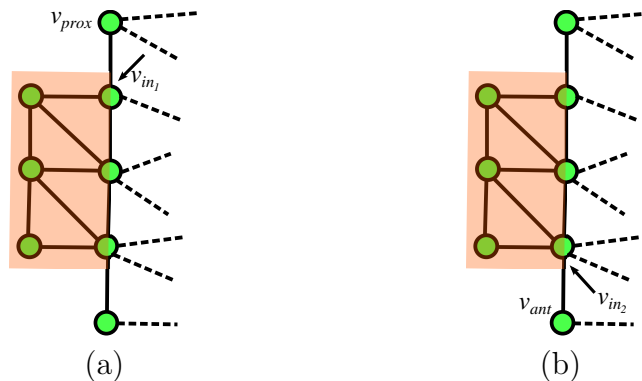


Figura 4.7: Vértices de transição internos: (a)  $v_{in_1}$  e (b)  $v_{in_2}$ .

**Algoritmo 7:** Algoritmo Vértices de Transição Internos**Entrada:** Conjunto de faces sobrepostas da malha  $M_2$ **Saída:** Vértices de transição interno  $v_{in_1}$  e  $v_{in_2}$  da malha  $M_2$ .

```

1 início
2   para cada face sobreposta  $f$  de  $M_2$  faça
3     para cada vértice  $v$  da face  $f$  que está na borda faça
4       se  $v_{prox}$  pertence a uma face não sobreposta então
5          $v$  é um vértice de transição interno
6          $v_{in_1} = v$ 
7       fim se
8       se  $v_{ant}$  pertence a uma face não sobreposta então
9          $v$  é um vértice de transição interno
10         $v_{in_2} = v$ 
11      fim se
12    fim
13  fim
14 fim

```

**Remover parte sobreposta**

Após os vértices de transição externo e interno serem obtidos as faces da malha  $M_1$  ou as da malha  $M_2$  que estão na região de sobreposição devem ser removidas. Sem perda de generalidade, foram removidas as faces de  $M_2$ . O **Algoritmo 8** descreve esse procedimento. O conjunto de faces marcadas como sobrepostas da malha  $M_2$  (linha 2) são removidas (linha 3).

**Algoritmo 8:** Algoritmo de Remoção de Faces**Entrada:** Conjunto de faces sobrepostas  $F$  e Malha  $M_2$ **Saída:** Malha  $M_2$ , com as faces sobrepostas removidas

```

1 início
2   para cada face  $f$  de  $F$  faça
3     remova  $f$  de  $M_2$ 
4   fim
5 fim

```

**Extrair as frentes das malhas**

Os vértices de transição externos e internos são utilizados para definir quais vértices devem ser utilizados para ligar as malhas. Duas frentes são definidas: frente externa e interna. A frente externa é formada pela sequência de vértices de  $M_1$  cujos extremos são  $v_{ex_1}$  e  $v_{ex_2}$ . A frente interna é formada pela sequência de vértices de  $M_2$  cujos extremos são  $v_{in_1}$  e  $v_{in_2}$ . A Figura 4.8 mostra a frente externa da malha  $M_1$  e a frente interna da malha  $M_2$  já com suas faces sobrepostas removidas.

O **Algoritmo 9** descreve os passos para extrair a frente de uma malha. Para extrair a frente externa são passados como parâmetro os vértices de transição externos,  $v_{ex_1}$  e  $v_{ex_2}$ . Os vértices

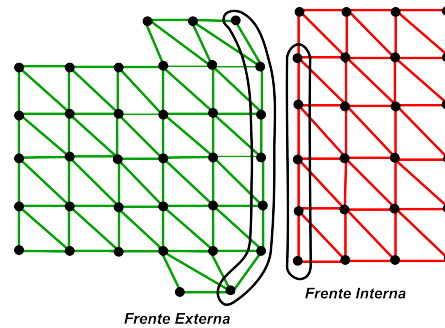


Figura 4.8: Faces da malha  $M_2$  removidas e frentes de ambas as malhas extraídas.

---

**Algoritmo 9:** Algoritmo de Extração de Frente

---

**Entrada:** Vértices de transição  $v_{t_1}$  e  $v_{t_2}$

**Saída:** Frente  $F$

```

1 início
2    $v = v_{t_1}$ 
3    $Frente = \emptyset$ 
4   enquanto  $v \neq v_{t_2}$  faça
5     |   adiciona  $v$  em  $F$ 
6     |    $v = v_{prox}$ 
7   fim enquanto
8   adiciona  $v_{t_2}$  em  $F$ 
9 fim
```

---

que estão entre  $v_{ex_1}$  e  $v_{ex_2}$  são adicionados em *Frente* (linhas 4-8). Para extrair a frente interna são passados os vértices de transição internos  $v_{in_1}$  e  $v_{in_2}$ . Da mesma maneira os vértices que estão entre  $v_{in_1}$  e  $v_{in_2}$  são adicionados em *Frente* (linhas 4-8). Depois do término as frentes externa e interna estão formadas.

### 4.3.2 Ligação da Malha

Após a remoção da região de sobreposição o próximo passo é ligar as duas malhas. Como ligar  $M_1$  e  $M_2$  com as sobreposições resolvidas? Ao se ligar duas malhas pode-se ter problemas de cruzamento de arestas ou perda de suavidade, comprometendo a qualidade da reformatação curvilínea.

#### Contrair os vértices mais próximos

Novamente, explorando o fato das duas malhas terem a sua região de sobreposição quase coplanares localmente usou-se as frentes extraídas que distam no máximo de um *voxel*, para costurá-las. Assim assegura-se que a aproximação linear seja preservada na resolução de um *voxel*. Com isso a suavidade da região onde ocorre a costura é mantida.

Como ligar as duas malhas, sem cruzar aresta? Dadas as suas frentes, interna e externa, não garante que as arestas não se cruzem se a ligação da malha for realizada da seguinte maneira: um vértice da frente interna (Figura 4.9(a)) liga com um vértice da frente externa, (Figura 4.9(b))

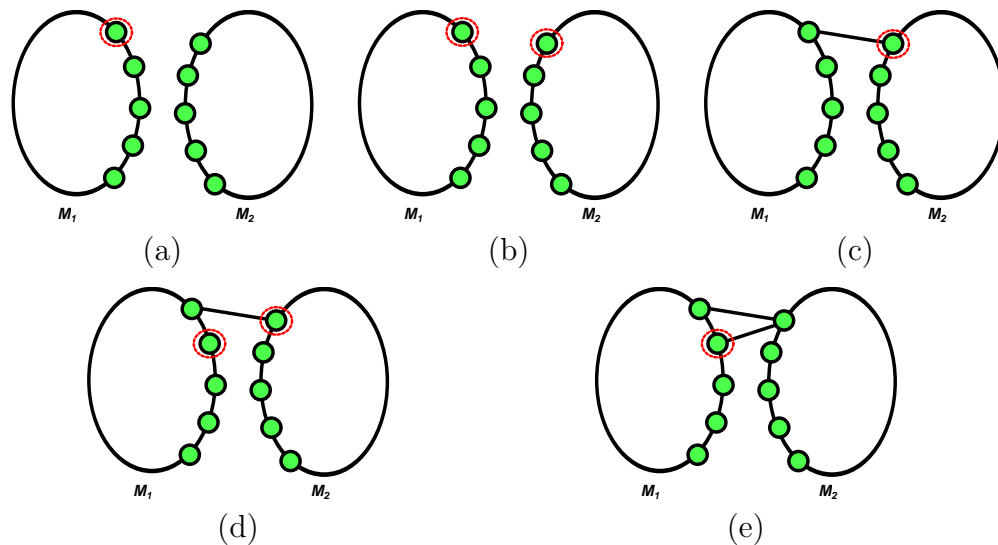


Figura 4.9: Processo de ligação intercalado. (a) Um vértice da frente da malha  $M_1$  é escolhido para ligar com (b) um vértice da malha  $M_2$ . (c) Após serem ligados, (d) o próximo vértice da malha  $M_1$  é selecionado para ser (e) ligado com o vértice da malha  $M_2$ . Esse procedimento continua até que todos os vértices tenham sido ligados.

e assim por diante, como ilustra a Figura 4.9. Um exemplo é apresentado na Figura 4.10. Propõe-se então uma outra alternativa. O primeiro passo é contrair os vértices da frente interna

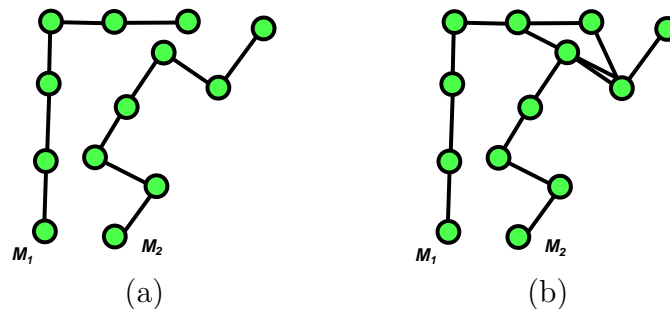


Figura 4.10: Duas frentes das malhas,  $M_1$  e  $M_2$ , sendo costuradas como mencionado na Figura 4.9. O processo é interrompido quando ocorre o cruzamento das arestas. (a) Frentes de ambas as malhas  $M_1$  e  $M_2$ . (b) Cruzamento das arestas durante o processo de costura.

com a externa. Um limiar  $\delta$  foi definido para decidir quando dois vértices podem se contrair. Esse limiar representa a distância mínima permitida para contrai-los. Para cada vértice  $v$  que compõe a frente interna, frente da malha  $M_2$  (Figura 4.11(a)), é encontrado um vértice  $w$  que pertence à frente externa, frente da malha  $M_1$ , tal que a distância entre eles seja mínima (Figura 4.11(b)). Caso essa distância seja menor que o limiar  $\delta$ , então o vértice  $v$  é contraído, (Figura 4.11(c)) caso contrário não. Quando dois vértices são contraídos podemos ter os seguintes casos:

- Dois vértices se contraem formando um buraco (Figura 4.12(a) e (b)). Isso ocorre porque o vértice anterior ficou acima do limiar  $\delta$  e algum outro vértice anteriormente já foi contraído.
- Dois vértices se contraem, contudo o vértice anterior também foi contraído, assim nenhum

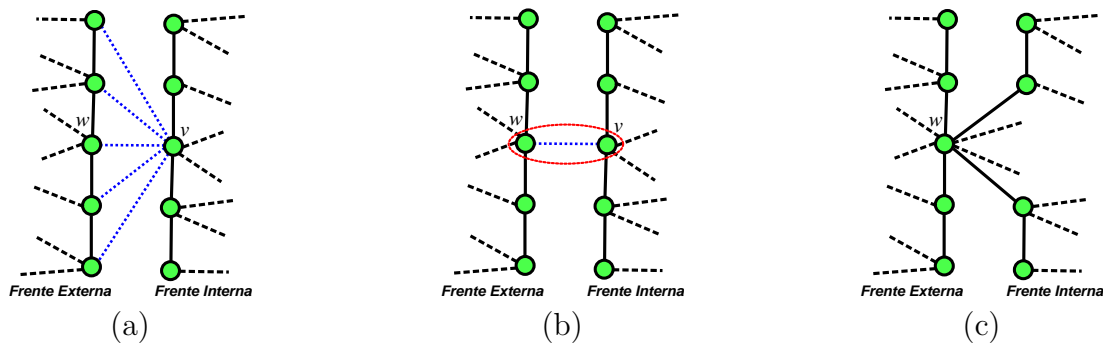


Figura 4.11: Contraíndo o vértice  $v$ . (a) A distância do vértice  $v$  até todos os outros vértices da frente externa é calculada. (b) A menor distância é encontrada. (c) O vértice  $v$  é então contraído.

buraco é formado, pois os dois vértices compartilham a mesma aresta (Figura 4.12(c) e (d)).

- Dois vértices que compartilham a mesma aresta, ou seja, que estão no mesmo triângulo são contraídos para um mesmo vértice. O resultado disso é a eliminação do triângulo, a qual esses vértices faziam parte (Figura 4.12(e) e (f)).

