João Paulo Peçanha Navarro de Oliveira

Método Iterativo Para Geração de Malhas Triangulares Com Distribuição Uniforme

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Orientador: Prof. D.Sc. Marcelo Bernardes Vieira

Coorientador: Marcelo Lobosco

Coorientador: Sócrates de Oliveira Dantas

Juiz de Fora

2012

de Oliveira, João Paulo Peçanha Navarro

Método Iterativo Para Geração de Malhas Triangulares Com Distribuição Uniforme/João Paulo Peçanha Navarro de Oliveira. – Juiz de Fora: UFJF/MMC, 2012.

XI, 63 p. 29, 7cm.

Orientador: Marcelo Bernardes Vieira

Coorientador: Marcelo Lobosco

Coorientador: Sócrates de Oliveira Dantas

Dissertação (mestrado) – UFJF/MMC/Programa de Modelagem Computacional, 2012.

Referências Bibliográficas: p. 59 – 63.

1. Geometria Computacional. 2. Malhas Triangulares. 3. Otimização Linear. I. Vieira, Marcelo Bernardes *et al.*. II. Universidade Federal de Juiz de Fora, MMC, Programa de Modelagem Computacional.

João Paulo Peçanha Navarro de Oliveira

Método Iterativo Para Geração de Malhas Triangulares Com Distribuição Uniforme

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Aprovada em 1 de Janeiro de 2012.

BANCA EXAMINADORA

Prof. D.Sc. Marcelo Bernardes Vieira - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Marcelo Lobosco
Universidade Federal de Juiz de Fora

Prof. D.Sc. Sócrates de Oliveira Dantas
Universidade Federal de Juiz de Fora

Prof. D.Sc. Alexandre Fontes da Fonseca
Universidade Estadual Paulista

Prof. D.Sc. Saul de Castro Leite
Universidade Federal de Juiz de Fora

Dedico este trabalho a minha querida avó materna Dagmar Maria de Almeida e Silva, que recentemente nos deixou.

AGRADECIMENTOS

Antes de mais nada agradeço aos meus pais, João e Eliane, que sempre apoiaram as minhas decisões, em todas as circunstâncias. É impossível mensurar o quanto eles foram importantes para minha formação e, mais que isso, como são importantes na minha vida. Aos meus irmãos Joane e Matheus só tenho a agradecer todo o carinho e atenção que dedicam a mim, sobretudo nos momentos mais difíceis. Agradeço também aos meus avós Helvécio, Josepha e Dolores por sempre terem se preocupado com minha educação fazendo todo o possível para a minha melhor formação.

Agradeço ao meu amor Suelen por toda paciência, sobretudo nos momentos de maior stress, durante esses quase 30 meses de mestrado. Sua participação nesse processo foi muito importante. Durante esse período ela exerceu muito bem sua profissão de psicóloga em um paciente muito complicado (eu).

Também não posso esquecer de agradecer aos ex-parceiros de faculdade, que agora são bons amigos, todo apoio motivacional e incentivo que não me deixaram desistir quando quase pensei. Foram importantes também no aspecto técnico, com horas de discussão sobre implementações e toda parafernália técnica necessária nesse trabalho. Meu obrigado aos jedis: Edão, J.P. Scoralick, Tássio e Thales McLeary. Que a força esteja com vocês. Agradeço também aos brothers da antiga, Antonio, Daniel e Marcim por sempre estarem por perto quando mais preciso.

Agradeço a UFJF e ao mestrado pela ótima formação que me proporcionaram.

Meu muito obrigado aos professores Alexandre e Saul por terem aceitado participar da minha banca mesmo sendo convidados tão em cima da hora.

E por último aos professores que guiaram minha formação acadêmica de forma tão competente. Eu era só um menino perdido na faculdade quando no 3º período me deram a oportunidade de começar a pesquisar e entrar para o mundo acadêmico. Hoje, quase cinco anos depois, me sinto plenamente formado para encarar qualquer problema em minha área. O meu muito obrigado aos professores Marcelo Bernardes, Marcelo Lobosco e Sócrates Dantas.

"I was too weak to give in

Too strong to lose

My heart is under arrest again

But I'll break loose

My head is giving me life or death

But I can't choose

I swear I'll never give in

I refuse"

Foo Fighters

RESUMO

A aproximação de superfícies contínuas através de malhas poligonais é importante em várias áreas do conhecimento. Esse tipo de malha é empregado em aplicações como simulações computacionais de engenharia e física, modelagem geométrica e animações. Os modelos de entrada muitas vezes apresentam baixa qualidade, seja na distribuição de seus elementos ou no alinhamento e forma dos polígonos. Neste trabalho é apresentado um método para recobertura de malhas triangulares dado um comprimento de aresta m. A malha de entrada é uma superfície triangular de variedade-2 com topologia e geometria arbitrária, com ou sem borda. Definido o comprimento de aresta alvo m, o algoritmo remove e insere vértices de acordo com um critério, ajustando a quantidade necessária de elementos que o objeto deve conter. Após esta etapa, o modelo entra em uma fase de relaxamento global utilizando uma variação do operador discreto de Laplace-Beltrami, que na formulação aqui proposta utiliza os k primeiros vizinhos de cada vértice, ao contrário da definição clássica que usa apenas os vizinhos mais próximos. Isto é feito de maneira iterativa até que se esgote o número máximo de iterações fornecido no início do processo. Ao final, tem-se uma malha com comprimento de aresta próximo a m e com baixo desvio padrão, i.e., vértices uniformemente distribuídos sobre o modelo; seus triângulos também tendem a ser equiláteros. Os resultados do remalhamento se mostraram quantitativamente satisfatórios, com baixo desvio padrão do comprimento das arestas. O espaço dual pode ser utilizado para geração de malhas trivalentes, compostas majoritariamente por hexágonos em superfícies de baixa curvatura.

Palavras-chave: Geometria Computacional. Malhas Triangulares. Otimização Linear.

ABSTRACT

The approximation of continuous surfaces by polygonal meshes is very important in several areas. In recent years we have seen the use of this kind of mesh in various industrial applications such as computer simulations, geometric modeling and animation. Often, the input models have poor quality, i.e., bad elements distribution and inconsistent polygon shape. One important feature of poligonal surfaces is vertex distribution and sometimes it is necessary to control the vertices distance with an uniform edge lentgh. This work presents a method for triangular remeshing given a target edge length m. The input is a 2-manifold triangular surface with arbitrary geometry and topology. The proposed algorithm removes and inserts vertices adjusting the amount of necessary elements. Then, the model enters in a global relaxation process using a variation of Laplace-Beltrami discrete operator, which uses the k nearest neighbors of each vertex. This is done in an iteratively fashion until the algorithm reaches the maximum iteration number. At the end, one have a grid with edge length near to m and low standard deviation (vertices uniformly distributed). The triangles also presents good connectivity and high isotropy; the results shown that our remeshing scheme quantitatively improved the mesh quality. The dual space of the final triangular surface can be used for trivalent remeshing. These models are important for physical simulations of nano-carbon surfaces and we dedicated a chapter of our work to discuss this subject.

Keywords: Computational Geometry. Triangular Meshes. Linear Optmization.

SUMÁRIO

1 Introdução

A representação digital de objetos reais é importante para diversas aplicações em computação gráfica. A discretização dos objetos de interesse é um componente chave em simulações computacionais, animações e visualização de dados. Destacamos duas principais abordagens para geração de modelos tridimensionais: modelagem direta (ou *design*), utilizando programas de edição como aplicações CAD (*computer-aided design*) e geração a partir de digitalizadores tridimensionais.

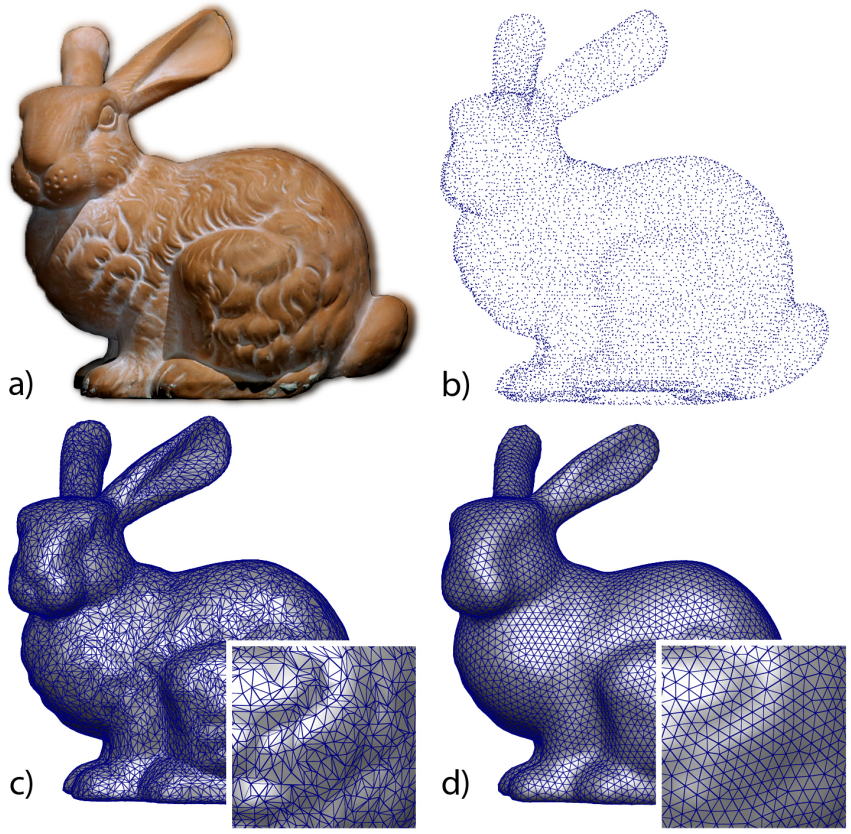

Figura 1.1: a) objeto real para digitalização b) nuvem de pontos obtida a partir do modelo original c) primeira triangulação do modelo d) modelo após o processamento utilizando o método proposto.

As técnicas de escaneamento geram uma discretização do domínio tipicamente em forma de pontos localizados sobre a superfície. O conjunto de dados gerados é normalmente referenciado como *nuvem de pontos* (Figura 1.1.b). A etapa de aquisição se interessa em obter um conjunto de amostras (pontos) fiéis ao objeto analisado, não se preocupando com a distribuição e conformidade dos mesmos. A maneira mais comum de

se combinar as informações obtidas nesta etapa é utilizando malhas poligonais. A Figura 1.1.c mostra a superfície final gerada a partir da *triangulação* da nuvem de pontos. Em uma malha triangulada, a conexão entre vértices utilizando-se de arestas e faces define uma interpolação linear que aproxima a superfície da melhor forma possível.

Os métodos de triangulação tendem a se preocupar mais com a fidelidade da superfície final do que com a regularidade, isotropia ou tamanho dos triângulos. Mesmo os modelos gerados por programas CAD podem apresentar vértices mal amostrados, redundantes ou triângulos com formas incoerentes. Para controlar e corrigir a forma final dos triângulos e a distribuição dos vértices na malha, mantendo a fidelidade com o modelo original, é necessário incluir na *pipeline* de processamento uma nova etapa, conhecida como remalhamento (*remeshing*) [1]. A Figura 1.1.d mostra o modelo *bunny* após sofrer remalhamento com distribuição uniforme desenvolvido neste trabalho.

Malhas triangulares altamente irregulares, como na Figura 1.1.c, são pouco úteis em aplicações de engenharia e de física. A confusa relação de adjacência entre vértices e triângulos podem ocasionar erros numéricos e comprometer o desenvolvimento do trabalho. Para isso, malhas regulares, em termos de sua conectividade e distribuição de vértices, são mais interessantes nessas aplicações.

1.1 Definição do Problema

Seja $\mathcal{M}$ uma superfície triangular bidimensional imersa no espaço $\mathbb{R}^3$ com *genus* e topologia arbitrários e de varidade-2. A malha $\mathcal{M}$ pode ser descrita como $\mathcal{M} = (\mathcal{V}, \mathcal{A}, \mathcal{P})$, onde $\mathcal{V}, \mathcal{A}$ e $\mathcal{P}$ é o conjunto de vértices, arestas e polígonos pertencentes à essa malha, respectivamente.

Dados $\mathcal{M}$ e a dupla (m, σ), com $m, \sigma \in \mathbb{R}$, desejamos obter uma malha $\mathcal{M}'$ com as seguintes características:

- A distribuição dos vértices sobre essa malha deve ser o mais uniforme possível: $\forall v_i, v_j \in \mathcal{V}'$, com $v_i \neq v_j$, tal que j está no conjunto de primeiros vizinhos de v_i, a distância entre os vértices, $\|v_i - v_j\|$, deve ter média próxima a m e variância σ^2.
- O *genus* de $\mathcal{M}'$ deve ser o mesmo de $\mathcal{M}$, ou seja, a topologia do objeto original não deve ser modificada.

- Geometricamente, $\mathcal{M}'$ deve ser o mais próximo possível de $\mathcal{M}$, i.e., $\|\mathcal{M} - \mathcal{M}'\|$ deve ser baixo. O termo $\|\mathcal{M} - \mathcal{M}'\|$ simboliza a diferença geométrica e volumétrica entre as duas malhas e é zero se $\mathcal{M} = \mathcal{M}'$.

1.2 Objetivos

O objetivo primário deste trabalho é apresentar um método que seja capaz de uniformizar as distâncias dos vértices em uma malha triangular. Este método deve manter a topologia e preservar ao máximo a geometria inicial do objeto. É importante, porém não essencial, que as curvaturas principais do modelo também sejam mantidas.

Como objetivo secundário destaca-se a geração de malhas trivalentes com distribuição uniforme de vértices. Malhas trivalentes são aquelas cujos vértices estão conectados a somente três vizinhos. Essas malhas são muito importantes em simulações físicas e portanto será dedicado um capítulo deste trabalho para a análise e aplicação do método proposto neste tipo de malha.

1.3 Visão Geral da Proposta

Neste estudo, a solução do problema é encontrada de modo iterativo. Estabelecidos dois parâmetros k e n, o algoritmo remove e insere vértices seguindo um critério específico e, logo após, o modelo processado passa por uma etapa de relaxamento global utilizando informações dos k primeiros vizinhos de cada vértice. Este procedimento é repetido até que o limite de iterações n seja alcançado.

Primeiramente, uma lista de prioridades de arestas é montada, apresentando maior importância os elementos que mais se desviam da média, $d_i = |a_i - m|$, onde a_i é a i-ésima aresta de $\mathcal{A}$. A partir disso, tais arestas sofrerão operações estelares (Seção 2.2) de acordo com o seu tamanho: caso $|a_i| > m + \sigma$, a aresta será refinada (Seção 2.2.1), porém, se $|a_i| < m - \sigma$, ela será simplificada (Seção 2.2.2). Para uma maior estabilidade, as arestas que passarem por uma destas operações terão seus vértices adjacentes marcados e suas arestas serão removidas da lista de prioridades.

Quando a mesma estiver vazia, o algoritmo entrará em uma fase de relaxamento global. Os vértices serão realocados na malha para que se obtenha uma configuração de "menor energia" (distribuição uniforme), no entanto, sem que sua conectividade seja alterada.

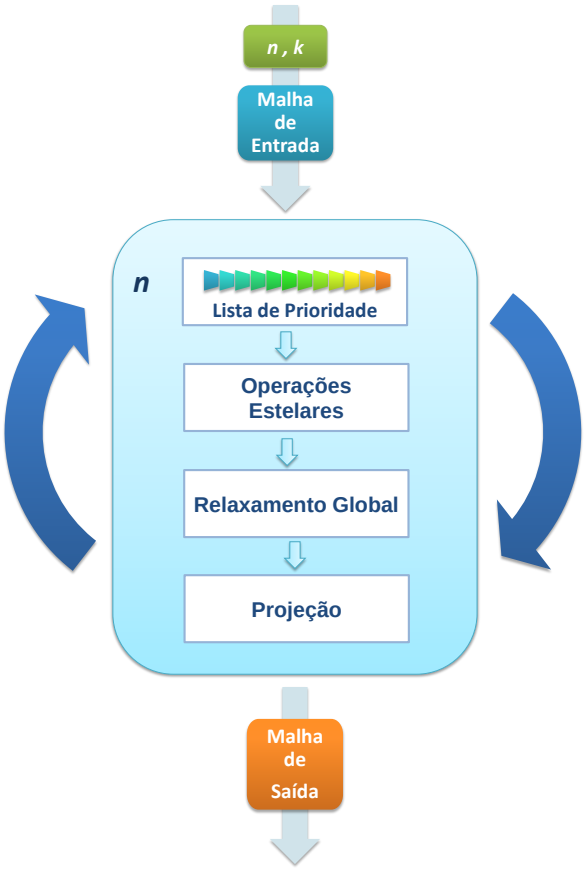

Figura 1.2: Visão geral do algoritmo.

Isto será feito via otimização linear; um modelo baseado no operador Laplaciano discreto (Seção 2.4) foi desenvolvido para que esse objetivo seja alcançado. A fidelidade com a malha original é preservada de forma satisfatória pois os vértices são movimentados tangencialmente ao modelo. Contudo, os vértice serão projetados de volta na superfície da iteração anterior.

O algoritmo termina quando o número máximo de iterações n é alcançado e neste momento, tem-se uma malha com comprimento de arestas próximos a m e será observado que os ângulos internos dos triângulos se aproximarão de 60°, de acordo com os experimentos realizados.

1.4 Trabalhos Relacionados

O remalhamento de superfícies é geralmente utilizado para melhorar a qualidade de uma malha. Essa qualidade é uma característica subjetiva e depende exclusivamente da aplicação que utilizará o modelo. A superfície pode ser modificada em termos da sua

quantidade de elementos e em alguns cenários é necessário que o modelo apresente poucos vértices e faces, mantendo a topologia e preservando a geometria original [2]. A qualidade também pode ser dada pela forma dos polígonos [3, 4], quantidade de vértices regulares [5, 6], alinhamento dos polígonos com as regiões de maior curvatura [7, 8, 9] e quantidade de elementos nessas regiões [5, 3], bem como pela distribuição dos vértices sobre o domínio [8].

Em uma distribuição uniforme, os vértices são igualmente amostrados por todo o modelo. As abordagens para se conseguir isso são dividas em dois grandes grupos: remalhamento explícito e parametrização global. No remalhamento explícito o modelo sofre modificações locais e/ou globais até que o objetivo seja alcançado. Surazhsky & Gotsman [5] desenvolveram um procedimento simples e eficaz para geração de malhas uniformes. A malha tem seus triângulos otimizados, em função da área, e em seguida passa por operações locais de regularização de vértices até que um mínimo global seja encontrado. Não é feito nenhum estudo sobre o comprimento das arestas, ou seja, a média da saída permanece a mesma da entrada. No artigo em questão foi introduzido o conceito de parametrização local, posteriormente formalizado por Ray *et al.* [10].

