

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 18/02/2004.

Assinatura: 

Modelador de visualizador de malhas não estruturadas bidimensionais

Ana Paula Resende Malheiro

Orientador: Prof. Dr. Luis Gustavo Nonato

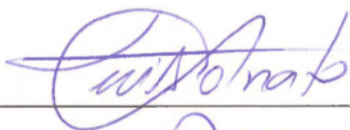
Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciência de Computação e Matemática Computacional.

USP – São Carlos
Fevereiro/2005


Aluno: Ana Paula Resende Malheiro

A Comissão Julgadora:

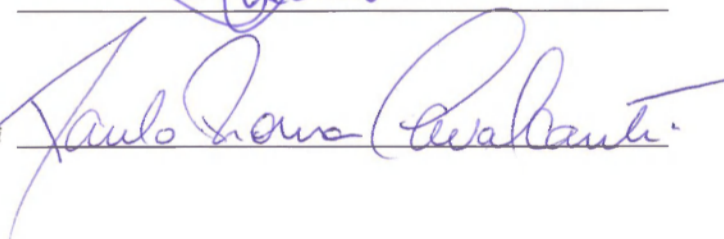
Prof. Dr. Luis Gustavo Nonato



Prof. Dr. Antonio Castelo Filho



Prof. Dr. Paulo Roma Cavalcanti



*Dedico à minha
mamãe Maria com
muito amor...*

Agradecimentos

Agradeço aquele que é a razão da minha existência, que me dá força para lutar dia após dia, sem o qual não chegaria até aqui, ao meu Deus Triunfo.

Aos meus pais Rubens (in memoriam) e Maria. Obrigada meus guerreiros, heróis e exemplos de vida, por me ensinar com amor e por sonhar comigo. Mami te amo!

A minha querida família, parte do meu alicerce, que sempre torceu e orou por mim, que me apoiou e compreendeu minha ausência em momentos especiais de suas vidas.

Aos meus amigos, de longe ou de perto, a todos que um dia passaram pela minha vida e marcaram minha história, meu muito obrigado.

Aos meus Pastores, de hoje ou algum dia, por me incentivarem sempre.

Aos professores do ICMC, especialmente ao Luis Gustavo Nonato e Antonio Castelo Filho, pelo auxílio e também pela maneira linda de ser, admiro muito vocês.

Aos professores do Curso de Matemática, da FCT/UNESP - Presidente Prudente, por participarem da construção do meu conhecimento e caráter.

A Capes pelo suporte financeiro.

Finalizo agradecendo a cada pessoa que compartilhou minhas lágrimas ou sorrisos, com piercing :o), que diretamente ou não tornou os dias em São Carlos possíveis e mais agradáveis, sou muito grata por cada atitude a meu respeito. Que Deus te abençoe.

Ana Paula, Paulinha, ou simplesmente Pepê.

Resumo

Este trabalho apresenta parte de um sistema de simulação integrado para escoamento de fluido incompressível bidimensional, usando malhas não estruturadas, chamado Umflow-2D.

O sistema consiste de três módulos: um módulo modelador, um módulo simulador e um módulo visualizador.

A parte do sistema apresentado neste trabalho é o módulo modelador, o módulo visualizador e o gerador de malhas.

O módulo modelador tem uma interface gráfica que auxilia o usuário a rotular o domínio computacional, a malha, as condições de contorno, e a inicialização de outros dados.

A malha não estruturada pode ser gerada usando dois algoritmos: Algoritmo de Refinamento de Chew e Ruppert.

O módulo de visualização é um sistema que permite a visualização dos resultados gerados pelo módulo de simulação.

Este trabalho usa uma estrutura de dados chamada Singular Handle Edge (SHE) para manipular e representar as malhas.

Abstract

This work presents an integrate simulation system, called Umflow-2D, wich aims at simulating two-dimensional incompressible fluid flow using unstructured meshes.

The system is divided three modules: modeling module, simulation module and visualization module.

In this work we present the modeling and visualization modules. Emphasis will be given to the mesh generation process built in the modeling module.

The modeling module has a graphic interface which helps the user to set up the computational domain, the mesh, the boundary conditions, and other initialization data.

The unstructured mesh can be generated by using two algorithms: Chew and Ruppert, both based on Delaunay refinement.

The visualization module is a system that permits the visualization of the results generated by the simulation module.

This work uses a data structure called Singular Handle Edge (SHE) to handle and represent the meshes.

Sumário

Lista de Figuras	vii
1 Introdução	1
1.1 Objetivo do Trabalho	6
1.2 Organização do Trabalho	7
2 Triangulação de Delaunay	9
2.1 Definições Básicas	10
2.2 Propriedades	14
2.3 Algoritmos	17
2.3.1 Algoritmo de <i>Flipping</i>	17
2.3.2 Algoritmo de <i>Flipping Incremental</i>	18
2.3.3 Algoritmo Incremental	19
2.3.4 Algoritmo de <i>Lifting</i>	20
2.3.5 Algoritmo Dividir para Conquistar	20
2.3.6 Outros Algoritmos	21
3 Refinamento de Delaunay	23
3.1 Algoritmo de Chew	24

Sumário

3.1.1	Definições Básicas	24
3.1.2	Inserção do Circuncentro Melhora a Qualidade	26
3.1.3	Garantia de Parada	27
3.2	Algoritmo de Ruppert	28
3.2.1	Algoritmo	29
3.2.2	Refinamento Dentro do PSLG	31
3.2.3	Convergência do Algoritmo e Boa Qualidade	33
3.3	Ângulos Pequenos	37
3.3.1	Abordagem de Ruppert	37
3.3.2	Abordagem de Shewchuk	40
3.3.3	Abordagem de Miller	42
3.3.4	Abordagem Adotada	45
4	Estrutura de Dados Topológica	49
4.1	Estrutura de Dados SHE	50
4.1.1	Definições Básicas	51
4.1.2	Entidades de Representação da SHE	52
4.2	Operadores de Morse	54
5	O Sistema	59
5.1	Modelador	61
5.2	Visualizador	68
6	Conclusão	71
	Bibliografia	73

Lista de Figuras

1.1	Exemplo de uma malha regular.	3
2.1	Triangulação de Delaunay 2D de um conjunto de pontos arbitrários.	9
2.2	(a) Conjunto de pontos 2D; (b) Fecho convexo do conjunto de pontos representado em (a).	10
2.3	(a) Simplexo de dimensão 0; (b) Simplexo de dimensão 1; (c) Simplexo de dimensão 2.	11
2.4	Conjunto de pontos e segmentos (à esquerda) e sua triangulação com restrição (à direita).	12
2.5	Triangulação de Delaunay de um conjunto de pontos e circuncírculos vazios de dois triângulos.	13
2.6	Triangulação de Delaunay e o seu dual, o Diagrama de Voronoi.	14
2.7	Triangulação de Delaunay e exemplos de círculos vazios das arestas (propriedade de arestas globalmente Delaunay).	15
2.8	(a) Aresta e é localmente Delaunay; (b) Aresta e não é localmente Delaunay; (c) <i>Flipping</i> de e é localmente Delaunay.	15
2.9	(a) Diagonais de um quadrilátero que são <i>flipping</i> uma da outra; (b) Diagonal externa e não é <i>flipável</i> no quadrilátero.	16
2.10	Passos do algoritmo de <i>Flipping</i> Incremental, para a inserção do ponto p	18