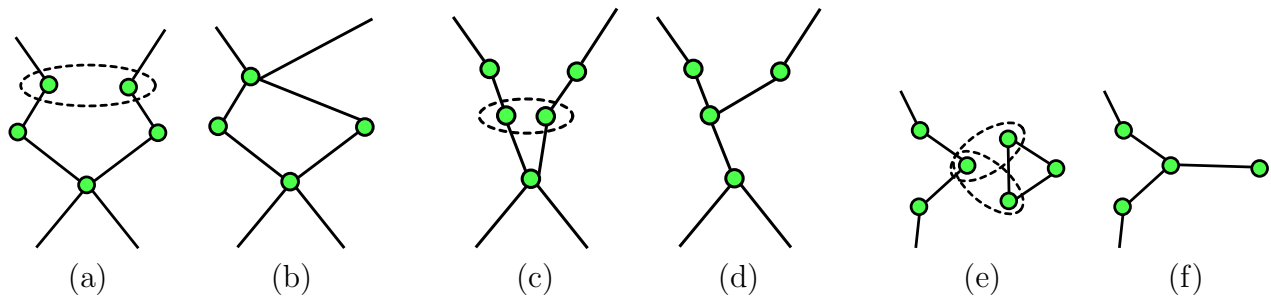


Figura 4.12: Contração de dois vértices.

O **Algoritmo 10** descreve os passos para ligar as frentes das malhas. Para cada vértice da frente interna é calculado qual é o vértice da frente externa que tem a menor distância até ele (linhas 4-7) . Caso a distância encontrada seja menor que  $\delta$  os vértices são contraídos (linhas 8 e 9). Se o vértice anterior não tiver sido contraído a contração do vértice atual formará um buraco (linhas 10 e 11).

### Preencher os buracos formados

Para preencher os buracos formados durante o passo de contração dos vértices foi utilizado um algoritmo que se baseia no ângulo entre as arestas para decidir como formar triângulos que preencham os buracos formados. Porém, como esses vértices estão no espaço tridimensional, mas sabe-se que eles são quase coplanares, decidiu-se por projetá-los primeiro em um plano que melhor aproxime os vértices do contorno do buraco para reduzir o problema de preenchimento de um buraco 3D para um buraco 2D. Para isso foi utilizado a equação de Martin Newell (Foley



**Algoritmo 10:** Algoritmo de Ligação de Vértices

---

**Entrada:** Conjunto dos vértices da frente externa  $V_{ex} = \{V_{ex_1}, V_{ex_2} \dots, V_{ex_n}\}$  e da frente interna  $V_{in} = \{V_{in_1}, V_{in_2}, \dots, V_{in_n}\}$  e  $\delta$

**Saída:** Malha  $M$ , conjunto  $B$  de buracos formados

```

1 início
2   para cada vértice  $v$  de  $V_{in}$  faça
3      $d = \infty$ 
4     para cada vértice  $w$  de  $V_{ex}$  faça
5       calcule a distância de  $v$  até  $w$ ,  $\text{dist}(v, w)$ 
6        $d = \min(d, \text{dist}(v, w))$ 
7     fim
8     se  $d < \delta$  então
9       contrair vértice  $v$  e  $w$ 
10      se vértice  $v$  não é o primeiro vértice da frente e o vértice  $v_{ant}$  não foi
11        contraído então
12           $B = v_{ant}$ 
13        fim se
14      fim se
15 fim
```

---

et al. 1990), que estima a normal de um plano  $(n_x, n_y, n_z)$  baseado nos vértices dados pelo contorno do buraco:

$$\begin{aligned}
n_x &= \sum_{i=0}^{N-1} (y_i - y_{prox_i})(z_i + z_{prox_i}) \\
n_y &= \sum_{i=0}^{N-1} (z_i - z_{prox_i})(x_i + x_{prox_i}) \\
n_z &= \sum_{i=0}^{N-1} (x_i - x_{prox_i})(y_i + y_{prox_i}),
\end{aligned}$$

onde  $(x_i, y_i, z_i)$  são as coordenadas dos vértices do contorno do buraco e  $(x_{prox_i}, y_{prox_i}, z_{prox_i})$  são as coordenadas dos seus vértices vizinhos subsequentes quando se percorre o buraco no sentido anti-horário (Figura 4.13) e  $N$  é o número de vértices do buraco. Assim, depois de ter estimado o plano, os vértices do buraco são projetados nele. A ideia é que a topologia da malha que preenche este buraco seja a mesma da malha que preencheria o buraco 3D.

Para preencher o buraco projetado é calculado para cada vértice  $v_k$  do buraco o ângulo interno formado pelos seus vértices adjacentes,  $v_{k-1}$  e  $v_{k+1}$ . Em seguida o menor ângulo é escolhido e uma aresta é criada com extremidades em  $v_{k-1}$  e  $v_{k+1}$ , criando assim um novo triângulo. Esse procedimento é repetido até que o buraco projetado esteja preenchido. Retornando as coordenadas dos pontos projetados para 3D tem-se então uma malha 3D de preenchimento do buraco original. O **Algoritmo 11** descreve o procedimento recursivo. Para cada vértice do contorno do buraco é calculado o ângulo entre suas arestas  $(v_{k-1}, v_k)$  e  $(v_k, v_{k+1})$  (linha 6).

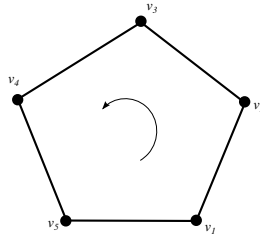


Figura 4.13: Vértices de um contorno do buraco. O vizinho a esquerda de  $v_1$ , é o vértice  $v_5$ , e o vizinho a direita, é o vértice  $v_2$

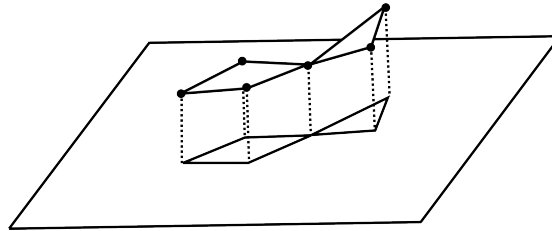


Figura 4.14: Pontos do contorno do buraco sendo projetados no plano.

---

**Algoritmo 11:** Algoritmo de Preenchimento de Buraco

---

**Entrada:** Vértices do contorno do buraco  $V = \{V_1, V_2, \dots, V_n\}$

**Saída:** Buraco preenchido

```

1 início
2   se conjunto  $V$  é vazio então
3     | retornar
4   fim se
5   para cada vértice  $v_k$  de  $V$  faça
6     | calcule o ângulo interno  $\alpha$  das arestas formada pelos seus vizinhos  $(v_{k-1}, v_k)$  e  $(v_k,$ 
7     |  $v_{k+1})$ 
8     fim
9     escolha o vértice  $v_k$  com o menor ângulo interno e forme um triângulo com vértices
10     $(v_{k-1}, v_k, v_{k+1})$ 
11    remova de  $V$  o vértice  $v_k$ 
12  Algoritmo de Preenchimento de Buraco ( $V$ )
13 fim
```

---

O vértice  $v_k$  com menor ângulo interno é escolhido e um novo triângulo é formado (linha 8). O vértice  $v_k$  é removido do contorno do buraco (linha 9) e o algoritmo é chamado novamente (linha 10). A Figura 4.15 exemplifica esse procedimento. Primeiramente os ângulos são calculados, (Figura 4.15(a)) depois o menor ângulo é escolhido (Figura 4.15(b)) e o novo triângulo é formado. Esse processo termina quando o buraco estiver completamente preenchido (Figura 4.15(f)). Ao final desse processo as malhas  $M_1$  e  $M_2$  estarão costuradas, formando assim uma só malha como mostra a Figura 4.16.

O **Algoritmo 12** resume todo o procedimento para costurar duas malhas. O mapa de profundidade é adquirido utilizando a função de *OpenGL*, `glReadPixels`, de ambas as malhas

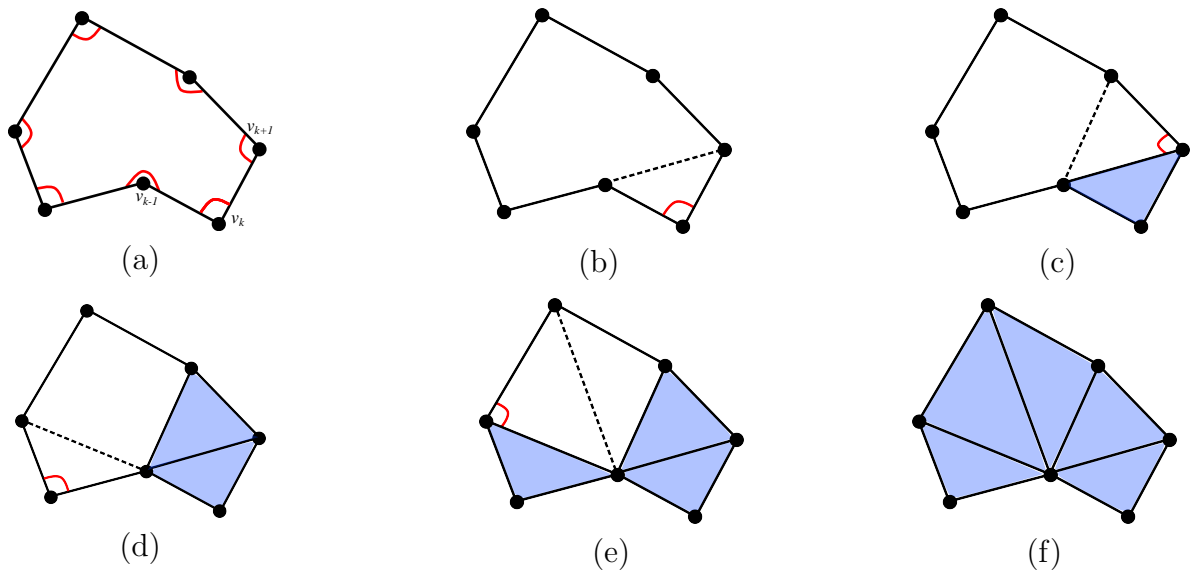


Figura 4.15: Processo de preenchimento do buraco formado durante a ligação da malha.

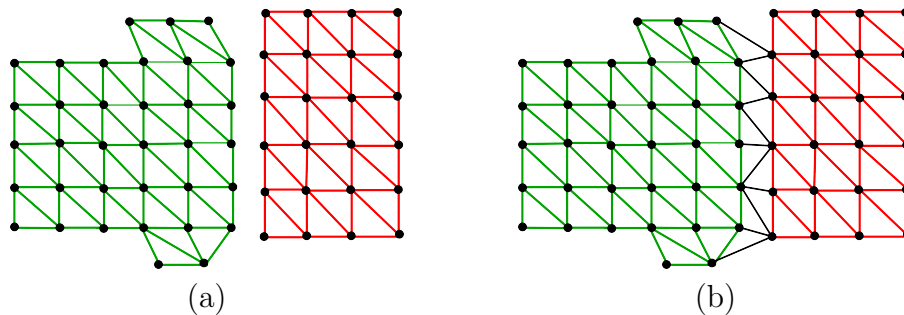


Figura 4.16: Nova malha formada a partir da costura das malhas  $M_1$  e  $M_2$ .

(linha 2). As faces sobrepostas da malha  $M_2$  são encontradas (linha 3). Em seguida os vértices de transição externos e internos são também encontrados (linhas 4 e 5). As faces sobrepostas da malha  $M_2$  são retiradas (linha 6). Os vértices de transição são utilizados para extrair a frente externa e interna (linhas 7 e 8). Então, as frentes interna e externa são utilizadas para ligar as duas malhas (linha 9). Depois da ligação é verificado se algum buraco foi formado (linha 10). Caso algum buraco tenha sido formado, eles são preenchidos (linha 12), do contrário, a costura está pronta.

## 4.4 Testes de Validação

A plataforma utilizada para os experimentos foi um *desktop* Intel <sup>®</sup>Core i7 2.8 GHz com 8GB de RAM e placa de vídeo NVIDIA GeForce GTX 650 Ti com 2GB de VRAM, com sistema operacional *Windows 7* e *Ubuntu 14.04*. A linguagem utilizada foi C++ e as bibliotecas usadas foram *wxWidgets* e *OpenGL*. A estrutura de dados utilizada para representar a malha foi *half-edge*.

