UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Reconstrução de Superfícies 3D a Partir de Nuvens de Pontos Usando Redes Neurais Auto-Organizáveis

JOÃO FERNANDO MARI

São Carlos – São Paulo
Maio / 2007

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

M332rs

Mari, João Fernando.
Reconstrução de superfícies 3D a partir de nuvens de pontos usando redes neurais auto-organizáveis / João Fernando Mari. -- São Carlos : UFSCar, 2008.
111 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Computação gráfica. 2. Aquisição 3D. 3. Reconstrução (geometria e modelagem computacional). 4. Redes neurais. I. Título.

CDD: 006.6 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

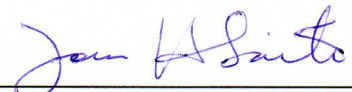
Programa de Pós-Graduação em Ciência da Computação

"Reconstrução de Superfícies 3D a partir de Nuvens de Pontos usando Redes Neurais Auto-Organizáveis"

JOÃO FERNANDO MARI

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:



Prof. Dr. José Hiroki Saito
(Orientador – DC/UFSCar)

Profa. Dra. Regina Borges de Araujo
(DC/UFSCar)

Prof. Dr. João Batista Destro Filho
(UFU)

São Carlos
Maio/2007

Dedico aos meus pais

João Mari e Maria de Fátima Geraldo Mari

AGRADECIMENTOS

A *Deus* pela vida.

Aos meus pais *João Mari* e *Maria de Fátima Geraldo Mari*, pelo amor e dedicação incondicionais, por tudo o que fizeram e fazem por mim. Devo tudo a eles.

Ao orientador *Prof. Dr. José Hiroki Saito* pelos ensinamentos, pela amizade e pela sabedoria com que conduziu a orientação deste trabalho.

A tia *Sônia Desiderato* e a tia *Madalena Desiderato* pela valorosa ajuda sem a qual provavelmente não teria chegado até aqui.

Ao meu tio *Rubens de Andrade* e tia *Odila Mari de Andrade* por ter me ajudado durante minha trajetória.

Ao tio *José Maris* e tia *Mariângela Maris* pelo apoio. Dedico também a todos os meus familiares.

Aos grandes amigos que fiz no GAPIS: *Alexandre L. M. Levada, Ana Luísa D. Martins, Cristiane, Débora C. Corrêa, Denis H. P. Salvadeo, Fabrício Breve, Gabriel M. Alves, Gustavo Poli, Leonardo Botega, Luís Castanheira, Mairum C. Andrade, Marcelo R. Zorzan, Michele Horta, Moacir Ponti Jr.* e *Talita P. Leite.* Agradeço também aos grandes amigos que fiz no DC: *Leandro, Ricardo* e *Edmilson*.

Aos amigos de longa data pelos momentos de diversão e descontração: *Alexandre, Fernando, Luciano, Luciano Japonês, Mauricio, Oscar, Paulo Sergio, Rodolfo* e *Thiago*.

Ao amigo *Gustavo Poli* pela amizade, pelas valiosas discussões e pela divertida convivência no laboratório mais legal da Federal.

Ao amigo *Marcelo R. Zorzan*, que considero um irmão de verdade, pela companhia nas madrugadas e finais de semana de trabalho no laboratório, pelos chopes no Trem Bom e pelos treinos de *jiu-jitsu*.

Ao *Prof. Dr. Marcos Luis Mucheroni* por ter me apresentado ao mundo da pesquisa durante a minha graduação.

A *Roberta Ribeiro*, pelo amor, pela companhia, pelo carinho e compreensão.

Aos grandes amigos de república *Laura Bussab* e *Vinicius*, pela feliz convivência.

A todos os professores do DC-UFSCar pelos valiosos ensinamentos.

A todos os funcionários do DC-UFSCar. As secretarias do PPG-CC *Cristina* e *Mirian*.

A todos que colaboraram de alguma forma, seja direta ou indiretamente, com o desenvolvimento deste trabalho.

Agradeço a CAPES pelo apoio financeiro.

"To doubt everything or to believe everything are two equally convenient solutions; both dispense with the necessity of reflection."

Henri Poincaré

RESUMO

O trabalho consiste na implementação de um algoritmo de reconstrução de superfícies 3D a partir de nuvens de pontos, usando Redes Neurais Auto-Organizáveis. Dentre as Redes Neurais Artificiais destaca-se o SOM (*Self-Organizing Map*) que é um modelo auto-organizável baseado na aprendizagem competitiva. O sistema proposto foi baseado no modelo de rede neural auto-organizável GCS (Growing Cell Structures), que é uma variação incremental do SOM, com alterações na forma de atualização das posições dos vizinhos diretos do nodo vencedor e também na forma como são inseridos novos nodos e removidos nodos inativos. O sistema foi aperfeiçoado pela inclusão da operação de troca de arestas, que melhorou a qualidade da malha gerada e a convergência do algoritmo. O algoritmo foi então denominado GCS-M (GCS-Modificado). Para avaliar os resultados obtidos por meio da implementação do sistema proposto foram utilizadas métricas para mensurar a qualidade da malha baseada no valor das medidas das distâncias mínimas entre a nuvem de pontos e os elementos que compõem a malha poligonal. Com esse mecanismo foram realizadas comparações dos resultados obtidos pelo GCS-M com os resultados obtidos pelos métodos tradicionais de reconstrução de superfícies. O algoritmo proposto se mostrou bastante eficiente, obtendo superfícies reconstruídas realisticamente, com medidas de erro baixas.

ABSTRACT

This work consists of the implementation of a 3D surface reconstruction algorithm from point clouds, using Self-Organizing Neural Networks. Among the Neural Networks the SOM (Self-Organizing Map) is distinguished, since it is a self-organized model based on competitive learning. The proposed system is based on the GCC (Growing Cell Structures), a self-organizing neural network that is a SOM incremental variation with some alteration in the way that the positions of the direct neighbors of the winner node are updated, and also in the way that new nodes are inserted and inactive nodes are removed. The system has been improved by the inclusion of the edge-swap operation, improving the quality of the generated mesh and the convergence of the algorithm. The algorithm than has been called GCS-M (GCS-Modified). To evaluate the results obtained by the implementation of the proposed system had been used metrics to evaluate the quality of the mesh, based on the values of the minimum distances between the point cloud and the elements that compose the polygon mesh. With this comparison mechanism, it had been performed comparisons of the GCS-M algorithm results and the traditional methods results of surface reconstruction. The obtained results show that the proposed algorithm is very efficient, obtaining realistically reconstructed surfaces with low error measures.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com os recentes avanços das tecnologias de aquisição volumétrica (*scanners* 3D), tem aumentado a necessidade de algoritmos para reconstrução de superfícies a partir das nuvens de pontos geradas por esses dispositivos. Informalmente, reconstrução de superfícies pode ser definida como um processo que gera uma malha que represente corretamente a forma e a topologia do objeto original, de onde é amostrado um conjunto de pontos. Obter a forma e a topologia de um objeto, partindo apenas de pontos desconexos não é uma tarefa trivial.

Desenvolver, testar e expandir métodos que sejam robustos e eficazes para este propósito é um importante campo de pesquisa na atualidade, mesmo que seja direcionado para um tipo específico de dispositivo de aquisição e/ou objeto alvo, visto que os diversos métodos disponíveis não suprem as necessidades impostas pelas diversas aplicações onde reconstrução de superfícies é uma etapa chave.

Reconstrução de superfícies é um problema que surge em inúmeras aplicações cientificas e de engenharia, como por exemplo: sistemas de aquisição volumétrica (scanner 3D) (FRANÇA et al., 2005), engenharia reversa e geração automática de modelos CAD a partir de modelos físicos (SON, PARK e LEE, 2002; XI e SHU, 1999), imagens médicas, visualização de dados científicos, cartografia, simulações numéricas de equações diferenciais parciais (GOIS, 2004; MALISKA, 1995).

Geralmente quando falamos de reconstrução de superfícies em imagens médicas, nos referimos aos aparelhos de *ressonância magnética* e *tomografia computadorizad*a. Estes aparelhos geram um conjunto de imagens de seções planares paralelas do interior do objeto. Ao extrair os contornos destas imagens, tem-se uma pilha de

contornos bidimensionais paralelos (Figura 1), a partir destes contornos é necessário reconstruir a superfície ao redor do objeto, mantendo a estrutura original. Alguns trabalhos envolvendo este tipo de dados são relatados por VARGAS et al. (2002); NONATO et al., (2001); BOISSONNAT (1988); MEYERS, SKINNER e SLOAN (1992).

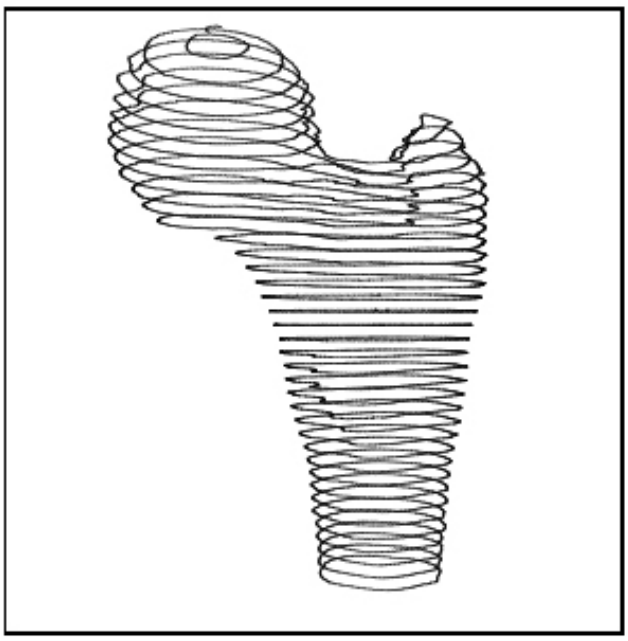


Figura 1: Pilha de seções planares obtidas por Tomografia Computadorizada. Fonte: (HOPPE, 1994)

Os sistemas de aquisição volumétricas (scanners 3D) (FRANÇA et al., 2005), merecem uma atenção especial neste trabalho, e são tratados no capítulo 2.

HOPPE et al., (1992) define formalmente o problema da reconstrução de superfícies da seguinte forma: Dado um conjunto de amostras de pontos *X* próximo ou sobre uma superfície desconhecida *U*, criar uma superfície *S* que se aproxima de *U*.

Algumas considerações devem ser feitas quanto ao conjunto de amostras de pontos *X*, e da superfície desconhecida *U* (HOPPE, 1994):

- *X* pode conter ruído;
- *X* pode não estar uniformemente amostrado;
- *X* pode não ser suficientemente denso e
- *U* pode possuir um tipo topológico arbitrário (suponha que *U* seja uma *variedade*).

Uma variedade topológica[1] ou simplesmente variedade, é um espaço topológico que é localmente Euclidiano (i.e., ao redor de qualquer ponto, existe uma vizinhança topologicamente equivalente a uma bola unitária aberta em $\mathbb{R}^n$). Informalmente, qualquer objeto que pode ser "planificado" é uma variedade (ROWLAND, 2006). Uma curva é uma variedade de dimensão 1, e uma superfície é uma variedade de dimensão 2.

Estas considerações são muito importantes quando se trata de dados reais, e servirão como parâmetros de avaliação para os métodos de reconstrução abordados neste trabalho.

1.1 Motivação

O problema da reconstrução de superfícies a partir de um conjunto de pontos desconexos já foi abordado de inúmeras maneiras, incluindo funções implícitas (HOPPE, 1992) e geometria computacional (EDELSBRUNNER e MÜCKE, 1994; AMENTA, BERN e KAMVYSSELIS, 1998). Uma revisão dessas abordagens pode ser encontrada em (GOIS, 2004). Tais métodos são eficientes, desde que uma série de exigências quanto às características do conjunto de pontos sejam satisfeitas, por exemplo, baixo nível de ruído, conjunto de pontos denso, e amostragem uniforme. Quando aplicados em nuvens de pontos geradas artificialmente, em ambientes controlados ou virtuais, geralmente estes métodos tem se mostrado eficientes, mas quando aplicados a problemas reais, onde diversos fatores interferem no processo de aquisição, muitas vezes é necessário que o conjunto de pontos amostrado passe por um estágio de pré-processamento, para garantir o sucesso da reconstrução.

---

[1] *Manifold* em inglês.

A busca por métodos alternativos que contornem estes problemas tem tomado vários caminhos. Uma abordagem que vem se mostrando bastante robusta e eficiente é a utilização de redes neurais, principalmente os modelos auto-organizáveis, baseados em aprendizagem competitiva. As abordagens usando redes neurais diferem das demais abordagens por não serem determinísticas, e sim baseadas em um sistema de aprendizagem estocástico, ou seja, caso o processo de reconstrução não funcione na primeira tentativa, é possível que, aplicando o algoritmo novamente tenha sucesso.

Outra característica muito interessante nos algoritmos baseados em redes neurais é que a resolução final da superfície gerada (malha de polígonos) não depende do número de pontos presentes na nuvem de pontos, como é na maioria dos algoritmos não baseados em redes neurais, mas podem ser definidos pelo usuário. Dessa forma o tempo necessário para finalizar a execução do algoritmo independe do tamanho da nuvem de pontos, e sim pela resolução final da malha de polígonos, definida previamente pelo usuário.

Este trabalho propõe e implementa um algoritmo para reconstrução de superfícies a partir de nuvens de pontos. O algoritmo proposto é baseado em um modelo de rede neural auto-organizável incremental GCS, que apresenta semelhanças com o algoritmo Neural Meshes também desenvolvido para reconstrução de superfícies. A principal diferença entre o algoritmo proposto neste trabalho, denominado GCS-Modificado (GCS-M), e o GCS e Neural Meshes é a inclusão de uma operação de troca de arestas, apresentada na Seção 7.6, que tornou o algoritmo mais robusto e menos sensível a variação dos parâmetros de taxa de aprendizado, como pode ser observado no Capítulo 8.

1.2 Organização da Dissertação

O texto presente nesta dissertação está organizado conforme é descrito abaixo:

O capítulo 2 apresenta uma introdução das técnicas de aquisição volumétrica, dando atenção especial aos métodos baseados em triangulação a laser. São feitas considerações relativas à representação de objetos tridimensionais em ambientes computacionais, assim como características gerais envolvendo o processo de reconstrução.

No capítulo 3 são descritos os principais métodos tradicionais de reconstrução de superfícies. Os métodos tratados neste capítulo são denominados tradicionais, ou clássicos, por não utilizarem princípios de redes neurais.

O capítulo 4 apresenta uma descrição detalhada dos modelos de redes neurais empregados nos experimentos de reconstrução de superfícies. São eles, o *Self-Organizing Map* (SOM), o *Growing Cell Structures* (GCS), o *Growing Neural Gas* (GNG), e o *Growing Grid* (GG).

O capítulo 5 traz uma revisão dos principais trabalhos envolvendo a utilização de redes neurais na resolução do problema de reconstrução de superfícies.

O capítulo 6 apresenta a proposta de trabalho, o algoritmo GCS-M, especifico para o problema de reconstrução de superfícies baseado no GCS (seção 4.3).

A metodologia utilizada e os detalhes da implementação são apresentados no capítulo 7.

O capítulo 8 traz os resultados e os mesmos são discutidos e comparados com outros algoritmos (*Power Crust e Tight Cocone*).

A conclusão e trabalhos futuros são mostrados no capítulo 9.

Em seguida são apresentadas as referências bibliográficas.

2 SISTEMAS DE AQUISIÇÃO VOLUMÉTRICA

Os sistemas de aquisição volumétrica são constituídos, basicamente, de três fases: sensoriamento, processamento de imagens, e reconstrução de superfícies (FRANÇA, 2004).

- **Sensoriamento:** é a fase onde a geometria da superfície da cena ou objeto em questão é capturada, resultando em um conjunto de dados numéricos brutos.
- **Processamento de imagens:** tem a função de processar os dados numéricos brutos obtidos durante o sensoriamento, para que, a partir dela, possam ser extraídas as informações necessárias para a fase de reconstrução de superfícies. Essas informações são muitas vezes denominadas nuvens de pontos.
- **Reconstrução de superfícies:** a partir das informações obtidas na fase de processamento de imagens, gera uma representação 3D do objeto.

Os sistemas de aquisição 3D mais eficientes são baseados no princípio da triangulação a laser ou no princípio de tempo de vôo (TOF). O princípio de tempo de vôo consiste em medir o intervalo de tempo entre a emissão e a recepção de um pulso de laser (FRANÇA, 2004; SINHA e JAIN, 1994). A profundidade *r* é então calculada pela Equação (2.1):

$$r = \frac{ct}{2} \tag{2.1}$$

onde *c* é a velocidade da luz e *t* é o tempo entre a emissão e a recepção do pulso do laser.

Sistemas de aquisição 3D possuem uma grande gama de aplicações, que tem crescido com o aumento da precisão desses sistemas.

Entre as principais aplicações pode-se citar (FRANÇA, 2004): modelagem e animação 3D, efeitos especiais e cinema, indústria de desenvolvimentos de jogos, simulação de postura e mobilidade de animais extintos via fósseis, mapeamento topológico de lugares remotos (MILLER, AMIDI e DELOUIS, 1999), mapeamento de cavernas, inspeção para detecção de falhas em estruturas de construções e peças, aplicações espaciais, prototipagem rápida e engenharia reversa, museus virtuais (LEVOY et al., 2000), e determinação do índice de massa corpórea (FRANÇA et al., 2004).

2.1 Sensores de profundidade

Os tipos de sensores podem ser classificados como ativos e passivos. Sensores ativos são os que fornecem e controlam a sua própria iluminação. Sensores passivos apenas absorvem as radiações do ambiente, e a partir delas busca extrair informações de profundidade da cena (SINHA e JAIN, 1994).

A próxima seção discute um tipo de sensor ativo do tipo baseado em triangulação de luz estruturada, utilizado no desenvolvimento do scanner 3D que fez a geração da maioria das nuvens de pontos utilizadas nos experimentos preliminares deste trabalho. A Figura 2 mostra uma taxonomia dos principais tipos de dispositivos de sensoriamento. FRANÇA (2004) e SINHA e JAIN (1994) trazem mais informações sobre cada um deles.

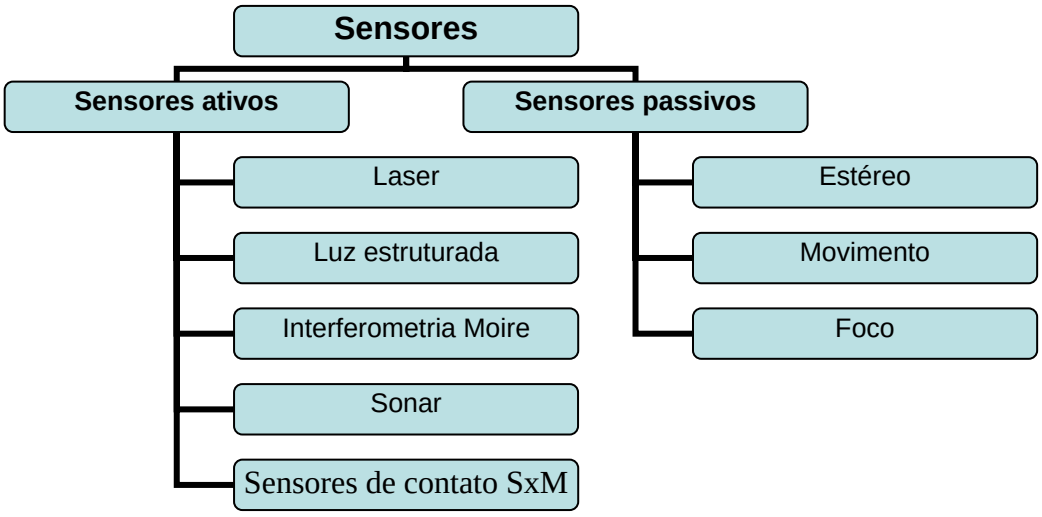


Figura 2: Taxonomia dos principais tipos de sensores utilizados para aquisição de informações tridimensionais.

2.2 Sensor ativo baseado em triangulação a laser

FRANÇA (2004) implementou um protótipo de scanner 3D baseado no método de triangulação a laser (SINHA e JAIN, 1994) e campo de visão variável. O modelo possui tamanho reduzido, é composto por um conjunto de câmera e laser, ideal para pequenos objetos (Figura 3). Um diagrama de blocos do sistema proposto é mostrado na Figura 4, e descrito a seguir (FRANÇA et al., 2005; FRANÇA, 2004).

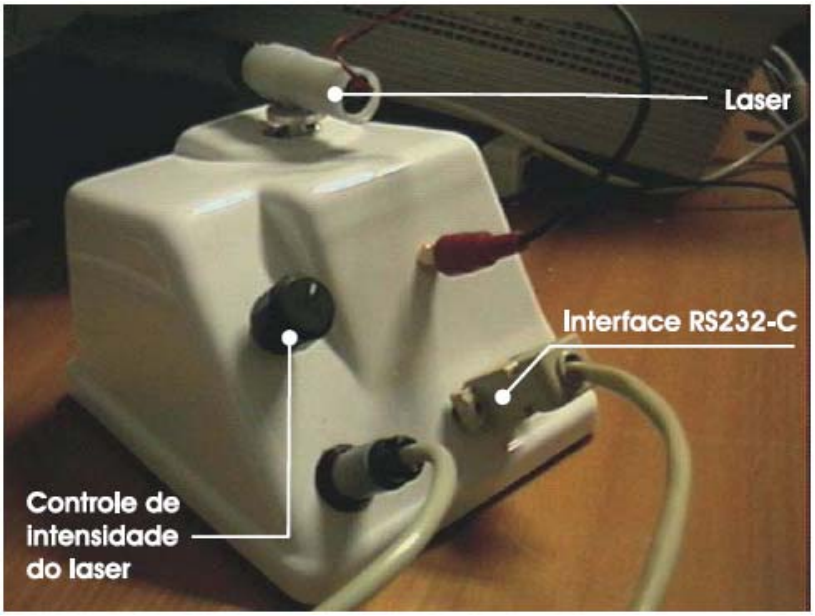


Figura 3: Fotografia do sistema de varredura implementado por FRANÇA (2004).
Fonte: (FRANÇA, 2004)

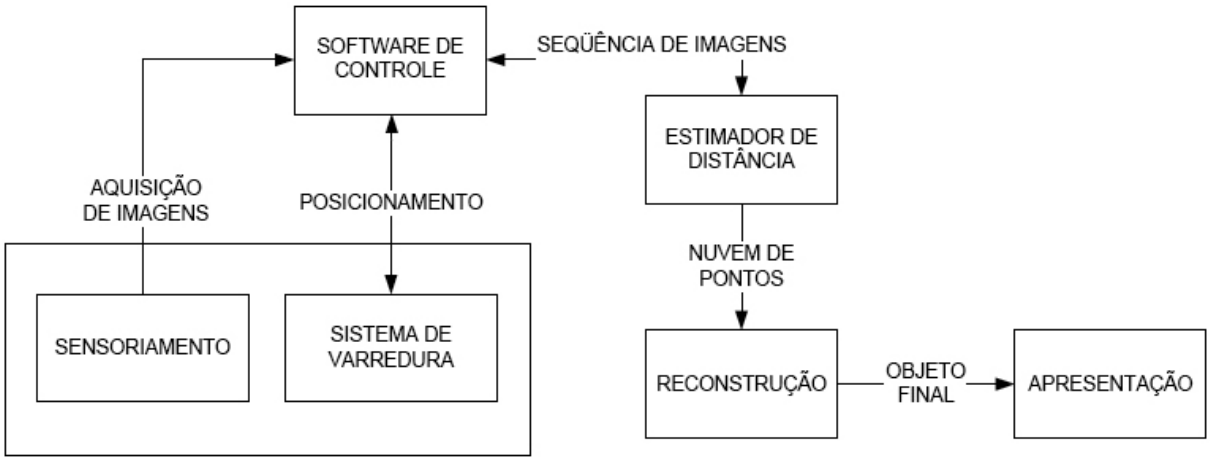


Figura 4: Estrutura básica do sistema de aquisição 3D implementado por FRANÇA (2004).
Fonte: (FRANÇA, 2004).

a) Sensoriamento: É responsável pela aquisição das imagens de entrada. É composto por uma câmera de vídeo, e uma placa de aquisição de vídeo com resolução de 640x480 pixels.

b) Sistema de varredura: É composto por um hardware controlador de movimento, um motor com *encoder* e redutor mecânico, e um laser com gerador de linha. O

controlador de movimento comanda o posicionamento, a velocidade, a aceleração do motor, e o acionamento do laser pela interface RS232-C.

c) Software de controle (3D): É uma rotina que recebe como entrada uma seqüência de quadros de vídeo, e os submete a um processo de correção de distorções causadas por parâmetros intrínsecos da câmera.

d) Estimador de distância: Este módulo utiliza métodos de calibração explícitos para o mapeamento de coordenadas 3D na imagem e uma abordagem implícita para correção da distorção da mesma. Na seqüência é aplicado um filtro de mediana para atenuação do ruído, detecção de bordas e limiarização. Baseado na técnica de triangulação ativa é possível determinar a profundidade a partir das imagens segmentadas. A saída do modulo é uma nuvem de pontos.

e) Reconstrução (3D): Este módulo recebe a nuvem de pontos com entrada, e é o responsável pelo alinhamento e fusão das diversas nuvens geradas. Também é responsável por gerar uma superfície poligonal 3D que represente perfeitamente o objeto alvo.

f) Apresentação: A interface gráfica permite que o usuário determine os limites de varredura, posicionando o laser nesses limites, escolha o modo de aquisição, determine a resolução de varredura horizontal, faça a segmentação de imagens, determine o limiar de profundidade, faça a estimação de distância, e salve o objeto 3D.

GAZZIRO (2005) implementou uma versão antropométrica do scanner 3D. Este modelo é composto por quatro conjuntos câmera/laser, cada um montado em um trilho vertical (Figura 5). Os conjuntos são movimentados ao longo dos trilhos por um motor de precisão, para obter amostras do corpo por inteiro. A principal aplicação deste modelo de scanner é a medição indireta do índice de massa corpórea de indivíduos.

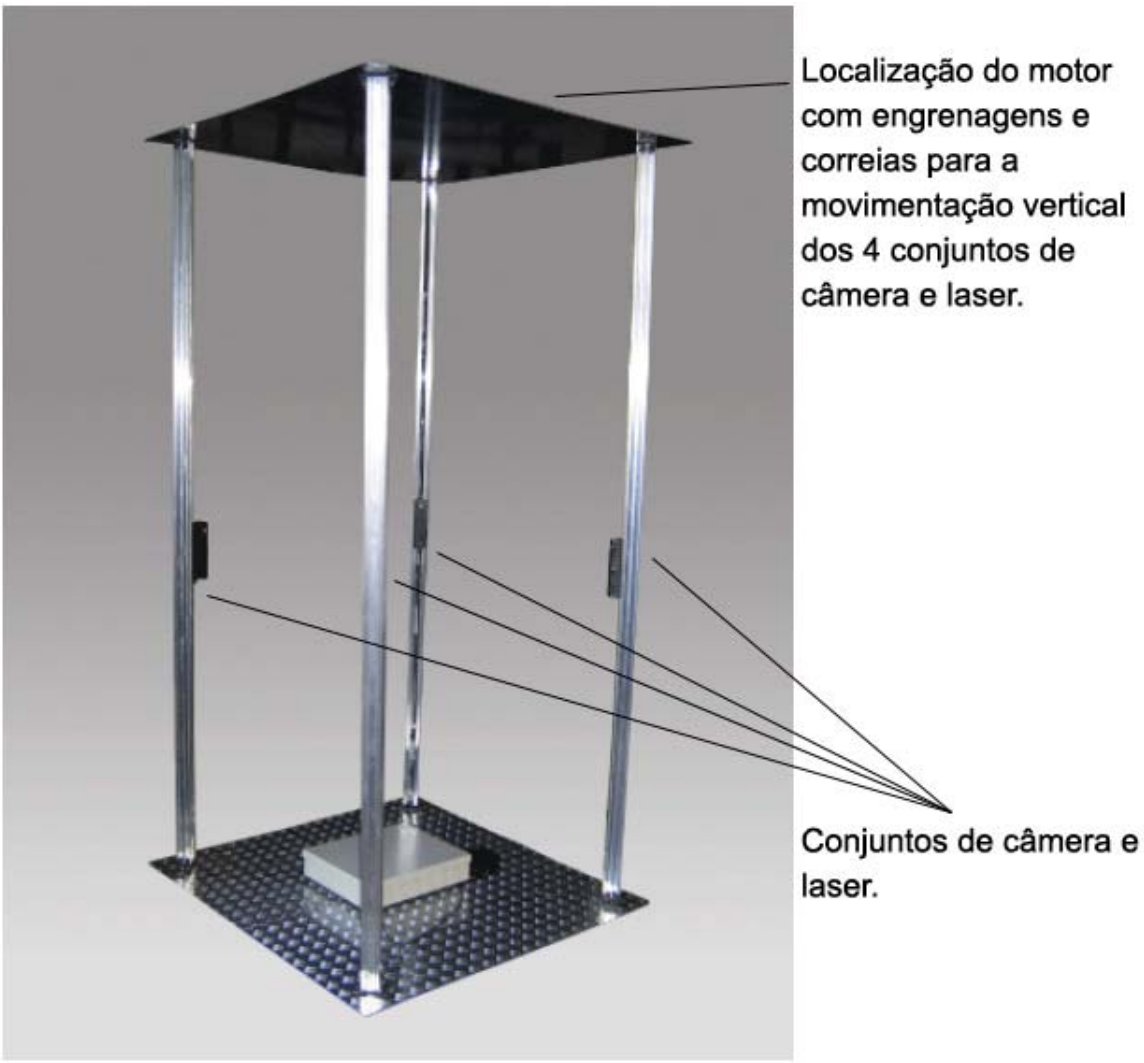


Figura 5: Scanner antropométrico 3D, implementado por GAZZIRO (2005).
Fonte: (GAZZIRO, 2005).

O método de triangulação a laser, consiste na projeção de um padrão de luz, por exemplo, uma linha vertical (ou horizontal) projetada por um laser, sobre o objeto. A projeção é capturada por uma câmera, e a distância até o objeto é calculada geometricamente, dado o conhecimento *a priori* das posições do laser e da câmera (SINHA e JAIN, 1994). A Figura 6 ilustra a configuração de um sistema de triangulação a laser.

O centro das lentes está localizado na origem, com distância focal *f* até o plano da imagem, e a distância entre o projetor e a câmera representada pela linha de base *b*. A projeção do laser forma um ângulo θ com a linha de base *b* (TIZIANI, 1997). O ponto 3D no espaço real é projetado no pixel da imagem (x', y').