Lista de Figuras

2.11	Idéia principal do algoritmo Incremental na inserção de um novo ponto s .	20
2.12	Fecho convexo do parabolóide e projeção das faces do mesmo sobre o plano xy , que é a triangulação de Delaunay.	21
2.13	Algoritmo Dividir para Conquistar (a) Triangulação de duas partes do conjunto de pontos; (b) Concatenação das duas triangulações representadas em (a).	21
3.1	Diagrama de ilustração das relações geométricas: (a) $d = 2r \sin \alpha$; (b) $\angle icj = 2\angle ikj$.	26
3.2	Malha gerada pelo Algoritmo de Refinamento de Chew.	28
3.3	Malha gerada pelos algoritmos de refinamento. (a) Algoritmo de Refinamento de Chew e (b) Algoritmo de Refinamento de Ruppert.	29
3.4	Divisão de segmento invadido.	30
3.5	Caso em que após a divisão do segmento invadido não é necessário a inserção do circuncentro.	31
3.6	Exemplo onde mesmo após a divisão do segmento invadido é necessário a inserção do circuncentro.	31
3.7	Ilustração da idéia principal para provar que o circuncentro é inserido dentro da triangulação.	33
3.8	<i>Local feature size (lfs)</i> de diferentes pontos.	34
3.9	Diferentes casos de raio de inserção.	35
3.10	Círculo ao redor do vértice de entrada.	38
3.11	Ilustração das <i>shield edges</i> e <i>spoke edges</i> .	38
3.12	Primeiro método de divisão da <i>shield edge</i> .	39
3.13	Segundo método de divisão da <i>shield edge</i> .	39
3.14	Triangulação da região dentro de <i>shield</i> , com uma sequência finita de tiras similares sucessivamente menor de triângulos bem formados.	39

Lista de Figuras

3.15	Divisão do segmento s : (a) Nenhum vértice de s possui um ângulo de entrada menor que 60° ; (b) Ambos os vértices de s possuem ângulo de entrada menor que 60°	41
3.16	Grupo de segmento: (a) Grupo de segmento de s ; (b) Dois grupos de segmentos distintos que compartilham o mesmo vértice.	41
3.17	O circuncírculo do $\triangle abc$ não contém nenhum ponto executável, $\angle acb < k$ e a e b são pontos médios em segmentos de entrada não disjuntos distintos, compartilhando o vértice x de entrada, e $\angle axb > \frac{\pi}{3}$	43
3.18	(a) Ângulo pequeno em v entre e_1 e e_2 ; (b) Existem mais arestas incidentes a v , além de e_1 e e_2	46
3.19	Inserção dos novos vértices de acordo com o comprimento de e_1 (c_{min}).	46
3.20	Inserção dos triângulos nas pontas garantem novos ângulos, externos aos triângulos, maiores que 60°	47
3.21	Ilustração da prova matemática do tratamento de ângulos pequenos.	47
3.22	Exemplo onde a inserção dos vértices não garante a formação do triângulo na ponta.	48
4.1	Malha triangular e as entidades da estrutura de dados topológica SHE.	54
4.2	Diagrama de classes da estrutura de dados topológica SHE.	54
4.3	Diferentes casos de handle (a) (-1)-handle, (b) 0-handle, (c) 1-handle e (d) 2-handle	55
4.4	Operador MO_{-1} (MEEETKH).	57
4.5	Operadores MO_0 (a)MST, (b)MET e (c)MEET.	57
4.6	Operadores MO_1 (a)MSSTH, (b) MSETH, (c)MSSTKC e (d)MSETKC.	58
4.7	Operadores MO_2 (a)MSSSTHH, (b)MSSSTHKC, e (c)MSSSTKCC.	58
5.1	Orientação das fronteiras de um PSLG.	63
5.2	Interface com o arquivo .vtk carregado.	63
5.3	(a) Algoritmo de Refinamento de Chew, (b) Algoritmo de Refinamento de Ruppert.	64

Lista de Figuras

5.4	Interface do modelador para entrada dos dados para simulação.	65
5.5	(a) Visualização da velocidade na direção u, (b) Visualização da velocidade na direção v, (c) Visualização da pressão.	69
5.6	Opção de wireframe no visualizador.	70
5.7	Opção de zoom no visualizador: aproximação do canto da Figura 5.3(a). .	70

Introdução

O aumento da capacidade de processamento dos computadores juntamente com o aprimoramento das técnicas numéricas e de geração de malha, têm permitido a realização de simulações complexas de escoamento de fluidos.

Um tipo especial de simulação que tem se beneficiado deste avanço é aquela envolvendo escoamentos com superfícies livres em domínios complexos, pois muitos problemas reais podem ser modelados com o uso de superfícies livres. Exemplos práticos deste tipo de simulação são: injeção em moldes, simulação de reservatórios de água, injeção de substâncias em estruturas orgânicas, entre outros. A importância de tais aplicações tem motivado o aprimoramento e o desenvolvimento de metodologias tanto no campo teórico quanto prático. Métodos tipo elementos finitos, métodos integrais, métodos Lagrangeanos e de coordenadas ortogonais são alguns exemplos das metodologias atualmente empregadas na tentativa de simular tais problemas.

Um ponto importante que tem sido cuidadosamente investigado nesse contexto é o problema de geração de malha, que visa decompor o domínio de interesse, a fim de via-

bilizar a aplicação dos métodos numéricos na aproximação das soluções das equações governantes. A importância do problema de geração de malha se deve ao fato de que uma decomposição não apropriada do domínio pode levar a soluções pouco confiáveis, mesmo com o uso de métodos numéricos eficientes.

Assim, a geração de malha com elementos de “qualidade” é uma componente chave para a simulação computacional de problemas físicos e de engenharias onde métodos numéricos como elementos finitos, diferenças finitas, e o método de volumes finitos são usados (Teng and Wong, 2000).

O objetivo geral na geração de malha é decompor um espaço geométrico em elementos simples (Edelsbrunner, 2000). Os elementos são restritos em tipo e forma, e o número de elementos não deve ser muito grande.

A forma mais simples de uma malha é a malha estruturada. Existem dois tipos de malhas estruturadas: malhas estruturadas geometricamente e estruturadas topologicamente. Exemplo de malha estruturada geometricamente é a malha regular, nesta malha todos elementos possuem o mesmo espaçamento em x e o mesmo espaçamento em y , Figura 1.1. Uma malha é topologicamente estruturada se sua estrutura topológica é isomorfa à de uma malha geometricamente estruturada. Por exemplo, podemos aplicar uma transformação conforme numa malha estruturada geometricamente, para gerar uma malha estruturada topologicamente (Teng and Wong, 2000).

