6
Refinamento

Este capítulo descreve como serão aplicadas as operações de refinamento à malha base a fim de se obter uma representação em multi-resolução. O processo de refinamento é feito essencialmente em duas etapas: Subdivisão (seção 6.1) e Adaptação (seção 6.2) dos triângulos. É importante destacar que estas etapas serão aplicadas paralelamente aos triângulos da malha. A função **RefinarMalhaBase**, que foi chamada na função **ExtrairMalhaMulti** (seção 4.3), mostra o processo do refinamento. As funções **IniciarRefinamento** e **AplicarRefinamentoAdapt** estão descritas na seção 6.3.

Algoritmo 14 RefinarMalhaBase (malha base M, erro refinamento e)

IniciarRefinamento (M, e)
AplicarRefinamentoAdapt(M, e)

6.1
Subdivisão dos Triângulos

Nesta etapa os triângulos são subdivididos visando a geração da estrutura hierárquica que representa a malha. Durante este processo há duas questões importantes a serem resolvidas. A primeira se refere ao critério que será utilizado para decidir se um triângulo será subdividido ou não. A segunda questão diz respeito ao conjunto de regras ou heurísticas que serão utilizadas para dividir o triângulo, ou seja, como serão conectados os novos conjuntos de vértices e arestas gerados. É importante destacar que esta etapa da subdivisão trata apenas da alteração da conectividade (topologia) da malha durante o refinamento, ou seja, não serão tratadas as alterações geométricas. Estas últimas serão tratadas durante a adaptação dos triângulos (seção 6.2).

6.1.1 Critério para Subdivisão dos Triângulos

De um modo geral, o critério utilizado para decidir se um triângulo T deve ser dividido baseia-se na variação da curvatura ao longo da região da isosuperfície S representada por T. Em [47], Wood utilizou a variância das distâncias de T à isosuperfície, como critério para dividir T. Tomando-se alguns pontos $p_1, p_2, ..., p_n$ sobre o triângulo T, e tomando o conjunto das distâncias $D = \{D_S(p_1), D_S(p_2), ...D_S(p_n)\}$ dos p_i a S, a variância $V_T(D)$ é definida como $V_T(D) = E(D^2) - E(D)^2$, onde $E(D) = (1/n)\sum_{i=1}^{n} D_S(p_i)$ é a média das distâncias dos p_i's a S.

Se a variância for grande ($V_T(D) > \delta$), significa que a curvatura varia muito na região representada por T (figura 6.1a). No outro extremo, se a variância for pequena ($V_T(D) \leq \delta$), significa que a curvatura não varia e o triângulo T não precisa ser dividido (figura 6.1b). Neste critério, uma segunda condição pode ser acrescentada no caso de a variância ser maior que um valor pré-definido: somente se as curvaturas nos vértices do triângulo T forem altas em relação à área de T, o triângulo deve ser dividido.

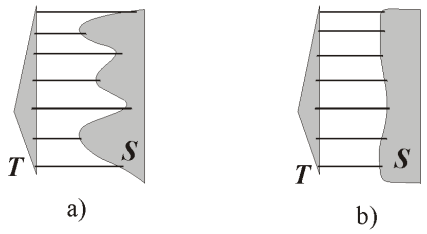

Figura 6.1: Variância das distâncias usada como critério de subdivisão dos triângulos.

6.1.2 Heurística de Subdivisão: Malhas 4-8

O processo de divisão dos triângulos é feito com base nos trabalhos [48, 52, 58]. Trata-se da *Subdivisão 4-8*, onde o refinamento aplicado à malha base gera *malhas 4-8* a partir de operações elementares de *bisseção* de arestas que, pela simplicidade, oferece uma série de vantagens em relação a outros métodos de subdivisão. Além disto, operações de bisseção refinam a malha consistentemente, evitando pós-processamento da malha para evitar erros na triangulações. Outra vantagem desta heurística de refinamento é que malhas 4-8 são *quadrangulações trianguladas*, ou seja combinam

propriedades estruturais de malhas triangulares e retangulares, que são utlizadas na maioria das aplicações.

Os métodos de refinamento de uma malha estão tipicamente relacionados com *tilings* regulares, que são revestimentos ou mosaicos do plano compostos por polígonos base que são regulares de n lados e idênticos. Os três tipos de tilings regulares existentes são os que possuem como polígono base: quadrado, triângulo equilátero e hexágono.

A maioria dos métodos de refinamento baseia-se ou em tilings quadrangulares ou em triangulares (não necessariamente regulares). Tanto os tilings triangulares, quanto os tilings quadrangulares possuem importantes propriedades estruturais que podem ser aplicadas às malhas. Há aplicações em que o ideal seria que as malhas pudessem usufruir das propriedades de ambos os tilings.

Uma importante classe de regras de refinamentos são as que mantêm os *tilings invariantes* durante o refinamento, ou seja, os tilings refinados são isomorfos ao grafo do tiling original. Tendo como motivação manter as propriedades dos tilings triangulares e dos quadrangulares e que os tilings sejam invariantes, muitos métodos [48, 52, 58] utilizam heurísticas de refinamento baseando-se nos *Tiling* $[4.8^2]$ *de Laves*:

Tilings $[4.8]^2$ de Lave [58]

São tilings definidos a partir de polígonos base que são triângulos retângulo isósceles (não são tiling regulares) e cujos vértices possuem alternadamente valência 4 ou 8 (figura 6.2a). Tilings $[4.8^2]$ são invariantes e possuem uma rica estrutura que pode ser explorada durante o refinamento, além de possuírem uma série de vantagens em relação aos tilings regulares. A estrutura básica destes tilings é um par de triângulos que forma um quadrado, chamado *bloco básico*, dividido ao longo de uma de suas diagonais. A aresta comum do par de triângulos é chamada de *aresta interna* ao bloco, enquanto as demais arestas são chamadas de *arestas externas* (figura 6.2b). O tiling $[4.8^2]$ forma uma *quadrangulação triangulada*.