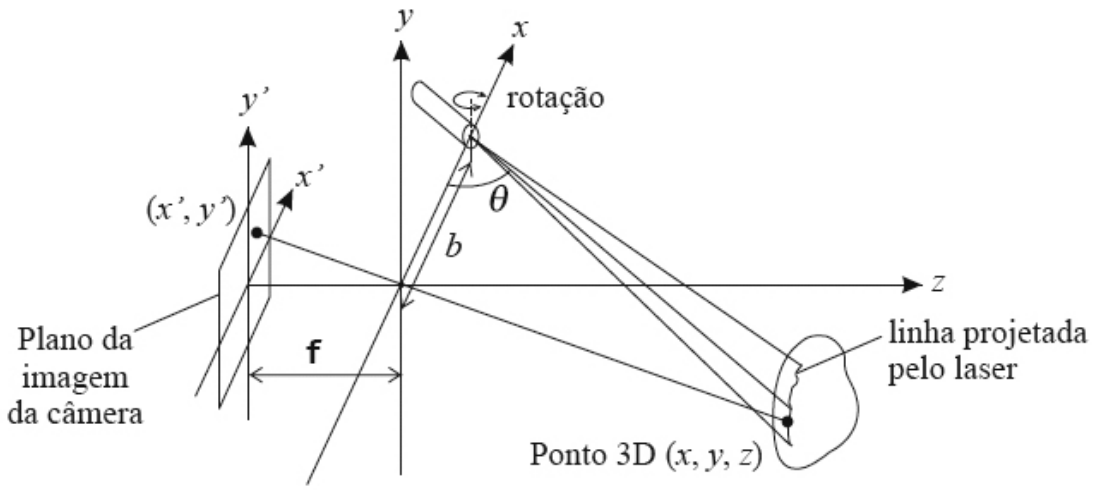


Figura 6: Configuração básica de um sistema de triangulação a laser. Fonte: (FRANÇA, 2004).

O valor de θ é conhecido pelo sistema de varredura, e a distância focal *f* é determinada pela calibração da câmera. Calcula-se a linha de base *b*, posicionando o laser de maneira que sua projeção incida no centro da imagem, apontando para um objeto a uma distancia conhecida *z* (Equação (2.2)):

$$b = -\frac{z}{\tan\theta} \tag{2.2}$$

Na seqüência, é realizada uma varredura girando-se o laser que projeta uma linha vertical no objeto alvo. As imagens são capturadas e os valores de θ para cada imagem são armazenados. Para cada imagem gerada, é necessário determinar os valores de x' e y' onde o laser é refletido, então, as coordenadas do ponto no mundo real são estimadas pelas Equações (2.3), (2.4) e (2.5) (FRANÇA, 2004; SINHA e JAIN, 1994).

$$x = \frac{bx'}{f\cot\theta - x'} \tag{2.3}$$

$$y = \frac{by'}{f\cot\theta - x'} \tag{2.4}$$

$$z = \frac{bf}{f\cot\theta - x'} \tag{2.5}$$

No modelo proposto por FRANÇA (2004) a varredura é executada pelo laser e também pela câmera, o que difere dos sistemas tradicionais baseados em triangulação, onde a câmera é mantida fixa e a varredura é executada pelo laser.

2.3 Representação de objetos tridimensionais

Esquemas de representação para objetos sólidos podem ser divididos em duas grandes categorias: representação de borda (B-rep)[2], e representação por particionamento do espaço.

Representação de borda (B-rep): descreve um objeto tridimensional como um conjunto de superfícies que separa o interior do objeto do ambiente.

Representação por particionamento do espaço: é usada para descrever propriedades do interior dos objetos.

Neste trabalho nos interessa a categoria **B-rep**. A representação de borda mais comum para um objeto gráfico tridimensional é um conjunto de polígonos de superfície que envolve o interior de um objeto. Muitos sistemas gráficos armazenam todas as descrições do objeto como um conjunto de polígonos de superfície. Isto simplifica e acelera a renderização e exibição dos objetos, pois todas as superfícies são descritas por equações lineares (HEARN e BAKER, 1997). A Figura 7 ilustra uma representação poligonal de um cilindro.

Uma organização conveniente para armazenar dados geométricos é criar três tabelas: uma tabela de vértices, uma tabela de arestas, e uma tabela de polígonos. A tabela de vértices armazena os valores das coordenadas de cada vértice do objeto. A tabela de arestas

[2] Do inglês Boundary-REPresentation.

contém ponteiros para a tabela de vértices, de forma a identificar os vértices de cada aresta. A tabela de polígonos contém ponteiros para a tabela de arestas, identificando as arestas de cada polígono (HEARN e BAKER, 1997). A Figura 8 ilustra este esquema para dois polígonos adjacentes de uma superfície poligonal.

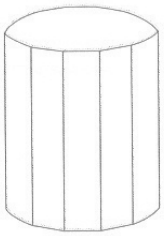


Figura 7: Superfície de um cilindro representado como uma malha de polígonos (*wireframe*). Fonte: (HEARN e BAKER, 1997).

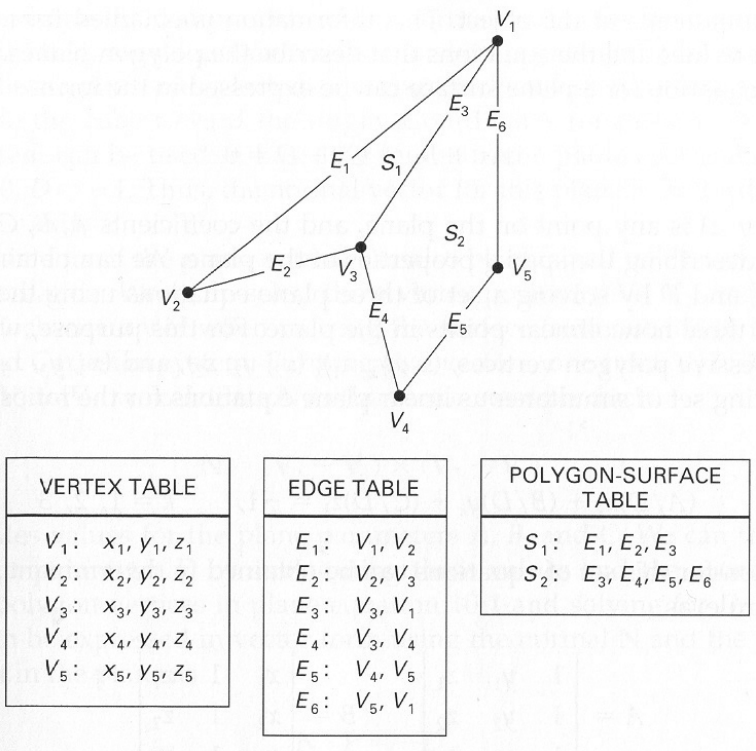


Figura 8: Representação da tabela de dados geométricos para duas superfícies de polígonos adjacentes, formado por seis arestas e cinco vértices. Fonte: (HEARN e BAKER, 1997).

É possível ainda incluir informações adicionais às tabelas mostradas na Figura 9 para prover operações rápidas de extração de informações. Por exemplo, pode-se expandir a tabela de arestas incluindo ponteiros para os polígonos que estas arestas fazem parte, assim

arestas comuns entre polígonos podem ser identificadas rapidamente (HEARN e BAKER, 1997) (Figura 9).

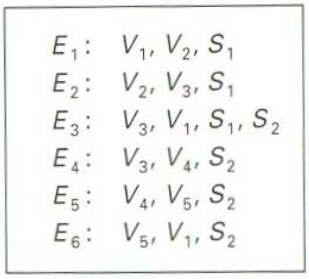


Figura 9: Tabela de vértices expandida pela inclusão de ponteiros para a tabela de polígonos. Fonte: (HEARN e BAKER, 1997).

2.4 Reconstrução

Grande parte dos dispositivos de aquisição volumétrica, incluindo scanner 3D, geram imagens de profundidade, ou nuvens de pontos. Uma imagem de profundidade é uma matriz 2D onde os valores de cada pixel representam a distância entre o objeto alvo e o sensor. Uma nuvem de pontos é um conjunto de pontos, representados por suas coordenadas no espaço $\mathbb{R}^3$, sem informação alguma quanto à conectividade entre eles.

A principal vantagem dos algoritmos que trabalham com pontos desconexos, é justamente o fato deles não fazerem nenhuma suposição quanto à conectividade entre os pontos (CURLESS e LEVOY, 1996).

Os scanners 3D baseados em laser, ou luz estruturada, só podem fazer a aquisição dos dados onde o sensor “toca” o alvo. Isso implica que reconstruir objetos volumétricos, geralmente envolve várias aquisições feitas em diferentes pontos de vista (FRANÇA, 2004). Dessa forma, um grande número de imagens de profundidade ou de nuvens de pontos é criado, e é necessário que elas sejam corretamente fundidas em uma única nuvem de pontos ou, depois de reconstruídas separadamente, em uma única malha poligonal

(Seção 2.3) (CURLESS e LEVOY (1996) propõem uma técnica para fusão de diversas imagens de profundidade).

O trabalho de dissertação de mestrado proposto tem por objetivo tratar apenas da reconstrução de superfícies a partir de nuvens de pontos, sendo a manipulação e reconstrução de superfícies a partir de *imagens de profundidade* fora do escopo do trabalho. As nuvens de pontos consideradas neste trabalho já deverão estar fundidas em uma representação única. Maiores detalhes são encontrados nas referências (CURLESS, 1997; CURLESS e LEVOY, 1996).

O capítulo 3 discute alguns dos principais métodos de reconstrução de superfícies, mas apenas os que não utilizam princípios de redes neurais, denominados neste trabalho como métodos tradicionais, ou clássicos. Os métodos de reconstrução baseados em redes neurais são tratados no capítulo 5.

2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma introdução aos sistemas de aquisição volumétrica, em especial os baseados no método de triangulação a laser, onde as informações das nuvens de pontos são obtidas. Uma introdução sobre a representação de objetos tridimensionais no computador é apresentada. Algumas considerações gerais sobre reconstrução de superfícies 3D a partir de nuvens de pontos são também apresentadas.

O próximo capítulo apresenta métodos de reconstrução de superfície que não utilizam princípios de redes neurais (aqui denominados métodos clássicos ou tradicionais).

3 RECONSTRUÇÃO DE SUPERFÍCIES POR MÉTODOS TRADICIONAIS

Os métodos de reconstrução de superfícies descritos neste capítulo adotam abordagens convencionais (sem o uso de redes neurais, aqui denominados métodos tradicionais, ou clássicos). Diversos métodos foram propostos na literatura para resolver o problema da reconstrução de uma superfície a partir de um conjunto de pontos desconexos. Quando utilizados com nuvens de pontos que apresentam algumas características especiais, como dados amostrados densos e uniformes, baixo nível de ruído, todos funcionam bem. Geralmente quando se trabalha com dados reais, estas características são difíceis de serem alcançadas.

Neste capítulo é mostrado o avanço destas técnicas em produzir garantias teóricas e práticas de reconstrução. Os métodos mais recentes possuem características que permitem contornar algumas dificuldades quanto às características das amostragens.

3.1 Métodos baseados em função implícita

HOPPE et al., (1992) propôs um algoritmo para reconstrução de superfícies composto por duas etapas.

Na primeira etapa é definida uma função $f : D \to \mathbb{R}$, onde $D \subset \mathbb{R}^3$ é uma região próxima aos dados, em que f estima uma distância geométrica com sinal para a superfície desconhecida *M*. O conjunto zero $Z(f)$ é a estimativa para *M*.

A segunda etapa utiliza um algoritmo para aproximar $Z(f)$ por um complexo simplicial. Sendo um simplexo de dimensão *k* o fecho convexo de $k+1$ pontos em $\mathbb{R}^3$ (Figura 10), um complexo simplicial é um conjunto finito de simplexos, que se cruzam apenas por suas faces comuns.

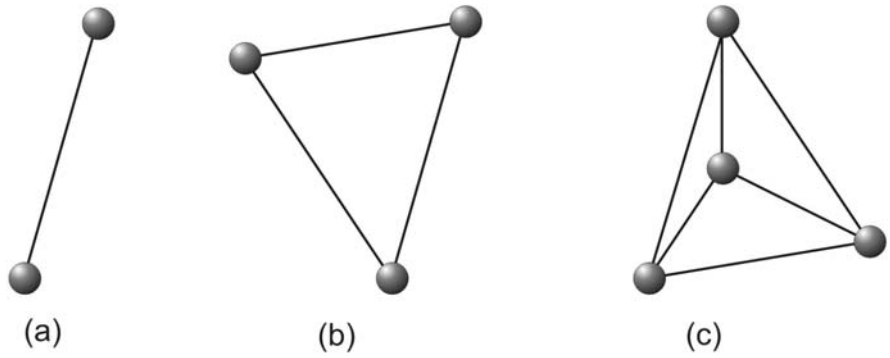


Figura 10: Simplexos. (a) Simplexo unidimensional (segmento de reta). (b) Simplexo bidimensional (triângulo). (c) Simplexo tridimensional (tetraedro).

Considera-se o plano tangente $PT(\mathbf{x}_i)$ associado com o ponto $\mathbf{x}_i$ seja representado por um ponto $\mathbf{o}_i$, chamado centro, junto com o vetor normal unitário $\hat{\mathbf{n}}_i$, a distância com sinal entre um ponto arbitrário $p \in \mathbb{R}^3$ até $PT(\mathbf{x}_i)$ é definida pela Equação (3.1):

$$dist_i(\mathbf{p}) = (\mathbf{p} - \mathbf{o}_i).\hat{\mathbf{n}}_i. \qquad (3.1)$$

O centro e a normal de $PT(\mathbf{x}_i)$ são determinados pelos *k* pontos de *X* mais próximos de $\mathbf{x}_i$. O cálculo do vetor normal $\hat{\mathbf{n}}_i$ é feito pela matriz de covariância simétrica e semi-definida positiva dada por (Equação (3.2)):

$$CV = \sum_{\mathbf{y} \in N_{\mathbf{x}_i}} (\mathbf{y} - \mathbf{o}_i) \otimes (\mathbf{y} - \mathbf{o}_i), \qquad (3.2)$$

onde $\otimes$ denota o operador vetorial do produto externo. Se $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$ são os autovalores de CV associados com os autovetores $\hat{\mathbf{v}}_i^1, \hat{\mathbf{v}}_i^2, \hat{\mathbf{v}}_i^3$, respectivamente, escolhe-se $\hat{\mathbf{n}}_i$ como sendo $\hat{\mathbf{v}}_i^3$

ou $-\hat{\mathbf{v}}_i^3$. A seleção determina a orientação do plano tangente, e deve ser feita de modo que os planos próximos sejam “consistentemente orientados”.

O método proposto por Hoppe é bastante extenso, e pode ser visto com detalhes em (HOPPE et al, 1992; HOPPE, 1994).

3.2 Alpha-shapes

O conceito de α*-shapes* é uma generalização do *fecho convexo* de um conjunto de pontos. Seja *X* um conjunto finito de pontos em $\mathbb{R}^3$, e α um número real no intervalo $0 \leq \alpha \leq \infty$. Quando $\alpha = \infty$, o α*-shape* é idêntico ao fecho convexo de *X*. Porém, à medida que α decresce, o α*-shape* se retrai pelo aparecimento gradual de cavidades. Essas cavidades podem se unir formando túneis, e até mesmo buracos podem aparecer (EDELSBRUNNER e MÜCKE, 1994). A Figura 11 ilustra a reconstrução de um conjunto de pontos pelo α*-shape*, para diversos valores α.

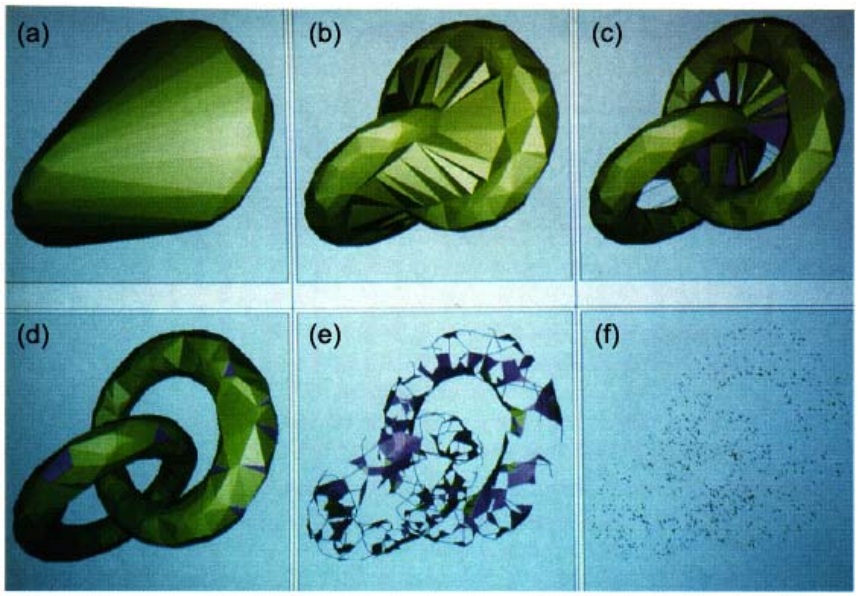


Figura 11: Reconstrução de uma nuvem de pontos por α-shapes. Na figura (a) α = + ∞, formando o fecho convexo do conjunto de pontos. Na figura (f) α = 0, assim restam apenas os pontos. As figuras (b), (c), (d) e (e) mostram valores intermediários do parâmetro α, sendo a figura (d) uma reconstrução obtida com um valor ideal para α. Fonte: (EDELSBRUNNER e MÜCKE, 1994)

3.3 Crust

O *Crust* foi proposto originalmente para duas dimensões (AMENTA, BERN e EPPSTEIN, 1998). A extensão do *Crust* 2D para o espaço tridimensional é apresentada em dois artigos (AMENTA e BERN, 1999) e (AMENTA, BERN e KAMVYSSELIS, 1998), e é denominado *raw crust*.

O *Raw Crust*, no entanto, apresenta garantias fracas de reconstrução, necessitando amostragens muito densas.

Para resolver esse problema são propostas duas etapas de pós-processamento, e o método passou a se chamar *Crust* 3D.

O algoritmo do *Raw Crust* é descrito a seguir (GOIS, 2004):

Para um conjunto de pontos X
Onde X é um subconjunto de uma superfície
para todo $\mathbf{x} \in X$ **faça**
    Calcular os pólos
fim
Calcule a triangulação de Delaunay de X unindo os pólos
Mantenha apenas os triângulos cujos vértices pertencem a X.

O algoritmo para detecção de pólos é mostrado a seguir (GOIS, 2004):

Considere o diagrama de Voronoi do conjunto de pontos X
para todo $\mathbf{x} \in X$ **faça**
    se **x** não pertence ao fecho convexo de X **então**
        seja $\mathbf{x}^+$ o vértice de Voronoi mais distante de **x**
    fim
    se x pertence ao fecho convexo de X **então**
        seja $\mathbf{x}^+$ um ponto no infinito fora do fecho convexo na direção $\mathbf{x}\mathbf{x}^+$ igual ao valor médio da direção normal das faces encontradas em **x**
    fim
    Entre todos os vértices de Voronoi de **x** tais que o ângulo $\mathbf{x}^+\mathbf{x}$v é maior que $\pi/2$, escolha o mais distante de **x** para ser o $\mathbf{x}^-$.
fim

No algoritmo apresentado foram considerados, o uso do diagrama de Voronoi e o calculo da triangulação de Delaunay cujas definições são as seguintes.

Diagrama de Voronoi: para um ponto $\mathbf{x} \in X$, a sua região de Voronoi é o conjunto de todos os pontos **p**, cuja distância Euclidiana é menor que ou igual a distância entre **p** e qualquer outro ponto de *X*. O conjunto de todas as regiões de Voronoi é chamado diagrama de Voronoi (EDELSBRUNNER e MÜCKE, 1994).

Triangulação de Delaunay: é o dual do diagrama de Voronoi. Para $0 \le k \le 3$, seja F_k o conjunto dos *k*-simplexos que pertencem ao fecho convexo de *T*, onde $T \subseteq X$, $|T| = k+1$, e sua *circunsfera*[3] não possui nenhum outro ponto de *X* em seu interior. O conjunto de todos os F_k é denominado triangulação de Delaunay (EDELSBRUNNER e MÜCKE, 1994).

Como foi visto, a garantia do *Raw Crust* é fraca, tornando necessária uma amostragem de pontos muito densa. Foram propostas duas etapas de pós-processamento para tornar mais eficiente a reconstrução (AMENTA e BERN, 1999):

Filtering by normal: Consiste em remover os triângulos cujo ângulo entre a normal e o vetor definido pelo pólo mais distante de um dos vértices do triângulo seja muito grande (maior que θ para o vértice de maior ângulo do triângulo, e maior que $3\theta/2$ para os outros vértices).

Trimming: Consiste em orientar os triângulos e pólos (dentro e fora) consistentemente, e extrair uma variedade linear por partes sem bordas.

---

[3] A *circunsfera* de um *k*-simplexo é a esfera que contém todos os vértices do *k*-simplexo (GOIS, 2004).

3.4 Power Crust

A idéia principal do *Power Crust* (AMENTA, CHOI e KOLLURI, 2001a; AMENTA, CHOI e KOLLURI, 2001b) é produzir uma aproximação linear por partes do eixo medial (situado na linha média de um corpo) utilizando triangulação de Delaunay com peso e o *Power Diagram* para obter uma *variedade* linear por partes da superfície.

Power Diagram de um conjunto de pontos com pesos é o diagrama de Voronoi utilizando a distância com peso ao invés da distância Euclidiana. Triangulação de Delaunay com peso é o dual do *Power Diagram* (GOIS, 2004).

Segue abaixo o algoritmo do *power crust* (GOIS, 2004):

---

para todo $\mathbf{x} \in X$ **faça**
    Calcule o conjunto de bolas polares P_x
fim
Calcule o *Power Diagram* de P_x
para todo $\mathbf{x} \in P_x$ **faça**
    Marque as bolas polares como internas ou internas à superfície
fim
Retorne as faces do *Power Diagram* separando as células internas e externas como o *Power Crust.*

A Figura 12 ilustra as etapas do algoritmo *Power Crus*t para o caso bidimensional: (a) Um objeto com seu eixo médio, e uma circunferência bitangente a curva, de centro em um ponto no eixo médio. (b) O diagrama de Voronoi da amostragem. No espaço bidimensional é possível selecionar todos os vértices de Voronoi como pólos, mas isto não é possível em espaços tridimensionais. (c) Bolas polares interiores e exteriores. (d) O *power diagram* do conjunto de bolas polares. (e) O *power crust* e o *power shape* (aproximação do eixo medial interno) (AMENTA, CHOI e KOLLURI, 2001a).

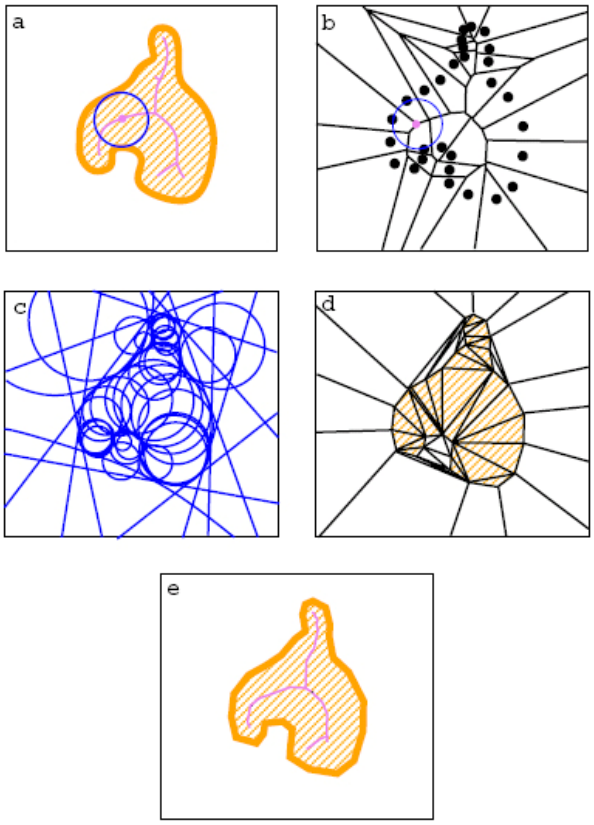


Figura 12: As etapas do algoritmo Power Crust 2D. Fonte: (AMENTA, CHOI e KOLLURI, 2001a).

3.5 Cocone

AMENTA et al. (2000) propôs um algoritmo denominado *cocone*. O *cocone* é baseado no *crust*, mas ao contrario deste, que exige o cálculo de duas triangulações de Delaunay, o *cocone* utiliza apenas uma, reduzindo o custo computacional.

Além da utilização de apenas uma triangulação de Delaunay, o *cocone* também elimina a etapa de pós-processamento *filtering by normal* (AMENTA, BERN e KAMVYSSELIS, 1998), realizada diretamente pelo *cocone*, continuando necessária à segunda etapa de pós-processamento *trimming*.

3.6 Tight Cocone

O *tight cocone* é uma etapa de pós-processamento do *cocone* (DEY e GOSMWAMI, 2002). Como saída, o algoritmo gera uma superfície denominada *Water Tight*.

Uma superfície é denominada *water tight* se é um 2-complexo simplicial em $\mathbb{R}^3$ cujo espaço gerado é o mesmo que a fronteira de uma 3-variedade em $\mathbb{R}^3$ (GOIS, 2004).

O algoritmo consiste em nomear os tetraedros de Delaunay como internos ou externos, baseado na superfície obtida pelo *cocone*, Tetraedros nomeados como externos são removidos, e a fronteira dos tetraedros restantes formam uma superfície *water tight* (DEY e GOSMWAMI, 2002).

3.7 Considerações Finais

Este capítulo apresentou alguns métodos que não são baseados em redes neurais mais utilizados em reconstrução de superfície. Mais do que uma revisão bibliográfica dos métodos que precederam o uso das redes neurais, este capítulo serve como fonte de idéias e inspirações que unidas com as características positivas apresentadas pelas redes neurais, na resolução do presente problema, seja possível desenvolver métodos que incluam as boas características das duas abordagens, neural e tradicional. O capítulo seguinte trata dos modelos de redes neurais que serão empregados no problema da reconstrução de superfícies.

4 REDES NEURAIS AUTO-ORGANIZÁVEIS

O objetivo deste capítulo é apresentar modelos de redes neurais auto-organizáveis que possam ser usadas para a reconstrução de superfícies 3D. A seção 4.1 introduz características comuns aos modelos abordados, bem como suas principais diferenças. As seções seguintes (4.2 - 0) são reservadas a uma descrição mais detalhada de cada modelo, respectivamente: *Self-Organizing Map* (SOM), *Growing Cell Structures* (GCS), *Growing Neural Gas* (GNG), e *Growing Grid* (GG), incluindo descrição das arquiteturas e algoritmos de aprendizagem. A seção 4.6 traz considerações pertinentes ao trabalho.

4.1 Introdução

Baseado em FRITZKE (1997) podemos considerar duas classes de redes neurais auto-organizáveis. A primeira classe é formada pelos modelos cuja topologia da camada de saída, ou da rede propriamente dita, possui uma dimensionalidade fixa (i.e. 1D ou 2D), e a segunda, pelos modelos onde a dimensionalidade da rede varia de acordo com a dimensionalidade dos dados. Na primeira classe temos o SOM, o GCS e o GG, e na segunda classe temos o GNG, considerando apenas os modelos presentes neste trabalho.

O SOM possui um número fixo de neurônios, enquanto os outros modelos (GCS, GNG, GG) são modelos incrementais, ou seja, o número de neurônios presentes na rede é definido ao longo do treinamento. O GG se assemelha ao SOM por possuir uma grade

regular, mesmo após a inserção de novos neurônios, enquanto o GCS e o GNG geram grades irregulares.

4.2 Self-Organizing Map (SOM)

O *Self-Organizing Map* (SOM) (KOHONEN, 1990; 1982) é um modelo de rede neural auto-organizável baseada em *aprendizagem competitiva*. Diferente da maioria dos modelos de redes neurais modernos, o SOM possui uma forte inspiração neurofisiológica.

O objetivo do SOM é realizar um mapeamento de um padrão de dimensão arbitrária, geralmente alta, em uma grade de neurônios discreta de baixa dimensionalidade, geralmente 1D ou 2D, e realizar este mapeamento buscando preservar ao máximo a topologia original dos dados.

4.2.1 Inspiração neurofisiológica

A ocorrência de uma organização topográfica no cérebro, especialmente no córtex, vem sendo especulada há aproximadamente cem anos, proporcionada principalmente pela observação de deficiências apresentadas por pessoas vitimas de lesões, hemorragias, tumores ou formações irregulares em diversas áreas do cérebro (KOHONEN, 1990).

Maiores evidências de que diferentes estímulos sensoriais formam mapas fisiológicos ordenados no córtex, foram obtidas principalmente por meio do trabalho de HUBBEL e WIESEL (1962), que utilizando micro-eletrodos no córtex visual de gatos, verificaram que quando eram apresentadas aos animais, imagens de segmentos de reta com

leves diferenças quanto à orientação, áreas localizadas em regiões próximas no córtex eram excitadas. Técnicas mais modernas utilizadas para o monitoramento de funções cerebrais de forma menos invasiva, como o *positron emission tomography* (PET), *gamma camera* e *magnetoencephalography* (MEG) (KOHONEN, 1990), trouxeram evidências diretas de que diferentes tipos de sinais são processados em áreas especializadas do cérebro, principalmente no córtex.

A formação desse mapa ordenado no córtex tem relação direta com o fenômeno de iteração lateral que ocorre entre os neurônios. Quando um neurônio é excitado, uma área circular ao seu redor, entre 50 e 100 μm de raio também sofre uma forte excitação. Ao redor desta, existe uma área circular de aproximadamente 200 até 300 μm de raio, que sofre uma ação inibitória, seguida por uma terceira região que sofre uma excitação fraca e pode alcançar até alguns centímetros de raio (KOHONEN, 1982). A Figura 13 ilustra o fenômeno de iteração lateral entre os neurônios, numa seção planar.

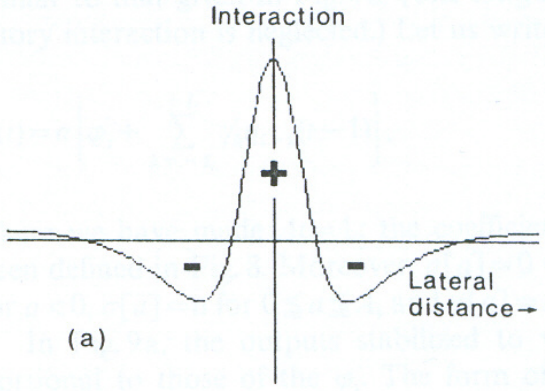


Figura 13: Ilustração do fenômeno de iteração lateral presente no córtex cerebral.
Fonte: (KOHONEN, 1982).