Malhas estruturadas são fáceis de gerar e manipular, além de permitir o uso de uma estrutura de dados simples, o que simplifica a implementação computacional. Além disso, a teoria numérica sobre esse tipo de discretização é bastante entendida. Contudo, o uso de malhas estruturadas limita a aplicabilidade de métodos numéricos para problemas cujo o domínio é simples e cuja função solução é bem comportada.

Para problemas com fronteiras geométricas complexas e com soluções que mudam rapidamente, fa-se necessário uma malha não estruturada que tenha uma característica local adaptativa, tendo seus elementos diferentes tamanhos. Um exemplo da necessidade do uso de uma malha não estruturada é a modelagem de um líquido dentro de uma garrafa, podemos estar interessados apenas no comportamento do líquido próximo da borda, sendo

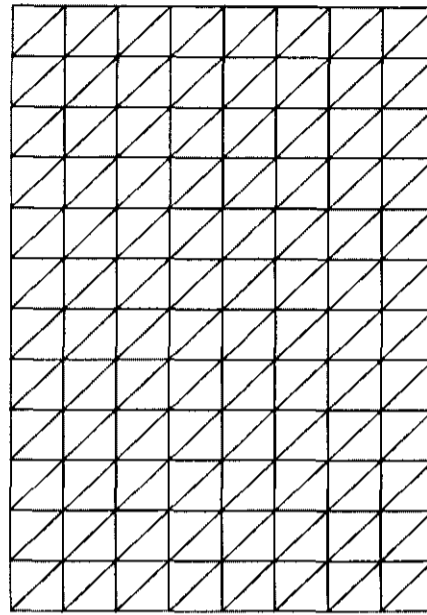


Figura 1.1: Exemplo de uma malha regular.

necessário uma discretização muito densa e fina nesta região, enquanto não é desejável uma malha com muitos pontos na região interior.

Existem diferentes formas de geração de malhas não estruturadas simpliciais, isto é, triangular ou tetraedral. Dentre elas podemos citar como as mais empregadas, as baseadas em:

1. Quadtree em 2D (Octree em 3D).

Primeiramente desenvolvida em 1980 pelo grupo de Mark Shephard em Rensselaer (Owen, 1998). Neste método, realiza-se uma decomposição do espaço em quadrados, buscando trilhar a fronteira do domínio. Quadrados (cubos) contendo o modelo geométrico são recursivamente subdivididos até que a resolução desejada seja encontrada. Após a decomposição, cada quadrado que intersecta a fronteira do domínio é, em geral, substituído por um polígono escolhido a partir de um conjunto de padrões pré-definidos. A triangulação dos quadrados e dos polígonos produz uma malha que aproxima o domínio de interesse. Uma das principais vantagens da abordagem quadtree é a adaptatividade da malha, isto é, elementos menores são ge-

rados próximos às regiões complicadas da fronteira do domínio (Bern et al., 1990). Por outro lado, tais técnicas tendem a introduzir direções preferenciais na malha (Srinivasan et al., 1990).

2. Avanço de Fronteira

É uma família popular de algoritmos, dois dos principais contribuidores para este método são Rainald Lohner na Universidade George Mason (Lohner, 1996) e S. H. Lo na Universidade de Hong Kong (Owen, 1998). Neste método, a fronteira do domínio é inicialmente subdividida e pontos são inseridos (pontos Steiner) em “camadas” ao redor de cada componente da fronteira (Lo, 1985). Uma fronteira ativa é mantida onde novos elementos são formados, e a fronteira avança para a área restante a fim de criar novos triângulos. É válido notar que esta estratégia de inserção de pontos tende a produzir triângulos orientados com as linhas de fluxo, o que pode aumentar a robustez dos métodos numéricos. Assim, as técnicas de avanço da fronteira têm se mostrado muito apropriadas para problemas de dinâmica de fluidos. Os principais problemas destas técnicas são: a dificuldade de assegurar que a subdivisão inicial da fronteira é adequada e grudar duas frentes que colidem no interior do domínio (Lohner, 1996).

3. Delaunay

Grande parte das técnicas têm utilizado o critério de Delaunay, também chamado de círculo vazio (esfera vazia em 3D). Dado um conjunto de pontos no plano, a triangulação de Delaunay é a única triangulação cujo circuncírculo dos triângulos não possui nenhum dos pontos do conjunto em seu interior (Fortune, 1992). As técnicas de geração de malha baseadas em triangulação de Delaunay têm tido grande destaque na comunidade de geometria computacional. A principal razão para este fato é o surgimento de arcabouços teóricos que permitem analisar de forma segura a qualidade e a organização da malha gerada (Ruppert, 1995) (Edelsbrunner, 2000). Uma das principais limitações desta abordagem está em satisfazer na prática as restrições impostas pela teoria. Por exemplo, se a fronteira do domínio possui ân-

gulos pequenos, a inserção de pontos Steiner precisa ser realizada com um cuidado especial (Shewchuk, 1999).

Alguns métodos buscam unir as vantagens da triangulação de Delunay com avanço da fronteira (Marcum and Weatherill, 1995) ou Delaunay com quadtree (Miller et al., 1995). Tais técnicas em geral produzem bons resultados, sendo um campo promissor de investigação. Um ponto fraco de tal metodologia é o custo computacional, que em geral é um pouco mais elevado.

Boas revisões a respeito de geração de malhas não estruturadas podem ser encontradas no trabalho de Bern e Eppstein (Bern and Eppstein, 1995) e também no artigo de Teng e Wong (Teng and Wong, 2000).

Uma triangulação ótima é a melhor de acordo com algum critério que mede o tamanho, formato e o número de triângulos (Bern and Eppstein, 1995). O tamanho e o formato do triângulo definem a sua qualidade.

No caso 2D, a qualidade de um triângulo abc está relacionada com o seu menor ângulo, que chamaremos de θ . Esta qualidade pode ser classificada de duas maneiras, pela medida do seu maior ângulo ou pela razão de aspecto. Segundo Edelsbrunner (Edelsbrunner, 2000), um bom limite inferior para o menor ângulo implica em bons limites para outras duas expressões de qualidade. O maior ângulo é no máximo $\pi - 2\theta$, assim se o menor ângulo θ é limitado, o maior também é.

O objetivo principal de um gerador de malha é construir uma triangulação T de modo que seu menor ângulo não seja menor que alguma constante, e o número de triângulos em T seja o menor possível. Algumas vezes aparece entre duas arestas de entrada um ângulo pequeno que não pode ser removido, uma maneira razoável de tratar esta dificuldade é aceitar ângulos finos de entrada e tentar isolá-los para que não gerem ângulos ainda menores na triangulação final (Edelsbrunner, 2000). O tratamento de ângulos pequenos é retratado na Seção 3 do Capítulo 3.

1.1 Objetivo do Trabalho

Nos últimos cinco anos, o grupo de Matemática Aplicada do ICMC, tem se dedicado à construção de ambientes de simulação de escoamentos incompressíveis com superfícies livres em duas e três dimensões, o qual denominamos Freeflow-2d e Freeflow-3d. Esses ambientes são constituídos de três módulos distintos e independentes: um módulo de modelagem (modelador), um módulo de simulação (simulador) e um módulo de visualização (visualizador).