Malha 4-8 [48, 52, 58]

Uma malha é chamada de *malha regular 4-8* se ela possui a mesma conectividade de um tiling $[4.8^2]$. O termo 'regular' aqui é empregado no sentido de que a valência dos vértices será sempre 4 ou 8. Assim, uma malha 4-8 possui a mesma estrutura de blocos (quadrangulação triangulada)

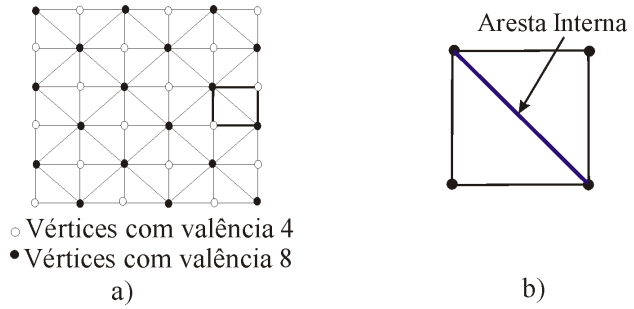

Figura 6.2: a)Malha 4-8 (tiling $[4.8^2]$). b)Bloco.

que o tiling $[4.8^2]$ e suas arestas podem similarmente ser classificadas como arestas internas e arestas externas. Cada triângulo possui uma aresta interna e duas arestas externas. Com isto é possível definir uma operação primitiva do refinamento que é a operação de *bisseção* de um bloco. A estrutura de dados utilizada é baseada na representação *halfedge* [48], onde as informações topológicas armazenadas são basicamente para definir halfedge, edge, face e vértice, conforme mostra a figura 6.3. Todas as informações de conectividades e vizinhanças necessárias podem ser inferidas a partir destas informações. O campo *subdiv* na halfedge terá fundamental importância no processo de bisseção.

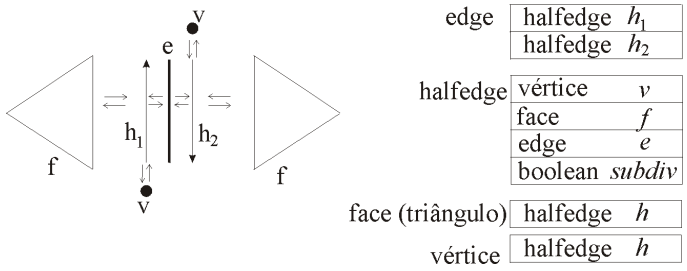

Figura 6.3: Estrutura topológica básica.

Bisseção

É uma heurística de refinamento do bloco definida a partir dos seguintes passos:

- (1) Inserir um vértice, chamado *vértice de subdivisão*, na aresta interna ao bloco.
- (2) Subdividir cada triângulo do bloco em dois sub-triângulos, ligando o vértice de subdivisão aos vértices opostos do triângulo (figura 6.4a).

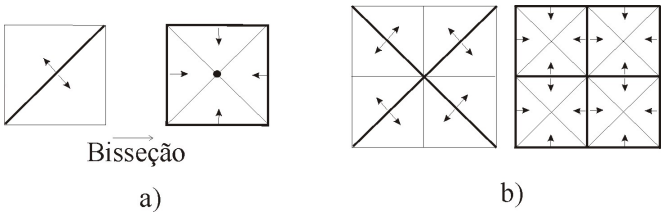

Figura 6.4: a)Bisseção de um bloco. b)Bisseção aplicada a 4 blocos.

A aresta interna ao bloco é uma de *aresta de subdivisão*, assim cada triângulo do bloco possui uma única aresta de subdivisão. Na implementação desta estrutura [48, 58] em cada triângulo é armazenada a halfedge da aresta de subdivisão daquele triângulo (fig 6.3). Na implementação, uma aresta é classificada como sendo de subdivisão se pelo menos uma de suas halfedges for de subdivisão (o campo *subdiv* da halfedge indica se ela é de subdivisão). Na figura 6.4 as arestas de subdivisão estão em negrito e as setas saindo das arestas indicam a halfedge (de subdivisão) de cada faces. A figura 6.4a mostra a bisseção aplicada ao bloco. É importante observar que, após a bisseção, as arestas de subdivisão passam a ser as que antes eram as arestas externas ao bloco. O pseudo código da bisseção de uma aresta de subdivisão e é mostrado na função **AplicarBissecao** (supor que há um operador $vert(i)$ definido na aresta e que retorna o seu vértice adjacente i). A função **AproximarSuperfície** move o novo vértice para a iso-superfície e será descrita na seção 6.2.

Algoritmo 15 AplicarBissecao (aresta de subdivisão e)

$v_1 \leftarrow e.vert(1)$, $v_2 \leftarrow e.vert(2)$,
Criar um novo vértice v, tal que $v \leftarrow 0.5(v_1 + v_2)$
Subdividir cada um dos triângulos que compartilham e
Atualizar as informações sobre as arestas de subdivisão
AproximarSuperficie (v_1, v_2,v)
Retornar v

Operações de bisseção em uma malha regular 4-8 produzem malhas regulares 4-8 (tilings invariantes), como mostra a figura 6.4b. É importante destacar que operações de bisseção podem ser aplicadas a malhas com uma topologia arbitrária (irregulares), desde que a malha esteja particionada em blocos de dois triângulos que compartilham uma aresta, ou seja, a malha seja uma *malha quadrangular triangulada* ou simplesmente, seja uma *malha quad-tri*. Portanto, antes de refinar uma malha qualquer através de operações de bisseção, é necessário um pré-processamento para torná-la uma malha quad-tri. Em [58] é descrito um algoritmo eficiente que transforma

uma malha triangular em uma malha quad-tri que possui aproximadamente o dobro do número de triângulos em relação à malha original.