4.2.2 Arquitetura

Considera-se um espaço vetorial de entrada *V* de dimensão *d*. A partir deste espaço um conjunto de amostras $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$ é selecionado de acordo com uma distribuição de probabilidade $p(\mathbf{x})$ desconhecida. Cada amostra é um padrão de entrada representado por um vetor (Equação (4.1)):

$$\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d. \tag{4.1}$$

A arquitetura básica do SOM consiste de duas camadas de neurônios. A camada de entrada é composta por *d* neurônios de entrada. A camada de saída (ou de Kohonen) consiste de um conjunto de *N* neurônios, denotado por (Equação (4.2)):

$$A = \{c_1, c_2, \ldots, c_N\}, \tag{4.2}$$

organizados em uma grade regular de baixa dimensionalidade, geralmente 1D ou 2D (grades com dimensões maiores são possíveis, porém incomuns), se considerarmos uma grade bidimensional $N = N_1 N_2$, sendo N_1 o número de colunas e N_2 o número de linhas. Para representar uma grade unidimensional, basta definir $N_1 = 1$.

Os neurônios da camada de saída estão conectados com os neurônios adjacentes por uma *relação de vizinhança*, que determina a estrutura ou *topologia* da grade de neurônios. Considerando uma grade unidimensional, cada neurônio c_i, está ligado com o neurônio diretamente à esquerda c_{i-1} e com o neurônio diretamente à direita c_{i+1}. Se considerarmos uma grade bidimensional a topologia pode ser retangular (cada neurônio possui quatro vizinhos) ou hexagonal (cada neurônio possui seis vizinhos), como é mostrado na Figura 14Figura 14.

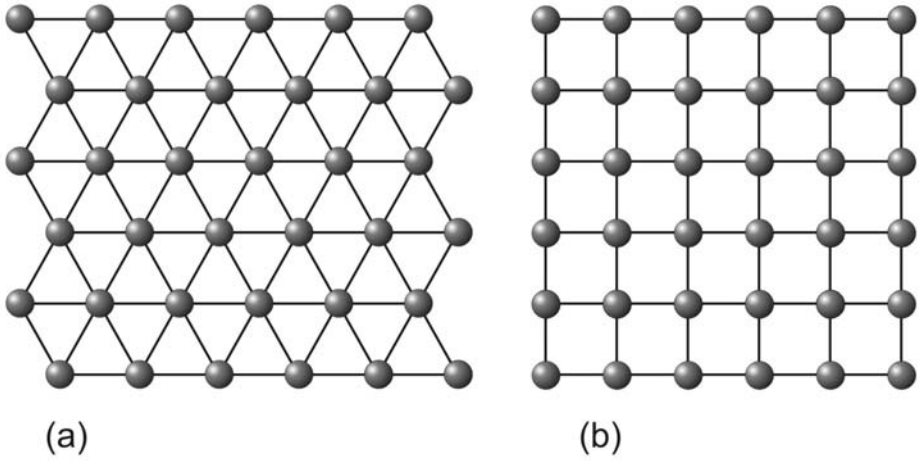


Figura 14: Estrutura do SOM: (a) mapa com topologia hexagonal (6 vizinhos diretos), (b) mapa com topologia retangular (4 vizinhos diretos).

Cada neurônio c_i possui associado um vetor peso sináptico (vetor de reconstrução) representado por (Equação (4.3)):

$$\mathbf{m}_i = \left[m_{i1}, m_{i2}, \ldots, m_{id} \right]^T \in \mathbb{R}^d, \tag{4.3}$$

que indica as coordenadas da sua localização no espaço vetorial de entrada *V*. A Figura 15 mostra um esquema bastante completo e didático da arquitetura do SOM, no espaço de saída.

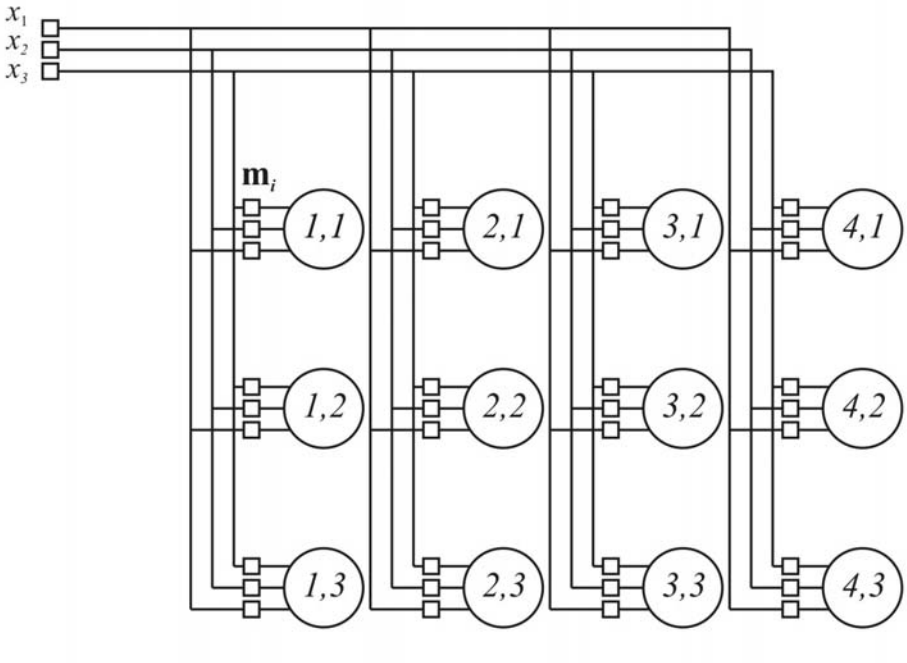


Figura 15: Arquitetura de um SOM com grade 2D e 12 neurônios.

4.2.3 Inicialização

Antes de iniciar o algoritmo de aprendizagem do SOM, os vetores peso sináptico $\mathbf{m}_i$, devem ter seus valores inicializados de maneira adequada. Atualmente são consideradas duas estratégias de inicialização: *randômica* e *linear*. Ambas as opções são coerentes com o principio da *aprendizagem competitiva* que diz: um conjunto de neurônios envolvidos em um processo competitivo são idênticos, exceto pelos valores de seus vetores peso sináptico, que devem responder *diferentemente* a um dado conjunto de sinais de entrada (RUMELHART e ZIPSER, 1985) citado por HAYKIN (2001). O processo de aprendizagem do SOM pode ser dividido em duas fases: uma fase de ordenação, seguida por uma fase de convergência. A seção 4.2.4 traz uma descrição completa das duas fases.

Inicialização randômica: A forma mais simples e geralmente mais utilizada de se inicializar o SOM é atribuindo valores arbitrários aos vetores de reconstrução $\mathbf{m}_i(0)$. Isso implica que nenhum estado de organização é considerado inicialmente. Partindo deste principio, é possível observar o processo de auto-organização sofrido pelos vetores de reconstrução, inicialmente desorganizados, geralmente durante as primeiras centenas de passos, durante a fase de ordenação (KOHONEN, 2001).

Inicialização linear: Considerando um SOM com uma grade *k*-dimensional, primeiro determina-se os *k* autovetores da matriz de autocorrelação de *X* que possuem os maiores autovalores. Estes autovalores definem um subespaço linear *k*-dimensional. Então o arranjo de neurônios é definido ao longo deste subespaço, com seu centróide coincidindo como a média de *X*, e as principais dimensões sendo as mesmas dos *k* maiores autovalores. Como os vetores de reconstrução $\mathbf{m}_i(0)$ já estão ordenados e sua densidade se aproxima, mesmo que de forma grosseira à distribuição de probabilidade $p(\mathbf{x})$, pode-se iniciar o

treinamento começando pela fase de convergência, pulando a fase de ordenação (KOHONEN, 2001).

4.2.4 Algoritmo de aprendizagem

Após a inicialização adequada dos vetores peso sináptico $\mathbf{m}_i(0)$ dos neurônios do SOM, tem inicio um algoritmo de aprendizagem iterativo. Para melhor entendimento, o algoritmo pode ser dividido em três processos essenciais (HAYKIN, 2001): *processo competitivo*, *processo cooperativo*, e *processo adaptativo*.

O processo competitivo

O objetivo do processo competitivo é determinar qual neurônio c_i possui o vetor peso sináptico $\mathbf{m}_i$, mais próximo (similar) do padrão de entrada $\mathbf{x}$ apresentado à rede.

A cada passo (iteração) t do algoritmo, um padrão $\mathbf{x}$ é selecionado aleatoriamente do conjunto de dados X, e a distância entre $\mathbf{x}$ e o vetor peso sináptico $\mathbf{m}_i$ de todos os neurônios da rede é computada, considerando alguma medida de distância representada por $d(\mathbf{x}, \mathbf{m}_i)$. O neurônio que possuir o vetor peso $\mathbf{m}_i$ que minimiza essa medida de distância (i.e. o vetor peso $\mathbf{m}_i$ mais similar à entrada $\mathbf{x}$) é considerado o *neurônio vencedor* (BMU[4]) denotado pelo índice w (Equação (4.4)):

$$w = \arg\min_i \left\{ d(\mathbf{x}, \mathbf{m}_i) \right\}. \qquad (4.4)$$

[4] Do inglês *Best Match Unit*, em português “unidade mais bem casada”, ou “unidade mais similar”.

A medida de distância mais utilizada na determinação do neurônio vencedor é a *distância Euclidiana*. Nesse caso, o neurônio vencedor *w* é determinado pela condição (Equação (4.5)):

$$w = \arg\min_i \left\{ \left\| \mathbf{x} - \mathbf{m}_i \right\| \right\}, \textit{ ou} \tag{4.5}$$

$$\left\| \mathbf{x} - \mathbf{m}_w \right\| = \min_i \left\{ \left\| \mathbf{x} - \mathbf{m}_i \right\| \right\}. \tag{4.6}$$

Outras medidas de distância podem ser adotadas, como por exemplo, a distância de Mahalanobis (FESSANT et al., 2001), e a distância de Hamming (COSTA, 1999), entre outras.

O processo cooperativo

O *processo cooperativo* compreende a definição de uma função de vizinhança h_{wi} centrada no neurônio vencedor *w*. O objetivo da função de vizinhança é definir uma região de neurônios cooperativos em torno do neurônio vencedor *w*. Para que o algoritmo convirja é necessário que $h_{wi}(t) \to 0$ quando $t \to \infty$ (KOHONEN, 2001).

KOHONEN (2001) descreve duas formas, bastante comuns de se implementar a função de vizinhança. A primeira, e mais simples delas, consiste em definir um conjunto de níveis de vizinhanças ao redor do neurônio vencedor *w*, denotado por $N_w(t)$. A função de vizinhança é definida da seguinte maneira (Equação (4.7)):

$$h_{wi}(t) = \begin{cases} 1, i \in N_w(t) \\ 0, i \notin N_w(t), \end{cases} \tag{4.7}$$

onde $N_w(t)$ é alguma função monotônica decrescente no tempo. A Figura 16 ilustra esta definição da função de vizinhança, considerando um mapa com topologia retangular e topologia hexagonal.

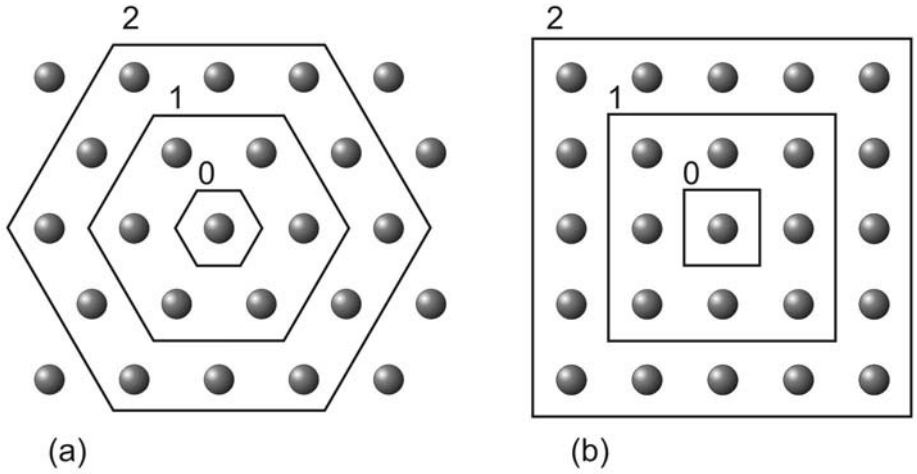


Figura 16: Níveis de vizinhança discreta: (a) grade hexagonal, (b) grade retangular.

A segunda forma, também bastante utilizada como *função de vizinhança* é a função Gaussiana (Figura 17 e Figura 18):

$$h_{wi}(t) = \exp\left(-\frac{\left\|\mathbf{r}_w - \mathbf{r}_i\right\|^2}{2\sigma^2(t)}\right), \tag{4.8}$$

onde o parâmetro $\sigma(t)$ (desvio padrão) que define o raio da função deve ser, por sua vez, alguma função monotônica decrescente no tempo. HAYKIN (2001) sugere o uso de uma função de decaimento exponencial descrito em (RITTER et al., 1992; OBERMAYER, RITTER e SCHULTEN, 1991):

$$\sigma(t) = \sigma(0)\exp\left(-\frac{t}{\tau_1}\right), \tag{4.9}$$

onde $\sigma(0)$ é o valor inicial de σ, e τ_1 é uma *constante de tempo* do SOM.

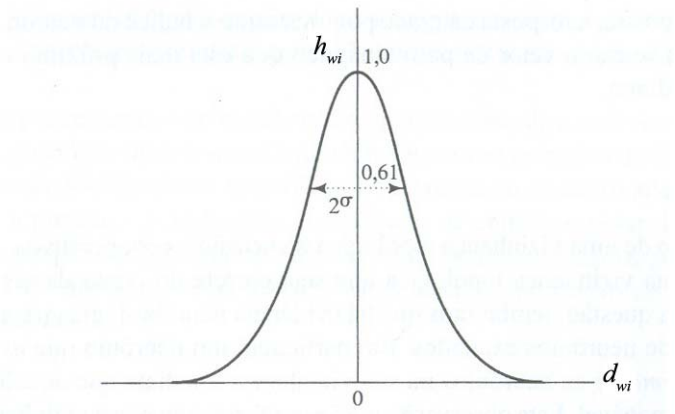


Figura 17: Função de vizinhança topológica Gaussiana unidimensional. Conforme a distância aumenta a partir do neurônio vencedor (0), a intensidade com que os vetores pesos são atualizados diminui. Fonte: (HAYKIN, 2001).

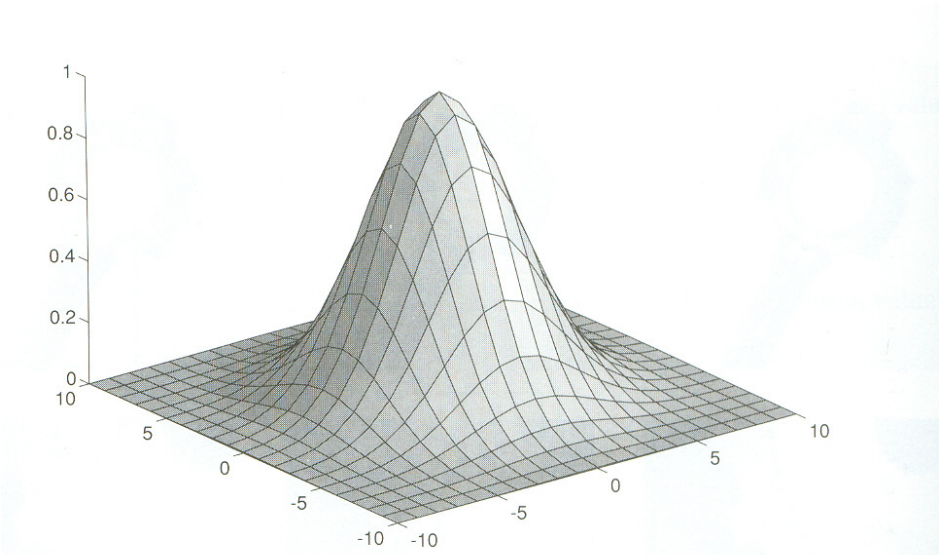


Figura 18: Função gaussiana bidimensional. Usada para modelar a função de vizinhança de um SOM bidimensional. Fonte: (JAIN, KASTURI e SCHUNK, 1995).

Note que o primeiro método, se ocupa apenas em definir até qual nível de vizinhança os neurônios terão seus vetores peso sináptico $\mathbf{m}_i$ atualizados a cada etapa *t* do algoritmo, e todos os neurônios dentro deste conjunto tem seus vetores peso $\mathbf{m}_i$ atualizados com a mesma magnitude do neurônio vencedor *w*. Já utilizando a função Gaussiana, a magnitude com que os neurônios são atualizados diminui em função da distância (no mapa de neurônios discreto) em relação ao neurônio vencedor *w* (Figura 17).

Processo adaptativo

Durante o processo adaptativo, ocorre a modificação dos vetores peso sináptico $\mathbf{m}_i$ em relação à entrada **x** de forma iterativa, utilizando a equação (4.10) (HAYKIN, 2001; KOHONEN, 2001; KOHONEN, 1990; KOHONEN; 1982):

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)\, h_{wi}(t)\big(\mathbf{x} - \mathbf{m}_i(t)\big), \qquad (4.10)$$

que é aplicada a todos os neurônios da grade dentro da região de vizinhança $h_{wi}(t)$ definida durante o processo cooperativo. A Equação (4.10) é responsável por mover os vetores peso sináptico pelo espaço de entrada *d*-dimensional na direção do sinal de entrada **x**, junto com um conjunto de neurônios vizinhos definidos pela função de vizinhança $h_{wi}(t)$ (Equação (4.7) ou (4.8)) (Figura 19), todos ponderados pelo parâmetro da taxa de aprendizagem $\alpha(t)$, que assim como a largura da função de vizinhança deve ser uma função monotônica decrescente no tempo. HAYKIN (2001) sugere um decaimento exponencial para $\alpha(t)$, denotado pela equação (4.11):

$$\alpha(t) = \alpha(0)\exp\left(-\frac{t}{\tau_2}\right), \qquad (4.11)$$

onde α_0 representa o valor inicial da taxa de aprendizagem, e τ_2 é outra constante de tempo do SOM.

Os três processos aqui mostrados são aplicados a cada iteração do algoritmo. O algoritmo por sua vez possui duas fases: a fase de ordenação e a fase de convergência.

Fase de ordenação: Partindo do pressuposto que um SOM é inicializado randomicamente, seus vetores de reconstrução $\mathbf{m}_i(0)$ se encontram em completa desordem. É durante a fase de ordenação, ou de auto-organização que ocorre a ordenação topológica dos vetores de peso. Esta fase exige um número total de iterações em torno de 1000 ou mais, e uma atenção

especial é requerida na escolha do tamanho da região de vizinhança $N_w(t)$ (KOHONEN, 2001): se o tamanho de $N_w(t)$ for muito pequeno no inicio do algoritmo o mapa não será ordenado por completo. Isso pode ser evitado iniciando $N_w(0)$ com um valor grande e ir diminuindo lentamente com o tempo até um valor dos vizinhos mais próximos ao neurônio vencedor *w*, ou até mesmo apenas *w*. Considerando um SOM com grade bidimensional, pode-se igualar o tamanho inicial $\sigma(0)$ da função de vizinhança ao raio da grade de neurônios, e especificar a constante de tempo aplicada na equação (4.8) na forma (equação (4.12)) (HAYKIN, 2001):

$$\tau_1 = \frac{1000}{\log \sigma(0)}, \tag{4.12}$$

supondo o número total de iterações ser 1000.

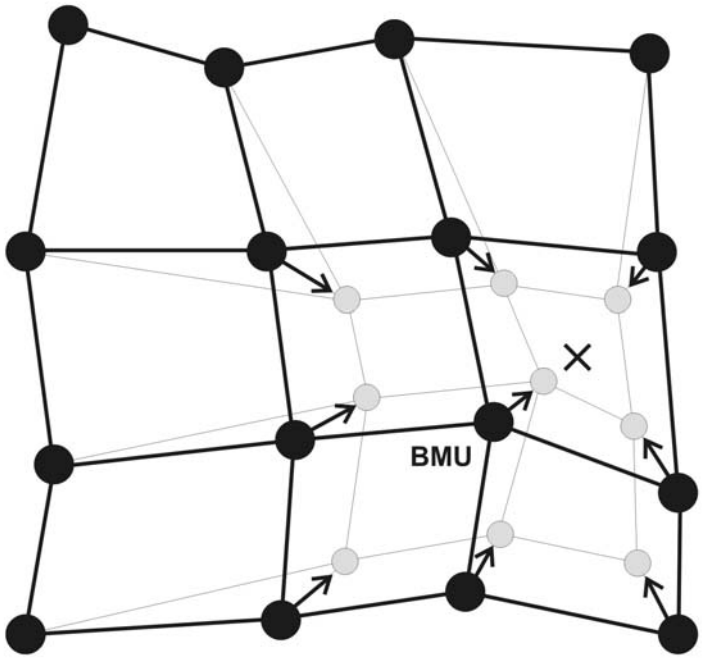


Figura 19: Ação da Equação (4.10) na atualização do neurônio vencedor (BMU) e sua vizinhança topológica na direção do sinal de entrada x, mostrada no espaço de entrada.

O parâmetro da taxa de aprendizado $\alpha(t)$ também merece atenção especial. HAYKIN (2001) sugere que $\alpha(t)$ inicie com um valor próximo a 0,1, e define os seguintes valores a serem utilizados pela Equação (4.11):

$$\alpha(0) = 0{,}1$$

$$\tau_2 = 1000$$

onde também é considerado um treinamento com 1000 iterações.

Fase de convergência: É durante a fase de convergência que ocorre a sintonia fina do mapa de características, necessária para produzir uma quantização estatística do espaço vetorial de entrada *V*. A fase de convergência leva muitos ciclos de iteração a mais do que a fase de ordenação, como regra geral escolhe-se um número de iterações maior que 500 vezes o número de neurônios da grade (KOHONEN, 2001; HAYKIN, 2001).

Durante a fase de convergência a função de vizinhança $h_{wi}(t)$ deve englobar apenas os vizinhos mais próximos do neurônio vencedor, podendo ser reduzida a apenas o próprio neurônio vencedor.

A taxa de aprendizado $\alpha(t)$ deve ser mantida em um valor pequeno, HAYKIN (2001) sugere um valor em torno de 0,01. É importante não permitir que este valor chegue a zero. A equação (4.11) satisfaz essa exigência, lembrando de alterar o parâmetro τ_2 para o número de iterações escolhido para a fase de convergência.

4.2.5 Algoritmo de aprendizado em lote

Assim como o algoritmo básico do SOM, o algoritmo em lote também é seqüencial. Ao invés de apresentar apenas um sinal por vez. A cada iteração, todo o conjunto de dados *X* é particionado de acordo com as regiões de Voronoi do mapa de vetores peso sináptico, ou seja,

cada vetor de dados pertence ao conjunto de dados do neurônio mais similar. Os vetores peso sináptico são atualizados como (equação (4.13)):

$$\mathbf{m}_i(t+1) = \frac{\sum_{j=1}^{N} h_{wi}(t)\mathbf{x}_j}{\sum_{j=1}^{N} h_{wi}(t)}, \qquad (4.13)$$

onde w é o neurônio vencedor para o sinal de entrada $\mathbf{x}_j$. O valor do novo vetor peso é a média dos dados amostrados, ponderada pela função de vizinhança $h_{wi}(t)$. O parâmetro da taxa de aprendizagem α não é utilizado (VESANTO, 2000).

4.3 Growing Cell Structures (GCS)

O *Growing Cell Structures* (GCS) (FRITZKE, 1991) é um modelo de rede neural auto-organizável incremental. Foi proposto primeiramente para solucionar alguns problemas do SOM, entre eles, a dificuldade de escolher a estrutura e o tamanho da rede, e definir uma função de decremento para vários parâmetros (FRITZKE, 1997).

4.3.1 Arquitetura do GCS

Para as redes GCS são definidas estruturas mínimas denominadas simplexos *k*-dimensionais.

No início do treinamento a rede é composta apenas por um único simplexo *k*-dimensional. Para $k=1$, o simplexo *k*-dimensional é um segmento de reta; para $k=2$ um triângulo; para $k=3$ um tetraedro; e para $k>3$ a estrutura obtida é denotada hipertetraedro (Figura 20). Os $(k+1)$ vértices de um simplexo correspondem aos neurônios, e as $(k+1)k/2$ arestas, às relações de vizinhança topológica entre os neurônios. Durante o treinamento, novos

neurônios são adicionados à rede, e às vezes alguns neurônios são removidos. Qualquer modificação feita na rede, como inserção ou remoção de neurônios, é realizada de maneira que após a operação a rede continue formada apenas por simplexos *k*-dimensionais (FRITZKE, 1993).

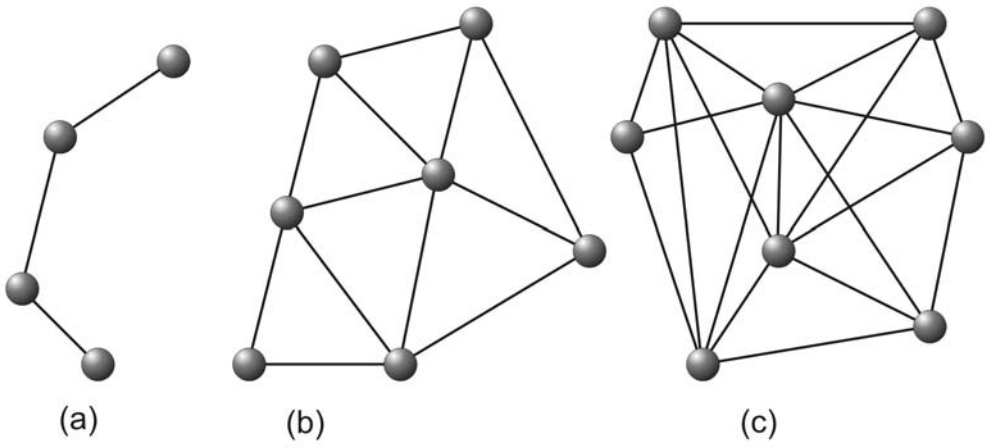


Figura 20: Topologia do GCS para: (a) *k*=1, (b) *k*=2, e (c) *k*=3.

4.3.2 Algoritmo de aprendizagem

A seguir é descrito o algoritmo de aprendizagem do GCS. O algoritmo é bastante similar ao utilizado pelo SOM, exceto por duas importantes diferenças: a taxa de aprendizagem é mantida constante durante todo o processo de treinamento, e apenas o neurônio vencedor e sua vizinhança topológica direta têm seus vetores peso sináptico atualizados.

O algoritmo é constituído dos seguintes passos:

1. Começar o treinamento com apenas um simplexo *k*-dimensional, inicializando os vetores peso sináptico $\mathbf{m}_i$ dos $(k+1)$ neurônios com valores aleatórios.
2. Selecionar um sinal de entrada **x**, de acordo com a distribuição de entrada $p(\mathbf{x})$.

3. Determinar o neurônio vencedor *w*, escolhendo o neurônio c_i que possuir o vetor de pesos $\mathbf{m}_i$ mais próximo do sinal de entrada **x**, considerando alguma métrica pré-definida, geralmente, a medida de distância Euclidiana (Equação (4.14)):

$$\|\mathbf{m}_w - \mathbf{x}\| = \min_{i \in A} \|\mathbf{m}_i - \mathbf{x}\|. \tag{4.14}$$

4. Somar a distância quadrática entre o sinal de entrada **x** e o vetor peso sináptico do neurônio vencedor *w* para calcular a variável de erro local E_w (Equação (4.15)):

$$\Delta E_w = \|\mathbf{m}_w - \mathbf{x}\|^2. \tag{4.15}$$

5. Mover $\mathbf{m}_w$ e seus vizinhos topológicos diretos na direção de **x** ponderados por ε_w e ε_n, respectivamente, de acordo com a Equação (4.16):

$$\begin{aligned} \Delta\mathbf{m}_w &= \varepsilon_w (\mathbf{x} - \mathbf{m}_w), \\ \Delta\mathbf{m}_n &= \varepsilon_n (\mathbf{x} - \mathbf{m}_i), \quad (\forall i \in N_w), \end{aligned} \tag{4.16}$$

onde N_w denota o conjunto de vizinhos topológicos diretos da unidade c_w.

6. Considerando λ algum valor determinado pelo usuário, que represente o intervalo de iterações entre a inserção de cada novo neurônio. Se o número de sinais de entrada gerados (ou passos do algoritmo) até o momento, for um inteiro múltiplo do parâmetro λ, inserir um novo neurônio, por meio de uma operação de subdivisão de aresta:

   - Determinar a unidade *i* que possui o maior erro acumulado (Equação (4.17)):

$$q = \arg\max_i \{E_i\}, \quad (\forall i \in A). \tag{4.17}$$

   - Inserir uma nova unidade *r* dividindo a maior aresta que emana de *q*. Considerando que esta aresta leva à unidade *f*, inserir duas novas conexões (q,r) e (r,f), e remover a conexão original (q,f). Para reconstruir a estrutura de forma que ela continue consistindo apenas de simplexos *k*-dimensionais, a nova unidade *r* deve estar conectada com todos os vizinhos comuns de *q* e *f*, i.e. com todas as unidades dentro do conjunto $N_q \cap N_f$.

- Obter os valores do vetor de reconstrução $\mathbf{m}_r$, interpolando os valores de $\mathbf{m}_q$ e $\mathbf{m}_f$ (Equação (4.18)):

$$\mathbf{m}_r = 0.5\left(\mathbf{m}_q + \mathbf{m}_f\right). \tag{4.18}$$

- Diminuir o valor das variáveis de erro de todos os vizinhos de r (Equação (4.19)):

$$\Delta E_i = -\frac{1}{\left|N_r\right|}\sum_{i \in N_r} E_i, \quad \left(\forall i \in N_r\right) \tag{4.19}$$

- Determinar a variável de erro da nova unidade r pelo valor médio de seus vizinhos por meio da Equação (4.20):

$$E_r = \frac{1}{\left|N_r\right|}\sum_{i \in N_r} E_i \tag{4.20}$$

7. Diminuir a variável de erro de todos os neurônios (Equação (4.21)):

$$\Delta E_c = -\beta E_c, \quad (\forall c \in A) \tag{4.21}$$

8. Se o critério de parada adotado não for satisfeito, voltar para o passo 2.

Incluir um critério para remoção de neurônios no GCS é possível quando for necessário gerar redes desconectadas, por exemplo, para representar agrupamentos (clusters) presentes nos dados. Uma estratégia para remoção de arestas é mostrada em FRITZKE (1993).