Os modeladores disponibilizam uma interface amigável para o usuário definir o domínio geométrico, injetores, fluidos, condições de fronteira e demais parâmetros relacionados com a simulação que se deseja realizar. Esses módulos foram fundamentais para tornar os sistemas Freeflow-2d e Freeflow-3d relativamente fáceis de serem utilizados. Os simuladores, ocupam-se da implementação da discretização, por diferenças finitas, das equações governantes, incluindo toda a gerência das condições de fronteira tanto nas paredes rígidas como na superfície livre. Os visualizadores são constituídos de um sistema que permite a visualização dos resultados e implementa várias ferramentas de visualização utilizando o estado da arte na área de computação gráfica.

Os ambientes Freeflow-2d e Freeflow-3d abordam apenas domínios decompostos em malhas uniformes. Este trabalho é uma nova etapa para o ambiente de simulação Freeflow, que agora permite simulações em domínios decompostos através de malhas não estruturadas. Este novo ambiente recebeu o nome de Umflow-2D.

Nesta nova etapa de desenvolvimento implementamos um modelador e um visualizador para o caso bidimensional, capazes de manipular malhas não estruturadas.

Implementamos também um gerador de malhas robusto e versátil, no qual a técnica adotada foi o refinamento de Delaunay (Shewchuk, 1999), implementado através de dois algoritmos: Algoritmo de Refinamento de Chew e Algoritmo de Refinamento de Ruppert.

A estrutura de dados topológica Singular Handle-Edge (SHE) (Nonato et al., 2003), desenvolvida pelo nosso grupo, foi utilizada como base para o modelador e visualizador.

Este trabalho foi desenvolvido em conjunto com o projeto Simulação Numérica de Es-

coamentos de Fluidos Utilizando Diferenças Finitas Generalizadas, pela mestrandia Fernanda Olegario dos Santos, sob a orientação do Prof. Dr. Antonio Castelo Filho, onde um simulador para malhas não estruturadas foi implementado. Também em conjunto com um projeto de Iniciação Científica, do aluno André Luiz Toyama Carneiro, orientado pelo Prof. Dr. Luis Gustavo Nonato, onde foi implementada a interface gráfica do modelador, estando esta ainda em aprimoramento.

Desta forma, temos um conjunto completo de softwares (modelador, simulador e visualizador) para simulação de escoamentos em domínios complexos, o que representa uma evolução para os sistemas de simulação de fluidos desenvolvidos pelo grupo de Matemática Computacional do ICMC-USP.

O trabalho foi acompanhado por especialistas (um grupo de professores e alunos) de Engenharia de Software, para que todos os módulos fossem de fácil compreensão, bem documentados e fáceis de serem estendidos.

1.2 Organização do Trabalho

A dissertação está organizada da seguinte forma:

Capítulo 2 Triangulação de Delaunay: Neste capítulo descrevemos conceitos básicos, propriedades da triangulação de Delaunay e alguns algoritmos.

Capítulo 3 Refinamento de Delaunay: O Algoritmo de Refinamento de Chew e o Algoritmo de Refinamento de Ruppert são apresentados neste capítulo, assim como o tratamento de ângulos pequenos e a abordagem implementada.

Capítulo 4 Estrutura de Dados Topológica: Apresentamos a estrutura de dados Singular Handle-Edge (SHE) utilizada e também os Operadores de Morse.

Capítulo 5 O Sistema: Neste capítulo apresentamos a metodologia e o resultado do sistema implementado, modelador e visualizador.

Capítulo 6 Conclusão: Neste capítulo concluímos o trabalho apresentado nesta dissertação.

Triangulação de Delaunay

Dentre as inúmeras técnicas de triangulação existentes, para os mais variados propósitos, em particular aquelas relacionadas ao espaço euclidiano de dimensão 2, a triangulação de Delaunay é uma das mais estudadas, possui um número significativo de algoritmos, e está intimamente ligada a outras importantes subdivisões do espaço, como o Diagrama de Voronoi e o Fecho Convexo de um mesmo conjunto de pontos. A Figura 2.1 ilustra a triangulação de Delaunay 2D de um conjunto de pontos arbitrários.

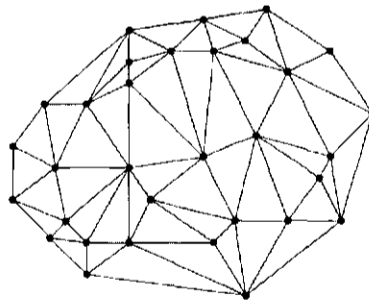


Figura 2.1: Triangulação de Delaunay 2D de um conjunto de pontos arbitrários.

Este capítulo descreve a triangulação de Delaunay, definições básicas, algumas pro-

priedades e algoritmos. Dados complementares sobre a triangulação de Delaunay, podem ser encontrados em (Bern and Eppstein, 1995), (Fortune, 1992), (Miller et al., 1995), (Edelsbrunner, 2000) e (Shewchuk, 1999), da mesma forma que as descrições feitas aqui.

2.1 Definições Básicas

Nesta seção apresentamos uma gama de definições, que usaremos para descrever a teoria da triangulação de Delaunay.

Definição 2.1 (Fecho Convexo) Dado um conjunto de pontos $P = p_1, \dots, p_n$, o fecho convexo de P é o conjunto de todas as combinações lineares convexas dos pontos em P . Uma combinação linear convexa é dada pela equação:

$$\text{Conv}(P) = \left\{ \sum_{i=1}^n \lambda_i p_i, \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0, \lambda_i \in \mathbb{R} \right\}$$

O fecho convexo $\text{Conv}(P)$ é o menor conjunto convexo que contém os pontos de P . Em duas dimensões é o menor polígono convexo que contém todos os pontos. Cada ponto de P ou é um vértice desse polígono (o fecho) ou está no seu interior. Na Figura 2.2 (a) temos um conjunto de pontos e em (b) a representação do seu fecho convexo.

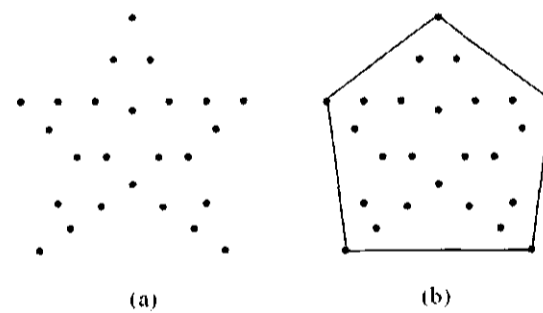


Figura 2.2: (a) Conjunto de pontos 2D; (b) Fecho convexo do conjunto de pontos representado em (a).

Definição 2.2 (Célula) Uma célula convexa afim gerada por v_0, v_1, \dots, v_n é o fecho convexo de v_0, v_1, \dots, v_n .