**Algoritmo 12:** Algoritmo de Costura de Malhas

---

**Entrada:** Malhas  $M_1$  e  $M_2$   
**Saída:** Uma única malha  $M_3$

- 1 **início**
- 2    $(M_{p_1}, M_{p_2}) =$  mapas de profundidade adquiridos da malha  $M_1$  e  $M_2$
- 3    $F_2 =$  Algoritmo de Classificação de Faces Sobrepostas ( $M_{p_1}, M_2$ )
- 4    $(v_{ex_1}, v_{ex_2}) =$  Algoritmo Vértices de Transição Externos ( $M_{p_2}, M_1$ )
- 5    $(v_{in_1}, v_{in_2}) =$  Algoritmo Vértices de Transição Internos ( $F_2, M_2$ )
- 6   Algoritmo Remoção Faces Sobrepostas ( $F_2$ )
- 7    $Frente_{ex} =$  Algoritmo de Extração de Frente ( $v_{ex_1}, v_{ex_2}$ )
- 8    $Frente_{in} =$  Algoritmo de Extração de Frente ( $v_{in_1}, v_{in_2}$ )
- 9    $(M_3, Buracos) =$  Algoritmo de Ligação de Vértices ( $Frente_{ex}, Frente_{in}$ )
- 10  **se**  $Buracos \neq \emptyset$  **então**
- 11    **para cada** *contorno do buraco*  $V_c$  *de*  $Buracos$  **faça**
- 12    | Algoritmo de Preenchimento de Buraco ( $V_c$ )
- 13    **fim**
- 14  **fim se**
- 15 **fim**

---

#### 4.4.1 Eficiência

Realizou-se uma sequência de costuras de malhas sobre a cabeça e o resultado final foi a malha sobre a cabeça ilustrada na Figura 4.17. Mediu-se o tempo gasto para costurar cada malha. A cada passo uma região sobre a cabeça foi demarcada. A costura é realizada par a par,

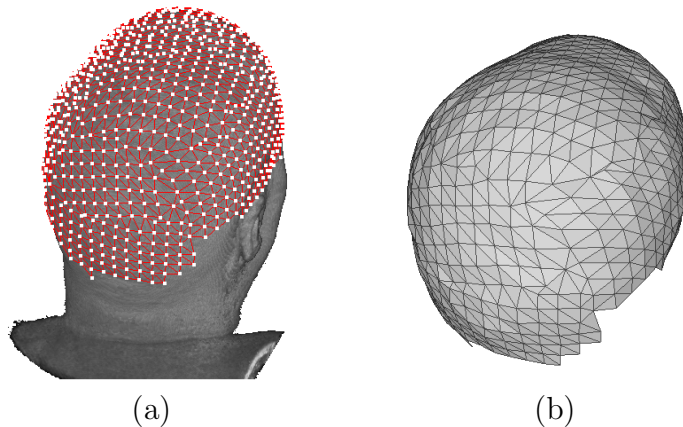


Figura 4.17: Resultado final de uma sequência de costura de malhas. (a) Malha sobre a cabeça é o resultado de uma sequência de costuras. (b) Visualização somente da malha sobre a cabeça.

isto é, a malha anterior e a atual são costuradas antes de uma nova região ser demarcada. A malha sobre a cabeça, ilustrada pela Figura 4.17, foi criada através de uma sequência de costura de malhas mostradas na Figura 4.18. A Tabela 4.1 apresenta o número de faces das malhas a serem costuradas, o número de faces da nova malha após a costura, o tempo gasto na etapa da remoção e ligação das malhas. O tempo total da costura é a soma das duas etapas. Os tempos medidos são, respectivamente, das costuras das malhas das Figuras 4.18(a), (b), (c), (d), (e) e

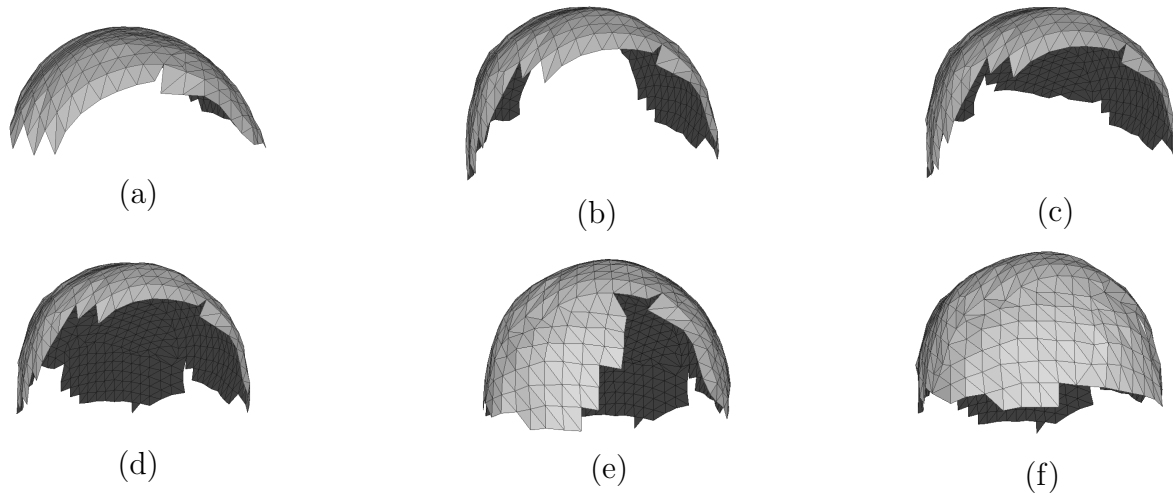


Figura 4.18: Sequência de costura entre malhas.

(f). O tempo da costura está relacionado ao tamanho da região de sobreposição e na maneira como as malhas são ligadas. Quanto maior for a região de sobreposição, maior será o tempo na etapa de remoção, e quanto maior for o número de buracos formados durante a ligação das malhas, maior será o tempo na etapa de ligação. Quando ambas as condições são favoráveis, o tempo total gasto pode ser muito pequeno mesmo que as malhas a serem costuradas sejam muito grandes como é o caso apresentado na penúltima linha da Tabela 4.1.

| Malha #1 | Malha #2 | Malha #3 | Remoção(s) | Ligação(s) | Tempo Total(s) |
|----------|----------|----------|------------|------------|----------------|
| 372      | 361      | 665      | 0,032      | 0,440      | 0,472          |
| 665      | 352      | 940      | 0,059      | 0,325      | 0,384          |
| 940      | 167      | 1054     | 0,094      | 0,076      | 0,170          |
| 1054     | 244      | 1204     | 0,073      | 0,138      | 0,211          |
| 1204     | 126      | 1302     | 0,039      | 0,053      | 0,092          |
| 1302     | 125      | 1361     | 0,043      | 0,073      | 0,116          |

Tabela 4.1: Tempo gasto para costurar as Malhas #1 e #2.

#### 4.4.2 Qualidade

A Figura 4.19 apresenta algumas costuras entre malhas para ilustrar visualmente a qualidade da costura obtida com o nosso algoritmo proposto. Podemos extrair a superfície da cabeça realizando demarcações e costuras em toda a cabeça. A Figura 4.20 apresenta uma sequência de costuras sendo realizadas para extrair a superfície da cabeça. A Figura 4.22 mostra a superfície final extraída e a cabeça da qual a superfície foi extraída. Utilizando a impressora 3D *Sinterstation HiQ<sup>TM</sup> SLS<sup>®</sup> system* do Centro de Tecnologia da Informação Renato Archer (CTI), o modelo mostrado na Figura 4.20(c) foi impresso com o material poliamida para propiciar uma outra maneira de avaliar a qualidade da superfície gerada. A Figura 4.21 apresenta o resultado da impressão: na Figura 4.21(a) o modelo aramado e na Figura 4.21(b) o modelo

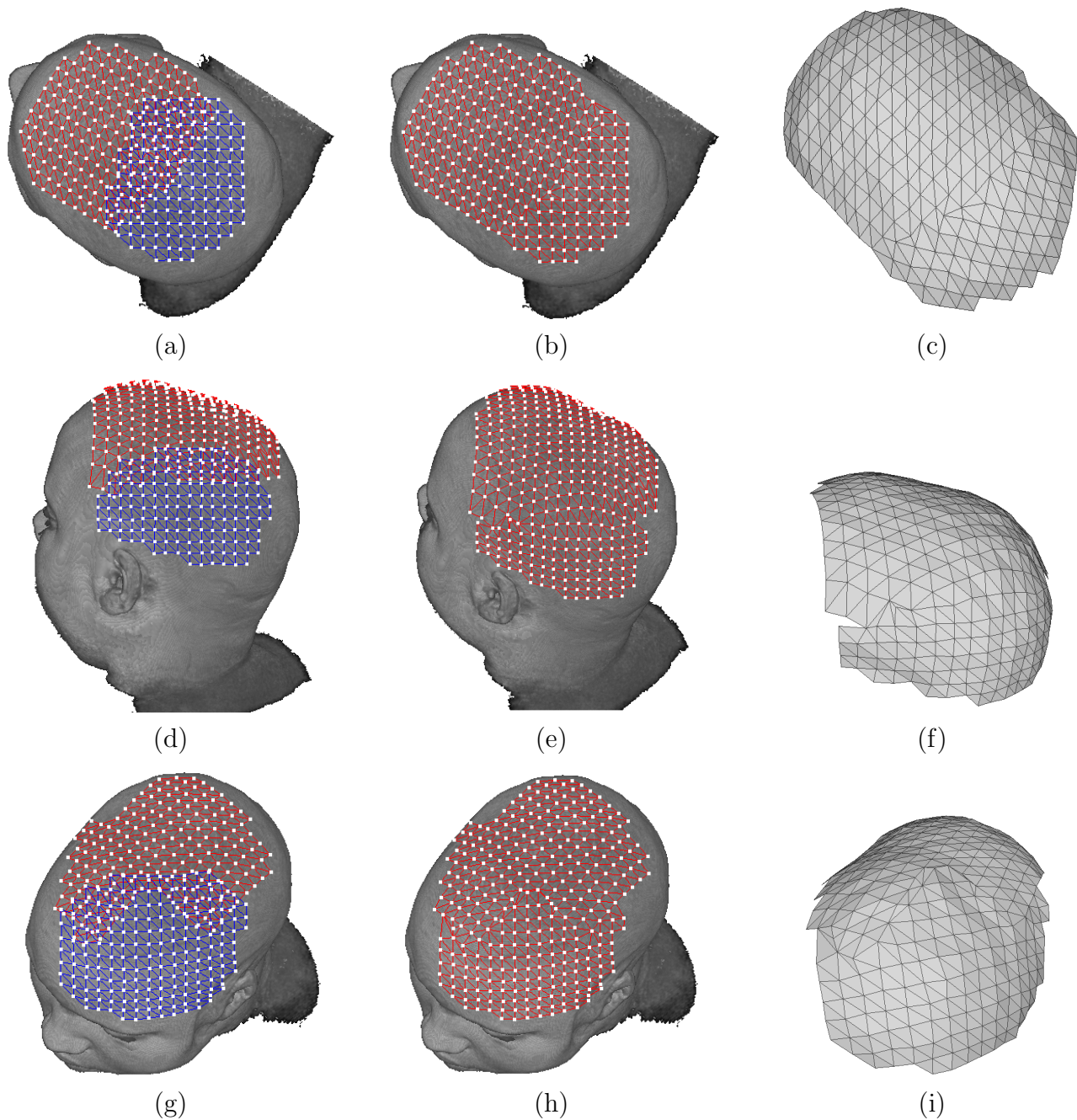


Figura 4.19: Costura entre duas malhas. Malha  $M_1$ , em vermelho, Malha  $M_2$ , em azul.

preenchido. Aumentando a resolução da malha pode-se alcançar uma melhor aproximação da superfície da cabeça. Contudo, não podemos assegurar o mesmo resultado para um volume de geometria arbitrária, diferente do nosso escopo de trabalho.

## 4.5 Discussões

Como nosso algoritmo proposto é baseado na visibilidade, a operação de costura só é possível caso a região onde se deseja costurar seja visível. Tomou-se como premissa que as demarcações

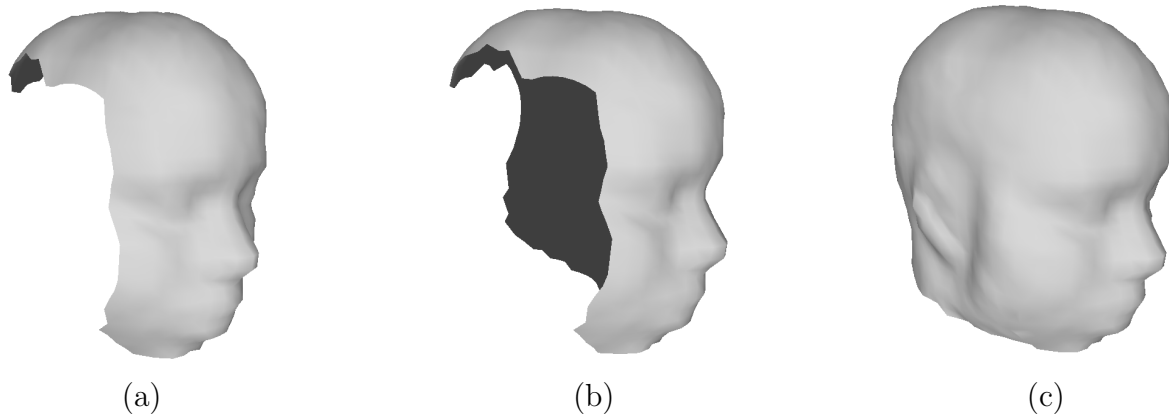


Figura 4.20: Extração da superfície da cabeça.