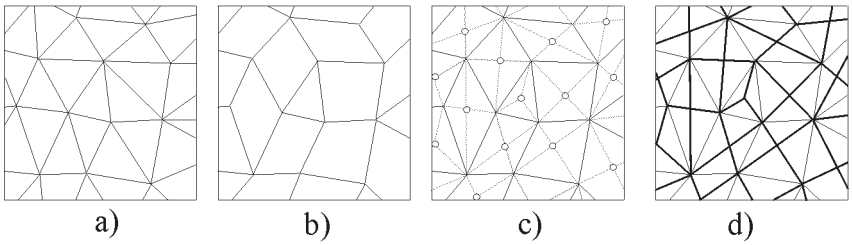

Figura 6.5: a)Malha triangular. b)Malha intermediária: blocos quadrilaterais e triangulares. c)Baricentros. d)Malha quad-tri.

A figura 6.5 mostra a idéia básica deste algoritmo: a figura 6.5a mostra a malha triangular original; as arestas são ordenadas decrescentemente por seus comprimentos em uma *priority queue* Q; para cada aresta e, retirada de Q, marcam-se as arestas que compartilham uma face com e, definindo um bloco quadrilateral (figura 6.5b); arestas marcadas não mais poderão originar novos blocos; após gerados todos os blocos (Q está vazia) é possível que restem alguns triângulos isolados e a estrutura intermediária contém blocos quadrilaterais e blocos triangulares; a próxima etapa (figura 6.5c) insere baricentros dentro de cada bloco, de modo que cada quadrilátero é subdividido em quatro triângulos e cada triângulo é subdividido em 3 triângulos; as arestas dos blocos, antes da inserção dos baricentros, são então definidas como arestas de subdivisão e as novas arestas ligadas aos baricentros são as arestas externas que delimitam os blocos; a figura 6.5d mostra a malha quad-tri resultante, com os blocos destacados. A função **GerarQuadTri** mostra o pseudo-código da idéia acima (supor a existência do campo *marca* na aresta e).

Algoritmo 16 GerarQuadTri (malha base M)

```
Marcar toda aresta e da malha M, ou seja, e.marca ← falso
Ordenar as arestas da malha M e armazená-las em uma priority queue Q
Enquanto Q ≠ ∅
    Retirar uma aresta e de Q
    Se e.marca =falso, Então
      Para toda aresta e1 que compartilha uma face com e
          e1.marca ← verdadeiro
      AplicarBissecao(e)
    FimSe
FimEnquanto
Subdividir cada triângulo que não formou um bloco
```

Refinamento Adaptativo de Malhas Quad-tri

Para refinar uniformemente uma malha quad-tri basta aplicar, em cada nível de refinamento, uma operação de bisseção a todos os blocos da malha. Após um número suficientemente grande de refinamentos, a valência dos vértices é 4 ou 8, exceto nos vértices da malha original. Como estes vértices originais ficam isolados entre si, a malha refinada é uma malha semi-regular. A figura 6.6 mostra o refinamento uniforme de uma malha, após três níveis de refinamento.

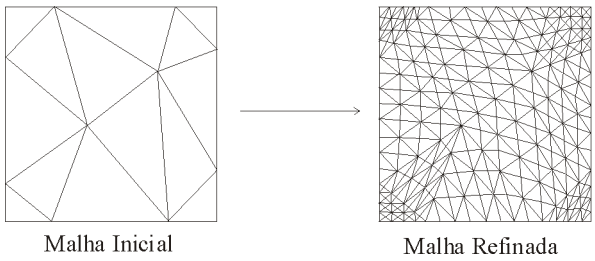

Figura 6.6: Refinamento uniforme.

No caso do refinamento adaptativo, um dos maiores desafios é manter a consistência da malha. Em muitos métodos a subdivisão de um triângulo só se mantém consistente se seus vizinhos também forem subdivididos, o que, por indução, pode acabar se tornando um refinamento uniforme. Este problema pode ser mais facilmente resolvido se o refinamento adaptativo tiver como operação primitiva a bisseção da aresta interna do bloco. Bisseção gera uma estrutura de malha hierárquica que suporta naturalmente resoluções variáveis. Supondo que, pelo critério de subdivisão, uma aresta e deva ser subdividida, há dois casos possíveis a serem analisados: a aresta e pode ser interna ou externa a um bloco. Se e é interna a um bloco b, basta aplicar a bisseção a b, ou seja, nenhum outro bloco precisa ser refinado para que a consistência da malha seja mantida. A figura 6.7a mostra a aresta de subdivisão e interna a um bloco em negrito e a figura 6.7b mostra o resultado do refinamento de e mantendo a consistência entre os triângulos da malha. Por outro lado, se a aresta a de subdivisão é externa a um bloco b, obrigatoriamente deve-se refinar a aresta interna a b e em seguida refinar a, de modo que a consistência seja mantida. Na figura 6.7a a aresta a não é de subdivisão, porém na figura 6.7b, a tornou-se uma aresta de subdivisão, devido ao refinamento anterior (a contém agora uma halfedge de subdivisão). Porém a é externa ao bloco superior em negrito. Neste caso é necessário que a aresta interna ao bloco seja subdividida (figura 6.7c) e em seguida a aresta a é refinada (figura 6.7d).