Cada célula afim contida no bordo de uma célula afim σ é chamada de face de σ .

Definição 2.3 (Dimensão da célula) A dimensão da célula convexa é dada pelo maior número de vetores linearmente independentes do conjunto $\{v_1 - v_0, v_2 - v_0, \dots, v_n - v_0\}$.

Definição 2.4 (Simplexo) Um simplexo de dimensão n em \mathbb{R}^2 , $n = 0, 1, 2$, é uma célula convexa afim de dimensão n gerada por $n + 1$ pontos.

A Figura 2.3 ilustra três simplexos de dimensões diferentes, temos em (a) um simplexo de dimensão 0 (vértice), em (b) um simplexo de dimensão 1 (aresta) e em (c) um simplexo de dimensão 2 (triângulo).

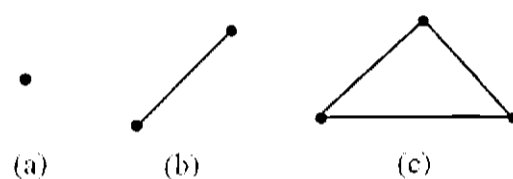


Figura 2.3: (a) Simplexo de dimensão 0; (b) Simplexo de dimensão 1; (c) Simplexo de dimensão 2.

Definição 2.5 (Face) Se um triângulo σ é gerado pelos pontos p_1, p_2, p_3 , então as arestas $(p_i, p_j), i \neq j$, e os pontos $p_i, i = 1, 2, 3$ são chamados de faces σ .

Definição 2.6 (Decomposição celular) Uma coleção C de células convexas em \mathbb{R}^2 é uma decomposição celular (ou complexo celular) de um conjunto $S \subset \mathbb{R}^2$ se:

1. $S = \bigcup_{\sigma \in C} \sigma$.
2. Se $\sigma, \varphi \in C \Rightarrow \sigma \cap \varphi = \emptyset$ ou $\sigma \cap \varphi \in C$ e $\sigma \cap \varphi$ é uma face de σ e φ .
3. Todo compacto de S intersecta um número finito de células de C .

Definição 2.7 (Complexo simplicial) Um complexo simplicial K é uma decomposição celular onde toda célula é um simplexo.

Definição 2.8 (Triangulação) Dado um conjunto de pontos P , não colineares em \mathbb{R}^2 , uma triangulação de P é um complexo simplicial K onde:

1. Todo vértice de K é um ponto de P .
2. Todo vértice ou aresta de K é face de algum triângulo de K .

Definição 2.9 (Estrela do vértice) A estrela do vértice v numa triangulação T é a união de todos simplexos em T que contém v .

Definição 2.10 (Link de um vértice) O link de um vértice v é a união de todas arestas que não contém v e estão nos triângulos que formam a estrela de v .

Definição 2.11 (Triangulação Restrita) Dado um conjunto de pontos P e um conjunto de segmentos S , onde os extremos dos segmentos de S pertencem à P , e quaisquer dois elementos de S não se intersectam a não ser em seus extremos, a triangulação de P restrita por S é uma triangulação de P que inclui todos os segmentos de S em seu conjunto de arestas.

A Figura 2.4, retirada da página http://www.cgal.org/Manual/doc_html/basic_lib/Triangulation_2_ref/Class_Constrained_triangulation_2.html, dia 28 de janeiro de 2004, ilustra um conjunto de pontos e segmentos (à esquerda) e a triangulação restrita (à direita).



Figura 2.4: Conjunto de pontos e segmentos (à esquerda) e sua triangulação com restrição (à direita).

Definição 2.12 (Posição geral) Dado um conjunto de pontos P , dizemos que os pontos de P estão em posição geral quando tais pontos não são colineares e nenhum sub-conjunto de P contém quatro pontos co-circulares.

Definição 2.13 (Triangulação de Delaunay) *Seja $P = p_0, p_1, \dots, p_n$ um conjunto de pontos em posição geral. A triangulação de Delaunay de P é a triangulação cuja união dos simplexos gera o fecho convexo de P e o circuncírculo de todo triângulo não contém nenhum ponto de P em seu interior.*

Na Figura 2.5 encontramos a triangulação de Delaunay de um conjunto de pontos arbitrários. Note que os círculos presentes, que representam os circuncírculos de dois triângulos quaisquer da triangulação, não contém nenhum ponto da triangulação em seus interiores, respeitando Definição 2.13.

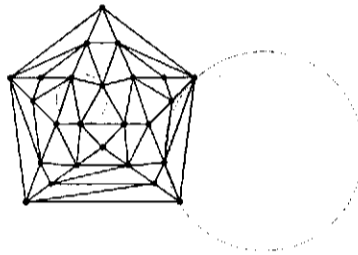


Figura 2.5: Triangulação de Delaunay de um conjunto de pontos e circuncírculos vazios de dois triângulos.

Definição 2.14 (Diagrama de Voronoi) *Seja $P \subset \mathbb{R}^n$ um conjunto de pontos. Para cada, $p, q \in P$, seja $B(p, q) = \{x : \|x - q\| = \|x - p\|\}$ o bissetor de p e q . Considere o semiplano $D(p, q) = \{x : \|x - p\| \leq \|x - q\|\}$ contendo p , e analogamente $D(q, p)$ o semiplano que contém q . Defina-se por $VR(p) = \bigcap_{q \in P, q \neq p} D(p, q)$ a Região de Voronoi do ponto p . O Diagrama de Voronoi é definido por:*

$$V(P) = \bigcup_{p \in P} VR(p).$$

Intuitivamente, $VR(p)$ é o conjunto dos pontos em \mathbb{R}^2 mais próximo de p do que de qualquer outro ponto de P .

A existência da triangulação de Delaunay está ligada ao Diagrama de Voronoi, como seu dual.

Se no circuncentro de cada triângulo da Triangulação de Delaunay for colocado um vértice, e entre cada par de novos vértices que estiverem em triângulos vizinhos, for colocado uma aresta coincidente com a mediatriz, teremos o Diagrama de Voronoi do conjunto de pontos iniciais.

A Figura 2.6, extraída da página <<http://www.cad.ece.ufmg.br/~renato/geocomp/delaunay/92.html>>, dia 28 de janeiro de 2004, retrata a Triangulação de Delaunay, e o seu dual, o Diagrama de Voronoi (em linhas mais fracas).

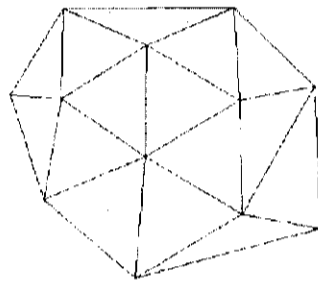


Figura 2.6: Triangulação de Delaunay e o seu dual, o Diagrama de Voronoi.

2.2 Propriedades

Nesta seção descrevemos propriedades da Triangulação de Delaunay, lemas e definições importantes. As provas dos lemas podem ser encontradas em (Shewchuk, 1999).