Figura 4.21: Impressão 3D da superfície obtida através do algoritmo de costura. (a) Modelo não aramado. (b) Modelo aramado.

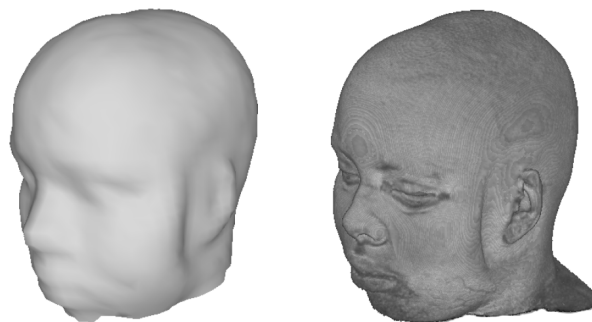


Figura 4.22: Superfície extraída, (imagem à esquerda), a partir da cabeça, (imagem à direita).

sobre a cabeça sempre tenham somente dois pontos de interseção. A Figura 4.23(a) apresenta uma demarcação com dois pontos de interseção e a Figura 4.23(b) mostra uma com mais de dois pontos de interseção. O nosso algoritmo não trata ainda o segundo caso. Conseguiu-se realizar a costura entre malhas em tempo interativo, como mostrado pela Tabela 4.1. Contudo, o algoritmo de costura não garante que o resultado seja livre do cruzamento de arestas em dois casos. O primeiro caso é no processo de contração dos vértices. Quando um vértice  $v$  é contraído com outro vértice  $w$ , mas a aresta  $e_v$  é maior do que a aresta  $e_w$ , a aresta  $e_v$  pode cruzar a aresta  $e_w$  (Figura 4.24). Uma solução para evitar esse cruzamento seria dividir a aresta  $e_v$  em



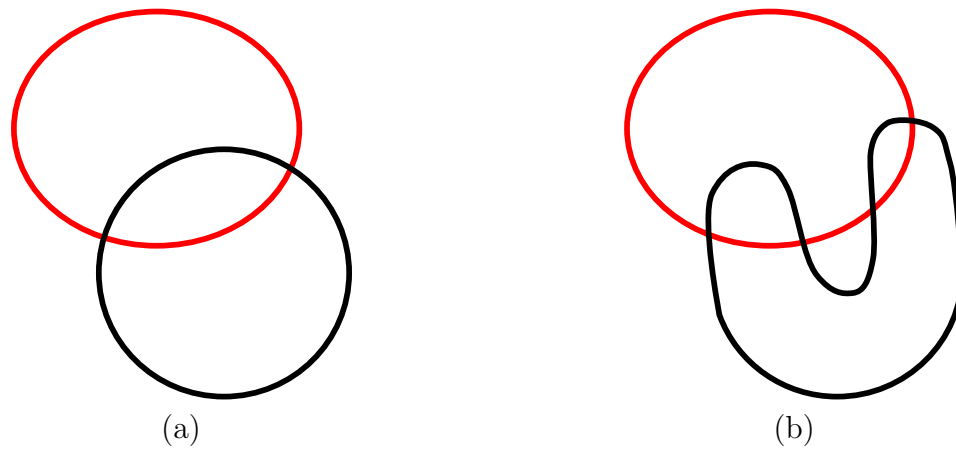


Figura 4.23: Demarcações: (a) com dois pontos de interseção e (b) com mais de dois pontos de interseção.

duas arestas. O segundo caso é no preenchimento dos buracos que podem ter sido criados após a contração dos vértices, como ilustra a Figura 4.25. O nosso algoritmo de preenchimento de buraco só assegura que não haja cruzamentos entre arestas se os buracos forem convexos. Uma solução seria dividir buracos côncavos em convexos (Keil 2000). Outro destaque é em relação à

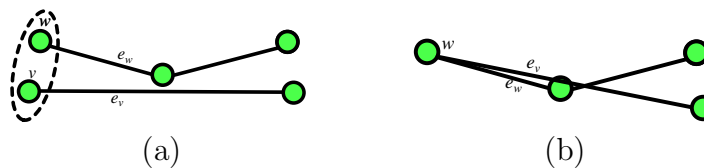


Figura 4.24: Cruzamento de arestas durante a contração do vértice.



Figura 4.25: Cruzamento de arestas durante o preenchimento do buraco. O triângulo com vértices,  $(v_{k-1}, v_k, v_{k+1})$ , é formado pois o menor ângulo interno encontrado foi entre as arestas  $(v_k, v_{k-1})$  e  $(v_k, v_{k+1})$ . Contudo a formação desse triângulo, ocasiona o cruzamento de arestas.

proximidade das frentes internas e externas utilizadas na costura. A pequena distância entre elas assegura um geometria mais suave na malha final. Por outro lado, aumenta a probabilidade de formação de faces degeneradas como mostradas pela Figura 4.26. Para estes casos foi aplicado o algoritmo de simplificação antes do deslocamento da malha resultante. Porém, em muitos casos, tal degeneração torna mais propícia a formação de faces degeneradas ao longo da geração de malhas de *offset*, aumentando a quantidade de chamadas do algoritmo de simplificação. Em





Figura 4.26: Faces degeneradas formadas durante a costura das malhas.

consequência, o tempo da reformatação aumenta. Para contornar esse problema poderíamos retriangular a malha final.

Embora o algoritmo de costura poderia ser utilizado na extração da superfície do volume, desenhando várias malhas e costurando-as par a par, vale ressaltar aqui que o objetivo da nossa costura não é esse. Existem algoritmos clássicos, como o *marching cubes* (Foley et al. 1990), mais apropriados para extração da superfície de um volume representado por enumeração espacial.

## Resultados

Neste capítulo são apresentados a integração do algoritmo de reformatação curvilínea com o algoritmo de costura de malhas para realizar a reformatação curvilínea arbitrária e a sua integração ao protótipo de visualização de neuroimagem, VMTK.

### 5.1 Reformatação Curvilínea em Extensão Arbitrária

Integrando as propostas apresentadas nos Capítulos 3 e 4 um usuário poderia selecionar mais de uma área sobre a cabeça de um paciente para realizar a reformatação curvilínea na extensão desejada, sem a restrição de que seja uma área visível. O **Algoritmo 13** descreve a nossa proposta de um algoritmo de reformatação curvilínea em extensão arbitrária a partir dos resultados obtidos. A entrada do algoritmo é composto pelas malhas que são criadas a partir das regiões demarcadas sobre a cabeça através de uma interface de interações. Estas malhas são costuradas par a par (Capítulo 4) antes da chamada do algoritmo de geração de malhas de *offset* e do algoritmo de voxelização para construir o volume de controle utilizado pelo módulo de renderização para selecionar os *voxels* visíveis (Capítulo 3).

O **Algoritmo 13** descreve todo o processo de interação para obter uma reformatação curvilínea somente na área definida pelo usuário. As regiões são demarcadas pelo usuário com o *mouse*. A partir das amostras adquiridas pelo *mouse* é construída o contorno da região de interesse  $R$  (linha 3–4). Uma malha é criada a partir da região demarcada e armazenada em um conjunto de *Malhas* (linha 5–6). Caso o usuário queira costurar as malhas, as duas últimas malhas são passadas para o algoritmo de costura (linha 9), pois a costura é feita par a par. Em seguida se o usuário quiser realizar a reformatação curvilínea é passado para o algoritmo todas as malhas do conjunto *Malhas* (linha 12).

### 5.2 Integração a um Ambiente de Exploração Visual de Neuro-Imagens

Observe no **Algoritmo 13** que as intervenções dos usuários são fundamentais para definir o seu fluxo de controle. Como integrar os dois algoritmos propostos com as ações dos usuários

**Algoritmo 13:** Algoritmo de Reformatação Curvilínea

---

**Saída:** Reformatação curvilínea sobre região demarcada

```

1 início
2   se usuário deseja demarcar região então
3     |   Captura as amostras selecionada pelo usuário
4     |   Constrói o contorno de região  $R$  com as amostras
5     |   malha,  $M_r$ , é criada a partir da região amostrada de  $R$ 
6     |   coloca  $M_r$  em Malhas
7   fim se
8   se usuário deseja costurar malhas então
9     |   Algoritmo de Costura de Malhas ( $Malhas[n-1]$ ,  $Malhas[n]$ ) (Algoritmo 12)
10  fim se
11  se usuário deseja reformatar regiões demarcadas então
12  |   Algoritmo de Reformatação Curvilínea (Malhas) (Algoritmo 4)
13  fim se
14 fim

```

---

num único ambiente? Reavaliando a implementação da versão de reformatação curvilínea proposta em (Wu et al. 2012), decidiu-se manter o mesmo ambiente de visualização exploratória, VMTK<sup>1</sup> (VMTK 2014), em que a versão anterior foi integrada. O custo de integração é baixo: basta adicionar a função da linha 9 e substituir a função da linha 12 do **Algoritmo 13** na versão existente.

### 5.2.1 VMTK

O VMTK é um protótipo de visualização interativa que fornece ferramentas para auxiliar os médicos no diagnóstico de lesões corticais em desenvolvimento por um grupo de pesquisadores da Faculdade de Engenharia Elétrica e de Computação da Unicamp. A interface do VMTK é ilustrada pela Figura 5.1. Após as neuroimagens serem importadas para o sistema, as funcionalidades de interações podem ser selecionadas. A inspeção visual pode ser feita utilizando as três visões 2D, sagital, axial e coronal ou com a visão 3D. Foi desenvolvida uma interface interativa de pintura sobre o escalpo do paciente para amostrar os pontos necessários para gerar a malha original sobre a qual é aplicado o algoritmo de geração de malhas de *offset*.

A versão de VMTK com que trabalhamos inicialmente já inclui a versão original de reformatação curvilínea, que só permite a seleção de uma região visível, como uma ferramenta de exploração. A Figura 5.2 apresenta o diagrama de blocos das principais funções desta versão do VMTK. Observe que já se encontra integrado um módulo de interações que pré-processa as ações dos usuários, transformando-as em amostras sobre o escalpo do paciente que são utilizadas para definir o contorno da região de interesse  $R$ . Para realizar esta operação basta clicar no botão “Paint”. E se quiser reformatar a região selecionada, basta clicar no botão “Crop”. Ressaltamos que, uma vez concluído este procedimento, o usuário pode explorar o cérebro do paciente em diferentes níveis de profundidade movimentando o *slider* da interface.

---

<sup>1</sup> *Visual Manipulation ToolKit*

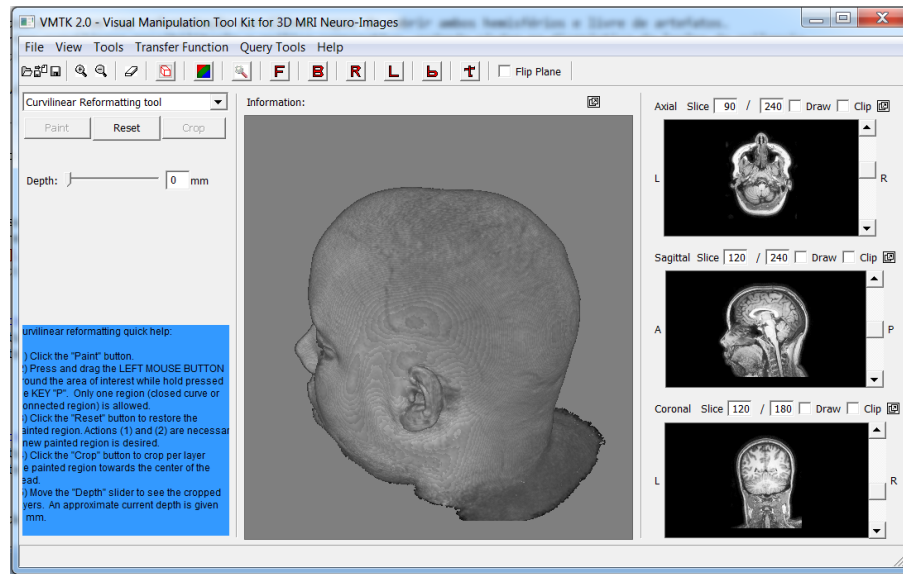


Figura 5.1: Interface de VMTK.