Em Botsch *et al.* [11] é apresentado um método para remalhamento uniforme explícito baseado em modificações locais seguido de relaxamento global. Valores superiores e inferiores são fixados e então as arestas fora deste limiar são tratadas por operações estelares, como realizado nesta dissertação. Embora eficiente, a abordagem não oferece liberdade na escolha desse limiar. Segundo os mesmos autores, o método garante sua funcionalidade apenas quando o comprimento de aresta desejado é muito próximo da média do modelo inicial, o que pode ser refutado, considerando a proposta do presente estudo. Também não são apresentados dados sobre a quantidade de vértices irregulares, desvio padrão ou médias finais das malhas testadas.

Remalhamento uniforme via parametrização global é uma abordagem elegante, porém computacionalmente lenta. Sendo assim, é comum restringir a topologia do objeto para modelos homeomorfos ao disco ou à esfera [12], dada a complexidade deste problema. Para superfícies arbitrárias um grafo de corte será calculado e somente então o modelo será aberto no plano [7]. A partir de uma parametrização global é possível redistribuir os vértices uniformemente de acordo com uma medida de distância [13, 8]. Percebe-se também a possibilidade de se reamostrar (e em seguida reconectar) os vértices de forma

que a nova superfície seja composta por polígonos não-triangulares como quadrados [7] ou hexágonos [9].

1.5 Contribuições

Existem diferenças entre esta dissertação e os trabalhos apresentados na subseção anterior. O algoritmo aqui proposto é capaz de controlar o comprimento de aresta do modelo com relativa precisão, o que não pode ser encontrado em [5, 11]. Embora a etapa de correção da forma dos triângulos (relaxamento) dos dois artigos sejam eficientes, no presente texto, foi proposta uma abordagem diferente da encontrada na literatura, utilizando Laplaciano e k-vizinhos. Além disso, nos artigos citados não é apresentada uma análise sobre o comprimento da aresta ou desvio padrão no modelo final, bem como também não se realiza um estudo das propriedades da malha durante o processamento. A seção de resultados apresentará com clareza todos esses dados e a partir deles serão discutidas as limitações do método.

As principais contribuições deste trabalho são:

- Apresentação de um algoritmo para uniformização de arestas em malhas triangulares, dado o comprimento de aresta e a variância desejados.
- Operações estelares baseadas em lista de prioridades. Elementos mais distantes do objetivo são ordenados e processados primeiro, visando uma maior estabilidade em operações deste tipo.
- Variação para o Laplaciano discreto com deslocamento tangente que não utiliza somente os primeiros vizinhos, como encontrado na literatura. A formulação proposta considera os k-primeiros vizinhos de um vértice, onde k é o número de saltos (ou arestas) que os separam.

2 Fundamentos

Esse capítulo tem por objetivo apresentar as definições matemáticas e os conceitos de geometria computacional necessários para o desenvolvimento das demais seções.

2.1 Definições e Conceitos

Uma *variedade* de dimensão δ é um espaço topológico onde, próximo de cada ponto, é possível remontar o espaço euclidiano δ-dimensional $\mathbb{R}^\delta$. Sendo assim, este trabalho tem interesse em variedades topológicas de dimensão 2, ou, daqui para frente, variedade-2.

Definição 1. *Uma variedade-2 (sem bordas) é um espaço topológico $\mathbb{X}$, tal que todo $x \in \mathbb{X}$ tem uma vizinhança aberta homeomorfa ao disco [14].*

Intuitivamente, esta definição indica que $\mathbb{X}$, localmente, tem a topologia do disco em todo seu domínio. Em computação gráfica, modelos tridimensionais são comumente representados como superfícies:

Definição 2. *Uma superfície $\mathcal{S}$ é uma variedade-2 compacta e orientável imersa em $\mathbb{R}^3$. Uma superfície sem bordas é chamada de superfície fechada, caso contrário, superfície aberta [14].*

Esta definição destaca que superfícies são objetos bidimensionais imersos em $\mathbb{R}^3$ e que somente as suas fronteiras exteriores são representadas, i.e., o interior do objeto é vazio.

Uma superfície é dita *compacta* se, e somente se, sua área for finita [15].

2.1.1 Representação de Superfície

Uma superfície pode ser representada, dentre outras maneiras, pela forma paramétrica [15] ou implícita [16]. Superfícies paramétricas são definidas por funções vetoriais $f : \Omega \to \mathcal{S}$ que mapeiam um espaço paramétrico bidimensional $\Omega \subset \mathbb{R}^2$ na superfície $\mathcal{S} = f(\Omega) \subset \mathbb{R}^3$. Na representação implícita, a superfície é descrita como o conjunto de zeros de uma função escalar $F : \mathbb{R}^3 \to \mathbb{R}$, i.e., $\mathcal{S} = \{x \in \mathbb{R}^3 | F(x) = 0\}$ [17].

Para formas mais complexas, muitas vezes não é possível descrever explicitamente uma única função que aproxime a forma desejada com a devida precisão. Assim, o domínio é

comumente dividido em sub-regiões e uma função é definida para cada um dos segmentos. Nesta definição *por partes*, cada função precisa aproximar a superfície apenas localmente, enquanto a definição global controla o número e tamanho dos segmentos [1]. A maneira mais comum de se definir uma superfície por partes é fragmentando o domínio Ω em polígonos bidimensionais, como triângulos, quadrados e hexágonos. A representação por partes de uma superfície será a usada neste trabalho.

2.1.1.1 Malhas Triangulares

Uma malha triangular é um coleção de triângulos sem nenhuma estrutura matemática particular. Entretanto, juntos, esses triângulos são capazes de descrever a geometria e topologia de uma superfície $\mathcal{S}$ com precisão. A princípio cada triângulo define, através de sua parametrização baricêntrica, um segmento da representação linear por partes da superfície. Cada ponto **p** no interior do triângulo [**a,b,c**] pode ser descrito de maneira única como uma combinação linear:

$$\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}, \tag{2.1}$$

onde

$$\alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0. \tag{2.2}$$

Escolhendo um triângulo arbitrário [**u,v,w**] no domínio paramétrico é possível encontrar um mapa $f : \mathbb{R}^2 \to \mathbb{R}^3$ com

$$\alpha\mathbf{u} + \beta\mathbf{v} + \gamma\mathbf{w} \quad \mapsto \quad \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}. \tag{2.3}$$

Esse mapeamento é suficiente para definir uma posição bidimensional para cada vértice. Assim, fica formalizada uma parametrização global para a malha [1].

Estruturalmente, uma malha triangular $\mathcal{M}$ pode ser representada por um grafo (complexo simplicial) $\mathcal{M} = (\mathcal{V}, \mathcal{A}, \mathcal{F})$. A malha $\mathcal{M}$ descreve a topologia da superfície pelo conjunto de vértices

$$\mathcal{V} = \{v_1, ..., v_V\} \tag{2.4}$$

e pelo conjunto de triângulos (faces)

$$\mathcal{F} = \{f_1, ..., f_F\}, \quad f_i \in \mathcal{V}^3. \tag{2.5}$$

A malha $\mathcal{M}$ ainda contém a relação de adjacência dos vértices, i.e., conjunto de arestas que os conectam:

$$\mathcal{A} = \{a_1, ..., a_A\}, \quad a_i \in \mathcal{V}^2. \tag{2.6}$$

Para superfícies imersas no espaço $\mathbb{R}^3$, associamos a cada vértice $v_i \in \mathcal{V}$ uma coordenada de posição $\mathbf{p}$:

$$\mathcal{P} = \{\mathbf{p}_1, ..., \mathbf{p}_V\}, \quad \mathbf{p}_i = \mathbf{p}(v_i) = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3. \tag{2.7}$$

É possível observar que mesmo após a associação de posições em $\mathbb{R}^3$ para os vértices, que são componentes *discretos* de $\mathcal{M}$, a superfície poligonal resultante continua sendo uma superfície *contínua* formada por funções lineares por parte (Eq. 2.3).

Os elementos que compõem uma malha poligonal fechada apresentam uma forte relação, dada pela *fórmula de Euler*. Essa fórmula, também conhecida como *característico de Euler* [18], associa o número de vértices, arestas e faces de uma malha fechada:

$$|\mathcal{V}| - |\mathcal{A}| + |\mathcal{F}| = 2(1 - g), \tag{2.8}$$

onde g é conhecido como o *genus* (gênero) da superfície, isto é, o número de furos que o objeto apresenta (Figura 2.1).

Como cada triângulo apresenta três arestas e cada aresta dois triângulos incidentes, as seguintes estatísticas da malha podem ser observadas:

- O número de triângulos é duas vezes o número de vértices: $|\mathcal{F}| \approx 2|\mathcal{V}|$.
- O número de arestas é três vezes o número de vértices: $|\mathcal{A}| \approx 3|\mathcal{V}|$.
- A média da valência dos vértices é 6.

2.1.2 Geometria dos Triângulos

Os triângulos são as primitivas poligonais mais utilizadas para representar uma malha em computação gráfica. Malhas compostas por triângulos são de fácil manuseio e apresentam

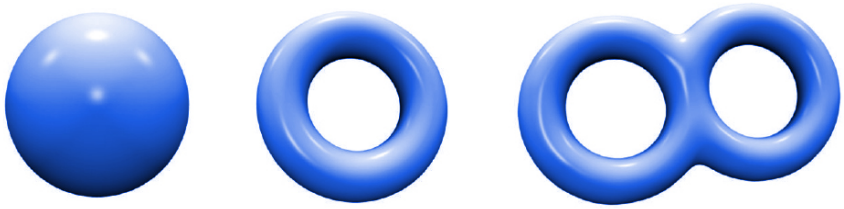

Figura 2.1: Esfera de *genus* 0, torus de *genus* 1 e torus duplo de *genus* 2. Imagem reproduzida de [17].

boa estabilidade, apesar da simplicidade deste elemento. Os pontos que formam um triângulo são coplanares, o que torna ainda mais interessante sua utilização por facilitar o cálculo de diversas propriedades como curvatura, planos tangentes e normais.

Os tipos de triângulos são:

- Equilátero ou equiângulo, todos os lados e ângulos iguais.
- Isósceles, pelo menos dois lados têm o mesmo tamanho.
- Escaleno, as medidas dos três lados são diferentes.
- Retângulo, contém um ângulo igual a 90º.
- Obtusângulo, possui um ângulo obtuso e dois ângulos agudos.
- Acutângulo, os três ângulos são agudos.

Seja [**a**,**b**,**c**] um triângulo qualquer em $\mathbb{R}^2$, como mostrado na Figura 2.2, a condição para sua existência é dada pela relação:

$$||\mathbf{b}-\mathbf{c}|-|\mathbf{c}-\mathbf{a}|| < |\mathbf{b}-\mathbf{a}| < |\mathbf{b}-\mathbf{c}|+|\mathbf{c}-\mathbf{a}| \tag{2.9}$$

Esta relação é importante para operações topológicas em malhas triangulares. Será mostrado que é possível remover ou adicionar vértices, arestas e faces da malha, sendo que estas operações serão válidas desde que os elementos gerados satisfaçam a Equação 2.9. Caso a mesma não seja atendida, a operação deve ser desfeita.

O ângulo α do triângulo da Figura 2.2 é dado por

$$\alpha = arccos\left(\frac{(\mathbf{b}-\mathbf{a}).(\mathbf{c}-\mathbf{a})}{\|\mathbf{b}-\mathbf{a}\|\,\|\mathbf{c}-\mathbf{a}\|}\right), \tag{2.10}$$

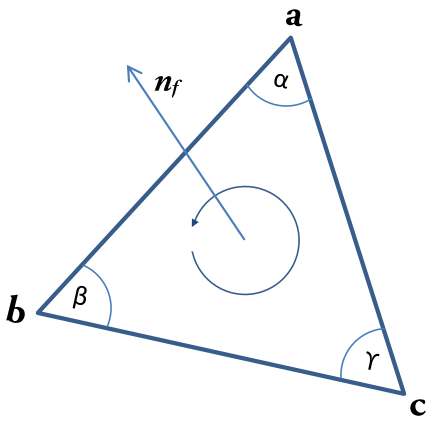

Figura 2.2: Triângulo [**a**,**b**,**c**].

e os demais são encontrados de maneira análoga. A normal da face, $\mathbf{n}_f$, é única para o triângulo e pode ser descrita como

$$\mathbf{n}_f = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}) \,. \tag{2.11}$$

2.1.2.1 Valência do Vértice

A *valência do vértice* é dada pela quantidade de arestas incidentes àquele vértice. Dado que um triângulo equilátero tem ângulos internos iguais à 60º, em um cenário ideal a valência de um vértice deve ser $360^o/60^o = 6$. Um vértice com valência 6 é chamado de *vértice regular* e vértices com valências diferentes de 6 são conhecidos como *irregulares* (Fig. 2.3) [5, 6]. É possível que uma malha seja composta somente por vértices regulares, entretanto, essa é uma forte restrição que pode introduzir distorções no modelo. Certamente a quantidade de vértices irregulares deve ser minimizado para que a estabilidade da malha seja aumentada, especialmente quando esta é aplicada em simulações de engenharia utilizando método dos elementos finitos ou simulações físicas de nanoestruturas. Denota-se $valence\,(v_i) \in \mathbb{N}$ a valência do vértice $v_i \in \mathcal{V}$.

2.1.2.2 Estrela do Vértice

Seja v_i um vértice de $\mathcal{M}$. A *estrela do vértice* de ordem k de v_i é o conjunto de vértices $\{v_1, ..., v_u\}$ que está separado de v_i por exatas k arestas. A primeira estrela de v_i é composta dos vértices adjacentes a este, $v_j \in \{v_1, ..., v_V\}$, com $V \leq valence(v_i)$, que compartilham uma mesma aresta. A segunda estrela de v_i, então, é o conjunto de vértices

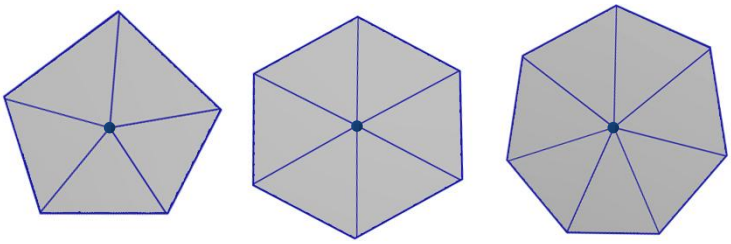

Figura 2.3: Vértices com valência 5 (esquerda), valência 6 (meio) e 7 (direita).

adjacentes a cada primeiro vizinho $v_j \in \{v_1, ..., v_V\}$. Denota-se $star_k(v_i) \subset \mathcal{V}$ o conjunto de vértices pertencentes à k-ésima estrela do vértice $v_i \in \mathcal{V}$. A Figura 2.4 destaca os vizinhos de ordem 1, 2 e 3 do vértice central (preto).

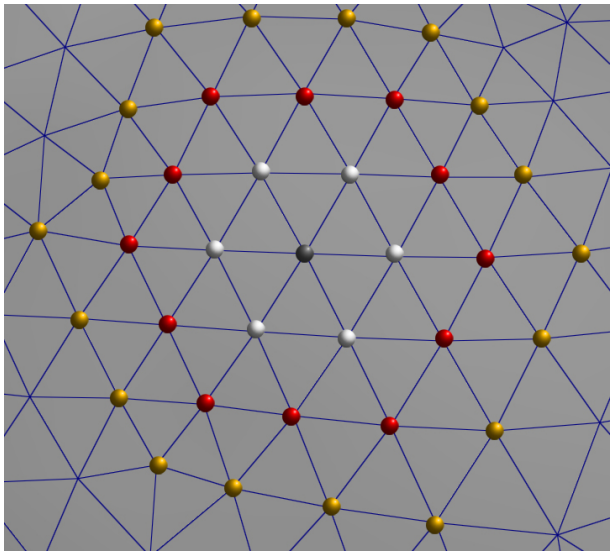

Figura 2.4: As k-estrelas, com k = 3, do vértice preto: 1-estrela são os vértices em branco, 2-estrela são os vértices em vermelho e 3-estrela os vértices em amarelo.

2.2 Operações Estelares

Em muitos casos existe a necessidade de se controlar as propriedades de uma malha triangular, como alterar seu número de elementos, corrigir a valência de seus vértices ou mesmo manipular o comprimento de suas arestas. Nesta seção será mostrado como manipular a estrutura de uma malha triangular $\mathcal{M}$ sem que sua topologia seja alterada. As operações capazes de realizar esse tipo de modificação são conhecidas como *operações estelares* [19] e é possível apresentá-las utilizando conceitos de topologia como linguagem descritiva [20]. Contudo, será dada uma interpretação mais intuitiva para cada um dos movimentos. As

operações serão organizadas de acordo com a natureza de sua modificação: refinamento, simplificação ou neutra.

2.2.1 Refinamento

As operações de refinamento compreendem os movimentos que causam um aumento no número de elementos em uma malha triangular. Entre as mais importantes destaca-se:

- Divisão de Aresta (*Edge Split*). Existem quatro vértices envolvidos em um *edge split*. Quando uma aresta é dividida, um novo vértice é adicionado em seu ponto médio e duas novas faces serão adicionadas na malha. A valência dos vértices opostos à aresta selecionada é aumentada em 1 (Figura 2.5) [20].

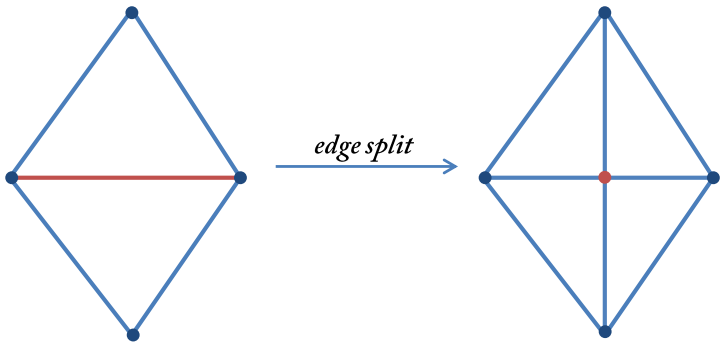

Figura 2.5: *Edge split*.