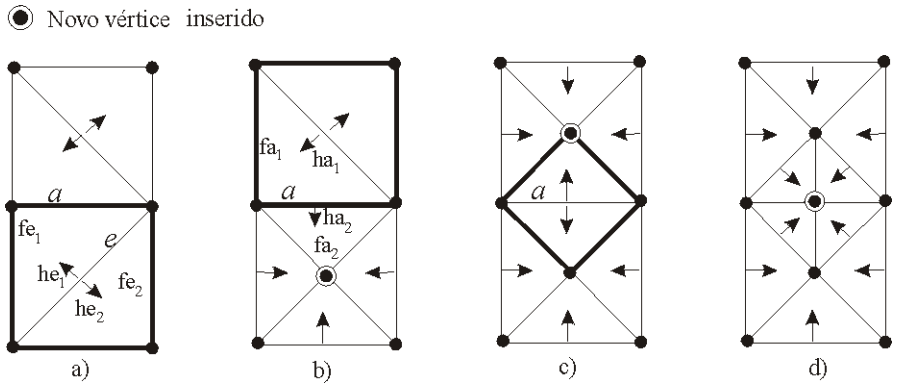

Figura 6.7: Refinamento adaptativo.

Uma forma de saber em quantos passos se realiza o refinamento de uma aresta de subdivisão é mantendo consistentemente armazenada a informação da halfedge de subdivisão de cada face. Seja e uma aresta de subdivisão que tem faces adjacentes fe_1 e fe_2. Se as as halfedges de e coincidirem com as halfedges de subdivisão das faces fe_1 e fe_2, então o refinamento de e se dá em apenas um passo, como ocorreu na figura 6.7a, onde as halfedges de subdivisão da face fe_1 e da face fe_2 coincidem com as halfedges da aresta e. Isto significa que fe_1 e fe_2 estão no mesmo nível hierárquico de refinamento. Já no caso da figura 6.7b, apenas uma halfedge da aresta de subdivisão a coincide com a halfedge de subdivisão da face (no cado a face fa_2). Isto significa que a face fa_1 possui outra aresta de subdivisão, além da aresta a, que deve ser refinada primeiro, pois fa_1 e fa_2 estão em níveis hierárquicos diferentes (figura 6.7c). Só em seguida é que a aresta a pode ser refinada (figura 6.7d). A função **RefinarAresta** mostra como se dá o refinamento adaptativo de uma aresta de subdivisão e adjacente às faces f_1 e f_2 (supor que há um operador $face(i)$ definido na aresta e que retorna sua face adjacente i).

Algoritmo 17 RefinarAresta (aresta de subdivisão e)

Se $e.face(1).h \neq e.h1$ e $e.face(1).h \neq e.h2$, **Então**
  $e_1 \leftarrow$ aresta equivalente a halfedge $e.face(1).h$
  AplicarBissecao(e_1)
FimSe
Se $e.face(2).h \neq e.h1$ e $e.face(2).h \neq e.h2$, **Então**
  $e_1 \leftarrow$ aresta equivalente a halfedge $e.face(2).h$
  AplicarBissecao(e_1)
FimSe
$v \leftarrow$**AplicarBissecao**(e)
Retornar v
FimSe

6.2 Adaptação dos Triângulos

À medida que a malha vai sofrendo mudanças topológicas, através da alteração da conectividade de seus vértices, arestas e faces (seção anterior), é também necessária a adaptação da malha à geometria da superfície. O objetivo desta etapa é a otimização da malha, no sentido de torná-la o mais próximo possível da superfície, além de manter a regularidade dos seus triângulos, sua suavidade e sua boa aparência. Visando manter as propriedades da malha apresentadas na seção 3.1, o método aqui apresentado realiza a otimização da malha basicamente através de duas etapas:

- As coordenadas dos novos vértices devem ser deslocadas para a superfície.
- A vizinhança local destes novos vértices deve ser reparametrizada de modo a manter a regularidade dos triângulos.

6.2.1 Aproximação à Superfície

No método aqui apresentado a aproximação de vértices à superfície é realizada de duas formas: utilizando a transformada de distância e através do cálculo de geodésicas.

Aproximação através da Transformada de Distância

A transformada de distância $\mathcal{T}_S$, aplicada a um objeto S, é uma importante ferramenta utilizada no processamento de diversos objetos gráficos (apêndice A). Dado um objeto $S \subset \mathbb{R}^3$ e um ponto $p \in \mathbb{R}^3$, $\mathcal{T}_S(p)$ representa a menor distância entre p e o objeto S. Assim, se um ponto p está em S, então $\mathcal{T}_S(p) = 0$, ou seja $S = \{p, \mathcal{T}_S(p) = 0\}$. Da mesma forma, se $T_S(p) > 0$, p é um ponto externo a S e se $\mathcal{T}_S(p) < 0$, p é um ponto no interior de S. No caso de volumes, S é uma iso-superfície implícita no dado volumétrico. A figura 6.8a mostra um objeto S, correspondente a uma iso-superfície do volume, enquanto a figura 6.8c mostra a transformada de distância $\mathcal{T}_S$ calculada sobre o plano π (figura 6.8b).