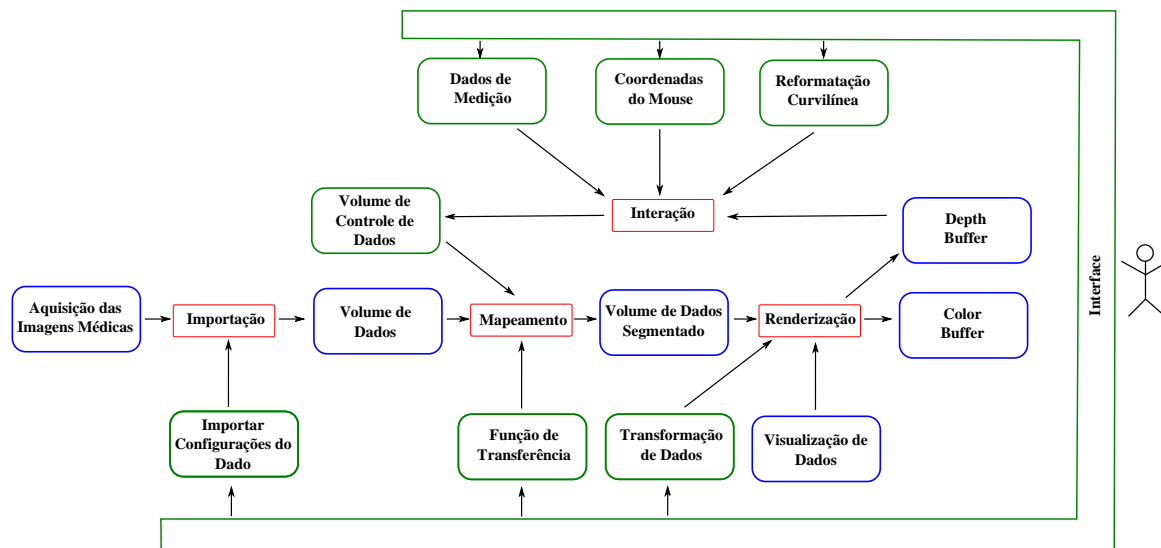


Figura 5.2: Arquitetura de VMTK.

Dispondo desta infra-estrutura, a integração da nova versão do algoritmo de reformatação curvilínea foi bastante simples. Tivemos que adicionar o módulo costura e substituir o módulo de reformatação curvilínea, como esquematiza a Figura 5.3. Em termos da interface, só tivemos que adicionar mais um botão “Sew” para diferenciar a ação na linha 9 do **Algoritmo 13** das outras duas operações.

Como resultado, o procedimento é análogo ao da versão anterior. Ele consiste em importar a neuroimagem (Figura 5.4(a)), remover os ruídos (Figura 5.4(b)), selecionar a operação “Paint” da ferramenta de reformatação e demarcar uma região sobre o volume (Figura 5.4(c)). Após a demarcação, a malha é criada (Figura 5.4(d)). Em seguida, giramos a cabeça e realizamos outra demarcação (Figura 5.4(e)), criando uma nova malha (Figura 5.4(f)). Ao apertar o botão “Sew” as duas malhas são costuradas e a reformatação curvilínea só é realizada quando o usuário

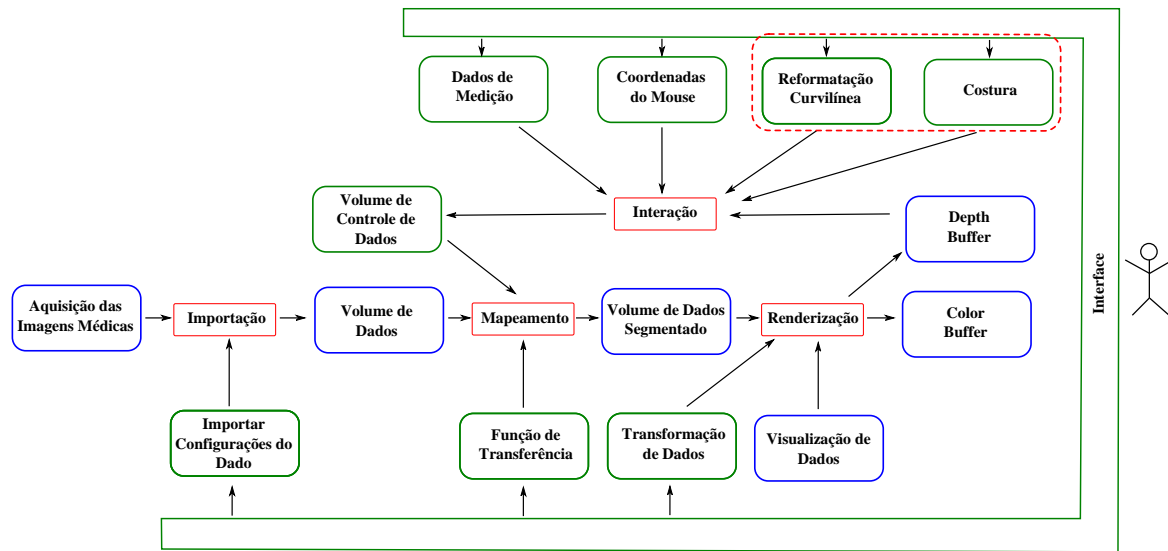


Figura 5.3: Nova arquitetura de VMTK após as alterações na reformatação curvilínea e inserção do módulo de interação costura.

clique no botão “Crop”(Figura 5.4(g)) e (Figura 5.4(h)).

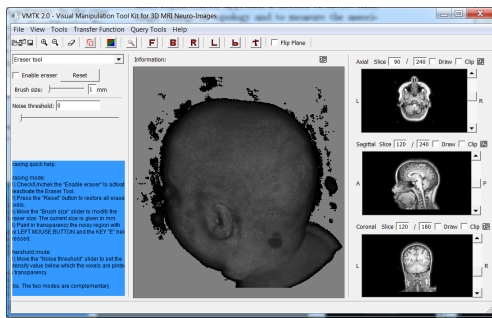
### 5.3 Experimentos

Com intuito de avaliar a interatividade do procedimento proposto, realizamos algumas reformatações curvilíneas em áreas de tamanhos diferentes e medimos os tempos das operações. A plataforma de teste foi um *desktop* Intel <sup>®</sup>Core i7 2.8 GHz com 8GB de RAM e placa de vídeo NVIDIA GeForce GTX 650 Ti com 2GB de VRAM, com sistema operacional *Windows* 7 e *Ubuntu* 14.02. A linguagem utilizada foi C++ e as bibliotecas usadas foram, *wxWidgets* e *OpenGL*. A Tabela 5.1 apresenta os tempos gastos pelos algoritmos de simplificação e de voxelização e o tempo total dos dois passos.

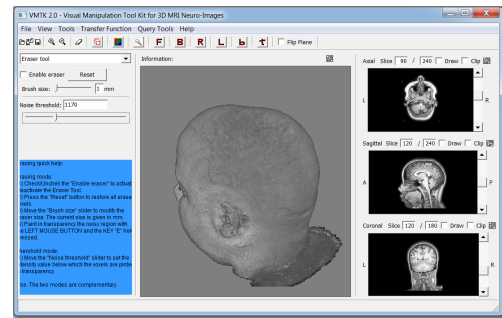
|            | # Facetas | Simplificação(s) | Voxelização(s) | Total Aproximado(s) |
|------------|-----------|------------------|----------------|---------------------|
| Figura 5.5 | 1335      | 15               | 19             | 34                  |
| Figura 5.6 | 1070      | 11               | 16             | 27                  |
| Figura 5.7 | 806       | 7                | 15             | 22                  |
| Figura 5.8 | 632       | 4                | 11             | 15                  |

Tabela 5.1: Tempo gasto para realizar a reformatação curvilínea. Tempo médio: 24,5s.

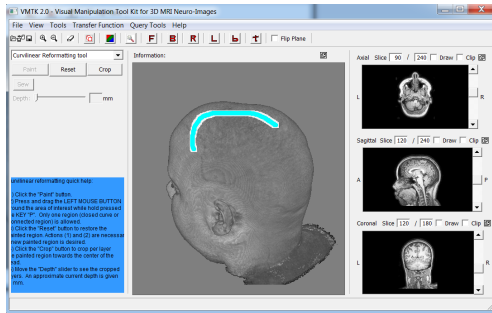
As Figuras 5.5 e 5.6 ilustram a reformatação curvilínea sendo realizada em toda parte superior da cabeça. Vale observar que, diferente da reformatação proposta por (Huppertz et al. 2008) e (Bergo & Falcao 2006), o espaço nativo do paciente é preservado e a reformatação é realizada somente na região demarcada pelo usuário. As Figuras 5.7 e 5.8, apresentam cortes curvilíneos parciais.



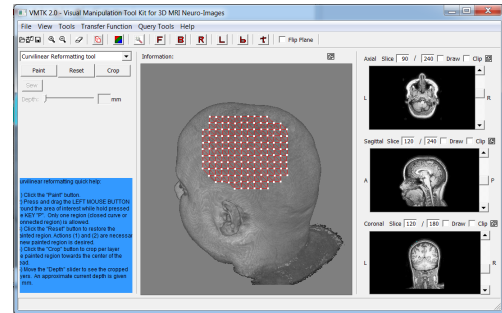
(a)



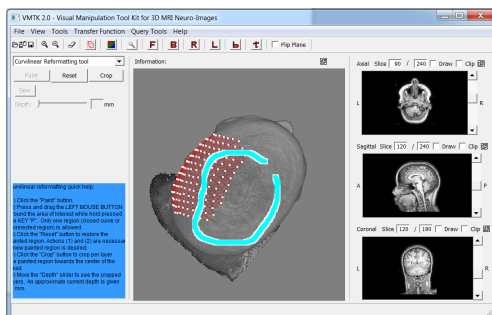
(b)



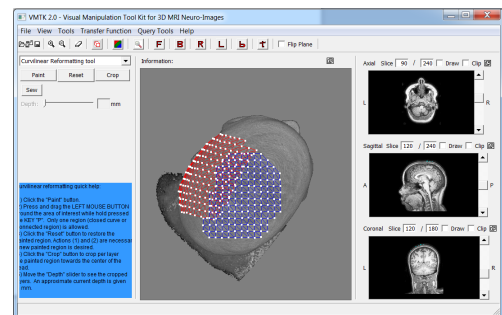
(c)



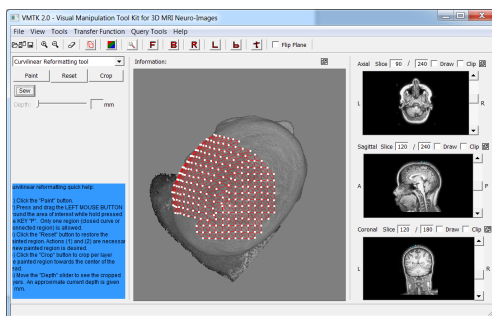
(d)



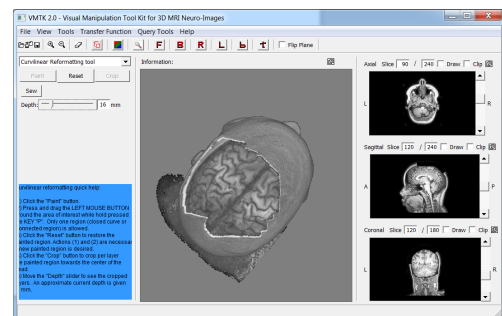
(e)



(f)



(g)



(h)

Figura 5.4: Procedimento da reformatação curvilínea: (a) importação da neuroimagem, (b) remoção de ruídos, (c) demarcação da região de interesse e (d) criação da malha. (e) Em seguida uma nova região é demarcada. (f) Novamente a malha é criada. (g) As duas malhas são costuradas e (h) a reformatação curvilínea é realizada.

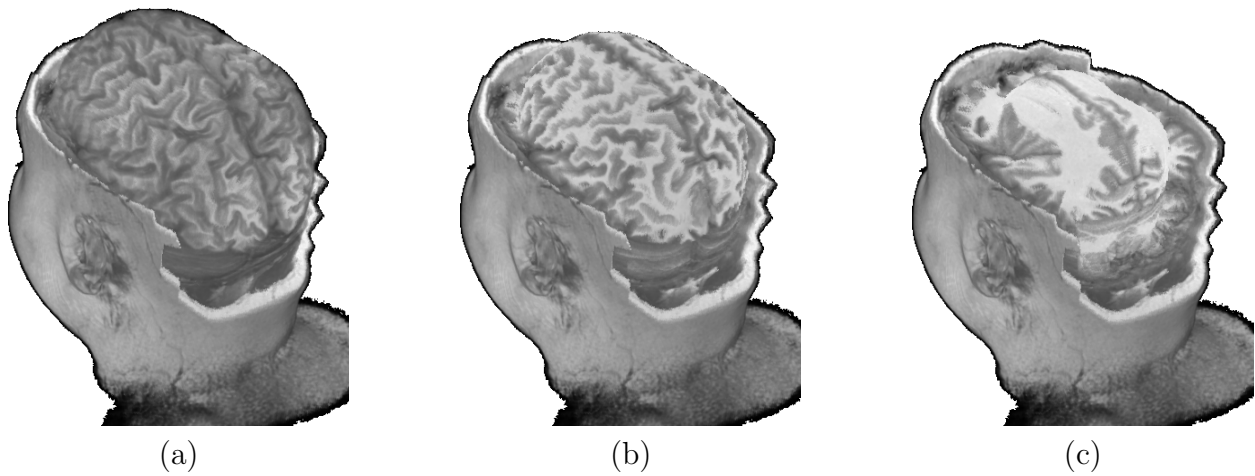


Figura 5.5: Reformatação curvilínea sendo realizada em toda parte superior da cabeça. Profundidade: (a) 16 mm, (b) 23 mm e (c) 33 mm.