- Divisão de Vértice (*Vertex Split*). Três ou quatro vértices são envolvidos. Para realizar uma divisão de vértice, duas arestas incidentes ao vértice dividido v_w precisam ser selecionadas. Estas separam as demais em dois grupos contendo $g_1, g_2 \in \mathbb{N}$ arestas. Após a divisão, serão adicionados à malha duas faces e um vértice v_u. A valência de dois vértices será aumentada em 1 e a valência dos vértices centrais v_w e v_u será dada por: $valence(v_w) = g_1 + 3$ e $valence(v_u) = g_2 + 3$ (Figura 2.6) [6].

2.2.2 Simplificação

As operações de simplificação são utilizadas parar remover (simplificar) elementos de uma malha, dentre as quais destaca-se:

- Colapso de Aresta (*Edge Collapse*). Quatro vértices estão envolvidos nessa operação. Uma aresta será selecionada e então as duas faces que a compartilham serão re-

movidas. Dos vértices incidentes a esta (v_w e v_u), apenas um v'_w restará ao final. Após o *edge collapse* as valências de dois vértices serão diminuídas em 1 e $valence(v'_w) = valence(v_w) + valence(v_u) - 4$ (Figura 2.6) [1, 6].

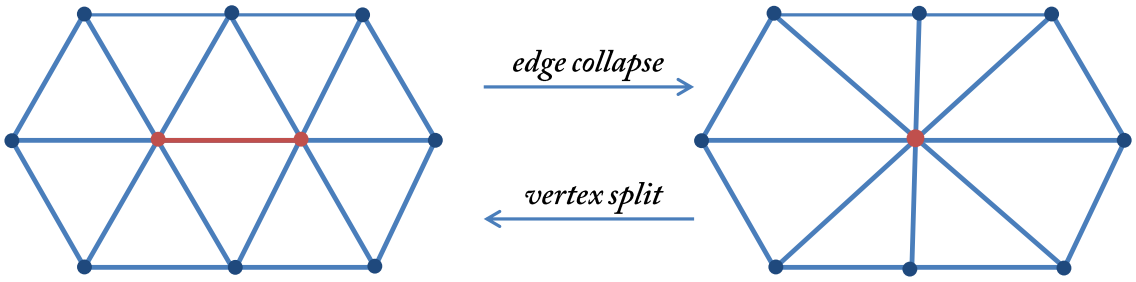

Figura 2.6: *Edge Collapse* e *Vertex Split*.

- Simplificação de Face (*Face Split*). Sejam $v_w \in \mathcal{V}$ e $u = valence(v_w)$. Nesta operação, v_w e u faces são removidos e um novo polígono de u lados é adicionado à malha. Note que para manter a consistência de $\mathcal{M}$, v_w deve ter $valence(v_w) = 3$, caso contrário, um polígono com lado diferente de 3 passará a fazer parte do modelo (Figura 2.7).

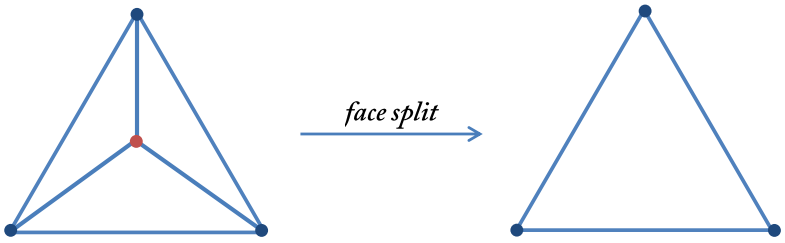

Figura 2.7: *Face Split*.

2.2.3 Operação Neutra

Chamou-se de operação neutra o movimento que não adiciona ou remove vértices e faces de uma malha. Um giro de aresta (*edge flip*) é um exemplo de operação que satisfaz essa condição. Dado quatro vértices v_1, v_2, v_3 e v_4, e uma aresta $\overline{v_1v_3}$, após a realização do *edge flip* uma nova aresta $\overline{v_2v_4}$ será criada e a anterior removida (Figura 2.8). A valência de v_1

e v_3 será diminuída em 1, e a de v_2 e v_4 aumentada em 1. De acordo com de Berg *et al.* [21], uma operação de giro de aresta não pode aumentar o ângulo mínimo dos triângulos envolvidos: sejam $f_1 = (v_1, v_3, v_4)$ e $f_2 = (v_1, v_2, v_3)$ dois triângulos contendo os ângulos $\{\alpha_1, ..., \alpha_6\}$, como mostrado na Figura 2.8 (triângulo da esquerda). Em uma operação de *edge flip* duas novas faces f_1' e f_2' serão geradas e com elas um novo conjunto de ângulos $\{\alpha_1', ..., \alpha_6'\}$. Chamamos a nova aresta $a_f = \overline{v_2 v_4}$ de *aresta ilegal* se

$$min\left(\alpha_i\right) < min\left(\alpha_i'\right), \quad 1 \leq i \leq 6, \tag{2.12}$$

caso seja constatado que a_f é ilegal, a operação deve ser desfeita.

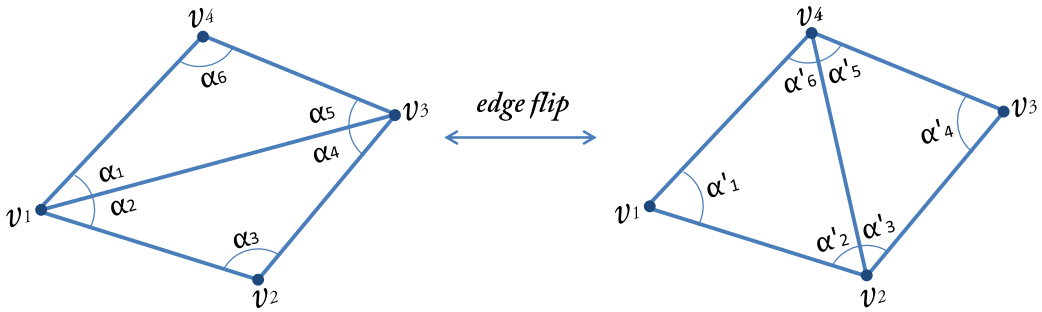

Figura 2.8: Girando uma aresta.

Outra modificação comumente utilizada em processamento de malhas é a realocação de vértices. A única alteração promovida por esse movimento diz respeito à localização do vértice. Dado um vértice v_w e seu vetor posição $\mathbf{p}\left(v_w\right) \in \mathbb{R}^3$, uma operação de realocação irá gerar um novo $\mathbf{p'}\left(v_w\right) \in \mathbb{R}^3$ sem que o número de vértices e faces seja alterado (Figura 2.9).

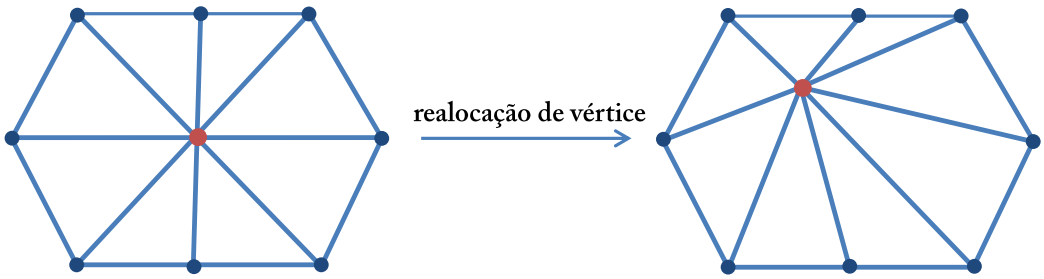

Figura 2.9: Realocação de vértice.

2.3 Remalhamento

Remalhamento (tradução livre para *remeshing*) é uma técnica chave para melhorar a qualidade de malhas usadas em aplicações industriais como simulação computacional e modelagem geométrica. Dois objetivos principais podem ser destacados neste assunto: o primeiro diz respeito a redução da complexidade da malha de entrada sobre algum aspecto; este processo é conhecido como *simplificação de malha* [2]. O segundo visa promover um melhoramento na qualidade da malha para que esta possa ser usada em aplicações comerciais. Todavia, diferentes aplicações demandam diferentes critérios de qualidade.

Botsch *et al.* [1] e Surazhsky & Gotsman [5] propuseram a seguinte definição para remalhamento:

Definição 3. *Dada uma superfície triangular representada por $\mathcal{M}$, calcule uma nova malha $\mathcal{M}'$, tal que seus elementos satisfaçam um determinado critério de qualidade e $\mathcal{M}'$ se aproxime de $\mathcal{M}$ de forma aceitável.*

Pode-se perceber que a definição é subjetiva quanto à noção de aproximação de $\mathcal{M}'$ em respeito a $\mathcal{M}$. Geometria, topologia, curvaturas e normais são características que podem ser levadas em consideração na verificação da diferença entre a superfície inicial e final.

A qualidade da malha refere-se às propriedades estruturais da mesma como regularidade, tamanho, densidade de amostragem, orientação, alinhamento e forma dos elementos que a compõem [1].

2.3.1 Estrutura Local

A estrutura local da malha é descrita pelo tipo, forma, orientação e distribuição de seus elementos.

- *Tipo.* Os elementos mais comuns presentes em malhas poligonais são triângulos [2] e polígonos quadrangulares [7]. Malhas triangulares são fáceis de se gerar e apresentam uma estrutura de adjacência simplificada entre seus elementos. Superfícies quadrangulares, embora mais complexas, são interessantes para aplicações de modelagem geométrica e *design*.

- *Forma*. Os componentes de uma malha podem ser classificados como isotrópicos [3] ou anisotrópicos [22]. Os elementos isotrópicos são localmente uniformes em todas as direções. Um triângulo é isotrópico quando sua forma se aproxima de um triângulo equilátero. De forma análoga, polígonos quadrangulares altamente isotrópicos são os que têm forma semelhante a um quadrado. A Figura 2.10 ilustra a ideia.

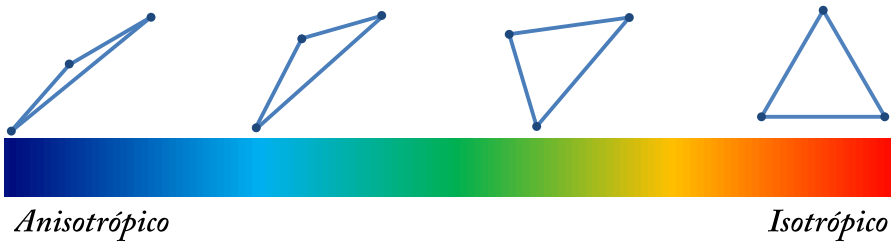

Figura 2.10: Isotropia de um triângulo.

- *Densidade*. Em uma distribuição uniforme, os elementos são igualmente dispostos ao longo de todo o domínio [8]. Uma distribuição não uniforme implica em uma maior concentração de vértices e faces em regiões de alta curvatura [5], no entanto, malhas de baixa qualidade podem apresentar uma não uniformidade na localização de seus elementos sem que ocorra uma maior concentração em áreas de curvatura acentuada.

- *Alinhamento e Orientação*. Converter uma superfície linear por partes em uma nova malha corresponde ao processo de reamostragem. Assim, regiões de alta curvatura podem ser afetadas ocasionando o aparecimento de artefatos visuais [9]. Visando diminuir esse efeito, os elementos devem seguir essas direções para que eles representem adequadamente as regiões tangentes descontínuas [1].

2.3.2 Estrutura Global

Um vértice em uma malha triangular é chamado regular se sua valência for igual à 6. Em malhas quadrangulares, seus elementos são regulares caso apresentem valência 4. Vértices que não atendem a essa demanda são conhecidos como irregulares. A estrutura global de uma malha pode ser classificada como irregular, semirregular, altamente regular ou regular [23].

- *Irregular*. São malhas que não apresentam nenhum tipo de regularidade em sua conectividade.

- *Semirregular*. Podem ser produzidas através de subdivisão regular de malhas iniciais de baixa qualidade. Apresentam poucos vértices regulares [24].

- *Altamente Regular*. Têm a maioria dos vértices regulares. Modificações locais através de operações estelares são suficientes para conseguir esse tipo de malha [5, 6]. Destacamos a parametrização global seguida de reamostragem de vértices no plano como uma outra maneira de se obter esse tipo de malha [8, 13, 7].

- *Regular*. Todos os vértices são regulares. Neste caso a malha pode ser armazenada de forma compacta em um vetor bidimensional [25].

2.4 Operador Laplaciano

O *Laplaciano* ∇^2 de uma função δ-dimensional $f : \mathbb{R}^\delta \to \mathbb{R}$ é dado por

$$\nabla^2 f = \frac{\partial^2 f}{\partial x_1^2} + ... + \frac{\partial^2 f}{\partial x_\delta^2} = \sum^{\delta} \frac{\partial^2 f}{\partial x_i^2}, \tag{2.13}$$

e descreve a taxa de dispersão em $\mathbb{R}^\delta$. Taubin [26] propôs em seu trabalho uma discretização do Laplaciano sobre uma função f, que descreve uma superfície qualquer, para aplicações em processamento de malhas. O operador Laplaciano discreto, denotado aqui como L, é aproximado linearmente em cada vértice por:

$$L(v_i) = \sum_{v_j \in star_1(v_i)} w_{ij} (v_i - v_j), \tag{2.14}$$

onde $star_1(v_i) \subset \mathcal{V}$ é a primeira estrela do vértice $v_i \in \mathcal{V}$ e w_{ij} é o peso da aresta $\overline{v_i v_j}$ com $\sum_{v_j \in star_1(v_i)} w_{ij} = 1$. Várias formas para w_{ij} foram propostas na literatura, como esquemas baseados em comprimento de aresta [26] e cotangentes [27].

O Laplaciano discreto foi primeiramente utilizado no processo de suavização de malhas. O algoritmo de suavização Laplaciana é relativamente simples: a posição de cada vértice v_i é alterada para a posição média dos vértices adjacentes. Assim, eles são movidos iterativamente para a "direção do Laplaciano" [26, 28]. A forma mais adequada de se implementar essa lógica é multiplicar o resultado obtido no Laplaciano por um $\Delta t \in \mathbb{R}$ com $\Delta t \leq 1$ e então adicionar essa fração de variação à posição atual do vértice, fazendo com que o resultado seja alcançado gradativamente [8].

Uma abordagem muito utilizada é a escolha de $w_{ij} = 1/valence\,(v_i)$ [28, 29]; o operador, nessa forma, é conhecido como Laplaciano discreto uniforme [1]. Neste, o vértice v_i se encontrará na média ponderada de sua 1-estrela ou, em outras palavras, estará posicionado no centro de gravidade dos seus primeiros vizinhos:

$$L\,(v_i) = \frac{1}{valence\,(v_i)} \sum_{v_j \in star_1(v_i)} (v_i - v_j)\,. \tag{2.15}$$

2.4.1 Laplaciano Como Otimização Linear

Na definição inicial do problema foi estabelecido que a distância de cada vértice em relação à seus vizinhos deve ter média m e variância σ^2. Logo, é possível descrever a variação da energia dos vértices sobre $\mathcal{M}$ como:

$$E\,(v_i) = \sum_{j \in star_1(v_i)} \|v_i - v_j\|^2 - m^2. \tag{2.16}$$

Percebe-se que a função de energia pode ser minimizada visando a solução do problema em relação à média. O mínimo de E_i será encontrado quando a derivada da Função 2.16 (chamada aqui de $\mathcal{L}$) for igual a zero, $\frac{\partial E_i}{\partial v_i} = \mathcal{L}\,(v_i) = \left(\frac{\partial E_i}{\partial v_{ix}}, \frac{\partial E_i}{\partial v_{iy}}, \frac{\partial E_i}{\partial v_{iz}}\right) = 0$. Antes desse desenvolvimento, note que a Equação 2.16 pode ser reescrita como

$$E\,(v_i) = \sum_{j \in star_1(v_i)} \|v_i - v_j\|^2 - m^2 = \tag{2.17}$$

$$\sum_{j \in star_1(v_i)} \langle v_i - v_j, v_i - v_j \rangle - m^2 = \tag{2.18}$$

$$\sum_{j \in star_1(v_i)} \langle v_i, v_i \rangle - 2\,\langle v_i, v_j \rangle + \langle v_j, v_j \rangle - m^2, \tag{2.19}$$

onde $<,>$ é o operador de produto interno.

Derivando a Equação 2.19 em função de v_i e igualando-a a zero tem-se

$$\frac{\partial E_i}{\partial v_i} = \sum_{j \in star_1(v_i)} 2v_i - 2v_j = 0 \tag{2.20}$$

$$\frac{1}{valence\,(v_i)} \sum_{j \in star_1(v_i)} (v_i - v_j) = 0 \tag{2.21}$$

$$\mathcal{L}\,(v_i) = 0. \tag{2.22}$$

A partir da Equação 2.22 todos os vértices podem ser representados na forma matricial

$$\mathcal{L}\mathbf{X} = 0, \tag{2.23}$$

onde $\mathcal{L}$ é uma matriz $r \times r$ e $\mathbf{X}$ um vetor $r \times 1$ representando as incógnitas. A montagem desta matriz será detalhada na Seção 3.2. Tem-se então definido os componentes necessários para a resolução do Laplaciano em função de equações lineares. Porém, o fato dessa matriz ser solucionada sem restrição não significa, necessariamente, que as novas posições se encontrarão sobre a superfície original $\mathcal{M}$.

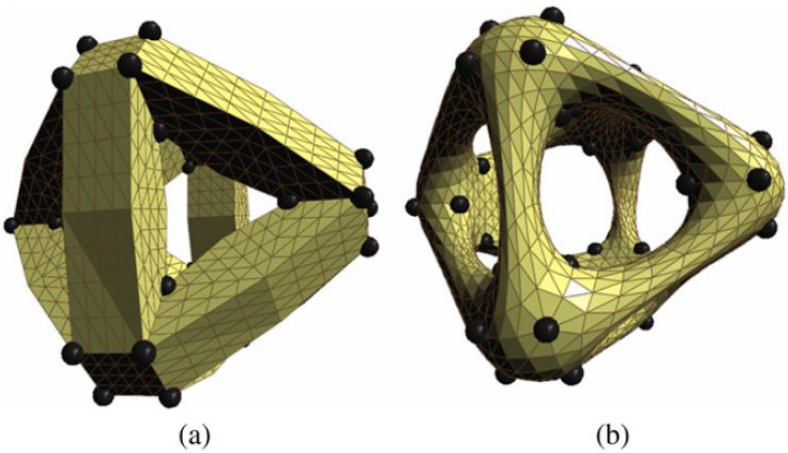

Figura 2.11: Aplicação do Laplaciano com restrição: a) superfície original b) vértices em preto são marcados como restrições do sistema e não têm suas posições alteradas. Imagem reproduzida de [28].