Dado um vértice v com coordenadas p, o deslocamento $\mathcal{F}_S(p)$, que move v em direção à superfície, será calculado com base nos métodos [34, 57],

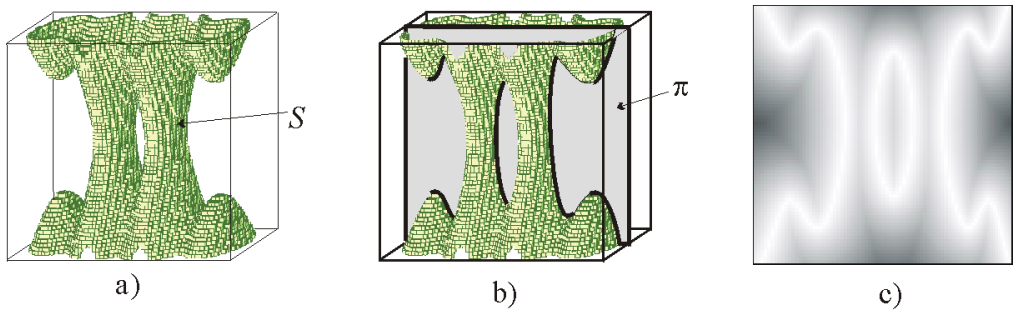

Figura 6.8: a)Iso-superfície S. b)Plano π cortando S. c)$\mathcal{T}_S$ mostrada sobre π.

que utilizam o gradiente da função no ponto p:

$$\mathcal{F}_S(p) = -\mathcal{T}_S(p)\nabla\mathcal{T}_S(p). \tag{6-1}$$

A notação $\mathcal{F}_S$ vem do fato de este deslocamento ser tratado em alguns trabalhos [47, 57] como uma força que puxa o vértice na direção da superfície S. Em [34, 57] em vez da tranformada de distância é utilizada uma função de densidade qualquer $D(p)$ para o cálculo de $\mathcal{F}_S(p)$. A vantagem das funções de distância para o cálculo de $\mathcal{F}_S(p)$ é que elas têm a propriedade de o vetor gradiente $\nabla\mathcal{T}_S(p)$ ser unitário em qualquer ponto p (exceto no eixo medial de S, onde $\nabla\mathcal{T}_S(p) = 0$). Portanto $\mathcal{F}_S(p)$ é um vetor, cujo módulo direção e sentido indicam o deslocamento necessário para que o vértice v coincida com a superfície. A figura 6.9a mostra o eixo medial (linhas brancas) da superfície S (linhas pretas). A figura 6.9b mostra o campo gradiente em volta da superfície S e do seu eixo medial. A figura 6.9c mostra o deslocamento de um ponto interno p e de um ponto externo q em direção à superfície S.

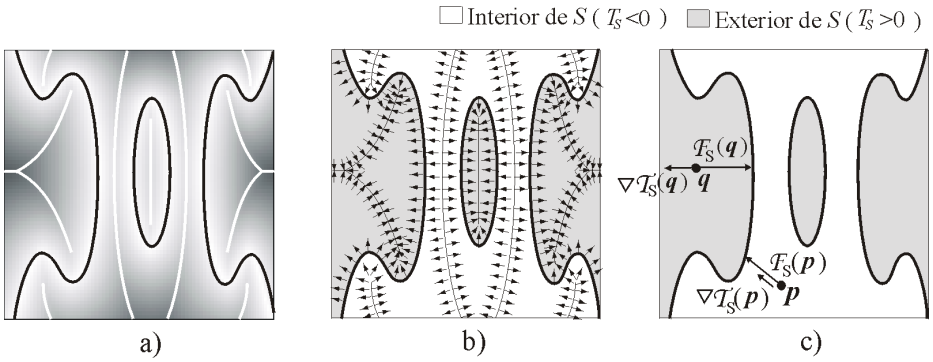

Figura 6.9: a) Eixo Medial. b) Campo gradiente. c) Cálculo do deslocamento.

A seção 6.1.2 mostrou que quando uma aresta (v_1, v_2) da malha base é refinada por bisseção, é criado um novo vértice v entre v_1 e v_2. Quando

este novo vértice é aproximado à superfície através do deslocamento $\mathcal{F}_S(p)$, a nova posição de v passa a ser $p + \mathcal{F}_S(p)$. Há situações em que $\mathcal{F}_S(p)$ pode resultar em um deslocamento errado do vértice v, fazendo com que a topologia da malha seja alterada durante o refinamento, como mostra a figura 6.10. A figura 6.10a mostra o deslocamento correto do vértice v, enquanto a figura 6.10b mostra o vértice v se deslocando para a posição errada. O erro da figura 6.10b ocorreu porque a aresta (v_1, v_2) corta o eixo medial da superfície, fazendo com que o novo vértice criado v esteja mais próximo de uma região da superfície que não está situada entre v_1 e v_2.

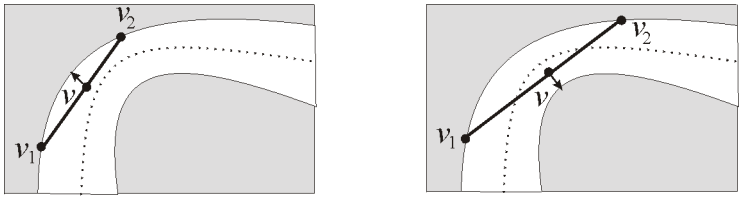

Figura 6.10: a)Deslocamento correto. b) Deslocamento errado.

Como a malha base possui uma baixa resolução, é comum encontrar arestas que interceptem o eixo medial, fazendo com que o problema apontado na figura 6.10b ocorra. A aplicação do deslocamento $\mathcal{F}_S(p)$ ao vértice v só funcionaria bem se v se encontra em uma *vizinhança tubular* da superfície, o que em geral só ocorre após alguns níveis de refinamento da malha. Neste caso, $\mathcal{F}_S(p)$ seria mais indicada para otimização da malha, ou seja, sabendo-se que v já se encontra próximo à superfície S, encontrar as novas coordenadas de v de modo que ele passe a se situar sobre a superfície. O deslocamento $\mathcal{F}_S$ será utilizado durante a reparametrização na seção 6.2.2.