Uma variação interessante do GCS é a utilização de redes fechadas (sem bordas), por exemplo, um anel para grades unidimensionais ou uma malha de triângulos fechada para grades bidimensionais, que FRITZKE (1991) propôs utilizá-las na resolução, por exemplo, do problema de caixeiro viajante (FRITZKE, 1997).

4.3.3 Exemplo de GCS

Na Figura 21 um GCS com rede bidimensional é utilizado para mapear um espaço de entrada com diferentes dimensionalidades em diferentes áreas. O espaço de entrada consiste de um arco, um segmento de reta (região 1D), um retângulo (região 2D), e um paralelepípedo (região 3D). O GCS realiza um mapeamento de padrões de entrada, de dimensão arbitrária em uma rede com dimensão fixa, ao mesmo tempo em que busca preservar a topologia original dos dados. É possível observar na Figura 21 que a área retangular é mais bem mapeada, pois possui a mesma dimensão da rede, já para mapear a área tridimensional a rede é 'dobrada' buscando preservar a topologia do espaço de entrada.

Quando um espaço de entrada é mapeado por uma rede com dimensionalidade menor, ocorre uma redução da dimensionalidade dos dados, possibilitando, por exemplo, visualizar de forma mais natural dados com dimensões elevadas quando mapeados sobre redes bidimensionais.

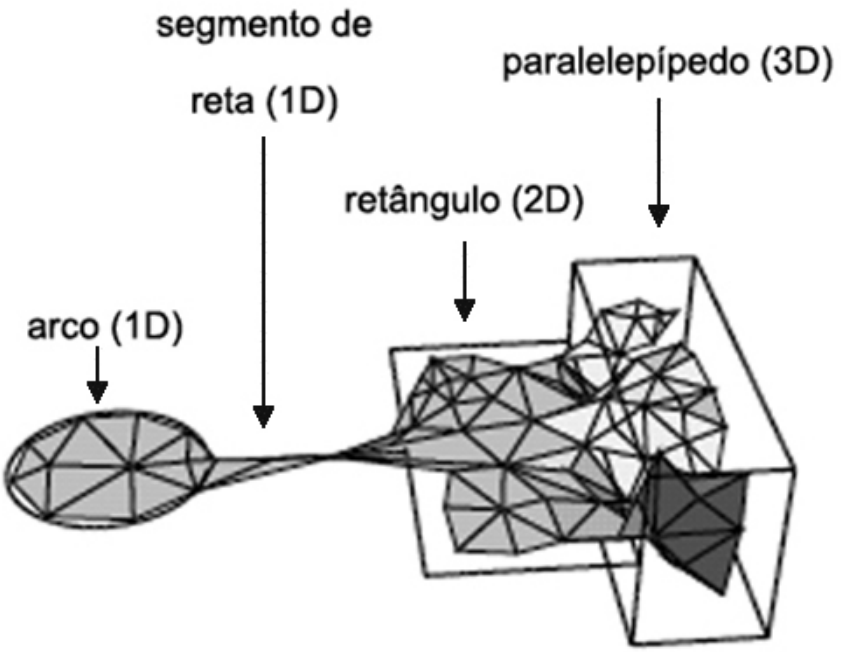


Figura 21: Um GCS com rede bidimensional mapeando uma distribuição de sinais de entrada com diferentes dimensões em diferentes áreas. Fonte: (FRITZKE, 1997).

4.4 Growing Neural Gas

O *growing neural gas* (GNG) (FRITZKE, 1995a) pode ser visto como uma variação do GCS, onde nenhuma restrição quanto à dimensionalidade do espaço de saída da rede é considerada (FRITZKE, 1997). A topologia da rede do GNG reflete a topologia do espaço de entrada, e assim pode possuir dimensionalidades diferentes em regiões diferentes, inviabilizando sua utilização na visualização de dados de dimensão elevada (FRITZKE, 1996). É justamente esta característica que permite ao GNG aprender a topologia correta de um conjunto de dados.

4.4.1 Arquitetura do GNG

No GNG a dimensionalidade do espaço de saída é dependente do espaço de entrada, e pode variar localmente. O mecanismo que controla o crescimento da rede no GCS, e a geração da topologia na Aprendizagem Hebbiana Competitiva (MARTINEZ, 1993) estão combinados no algoritmo do GNG.

A Aprendizagem Hebbiana Competitiva por si só não altera os pesos das unidades. Ele apenas gera um determinado número de conexões entre as unidades da rede, e o conjunto de arestas gerado é um subconjunto da triangulação de Delaunay. O principio deste método é o seguinte: para cada padrão de entrada **x**, conecte os dois neurônios mais similares (considerando alguma medida, como a distância Euclidiana, por exemplo), por uma aresta.

Geralmente a Aprendizagem Hebbiana Competitiva é utilizada em conjunto com outros métodos, como por exemplo, em (MARTINEZ e SCHULTEN, 1994) ela é usada em conjunto com o *Neural Gas* (MARTINEZ e SCHULTEN, 1991). Dessa forma, em

conjunto com o *Neural Gas*, ela pode ser vista como uma rede de representação da topologia, assim como o GNG.

4.4.2 Algoritmo de aprendizagem

Os passos do algoritmo de aprendizado do GNG são descritos na seqüência:

1. Inicializar o conjunto *A* contendo dois neurônios c_1 e c_2, atribuindo valores aleatórios a seus vetores sinápticos $\mathbf{m}_1$ e $\mathbf{m}_2$ em $\mathbb{R}^d$ (Equação (4.22)):

$$A = \{c_1, c_2\} \tag{4.22}$$

   Inicializar o conjunto de conexões *C*, de forma a conter uma aresta ligando c_1 e c_2, e a *idade*, que mede a atividade das conexões, em zero (Equações (4.23) e (4.24)):

$$C = \{(c_1, c_2)\}, \tag{4.23}$$

$$idade_{(c_1,c_2)} = 0. \tag{4.24}$$

2. Selecionar um padrão de entrada $\mathbf{x}$ de acordo com $p(\mathbf{x})$
3. Determinar os dois neurônios com vetores peso sináptico mais similares a x, e identificando-os pelos índices w_1 e w_2 (Equações (4.25) e (4.26)):

$$w_1 = \arg\min_i \{\|\mathbf{x} - \mathbf{m}_i\|\} \quad (\forall i \in A), \tag{4.25}$$

$$w_2 = \arg\min \{\|\mathbf{x} - \mathbf{m}_i\|\} \quad (\forall i \in A \setminus w_1). \tag{4.26}$$

4. Se ainda não existir, inserir uma conexão entre w_1 e w_2 em C (Equação (4.27)):

$$C = C \cup \{w_1, w_2\}. \tag{4.27}$$

   Em qualquer caso, definir a idade da conexão entre w_1 e w_2 em zero (Equação (4.28)):

$$idade_{(w_1,w_2)} = 0. \tag{4.28}$$

5. Somar o quadrado da distância entre o sinal de entrada **x** e o vetor peso sináptico mais similar w_1 em uma variável de erro local (Equação (4.29)):

$$\Delta E_{w_1} = \left\| \mathbf{m}_{w_1} - \mathbf{x} \right\|^2 . \tag{4.29}$$

6. Mover $\mathbf{m}_{w_1}$ e sua vizinhança direta do neurônio w_1 na direção de **x** ponderados pelos parâmetros ε_w e ε_n, respectivamente (Equações (4.30) e (4.31)):

$$\Delta \mathbf{m}_{w_1} = \varepsilon_w \left(\mathbf{x} - \mathbf{m}_{w_1} \right), \tag{4.30}$$

$$\Delta \mathbf{m}_i = \varepsilon_n \left(\mathbf{x} - \mathbf{m}_i \right), \quad \left(\forall i \in N_{w_1} \right). \tag{4.31}$$

7. Incrementar a idade de todas as arestas que emanam de w_1 (Equação (4.32)):

$$idade_{(w_1,i)} = idade_{(w_1,i)} + 1, \quad \left(\forall i \in N_{w_1} \right) \tag{4.32}$$

8. Remover as arestas com idade maior que $idade_{\max}$, e se esta operação resultar em unidades sem arestas, removê-las também.
9. Se o número de sinais de entrada gerados até o momento for maior que um inteiro múltiplo de um parâmetro λ, inserir um novo neurônio, da seguinte forma:
    - Determinar a unidade q com o maior erro acumulado (Equação (4.33)):

$$q = \arg\max_i \left\{ E_i \right\}, \quad \left(\forall i \in N_{w_1} \right). \tag{4.33}$$

    - Criar uma nova unidade *r*, e determinar seu vetor de pesos interpolando os vetores de *q* e seu vizinho *f* com a maior variável de erro (Equações (4.34) e (4.35)):

$$A = A \cup \{r\}, \tag{4.34}$$

$$\mathbf{m}_r = 0,5\left(\mathbf{m}_q + \mathbf{m}_f \right). \tag{4.35}$$

    - Inserir arestas conectando a nova unidade *r* com as unidades *q* e *f*, e remover a aresta original entre *q* e *f* (Equações (4.36) e (4.37)):

$$C = C \cup \left\{ (r,q), (r,f) \right\}, \tag{4.36}$$

$$C = C \setminus (q, f). \tag{4.37}$$

- Diminuir a variável de erro de q e f (Equações (4.38) e (4.39)):

$$\Delta E_q = -\alpha E_q, \tag{4.38}$$

$$\Delta E_f = -\alpha E_f. \tag{4.39}$$

Obter a variável de erro de r interpolando as variáveis de erro de q e f de acordo com a Equação (4.40):

$$E_r = 0,5\left(E_q + E_f\right) \tag{4.40}$$

10. Diminuir a variável de erro de todos os neurônios (Equação (4.41)):

$$\Delta E_i = -\beta E_i \quad (\forall i \in A) \tag{4.41}$$

11. Se o critério de parada definido não for satisfeito, voltar para o passo 2.

4.4.3 Exemplo de GNG

A Figura 22 traz um exemplo do algoritmo GNG em ação, mostrando a rede em seis etapas. O GNG se adapta a distribuições de sinais com diferentes dimensionalidades em diferentes regiões do espaço de entrada. No inicio do treinamento (Figura 22a) a rede consiste de apenas dois neurônios posicionados randomicamente no espaço de entrada. Conforme o treinamento avança o número de neurônios cresce (Figura 22b, c, d, e). A última rede mostrada (Figura 22f) não é necessariamente a rede final, uma vez que o processo, a principio, pode continuar indefinidamente. Os parâmetros usados na simulação são: $\lambda = 200$, $\varepsilon_w = 0,05$, $\varepsilon_n = 0,0006$, $\alpha = 0,5$, $idade_{\max} = 88$, e $\beta = 0,0005$ (FRITZKE, 1997).

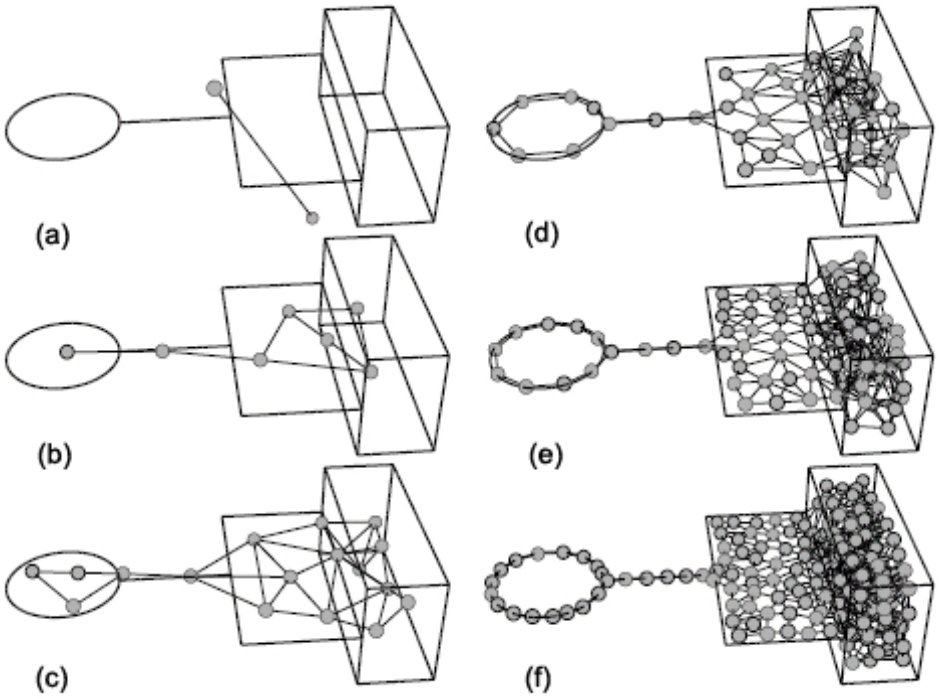


Figura 22: Seqüência de passos ilustrando o processo de aprendizado de topologia realizado pelo GNG. Fonte: (FRITZKE, 1997).

É possível observar na Figura 22 que o conjunto de dados possui áreas unidimensionais, bidimensionais e tridimensionais, respectivamente da esquerda para a direita. (a) A rede começa com apenas dois neurônios. (b) Estrutura da rede após 1000 iterações e com 7 neurônios. (c) Após 3000 iterações e 17 neurônios. (d) Após 9600 iterações e 50 neurônios. Nesta etapa a rede já possui informações corretas, ainda que de maneira precária, quanto à topologia das diferentes áreas do espaço de entrada. (e) Após 19600 iterações e 100 neurônios. (f) Após 39600 iterações e 200 neurônios. Nesta etapa a topologia correta de todas as áreas do espaço de entrada já foi inferida e representada precisamente.

4.5 Growing Grid

O *Growing Grid* (GG) (FRITZKE, 1995b) pode ser visto como uma variante incremental do SOM. Os princípios do mecanismo de crescimento dos GCS e do GNG são aplicados a uma grade retangular de neurônios. O processo de treinamento do GG é semelhante ao do SOM, porém, e também possui duas fases: fase de incremento e fase de convergência.

A fase de incremento é iniciada com uma estrutura mínima, e cresce por meio da inserção de colunas ou linhas inteiras, de forma a manter uma grade regular. Assim como no GCS o parâmetro da taxa de aprendizagem é mantida constante durante esta fase. A rede cresce até alcançar um tamanho pré-estabelecido ou outro critério seja satisfeito. Durante a fase de convergência o tamanho da rede não é mais alterado, e uma função de decaimento é aplicada à taxa de aprendizado para garantir um mapeamento bem ajustado (FRITZKE, 1997).

4.6 Considerações Finais

Este capítulo apresenta os principais modelos de redes neurais com características auto-organizáveis, e baseadas em aprendizagem competitiva, com maior atenção para o SOM. Alguns dos modelos, como o GCS, GNG e GG, possuem características incrementais, ou construtivas. Isso significa que a estrutura e/ou número de neurônios é definido pelo próprio algoritmo durante o processo de aprendizagem.

O próximo capítulo apresenta uma revisão dos principais trabalhos a empregar redes neurais em reconstrução de superfícies.

5 RECONSTRUÇÃO DE SUPERFÍCIES POR REDES NEURAIS

Uma característica comum na maioria dos trabalhos que aplicam redes neurais auto-organizáveis na reconstrução de superfícies é a analogia feita entre os elementos da camada de saída desta rede e uma malha poligonal 3D (LIOU e KUO, 2005, BOUDJEMAI, ENBERG e POSTAIRE, 2003; IVRISSIMTZIS, JEONG e SEIDEL, 2003a; YU, 1999): os neurônios (ou células) da rede neural representam os vértices da malha 3D, os vetores peso sináptico de cada neurônio representam as coordenadas deste vértice no espaço $\mathbb{R}^3$. A relação de vizinhança entre os neurônios vizinhos corresponde às arestas da malha 3D, e o conjunto de pontos desorganizados é equivalente ao conjunto de vetores X de padrões de entrada da rede neural.

5.1 A Abordagem de Yu

Um dos primeiros trabalhos a empregar redes neurais artificiais é apresentado por YU (1999). Uma analogia interessante, para compreender como Yu propôs a utilização do SOM para reconstruir uma superfície é imaginar a rede neural (ou a malha) original como uma folha elástica, onde durante o processo de aprendizagem, os neurônios (ou vértices) aos pouco vão se posicionando de forma a envolver a nuvem de pontos, e assim assumir a sua forma. Quando é considerado um SOM com topologia retangular, os polígonos que formam a malha

são retângulos, já quando é considerado um SOM com topologia hexagonal, a malha é formada por triângulos (Figura 14).

Ao contrário dos métodos clássicos (capítulo 3), que buscam criar conexões entre os pontos mais próximos para assim construir a malha, o método de Yu, considera uma malha com uma topologia equivalente a do objeto original, usando informação *a priori* do objeto para isso. Partindo deste ponto, o algoritmo de aprendizado do SOM é responsável por mover os vértices da malha para próximo da nuvem de pontos de forma ordenada, envolvendo a nuvem de pontos e modelando a forma do objeto.

Troca de Arestas: Algumas vezes é possível que o SOM encontre dificuldades em aproximar estruturas côncavas presentes na superfície. A superfície reconstruída pelo SOM, pode apresentar algumas faces alongadas defeituosas conectando partes separadas destas estruturas e conseqüentemente cobrindo-as (Figura 23).

Figura 23: Simplesmente o algoritmo SOM apresenta dificuldades em aproximar estruturas côncavas, este problema que pode ser resolvido por heurísticas. Fonte: (YU, 1999).

Para localizar essas faces alongadas que não fazem parte da superfície do objeto real, Yu utilizou um método relativamente simples, que consiste em calcular a distância mínima entre o centróide de cada triângulo da malha 3D e o conjunto de pontos. Considerando o triangulo *T*, e $d(\mathbf{x},\mathbf{y})$ uma função que retorna a distância entre dois vetores arbitrários (**x** e **y**), a distância mínima (*DM*) é definida como (Equação (5.1)):

$$DM(T) = \min_{\mathbf{x} \in X} d\left(\mathbf{x}, \text{centroide de } T\right), \qquad (5.1)$$

onde *X* é o conjunto de pontos desorganizados. São considerados triângulos defeituosos, aqueles que possuem a distância mínima (*DM*) maior do que a média das distâncias mínimas de todos os outros triângulos.

Uma vez localizados os triângulos defeituosos, é necessário removê-los para aumentar a qualidade da superfície reconstruída. Yu optou por utilizar uma operação sobre a malha, chamada “troca de arestas” (HOPPE, 1993).

O procedimento adotado consiste em, para cada triângulo problemático, localizado usando a Equação (5.1) , escolher qual das arestas do triângulo deve sofrer a operação de “troca de aresta”, calculando o desvio $Dev(e)$ de uma aresta, da forma (Equação (5.2)):

$$Dev(e) = DM\left(T_1\right) + DM\left(T_2\right), \qquad (5.2)$$

onde *e* representa alguma aresta da malha, e *T1* e *T2*, dois triângulos que compartilham a aresta *e*.

Primeiro, uma troca simples é testada, sendo aceita, se e somente se, o desvio $Dev(e)$ da nova aresta for menor que da anterior e a distância mínima *DM* do menor dos dois novos triângulos for menor que um limiar determinado. Se a troca simples falhar, deve se efetuar uma troca dupla, que consiste em manter a primeira troca e então, trocar a aresta com o segundo maior desvio $Dev(e)$. A troca dupla é aceita se e somente se, as mesmas condições impostas para a aceitação da troca simples forem satisfeitas. Se a troca dupla também falhar, a malha deve voltar à sua estrutura original, e nenhuma troca de arestas é realizada.

Aprendizagem em *multiresolução*: Com o objetivo de melhorar o desempenho do algoritmo, Yu aplicou o algoritmo de aprendizagem do SOM em múltiplas resoluções da malha. Características de baixa freqüência podem ser aprendidas (i.e., aproximadas) por malhas de baixa resolução, diminuindo o custo computacional. Após as

características de baixa resolução terem sido aproximadas, o que resta são apenas os detalhes em alta freqüência. Após cada aplicação do algoritmo, são executadas as trocas de arestas necessárias, a resolução da malha é aumentada, via *poligonalização* dos triângulos da malha, e o processo se repete até a malha alcançar a qualidade desejada. A Figura 24 ilustra o processo de aprendizagem em *multiresolução,* usando o "Coelho de Stanford" (STANFORD, 2007).

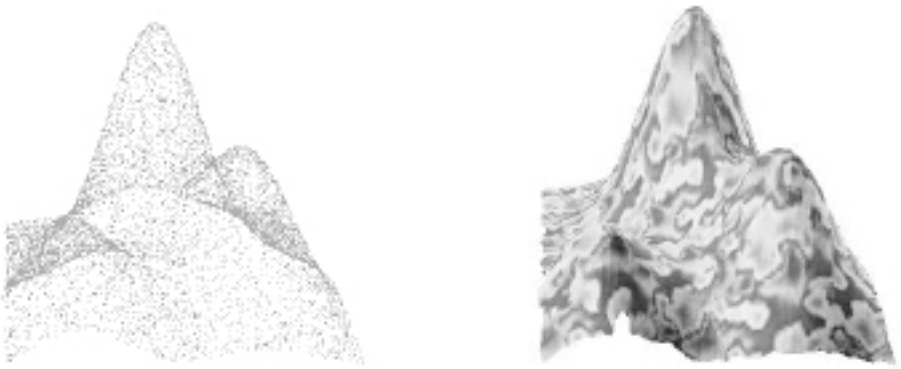
(a) **(b)** **(c)**

Figura 24: O processo de aprendizagem em multiresolução, mostrado em três fases, sendo: (a) baixa resolução; (b) média resolução; (c) alta resolução. Fonte (YU, 1999)

Resultados: Como resultados, Yu reconstruiu duas nuvens de pontos: A primeira, uma superfície aberta (com borda) gerada artificialmente a partir de uma mistura de três funções Gaussianas. Para reconstrução foi utilizado um SOM com topologia retangular, e formato global da grade de neurônios plana. A Figura 25 ilustra o experimento.

A segunda nuvem de pontos utilizada foi a do "Coelho de Stanford", um conjunto de pontos bastante utilizado para testes e demonstrações. O "Coelho de Stanford" foi reconstruído por um SOM com formato global esférico, baseado na *subdivisão* do icosaedro. Iniciando com icosaedro em alguma freqüência baixa, após algumas iterações do SOM, são efetuadas as operações de "troca de arestas" necessárias para corrigir as possíveis áreas defeituosas. Então a freqüência do icosaedro é aumentada, e o algoritmo é aplicado

novamente, seguindo as regras do aprendizado em *multiresolução* (Figura 24). A Figura 26 ilustra os resultados.



(a) **(b)**

Figura 25: Resultado da reconstrução da nuvem de pontos, de uma superfície aberta (com borda). Foi utilizado um SOM com grade de neurônios plana (2D). (a) nuvem de pontos; (b) superfície reconstruída. Fonte: (YU, 1999).

(a) **(b)**

Figura 26: Resultados da reconstrução da nuvem de pontos do coelho de Stanford. Foi utilizado um SOM com formato da grade esférico, baseado na subdivisão do icosaedro. (a) nuvem de pontos; (b) superfícies reconstruída. Fonte: (YU, 1999).

5.2 Outros trabalhos envolvendo o SOM

Em outro trabalho que utiliza SOM para reconstrução de superfícies, BOUDJEMAÏ, ENBERG e POSTAIRE (2003), apresentam alguns exemplos de nuvens de pontos bidimensionais e tridimensionais reconstruídas via SOM com grade 2D plana tradicional, visando mostrar principalmente que a reconstrução obtida pela grade plana sobre a nuvem de pontos volumétrica, não é tão eficiente quanto a obtida sobre a nuvem de pontos bidimensional.

Para resolver tal problema, os autores propõem a implementação do SOM com grade toroidal (Figura 27a) e grade esférica (Figura 27b, c, d).

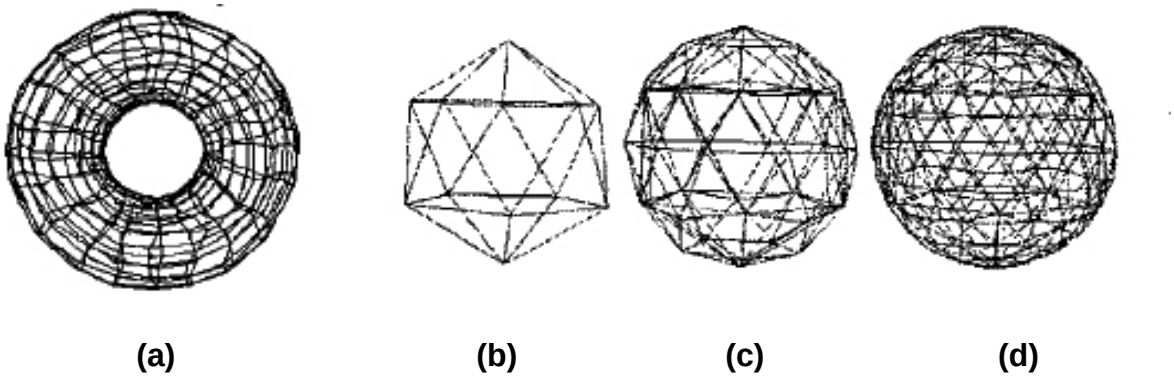


Figura 27: Configurações de grades de neurônios possíveis para o SOM. (a) formato toroidal. (b, c, d) formato de esfera, baseado na *poligonalização* do icosaedro. (b) Freqüência 0. (c) Freqüência 1. (d) Freqüência 2. Fonte: (BOUDJEMAÏ, ENBERG e POSTAIRE, 2003).

A implementação com grade esférica é bastante semelhante à proposta em YU (1999), diferindo desta por não utilizar aprendizado em *multiresolução* nem operações de "troca de aresta". Da mesma forma que YU (1999), a grade esférica é obtida pela *poligonalização* de um icosaedro, entretanto, diferente de YU (1999) é utilizado para isto, um esquema chamado subdivisão *butterfly* (ZORIN, SCHRÖDER, e SWELDENS, 1996). Os resultados obtidos, utilizando o SOM com grade toroidal e esférica é mostrado na Figura 28.

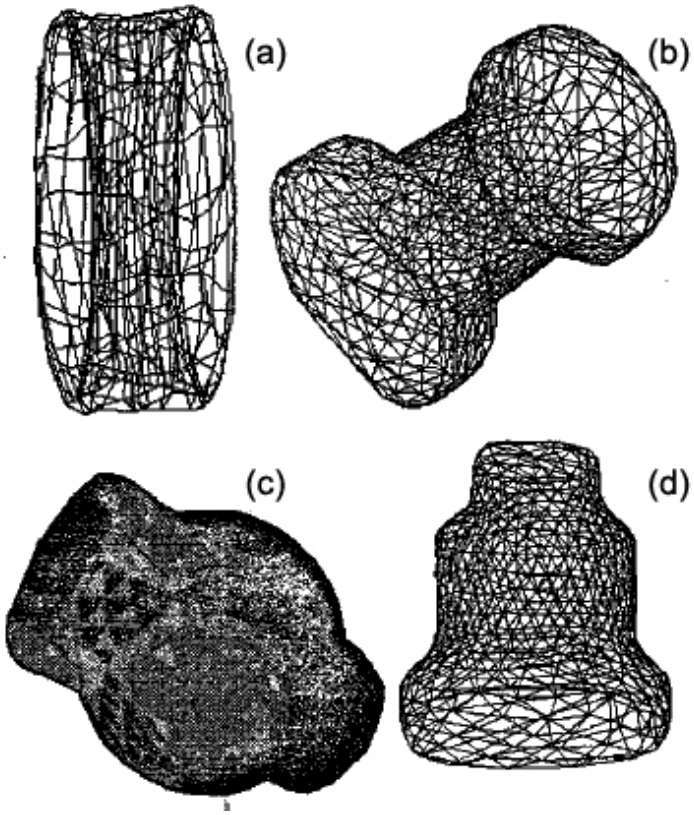


Figura 28: Resultados da reconstrução utilizando o SOM: (a) grade toroidal, (b), (c) e (d) grades esféricas. Fonte: (BOUDJEMAÏ, ENBERG e POSTAIRE, 2003).

LIOU e KUO (2005) obtiveram bons resultados reconstruindo superfícies de objetos que são variedades de gênero-zero[5] (homeomorfas a uma esfera), estendendo uma variante do SOM, o *Conformal Self-Organizing Map* (CSM), e criando o *Conformal Spherical* SOM (CSSM). O processo de criação da esfera é o mesmo das outras abordagens citadas anteriormente. Através da poligonalização do icosaedro é obtido um domo geodésico, que então é modelado pelo algoritmo de aprendizagem. A principal vantagem da abordagem de LIOU e KUO (2005) está justamente nas características do algoritmo CSM. Alguns exemplos dos resultados obtidos podem ser vistos na Figura 29. Os resultados foram comparados com os obtidos por YU (1999) e se mostraram superiores.

[5] O gênero de uma superfície orientável representa o número de "túneis" existem nesta. Por exemplo, um toróide possui gênero 1, já a esfera possui gênero 0. Superfícies orientáveis são aquelas onde é possível distinguir dois lados, como por exemplo, a esfera e o toróide, quando isso não é possível a superfície é dita não orientável, por exemplo a fita de Möbius (MAA, 2006).

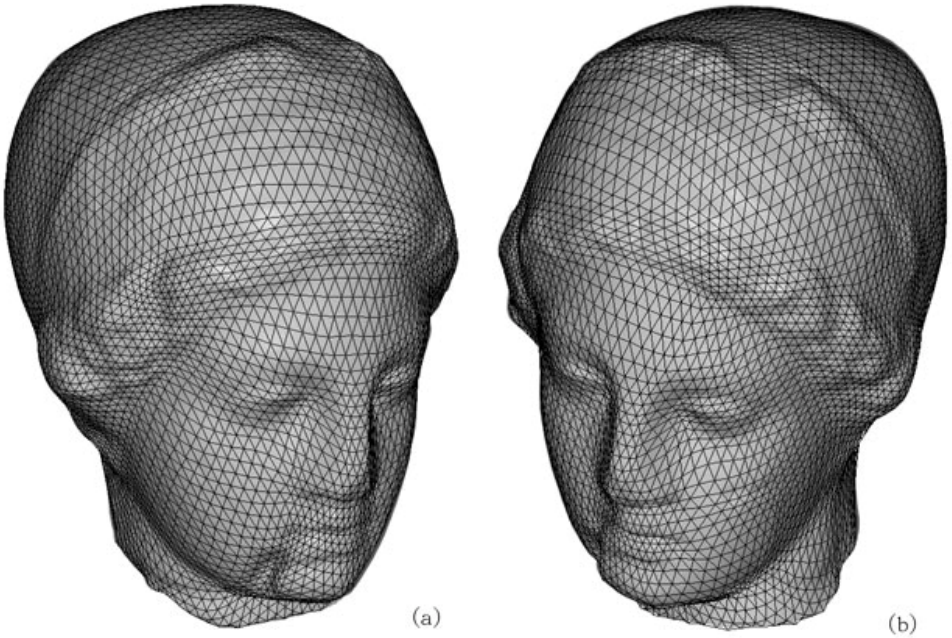


Figura 29: Reconstrução de um modelo da cabeça de Vênus, realizada por Conformal Spherical Self-Organizing Map (CSSM). Vista por dois ângulos diferentes (a) e (b). Fonte: (LIOU e KUO, 2005).