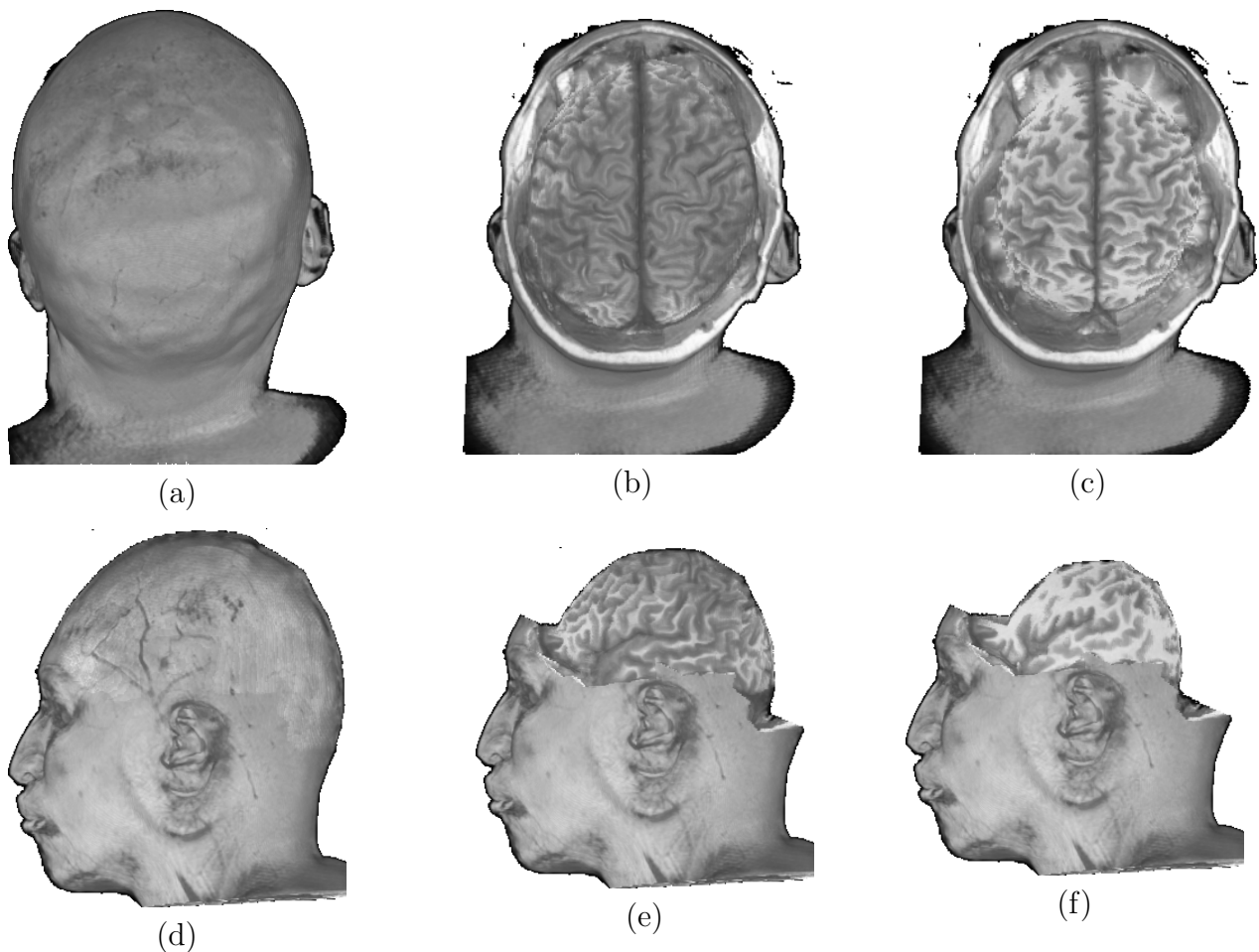


Figura 5.6: Reformatação curvilínea sendo realizada em toda parte superior da cabeça. Profundidade: (a) e (d) 0mm, (b) e (e) 25mm e (c) e (f) 35mm.

## 5.4 Diagnóstico de Displasia Cortical Focal

A neuroimagem é bastante utilizada para diagnósticos não invasivo de lesões do tipo displasia cortical focal. Estas lesões podem ser visualmente identificadas pelo espessamento da espessura

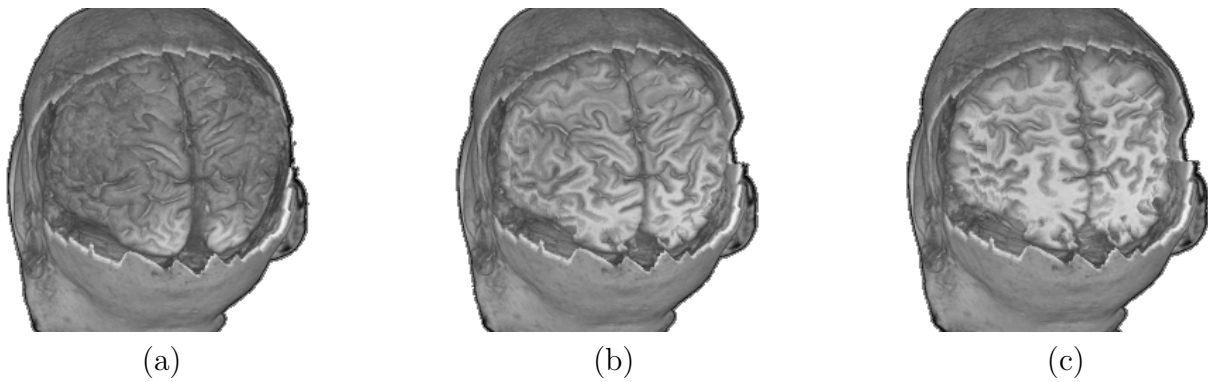


Figura 5.7: Reformatação curvilínea sendo realizada na parte posterior da cabeça. Profundidade: (a) 15mm, (b) 23 mm e (c) 31mm.

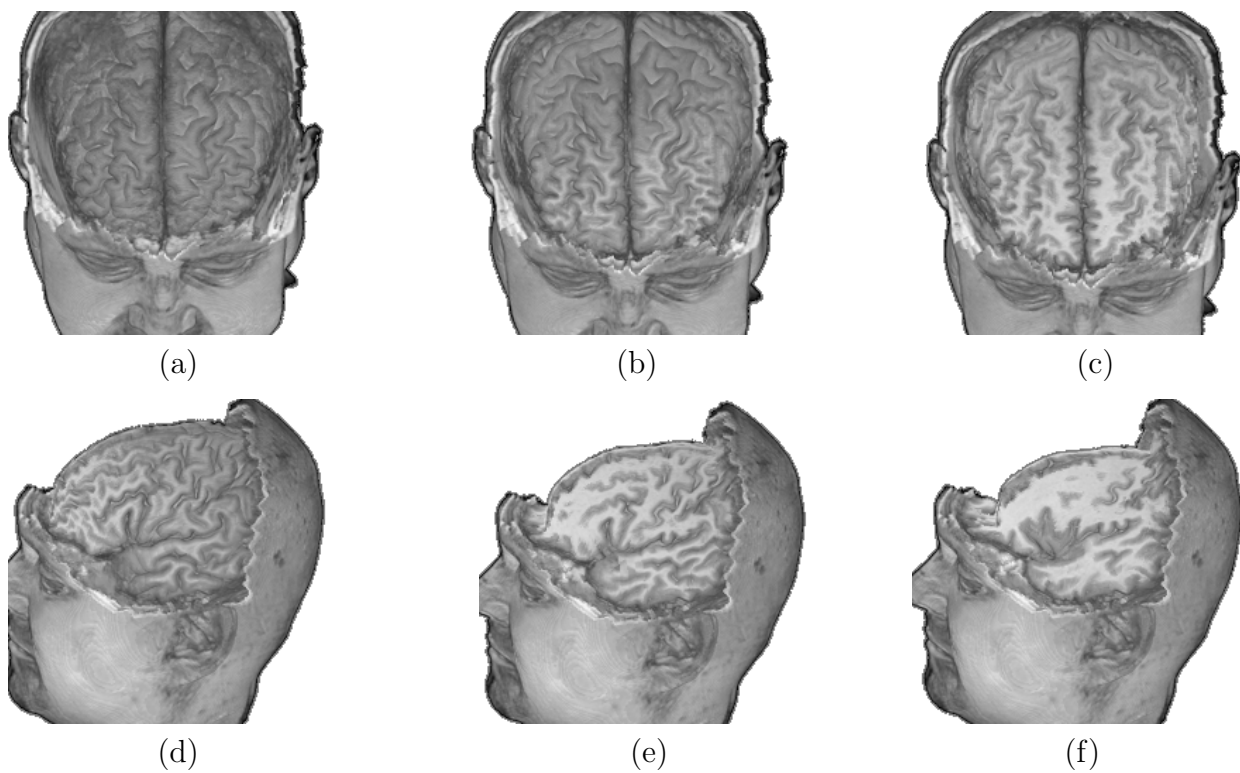


Figura 5.8: Reformatação curvilínea sendo realizada na parte frontal da cabeça. Profundidade: (a) 12mm, (b) 18mm, (c) 24mm, (d) 22mm, (e) 30mm e (f) 38mm.

cortical, alteração do padrão de giro ou pelo borramento na transição entre a substância branca e cinzenta do cérebro. Bastos *et al.* notaram que muitas lesões sutis, que não são visíveis em reformatações multiplanares, podem ser visualmente perceptíveis em reformatações curvilíneas conduzidas com referência em superfície cortical (Bastos et al. 1999). Por simplicidade computacional, propomos fazer as reformatações a partir do escalpo do paciente. Através dos testes realizados, o nosso procedimento se mostrou satisfatório para visualizar lesões que são suspeitas de focos epileptogênicos como ilustram alguns casos clínicos nas Figuras 5.9, 5.10 e 5.11. Cada caso é apresentado utilizando a reformatação curvilínea proposta e a reformatação multiplanar.



No primeiro caso a região suspeita se encontra no giro pré-central esquerdo, próximo da fissura longitudinal (Figura 5.9). Observe na Figura 5.9(a) um borramento sutil na transição entre a massa cinzenta e a massa branca, imperceptível em reformatações multiplanares exibidas nas Figuras 5.9.(b)–(d).

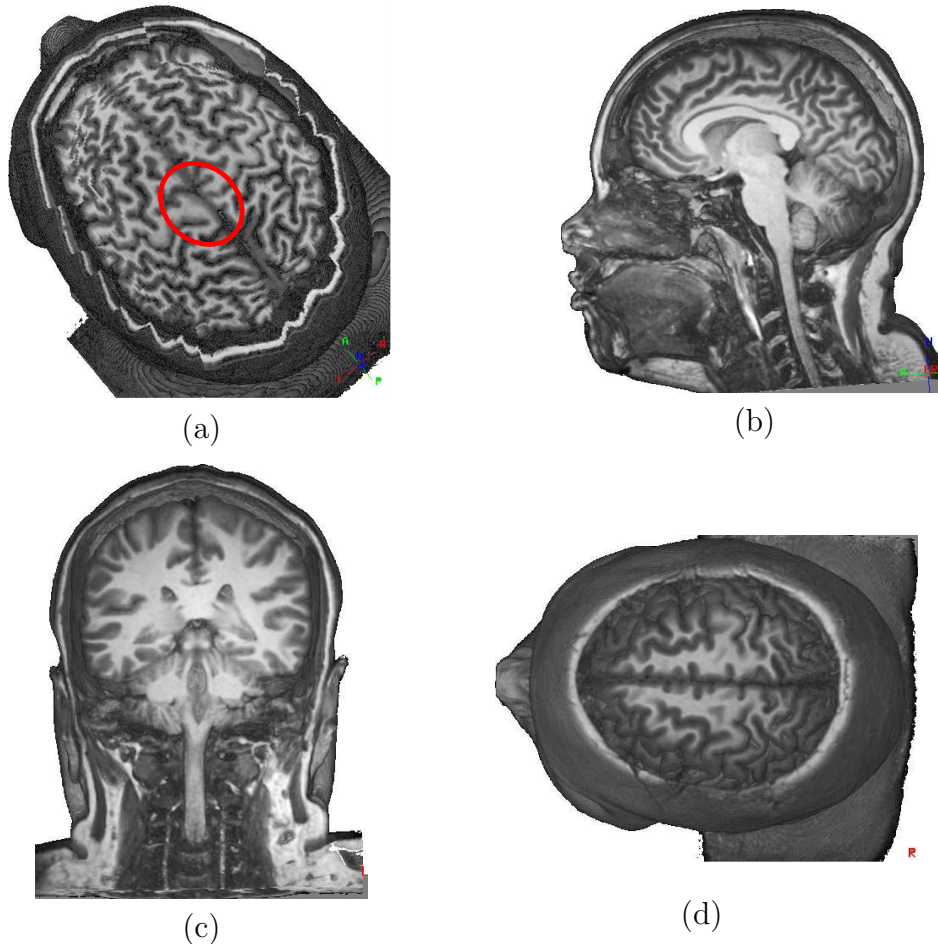


Figura 5.9: Suspeita de lesão de displasia cortical focal: (a) vista do topo do cérebro usando a reformatação curvilínea, (b) em corte sagital, (c) em corte coronal e (d) em corte axial.