Definição 2.15 (Aresta globalmente Delaunay) *Numa triangulação uma aresta é globalmente Delaunay se existe um círculo vazio passando pelos seus vértices.*

Toda aresta de bordo é globalmente Delaunay.

Lema 2.1 *Seja T uma triangulação. Se todos os triângulos são Delaunay, ou seja, possuem circunferência vazia, então todas as arestas são globalmente Delaunay e vice-versa.*

Na Figura 2.7 temos a triangulação de Delaunay para um conjunto arbitrário de pontos. Sendo suas arestas globalmente Delaunay, pelo Lema 2.1, ilustramos como exemplo os círculos vazios de algumas delas.

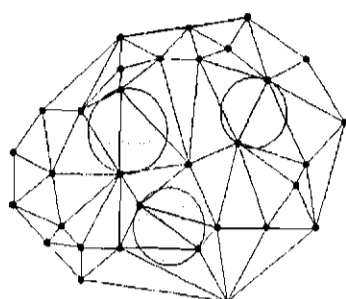


Figura 2.7: Triangulação de Delaunay e exemplos de círculos vazios das arestas (propriedade de arestas globalmente Delaunay).

Definição 2.16 (Aresta localmente Delaunay) *Seja e uma aresta compartilhada por dois triângulos t_1 e t_2 , v_1 e v_2 são os vértices opostos a e em t_1 e t_2 . A aresta e é chamada localmente Delaunay se o circuncírculo de t_1 não inclui v_2 e vice-versa (Figura 2.8 (a)).*

Note que a Figura 2.8 (b) ilustra um exemplo de uma aresta que não é localmente Delaunay.

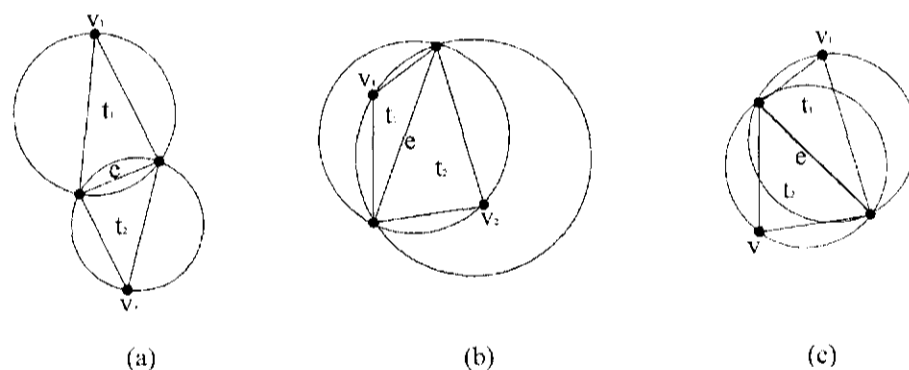


Figura 2.8: (a) Aresta e é localmente Delaunay; (b) Aresta e não é localmente Delaunay; (c) Flipping de e é localmente Delaunay.

Lema 2.2 *Seja T uma triangulação onde todas as arestas são localmente Delaunay. Então todas as arestas de T são globalmente Delaunay.*

Definição 2.17 (Flipping) *Dado um conjunto de quatro pontos e o quadrilátero formado por eles, se este quadrilátero é convexo então ele possui duas diagonais, chamadas flipping uma da outra, Figura 2.9 (a).*

Nem todo quadrilátero possui *flippings* com diagonais internas. Se uma diagonal é externa (quadrilátero não convexo), dizemos que as diagonais não são *flipáveis* no quadrilátero, Figura 2.9 (b).

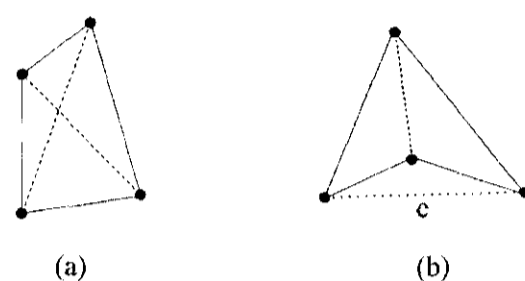


Figura 2.9: (a) Diagonais de um quadrilátero que são *flipping* uma da outra; (b) Diagonal externa c não é *flipável* no quadrilátero.

Lema 2.3 *Seja e uma aresta de uma triangulação T . Então ou e é localmente Delaunay, ou é flipável e o flipping de e é localmente Delaunay (Figura 2.8 (b) e (c)).*

Definição 2.18 (Pontos visíveis) *Dado um conjunto de pontos P e um conjunto de segmentos S , dizemos que dois pontos a e b de P são visíveis se o segmento ab não cruzar nenhum segmento de S , ou seja, não intersectar o interior de nenhum segmento de S .*

Definição 2.19 (Triangulação de Delaunay com restrição) *Seja $P \subset \mathbb{R}^2$ um conjunto de pontos e S um conjunto de segmentos de retas cujos extremos estão em P . Uma aresta ab onde $a, b \in P$ pertence a triangulação de Delaunay com restrição (TDR) se uma das condições abaixo for verdadeira:*

- $ab \in S$
- a e b são visíveis um ao outro e existe um círculo passando por ab que não contém nenhum ponto de P que seja visível a partir dos pontos $x \in ab$.

Definição 2.20 (Aresta localmente Delaunay restrita) *Seja $P \subset \mathbb{R}^2$ um conjunto de pontos, S um conjunto de segmentos de retas cujos extremos estão em P , e K uma triangulação de P restrita à S . Uma aresta ab de K é localmente Delaunay restrita se:*

- $ab \in S$.
- ou ab está no bordo do fecho convexo de P
- ou ab pertence a dois triângulos abc e abd , onde d está fora do circuncírculo de abc , e c está fora do circuncírculo de abd .

Apresentamos algumas propriedades da triangulação de Delaunay, sendo estas propriedades, conseqüências das definições e dos lemas anteriores.

1. A triangulação de Delaunay é única.
2. Todo d -simplexo da triangulação possui uma esfera vazia passando pelos seus vértices.
3. Em \mathbb{R}^2 , a triangulação de Delaunay maximiza o menor dos ângulos internos dos triângulos.
4. Minimiza o maior circuncírculo.
5. Minimiza o maior círculo mínimo (círculo que contém o triângulo e não é o circuncírculo), em \mathbb{R}^2 .

Os itens 3, 4 e 5 são conseqüência do Lema 2.3.

2.3 Algoritmos

Diante do número significativo de algoritmos existentes para construir a triangulação de Delaunay, faremos aqui apenas uma breve descrição dos mais conhecidos.

2.3.1 Algoritmo de *Flipping*

Em \mathbb{R}^2 , o algoritmo de *Flipping* consiste em, a partir de uma triangulação qualquer, *flipar* todas as arestas não localmente Delaunay.

Teorema 2.1 *Dada uma triangulação 2D, o algoritmo de flipping termina em $O(n^2)$, onde n é o número de vértices, produzindo uma triangulação de Delaunay.*

A prova deste teorema pode ser encontrada em (Bern and Eppstein, 1995).