5.3 Considerações Finais

Foram apresentados neste capítulo alguns trabalhos relevantes, desenvolvidos anteriormente sobre reconstrução de superfícies utilizando redes neurais, principalmente o modelo SOM. O próximo capítulo trata da apresentação do sistema desenvolvido para a dissertação que consiste no uso da Rede Neural Auto-Organizável CGS-Modificada combinada com a técnica de troca de arestas.

6 PROPOSTA DE TRABALHO

O trabalho desenvolvido para a presente dissertação consiste na proposta e implementação de uma algoritmo especifico para o problema de reconstrução de superfícies a partir de nuvens de pontos, baseado no algoritmo GCS, apresentado na seção 4.3, combinado com a técnica de troca de arestas para evitar distorções que possam ocorrer em estruturas côncavas presentes na superfície. O capitulo prossegue com a descrição do algoritmo.

6.1 Introdução

Baseado no modelo de Rede Neural Auto-Organizável *Growing Cell Structures*, originalmente proposto por Fristzke (FRISTZKE, 1993) como uma extensão incremental do *Self-Organizing Map* (KOHONEN, 1982; 1990), o algoritmo proposto possibilita realizar a reconstrução de superfícies a partir de uma nuvem de pontos com um custo computacional independente da quantidade de pontos existentes. O custo computacional do algoritmo, assim como da maioria dos algoritmos de reconstrução baseados em redes neurais é proporcional à densidade final da malha, que é definida pelo usuário. Portanto, se o usuário necessitar de uma malha de polígonos mais densa, o algoritmo necessita de um número maior de iterações, enquanto que se uma malha com densidade de polígonos inferior satisfizer as necessidades do usuário, um número menor de iterações será suficiente, sem a necessidade de nenhum tipo de pré-processamento sobre a nuvem de pontos.

O primeiro artigo a descrever a utilização do GCS para a reconstrução de superfícies é devido a IVRISSIMTZIS, JEONG & SEIDEL (2003a). Neste artigo o algoritmo é denominado *Growing Cell for Surface Reconstruction*, e apenas os módulos de aprendizagem da geometria e mudança na conectividade estão presentes. Por este motivo, o algoritmo proposto é aplicável somente a superfícies 2-variedade fechadas (sem bordas) de gênero topológico zero. No artigo seguinte (IVRISSIMTZIS, JEONG & SEIDEL, 2003b), com a inclusão do módulo de aprendizagem da topologia, o algoritmo foi denominado *Neural Meshes* e passou a lidar com superfícies 2-variedade fechadas e abertas, com gênero topológico igual a zero ou maiores, lembrando que o gênero topológico de uma superfície pode ser descrito informalmente como o número de túneis presentes na superfície. A parte do sistema desenvolvido para a dissertação relativo ao uso do GCS é semelhante ao *Neural Meshes*, denominado GCS-M (GCS-Modificado).

A Tabela 1, a seguir, apresenta alguns símbolos utilizados neste capítulo. A malha poligonal 3D é equivalente a uma Rede Neural Auto Organizável, como foi mostrado no capítulo 5. Por isso a equivalência com a simbologia utilizada nas descrições anteriores referentes aos modelos de redes neurais também é incluída na tabela quando necessário.

Tabela 1: Simbologia utilizada neste capítulo

Símbolo	**Sistema Desenvolvido (GCS-M)**	**Equivalente no SOM**
M	Malha poligonal.	Rede neural
v	Vértice.	Neurônio
v_w	Vértice vencedor	Neurônio vencedor

α_w	Parâmetro de aprendizagem do vértice vencedor.	Taxa de aprendizagem do neurônio vencedor
α_n	Parâmetro de aprendizagem dos vértices vizinhos de v_w .	Taxa de aprendizagem dos neurônios vizinhos do neurônio vencedor
τ	Contador de sinal de cada vértice	Equivale ao Erro local dos neurônios, no GCS.
α	Constante de ponderação do contador de sinal	----------
C_{vs}	Intervalo entre as operações de subdivisão de vértices	Intervalo entre cada inserção de um novo neurônio.
C_{ec}	Constante utilizada para calcular o intervalo entre as remoções de arestas	----------

6.2 Algoritmo de Aprendizagem do Sistema Desenvolvido

A seguir é apresentada uma descrição dos principais passos do algoritmo implementado. O algoritmo pode ser dividido em três módulos, para melhor compreensão: módulo de aprendizagem da geometria, mudança da conectividade e aprendizagem da topologia.

Aprendizagem da Geometria

- Selecionar aleatoriamente um ponto $\mathbf{x} \in X$.
- Determinar o vértice vencedor v_w, o vértice mais próximo a $\mathbf{x}$ considerando a distância Euclidiana (Equação (6.1)):

$$\|v_w - \mathbf{x}\| = \min_{i \in M} \|v_i - \mathbf{x}\|. \qquad (6.1)$$

- Atualizar a posição do vértice vencedor v_w, movendo-o na direção de $\mathbf{x}$, de acordo com a Equação (6.2):

$$v'_w = (1-\alpha_w)v_w + \alpha_w \mathbf{x}, \tag{6.2}$$

onde α_w é a taxa de aprendizagem do nodo vencedor, que da mesma forma que no GCS é constante durante todo o processo de aprendizagem.

- Atualizar as posições do conjunto de vizinhos diretos v_n do nodo vencedor v_w. No GCS e na maioria das redes neurais auto-organizáveis, como o SOM, a atualização dos nodos vizinhos é efetuada por meio da combinação linear da posição dos vértices v_n com o ponto $\mathbf{x}$. Neste caso, é aplicado um processo de suavização Laplaciana, na direção tangencial de cada vértice v_n. Para isso, primeiro calcula-se o Laplaciano L de todos os vértices v_n pela Equação (6.3):

$$L(v_n) = \frac{1}{valencia(v_n)} \sum_{v_k \in 1-ring(v_n)} (v_k - v_n), \tag{6.3}$$

onde $1-ring(v_n)$ é o conjunto de todos os nodos vizinhos de v_n ordenados.

Determina-se então a componente tangencial de L pela Equação (6.4):

$$L_t(v_n) = L(v_n) - (L(v_n)\cdot \mathbf{n})\mathbf{n}, \tag{6.4}$$

onde $\mathbf{n}$ é o vetor normal do vértice representado pelo vértice v_i.

Os vetores de pesos v_n dos nodos v_n são então atualizados por uma fração de L_t, determinado pela constante de aprendizagem α_n. A atualização é definida pela Equação (6.5):

$$v'_n = v_n + \alpha_n L_t, \quad \forall v_n \in 1-ring(v_w), \tag{6.5}$$

- Incrementar o contador de sinal do nodo vencedor pela Equação (6.6):

$$\tau_{v_w} = \tau_{v_w} + 1 \tag{6.6}$$

- Decrementar os contadores de sinal de todos os demais nodos, seguindo a Equação (6.7):

$$\tau_{v_i} = \alpha \tau_{v_i}, \tag{6.7}$$

onde α é uma constante de ponderação. Nota-se que a Equação (6.7) faz com que os nodos que foram vencedores mais recentemente têm seus contadores de sinais mais ativos.

Mudança da conectividade

- Seja C_{vs} uma constante inteira. A cada C_{vs} iterações do módulo de aprendizagem da geometria, selecionar o vértice v_t, com o maior contador de sinal e efetuar uma operação de subdivisão de vértice. Uma vez que o contador de sinal de cada nodo mede a freqüência de ativação daquele nodo, os nodos com os maiores valores do contador de sinal, estão próximos de uma região no espaço de entrada com alta concentração de pontos, portanto a operação de subdivisão de vértices cria novos vértices nesta região da malha. A operação de subdivisão de vértices é mais bem detalhada na próxima seção.
- Seja C_{ec} uma constante e *n* o número de vértices presentes na malha na iteração atual. A cada $C_{ec} \cdot n$ iterações do módulo de aprendizagem da geometria os vértices inativos (que não participam do processo competitivo) devem ser removidos. São considerados vértices inativos todos aqueles que não foram eleitos vencedores nas ultimas $C_{ec} \cdot n$. A remoção dos nodos inativos é realizada por operações de remoção de arestas. Na seção 7.5.2 a operação de remoção de arestas é explicada em mais detalhes.

As operações de subdivisão de vértices e remoção de arestas são complementares, como pode ser mais claramente observado na Figura 30. Quando é necessário aumentar a densidade da malha subdivide-se o vértice mais ativo, e quando é

necessário eliminar vértices inativos aplica-se a operação de remoção de arestas em uma de suas arestas.

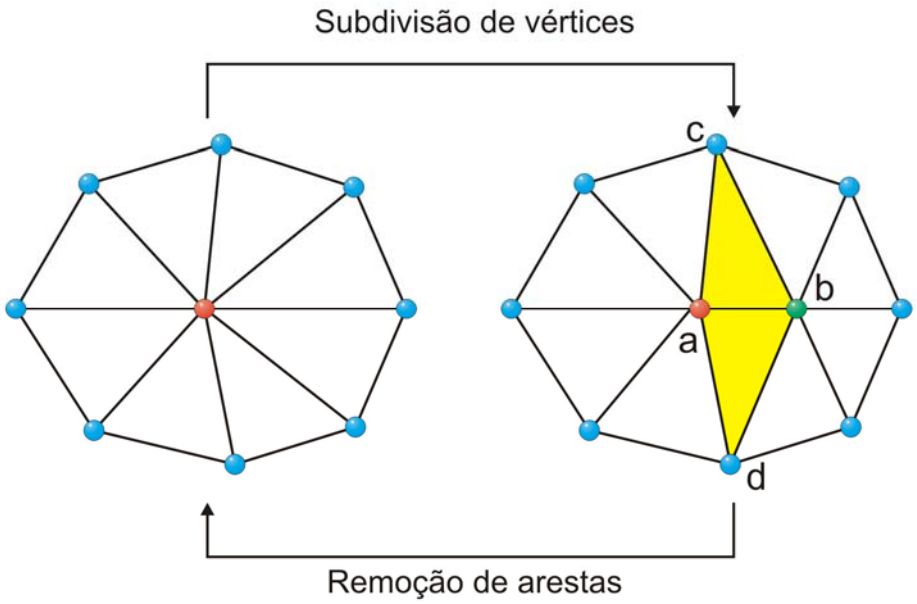


Figura 30: As operações complementares de subdivisão de vértices e remoção de arestas.

Aprendizagem da topologia

- A área dos triângulos é inversamente proporcional à densidade dos pontos naquela região. Partindo deste principio, é possível detectar regiões de borda na nuvem de pontos e criar as bordas de maneira correta na superfície poligonal M através de operações de remoção de triângulos. A operação de remoção de triângulos é chamada imediatamente após cada chamada da operação de remoção de arestas.
- Unir as bordas da malha que estão relativamente próximas, criando túneis na malha, e alterando seu gênero topológico. Para determinar a proximidade entre duas arestas é utilizada a distância de Hausdorff (ROGERS, 1999) entre as bordas, aproximada pela distância de Hausdorff entre o conjunto de vértices das bordas. A distância de Hausdorff entre duas superfícies é definida como:

$$d_H(S,S') = \max_{\mathbf{x}\in S}\left(\min_{\mathbf{x}'\in S'}\|\mathbf{x}-\mathbf{x}'\|\right) \qquad (6.8)$$

6.3 Considerações Finais

Neste capítulo foi descrito o algoritmo modificado do GCS, denotado GCS-Modificado (GCS-M) proposto para implementação. As operações de adaptação da posição dos vértices, que permitem ao algoritmo aprender a geometria do objeto a partir da nuvem de pontos foram detalhadas, bem como as operações sobre a malha de que permitem aumentar resolução da malha de forma correta. O próximo capítulo apresenta as metodologias, materiais e os detalhes da implementação realizada.

7 METODOLOGIA E IMPLEMENTAÇÃO

Este trabalho consiste na descrição da metodologia de implementação do algoritmo GCS-M, descrito no capítulo anterior. A seção 7.1 apresenta os algoritmos implementados e as nuvens de pontos utilizadas. A seção 7.2 descreve os materiais e ferramentas empregadas na elaboração do trabalho. A seção 7.3 apresenta a biblioteca CGAL e superfícies poliédricas. A seção 7.4 apresenta detalhes da implementação do algoritmo. A seção 7.5 detalha a implementação das operações sobre a malha. A seção 7.6 detalha como é combinada a técnica de troca de arestas. A seção 7.7 define as métricas para avaliação da qualidade do algoritmo. A seção 7.8 traz as considerações finais do capítulo.

7.1 Algoritmos Implementados e as Nuvens de Pontos Utilizadas

A implementação do algoritmo GCS-M realizada neste trabalho inclui os módulos de aprendizagem da geometria e modificação da conectividade, apresentados no capítulo anterior.

As nuvens de pontos utilizadas foram o bem conhecido Coelho de Stanford (STANFORD, 2007) e a Cabeça do Manequim[6] (HOPPE, 1994), mostrados na Figura 31 e Figura 32, respectivamente.

[6] http://research.microsoft.com/~hoppe/

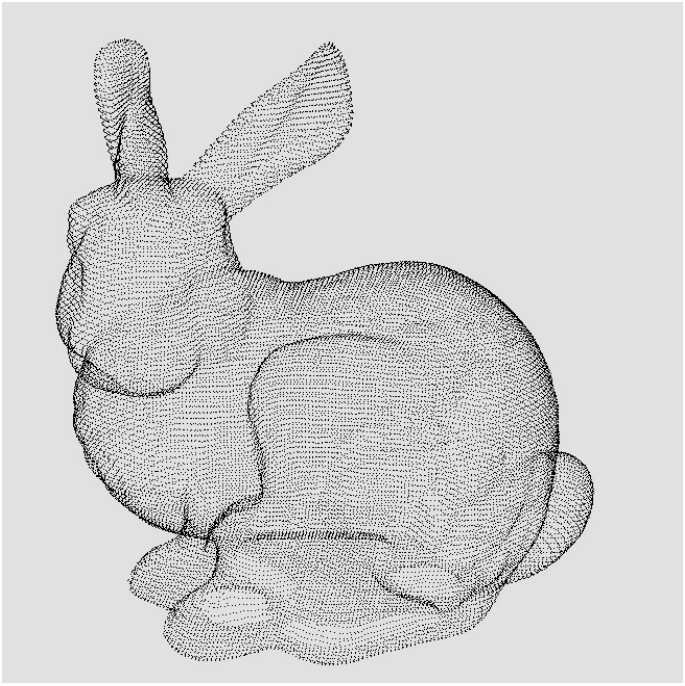


Figura 31: Nuvem de pontos do Coelho de Stanford, utilizado para validação do algoritmo implementado. Possui 35.947 pontos.

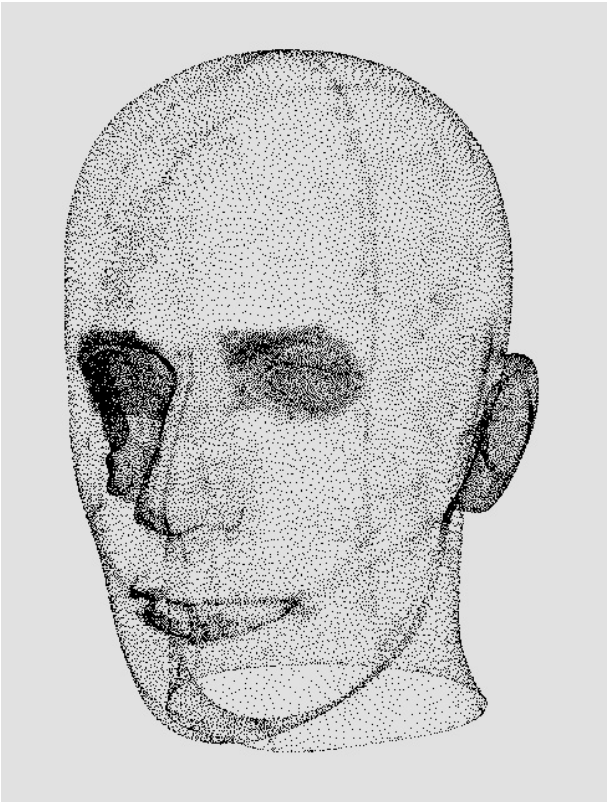


Figura 32: Nuvem de pontos obtida da cabeça de um manequim, utilizado para validação do algoritmo implementado. Possui 41.117 pontos.

Foi implementada uma forma eficiente de incluir a operação de troca de arestas durante o processo de aprendizagem, melhorando a qualidade da malha e a convergência do algoritmo.

7.2 Materiais e Ferramentas de Desenvolvimento

Os algoritmos foram implementados utilizando a linguagem C++ (STROUSTRUP, 2000). Recursos da linguagem C++ como *templates* e as estruturas de dados genéricas da biblioteca STL (STEPANOV & LEE, 1995) foram intensamente empregados. O compilador utilizado foi o MINGW / GCC 3.4.2[7], em conjunto com o editor de códigos Bloodshed Dev C++ 5.0 beta 9.2 (4.9.9.2)[8], ambos softwares de fonte aberto (*open source*).

Todos os softwares funcionando sobre o sistema operacional *MS Windows XP Professional x64*. A configuração do PC utilizado na geração dos resultados microprocessador AMD Athlon 64 X2, 2.2 GHz e 2 Gb de memória RAM.

Para armazenar e manipular adequadamente a malha poligonal 3D foi utilizada a implementação de uma superfície poliédrica (conjunto de polígonos que representa o limite entre o interior e o exterior de um sólido poliédrico) pela classe genérica (parametrizável) *CGAL::Polyhedron_3<Traits>* (KETTNER, 1999) disponível na biblioteca CGAL (*Computational Geometry Algorithms Library*). A classe *CGAL::Polyhedron_3<Traits>* implementa uma superfície poliédrica em três dimensões, bastante robusta, eficiente e extensível. Mais detalhes sobre a biblioteca CGAL e a superfície poliédrica *CGAL::Polyhedron_3<Traits>* são mostrados na seção 7.3.

---

[7] http://www.mingw.org/
[8] http://www.bloodshed.net/devcpp.html

As nuvens de pontos são armazenadas em arquivos texto tipo ASCII. A estrutura desses arquivos é bastante simples. A primeira linha contém o número total de pontos, em seguida cada linha armazena as coordenadas *x*, *y* e *z* de cada ponto. Os valores são separados por espaços simples. Por convenção os arquivos contendo as nuvens de pontos possuem extensão PTS. Na Figura 33, é apresentado um exemplo da estrutura de um arquivo de nuvem de pontos:

```
35947
-0.0378297 0.12794 0.00447467
-0.0447794 0.128887 0.00190497
-0.0680095 0.151244 0.0371953
-0.00228741 0.13015 0.0232201
-0.0226054 0.126675 0.00715587
-0.0251078 0.125921 0.00624226
...
-0.0677458 0.149084 -0.0355035
-0.0221568 0.157426 -0.00647102
-0.0704544 0.150585 -0.0434585
-0.0310262 0.153728 -0.00354608
-0.0400442 0.15362 -0.00816685
```

Figura 33: Estrutura do arquivo de nuvem de pontos

As malhas poligonais geradas pelo algoritmo são gravadas em arquivos texto no formato Geomview OFF (*Object File Format*). Um exemplo da estrutura de um arquivo tipo OFF básico é mostrado na Figura 34:

Um arquivo no formato OFF inicia com a palavra OFF na primeira linha, a segunda apresenta três valores inteiros: número de vértices, número de faces e número de arestas, podendo este último ser 0. As próximas linhas armazenam triplas que correspondem as coordenadas dos vértices da malha no espaço tridimensional. Ao final das linhas contendo as informações dos vértices, começam as linhas contendo informações sobre as faces, o primeiro valor de cada uma dessas linhas, representa o número de lados da face, seguindo dos índices dos vértices que a compõem. Arquivos OFF podem ser visualizados pelos softwares

GEOMVIEW[9] em sistemas Linux e em sistemas Windows pelos softwares Javaview[10] e Meshlab[11].

```
OFF
12 20 0
0.850651 0. -0.525731
0.850651 0. 0.525731
0.525731 -0.850651 0.
0. -0.525731 -0.850651
0. 0.525731 -0.850651
0.525731 0.850651 0.
0. 0.525731 0.850651
0. -0.525731 0.850651
-0.850651 0. 0.525731
-0.525731 -0.850651 0.
-0.850651 0. -0.525731
-0.525731 0.850651 0.
3 0 4 5
3 0 2 3
3 0 3 4
3 2 7 9
3 2 9 3
3 3 10 4
3 4 10 11
3 3 9 10
3 8 11 10
3 8 10 9
3 7 8 9
3 6 11 8
3 6 8 7
3 4 11 5
3 5 11 6
3 1 5 6
3 1 7 2
3 1 6 7
3 0 1 2
3 0 5 1
```

Vértices

Faces

Figura 34: Exemplo da estrutura básica de um arquivo tipo OFF.

[9] www.geomview.org
[10] http://www.javaview.de/
[11] http://meshlab.sourceforge.net/

7.3 A Biblioteca CGAL

A biblioteca CGAL (CGAL, 2007) contém primitivas, estruturas de dados e algoritmos para geometria computacional (GIEZEMAN, VELTKAMP & WESSELINK, 1999).

A biblioteca CGAL é dividida em várias partes. A parte básica, ou *kernel*, consiste em primitivas, objetos geométricos de tamanho constante (pontos, linhas e esferas), assim como predicados (testes de orientação de pontos, testes de intersecção, etc.). A parte seguinte da biblioteca é composta por vários algoritmos e estruturas de dados comuns em geometria computacional, como fecho convexo, triangulações e outros. A última parte da biblioteca inclui superfícies poliedrais, funções de entrada e saída, visualização, etc.

7.3.1 Superfícies Poliédricas e a Estrutura de Dados *Halfedge*

Uma Superfície poliédrica *CGAL::Polyhedron_3<Traits>* em três dimensões consiste de vértices *V*, arestas *E*, faces *F* e na relação de incidência entre eles. Cada aresta é representada por duas semi-arestas (*halfedges*) com orientações opostas. As incidências armazenadas em uma semi-aresta são ilustradas na Figura 35. Cada aresta que compõe a estrutura de dados *Halfedge* é decomposta em duas semi-arestas. Na Figura 35 as semi-arestas *halfedge* e *opposite halfedge* compõem uma única aresta. As tabelas abaixo da figura mostram respectivamente as referências mantidas por cada item (Vértice, Semi-aresta e Face) que definem as relações de incidência entre eles (KETTNER, 1999).

Cada vértice representa pontos no espaço tridimensional. Arestas são segmentos de reta que conectam dois vértices. Faces são polígonos planares sem buracos. As faces são definidas pela seqüência circular de semi-arestas ao longo de sua borda. Por

convenção, as semi-arestas são orientadas no sentido anti-horário ao redor das Faces (Figura 36 (a)), quando estas são vistas pelo lado de fora do poliedro. Isso implica que as semi-arestas são orientadas no sentido horário ao redor dos vértices (Figura 36(b)).

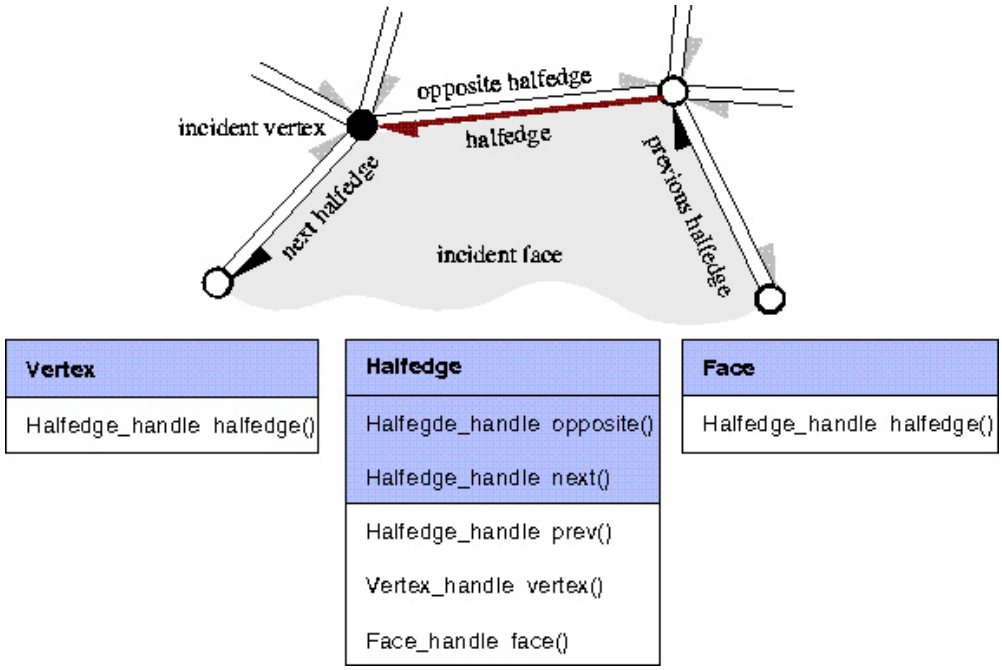


Figura 35: Relação de incidências de uma semi-aresta na estrutura de dados Halfedge. Fonte: (KETTNER, 1999).

A superfície poliédrica *CGAL::Polihedron_3<Traits>* é implementada utilizando a estrutura de dados *Halfedge CGAL::HalfedgeDS<Traits, Items, Alloc>* da biblioteca CGAL. A estrutura de dados *Halfedge* possibilita representar superfícies poligonais 2-*manifold* orientáveis inseridas em dimensões arbitrarias. A estrutura *Halfedge* é capaz de armazenar e manipular informações de incidência de vértices, bordas e faces. Cada aresta é decomposta em duas semi-arestas (*halfedge*) com orientações opostas. Cada semi-aresta armazena referências para um vértice incidente e uma face incidente. Cada vértice e cada face armazenam uma referência para a semi-aresta incidente. A estrutura *Halfedge* e a relação de incidências entre seus elementos podem ser vistas na Figura 35.

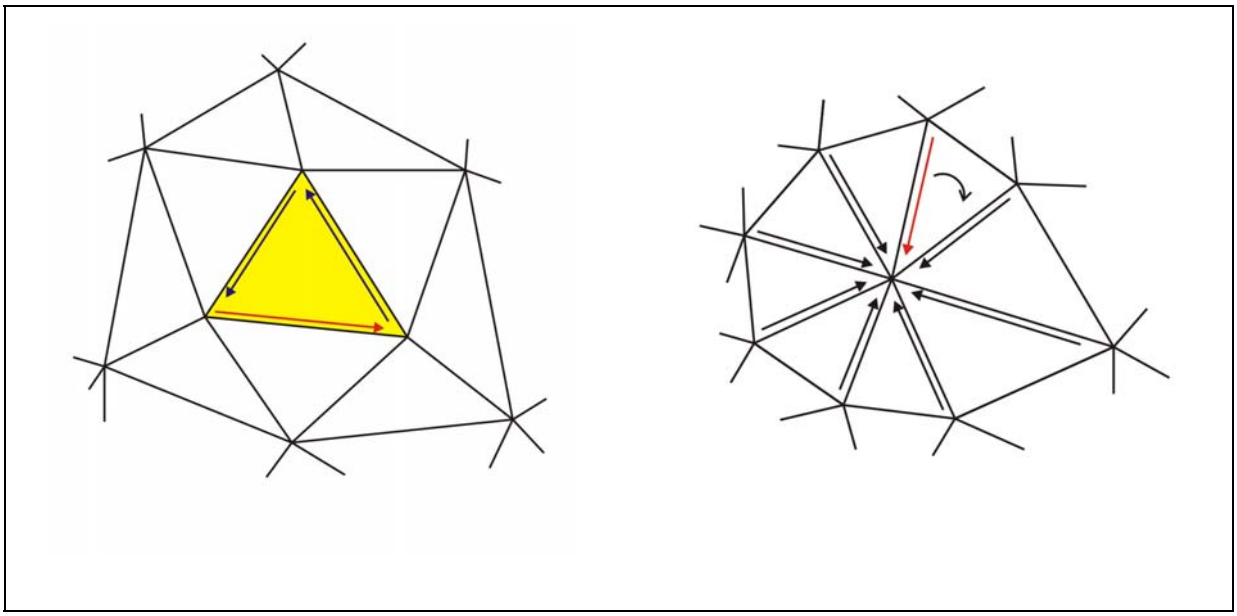


Figura 36: (a) Orientação das semi-arestas que compõem uma face na estrutura *Halfedge* (sentido anti-horário). (b) Orientação das semi-arestas incidentes em um vértice (sentido horário).

A estrutura de dados *Halfedge* proporcionada pela biblioteca CGAL é também conhecida na literatura como *FE-structure* (WEILER, 1985), *halfedge* (MÄNTYLA, 1988), lista de arestas duplamente conectadas (DCEL) (BERG, 1997).

A Figura 37 ilustra o projeto de software da classe *CGAL::Polyhedron_3<Traits>*. A classe *CGAL::Polyhderon_3<Traits>* é construída sobre a classe *CGAL::Halfedge_DS< Traits, Items, Alloc >* que implementa a estrutura de dados *Halfedge*, e é responsável pelo gerenciamento correto das relações de incidência em uma superfície poliédrica 2-*manifold* de dimensão arbitraria. As informações que a classe *CGAL::Halfedge_DS<Traits, Items, Alloc>* é responsável por gerenciar a camada inferior chamada *items* (vértices, semi-arestas e faces) (KETTNER, 1999).

A camada *items* fornece espaço para o armazenamento das informações atuais, i.e., variáveis membro e funções de acesso de *Vertex*, *Halfedge* e *Face*. *Halfedge* fornece uma referência para a semi-aresta seguinte, e para a semi-aresta oposta. Opcionalmente pode fornecer uma referência para a semi-aresta anterior, para o vértice incidente e para a face

incidente. *Vertex* e *Face* podem estar vazias e opcionalmente providenciar uma referência para a semi-aresta incidente.

A camada *Halfedge_datra_structure* é responsável pela organização dos *items*. Geralmente é implementado usando uma lista duplamente encadeada, porém uma implementação usando um vetor também é fornecida. Nessa camada são definidos manipuladores e iteradores para os *items*. Estes tipos são fornecidos para a declaração dos próprios *items*, para referenciá-los e iterar entre eles.