É preciso introduzir no sistema restrições que controlem as novas posições dos vértices como, por exemplo, a criação de pontos de controle [28]. Vértices que não terão suas posições alteradas serão selecionados e passarão a ser restrições do sistema linear. Essa é uma abordagem interessante para algumas aplicações mas, dependendo da escolha dos vértices, pode apresentar distorções indesejadas (Figura 2.11). No capítulo seguinte será

proposta uma restrição que mantém os vértices sobre a superfície original de maneira eficiente.

Como mostrado em [29, 28, 30], o resultado da Equação 2.22 corresponde ao Laplaciano uniforme discreto da Equação 2.15 para solução via otimização linear. A Equação 2.16 é uma forma coerente de se descrever a distribuição das energias dos vértices na malha para o problema desta dissertação. Entretanto, é observado que o resultado obtido após sua derivação é o Laplaciano discreto convencional. Sabe-se que este operador em sua forma natural (Eq. 2.15) não consegue distribuir uniformemente os vértices na malha de maneira satisfatória: os vértices com maior valência se apresentarão mais distantes uns dos outros (Figura 2.12 esquerda), enquanto os de menor permanecerão mais próximos (Figura 2.12 direita).

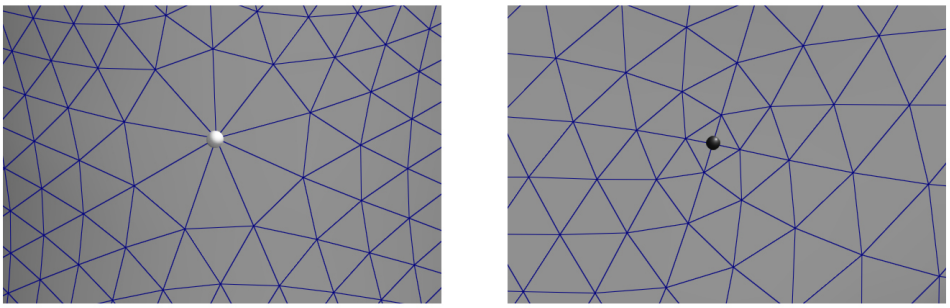

Figura 2.12: À esquerda, o vértice em branco apresenta valência 9 e após a suavização Laplaciana sua 1-estrela se afasta. O contrário acontece à direita, onde os primeiros vizinhos do vértice preto, com valência 4, se aproximam.

A seguir será apresentada uma variação do Laplaciano discreto que é capaz de amenizar o efeito observado na Figura 2.12. A correção da valência dos vértices também será levada em conta durante o processo.

2.5 Regularidade da Malha

Situações como a vista na Figura 2.12 devem ser evitadas em malhas uniformes. É possível suavizar as distribuições dos vértices utilizando uma variação do Laplaciano que considera os primeiros k-vizinhos de cada elemento. Entretanto, um componente importante neste tipo de malha e que também influencia neste comportamento é a valência dos vértices. Quanto mais regulares os vértices, maior a tendência da malha atingir uma média e menor o desvio padrão de suas distâncias. O efeito da Figura 2.12 induz a crer que quanto maior

$|valence\,(v_i) - 6|$, menos uniforme será a distribuição da estrela de v_i, o que de fato acontece. Então, deve-se buscar sempre uma configuração de valência de vértices que se aproxime de 6.

Todo vértice pode ser convertido em um ou mais vértices com 5, 6 ou 7 arestas adjacentes, via *vertex split* (Seção 2.2.1), como apresentado em [6]. Por exemplo, seja v_w ($valence(v_w) = 8$) tal que a sua 1-estrela é composta somente por vértices regulares, como na Figura 2.6 (direita). Após uma operação de *vertex split* dois vértices terão sua valência incrementadas em 1 e a valência do elemento que sofreu divisão será 6. Nota-se que é possível estender esse comportamento para os demais vértices v_i com $valence\,(v_i) > 7$.

3 Método Proposto

O algoritmo para uniformização de arestas recebe como entrada a tupla $(\mathcal{M}, m, \sigma, k, n)$, onde $\mathcal{M}$ é a malha triangular, m a distância entre os vértices (comprimento desejado de aresta), σ a variância, k o número de anéis para a etapa de otimização Laplaciana e n o número de iterações.

$\mathcal{M}'_0 = \mathcal{M}'_1 =$ `Copia(`$\mathcal{M}$`)`
for $i = 1$ **to** n **do**
  $m_i =$ `MIN(`*2*`CalculaMediaArestas(`$\mathcal{M}'_i$`),` m`)`
  $L_p =$ `CriaListaPrioridades(`$\mathcal{M}'_i$`,` m_i `,` σ`)`
  `OperacoesEstelares(`$\mathcal{M}'_i$`,` L_p`)`
  `CorrigeValencia(`$\mathcal{M}'_i$`)`
  `RelaxamentoGlobal(`$\mathcal{M}'_i$`,` k`)`
  `Projecao(`$\mathcal{M}'_{i-1}$`,` $\mathcal{M}'_i$`)`
end
return $\mathcal{M}'_n$

Algoritmo 1: RemalhamentoUniforme$(\mathcal{M}, m, \sigma, k, n)$

O esquema de funcionamento é como apresentado no Algoritmo 1. Este capítulo descreverá com detalhes as etapas realizadas começando pela montagem da lista de prioridades e operações estelares.

3.1 Transformações Estelares Com Lista de Prioridades

O comprimento de aresta alvo m é arbitrário. O número de elementos na malha deve se adaptar a essa demanda: quanto maior m, menor a quantidade de faces e vértices, enquanto no caso oposto, mais vértices e triângulos são necessários para representar o modelo. Qual seria o critério ideal para adicionar e remover vértices na malha? Fundamentalmente, o tamanho da aresta.

Existem casos onde o comprimento de aresta desejado m é consideravelmente maior que a média da malha fornecida. Para uma maior estabilidade é importante que ocorra uma transição suave entre a média das arestas do modelo no passo i e $i + 1$. A cada iteração, o comprimento de aresta alvo é alterado para uma variável local m_i que recebe

o mínimo entre a média final m e duas vezes a média das arestas de $\mathcal{M}'_i$ (Algoritmo 1). Dessa forma é possível garantir que a diferença das médias de $\mathcal{M}'_i$ e $\mathcal{M}'_{i+1}$ não assume valores que possam comprometer a estabilidade do algoritmo gerando triângulos inválidos, alterando a topologia da malha ou mesmo causando erros numéricos.

É intuitivo pensar que arestas demasiadamente longas devam ser refinadas. Arestas com grande comprimento precisam ser divididas e isto é possível, dentre outras maneiras, pela operação estelar de *edge split*. Para $a_l \in \mathcal{A} \subset \mathcal{M}'_i$ considerada longa, em sua posição média será fixado um novo vértice que a dividirá em duas novas arestas com comprimento $|a_l|/2$. De forma semelhante, as arestas curtas precisam ser simplificadas (removidas) do modelo, por exemplo, via *edge collapse*.

Portanto, é necessário definir as arestas *longas* e *curtas* da malha que sofrerão *edge split* e *edge collpase*, respectivamente. A classificação foi feita da seguinte forma: a aresta a_j é longa caso $|a_j| > m_i + \sigma$, e a_j será curta se $|a_j| < m_i - \sigma$. As arestas com comprimento no intervalo fechado $[m_i - \sigma, m_i + \sigma]$ permanecerão inalteradas. Este critério foi escolhido pois, assim, o algoritmo tende a convergir de maneira mais rápida para a média m.

```
listaPrioridade
foreach a_i ∈ A ⊂ M'_i do
    if |a_i| > m_i + σ then
        listaPrioridade.Adiciona(a_i)
    else if |a_i| < m_i − σ then
        listaPrioridade.Adiciona(a_i)
    end
end
OrdenaInversa(listaPrioridade)
return listaPrioridade
```

Algoritmo 2: CriaListaPrioridades($\mathcal{M}'_i, m_i, \sigma$)

Antes de se realizar as operações estelares é interessante estabelecer uma ordem, ou prioridade, para cada aresta fora do intervalo supracitado. As arestas podem ser organizadas de acordo com o tipo de operação que realizam ou pelo impacto que causam na média e desvio final da superfície processada. Um *edge split* ou *edge collapse*, por si só, não apresentam características fortes que justifiquem a aplicação de uma antes da outra. Então, o aspecto quantitativo das operações será considerado na montagem da lista de arestas que serão primeiro processadas. Assim, as arestas com mais importância serão as que apresentarem maior desvio $d_j = |a_j - m_i|$. Desta forma fica definido o critério para

geração da *lista de prioridades* L_p. Formalmente teremos L_p, com $|L_p| = t$, sendo um conjunto de arestas $\{a_1, ..., a_t\} \subset \mathcal{A} \subset \mathcal{M}'_i$, com $d_1 \geq d_2, ..., d_{t-1} \geq d_t$, onde $a_j \in \{a_1, ..., a_t\}$ se e somente se $|a_j| \notin [m_i - \sigma, m_i + \sigma] \in \mathbb{R}$ (Algoritmo 2).

Após a montagem de L_p, as operações estelares podem ser realizadas. O algoritmo percorre a lista e para cada elemento verifica o seu tipo (aresta longa ou curta) e então aplica o respectivo movimento (*edge split* ou *edge collapse*). Para aumentar a estabilidade do método e evitar que um vértice tenha mais do que uma aresta modificada, os vértices adjacentes a uma aresta que tenha sofrido alteração são marcados como visitado. Daí em diante, as arestas marcadas não sofrem nenhuma operação estelar na iteração corrente. O processamento termina quando a lista de prioridades for completamente percorrida (Algoritmo 3).

```
foreach a_i ∈ L_p do
    if VerticeVisitado(a_i) then
        continue
    else if |a_i| > m_i + σ then
        EdgeSplit(a_i)
        VisitaVerticesAresta(a_i)
    end
    else if |a_i| < m_i − σ then
        EdgeCollapse(a_i)
        VisitaVerticesAresta(a_i)
    end
end
```

Algoritmo 3: OperacoesEstelares($\mathcal{M}'_i, L_p$)

3.1.1 Pós-Processamento

O processamento seguindo a ordem de prioridade estabelecida por L_p tende a fazer com que a média desejada seja alcançada com mais eficiência. Contudo, nota-se que esse esquema não leva em conta a valência dos vértices. Observou-se que naturalmente o algoritmo organiza a valência de seus vértices em torno de 6. Para contribuir com este processo é possível realizar um esquema simples para correção da valência dos vértices baseado em operações de *edge flip*, como descrito em [1, 5, 11]. Considere os dois triângulos da Figura 3.1. Para cada aresta da malha, o algoritmo realiza uma operação de *edge flip* e

caso a valência dos vértices envolvidos neste movimento se aproxime de 6, a aresta girada é aceita, caso contrário, a operação é desfeita. O Algoritmo 4 detalha a ideia.

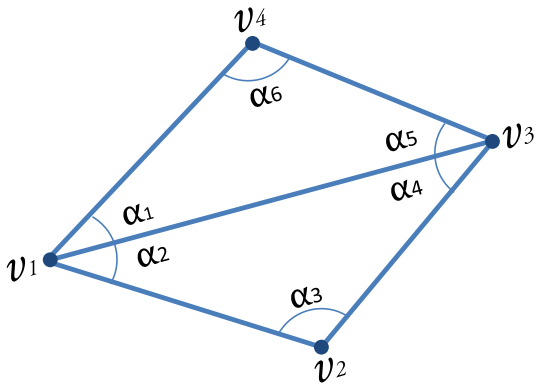

Figura 3.1: Vértices v_1, v_2, v_3 e v_4 formando dois triângulos $f_1 = (v_1, v_2, v_3)$ e $f_2 = (v_1, v_3, v_4)$. Os vértices adjacentes à aresta $a_i = \overline{v_1 v_3}$ são v_1 e v_3, e os vértices opostos a a_i são v_2 e v_4. O conjunto de vértices dos triângulos adjacentes à a_i é $\{v_1, v_2, v_3, v_4\}$.

foreach $a_i \in \mathcal{A} \subset M_i'$ **do**
  $v_n =$ `BuscaVerticesAdjacentes(`a_i`)`
  desvioPre $= |valence(v_1) - 6| + |valence(v_2) - 6| + |valence(v_3) - 6| + |valence(v_4) - 6|$
  `EdgeFlip(`a_i`)`
  desvioPos $= |valence(v_1) - 6| + |valence(v_2) - 6| + |valence(v_3) - 6| + |valence(v_4) - 6|$
  if desvioPre $\leq$ desvioPos **then**
    `EdgeFlip(`a_i`)`
  end
end

Algoritmo 4: CorrigeValencia($\mathcal{M}_i'$). Note que o segundo *edge collapse* desfaz o primeiro.

Na seção de resultados será discutida a evolução da valência dos vértices pelas iterações e serão apresentados os números que comprovam a eficiência do algoritmo de correção de valência.

3.2 Otimização Global Com *k*-vizinhança

Após a eliminação de arestas longas e curtas do modelo e da correção da conectividade dos vértices, é necessário redistribui-los de maneira adequada. É importante que a realocação

dos elementos aproxime ao máximo a geometria inicial, e isto pode ser feito através da movimentação tangencial dos pontos. Será imposto ao sistema linear uma restrição que força os vértices a se movimentarem apenas em seu espaço tangente. Para cada vértice é indicado a seguinte formulação de energia:

$$E_i = \sum_{j \in star_k(v_i)} w_{ij} \left\| v_i + \mathbf{t}_i - v_j - \mathbf{t}_j \right\|^2, \tag{3.1}$$

onde $\mathbf{t}_i$ e $\mathbf{t}_j$ são, respectivamente, os vetores tangentes a v_i e v_j que promoverão o deslocamento apenas no espaço de interesse e w_{ij} a função de ponderação baseada nos k primeiro vizinhos. Vetores tangentes a vértices de uma superfície triangular podem ser encontrados através da normal de cada v_i, como mostrado adiante (Eq. 3.4).

Foi dito (Seção 2.4) que o operador Laplaciano uniforme, que se utiliza apenas da 1-estrela do vértice, não é capaz de distribuir de maneira satisfatória os vértices irregulares. Em vez de se utilizar apenas os primeiros vizinhos, será também considerada a contribuição de todos os vértices até a sua k-ésima estrela. Então, o vértice v_i sofrerá influência desses vizinhos e tenderá à posição central do conjunto, deixando, portanto, mais suave a distribuição dos elementos pertencentes a esse sistema local. Utilizando-se de uma analogia da física, considere cada vértice como um átomo. Em sistemas físicos é comum que os elementos mais próximos de um átomo exerçam uma maior influência sobre ele e à medida que o raio de atuação aumenta, mais elementos passam a interferir em seu comportamento, porém de maneira mais branda. Voltando aos vértices, pode-se reproduzir esse comportamento fazendo com que v_j interaja com v_i levando em consideração, por exemplo, um valor fixo como o número da estrela de v_j em relação a v_i. Assim, define-se $w_{ij} = \frac{1}{S_j}$, onde S_j representa a estrela de v_j, ou o número de saltos necessários, a partir de v_i, para se chegar em v_j. Desta maneira, os elementos mais distantes de v_i exercerão uma menor influência em seu "comportamento".

A formulação de energia 3.1 é quadrática em cada vértice, então, suas derivadas parciais são expressões lineares. O mínimo é único e pode ser encontrado quando as derivadas em relação aos vértices forem zero, resultando em um sistema linear esparso.

As posições dos vértices são conhecidas e as incógnitas serão os vetores tangentes a

cada ponto. Dessa forma tem-se um sistema

$$\mathbf{AX} = \mathbf{b} \quad \mapsto \quad \left(\frac{\partial E}{\partial v}\right)\mathbf{X} = 0, \tag{3.2}$$

onde $\mathbf{X}$ são as incógnitas do sistema (vetores $\mathbf{t}_i$).

A energia 3.1 pode ser escrita como

$$\begin{aligned} E_i &= \sum_{j \in star_k(v_i)} w_{ij} \left\| v_i + \mathbf{t}_i - v_j - \mathbf{t}_j \right\|^2 \\ &= \frac{1}{S_j} < v_i + \mathbf{t}_i - v_j - \mathbf{t}_j >< v_i + \mathbf{t}_i - v_j - \mathbf{t}_j > \\ &= \frac{1}{S_j}(< v_i, v_i > + < v_j, v_j > + < \mathbf{t}_i, \mathbf{t}_i > + < \mathbf{t}_j, \mathbf{t}_j > + \\ &2\left(< v_i, \mathbf{t}_i > + < v_j, \mathbf{t}_j > - < v_i, v_j > - < v_i, \mathbf{t}_j > - < v_j, \mathbf{t}_i >\right)), \end{aligned}$$

e em seguida derivada em função de v_i:

$$\frac{\partial E_i}{\partial v_i} = \mathcal{L}\left(v_i\right) = 2 \sum_{j \in star_k(v_i)} \frac{1}{S_j}(v_i + \mathbf{t}_i - v_j - \mathbf{t}_j) \Rightarrow \sum_{j \in star_k(v_i)} \frac{1}{S_j}(v_i + \mathbf{t}_i - v_j - \mathbf{t}_j) = 0. \tag{3.3}$$

Para que $\mathbf{t}_i$ seja um vetor pertencente ao plano tangente do vértice em relação à superfície, a seguinte restrição de ortogonalidade será acrescentada ao sistema:

$$\mathcal{T}_i = < \mathbf{t}_i, \mathbf{n}_i > = 0, \tag{3.4}$$

onde $\mathbf{n}_i$ é a normal do vértice que pode ser aproximada por:

$$\mathbf{n}_i = \sum_{j \leq valence(v_i)} \mathbf{n}_{fj}, \tag{3.5}$$

tal que $\mathbf{n}_{fj}$ é a normal da face j, dada pela Equação 2.11, onde $f_j \in \mathcal{F} \subset \mathcal{M}'$ é a j-ésima face que contém o vértice v_i.