Como proceder então, já que a malha base pode ter arestas interceptando o eixo medial? Uma pergunta natural a ser feita é: dada uma aresta (v_1, v_2), como decidir se ela intercepta ou não o eixo medial? Caso ela intercepte, como encontrar o vetor deslocamento correto que deve ser aplicado ao novo vértice v, gerado pela bisseção? A figura 6.11a mostra como este problema pode ser complicado. Dependendo da geometria da superfície em uma vizinhança de v, fica difícil fazer uma análise do vetor deslocamento. Uma heurística mais simples e segura para encontrar corretamente o vetor deslocamento pode ser elaborada a partir da geodésica entre os dois vértices v_1 e v_2 sobre a superfície.

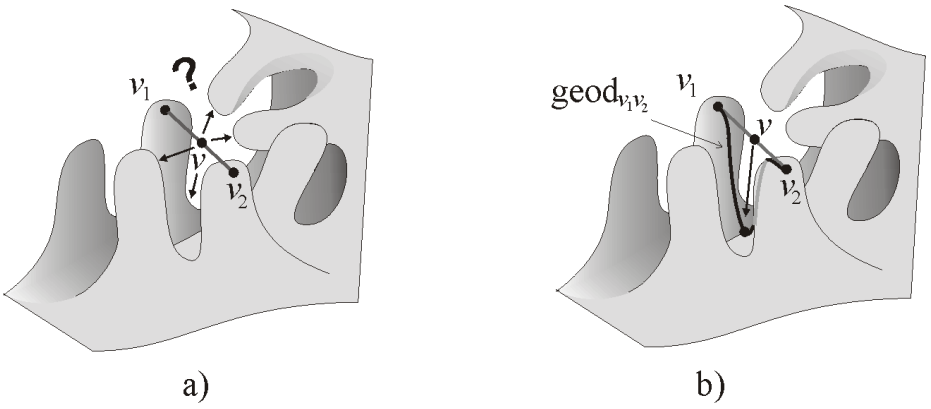

Figura 6.11: a)Deslocamento correto. b) Deslocamento errado.

Aproximação através da Geodésica

Os vértices da malha base foram definidos como os c-vértices do grafo de conectividade que formam os centros dos discos da cobertura de discos. Sabe-se que o grafo de conectividade é a representação da superfície em mais alta resolução. Assim, refinar a malha base significa buscar adequadamente novos c-vértices do grafo de conectividade e incluí-los na malha. Quando a aresta (v_1, v_2) da malha base é refinada, para que a topologia correta da malha seja mantida, o novo vértice v (ponto médio de v_1 e v_2) deve ser associado a um ponto da superfície que esteja em uma região entre v_1 e v_2 (figura 6.10a). Uma forma de calcular corretamente o deslocamento do novo vértice refinado v, é associando v ao ponto médio g_m de $geod_{(v1,v2)}$, geodésica entre v_1 e v_2 no grafo de conectividade $\mathcal{G}$ (figura 6.11b). Uma vez que a geodésica é a menor curva sobre $\mathcal{G}$, situada entre v_1 e v_2, o deslocamento de v para o ponto médio $g_m$vai sempre mover v para uma região entre v_1 e v_2. Nos casos onde a aresta (v_1, v_2) não intercepta o eixo medial, este deslocamento é semelhante ao deslocamento $\mathcal{F}_S$. Devido às restrições aplicadas à geração da cobertura de discos (seção 5.2), existirá apenas uma geodésica entre os vértices v_1 e v_2.

A implementação da geodésica $geod_{(v1,v2)}$ é feita de forma análoga à geração de um disco, mostrada na função **GerarDisco** (seção 5.2.1). O vértice v_1 é tomado como o centro do disco e a propagação para inclusão de novos vértices no disco deve seguir até que o vértice v_2 seja alcançado. Nesta nova versão de geração de discos cada vértice que for atingido pela propagação deve guardar o vértice de onde a propagação partiu. Assim, quando o vértice v_2 for atingido, basta fazer uma busca linear de volta até o v_1 para identificar os vértices que compõem a geodésica $geod_{(v1,v2)}$. A função **CalcularGeodesica** mostra o pseudo-código deste processo.

Algoritmo 18 CalcularGeodesica (vért inicial v_1, vért final v_2)

Gerar um disco sobre o grafo $\mathcal{G}$, com centro em v_1, até encontrar v_2
$v \leftarrow v_2$
Enquanto $v \neq v_1$
    Armazenar v em $geod_{v1,v2}$
    $v \leftarrow v$.anterior
FimEnquanto
Retornar o ponto médio de $geod_{v1,v2}$

O cálculo da geodésica requer alguns cuidados para os vértices da borda. Suponha que o vértice v_1 (da aresta (v_1, v_2) que será refinada) está na borda da malha base. Antes de mover o vértice médio v da aresta (v_1, v_2) com a função **CalcularGeodesica**, o deslocamento deve ser aplicado primeiramente ao vértice v_1 para que ele seja deslocado para borda da superfície. Este deslocamento de v_1 é necessário porque a primeira vez que a aresta (v_1, v_2) for refinada, v_1 é centro de uma célula de Voronoi e pode estar afastado da borda da superfície em si (grafo de conectividade), como mostra a figura 6.12a.