A camada *Polyhedron* que define a superfície de um sólido poliédrico genérico. Esta camada restringe os vértices ao espaço dimensional 3D, proporciona maior facilidade de uso e define os circuladores. Os circuladores ao redor dos vértices permitem percorrer de forma ordenada as arestas ao redor dele, enquanto os circuladores ao redor das faces permitem percorrer as semi-arestas que formam a face.

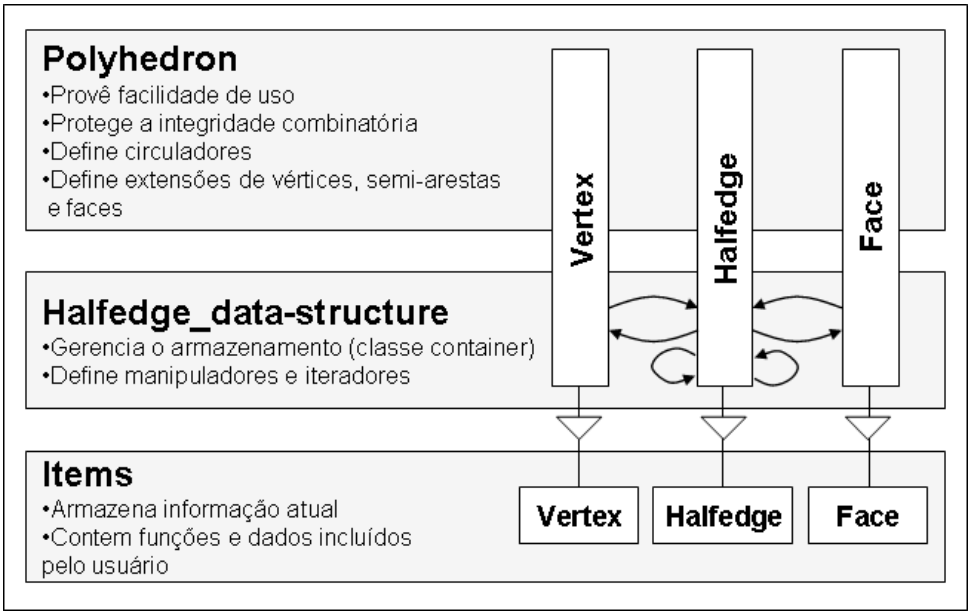


Figura 37: Projeto da classe *CGAL::Polyhderon_3<Traits>*. Adaptado de (KETTNER, 1999).

7.4 Detalhes da Implementação

A implementação realizada consiste em duas classes principais: *GSM_M* e *Point_cloud*. A classe *GCS_M* é composta basicamente de um objeto *CGAL::Polyhedron_3<Traits>* e uma lista duplamente encadeada de objetos do tipo *Neurônio*. Foi utilizada a implementação de lista presente na STL, *std::list<T, Alloc>*. Foi escolhida a lista encadeada pela necessidade de retiradas rápidas de elementos no meio da estrutura, durante as operações de remoção de arestas.

Cada objeto *Neurônio* possui uma referência a um vértice da superfície poliédrica e mais duas variáveis que armazenam, respectivamente, o valor do contador de sinais, utilizado pela operação de subdivisão de vértices e o número de vezes que o Neurônio é vencedor nas últimas $C_{ec} \cdot n$ iterações, utilizado na operação de remoção de arestas.

Todas as funções que envolvem buscas, incluído: busca pelo vértice vencedor, busca pelo maior contador de sinal, busca pelos vértices inativos, são executadas nesta lista. A idéia de uma lista externa a estrutura de dados da malha poligonal foi utilizada em (SALEEM, 2004). A Figura 39 ilustra a utilização da lista associada à malha poligonal.

A classe *Point_cloud* é implementada por um vetor contendo todos os pontos pertencentes à nuvem. Foi utilizada a classe *std::vector<T, Alloc>* da biblioteca STL. A escolha de implementar a nuvem de pontos como um vetor foi motivado pela possibilidade de acesso direto a qualquer elemento pelo seu índice, o que permitiu implementar de maneira fácil e eficiente uma função que, a cada iteração seleciona um ponto aleatoriamente para ser apresentado ao algoritmo.

Por meio da visualização da estrutura em camadas da aplicação desenvolvida, apresentada na Figura 38 pode-se observar que toda aplicação foi toda construída sobre a linguagem de programação C++. Foram utilizadas as bibliotecas STL e CGAL, mais

especificamente as classes std::vector<T, Alloc> e std::list<T, Alloc> da biblioteca STL para a implementação das classes Point_cloud e GCS_M, respectivamente. Com relação à biblioteca CGAL, a classe *CGAL::Polihedron_3<Traits>* foi largamente utilizada, uma vez que permitiu o armazenamento e manipulação eficiente da malha poligonal 3D, e em conjunto com a classe *std::list<T, Alloc>* são a base da classe principal da aplicação, a classe *GCS_M*. A Figura 39 ilustra a estrutura desta classe mais adequadamente.

A estrutura básica da classe *GCS_M* é composta por uma lista duplamente encadeada (*std::list<T, Alloc>*) de *Neurônios* ou nodos, que armazenam, cada um, uma referência para um dos vértices da malha 3D, implementada pela classe *CGAL::Polyhedron_3<Traits>* da biblioteca CGAL. Todos os neurônios permitem armazenar informações como os valores do contador de sinal do vértice que ele referencia e o número de vezes que o mesmo foi vencedor. Todas as operações de busca, como vértice vencedor, maior contador de sinal e vértices inativos são executados na lista duplamente encadeada.

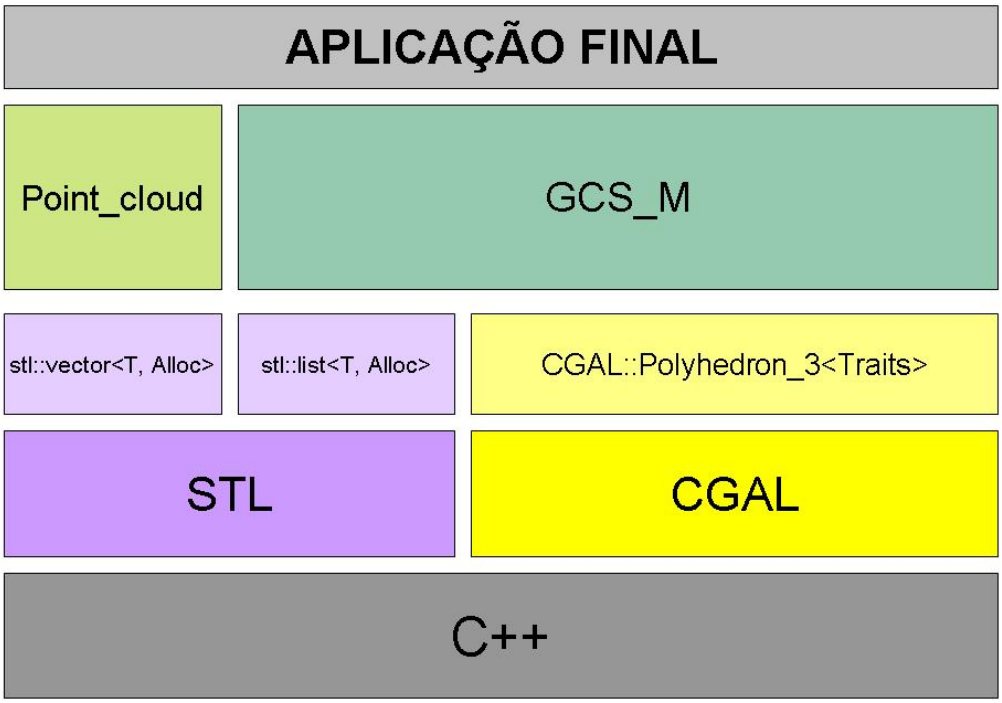


Figura 38: Estrutura de camadas da implementação do algoritmo

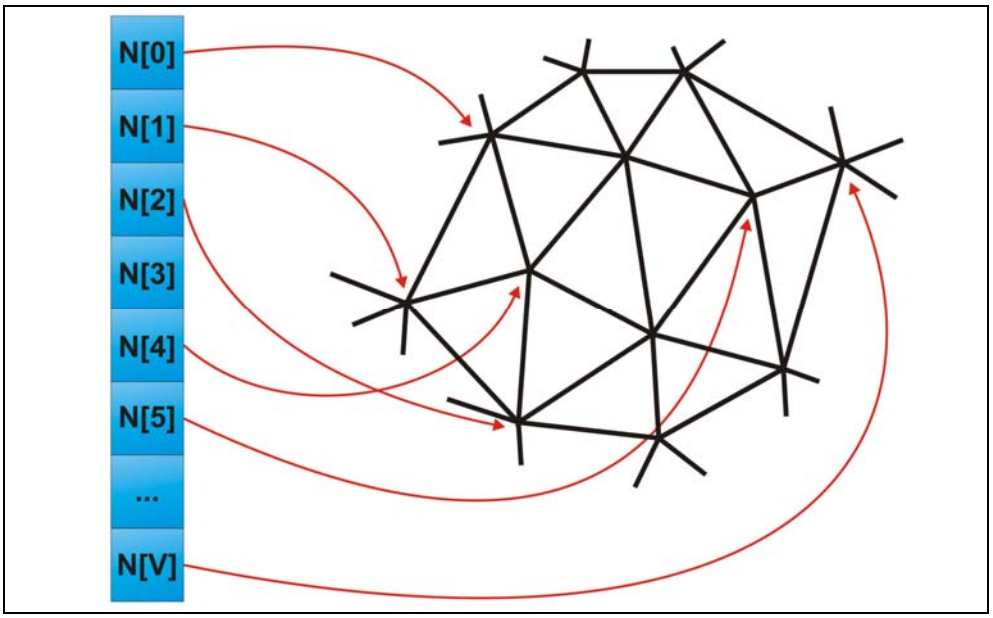


Figura 39: Detalhes da estrutura da classe GCS-M apresentada na Figura 38.

7.5 Implementação das Operações Sobre a Malha

As operações sobre a malha são de crucial importância para o *GCS_M*, uma vez que permitem que a malha "cresça" até atingir o nível de resolução desejado pelo usuário, permite também a eliminação de vértices inativos. O processo de aprendizagem da topologia ocorre por meio de operações sobre a malha, no caso a remoção de triângulos e fusão de bordas. Esta seção descreve as implementações das operações de subdivisão de vértices (seção 7.5.1) e remoção de arestas (seção 7.5.2).

7.5.1 Subdivisão de vértices

A operação de subdivisão de vértices tem como função gerar um novo nodo na malha poligonal, aumentando a densidade local da mesma, ao mesmo tempo em que preserva a

topologia da malha. Se antes da operação de subdivisão de vértices, a malha 3D for, por exemplo, um 2-*variedade* fechado de gênero zero, ela manterá estas características topológicas após a operação.

No GCS a operação responsável pela inserção de novos nodos é a subdivisão de arestas, porém, a subdivisão de vértices é mais adequada ao problema da reconstrução de superfícies, uma vez que distribui a valência do vértice original com o novo vértice (Figura 40).

A seguir é mostrado o algoritmo utilizado para a operação de subdivisão de vértices:

Subdivisão de vértices:

- Localizar o vértice v_t com o maior valor do contador de sinal τ_t, esse é o vértice que será subdividido.
- Localizar entre as arestas incidentes em v_t, a aresta mais longa e_t. O vértice da aresta e_t oposto à v_t é denotado v_o (Figura 40 (a)).
- Partindo da aresta e_t, percorrem-se as demais arestas incidentes em v_t, ao mesmo tempo no sentido horário e anti-horário, até encontrar duas delas que melhor dividem o conjunto de arestas aproximadamente na metade, são denotadas e_l e e_r, respectivamente (Figura 40 (b)).
- O novo vértice v_c é criado no meio da aresta e_t. E as arestas são redistribuídas entre os dois vértices (Figura 40 (c)).

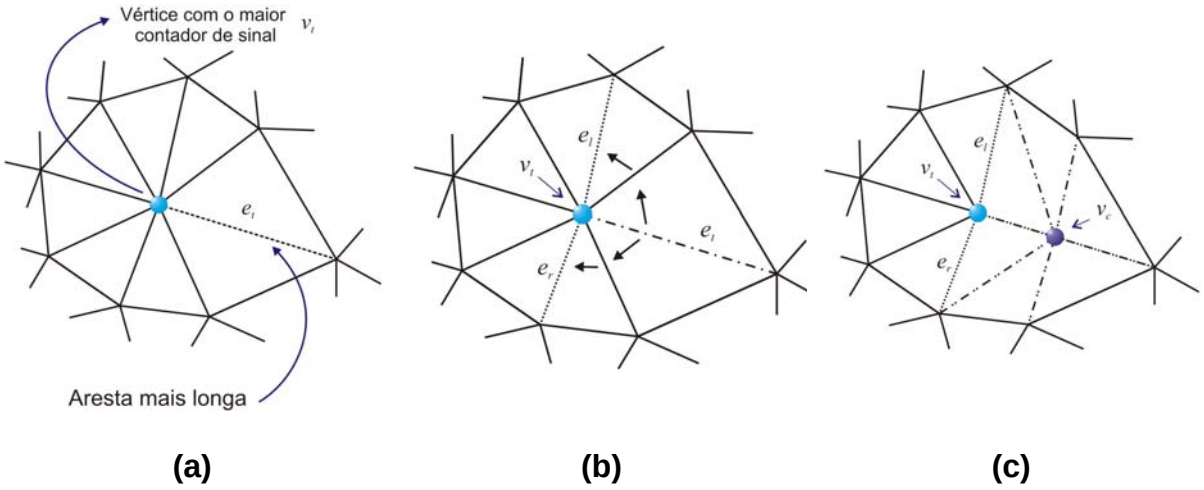


Figura 40: Operação de subdivisão de vértices. (a) Aresta mais longa adjacente a vértice alvo. (b) Localização das arestas que dividem a estrela de arestas ao meio. (c) Inserção do novo vértice e redefinição das arestas e faces.

7.5.2 Remoção de Arestas

A cada $C_{ec} \cdot n$ iterações os vértices inativos são eliminados de M por uma operação de remoção de arestas. O motivo da escolha do intervalo $C_{ec} \cdot n$ é permitir que cada um dos n vértices da malha tenham C_{ec} chances de serem vencedores, caso não sejam, é bastante provável que eles estejam localizados em regiões longe da nuvem de pontos aumentando o erro médio e, possivelmente cobrindo regiões côncavas da superfície original. Por isso a operação de remoção de arestas é de suma importância na qualidade final do algoritmo.

O algoritmo para efetuar a remoção de arestas é mostrado a seguir:

Remoção de arestas

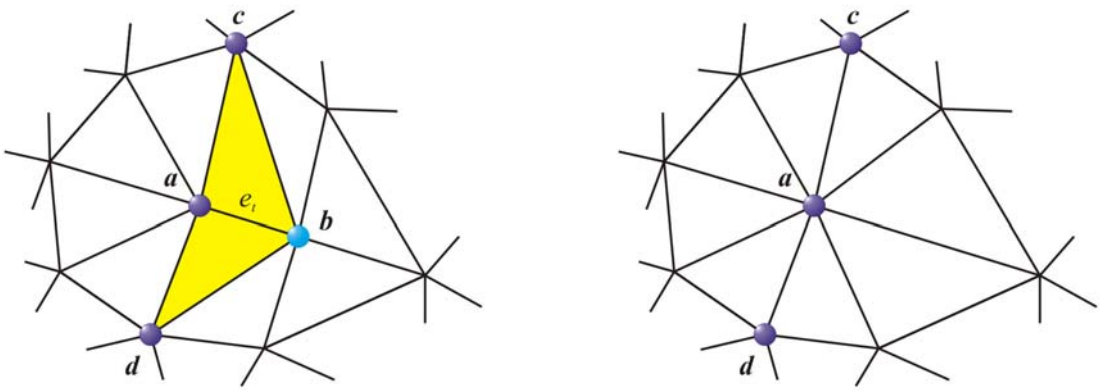
- Ao final de cada $C_{ec} \cdot n$ iterações, localizar todos os vértices que não foram declarados vencedores, esses vértices são denotados por *b* na Figura 41.

- Para cada v_t, deve-se localizar entre as arestas incidentes, a aresta com o menor erro de regularidade *E* por meio da Equação (7.1):

$$E = \frac{1}{3}\sqrt{(a+b-10)^2 + (c-7)^2 + (d-7)^2} \qquad (7.1)$$

onde *a*, *b* e *c*, são mostradas na Figura 41.

- A aresta que possui o menor valor de regularidade *E*, denotada por e_t, é eliminada, assim como os dois polígonos nela incidentes (Figura 41).



(a) **(b)**

Figura 41: Operação de remoção de arestas. (a) A aresta e_t *será removida da mesma forma que os dois triângulos adjacentes a ela. (b) Malha após a remoção da aresta, nota-se que o vértice inativo b também foi removido.*

É importante realizar a verificação da validade da operação de remoção de arestas, para que a malha resultante continue sendo uma variedade, e mantenha as características e tipo topológico inalterados. Nota-se na Figura 41 após a operação de remoção de arestas os vértices *c* e *d* terão valência $valência(c)-1$ e $valência(d)-1$, o vértice *b* desaparece e a valência do vértice *a* passa a ser $valência(a)+valência(b)-4$. Como é aconselhável não manter vértices com valência menor ou igual a três, deve-se evitar remoção

de arestas em que $valência(c) <= 4$, $valência(d) <= 4$ e $valência(a) + valencia(b) < 8$, para evitar inconsistência na malha e/ou geração de malhas não-*variedade*.

7.6 Troca de Arestas

A operação de troca de arestas é uma operação fundamental no algoritmo proposto por YU (YU, 1999), uma vez que é responsável por evitar que a malha "cubra" áreas côncavas presentes na superfície original (ver seção 5.1). Apesar de esta função ser executada pela operação de remoção de arestas, é possível observar, principalmente em malhas de baixa densidade reconstruídas pelo GCS-M, algumas estruturas cuja posição das arestas não é a mais adequada aumentando o erro e a qualidade visual da malha. Por este motivo um dos tópicos deste trabalho estuda a melhor forma de empregar esta operação no algoritmo, pois o custo computacional do mesmo é relativamente elevado, uma vez que para cada triangulo, é necessário percorrer todos os pontos da nuvem de pontos a fim de encontrar o ponto mais próximo.

Observou-se que após a realização das operações de remoção de arestas, ocorriam algumas faces mal posicionadas próximas do local onde a aresta foi removida. É proposto, então, efetuar a operação de troca apenas nas arestas incidentes no vértice resultante da operação de remoção de arestas (o vértice **a** na Figura 41), e observar os resultados.

O algoritmo proposto é o seguinte:

- Após efetuar cada operação de remoção de arestas:
    - Percorrer todas as arestas incidentes no vértice resultante;
    - Localizar a arestas com maior desvio, sendo o desvio de uma aresta e definido pela Equação (7.2):

$$Dev(e) = DM(t_1, X) + DM(t_2, X), \quad (7.2)$$

onde $DM(t, X)$ é a distância mínima entre o triângulo *t* e o conjunto de pontos *X*, definido pela Equação (7.3):

$$DM(t, X) = \min_{\mathbf{x}_i \in X} \left\| centroide(t) - \mathbf{x}_i \right\|, \quad (7.3)$$

- Aplicar a troca de aresta sobre a aresta com o maior desvio (Figura 42 (a));
- Caso o desvio na nova aresta seja menor, então a troca é mantida, caso contrário, a troca é desfeita.

A Figura 42 ilustra os principais passos do procedimento. Na seção 8.3 são apresentados os resultados.

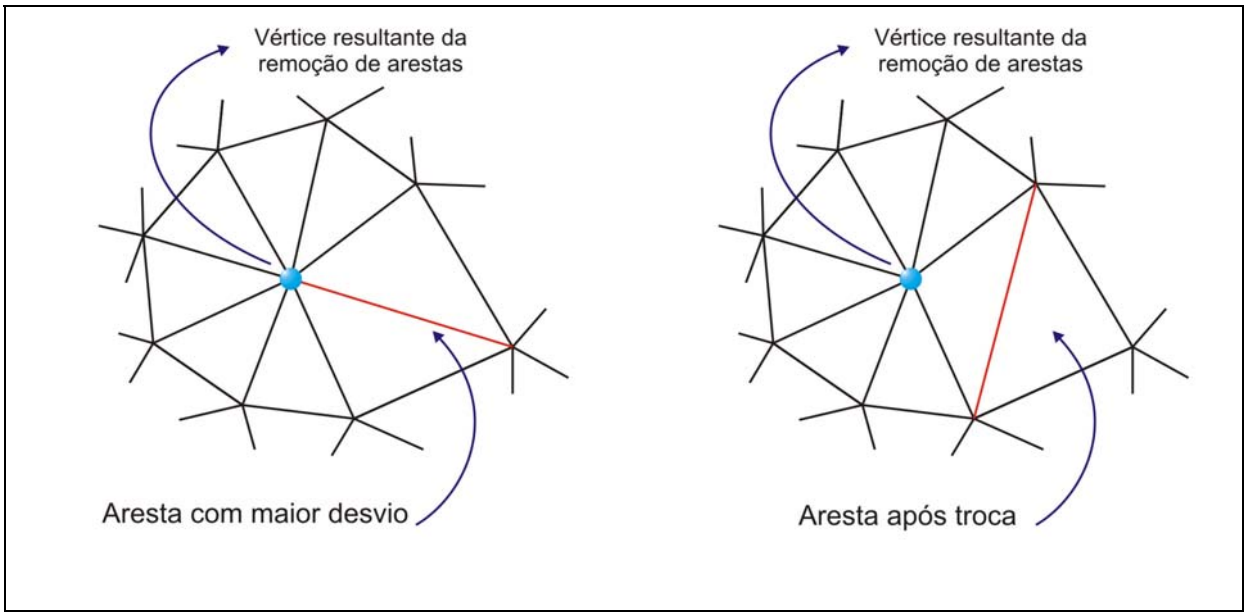


Figura 42: (a) Após a operação de remoção de arestas, deve-se localizar o vértice remanescente que pertencia à aresta removida e determinar sua aresta incidente com o maior desvio. (b) Aplica-se a operação de troca de aresta, caso o desvio diminua a troca é mantida, caso contrário ela é desfeita.

7.7 Métricas para Avaliação da Qualidade da Malha

Com o objetivo de mensurar a qualidade da superfície reconstruída e consequentemente a eficácia do algoritmo implementado, assim como, proporcionar a possibilidade de efetuar

comparações com outros métodos presentes na literatura, algumas métricas foram selecionadas para quantificar a qualidade das reconstruções obtidas. Três fatores foram escolhidos: o erro médio de aproximação entre a superfície gerada e a nuvem de pontos, a qualidade dos triângulos (faces) que formam a malhas, e a distribuição da valência entre os vértices.

7.7.1 Erro Médio entre a Malha e a Nuvem de Pontos

Considerando a relação entre as diversas formas de mensurar o erro médio entre a superfície reconstruída e a nuvem de pontos, é possível separar quatro delas (Tabela 2):

Tabela 2: Definição das métricas possíveis para quantificar a distancia média entre a superfície gerada e a nuvem de pontos

	Descrição	Equação
1	Média das distâncias mínimas entre cada ponto x e o conjunto de vértices V.	$\overline{DM}(X,V)=\frac{1}{size(X)}\sum_{j=0}^{j=size(X)} DM(\mathbf{x}_j,V)$, onde $DM(\mathbf{x},V)=\min_{v_i\in M}\left\|\mathbf{x}-v_i\right\|$.
2	Média das distâncias mínimas entre cada ponto e o conjunto dos centróides dos triângulos da malha.	$\overline{DM}(X,T)=\frac{1}{size(T)}\sum_{j=0}^{j=size(T)} DM(\mathbf{x}_j,T)$, onde $DM(\mathbf{x},T)=\min_{t_i\in M}\left\|\mathbf{x}-centroide(t_i)\right\|$.
3	Media das distâncias mínimas entre cada vértice v e a nuvem de pontos	$\overline{DM}(V,X)=\frac{1}{size(V)}\sum_{j=0}^{j=size(V)} DM(v_j,X)$, onde $DM(v,X)=\min_{x_i\in X}\left\|v-\mathbf{x}_i\right\|$.
4	Média das distâncias mínimas entre o centróide de cada triângulo e a nuvem de pontos.	$\overline{DM}(T,X)=\frac{1}{size(T)}\sum_{j=0}^{j=size(T)} DM(v_j,T)$, onde $DM(t,X)=\min_{\mathbf{x}_i\in X}\left\|centroide(t)-\mathbf{x}_i\right\|$.

Quando, por exemplo, emprega-se a métrica descrita na linha 1 da Tabela 2, da mesma forma que a descrita na linha 3, pode ocorrer que mesmo estando todos os vértices da malha localizados próximos, algumas faces estejam localizadas muito distantes da nuvem de

pontos, talvez até cobrindo regiões côncavas. Isso pode ser observado no trabalho de Yu (YU, 1999), quando é aplicado o procedimento de troca de arestas. Portanto, esta medida não é muito adequada, principalmente para algoritmos onde o nível de refinamento da malha pode ser escolhido pelo usuário, como é o caso da maioria dos algoritmos de reconstrução baseados em redes neurais, como o apresentado neste trabalho, tornando a comparação injusta.

Uma abordagem que obtenha os valores apenas das distâncias mínimas entre os centróides dos triângulos e as nuvens de pontos também é possível (linhas 2 e 4, na Tabela 2). Porque, caso ocorram triângulos incorretos na superfície, a presença deles influencia no valor final da qualidade da malha, enquanto utilizando apenas as informações dos vértices é possível que uma malha com graves defeitos apresente um valor de erro muito baixo.

A partir da discussão acima, foi determinado que uma métrica mais robusta e adequada ao problema de reconstrução de superfícies é o valor da média das distâncias mínimas entre cada elemento do conjunto formado pela união do conjunto dos vértices e do conjunto dos centróides dos triângulos, como mostra a Equação (7.4):

$$DM(M, X) = \min_{x_i \in X} \left\| e - x_i \right\|, \quad M = V \cup centroide(T), e \in M \qquad (7.4)$$

A filosofia por trás desta métrica é permitir uma comparação menos injusta entre algoritmos pertencentes a paradigmas diferentes como é o caso deste trabalho onde o *Power Crust* e o *Tight Cocone* pertencem ao grupo de algoritmos baseados em interpolação e o GCS-M que é baseado na teoria das Redes Neurais, uma vez que polígonos mal posicionados que estejam cobrindo regiões côncavas não aumentariam a medida de erro considerada na Equação mostrada na linha 1 da Tabela 2.

7.7.2 Qualidade dos Triângulos e Distribuição de Valências

Como foi apontado por (BRITO JUNIOR, 2004) a medida de erro baseada apenas na média das distâncias mínimas entre a malha gerada e a nuvem de pontos, não traz informações sobre a qualidade visual da malha reconstruída. Para isso o autor propõe uma métrica para avaliar a qualidade individual de cada triângulo (polígonos em geral) presente na malha. A obtenção da sua média proporciona uma medida coerente da regularidade dos polígonos, portanto da qualidade visual da malha.

A métrica proposta em (BRITO JUNIOR, 2004) consiste em obter a razão entre a distância mínima e máxima dos vértices do polígono *P* em relação ao seu centróide, dada pela Equação (7.5) e ilustrada na Figura 43:

$$R(P) = \frac{\min \left\| v - centroide(P) \right\|}{\max \left\| v - centroide(P) \right\|}, \qquad (7.5)$$

Outra medida que deve ser levada em consideração, também com relação à qualidade visual da malha, é distribuição das valências (número de vértices incidentes) dos vértices. Vértices com valências muito altas não são visualmente agradáveis. Quanto mais uniforme a distribuição das valências mais visualmente agradável é a malha.

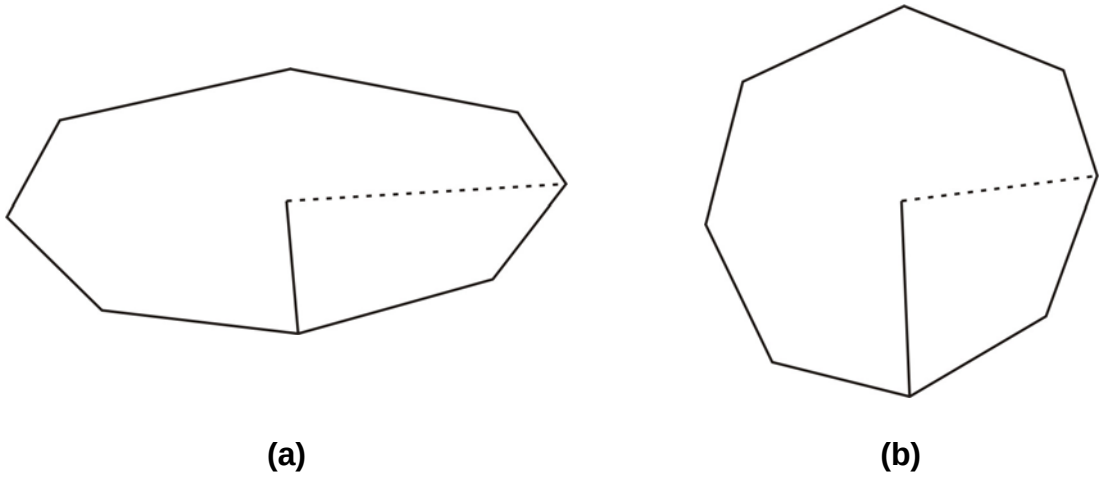


Figura 43: Métrica para qualidade dos polígonos. Linha continua representa a menor distancia e linha pontilhada a maior distância. (a) Polígono com qualidade ruim. (b) Polígono com boa qualidade.

7.8 Considerações Finais

Este Capítulo apresentou detalhes da implementação do algoritmo proposto, incluindo linguagem de programação utilizada, bibliotecas utilizadas, configuração dos computadores utilizados, bem como as nuvens de pontos utilizadas nos testes.

Descrições detalhadas das classes que compõem a implementação são apresentadas. Assim como a classe *CGAL::Polihedron_3<Traits>* implementado na biblioteca CGAL, utilizada para armazenar a superfície poligonal gerada pelo algoritmo.

Foram definidas também métricas para poder mensurar a qualidade das superfícies geradas. O próximo capitulo são apresentados os resultados e respectivas análises.

8 RESULTADOS OBTIDOS

Os resultados obtidos com a implementação do algoritmo descrita no capítulo anterior são apresentados neste capitulo. A seção 8.1 apresenta a primeira série de experimentos realizada, buscando estabelecer a melhor combinação entre os parâmetros de taxa de aprendizagem do vértice vencedor e taxa de aprendizagem dos vizinhos. A seção 8.1 apresenta a execução do algoritmo utilizando a melhor combinação dos parâmetros obtidos na seção anterior, com uma maior resolução da malha poligonal final, i.e. maior número de vértices e polígonos, e compara os resultados com os algoritmos *Power Crust* e *Tight Cocone*. A seção 8.3 inclui os resultados obtidos com a inclusão da operação de troca de arestas. A seção 8.4 apresenta a análise dos resultados.