Para o próximo caso a região suspeita se encontra no giro frontal médio direito, sutilmente visível no corte coronal como destacado na Figura 5.10(d). A visualização da reformatação curvilínea mostrou um espessamento cortical na junção dos sulcos destacados nas Figuras 5.10(a) e (c), corroborando o achado feito pela reformatação multiplanar. O fato da reformatação ser de extensão arbitrária permitiu ainda uma análise visual comparativa dos padrões dos giros e sulcos entre os dois hemisférios cerebrais, como ilustra a Figura 5.10(a) e (b).

No último caso foi observada, através dos estudos das imagens reformatadas multiplanarmente, alterações no padrão de sulcos e giros com leve espessamento cortical no giro frontal superior direito e no giro frontal médio direito (Figura 5.11(d)). Com a reformatação curvilínea, tal alteração do padrão ficou mais visível permitindo identificar com maior precisão a extensão da alteração (Figura 5.11(a) e (c)). Além disso, o refatiamento curvilinear em extensão maior permitiu uma investigação comparativa dos padrões do lado direito com os do lado esquerdo

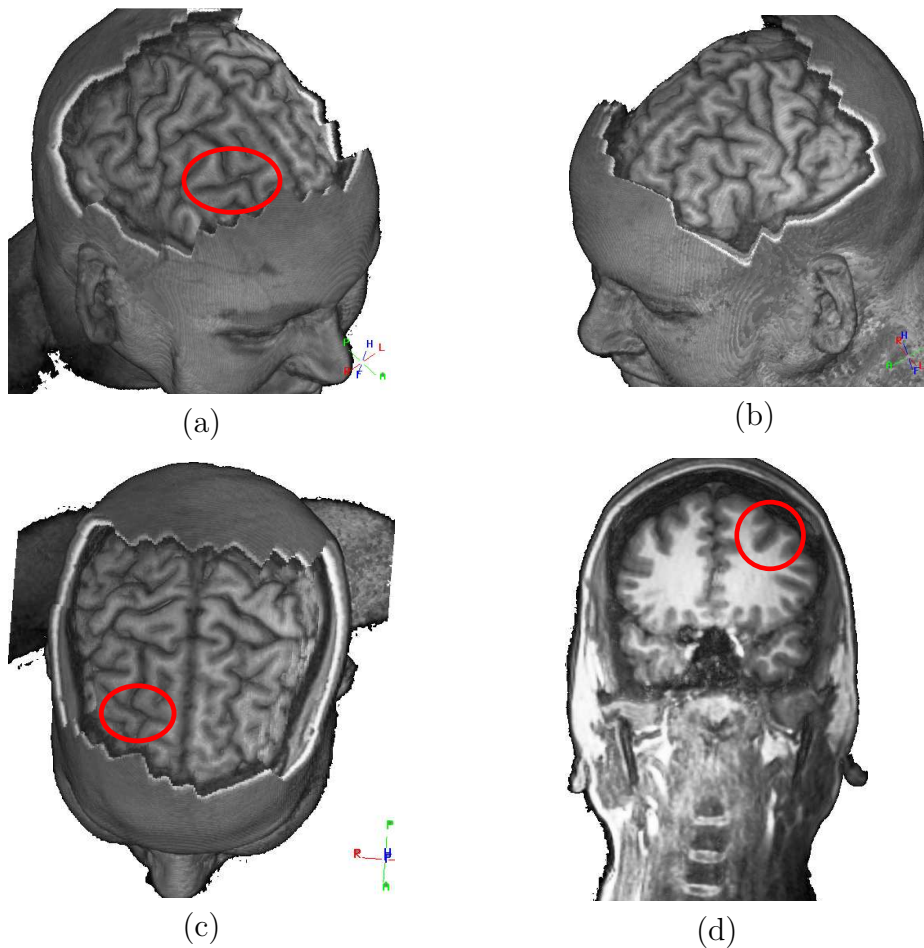


Figura 5.10: Suspeita de lesão de displasia cortical focal: (a) vista do lado direito, (b) lado esquerdo, (c) e topo da cabeça após a reformatação curvilínea. (d) Reformatação multiplanar

(Figura 5.11(a) e (b)).

## 5.5 Discussões

A reformatação curvilínea proposta cria uma malha baseada na região amostrada sobre o volume, nesse caso, sobre o escalpo do paciente. Contudo, a nossa premissa é que a superfície da cabeça do paciente seja próxima da superfície do cérebro, ou seja, uma superfície curvilínea e sem picos. Entretanto, nem toda superfície atende a essa premissa, conforme apresentado pela Figura 5.12. Nesses casos não podemos garantir a consistência entre o que se vê e o que é de fato revelado pelo processo de uma reformatação curvilínea. Como a malha criada é uma aproximação da região demarcada sobre a cabeça, ao ser deslocada, a malha rotula os *voxels* que ela ocupa no volume. Como a geometria do escalpo é diferente do envoltório do córtex cerebral, a malha, que é uma aproximação do escalpo, pode não alcançar a superfície da cabeça por inteiro, apenas algumas partes, como mostrado pela Figura 5.13. Porém todos recebem o mesmo rótulo em termos do valor de profundidade. Assim, nem todos os *voxels* que receberam o mesmo rótulo deveriam ser vistos. Embora não usual, os nossos colegas médicos do Laboratório de

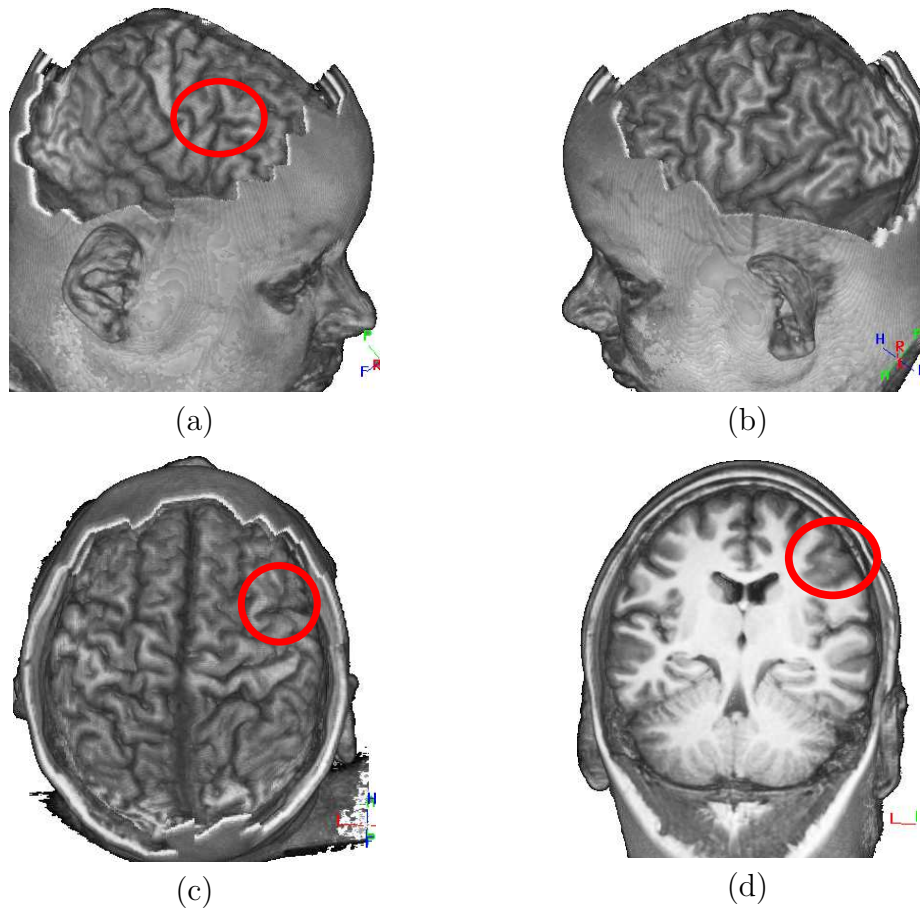


Figura 5.11: Suspeita de lesão de displasia cortical focal: (a) vista do lado direito, (b) lado esquerdo, (d) e topo da cabeça após a reformatação curvilínea. (c) Reformatação multiplanar

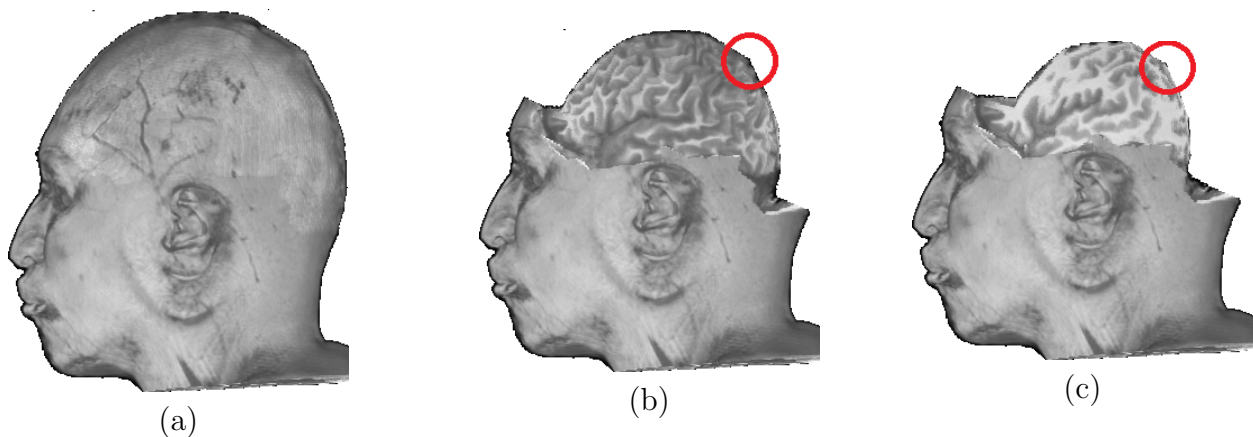


Figura 5.12: Superfície da cabeça que apresenta alguns picos devido a não suavidade da superfície.

Neuroimagem (LNI) da UNICAMP, acreditam que tais discrepâncias não devem comprometer o resultado de um diagnóstico desde que a distância entre a geometria da malha e do escalpo seja informada de acordo com o deslocamento da malha. Uma possível solução seria construir a malha a partir de amostras sobre o córtex. Assim, a malha teria a mesma superfície do córtex cerebral.

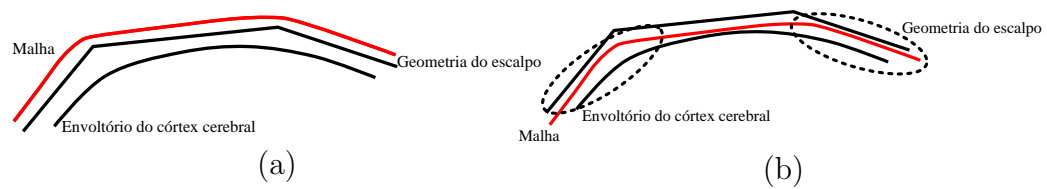


Figura 5.13: A geometria do escalpo se diferencia do envoltório do córtex cerebral. (a) A malha, uma aproximação da geometria do escalpo, é deslocada. (b) Quando a malha tangencia o envoltório do córtex cerebral, existem algumas regiões da malha que ainda não tangenciaram o envoltório.