Seja $\mathcal{V} \subset \mathcal{M}', |\mathcal{V}| = r$. A partir das Equações 3.3 e 3.4 o sistema linear com restrição

pode ser escrito na seguinte forma matricial:

$$\mathbf{AX} = \mathbf{b} \quad \mapsto \quad \begin{pmatrix} \mathcal{L} \\ \mathcal{T} \end{pmatrix} \mathbf{X} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{3.6}$$

onde $\mathcal{L}$ é a matriz Laplaciana com dimensão $r \times r$ contendo as equações de vértices 3.3 e $\mathcal{T}$ é uma matriz $3 \times r$ composta pelas restrições (Eq. 3.4) de plano tangente de cada vértice e $\mathbf{X}$ os vetores tangentes $\mathbf{t}_i$. O sistema é resolvido por mínimos quadrados já que apresenta mais equações que incógnitas. Nota-se também que o resultado não necessariamente terá $< \mathbf{t}_i, \mathbf{n}_i >= 0$, mas existe uma tendência de que isso ocorra, limitando, assim, as possibilidades para o deslocamento de cada vértice.

Para malhas em $\mathbb{R}^3$ é possível resolver o sistema individualmente para cada coordenada, ou seja, $\mathcal{L}\mathbf{x} = 0, \mathcal{L}\mathbf{y} = 0, \mathcal{L}\mathbf{z} = 0$, como observado em [29]. Outra maneira é condensar as três coordenadas em uma única matriz

$$\mathbf{AX} = \mathbf{b} \quad \mapsto \quad \begin{pmatrix} \mathcal{L}_x & 0 & 0 \\ 0 & \mathcal{L}_y & 0 \\ 0 & 0 & \mathcal{L}_z \\ \multicolumn{3}{c}{\mathcal{T}} \end{pmatrix} \begin{pmatrix} \mathbf{X}_x \\ \mathbf{X}_y \\ \mathbf{X}_z \end{pmatrix} = 0 \tag{3.7}$$

e solucionar o sistema em apenas um passo [28].

Dada a matriz final (Eq. 3.7) é possível definir as posições tangentes (ótimas) de cada vértice. A solução deste problema requer encontrar as incógnitas de um sistema linear esparso. Para isso, foi utilizado o método de solução direta de Cholesky [31]. Como a matriz $\mathcal{L}$ é esparsa, $\mathcal{L}^T\mathcal{L}$ também será, e assim, uma fatoração $\mathcal{L}^T\mathcal{L} = \mathbf{R}^T\mathbf{R}$ é encontrada, onde $\mathbf{R}$ é uma matriz triangular superior. Após isso, as novas coordenadas (x, y, z) são definidas resolvendo dois sistemas triangulares $\mathbf{R}^T\mathbf{RX} = \mathcal{L}^T\mathbf{b}$, que são $\mathbf{R}^T\bar{\mathbf{X}} = \mathcal{L}^T\mathbf{b}$ e $\mathbf{RX} = \bar{\mathbf{X}}$ [32].

No trabalho de Botsch *et al.* [33] é feito um estudo comparativo entre os diferentes métodos numéricos empregados na solução de sistemas lineares em processamento geométrico. Algoritmos como decomposição LU, métodos iterativos, *multigrid* e decomposição de Cholesky foram testados. O último apresentou os melhores resultados em termos de estabilidade numérica e eficiência computacional para matrizes altamente esparsas. Por esse motivo o mesmo foi usado nesta dissertação e em [11, 28, 29, 32].

3.3 Projeção Dos Vértices Na Malha

Para garantir que todas as posições encontradas, após o deslocamento tangente, estejam o mais próximo possível da malha de entrada, uma etapa de projeção dos vértices da malha da i-ésima iteração $\mathcal{M}'_i$ sobre o modelo da iteração anterior, $\mathcal{M}'_{i-1}$, foi incluída no processamento. Após a solução da matriz 3.7, para cada $v'_i \in \mathcal{V}'$ é associada uma posição $\mathbf{p}_i$, onde $\mathbf{p}_i$ está exatamente sobre a superfície gerada no passo anterior $\mathcal{M}'_{i-1}$. A projeção na malha do passo anterior garante um deslocamento mais suave dos vértices entre os passos i e $i+1$. Ela também conduz a uma convergência mais rápida da média do comprimento das arestas ao valor desejado m.

O subproblema de encontrar os pontos mais próximos (*nearest points* [1]) entre dois conjuntos é um problema de alta complexidade computacional. No presente caso, tem-se um conjunto de vértices $\{v_1, ..., v_n\} \subset \mathcal{V}'$ e deseja-se achar $\{\mathbf{p}_1, ..., \mathbf{p}_n\}$ onde $\mathbf{p}_i$ encontra-se sobre uma face, aresta ou mesmo vértice de $\mathcal{M}'_{i-1}$. É importante assegurar que a distância euclidiana entre v'_i e $\mathbf{p}_i$ seja pequena. Assim, de todos os possíveis pontos $\mathbf{p}$ que se encontram sobre a superfície $\mathcal{M}'_{i-1}$, interessa-se encontrar os respectivos $\mathbf{p}_i$ tal que $\|v'_i - \mathbf{p}_i\|$ seja baixo.

O algoritmo projeta cada vértice v'_i nos componentes (vértices, arestas e faces) de $\mathcal{M}$ que sejam mais próximos. Para todo vértice $v'_i \in \mathcal{V}'$ são realizados os seguintes passos:

- Encontre sobre a superfície $\mathcal{M}'_{i-1}$ o vértice v_l que tenha a menor distância para o vértice da vez v'_i;
- Cada triângulo adjacente à v_l forma um plano. Realize uma projeção ortogonal [34] de v'_i em cada um desses planos; isso irá gerar um conjunto de pontos projetados $\{v_p\}$;
- Caso v_p não esteja dentro do triângulo formador do plano (Fig. 3.2.a), projete-o sobre uma das arestas do triângulo correspondente, escolhendo a menor distância como critério (Fig. 3.2.b).
- Dentre todos os pontos $\{v_p\}$, escolha o que apresente a menor distância euclidiana, $\|v'_i - v_{pm}\|$, onde $v_{pm} \in \{v_p\}$;
- A nova posição para v'_i será então v_{pm}: $v'_i \leftarrow v_{pm}$.

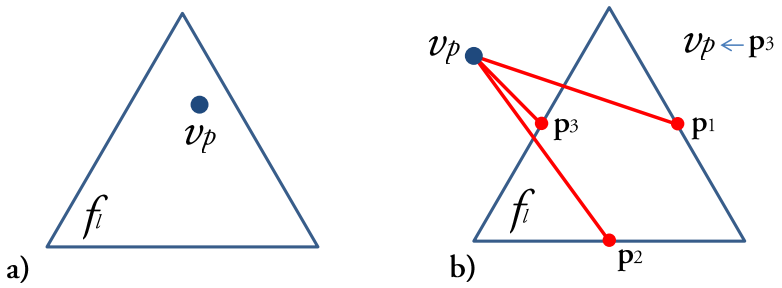

Figura 3.2: a) vértice v_p dentro do triângulo e b) v_p projetado na aresta mais próxima do triângulo formador do plano.

Este processo é repetido para todos os vértices do modelo. Nota-se que o v_{pm} encontrado não representa, necessariamente, a solução exata do problema. Contudo, a implementação deste procedimento apresentou boa estabilidade e resultados satisfatórios, apesar de sua simplicidade. É importante salientar que esse método pode levar a inconsistências na malha em regiões de curvatura negativa. Porém, a média alvo m pode ser ajustada progressivamente de forma a reduzir ou eliminar essa possibilidade. Com menores deslocamentos de m_i, menor a chance de haver projeções indevidas, ao custo de mais iterações.

4 Resultados e Discussão

Esta seção mostrará os resultados da aplicação do método de uniformização em diferentes cenários e serão apresentadas interpretações para os dados. O algoritmo foi implementado utilizando a linguagem de programação C++ na plataforma Linux, com distribuição Ubuntu v11.04. A classe para manipulação geométrica foi o gcgPOLYGON, componente da biblioteca para tratamento de elementos gráficos desenvolvido pelo Grupo de Computação Gráfica da UFJF [35]. O gcgLIB implementa a estrutura de dados *half-edge* para representação de malhas poligonais. A conectividade entre os elementos do modelo é organizada de forma simples e as operações estelares utilizando-se dessa estrutura são extremamente eficientes. Os testes foram rodados em uma máquina com processador Intel Core i5 (2410M), com 6GB de memória RAM e placa de vídeo GTM540M.

Modelo	Vértices	Faces	Média Arestas
bunny	9002	18000	0.601986
egea	8268	16536	0.975998
rockerarm	10044	20000	0.599984
fertility	13971	27942	0.661315

Tabela 4.1: Informações dos modelos utilizados nos testes.

A Tabela 4.1 apresenta as configurações iniciais dos modelos utilizados nos testes. São eles: *bunny* (Fig. 4.1), *rockerarm* (Fig. 4.4), *egea* (Fig. 4.5) e *fertility* (Fig. 4.9).

As seguintes métricas foram utilizadas:

$$\bar{x} = \sum_{i=1}^{n_{\mathcal{A}}} \frac{a_i}{n_{\mathcal{A}}}, \tag{4.1}$$

tal que $\bar{x}$ é a média, $n_{\mathcal{A}}$ o número de arestas de $\mathcal{A} \subset \mathcal{M}'$ e

$$s = \sqrt{\frac{1}{n_{\mathcal{A}} - 1} \sum_{i=1}^{n_{\mathcal{A}}} (a_i - \bar{x})^2}, \tag{4.2}$$

é o desvio padrão da malha processada.

Primeiramente foi estudado o comportamento dos dados do modelo em função do número de iterações. Na Figura 4.1 (direita) é mostrada a malha final processada após 50 iterações. Nota-se claramente a melhora na distribuição dos elementos que compõem o modelo e na Figura 4.2 é apresentado o estado da malha, com aproximação na cabeça do

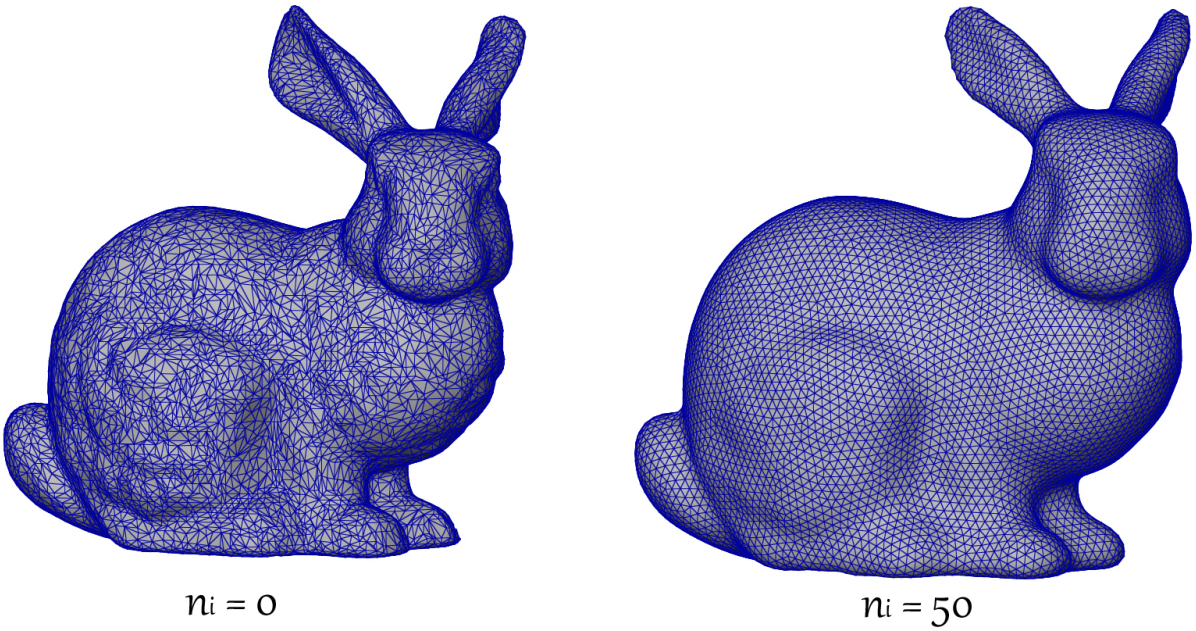

Figura 4.1: Malha de entrada (esquerda) e malha processada após 50 iterações (direita) para $m = 0.5, \sigma = 0.1$ e $k = 2$.

modelo, nos passos $n_i \in \{0, 2, 10, 30\}$, com demais parâmetros $m = 0.5$, $\sigma = 0.1$ e $k = 2$.

Percebe-se que mesmo utilizando o deslocamento tangente seguido de projeção na malha da iteração anterior, o resultado final é ligeiramente suavizado, podendo isso ser percebido melhor ao longo do corpo do coelho (Fig. 4.1 direita). Independente do artifício aplicado para se manter os vértices processados sobre o modelo de entrada, essa suavização é inerente à malhas altamente uniformes, como a da figura em questão. Quando tem-se uma restrição tão forte como a de uniformização, a perda de geometria é verificada. Assim, a preservação das características originais é inversamente proporcional à proximidade dos comprimentos das arestas de um valor constante. Entretanto, mesmo com a perda natural de informação, aplicando as técnicas descritas nesta dissertação pode-se perceber que os contornos originais são mantidos com satisfatória precisão (Fig. 4.1 direita).

Iteração	$\bar{x}$	s	Num. Vértices	Vert. Regulares (%)
$n_i = 0$	0.601986	0.246865	9002	36.6%
$n_i = 5$	0.496593	0.064823	10576	72.3%
$n_i = 10$	0.504868	0.053896	9989	79.3%
$n_i = 20$	0.509861	0.043432	9525	84.0%
$n_i = 30$	0.511745	0.039673	9308	86.3%
$n_i = 50$	0.512147	0.037425	9128	87.1%

Tabela 4.2: Avanço das propriedades do modelo a cada iteração com os parâmetros da Figura 4.1.

A Tabela 4.2 exibe as informações do modelo em cada iteração. Como esperado, ao final do processamento a média das arestas do modelo tende ao valor desejado m. O desvio padrão s decai significativamente, sobretudo nas primeiras iterações, até um valor relativamente baixo indicando um alto grau de uniformidade na distribuição dos vértices. É percebido que quanto mais vértices regulares, mais a variância das arestas diminui e, consequentemente, o modelo tende à média desejada m com maior facilidade. Ao final do processo o modelo apresenta uma quantidade baixa de vértices irregulares (por volta de 12.9%) se comparado ao seu estado inicial.

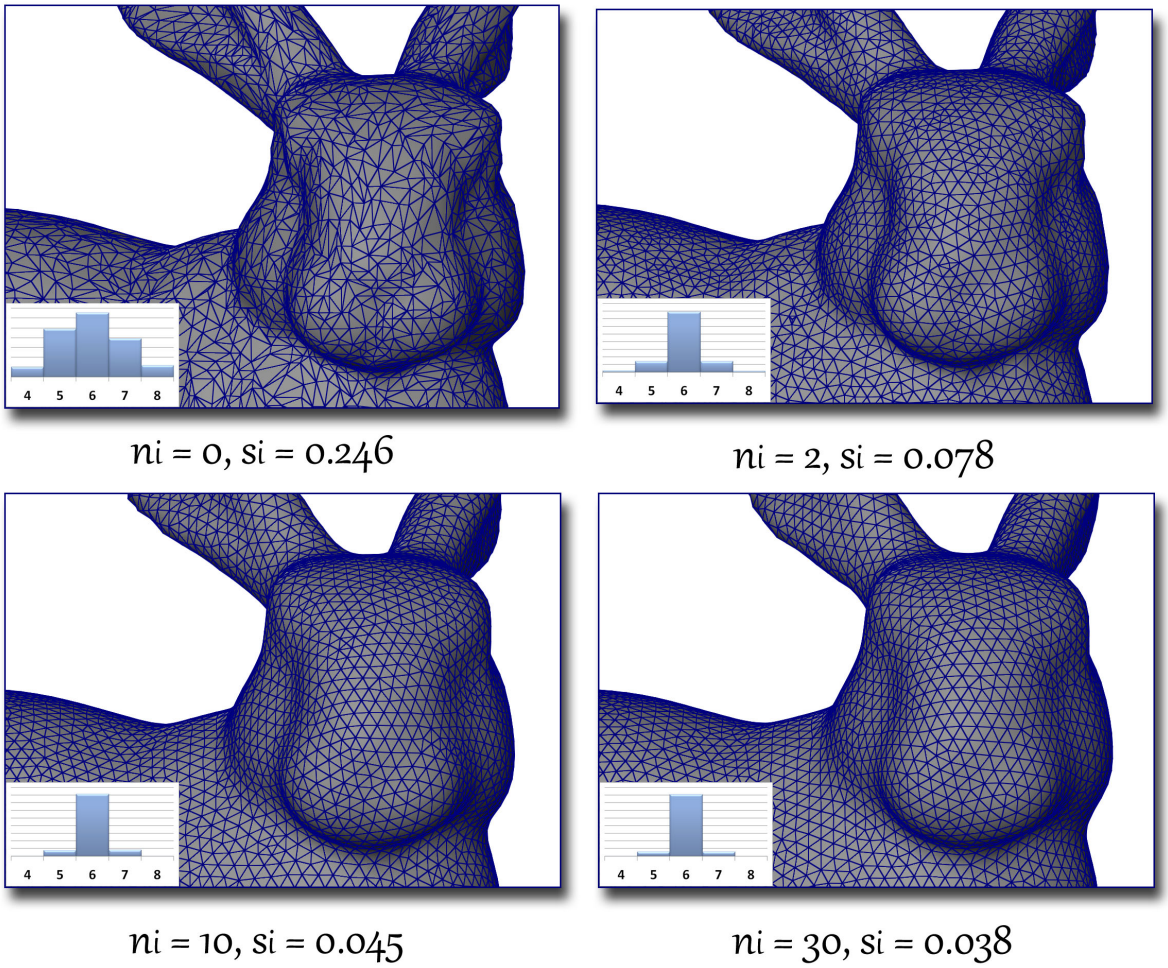

Figura 4.2: A aproximação do coelho revela que a distribuição dos vértices é corrigida e a conectividade é gradativamente regularizada à medida que o número de iterações aumenta. Os histogramas nos cantos inferiores esquerdos representam o número de vértices no modelo, na i-ésima iteração, com valência 4, 5, 6, 7 e 8. Os valores decrescem conforme o algoritmo avança.

Conclui-se que o número de iterações está intimamente ligado à precisão do comprimento de arestas desejado. Quanto maior n, mais uniforme será a distribuição dos

elementos pela malha final e menos informação será preservada. A escolha de n depende exclusivamente da aplicação da malha final. Como exemplo é possível citar simulações computacionais em engenharia onde grades altamente uniformes são sinônimos de estabilidade numérica na solução deste tipo de problema. Nota-se também que o algoritmo é capaz de entregar uma malha com média próxima a m em poucas iterações (5 iterações, como mostrado na tabela 4.2).