8.1 Escolha dos parâmetros

Para obter a melhor combinação entre os parâmetros de taxa de aprendizagem do vértice vencedor e taxa de aprendizagem dos vértices vizinhos, o algoritmo foi executado uma série de vezes com diversas combinações destes parâmetros. Os parâmetros testados são mostrados na Tabela 3.

As informações contidas na Tabela 3, estão representadas pelos gráficos apresentados na Figura 44 e na Figura 45, onde é possível observar a variação da qualidade frente às modificações nas combinações de parâmetros. A métrica utilizada foi a média das

distâncias mínimas entre cada triângulo (centróide) e a nuvem de pontos ($\overline{DM}(t, X)$). A superfície gerada possui 6000 vértices.

Tabela 3: Qualidade das reconstruções obtidas com diferentes combinações de parâmetros de taxa de aprendizagem do vértice vencedor (linhas) e taxa de aprendizagem da vizinhança (colunas). Os valores em vermelho representam os piores resultados e os azuis os melhores.

		A	B	C	D	E
		0,002	0,004	0,006	0,008	0,01
1	0.01	**0,000911269**	**0,000885748**	**0,000873499**	**0,000862533**	**0,00082523**
2	0.02	0,000697305	0,000665715	0,00066782	0,000663442	0,00065757
3	0.03	0,000650516	**0,000628239**	0,000633642	0,00063704	0,000633458
4	0.04	0,000650862	0,000631647	0,000625509	0,000605686	0,000596566
5	0.05	0,000660446	0,000657881	0,000592455	0,000612181	0,000621695
6	0.06	0,000688794	0,000629175	**0,000603792**	0,000596801	0,000621471
7	0.07	0,000648519	0,000634258	0,000612099	**0,000596651**	0,000601843
8	0.08	0,000686611	0,00064136	0,000623977	0,000597131	**0,000584766**
9	0.09	**0,000648196**	0,000660064	0,000607752	0,000608374	0,000601259
10	0.10	0,000666018	0,000641958	0,000630562	0,000612315	0,000594857
11	Máximo	**0,000911269**	0,000885748	0,000873499	0,000862533	0,00082523
12	Mínimo	0,000648196	0,000628239	0,000592455	0,000596651	**0,000584766**

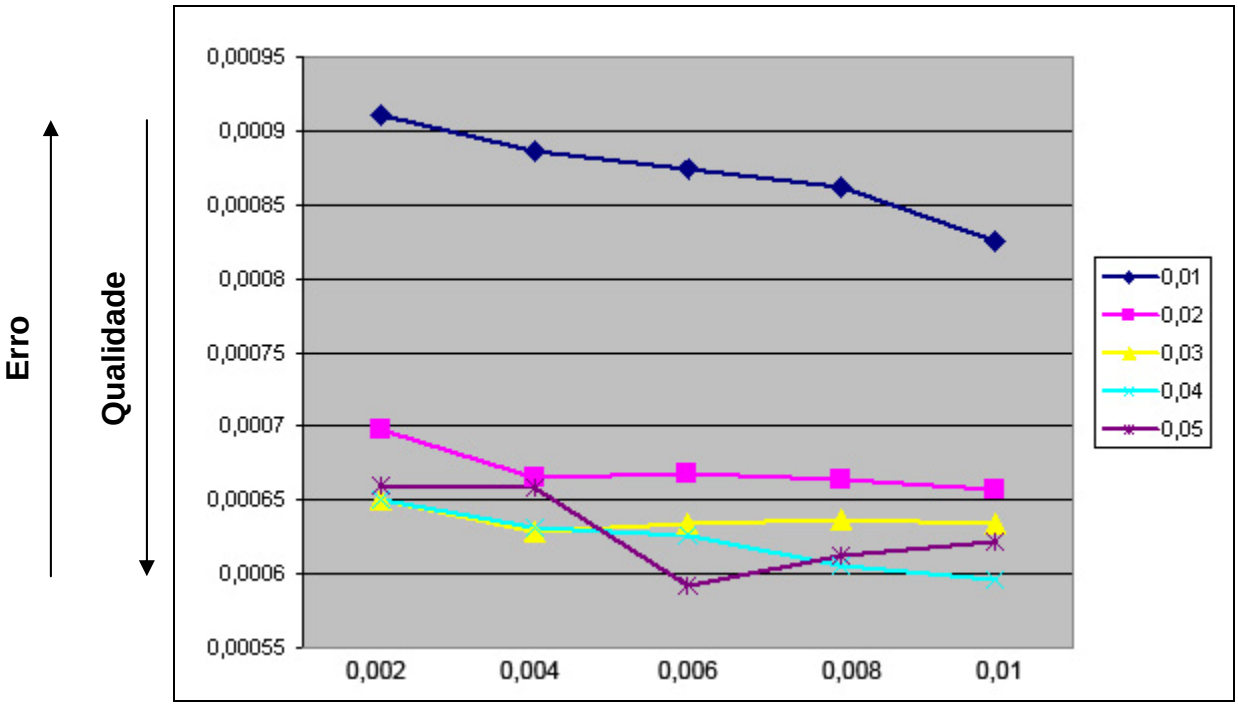


Figura 44: Evolução da qualidade da malha gerada pelo algoritmo quando se varia a taxa de aprendizagem da vizinhança em relação à taxa de aprendizagem do vértice vencedor. Cada linha representa um determinado valor da taxa de aprendizagem do vértice vencedor, o eixo x representa a variação do taxa de aprendizagem da vizinhança. O eixo y representa a qualidade da malha. Este gráfico engloba as cinco primeiras linhas da Tabela 3.

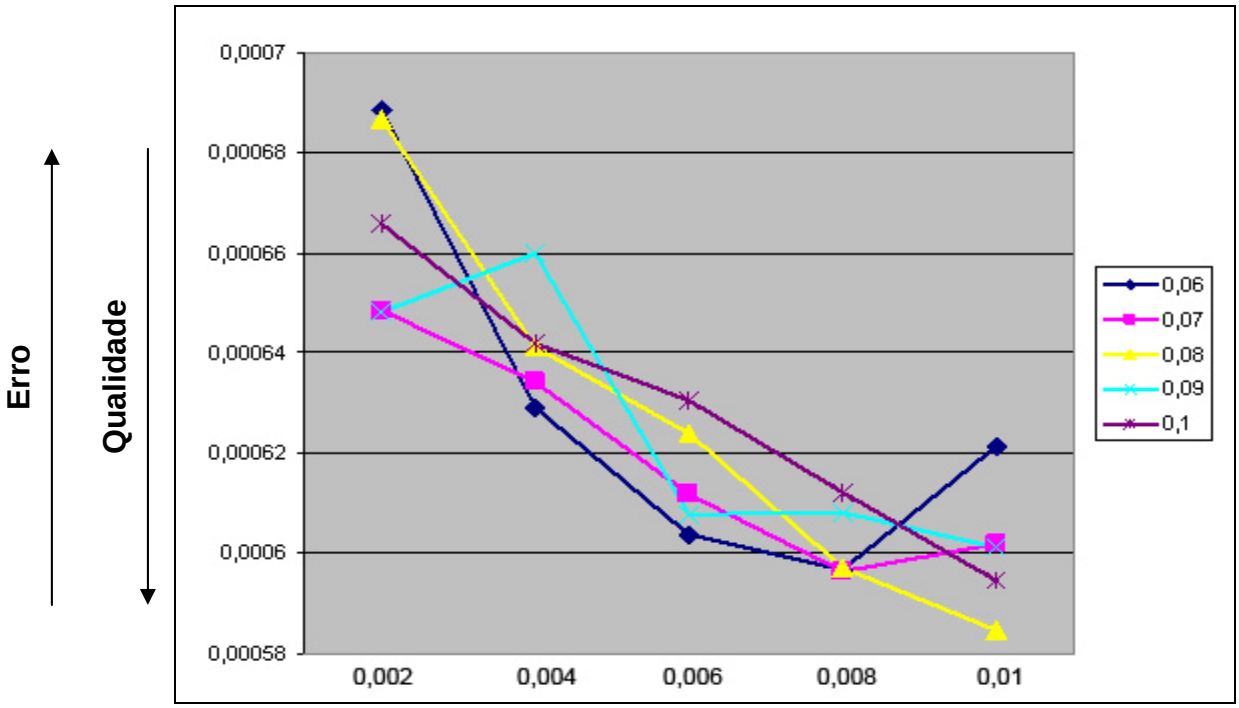
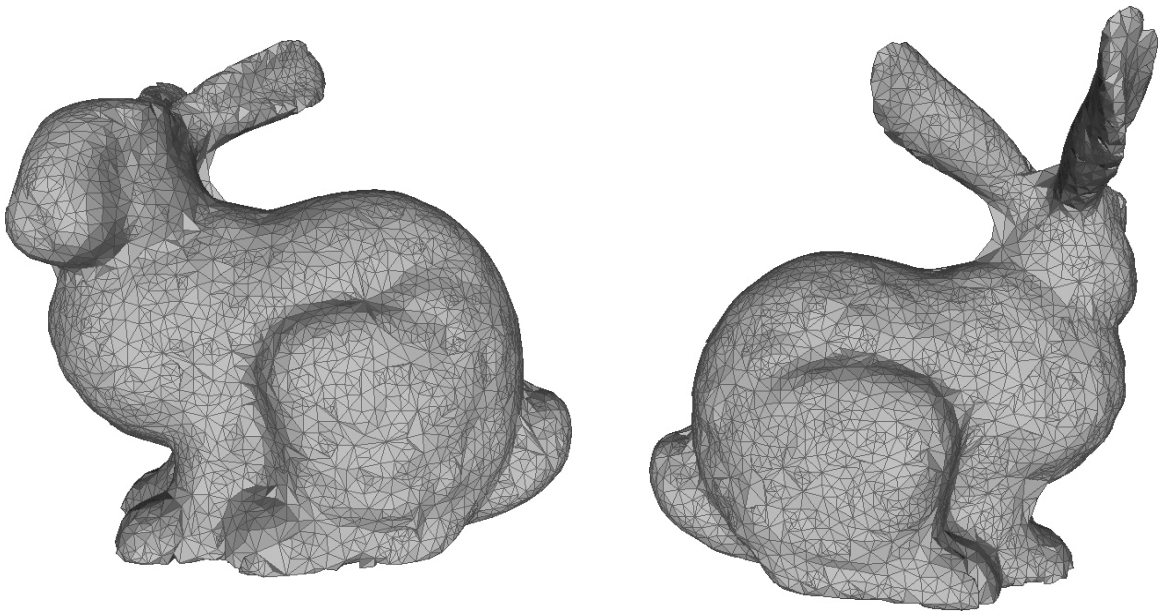


Figura 45: O mesmo que o gráfico acima, porém para as cinco últimas linhas da Tabela 3.

As imagens das superfícies que obtiveram as melhores medidas estão na Figura 46. De acordo com a tabela os parâmetros utilizados foram: 0.08 para o a taxa de aprendizagem de vértice vencedor e 0,01 para os vizinhos. O número de vértices pertencentes na malha é 6000.



(a) **(b)**

Figura 46: Reconstrução obtida pelo algoritmo com os parâmetros que obtiveram as melhores medidas de qualidade (Tabela 3).

8.2 Comparação com métodos clássicos

Para comparar os resultados obtidos com a implementação do algoritmo GCS-M, foram escolhidos os algoritmos *Power Crust*[12] (AMENTA, CHOI e KOLLURI, 2001a) e o *Tight Cocone*[13] (DEY e GOSMWAMI, 2002). Estes algoritmos foram escolhidos não só por serem bastante conhecidos na literatura, como também pelo fato de seus pesquisadores disponibilizarem os códigos fonte de implementações na internet.

A malha poligonal gerada por estes algoritmos é uma função do tamanho da nuvem de pontos, por isso não é possível o usuário escolher a densidade final da malha. Para que a comparação fosse mais justa foram geradas reconstruções com número de vértices próximos do número de pontos presentes nas nuvens. É sabido que o *Tight Cocone* gera uma

[12] http://www.cs.utexas.edu/users/amenta/powercrust/
[13] http://www.cse.ohio-state.edu/~tamaldey/cocone.html

malha com o número de vértices igual ao número de pontos, na verdade, os vértices são os próprios pontos presentes na nuvem. O *Power Crust* gera uma grande quantidade de vértices extras na superfície da malha, e sua malha é composta por polígonos com número arbitrário de lados, não apenas triângulos como o GCS-M e o *Tight Cocone*.

A Figura 47 e a Figura 48 apresentam a reconstrução da nuvem de pontos do Coelho de Stanford (35.947 pontos) efetuada pelo GCS-M até alcançar a resolução de 30.000 vértices. Os parâmetros utilizados na execução são mostrados na Tabela 4.

Tabela 4: Parâmetros utilizados na execução de algoritmo GCS-M para obter a malha abaixo, sem a operação de troca de arestas.

Coelho de Stanford	
Número de Vértices:	30.000
Taxa de aprendizagem do vértice vencedor (α_w):	0.08
Taxa de aprendizagem dos vértices vizinhos (α_n):	0.01
Constante de ponderação do contador de sinal (α):	0.95
Intervalo entre subdivisão de vértices (C_{vs}):	100
Intervalo entre remoção de arestas (C_{ec}):	20

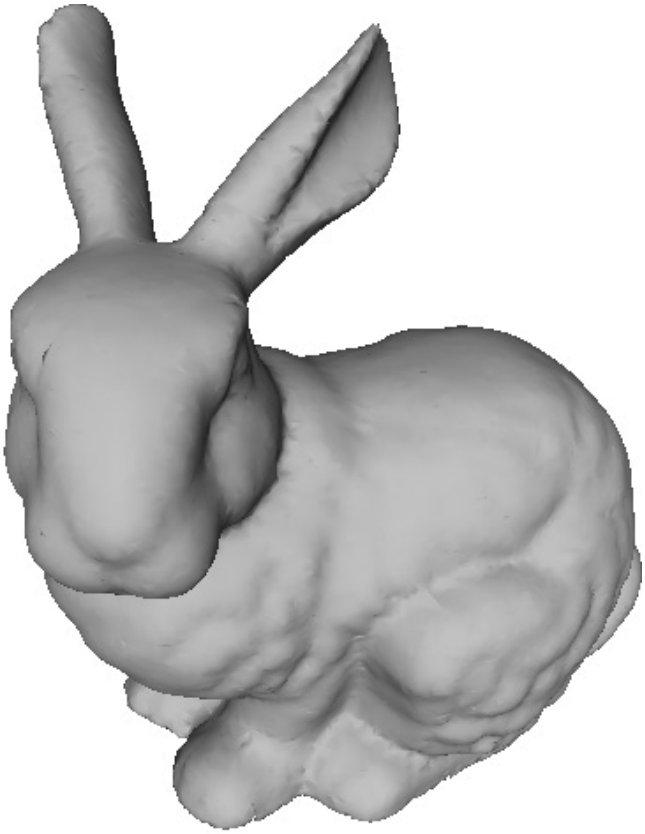
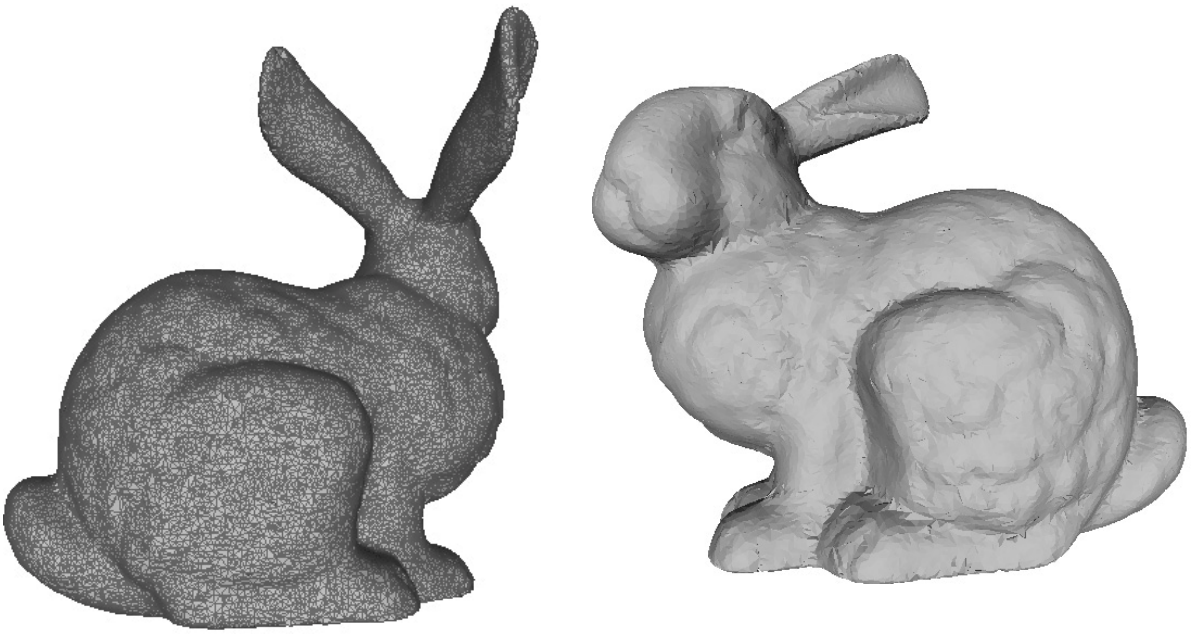


Figura 47: Resultado da reconstrução da nuvem de pontos do Coelho de Stanford, com 35.947 pontos. A malha poligonal foi gerada com 30.000 vértices.



(a) **(b)**

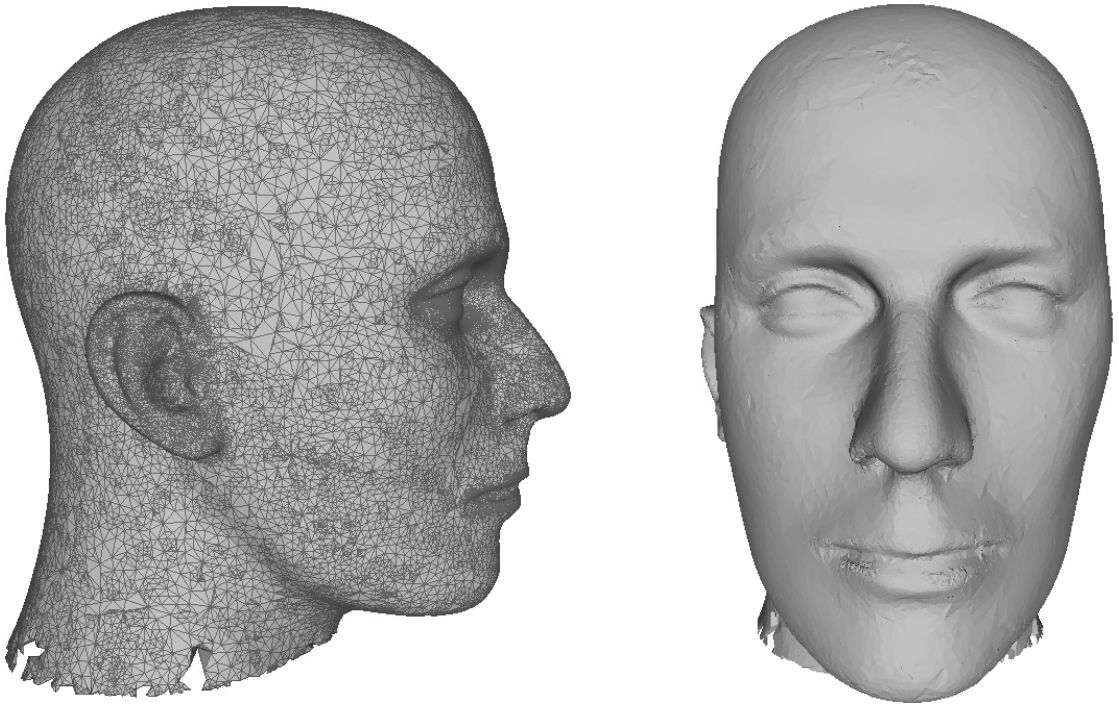
Figura 48: (a) Mesmo resultado apresentado na Figura 47, agora por outro ângulo e com arestas visíveis, onde é possível observar os polígonos. (b) Também o mesmo resultado por um ângulo diferente (apenas a superfície).

A Figura 49 a apresenta a nuvem de pontos da Cabeça do Manequim (41117 pontos) reconstruída pelo GCS-M até alcançar a resolução de 40000 vértices. Os parâmetros utilizados na execução são mostrados na Tabela 5.

Tabela 5: Parâmetros utilizados na execução de algoritmo GCS-M para obter a malha abaixo, sem a operação de troca de arestas.

Cabeça do Manequim	
Número de Vértices:	40.000
Taxa de aprendizagem do vértice vencedor (α_w):	0.08
Taxa de aprendizagem dos vértices vizinhos (α_n):	0.01
Constante de ponderação do contador de sinal (α):	0.95
Intervalo entre subdivisão de vértices (C_{vs}):	100
Intervalo entre remoção de arestas (C_{ec}):	20

Figura 49: Resultado da reconstrução da nuvem de pontos de uma cabeça humana, com 41.117 pontos. A malha poligonal foi gerada com 40.000 vértices.



(a) **(b)**

Figura 50: (a) Mesmo resultado apresentado na Figura 49, agora por outro ângulo e com as arestas e polígonos visíveis. (b) Também o mesmo resultado por um ângulo diferente.

8.2.1 GCS-M vs. Power Crust vs. Tight Cocone

Comparação dos resultados entre o algoritmo implementado (GCS-M) e algoritmos tradicionais na literatura: *Power Crust* e *Tight Cocone*. A Tabela 6 e a Tabela 7 apresentam as comparações relativas à nuvem de pontos do Coelho de Stanford, enquanto a Tabela 8 e a Tabela 9 incluem as comparações referentes à nuvem de pontos da Cabeça do Manequim.

Tabela 6: Medidas de qualidade obtidas pelos algoritmos *GCS-M*, *Power Crust* e *Tight Cocone*

COELHO DE STANFORD			
Métrica	**GCS-M**	**Power Crust**	**Tight Cocone**
$\overline{DM}(X,V)$	0.000460632	0.000121052	0.0*
$\overline{DM}(X,T)$	0.000450135	0.000213711	0.000614375
$\overline{DM}(V,X)$	0.000397018	0.00035896	0.0*
$\overline{DM}(V,T)$	0.000499078	0.000443562	0.000644152
$\overline{DM}(M,X)$	0.000465057	0.000393575	0.000429427
Qualidade dos polígonos	0.562959	0.519418	0.617074

*Como o conjunto de vértices gerado pelo *Tight Cocone* é o conjunto de pontos em si, é natural que o erro seja 0.0, caso a reconstrução esteja correta.

Tabela 7: Distribuição das valências entre os vértices.

Valência	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**>10**
GCS-M	0	1727	5854	6156	6086	4361	2572	1793	818	933
Power Crust	0	170524	106714	0	0	0	0	0	0	0
Tight Cocone	0	16	679	4921	24707	4990	587	32	7	5

Tabela 8: Medidas de qualidade obtidas pelos algoritmos *GCS-M*, *Power Crust* e *Tight Cocone sem a operação de troca de arestas.*

CABEÇA DE MANEQUIM			
Métrica	**GCS-M**	**Power Crust**	**Tight Cocone**
$\overline{DM}(X,V)$	0.0212383	0.0006585533	0.0*
$\overline{DM}(X,T)$	0.0222745	0.0036306	0.032026
$\overline{DM}(V,X)$	0.0204768	0.0103261	0.0*
$\overline{DM}(V,T)$	0.0274762	0.0174615	0.03476

$\overline{DM}(M, X)$	0.025143	0.0132989	0.0231588
Qualidade dos polígonos	0.591143	0.510509	0.685761

* Como o conjunto de vértices gerado pelo *Tight Cocone* é o conjunto de pontos em si, é natural que o erro seja 0.0, caso a reconstrução esteja correta.

Tabela 9: Distribuição das valências entre os vértices.

Valência	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**>10**
GCS-M	0	889	8936	8196	10188	5680	3439	1719	879	1191
Power Crust	0	245.414	183.877	0	0	0	0	0	0	0
Tight Cocone	7	178	2104	7354	22312	6886	1695	391	93	30

8.3 GCS-M + Troca de Arestas

Esta seção discute os resultados obtidos com o algoritmo proposto GCS-M com a operação de troca de arestas para o “Coelho de Stanford”. Foram realizados testes utilizando os parâmetros de taxa de aprendizagem do vértice vencedor e de sua vizinhança direta cuja execução gerou as malhas com pior e melhor qualidade, considerando as informações da Tabela 3. A seção 8.3.1 apresenta os resultados e comparações para os parâmetros com pior medida de qualidade, enquanto a seção 8.3.2 apresenta os resultados para os parâmetros que obtiveram a melhores qualidades.

8.3.1 Combinação de Parâmetros: Pior Caso

Os parâmetros utilizados para ambas as execuções (com e sem troca de arestas) estão na Tabela 10. A Tabela 11 apresenta as métricas relativas à qualidade da malha obtidas com a execução do algoritmo com e sem a operação de troca de arestas. A malha foi gerada com 6000 vértices.

Tabela 10: Parâmetros utilizados na reconstrução do modelo do Coelho que gerou as malhas com a pior medida de qualidade

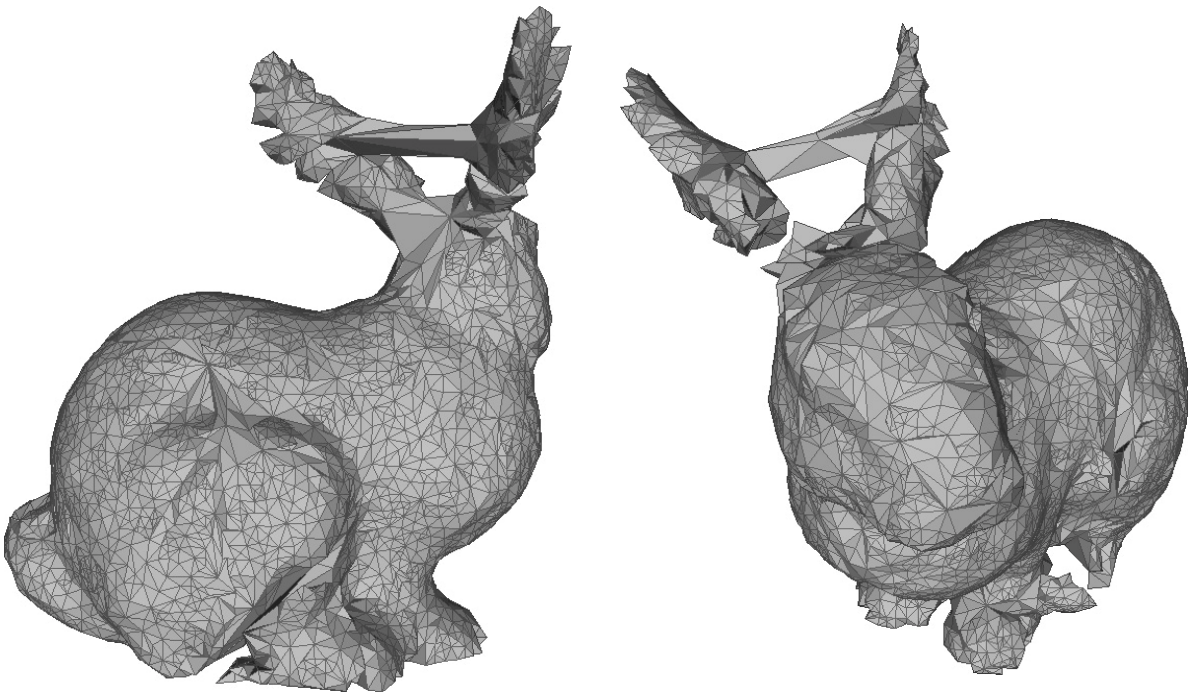
Coelho de Stanford	
Número de Vértices:	6000
Taxa de aprendizagem do vértice vencedor (α_w):	0.01
Taxa de aprendizagem dos vértices vizinhos (α_n):	0.002
Constante de ponderação do contador de sinal (α):	0.95
Intervalo entre subdivisão de vértices (C_{vs}):	100
Intervalo entre remoção de arestas (C_{ec}):	20

Tabela 11: Comparativo entre as medidas de qualidade obtidas das superfícies geradas pelo algoritmo com a operação de troca de arestas e sem a operação de troca de arestas.

COELHO DE STANFORD		
Métrica	**GCS-M + Troca de Arestas**	**GCS-M**
$\overline{DM}(X,V)$	0.00148284	0.0015
$\overline{DM}(X,T)$	0.00128097	0.00131667
$\overline{DM}(V,X)$	0.000802037	0.000842212
$\overline{DM}(V,T)$	0.000860746	0.000916076
$\overline{DM}(M,X)$	0.000841172	0.000891449
Qualidade dos polígonos	0.572324	0.572569

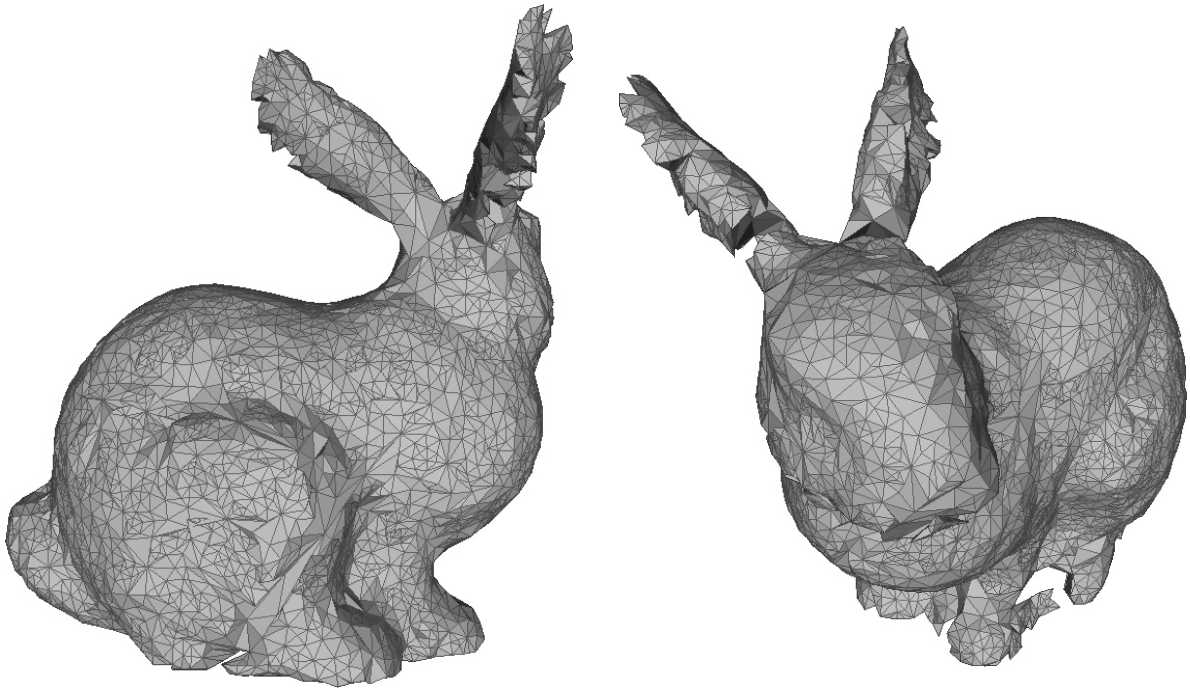
A Figura 51 e a Figura 52 apresentam as imagens obtidas por meio da execução do algoritmo com os parâmetros de taxa de aprendizagem de vértice vencedor 0.01 e a taxa de aprendizagem de sua vizinhança 0.002, parâmetros cuja combinação gerou a malha com a pior medida de qualidade (Tabela 3), mostrada na seção 8.1. A Figura 51 mostra as imagens da superfície obtida pelo algoritmo sem a inserção da operação de troca de arestas, proposta na seção 7.6, enquanto as imagens da Figura 52 mostram a superfície gerada pelo algoritmo com a operação de troca de arestas.