## Conclusões e Trabalhos Futuros

O objetivo deste trabalho foi proporcionar uma ferramenta de exploração das neuroimagens de forma a dar suporte aos médicos para identificar lesões corticais sutis com base na análise comparativa dos dois hemisférios. Esta ferramenta consiste em reformatação curvilínea da região selecionada pelo usuário, atuando de forma complementar às reformatações multiplanares axiais, sagitais e coronais. Para alcançar esse objetivo teve-se que:

1. aprimorar o algoritmo de geração de malhas de *offset*, proposto em (Wu et al. 2012) e
2. estender a aquisição de amostras de um ponto de vista à aquisição de múltiplos pontos de vista.

A nossa hipótese para tentar aprimorar o algoritmo de geração de malhas de *offset*, proposto em (Wu et al. 2012), era que, utilizando um algoritmo de simplificação, poderíamos evitar as auto-interseções locais. Aplicando um algoritmo de simplificação conseguiu-se redimensionar as áreas das facetas triangulares das malhas tirando-as do estado de degeneração. A nossa hipótese foi validada através dos testes realizados no Capítulo 3. Para estender a reformatação curvilínea a múltiplos cortes, nossa hipótese foi que, explorando devidamente os recursos disponíveis nas GPUs, seria possível realizar a costura entre malhas de geometria arbitrária, sem recorrer ao complexo cômputo de interseções. Essa hipótese foi validada no Capítulo 4. Testamos o nosso algoritmo com as imagens adquiridas pelo escaneador Philips Achieva 3T do Hospital das Clínicas da Universidade Estadual de Campinas. As dimensões dos volumes que utilizamos foram  $180 \times 240 \times 240$ . O tempo médio para processar completamente a reformatação curvilínea cobrindo quase todo o couro cabeludo de um paciente, como na Figura 5.5, é menor do que 1 minuto, sendo que a costura das malhas demorou menos de 1 segundo. Estes resultados propiciam a utilização do nosso algoritmo em um ambiente interativo.

Concluiu-se algumas vantagens e desvantagens de se utilizar a reformatação curvilínea proposta em nosso trabalho para neuroimagens. As vantagens são:

1. Poucas Demarcações: Não são necessárias várias demarcações sobre várias fatias, como no *BrainSight* (Bastos et al. 1999). Apenas demarcações na região de interesse são necessárias.
2. Tempo Interativo: Supôs-se que o corte ilustrado pelas Figuras 5.5 e 5.6 sejam o maior corte de interesse possível, pois ele fornece toda a visão do córtex cerebral. Pelos tempos

apresentados na Tabela 5.1 podemos afirmar que para os volumes de resolução  $180 \times 240 \times 240$  o tempo do processo é interativo.

3. Espaço Nativo: Nossa reformatação curvilínea proposta acontece no espaço nativo do paciente. Esta condição torna o nosso procedimento favorável para reuso em neuroplanejamento e neuronavegação.
4. Corte Restrito: A reformatação ocorre somente na região de interesse, preservando as estruturas de referência.

Das desvantagens destacamos:

1. Parâmetros: O critério utilizado para escolha dos parâmetros do algoritmo de simplificação não foi validado exaustivamente.
2. Inconsistência: Quando a superfície da cabeça apresenta picos, os resultados visualizados de uma reformatação curvilínea podem não ser consistentes com os dados esperados. Contudo, mostramos como contornar esse problema na Seção 5.5.
3. Limitações em Costuras: Se a geometria das áreas demarcadas não forem convexas o algoritmo pode falhar como discutimos na Seção 4.5.

Embora a motivação deste trabalho seja aprimorar ferramentas visuais para diagnóstico de focos epileptogênicos por imagens não foram conduzidos experimentos com os médicos para validar o seu valor clínico. Todos os testes apresentados foram realizados de forma conjunta, com o nosso apoio direto para manipular o protótipo. Pretende-se, a curto prazo, redesenhar a interface deste protótipo de acordo com as sugestões do corpo clínico com que temos contato e realizar avaliações de sua usabilidade e funcionalidade com os seus potenciais usuários.

Outra aplicação que vislumbramos ao longo do desenvolvimento deste projeto foi o suporte ao posicionamento preciso de eletrodos ou optodos no couro cabeludo de um paciente para avaliar as atividades epileptiformes nas regiões suspeitas. Para isso, precisamos acoplar um digitalizador ao nosso protótipo e incluir um sistema de calibração do digitalizador em relação ao volume escaneado previamente.

Foi utilizado apenas operações de contrações de *half-edge*. Porém gostaríamos de incluir a operação de *flip* para verificar se temos um ganho de eficiência. Em termos de implementação é importante ressaltar que a estrutura de dados, *half-edge*, usada para representar a malha foi implementada pelo nosso grupo de pesquisa. Ela contribuiu muito no aumento do desempenho dos algoritmos de simplificação e costura. Porém, o seu manuseio requer cuidados, pois é uma estrutura que possui várias referências cuja consistência precisa ser mantida em cada atualização. Existem bibliotecas como *OpenMesh* (OpenMesh 2014) e *CGAL* (CGAL 2014) que oferecem funções de mais alto nível para representar e manipular malhas poligonais. A utilização dessas bibliotecas seria uma alternativa para evitar o manuseio direto das estruturas de dados *half-edge*.

Como trabalho futuro, gostaríamos de analisar outros algoritmos de simplificação e voxelização para comparar com a qualidade e o tempo dos algoritmos que foram utilizados. Vimos que quando duas malhas são costuradas, faces degeneradas podem ser formadas. Isso provoca

---

um aumento do tempo na reformatação curvilínea, pois o algoritmo de simplificação é chamado mais vezes. Realizar a costura sem a formação de faces degenerada deixaria a reformatação mais rápida. Analisar outras maneiras de costura de malhas poderia nos ajudar a solucionar esse problema. Ao final do nosso trabalho foi desenvolvido um algoritmo de reformatação curvilínea que permite de forma interativa explorar o córtex do cérebro humano, a fim de auxiliar os médicos a encontrarem lesões de displasia cortical focal.

# Bibliografia

- Abdel Razek, A. A. K., Kandell, A., Elsorogy, L. G., Elmongy, A. & Basett, A. A. (2009). Disorders of cortical formation: MR imaging features., *AJNR. American journal of neuro-radiology* **30**(1): 4–11.
- Aspert, N., Santa-Cruz, D. & Ebrahimi, T. (2002). Mesh: measuring errors between surfaces using the hausdorff distance, *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, Vol. 1, pp. 705–708 vol.1.
- Bastos, A. C., Comeau, R. M., Andermann, F., Melanson, D., Cendes, F., Dubeau, F., Fontaine, S., Tampieri, D. & Olivier, A. (1999). Diagnosis of subtle focal dysplastic lesions: Curvilinear reformatting from three-dimensional magnetic resonance imaging, *Annals of Neurology* **46**(1): 88–94.
- Baumgart, B. G. (1972). Winged edge polyhedron representation., *Technical report*, Stanford, CA, USA.
- Bergo, F. & Falcao, A. (2006). Fast and automatic curvilinear reformatting of mr images of the brain for diagnosis of dysplastic lesions, *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, pp. 486–489.
- Botsch, M., Pauly, M., Rossl, C., Bischoff, S. & Kobbelt, L. (2006). Geometric modeling based on triangle meshes, *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, ACM, New York, NY, USA.
- CGAL (2014). Computational geometry algorithms library. <http://www.cgal.org/>. Acessado em Agosto de 2014.
- Cignoni, P., Rocchini, C. & Scopigno, R. (1996). Metro: Measuring error on simplified surfaces, *Technical report*, Paris, France, France.
- Dey, T. K., Edelsbrunner, H., Guha, S. & Nekhayev, D. V. (1998). Topology preserving edge contraction, *Publ. Inst. Math. (Beograd) (N.S)* **66**: 23–45.
- Dong, Z., Chen, W., Bao, H., Zhang, H. & Peng, Q. (2004). A smart voxelization algorithm.



- Eisemann, E. & Décoret, X. (2006). Fast scene voxelization and applications, *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D '06*, ACM, New York, NY, USA, pp. 71–78.
- Figueiredo, L. H. D. E. (1999). Intersecting and Trimming Parametric Meshes on Finite-Element Shells, *0*(0): 1–28.
- Foley, J. D., van Dam, A., Feiner, S. K. & Hughes, J. F. (1990). *Computer Graphics: Principles and Practice (2Nd Ed.)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Friston, K., Ashburner, J., Kiebel, S., Nichols, T. & Penny, W. (eds) (2007). *Statistical Parametric Mapping: The Analysis of Functional Brain Images*, Academic Press.
- Garland, M. & Heckbert, P. S. (1997). Surface simplification using quadric error metrics, *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97* pp. 209–216.
- Guibas, L. & Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of voronoi, *ACM Trans. Graph.* **4**(2): 74–123.
- Hadwiger, M., Kniss, J. M., Rezk-salama, C., Weiskopf, D. & Engel, K. (2006). *Real-time Volume Graphics*, A. K. Peters, Ltd., Natick, MA, USA.
- Huppertz, H.-J., Kassubek, J., Altenmüller, D.-M., Breyer, T. & Fauser, S. (2008). Automatic curvilinear reformatting of three-dimensional {MRI} data of the cerebral cortex, *NeuroImage* **39**(1): 80 – 86.
- Jung, W., Shin, H. & Choi, B. K. (2004). Self-intersection Removal in Triangular Mesh Offsetting, **1**: 477–484.
- Kabat, J. & Król, P. (2012). Focal cortical dysplasia - review, *Polish journal of radiology / Polish Medical Society of Radiology* **77**(2): 35–43.
- Karabassi, E.-A., Papaioannou, G. & Theoharis, T. (1999). A fast depth-buffer-based voxelization algorithm, *J. Graph. Tools* **4**(4): 5–10.
- Keil, J. M. (2000). Chapter 11 - polygon decomposition, in J.-R. S. Urrutia (ed.), *Handbook of Computational Geometry*, North-Holland, Amsterdam, pp. 491 – 518.
- Lee, S. K. & Kim, D.-W. (2013). Focal cortical dysplasia and epilepsy surgery., *Journal of epilepsy research* **3**(2): 43–7.
- Liu, S. & Wang, C. C. L. (2011). Fast Intersection-free Offset Surface Generation from Freeform Models with Triangular Meshes, *IEEE Transactions on Automation Science and Engineering* **8**(852): 347–360.
- Mantyla, M. (1988). *Introduction to Solid Modeling*, W. H. Freeman & Co., New York, NY, USA.

- Meneses, M. S., Hertz, A., Gruetzmacher, C., Blattes, S. F., Silva, E. B. d., Vosgerau, R. A., Laroca, H. & Kowacs, P. A. (2006). Epilepsia e desordens de malformação do desenvolvimento cortical, *Journal of Epilepsy and Clinical Neurophysiology* **12**: 149 – 154.
- MeshLab (2014). Meshlab. <http://meshlab.sourceforge.net/>. Acessado em Agosto de 2014.
- OpenGL (2014). Opengl. [https://www.opengl.org/wiki/Depth\\_Buffer\\_Precision](https://www.opengl.org/wiki/Depth_Buffer_Precision). Acessado em Novembro de 2014.
- OpenMesh (2014). Computer graphics group, rwth aachen. <http://openmesh.org/>. Acessado em Agosto de 2014.
- Pham, B. (1992). Offset curves and surfaces: a brief survey, *Computer-Aided Design* **24**(4): 223–229.
- Saeed, S., de Pennington, a. & Dodsworth, J. (1988). Offsetting in geometric modelling, *Computer-Aided Design* **20**(2): 67–74.
- Shostko, A. A., Hner, R. L. O. & Sandberg, W. C. (1999). Surface Triangulation Over, **1376**(June 1998): 1359–1376.
- Stroud, I. (2006). *Boundary Representation Modelling Techniques*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Turk, G. & Levoy, M. (1994). Zippered polygon meshes from range images, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '94*, ACM, New York, NY, USA, pp. 311–318.
- VMTK (2014). Visual manipulation tool for 3d mr neuro-images. <http://www.dca.fee.unicamp.br/projects/mtk/wu/vmtk2.html>. Acessado em Agosto de 2014.
- Ware, C. (2004). *Information Visualization: Perception for Design*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Wu, S.-T., Yasuda, C. & Cendes, F. (2012). Interactive curvilinear reformatting in native space, *Visualization and Computer Graphics, IEEE Transactions on* **18**(2): 299–308.
- Yi, I. L., Lee, Y.-s. & Shin, H. (2008). Mitered Offset of a Mesh Using QEM and Vertex Split, *Proceedings of the 2008 ACM symposium on Solid and physical modeling* pp. 315–320.