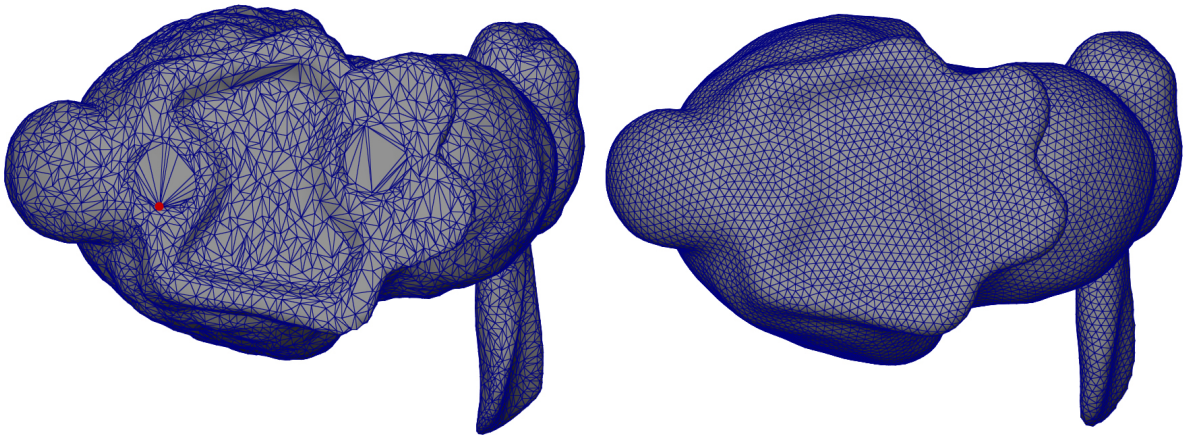

Figura 4.3: A base do *bunny* é a parte mais irregular do modelo. O vértice marcado em vermelho apresentou valência 21 e suas arestas adjacentes, comprimentos completamente desproporcionais ao restante da malha.

O algoritmo se mostrou robusto para o remalhamento de modelos com baixa qualidade. A Figura 4.3 (esquerda) ilustra esse tipo de situação. Mesmo para essa malha, o método foi capaz de corrigir a irregularidade dos vértices e faces com eficiência – Fig. 4.3 (direita).

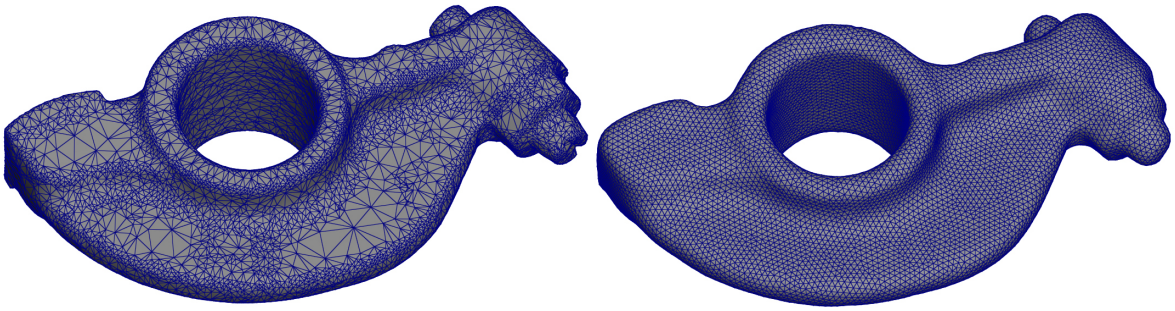

Figura 4.4: Modelo *rockerarm* após reamostragem dos vértices.

Além da quantidade de iterações, o comprimento da aresta é um componente importante quando se deseja ou não preservar a geometria original. Quanto maior o comprimento da aresta, mais informação é descartada. Se m for menor que a média da superfície inicial, tem-se um caso de super-amostragem, ou seja, a quantidade de vértices e faces

aumentará e mais informação o modelo final irá conter.

A Figura 4.5 apresenta o modelo *rockerarm* após o remalhamento com parâmetros $m = 0.3, \sigma = 0.06, n = 30$ e $k = 2$. Tem-se um caso de super-amostragem, onde a geometria do modelo inicial é satisfatoriamente preservada, como pode ser notada na imagem.

Iteração	Média	Desvio Padrão	Num. Vértices	Vert. Regulares (%)
$n_i = 0$	0.599984	0.341534	10044	37.9%
$n_i = 30$	0.308452	0.021501	38482	85.9%

Tabela 4.3: Dados do modelo da Figura 4.5. Apresentando média e desvio padrão das arestas, número de vértices e porcentagem de vértices regulares.

A Tabela 4.3 apresenta os dados da malha de entrada ($n_i = 0$) e da malha de saída ($n_i = 30$) e novamente percebe-se que a qualidade da malha é melhorada quantitativamente: média desejada alcançada, baixo desvio padrão nas distâncias dos vértices e boa conectividade dos mesmos.

A Figura 4.6 apresenta a superfície *rockerarm* gerado com diferentes comprimentos de aresta. Percebe-se que os detalhes do modelo (regiões de alta curvatura) são descartados à medida que m aumenta. Este processo, conhecido como decimação (simplificação) de malha é comum quando deseja-se um objeto compacto, com poucos elementos e tamanho físico reduzido. Malhas uniformes não são ideais para este tipo de representação, visto que sua forte restrição faz com que a perda de informação seja mais significativa, sobretudo nas regiões de maior saliência. O caráter iterativo do algoritmo aqui proposto contribui negativamente para que esse comportamento seja propagado a cada passo do processamento.

É importante ressaltar que, muito embora a geometria do objeto seja comprometida à medida que a aresta é aumentada, o método consegue entregar uma malha nos moldes do que foi proposto desde o início: superfície final com comprimento de aresta próximo a m tal que $m \approx m_n \pm \sigma$. O fato pode ser comprovado através dos dados da Tabela 4.4. O algoritmo alcança a média desejada mesmo que com poucos vértices.

O parâmetro k desempenha um papel interessante no algoritmo. A utilização de $k > 1$ mostrou-se eficiente na maioria dos testes realizados, gerando melhores resultados que a formulação mais recorrente na literatura ($k = 1$, Laplaciano uniforme). Foi verificado que a utilização de $k = 2$ é o suficiente para se obter bons números na maioria dos casos. Entretanto, em alguns cenários é preciso que este valor seja aumentado, visando uma

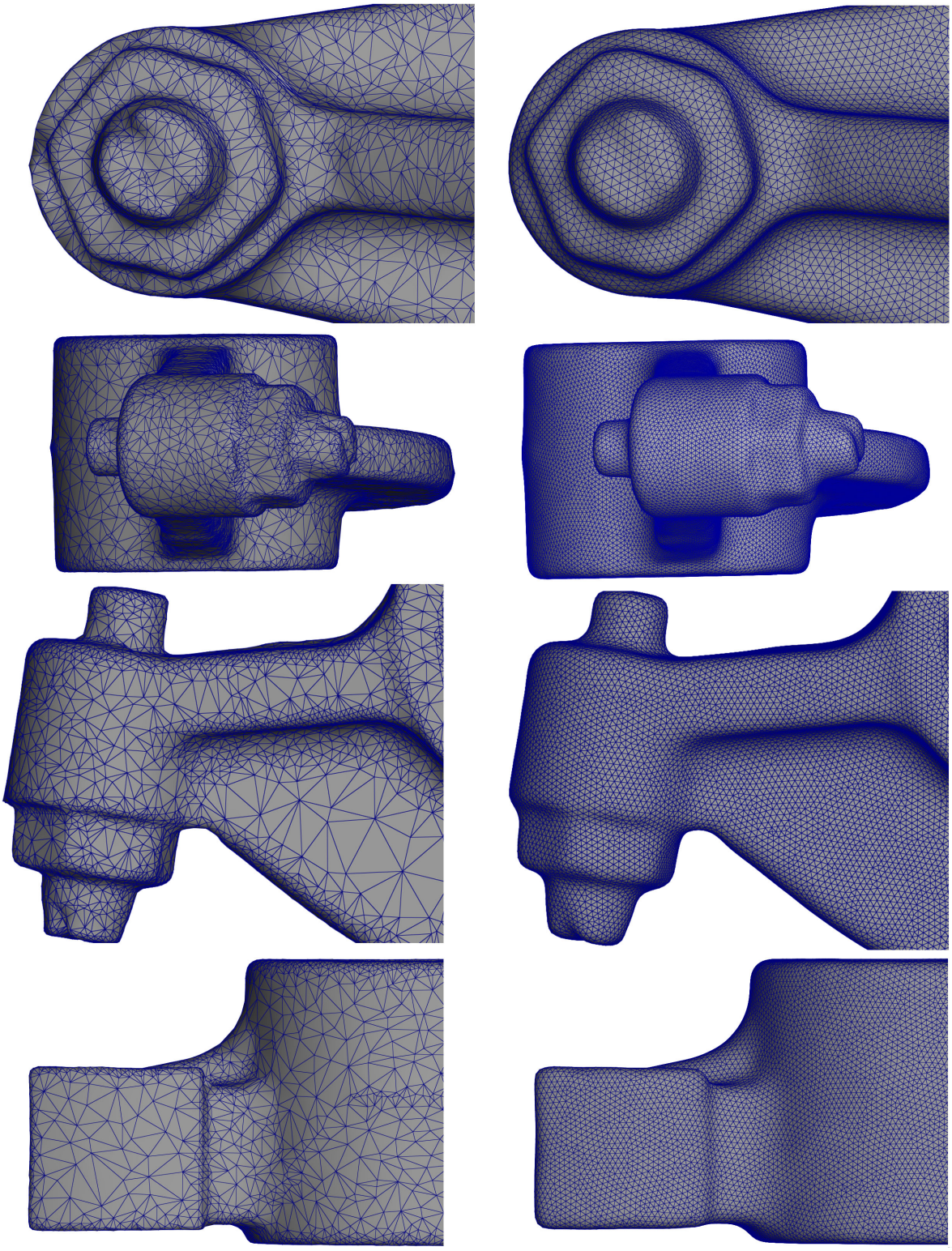

Figura 4.5: Remalhamento de alta qualidade.

melhor distribuição.

A Tabela 4.5 mostra os dados finais dos processamentos realizados para o modelo da Figura 4.7. É possível verificar que para todos os testes, as médias e desvios das arestas

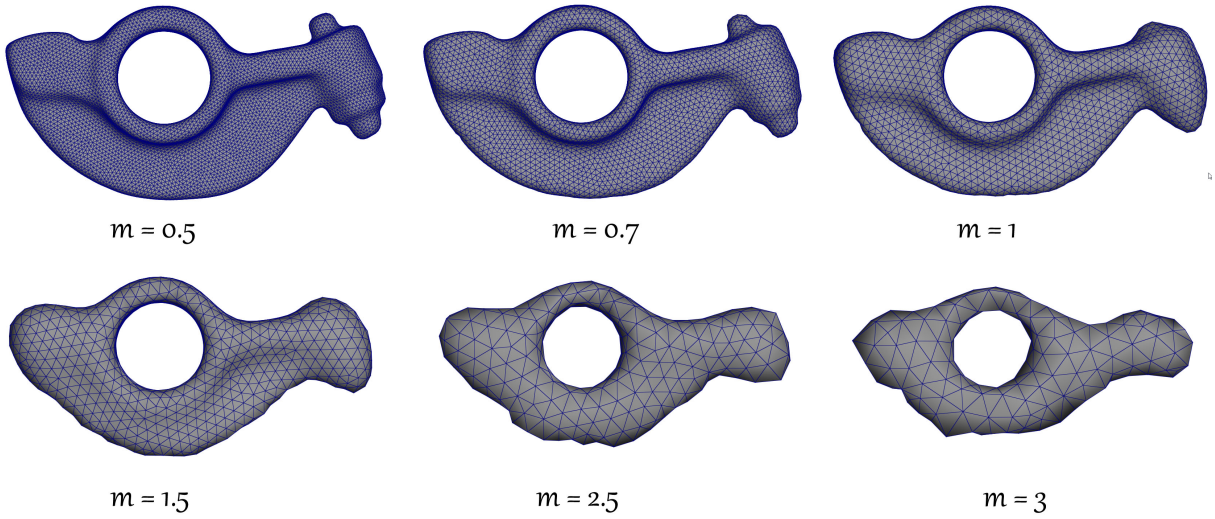

Figura 4.6: *Rockerarm* processado para diferentes comprimentos de aresta. Cada imagem corresponde a um modelo gerado com m das imagens, $\sigma = 0.2m, n = 30$ e $k = 2$. Os dados de cada objeto estão presentes na Tabela 4.4.

(m, σ, n, k)	$\bar{x}$	s	Num. Vértices	Vert. Regulares (%)
Modelo Inicial	0.599984	0.341534	10044	37.9%
$(0.5, 0.1, 30, 2)$	0.520501	0.043505	13550	85.2%
$(0.7, 0.14, 30, 2)$	0.715916	0.055755	6621	85.7%
$(1, 0.2, 30, 2)$	1.019295	0.081193	2988	86.3%
$(1.5, 0.3, 30, 2)$	1.486135	0.156538	1109	80.7%
$(2.5, 0.5, 30, 2)$	2.436371	0.362264	332	72.4%
$(3, 0.6, 30, 2)$	2.840112	0.470012	210	69.5%

Tabela 4.4: Dados do modelo da Figura 4.5. Apresentando média e desvio padrão das arestas, número de vértices e porcentagem de vértices regulares.

alcançaram menores valores quando $k > 1$. O parâmetro σ tem um papel regulador importante; foi verificado empiricamente que σ atinge melhores resultados quando assume valores que correspondem à $20 - 25\%$ do valor da média desejada. Também foi observado que à medida que σ aumenta é importante que k também assuma valores maiores para que $\bar{x}$ se aproxime de m e o desvio final s seja baixo.

Uma possível interpretação das propriedades da malha para diferentes valores σ é a que segue: é de se pensar que quanto menor σ, mais a malha tenderá a um baixo desvio padrão. Contudo, isso não é verificado devido a alta quantidade de operações estelares que um σ muito baixo promoverá. Muitas operações por iteração significa uma malha menos estável e, consequentemente, maiores valores para s. Quando σ aumenta, menos arestas entrarão na lista de prioridades e então menos elementos serão uniformizados. Observa-se, assim, arestas com desvio padrão acentuado. Um ponto de equilíbrio que minimiza o desvio padrão das amostras foi encontrado para $\sigma \approx 0.2m$, como apresentado

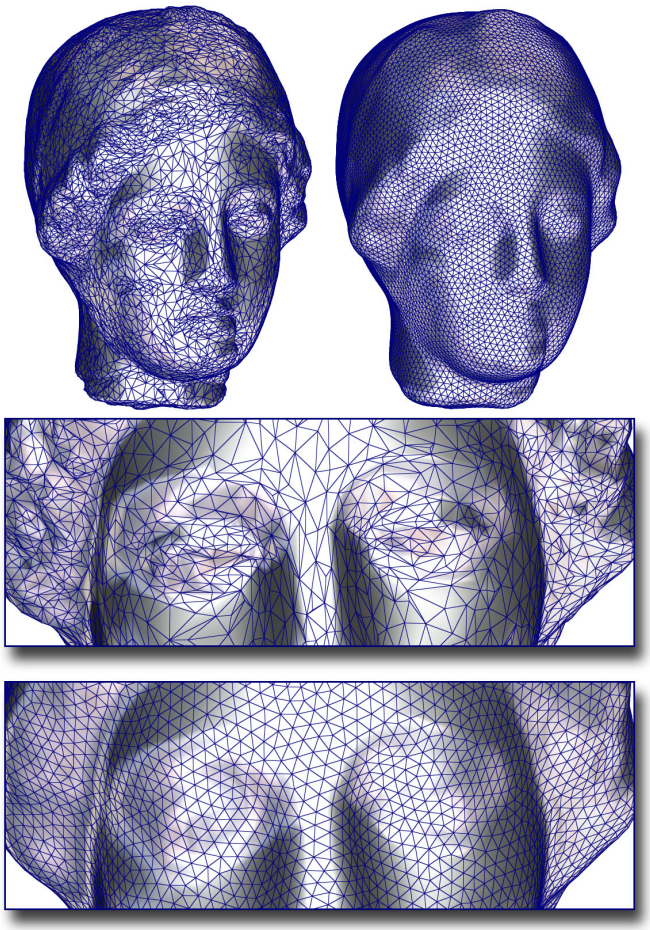

Figura 4.7: Modelo *egea* após remalhamento (recorte inferior) com parâmetros $m = 0.8, \sigma = 0.2, n = 20$ e $k = 4$.

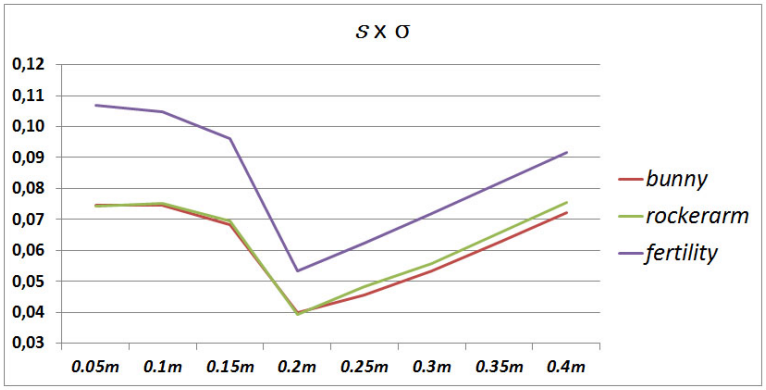

Figura 4.8: Desvio padrão (s) dos objetos processados em função do valor de entrada σ. Quando $\sigma \approx 0.2m$ os modelos apresentam um menor s.

no gráfico da Figura 4.8. Neste, os parâmetros (m, σ, n, k) foram *bunny*$(0.5, 0.5p, 20, 2)$, *fertility*$(0.7, 0.7p, 20, 2)$ e *rockerarm*$(0.5, 0.5p, 20, 2)$ onde $p \in [0.05, 0.4]$.