2.3.2 Algoritmo de *Flipping Incremental*

Suponha que a triangulação de Delaunay já tenha sido calculada para os pontos $p_1, p_2, p_3, \dots, p_{i-1}$ e que p_i será inserido como um novo vértice. O algoritmo de *Flipping Incremental* encontra um triângulo abc que contém o ponto p_i , cria os triângulos p_iab , p_ibc e p_ica , e verifica se as arestas ab , bc e ca são localmente Delaunay. Em caso negativo o *flipping* é realizado e as outras arestas no *link* de p_i (Definição 2.10) também são testadas. O algoritmo começa com três vértices no infinito e termina quando não existem mais arestas a serem testadas.

A Figura 2.10 ilustra os passos do algoritmo de *Flipping Incremental* para a inserção do ponto p .

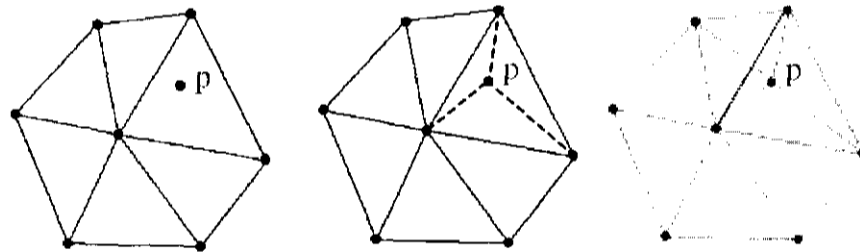


Figura 2.10: Passos do algoritmo de *Flipping Incremental*, para a inserção do ponto p .

Alguns lemas que validam o algoritmo:

Lema 2.4 p_ia , p_ib e p_ic são arestas Delaunay

Prova:

Pela propriedade de circuncírculo vazio da triangulação de Delaunay, considerando o triângulo abc pertencente à triangulação, seu circuncírculo é vazio.

O círculo passando por p_ia , p_ib e p_ic também serão vazios, uma vez que os mesmos serão menores que o circuncírculo do triângulo abc .

Assim as arestas $p_i a$, $p_i b$ e $p_i c$ são arestas (globalmente) Delaunay. ■

Lema 2.5 *Qualquer aresta incidente a p_i por flipping é localmente Delaunay.*

Prova:

Este lema é provado pelo lema 2.3, onde qualquer aresta incidente a p_i é uma aresta da triangulação então ou a aresta é localmente Delaunay, ou é *flipável* e o seu *flipping* é localmente Delaunay. ■

Lema 2.6 *Nenhuma aresta é testada mais de uma vez para cada novo ponto.*

Prova:

Uma aresta s é suspeita se e somente se existe uma aresta s' entre s e p_i , e s' for *flipada*.

Como a aresta *flipada* é incidente em p_i , esta situação pode ocorrer no máximo uma vez a cada inserção. ■

Como cada aresta é testada apenas uma vez, temos $O(n)$ para cada novo ponto inserido.

Este algoritmo também pode ser implementado em 3D.

2.3.3 Algoritmo Incremental

Supondo que a triangulação de Delaunay já tenha sido calculada para os pontos $p_1, p_2, p_3, \dots, p_{i-1}$. O algoritmo Incremental encontra os triângulos que contenham p_i dentro de seus circuncírculos, e remove tais triângulos, gerando um polígono estrelado com relação a p_i , ou seja, p_i “pode ver” os vértices do polígono gerado. Após a remoção dos triângulos, liga-se p_i aos vértices do bordo da região removida (vértices do polígono), Figura 2.11 (Fortune, 1992).

Se p_i está fora da triangulação é necessário ligá-lo também com as arestas visíveis.

Lema 2.7 *Cada aresta com um extremo em p_i e outro no bordo do polígono gerado pela remoção dos triângulos é Delaunay.*

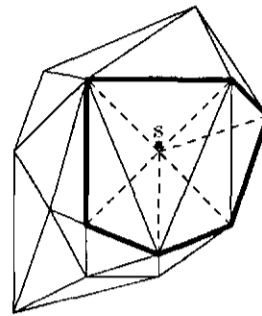


Figura 2.11: Idéia principal do algoritmo Incremental na inserção de um novo ponto s .

Prova: Esta prova segue o raciocínio idêntico ao do lema 2.4.

Como as demais arestas já eram localmente Delaunay, então a triangulação é Delaunay.

2.3.4 Algoritmo de *Lifting*

O algoritmo de *Lifting* tem como princípio básico projetar os pontos a serem triangulados sobre o parabolóide de coordenadas $(x, y, x^2 + y^2)$, onde (x, y) são as coordenadas cartesianas dos pontos, e calcular o fecho convexo do parabolóide. Após o término do cálculo do fecho convexo, remove-se os triângulos da “tampa” do parabolóide. As faces restantes do parabolóide são projetadas sobre o plano xy . Pode ser mostrado que esta projeção é a triangulação de Delaunay dos pontos. A Figura 2.12 retirada de <http://www.cs.berkeley.edu/~jrs/stab.html>, no dia 25 de fevereiro de 2004, ilustra o algoritmo.

2.3.5 Algoritmo Dividir para Conquistar

O algoritmo Dividir para Conquistar, também conhecido como *Divide and Conquer*, divide o conjunto de pontos em duas partes, aproximadamente com o mesmo tamanho, com uma linha vertical. Recursivamente, computa a triangulação das duas partes e então concatena as duas triangulações, Figura 2.13.

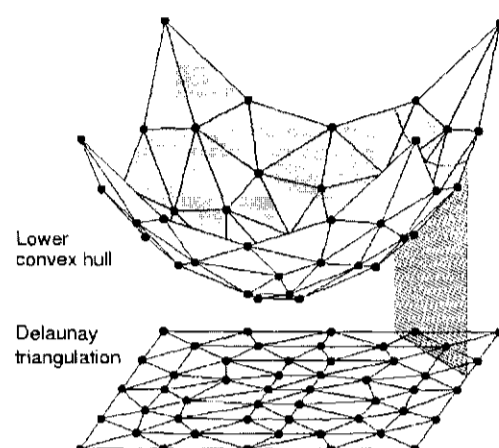


Figura 2.12: Fecho convexo do parabolóide e projeção das faces do mesmo sobre o plano xy , que é a triangulação de Delaunay.

A concatenação leva um tempo linear no pior caso, com isso temos uma ordem de $O(n \lg n)$. Sua implementação é bastante complexa, contudo foi o primeiro algoritmo a obter complexidade $O(n \lg n)$.

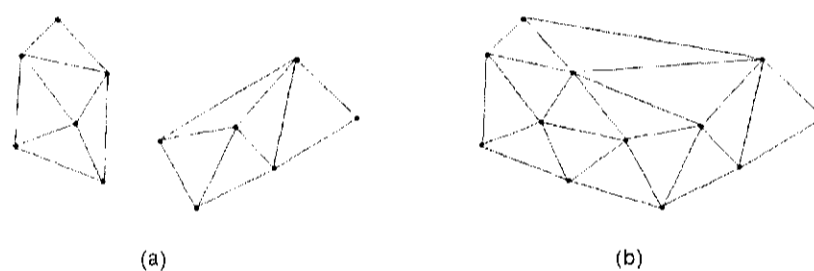


Figura 2.13: Algoritmo Dividir para Conquistar (a) Triangulação de duas partes do conjunto de pontos; (b) Concatenação das duas triangulações representadas em (a).