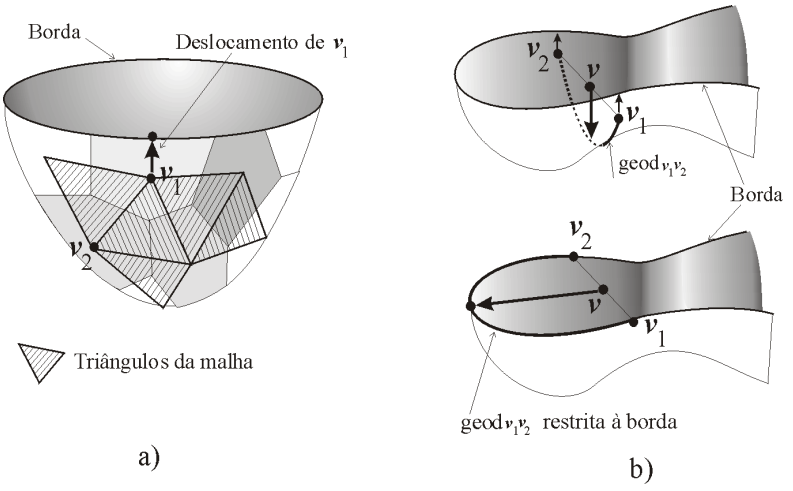

Figura 6.12: a)Deslocamento de v_1 para a borda. b) Deslocamento do vértice v para a borda.

A função **MoverParaBorda** mostra o deslocamento do vértice v_1: deve-se gerar um disco (função **GerarDisco**) com centro em v_1, mas a propagação deve seguir até que um c-vértice v_{fim} que esteja na borda do grafo $\mathcal{G}$ seja encontrado. O vértice v_1 deve ser deslocado para a mesma posição de v_{fim} (para saber se um c-vértice v do grafo $\mathcal{G}$ está na borda, basta utilizar um *flag* que deve ser iniciado durante a geração de $\mathcal{G}$, indicando se v foi gerado a partir de um voxel da borda do volume).

Algoritmo 19 MoverParaBorda (vertice v_1)

Gerar um disco sobre $\mathcal{G}$, com centro em v_1, até encontrar um vért v_{fim} da borda de $\mathcal{G}$
$v_1 \leftarrow v_{fim}$

Se além de v_1, o vértice v_2 também estiver na borda, ambos devem ser deslocados para a borda utilizando a função **MoverParaBorda** (figura 6.12b superior) . Mas, neste caso, o vértice médio v não poderá ser deslocado através da função **CalcularGeodesica**. A questão é que como tanto v_1 quanto v_2 estão na borda, seria razoável que o ponto médio v também estivesse na borda, ou seja, deslocar v para o ponto médio da geodésica $geod_{v1,v2}$ não parece uma boa idéia, como mostra a figura 6.12b superior. Uma outra função, chamada **CalcularGeodBorda**, calcula a geodésica restrita aos vértices da borda e move v para seu ponto médio (figura 6.12b inferior).

Algoritmo 20 CalcularGeodBorda (vert inicial v_1, vért final v_2)

Gerar um disco sobre o grafo $\mathcal{G}$, com centro em v_1, até encontrar v_2, colocando no disco apenas c-vértices que estejam na borda do grafo
$v \leftarrow v_2$
Enquanto $v \neq v_1$
    Armazenar v em $geod_{v1,v2}$
    $v \leftarrow v$.anterior
FimEnquanto
Retornar o ponto médio de $geod_{v1,v2}$

Finalmente a função para aproximar um vértice à superfície pode ser implementada como mostra o pseudo código abaixo. Esta função é utilizada pela função **AplicarBissecao** para mover o vértice para o superfície durante o refinamento.

Algoritmo 21 AproximarSuperficie (v inicial v_1, v final v_2, v médio v)

Se v_1 é um vértice da borda, **Então**
    MoverParaBorda(v_1)
Se v_2 é um vértice da borda, **Então**
    MoverParaBorda(v_2)
Se v_1 e v_2 são vértices da borda, **Então**
    $v \leftarrow$**CalcularGeodBorda**(v_1,v_2)
Senão
    $v \leftarrow$**CalcularGeodesica** (v_1,v_2)

6.2.2 Reparametrização

À medida que novos vértices são gerados durante o refinamento e aproximados à superfície, os triângulos da malha tendem a se tornar menos regulares, mesmo partindo de uma malha base com triângulos regulares. Com isto, faz-se necessário reparametrizar localmente os triângulos adjacentes a um novo vértice v inserido na malha, de modo a manter a regularidade dos mesmos. Este processo é feito aplicando-se ao vértice v um deslocamento $\mathcal{F}_T$ em uma direção tangente à superfície. Este deslocamento é definido como a componente tangencial do operador *Laplaciano* U, definido como [47, 57]:

$$U(p) = \frac{1}{n}\sum_{i=1}^{n}(p_i - p), \tag{6-2}$$

onde p representa as coordenadas de v e p_i são as coordenadas dos vizinhos de v. O deslocamento $\mathcal{F}_T$ (componente tangencial) é definido como $\mathcal{F}_T(p) = \tau[U(p) - (U(p).\vec{n})\vec{n}]$, onde τ é uma constante e $\vec{n}$ é a normal ao vértice v. A figura 6.13 mostra como $\mathcal{F}_T$ é definido.

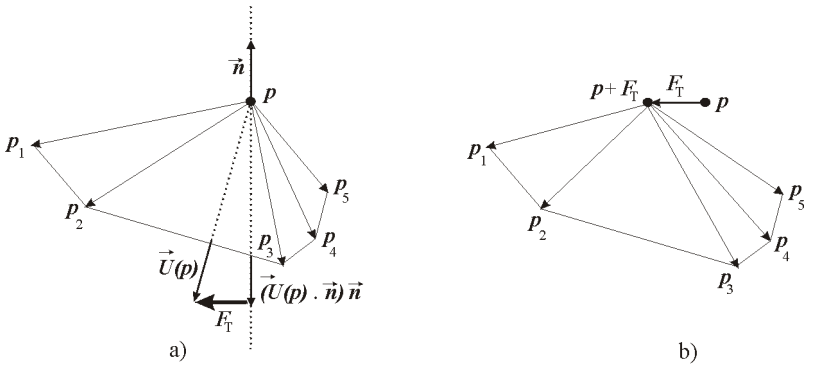

Figura 6.13: a)Coordenada original: p. b) Componente Laplaciana: $p + \mathcal{F}_T(p)$.