A Figura 4.9 apresenta o resultado visual dos dados do gráfico da Figura 4.8. Nota-se que quando σ é baixo, a distribuição dos vértices não se mantém muito uniforme e existe

(m, σ, n, k)	$\bar{x}$	s	Num. Vértices	Vert. Regulares (%)
Modelo Inicial	0.599984	0.341534	10044	37.9%
$(0.8, 0.1, 20, 1)$	0.775103	0.135009	10648	**68.2%**
$(0.8, 0.1, 20, 2)$	0.785205	**0.126693**	10531	67.7%
$(0.8, 0.1, 20, 3)$	0.793493	0.129107	10450	66.6%
$(0.8, 0.1, 20, 4)$	**0.799652**	0.130791	10380	66.2%
$(0.8, 0.2, 20, 1)$	0.792564	0.076465	10447	80.1%
$(0.8, 0.2, 20, 2)$	**0.801297**	0.073887	10282	81.7%
$(0.8, 0.2, 20, 3)$	0.814370	**0.071990**	9921	84.0%
$(0.8, 0.2, 20, 4)$	0.825473	0.073180	9617	**84.8%**
$(0.8, 0.35, 20, 1)$	0.772455	0.128458	10906	**76.6%**
$(0.8, 0.35, 20, 2)$	0.781785	0.127766	10836	73.1%
$(0.8, 0.35, 20, 3)$	0.789769	0.127981	10789	71.6%
$(0.8, 0.35, 20, 4)$	**0.796638**	**0.127747**	10690	72.5%
$(0.8, 0.41, 20, 1)$	0.769633	**0.152831**	10910	**74.2%**
$(0.8, 0.41, 20, 2)$	0.781499	0.156026	10827	70.0%
$(0.8, 0.41, 20, 3)$	0.791372	0.156487	10799	69.0%
$(0.8, 0.41, 20, 4)$	**0.794345**	0.156241	10850	69.1%

Tabela 4.5: Valores do remalhamento de *egea* (Fig. 4.7).

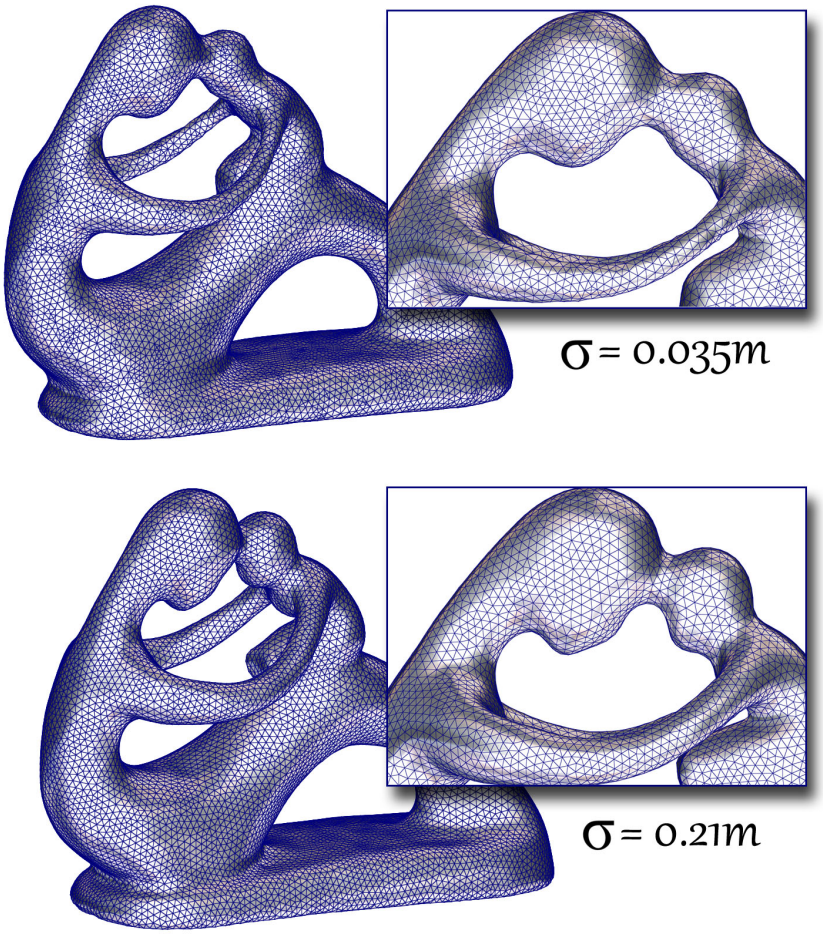

Figura 4.9: Malhas processadas com $\sigma = 0.0035m$ (superior) e $\sigma = 0.21m$ (inferior). Repara-se a perda de informação no braço do modelo para σ pequeno.

também a perda de informação, o que pode ser evidenciado nos braços do modelo − Fig. 4.9 (superior). Em contrapartida, o objeto processado com $\sigma = 0.21m$ apresenta uma maior uniformidade e fidelidade com o modelo original − Fig. 4.9 (inferior).

O algoritmo proposto pode ser dividio em três fases principais:

- $F1$: etapa de criação da lista de prioridades e operações estelares.
- $F2$: otimização linear com Laplaciano de ordem k.
- $F3$: projeção da malha no modelo da iteração anterior.

Dentre estas, $F1$ é a que apresenta menor complexidade computacional e tempo de processamento. As etapas $F2$ e $F3$ são as que demandam maior recurso de espaço/memória ($F2$) e de tempo computacional ($F3$). A etapa de projeção realiza uma busca exaustiva por $v_k \in \mathcal{M}$ que mais se aproxima de $v_i \in \mathcal{M}'$, para que as faces e arestas adjacentes do vértice encontrado sejam utilizadas na verificação. Sem nenhum tratamento especial tem-se, inicialmente, uma busca com complexidade $\mathcal{O}(n^2)$: para cada vértice é preciso varrer todo o espaço em busca do vizinho mais próximo. Isso torna o processo de decimação extremamente lento, o que vai de encontro a uma das principais vantagens dos métodos de remalhamento explícito que é a velocidade do processamento. Durante a implementação, notou-se que algumas configurações se tornaram impraticáveis devido ao grande número de projeções demandados na fase $F3$. Como exemplo considere a Tabela 4.6.

m	Num. Vértices	$F1$	$F2$	$F3$	Total
0.2	96971	0.44s	3.35s	1024s	1027.79s
0.3	67311	0.38s	2.73s	673s	676.11s
0.4	39284	0.29s	1.97s	389s	391.26s
0.5	24710	0.22s	1.39s	215s	216.61s
0.6	16897	0.16s	0.98s	108s	109.14s

Tabela 4.6: Tempo de processamento para o modelo *fertility* variando m e utilizando projeção com busca quadrática e $n = 5$.

O pior cenário é quando $m = 0.2$. Neste, tem-se uma grande quantidade de vértices e o tempo de processamento ultrapassa os 15 minutos, para *apenas* 5 iterações. É um tempo alto para este tipo de remalhamento que prima pela simplicidade e velocidade. O gargalo da aplicação é a última fase ($F3$), onde a projeção por busca quadrática é empregada. Observa-se que o tempo das demais etapas foi satisfatório, sobretudo no primeiro teste,

com $m = 0.2$ e 96.000 vértices, onde realiza-se a solução de uma matriz 55.000×55.000 (média do número de vértices), cinco vezes, em pouco mais de 3 segundos.

Para a solução deste problema foi utilizada uma busca baseada em particionamento espacial. Um particionamento da malha $\mathcal{M}$ em u segmentos, chamados aqui de $\mathcal{P}$ é dado por $\mathcal{M} = \mathcal{P}_1 \cup ... \cup \mathcal{P}_u$, tal que $\mathcal{P}_1 \cap ... \cap \mathcal{P}_u = 0$. As formas de se realizar essa divisão espacial geralmente envolvem algoritmos de computação gráfica como *Octree*, *Kd-Tree* ou BSP [34].

m	Num. Vértices	$F1$	$F2$	$F3$	Total (*Kd-Tree*)	Total (Quadrático)
0.2	96971	0.45s	3.09s	133.1s	136.54s	1027.79s
0.3	67311	0.40s	2.75s	105.5s	108.15s	676.11s
0.4	39284	0.30s	1.96s	72.9s	74.26s	391.26s
0.5	24710	0.23s	1.37s	46.4s	47.59s	216.61s
0.6	16897	0.16s	0.95s	30s	31.11s	109.14s

Tabela 4.7: Tempo de processamento para o modelo *fertility* variando m e utilizando projeção com *Kd-Tree* e $n = 10$.

A idéia é simples: o espaço será particionado em octantes (para o algoritmo *Octree*) ou dividido em segmentos de tamanho variável utilizando-se de planos alinhados aos eixos de $\mathbb{R}^3$. Para este último, o algoritmo *Kd-Tree* (árvore k-dimensional) é comumente aplicado. Este método organiza o espaço de busca de maneira hierárquica para que o acesso aos dados seja feito de forma direcionada, ou seja, encontrar um vizinho para um vértice v_i qualquer, não significa mais verificar todos os elementos do sistema, mas sim avaliar somente as distâncias dos vértices que estão em partições próximas a este.

Após a implementação da busca por *Kd-tree*, notou-se uma melhora significativa no tempo de processamento, sobretudo nos casos de menor m (Tab. 4.7). As fases $F1$ e $F2$ não são influenciadas pela etapa de projeção, o que pode ser evidenciado pela pouca diferença de tempo entre as respectivas colunas das Tabelas 4.6 e 4.7. O ganho de desempenho deve-se ao fato da busca ser realizada com uma complexidade $\mathcal{O}(nlog_n)$, em vez de $\mathcal{O}(n^2)$. De uma maneira geral, o tempo de processamento do método proposto com *Kd-Tree* se mostrou satisfatório.

5 Aplicações

Uma das principais aplicações para malhas altamente uniformes são as simulações computacionais. A perda de geometria, inerente à superfícies desse tipo, é justificada principalmente quando o objeto tridimensional serve como entrada para algum processo de cálculo. Assim, o baixo desvio padrão das arestas e a boa conectividade e distribuição dos vértices significam estabilidade numérica para a simulação.

Uma das motivações desta dissertação é a geração de malhas hexagonais uniformes empregadas como *input* em simulações físicas de nano-estruturas de carbono. Essas estruturas podem ser geradas a partir de malhas triangulares arbitrárias, como em Nieser *et al.* [9], ou por espaços paramétricos [36]. O segundo método, embora simples e eficiente, requer um conhecimento matemático apurado para se construir superfícies complexas.

A abordagem que será adotada para a criação de objetos hexagonais é a do *espaço dual* da malha triangular de entrada. O espaço dual é formado por vértices centroides aos triângulos e arestas que os conectam com os centros vizinhos. A partir de cada vértice (dual) saem três arestas direcionadas aos centros das faces adjacentes (triângulo primal), como mostrado na Figura 5.1.

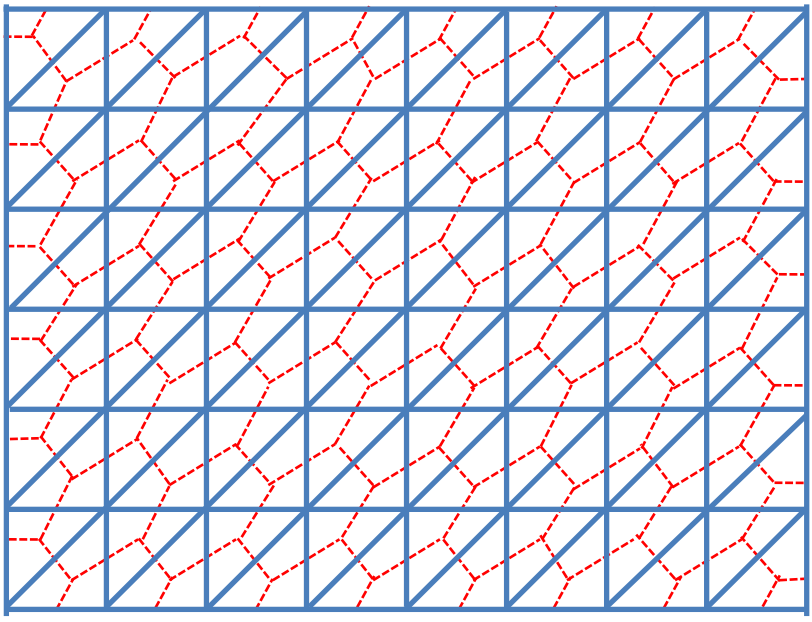

Figura 5.1: Malha primal e dual: a primeira é a grade de referência (em azul), enquanto a segunda é representada pelos traços em vermelho.

Nota-se que cada vértice no espaço dual $\mathcal{D}$ terá exatamente três arestas incidentes, o que caracteriza a malha gerada como regular. Cada vértice primal será o centroide dos polígonos no espaço oposto. É fácil observar que, caso os vértices da malha primal não sejam regulares, $\mathcal{D}$ será composto por faces não hexagonais. Dessa forma, o correto é referenciar $\mathcal{D}$ como uma malha *trivalente*, ou seja, composta somente por vértices com valência 3, mas que podem apresentar polígonos não hexagonais como quadrados, pentágonos e heptágonos.

Em nano-estruturas de carbono, os átomos tendem a se manter a uma distância constante uns dos outros. Para geração deste tipo de superfície através de malhas duais é necessário que o objeto triangular de entrada apresente uma boa uniformidade em suas arestas. Só assim a malha dual será capaz de representar, satisfatoriamente, as estruturas em questão.

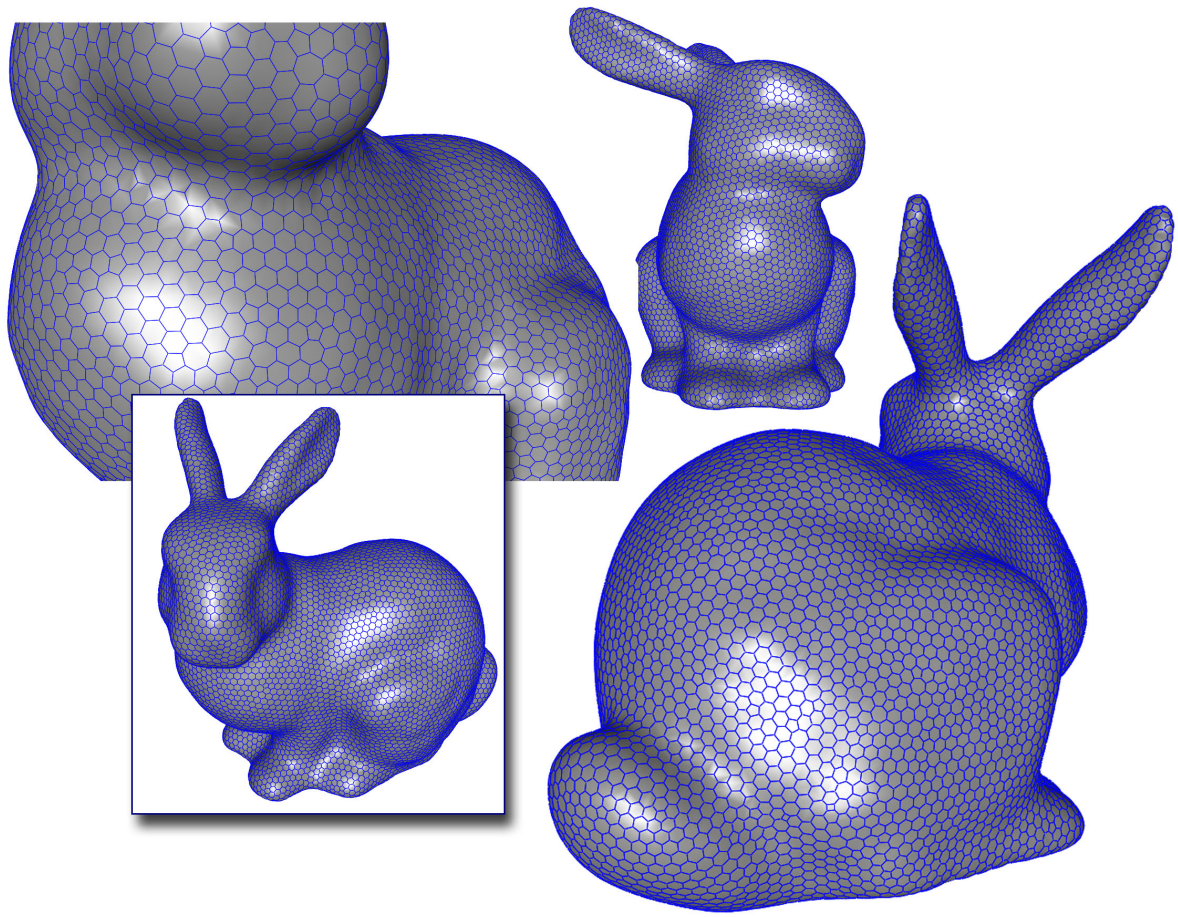

Figura 5.2: Malha dual do modelo *bunny* após o remalhamento uniforme com $m = 0.5, \sigma = 0.1, n = 20$ e $k = 2$.

Na Figura 5.2 é mostrado o dual da malha triangular do modelo *bunny*. A superfície

triangular foi gerada com os parâmetros $(m, \sigma, k, n) = (0.5, 0.1, 20, 2)$ e logo em seguida foi calculada a malha trivalente correspondente. Percebe-se uma boa distribuição dos vértices na figura em questão e, por isso, a malha pode ser utilizada para simulações físicas como dinâmica molecular.

As formulações de interação entre átomos variam de modelos simples, como o conhecido Lennard-Jones [37], a cálculos mais complexos, como utilizados no potencial REBO [38, 39]. Na Figura 5.3 é apresentado a distribuição das energias pela malha trivalente, calculada a partir do dual do modelo *bunny*, antes e depois de passar pelo remalhamento uniforme. As cores mais próximas do azul indicam menor energia e as mais próximas do vermelho maior.