É possível notar, principalmente, que a inserção desta operação foi capaz de evitar o erro grosseiro ocorrido na região da orelha do coelho. Isso permite concluir que a operação de troca de arestas proposta na seção 7.6 não apenas melhora a qualidade da malha, como o torna o algoritmo mais robusto.



(a) **(b)**

Figura 51: Superfície gerada pelo algoritmo implementado sem a inclusão da operação de troca de arestas. Foram utilizados os parâmetros que obtiveram os piores valores de qualidade (Tabela 3).



(a) **(b)**

Figura 52: Superfície gerada pelo algoritmo implementado com a inclusão da operação de troca de arestas. Foram utilizados os parâmetros que obtiveram os piores valores de qualidade. (Tabela 3).

8.3.2 Combinação de Parâmetros: Melhor Caso

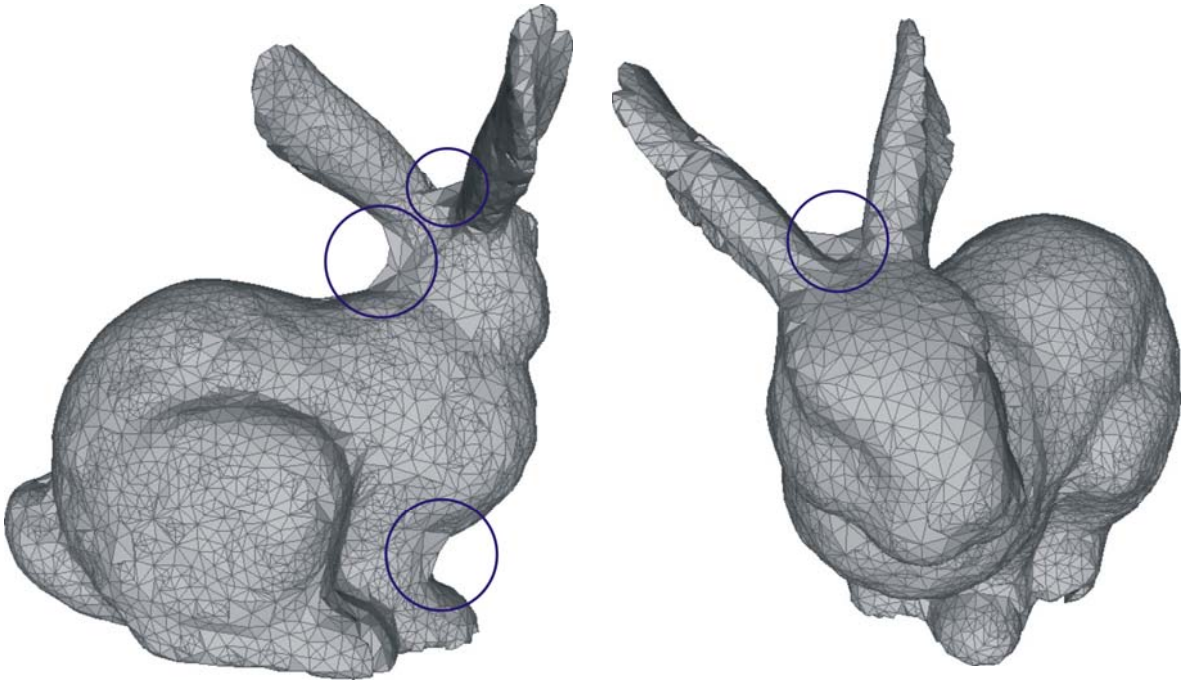
A Figura 53 e Figura 54 apresentam, respectivamente, as imagens obtidas das superfícies geradas pelo algoritmo utilizando parâmetros de taxa de aprendizagem do vértice vencedor 0.08 e 0.01 para a taxa de aprendizagem da vizinhança direta do vértice vencedor. Esta combinação de parâmetros resultou na malha com a melhor medida de qualidade, como pode ser visto na Tabela 3 (linha 8, coluna E). Na Figura 53 não foi aplicada à operação de troca de arestas, enquanto nas imagens da Figura 54 a operação foi aplicada. Apesar de menos evidente em relação às imagens da Figura 51, é possível notar a melhora em algumas áreas da superfície, como entre as orelhas e atrás do pescoço do coelho.

Tabela 12: Parâmetros utilizados na reconstrução do modelo do Coelho que gerou as malhas com a melhor medida de qualidade

Coelho de Stanford	
Número de Vértices:	6000
Taxa de aprendizagem do vértice vencedor (α_w):	0.08
Taxa de aprendizagem dos vértices vizinhos (α_n):	0.01
Constante de ponderação do contador de sinal (α):	0.95
Intervalo entre subdivisão de vértices (C_{vs}):	100
Intervalo entre remoção de arestas (C_{ec}):	20

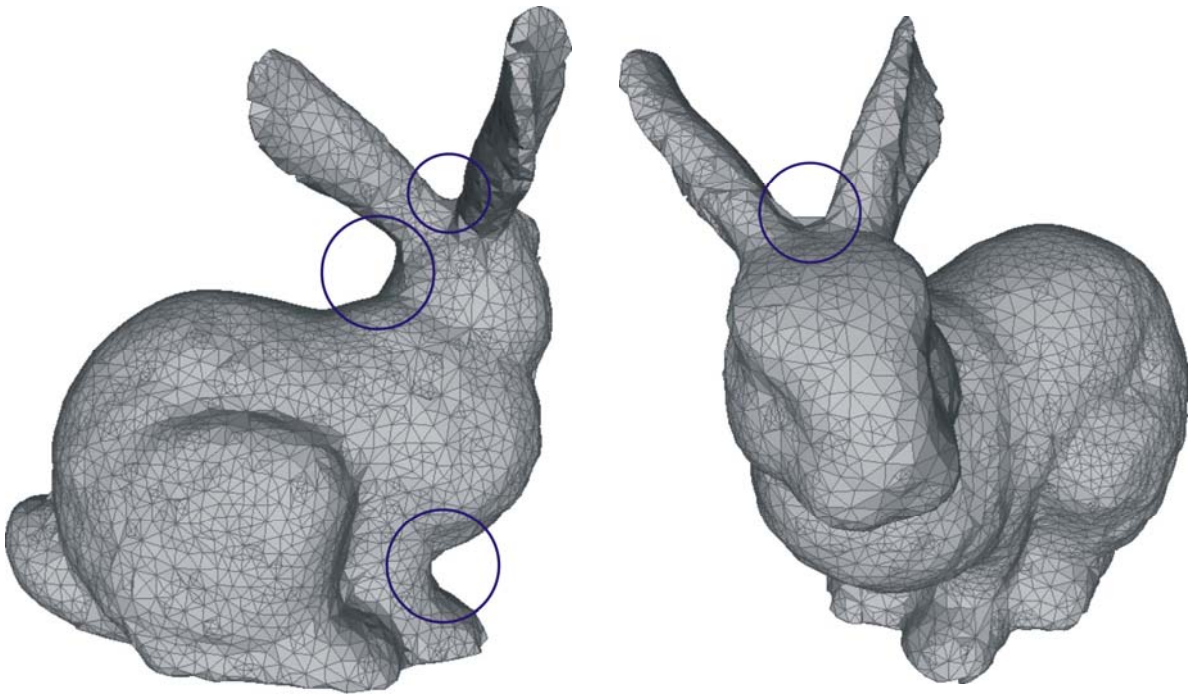
Tabela 13: Comparativo entre as medidas de qualidade obtidas das superfícies geradas pelo algoritmo com a operação de troca de arestas e sem a operação de troca de arestas.

Métrica	**GCS-M + Troca de Arestas**	**GCS-M**
$\overline{DM}(X,V)$	0.00129623	0.00130669
$\overline{DM}(X,T)$	0.00102182	0.00103236
$\overline{DM}(V,X)$	0.000540991	0.000550733
$\overline{DM}(V,T)$	0.000581697	0.000596268
$\overline{DM}(M,X)$	0.000568125	0.000581086
Qualidade dos polígonos	0.63058	0.631307



(a) (b)

Figura 53: Superfície gerada pelo algoritmo implementado sem a inclusão da operação de troca de arestas. Foram utilizados os parâmetros que obtiveram os melhores valores de qualidade (Tabela 3).



(a) (b)

Figura 54: Superfície gerada pelo algoritmo implementado com a inclusão da operação de troca de arestas. Assim cimo na Figura 53 foram utilizados os parâmetros que obtiveram os melhores valores de qualidade (Tabela 3).

8.4 Análise dos Resultados

Com base nos resultados obtidos, é possível observar que os algoritmos de reconstrução com base em modelos de Redes Neurais Auto-Organizáveis, como é o caso do GCS-M, possuem diversas características favoráveis, como o fato de o tempo do treinamento ser independente do tamanho da nuvem de pontos, o que permite também que a densidade da malha reconstruída possa ser escolhida pelo usuário.

No entanto, os algoritmos de reconstrução que seguem o paradigma neural ainda têm muito a evoluir. Observa-se que o número de projetos de pesquisa envolvendo esses métodos ainda é muito inferior, por exemplo, quando comparado com os métodos baseados em geometria computacional, como o *Power Crust* e o *Tight Cocone*. Estes dois algoritmos utilizados para os testes de comparação, além de bastante pesquisados, ambos são fruto de melhorias sobre outros algoritmos: *Crust* (AMENTA, BERN e EPPSTEIN, 1998) e *Cocone* AMENTA et al. (2000) respectivamente.

8.4.1 Principais contribuições

Como contribuições principais deste trabalho, podem-se citar:

- A inclusão da operação de troca de arestas no algoritmo GCS-M, proporciona uma melhor qualidade na malha reduzindo as medidas de erro, e melhorando a convergência do algoritmo.
- A proposta e implementação do algoritmo GCS-M utilizando a biblioteca CGAL, estendendo a superfície poliédrica *CGAL::Polyhedron_3<Traits>*.

- A análise da evolução dos parâmetros de aprendizagem do vértice vencedor vs. os parâmetros de aprendizagem dos vértices pertencentes a sua vizinhança direta.
- A comparação com algoritmos *Power Crust* e *Tight Cocone*, ambos bastante citados na literatura.

8.5 Considerações finais

Neste capítulo, foram apresentados os resultados obtidos com a implementação do algoritmo GCS-M. Análises da qualidade da malha 3D foram obtidos e comparados com algoritmos tradicionais baseados em geometria computacional (*Power Crust* e *Tight Cocone*). Os resultados da inclusão da troca de arestas foram apresentados. Uma vez discutidos os resultados o próximo capítulo traz as conclusões e trabalhos futuros.

9 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou o projeto e implementação de um algoritmo de reconstrução de superfícies a partir de nuvens de pontos, usando Redes Neurais Auto-Organizáveis. O algoritmo em questão é baseado no modelo incremental de Rede Neural Auto-Organizável GCS, e foi denominado GCS-Modificado (GCS-M).

Foi verificado, por meio dos experimentos realizados, que o algoritmo proposto é capaz de gerar malhas com qualidade muito próxima e até mesmo superior em relação às superfícies geradas por métodos tradicionais na literatura, como pode ser observado nas comparações realizadas com *Power Crust* e o *Tight Cocone*.

A inclusão da operação de troca de arestas, o que foi modificado no algoritmo original, melhorou a qualidade das superfícies geradas e tornou o algoritmo mais robusto. Com base nos resultados obtidos nota-se que o algoritmo aqui apresentado tende a se tornar menos sensível às variações de parâmetros. Sem a operação de troca de arestas, algumas combinações de parâmetros podem gerar superfícies com erros grosseiros, principalmente nas regiões côncavas, esses erros são reduzidos durante o treinamento com a inserção do procedimento de trocas de arestas, e mesmo quando utilizadas as combinações de parâmetros que geram os melhores resultados, pequenas imperfeições na superfície são eliminadas por esta técnica.

Os algoritmos baseados em Redes Neurais são capazes de lidar de maneira eficiente com o problema de reconstrução de superfícies. Apesar de algoritmos de reconstrução pertencentes a esse paradigma estarem sendo pesquisados há aproximadamente

uma década, o número de modelos ainda é bastante inferior aos modelos baseados em geometria computacional, por exemplo, portanto ainda existe uma grande área a ser explorada nesta linha de pesquisa.

9.1 Trabalhos futuros

São sugeridos os seguintes itens para trabalhos futuros:

- Implementação da busca pelo nodo vencedor em uma árvore *Kd-tree*, otimizada para busca por vizinhos mais próximos em dimensões arbitras, o que melhoraria o custo computacional do algoritmo;
- Inserir os módulos de aprendizagem da topologia, possibilitando ao algoritmo reconstruir superfícies de tipos topológicos diferentes, identificando-os durante o treinamento; e
- Combinação do algoritmo GCS-M proposto neste trabalho com algoritmos tradicionais, principalmente os baseados em geometria computacional, buscando o desenvolvimento de técnicas que reúnam as qualidades dos dois paradigmas.

10 REFERÊNCIAS BIBLIOGRÁFICAS

AMENTA N.; BERN M.; EPPSTEIN D. The crust and the β-Skeleton: combinatorial curve reconstruction. **Graphical Models and Image Processing**, v. 60, n. 2, p. 125-135, Mar. 1998.

AMENTA, N.; BERN, M.; KAMVYSSELIS, M. A new Voronoi-based surface reconstruction algorithm. In: SIGGRAPH '98. **Proc of the 25th annual conference on Computer graphics and interactive techniques**. ACM Press, p. 415-421, 1998.

AMENTA, N.; BERN, M. Surface reconstruction by Voronoi filtering. **Discrete and Computational Geometry,** Springer New York, v. 22, n. 4, p. 481-504, Dez. 1999.

AMENTA N.; CHOI S.; KOLLURI R. K. The power crust. In: ACM SYMPOSIUM ON SOLID AND PHYSICAL MODELING. **Proceedings of the sixth ACM symposium on Solid modeling and applications**, ACM Press, p. 249-266, 2001a.

AMENTA N.; CHOI S.; KOLLURI R. K. The power crust, unions of balls, and the medial axis transform. **Computational Geometry,** Elsevier Science, v.19, n. 2, p. 127-135, Jul., 2001.

AMENTA N.; CHOI S.; DEY T. K.; LEEKHA N. A simple algorithm for homeomorphic surface reconstruction**.** IN: ANNUAL SYMPOSIUM ON COMPUTATIONAL GEOMETRY. **Proceedings of the sixteenth annual symposium on Computational geometry**, ACM Press, p. 213-222, 2000.

BARHAK, J.; FISCHER, A. **Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques**. In: IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, v. 7, n. 1, p. 1 – 16. Jan.-Mar. 2001.

BERMAN, A. M. **Data Structures via C++:** objects by evolution. USA, Oxford University Press, 1997, 496 p.

BOISSONNAT, J. D. Shape reconstruction from planar cross sections. **Computer Vision, Graphics, and Image Processing**, Academic Press Professional, Inc., v. 44, n. 1, p. 1-29, 1988.

BOUDJEMAI, F.; ENBERG, P. B.; POSTAIRE, J. G. Surface modeling by using self organizing maps of Kohonen. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2003, v. 3, p. 2418- 2423 v. 3, 5-8 Out. 2003.

BOUDJEMAÏ, F.; ENBERG, P. B.; POSTAIRE, J. G. Dynamic adaptation and subdivision in 3d-som: application to surface reconstruction. In: 17th IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE (ICTAI ‘05), p. 425-430, 2005.

BOURKE, P. Platonic solids: regular polytopes in 3D. Disponível em: <http://astronomy.swin.edu.au/~pbourke/geometry/platonic/>. Acesso em: 14 jun. 2006.

BRITO JUNIOR, A. M. **Uma aplicação de redes neurais auto-organizáveis à reconstrução tridimensional de superfícies**. 2005. 96 f. Tese (Doutorado em Engenharia Elétrica) Universidade Federal do Rio grande do Norte (UFRN), Natal, 2005.

CGAL Editorial Board. CGAL-3.1 User and reference manual. Disponível em <http://www.cgal.org/Manual/3.1/doc_html/cgal_manual/title.html>. Acesso em: 24 abr. 2007.

COSTA, J. A. F. **Classificação automática e análise de dados por redes neurais auto-organizáveis**. 1999. 345 f. Tese (Doutorado em Engenharia Elétrica) UNICAMP, Campinas, 1999.

CURLESS, B. L. **New methods for surface reconstruction from range images**. 1997. 189 f. Tese (Doutorado) - Department Of Electrical Engineering, Stanford University, Stanford, Jun. 1997.

CURLESS, B. L.; LEVOY, M. A volumetric method for building complex models from range images. In: SIGGRAPH '96. **Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques**. ACM Press, p. 303-312, 1996.

DEY, T. K.; GOSWAMI, S. Tight cocone: a water tight surface reconstruction. Relatório Técnico OSU-CISRC-12/02-TR31, The Ohio State University, 2002.

EDELSBRUNNER, H.; MÜCKE, E. P. Three-dimensional alpha shapes. **ACM Transactions On Graphics,** v. 13, n. 1, p. 43-72, Jan. 1994.

FESSANT, F.; AKNIN, P.; OUKHELLOU, L.; AND MIDENET, S. Comparison of Supervised Self-organizing maps using euclidian or mahalanobis distance in classification context. In: INTERNATIONAL WORK-CONFERENCE ON ARTIFICIAL AND NATURAL NEURAL NETWORKS, 6, 2001. **Proceedings of the 6th international Work-Conference on Artificial and Natural Neural Networks: Connectionist Models of Neurons, Learning Processes and Artificial intelligence-Part I**, Springer-Verlag, p. 637-644, Jun. 2001.

FRANÇA, G. G. D. M. **Desenvolvimento de um sistema de aquisição de informações volumétricas usando método de triangulação a laser e campo de visão variável**. 2004. 121 f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação – Universidade Federal de São Carlos (UFSCar), São Carlos, 2004.

FRANÇA, J. G. D. M.; GAZZIRO, M. A.; IDE, A. N.; SAITO, J. H. A 3d scanning system based on laser triangulation and variable field of view. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING 2005, ICIP 2005, v. 1, p. I-425-428, Set. 2005.

FRANÇA, J. G. D. M.; GAZZIRO, M. A.; QUEIROZ, R. C. A. S.; MENDONÇA, M. J. M.; SAITO, J. H. Scanner antropométrico para determinação do percentual de gordura. In: III Congresso Latino-Americano de Engenharia Biomédica, 2004, João Pessoa. **IFMBE Proceedings,** v. 5. p. 745-748 , 2004.

FRITZKE, B. Unsupervised clustering with growing cell structures. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. **Proceedings of IJCNN '91**. Seattle: IEEE Service Center, v. 2, p. 531-536, 1991.

FRITZKE, B., Growing cell structures - a self-organizing network for unsupervised and supervised learning, Relatório Técnico ICSI TR 93-026, International Computer Science Institute, University of California, Berkeley, 1993.

FRITZKE, B. A growing neural gas network learns topologies. In: TESAURO, G.; TOURETZKY, D. S.; LEEN, T. K., editores, **Advances in Neural Information Processing Systems 7**, p. 625-632. Cambridge: MIT Press, 1995a.

FRITZKE, B. Growing Grid: a self-organizing network with constant neighborhood range and adaptation strength. **Neural Processing Letters**, D Facto Publishing, v. 2, n. 5, p. 9-13, 1995b.

FRITZKE, B. Growing Self-Organizing networks - why? In: EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS. Proceedings of ESANN '96. p. 61-72, 1996.

FRITZKE, B. Unsupervised ontogenic networks. In: FIESLER E., BEALE R. **Handbook of Neural Computation**, IOP Publishing and Oxford University Press, 1997, p. C2.4:1-C2.4:16.

FULLER, R. B. **Synergetics**. New York, MacMillan, 1975.

GAZZIRO, M. A. **Projeto e construção de um scanner antropométrico baseado no método de triangulação a laser**. 2005. 61 f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação – Universidade Federal de São Carlos (UFSCar), São Carlos, 2005.

GIEZEMAN, G-J. VELTKAMP, R. WESSELINK, W. **Getting Started with CGAL.** Dez**,** 1999. Disponivel em < http://citeseer.ist.psu.edu/265158.html>. Visto em: 24 abr. 2007.

GOIS, J. P. **Reconstrução de superfícies a partir de nuvens de pontos**. 2004. 131 f. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Carlos, Abr. 2004.

GOTSMAN, C.; GU, X.; SHEFFER, A. Fundamentals of spherical parameterization for 3D meshes. **ACM Transactions on Graphics (TOG)**. ACM Press, v. 22, n. 3, p. 358-363, Jul. 2003.

HAYKIN, S. **Redes neurais:** princípios e práticas. Tradução Paulo Martins Engel. 2 ed. Porto Alegre: Bookman, 2001. 900 p.

HEARN, D.; BAKER M. P. **Computer graphics, C version**. 2. ed. Prentice Hall, 1996, 652 p.

HOFFMANN, G. Sphere tessellation by icosahedron subdivision. 2002.

HOPPE, H., DEROSE, T.; DUCHAMP, T.; MCDONALD, J.; STUETZLE W. Surface reconstruction from unorganized points. COMPUTER GRAPHICS (Proceedings of SIGGRAPH ‘92), v. 26, n. 2, p. 71-78, Jul. 1992.

HOPPE, H., DEROSE, T.; DUCHAMP, T.; MCDONALD, J.; STUETZLE W. Mesh optimization. COMPUTER GRAPHICS (Proceedings of SIGGRAPH ‘93), v. 26, p. 19-26, 1993.

HOPPE, H. **Surface reconstruction from unorganized points**. 1994. 116 f. Tese (Doutorado) - Department of Computer Science and Engineering, University of Washington, Washington, Jun. 1994.

HUBEL, D.; WIESEL, T. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. **Journal of Physiology**, v. 160, n. 1, p. 106-154, Jan. 1962.

IVRISSIMTZIS, I.P.; JEONG, W. K.; SEIDEL, H. P. Using growing cell structures for surface reconstruction. In: SHAPE MODELING INTERNATIONAL, 2003, p. 78-86, 12-15, Mai. 2003a.

IVRISSIMTZIS, I.P.; JEONG, W. K.; SEIDEL, H. P. Neural meshes: statistical learning methodds in surface reconstruction. Relatório Técnico MPI-I-2003-4-007, Max-Plank-Institut für Informatik, Abr. 2003b.

JAIN, R. C.; KASTURI, R.; SCHUNCK, B. G. **Introduction to machine vision.** McGraw-Hill Education, 1995, 549 p.

KETNNER, L. Using generic programming for designing a data structure for polyhedral surfaces. **Comput . Geom. Theory Appl.**, v. 13, p. 65-90, 1999.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological Cybernetics**, Springer Berlin / Heidelberg, v. 43, n. 1, p. 59-69, Jan. 1982.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, v. 78, n. 9, p. 1464-1480, Set. 1990.

KOHONEN, T. **Self-organizing Maps**. 3. ed. Springer, 2001, 501 p.

KOHONEN, T.; HYNNINEN, J.; KANGAS, J.; LAAKSONEN, J. SOM PAK: The Self-Organizing Map program package. Relatório Técnico A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Jan. 1996.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. The digital Michelangelo project: 3D scanning of large statues. In: SIGGRAPH '00. **Proceedings of the 27th annual conference on Computer graphics and interactive techniques**. ACM Press/Addison-Wesley Publishing Co., p. 131-144, 2000.

LIOU, C. Y.; KUO, Y. T. Conformal self-organizing map for a genus-zero manifold. **The Visual Computer,** Springer Berlin / Heidelberg, v. 21, n. 5, p. 340-353, Jun. 2005.

MAA. The Mathematical Association of America. Disponível em: <http://www.maa.org/cvm//1998/01/tprppoh/article/Glossary/Genus.html>. Acesso em: 14 jun. 2006.

MALISKA, C. R. **Transferência de calor e mecânica de fluidos computacional**. LTC – Livros Técnicos e Científicos Editora, 1995.

MÄNTYLÄ, M. **An Introduction to Solid Modeling**. Computer Science Press, Rockville, MD, 1988.

MARTINEZ, T. M. Competitive Hebbian learning rule forms perfectly topology preserving maps. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, Amsterdam: Springer, p. 427-434, 1993.

MARTINEZ, T. M.; SCHULTEN, K. J. A "neural gas" network learns topologies. In: KOHONEN, T.; MÄKISARA, K.; SIMULA, O.; KANGAS, J.; editores. **Artificial Neural Networks**, Amsterdan: North-Holland, p. 397-402, 1991.

MARTINEZ, T. M.; SCHULTEN, K. J. Topology representing netwoks. **Neural Networks**, v. 7, n. 3, p. 507-522, 1994.

MEYERS, D.; SKINNER, S.; SLOAN, K. Surface from contours. **ACM Transactions On Graphics**, v. 11, n. 3, p. 228-258, Jul. 1992.

MILLER; AMIDI; DELOUIS. Arctic Test Flights of the CMU Autonomous Helicopter. In: **Proceedings of the Association for Unmanned Vehicle Systems International 1999**, 26th Annual Symposium, Jul., 1999.

NONATO L. G.; MINGHIM R.; OLIVEIRA M. C. F.; TAVARES G. A novel approach for delaunay 3d reconstruction with a comparative analysis in the light of applications. **Computer Graphics Forum**, v. 20, n. 2, p. 161-174, Jun. 2001.

OBERMAYER, K.; RITTER, H.; SCHULTEN, K. Development and spatial structure of cortical feature maps: a model study. ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, San Mateo, CA: Morgan Kaufmann, v. 3, p. 774-780, 1991.

RITTER, H.; MARTINEZ, T.; SCHULTEN, K. **Neural computation and self-organizing maps:** an introduction. New York: Addison-Wesley Publishing Company, 1992. 306 p.

RITTER, H. Self-organizing maps in non-euclidean spaces. In: E. OJA; KASKI, S., **Kohonen maps**, Springer, p. 97-108, 1999.

ROGERS, C. A. Hausdorff Measures. Cambridge, England: Cambridge University Press, 1999.

ROWLAND, T. "Manifold." From *MathWorld*--A Wolfram Web Resource, criado por WEISSTEIN, E. W. Disponível em: <http://mathworld.wolfram.com/Manifold.html>. Acesso em: 14 Jun. 2006.

RUMELHART, D. E.; ZIPSER, D. Feature discovery by competitive learning. **Cognitive Science**, v. 9, p. 75-112, 1985.

SALEEM, W. **A flexible framework for learning-based surface reconstruction**. 2004. 124 f. Dissertação (Mestrado em Ciência da Computação – Computer Science Department – University of Saalard), Dez 2004.

SINHA, S. S., JAIN, R. Range image analysis. In: YOUNG, T. Y. **Handbook of pattern recognition and image processing:** computer vision. Califórnia: Academic Press, Inc., 1994. p. 185-237.

SON S.; PARK H.; LEE K.H. Automated laser scanning system for reverse engineering and inspection. **International Journal Of Machine Tools And Manufacture**, Elsevier Science, v. 42, n. 8, p. 889-897, Jun. 2002,

STANFORD UNIVERSITY. The Stanford 3D Scanning Repository. Disponível em: <http://graphics.stanford.edu/data/3Dscanrep/>. Acesso em: 14 jun. 2006.

STEPANOV, A.; LEE, M. The Standard Template Library. Relatório Técnico 95-11(R.1), HP Laboratories, nov 1995.

STROUSTRUP, B. The C++ programming language. 3. ed. Addison-Wesley Professional, 2000, 1030 p.

SUN Y.; DUMONT C.; ABIDI, M. A. Mesh-based Integration of Range and Color Images. In: AEROSENSE: AEROSPACE/DEFENSE SENSING AND CONTROLS, SENSOR FUSION: ARCHITECTURES, ALGORITHMS, AND APPLICATIONS. **Proceedings of SPIE AeroSense**, v. 4051, p. 110-117, 2000.

TIZIANI, H. J. Optical Metrology of Engineering Surfaces-Scope and Trends. In: RASTOGI P. K.. **Optical Measurement Techniques and Applications.** Norwood: Artech House Publishers, Jul. 1997. p. 15-22.

VARGAS, A. J. Q.; NONATO, L. G.; OLIVEIRA, M. C.; MINGHIN, R. Beta-connection: an approach to generate families of models from planar sections. In: SIBGRAPI ’02, p. 187-194, 2002.

VESANTO, A. J. Q. **Using SOM in data mining**. Abr. 2000. 50 f. Tese (Licenciatura) – Department of Computer Science and Enginnering, Helsinki University of Technology, Espoo, Finland, 2000.

VESANTO A. J. Q.; HIMBERG J.; ALHONIEMI E.; PARHANKANGAS J. SOM Toolbox for Matlab 5. Tech. Rep. A57, Helsinki University of Technology, Abr. 2000.

WEILER, K. Edge-based data structures for solid modeling in a curved surface environment. **IEEE Comput. Graph. Appl.,** 5(1):21-40, 1985.

XI, F.; SHU, C. CAD-based path planning for 3-D line laser scanning. **Computer-Aided Design**, Elsevier Science, v. 31, n. 7, p. 473-479, Jun. 1999.

YU, Y. Surface reconstruction from unorganized points using self-organizing neural networks, In: IEEE VISUALIZATION '99, **Proceedings of IEEE Visualization '99**, San Francisco, p. 61-64, Out. 1999.

ZORIN, D.; SCHRÖDER, P.; SWELDENS, W. Interpolating subdivision for meshes with arbitrary topology. In: SIGGRAPH '96. **Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques.** ACM Press, New York, NY, p. 189-192, 1996.