A aplicação do deslocamento tangencial $\mathcal{F}_T$ ao vértice v otimiza os triângulos que compartilham v, porém pode afastar o vértice v da superfície S (figura 6.14a).

Para garantir que v estará sobre a superfície, pode-se aplicar o deslocamento $\mathcal{F}_S$ (seção 6.2.1), uma vez que o vértice já se encontra próximo à superficie (figura 6.14b). Pode-se questionar se a aplicação do deslocamento $\mathcal{F}_S$ ao vértice v não afeta o resultado obtido anteriormente pelo deslocamento $\mathcal{F}_T$, ou vice-versa. A questão é que $\mathcal{F}_T$ e $\mathcal{F}_S$ atuam no vértice em direções ortogonais: enquanto $\mathcal{F}_T$ desloca v na direção tangente à superfície, $\mathcal{F}_S$ desloca o vértice na direção normal (direção do gradiente),

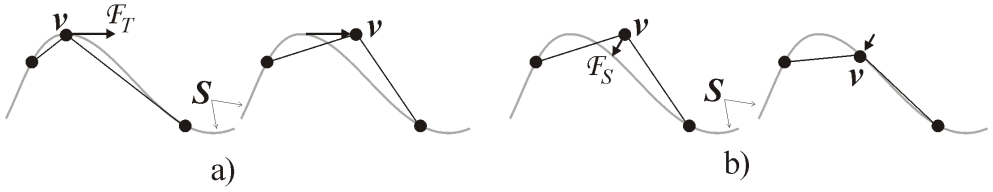

Figura 6.14: a)Aplicação da componente tangencial. b) Aproximação à superfície.

que são direções independentes. Além disso estas forças atuam localmente ao vértice v, não interferindo em outras regiões, o que facilita o controle da otimização da malha. A figura 6.14a mostra o resultado da componente tangente afastando o vértice v da superfície; a figura 6.14b mostra a atuação da componente $\mathcal{F}_S$ aproximando v à superfície.

A função **AplicarCompTanLap** mostra o pseudo código do deslocameneto tangencial $\mathcal{F}_T$, seguido do deslocamento $\mathcal{F}_S$.

Algoritmo 22 AplicarCompTanLap (vértice v)

$p \leftarrow v$.coord

$U(p) \leftarrow \frac{1}{n}\sum_{i=1}^{n}(p_i - p)$
$\mathcal{F}_T(p) \leftarrow \tau[U(p) - (U(p).\vec{n})\vec{n}]$
$p \leftarrow p + \mathcal{F}_T(p)$

$\mathcal{F}_S(p) \leftarrow -\mathcal{T}_S(p)\nabla\mathcal{T}_S(p)$
$p \leftarrow p + \mathcal{F}_S(p)$
$v.coord \leftarrow p$

6.3 Implementação do Refinamento

Esta seção descreve as funções **IniciarRefinamento** e **AplicarRefinamentoAdapt**, chamadas na função principal do refinamento, **RefinarMalhaBase** no início deste capítulo. Estas duas funções fazem simultaneamente a subdivisão dos triângulos (6.1), de acordo com a operação básica de bisseção, e a adaptação dos triângulos à geometria da superfície (6.2).

A função **IniciarRefinamento** converte a malha base M em uma malha quad-tri, através da chamada à função **GerarQuadTri** (seção 6.1.2).

Lembre-se que ao converter a malha em uma quad-tri, os novos vértices inseridos durante a bisseção já são deslocados para a superfície.

Algoritmo 23 IniciarRefinamento (malha M, *erro*, p. queue Q_{REF})

GerarQuadTri (M)
Para todo vértice v da quad-tri M
    AplicarCompTanLap(v)
Para toda aresta $e \in M$ cujo critério de subdivisão é maior que *erro*
    Inserir e na priority queue Q_{REF}

Em seguida, a função **AplicarCompTanLap** faz a reparametrização dos vértices da quad-tri. Após a reparametrização, o critério de subdivisão é verificado em todas as arestas da quad-tri: as arestas que satisfazem ao critério de subdivisão, ou seja, o valor avaliado em cada aresta é maior que um determinado *erro*, serão inseridas na priority queue Q_{REF}, ordenada decrescentemente de acordo com este valor.

A função **AplicarRefinamentoAdapt** percorre as arestas de Q_{REF}, e aplica o refinamento adaptativo às arestas de subdivisão, através da função **RefinarAresta** (seção 6.1.2).

Algoritmo 24 AplicarRefinamentoAdapt (quadtri M, *erro*, pqueue Q_{REF})

Enquanto $Q_{REF} \neq \varnothing$
    Retirar uma aresta e da priority queue Q_{REF}
    Se e for uma aresta de subdivisão
        $v \leftarrow$ **RefinarAresta**(e)
        AplicarCompTanLap(v)
        Inserir em Q_{REF} as novas arestas que satisfazem ao critério de subdivisão
    FimSe
FimEnquanto
Para todo vértice v da quad-tri M
    AplicarCompTanLap(v)

Lembre-se que a função **RefinarAresta** usa o processo de bisseção que já aproxima o vértice refinado à superfície. A etapa seguinte é aplicar a componente tangencial laplaciana aos novos vértices refinados. Em seguida deve-se verificar quais das novas arestas, surgidas durante a chamada de **RefinarAresta**, satisfazem ao critério de subdivisão e inserí-las em Q_{REF}. Terminado o refinamento ($Q_{REF} = \varnothing$), é aplicada uma reparametrização em todos os vértices da malha para garantir um boa razão de aspecto aos triângulos.