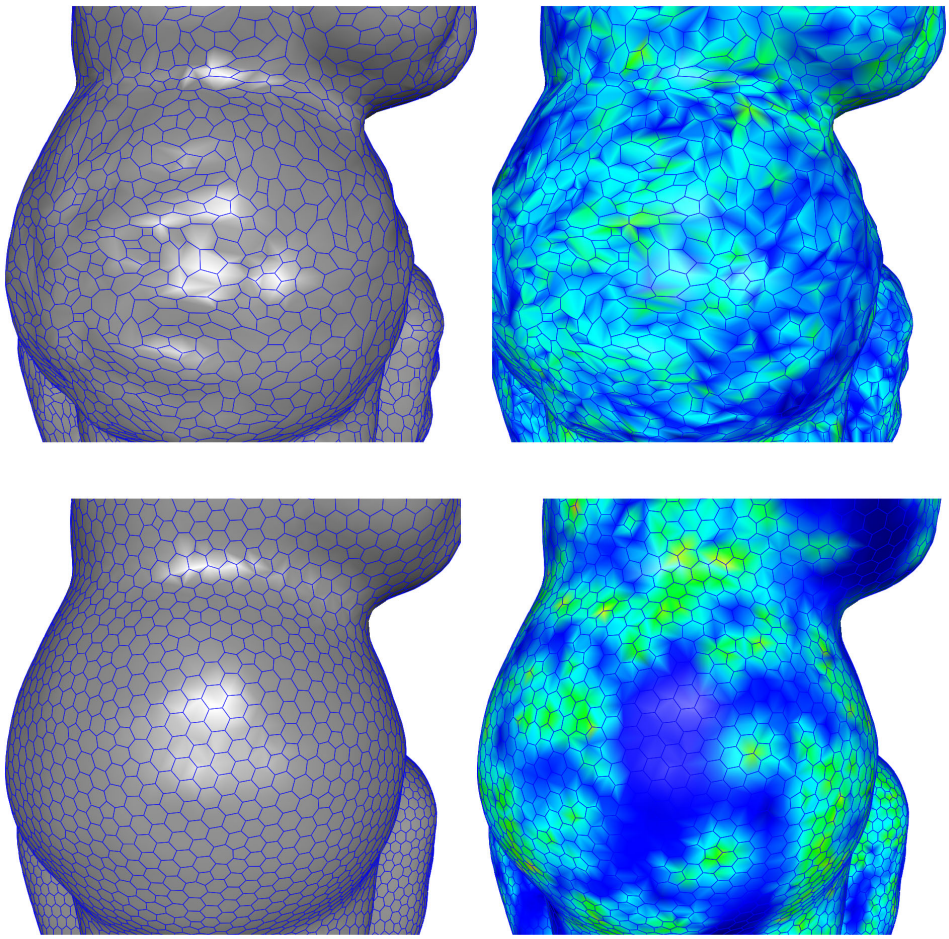

Figura 5.3: Distribuição das energias dos átomos em um sistema molecular com a malha dual do coelho antes (superior) e depois (inferior) do remalhamento.

É possível inferir que a superfície após o remalhamento (inferior) se apresentará mais estável para simulações de dinâmica molecular do que a original (superior).

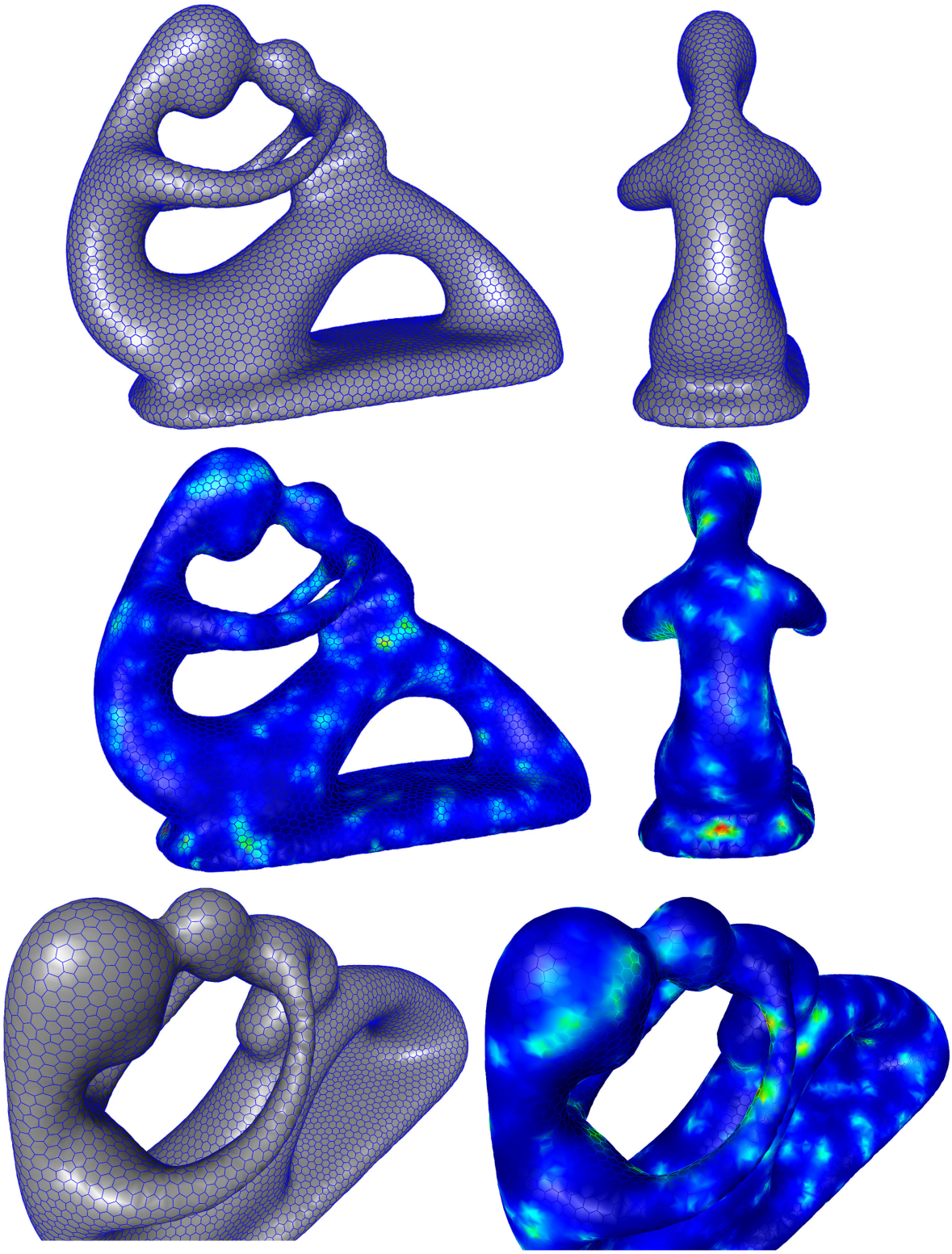

Figura 5.4: Malha trivalente do modelo triangular *fertility* após remalhamento com $(m, \sigma, k, n) = (1, 0.3, 10, 3)$. As cores representam as energias dos átomos (vértices) para o potencial de Lennard-Jones.

6 Conclusão

Neste trabalho foi apresentado um método iterativo para geração de malhas poligonais com distribuição uniforme. Dada uma superfície triangular como entrada, juntamente com a tupla (m, σ, n, k), o algoritmo retorna uma malha com comprimento e desvio padrão de arestas próximos a m e σ, respectivamente.

Arestas são removidas ou adicionadas do modelo caso apresentem um desvio maior que $d_i = |a_i - \sigma|$. Para uma maior estabilidade e convergência foi introduzido o conceito de lista de prioridades, onde os elementos mais distantes do objetivo são processados primeiro. A lista se mostrou eficiente na organização destes elementos visto que o algoritmo consegue atingir a média alvo em poucas iterações, como mostrado nos resultados. Observou-se que um menor desvio padrão das arestas no objeto final não é necessariamente alcançado quando σ adota um valor baixo. Quando σ for pequeno, maior será a quantidade de elementos na lista de prioridades e, por conseguinte, mais operações estelares serão realizadas causando menos estabilidade nas propriedades quantitativas do modelo. Foi constatado empiricamente que valores para σ entre 20% e 25% do comprimento de aresta desejado conseguem gerar malhas uniformes com um menor desvio padrão.

Após as operações estelares o algoritmo entra em uma fase de otimização linear. As equações são montadas através de uma variação do operador Laplaciano discreto, que se utiliza dos k primeiros vizinhos de cada vértice, ao contrário da formulação clássica onde somente a 1-estrela é considerada. Mostrou-se na seção de resultados que essa abordagem é capaz de reamostrar os vértices uniformemente; os menores desvios são obtidos quando k adota valor 2. Também observou-se que quando σ é grande, k deve ser aumentado para que uma melhor distribuição seja alcançada.

Uma aplicação direta para malhas uniformes é a geração de objetos trivalentes (compostos por quadrados, pentágonos, hexágonos e heptágonos) de boa qualidade visual. Estas estruturas são interessantes em simulações físicas de dinâmica molecular e podem representar nano-estruturas de carbono. Além disso, foi discutido que malhas trivalentes de alta qualidade são importantes na estabilidade dos métodos numéricos e podem ser empregadas na resolução dos modelos de interação atômica.

6.1 Trabalhos Futuros

Alguns dos parâmetros iniciais do algoritmo podem ser automaticamente calculados. O número total de iterações pode ser definido em função de um critério de parada, como por exemplo, o desvio padrão das médias das arestas. As condições de convergência neste caso precisam ser estabelecidas. Pretende-se em um trabalho futuro investigar as possíveis condições que encontrem um n ótimo para a solução do problema.

Os parâmetros k e σ ideais também podem ser automaticamente estabelecidos quando o objetivo for a geração de modelos altamente uniformes: à medida que σ cresce, k também deve adotar um valor alto. É de interessa a elaboração de um estudo formal que explique os verdadeiros motivos dessa relação. Também foi observado que o σ ideal tem cerca de 20% do valor da média desejada m. Pretende-se pesquisar e estabelecer formalmente um critério para a definição deste parâmetro.

Ainda como trabalhos futuros destaca-se a aplicação do modelo final gerado à outras ferramentas de simulação e modelagem. Como as malhas geradas são superfícies, ou seja, modelos bidimensionais imersos em $\mathbb{R}^3$, em algumas situações é necessário que o interior deste objeto seja totalmente preenchido. Para isso, uma linguagem de geração de volumes como apresentada em [40] pode ser adaptada para o problema atual. Do ponto de vista de simulações, o modelo *dual* da malha processada pode ser utilizado em simulações de nano-estruturas como feito para materiais ferromagnéticos em [41, 42].

REFERÊNCIAS

[1] BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., UNO LEVY, B., *Polygon Mesh Processing*. AK Peters, 2010.

[2] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., STUETZLE, W., "Mesh optimization". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, *SIGGRAPH '93*, pp. 19–26, ACM: New York, NY, USA, 1993.

[3] ALLIEZ, P., VERDIÈRE, E. C. D., DEVILLERS, O., ISENBURG, M., "Isotropic Surface Remeshing". In: *Proceedings of the Shape Modeling International 2003*, *SMI '03*, pp. 49–, IEEE Computer Society: Washington, DC, USA, 2003.

[4] MING YAN, D., LÉVY, B., LIU, Y., SUN, F., PING WANG, W., "Isotropic Remeshing with Fast and Exact Computation of Restricted Vor onoi Diagram". In: *ACM/EG Symposium on Geometry Processing / Computer Graphics Forum*, 2009.

[5] SURAZHSKY, V., GOTSMAN, C., "Explicit Surface Remeshing". In: *Proceedings of Eurographics Symposium on Geometry Processing*, pp. 17–28, Aachen, Germany, June 2003.

[6] LI, Y., ZHANG, E., KOBAYASHI, Y., WONKA, P., "Editing operations for irregular vertices in triangle meshes". In: *ACM SIGGRAPH Asia 2010 papers*, *SIGGRAPH ASIA '10*, pp. 153:1–153:12, ACM: New York, NY, USA, 2010.

[7] BOMMES, D., ZIMMER, H., KOBBELT, L., "Mixed-integer quadrangulation", *ACM Trans. Graph.*, v. 28, n. 3, pp. 77:1–77:10, July 2009.

[8] ALLIEZ, P., MEYER, M., DESBRUN, M., "Interactive geometry remeshing", *ACM Trans. Graph.*, v. 21, n. 3, pp. 347–354, July 2002.

[9] NIESER, M., PALACIOS, J., POLTHIER, K., ZHANG, E., "Hexagonal global parameterization of arbitrary surfaces". In: *ACM SIGGRAPH ASIA 2010 Sketches*, *SA '10*, pp. 5:1–5:2, ACM: New York, NY, USA, 2010.

[10] RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., ALLIEZ, P., "Periodic global parameterization", *ACM Trans. Graph.*, v. 25, n. 4, pp. 1460–1485, Oct. 2006.

[11] BOTSCH, M., KOBBELT, L., "A remeshing approach to multiresolution modeling". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, *SGP '04*, pp. 185–192, ACM: New York, NY, USA, 2004.

[12] LÉVY, B., "Constrained Texture Mapping". In: *ACM SIGGRAPH conference proceedings*, Addison Wesley, Aug 2001.

[13] PIETRONI, N., TARINI, M., CIGNONI, P., "Almost isometric mesh parameterization through abstract domains", *IEEE Transaction on Visualization and Computer Graphics*, v. 16, n. 4, pp. 621–635, July/August 2010.

[14] BOOTHBY, W., *An Introduction to Differentiable Manifolds and Riemannian Geometry*. N. Nº 1-2, *Pure and Applied Mathematics*, Academic Press, 1975.

[15] DO CARMO, M., *Differential geometry of curves and surfaces*. N. p. 2, Prentice-Hall, 1976.

[16] VELHO, L., GOMES, J., DE FIGUEIREDO, L. H., *Implicit Objects in Computer Graphics*. 1st ed. Springer-Verlag: New York, LLC, 2002.

[17] BOTSCH, M., PAULY, M., ROSSL, C., BISCHOFF, S., KOBBELT, L., "Geometric modeling based on triangle meshes". In: *ACM SIGGRAPH 2006 Courses*, *SIGGRAPH '06*, ACM: New York, NY, USA, 2006.

[18] COXETER, H., *Introduction to Geometry*. *Wiley Classics Library*, John Wiley & Sons, 1989.

[19] LEWINER, T., LOPES, H., MEDEIROS, E., TAVARES, G., VELHO, L., "Topological mesh operators", *Computer Aided Geometric Design*, v. 27, n. 1, pp. 1–22, january 2010.

[20] VELHO, L., "Stellar subdivision grammars". In: *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, *SGP '03*, pp. 188–199, Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2003.

[21] DE BERG, M., KREFELD, V., SCHWARZKOPF, O., *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[22] ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., DESBRUN, M., "Anisotropic polygonal remeshing", *ACM Trans. Graph.*, v. 22, n. 3, pp. 485–493, July 2003.

[23] ALLIEZ, P., UCELLI, G., GOTSMAN, C., ATTENE, M., "Recent advances in remeshing of surfaces". In: *Shape Analysis and Structuring, Mathematics and Visualization*, Springer, 2008.

[24] GUSKOV, I., VIDIMČE, K., SWELDENS, W., SCHRÖDER, P., "Normal meshes". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, *SIGGRAPH '00*, pp. 95–102, ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 2000.

[25] SANDER, P. V., WOOD, Z. J., GORTLER, S. J., SNYDER, J., HOPPE, H., "Multi-chart geometry images". In: *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, *SGP '03*, pp. 146–155, Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2003.

[26] TAUBIN, G., "A signal processing approach to fair surface design". In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, *SIGGRAPH '95*, pp. 351–358, ACM: New York, NY, USA, 1995.

[27] DESBRUN, M., MEYER, M., SCHRÖDER, P., BARR, A. H., "Implicit fairing of irregular meshes using diffusion and curvature flow". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, *SIGGRAPH '99*, pp. 317–324, ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1999.

[28] LIU, L., TAI, C.-L., JI, Z., WANG, G., "Non-Iterative Approach for Global Mesh Optimization", .

[29] SORKINE, O., COHEN-OR, D., "Least-squares Meshes". In: *Proceedings of Shape Modeling International*, pp. 191–199, IEEE Computer Society Press, 2004.

[30] NEALEN, A., IGARASHI, T., SORKINE, O., ALEXA, M., "Laplacian mesh optimization". In: *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, *GRAPHITE '06*, pp. 381–389, ACM: New York, NY, USA, 2006.

[31] GOLUB, G. H., VAN LOAN, C. F., *Matrix computations (3rd ed.)*. Johns Hopkins University Press: Baltimore, MD, USA, 1996.

[32] SORKINE, O., *Laplacian Mesh Processing*, Ph.D. Thesis, School of Computer Science, Tel Aviv University, 2006.

[33] BOTSCH, M., BOMMES, D., KOBBELT, L., "Efficient linear system solvers for mesh processing". In: *Proceedings of the 11th IMA international conference on Mathematics of Surfaces*, *IMA'05*, pp. 62–83, Springer-Verlag: Berlin, Heidelberg, 2005.

[34] FOLEY, J. D., VAN DAM, A., FEINER, S. K., HUGHES, J. F., *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1990.

[35] GCG, "gcgLIB", `http://www.gcg.ufjf.br`, 2012.

[36] PAMPANELLI, P. P., PEÇANHA, J. A. P., CAMPOS, A. M., VIEIRA, M. B., LOBOSCO, M., DANTAS, S. D. O., "Rectangular Hexagonal Mesh Generation for Parametric Modeling". In: *Proceedings of the 2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*, *SIBGRAPI '09*, pp. 120–125, IEEE Computer Society: Washington, DC, USA, 2009.

[37] LENNARD-JONES, J. E., "Cohesion", *Proceedings of the Physical Society*, v. 43, n. 5, pp. 461–482, Sept. 1931.

[38] BRENNER, D. W., SHENDEROVA, O. A., HARRISON, J. A., STUART, S. J., NI, B., SINNOTT, S. B., "A second-generation reactive empirical bond order (REBO) potential energy expression for hydrocarbons", *Journal of Physics: Condensed Matter*, v. 14, n. 4, pp. 783, 2002.

[39] WANG, Z., *Reactive empirical bond-order (REBO) potential*, 01 2006.

[40] SOUZA, A., FERREIRA, R., PEÇANHA, J., CAMPOS, F., LOBOSCO, M., VIEIRA, M., DANTAS, S., "Simulação de Compostos Ferromagnéticos: Características Para Qualidade da Ferramenta", *XII Encontro de Modelagem Computacional*, 2009.

[41] PEÇANHA, J., CAMPOS, A., PAMPANELLI, P., LOBOSCO, M., VIEIRA, M., DANTAS, S., "Um Modelo Computacional para Simulação de Interação de Spins em Elementos e Compostos Magnéticos", *XI Encontro de Modelagem Computacional*, 2008.

[42] CAMPOS, A. M., PEÇANHA, J. A. P., PAMPANELLI, P., DE ALMEIDA, R. B., LOBOSCO, M., VIEIRA, M. B., DE O. DANTAS, S., "Parallel implementation of the heisenberg model using Monte Carlo on GPGPU". In: *Proceedings of the 2011 international conference on Computational science and its applications - Volume Part III*, *ICCSA'11*, pp. 654–667, Springer-Verlag: Berlin, Heidelberg, 2011.