2.3.6 Outros Algoritmos

Além dos algoritmos descritos, existem ainda vários outros, como o Embrulho para Presente, o *Plane-Sweep* de Fortune, além de um número significativo de algoritmos que calculam o Diagrama de Voronoi (Aurenhammer and Klein, 2000), possibilitando assim a construção da triangulação de Delaunay.

Existem também algoritmos para a construção de uma triangulação de Delaunay com

restrição, realizado não somente sobre um conjunto de vértices, mas recebe como entrada um *Planar Straight Line Graph* (PSLG), que consiste de um conjunto de vértices e segmentos, onde os segmentos pertencentes ao PSLG devem pertencer a malha de saída. Dentre os autores, que implementaram tais algoritmos, podemos citar Shewchuck <<http://www.cs.berkeley.edu/~jrs/mesh/>> e Edelsbrunner (Edelsbrunner and Tan, 1992). No trabalho de Edelsbrunner existe referência de trabalhos anteriores.

Refinamento de Delaunay

Em métodos de elementos finitos a qualidade dos elementos que decompõem o domínio influencia na qualidade da solução das equações. Este fato tem gerado uma grande pesquisa na área de geração de malhas com razão de aspecto garantida, chamada malha de qualidade (Cheng and Dey, 2002). Uma considerável literatura sobre este assunto pode ser encontrada em (Bern and Eppstein, 1995).

Os algoritmos de refinamento de Delaunay, em geral, buscam criar uma triangulação com boa qualidade e um número não muito grande de triângulos. Estes algoritmos, em geral, incluem os centros dos circuncírculos (ou circuncentros) dos triângulos de baixa qualidade para melhorar a qualidade da malha.

Os algoritmos destacados neste capítulo, e que foram implementados, são os algoritmos de Chew e de Ruppert. Chew propôs um método simples de inserção do circuncentro para o problema 2D, que produz uma malha uniforme de triângulos de qualidade (Chew, 1989). Ruppert mostrou como a inserção do circuncentro pode ser usada para produzir uma malha de qualidade com tamanho ótimo (Ruppert, 1995).

3.1 Algoritmo de Chew

O algoritmo de refinamento de Chew é uma técnica de geração de malhas relativamente simples, baseada no trabalho anterior de Chew (Chew, 1989). O refinamento é realizado numa triangulação de Delaunay, sua operação principal é a inserção de pontos, onde a cada inserção a triangulação é refeita localmente. Este processo é repetido até que a malha satisfaça um critério de qualidade.

As malhas geradas pelo algoritmo de Chew possuem algumas características, dentre elas:

- Fronteiras (tanto internas como externas) e vértice especificados pelo usuário são respeitados. Triângulos não cruzam as bordas e os vértices especificados pelo usuário nunca são eliminados.
- Todos os triângulos têm ângulos entre 30 e 120 graus, com exceção dos que respeitam a especificação da borda, ou seja, onde o polígono de entrada já possui um ângulo inferior a 30 graus.
- O usuário pode controlar o tamanho do triângulo dependendo do polígono de entrada.

Esta técnica pode ser estendida para casos 3D gerando uma malha de qualidade para superfícies onde os triângulos apresentam as mesmas características do caso 2D (Chew, 1993).

3.1.1 Definições Básicas

Nesta seção descrevemos algumas definições, lemas e teoremas, necessários para a compreensão da idéia principal do algoritmo de Chew, baseado em (Chew, 1993).

Definição 3.1 (Triângulo bem formado) Chamamos de triângulo bem formado ao triângulo que possui todos os seus ângulos maiores ou iguais a 30 graus.

Definição 3.2 (Triângulo de bom tamanho) *Um triângulo é dito de bom tamanho se ele satisfaz uma função de classificação fornecida pelo usuário.*

A função de classificação pode usar algum critério escolhido pelo usuário, como por exemplo, o triângulo de bom tamanho será aquele que se ajusta dentro de um círculo de raio $\Delta > 0$. Outra possível escolha do usuário para triângulos de bom tamanho é deixar triângulos interiores completamente sem restrição, mas requerer que os triângulos com arestas de bordas, tenham suas arestas de bordas menores que uma constante escolhida. O critério pode assumir um nível arbitrário de complexidade, variando desde uma simples intuição do usuário à uma solução numérica antecipada (Chew, 1993).

Definição 3.3 (Triângulo de qualidade inferior) *Um triângulo que não é bem formado e nem de bom tamanho, chamaremos de triângulo de qualidade inferior.*

Definição 3.4 (Vértice solicitado) *Um vértice solicitado é aquele pertencente à borda ou que é especificamente escolhido pelo usuário.*

Definição 3.5 (Vértice circuncêntrico) *Um vértice circuncêntrico é um vértice adicional que é introduzido pelo algoritmo.*

Os únicos vértices que podem ser removidos durante o algoritmo de Chew são os vértices circuncêntricos, ou seja, os vértices solicitados devem ser retidos.

Algoritmo 1 Algoritmo de Chew

Dada um conjunto de pontos P e um conjunto de segmentos S ;

- 1: Construir a triangulação de Delaunay a partir de P e restrita à S ;
 - 2: **enquanto** todos os triângulos não satisfizerem as definições 3.1 e 3.2 **faça**
 - 3: $t \leftarrow$ maior triângulo que não satisfaz as definições;
 - 4: Inserir o circuncentro de t ;
 - 5: Atualizar a triangulação;
 - 6: **fim do enquanto**
-

3.1.2 Inserção do Circuncentro Melhora a Qualidade

A principal idéia do algoritmo de Chew está em inserir os circuncentros dos triângulos de qualidade inferior, ou seja, a cada interação verifica-se se o triângulo satisfaz os critérios de triângulo bem formado e de bom tamanho, de acordo com as definições 3.1 e 3.2.

O princípio de inserir o circuncentro de triângulos de qualidade inferior está baseado num fato geométrico bastante conhecido, que podemos descrever da seguinte forma:

Dado um triângulo Δ_{ijk} com circuncentro c e circunraio r (Figura 3.1 (a))(Shewchuk, 1999), suponhamos que o comprimento da menor aresta ij é d , e o ângulo oposto desta aresta é $\alpha = \angle ikj$.

Dado $\beta = \angle jkc$, como na Figura 3.1 (b), os triângulos Δkci e Δkcj são isosceles, $\angle kci = 180^\circ - 2(\alpha + \beta)$ e $\angle kcj = 180^\circ - 2\beta$. Subtraindo $\angle kci$ do $\angle kcj$, obtemos que $\angle icj = 2\alpha$. Mesmo que β seja negativo a propriedade permanece.

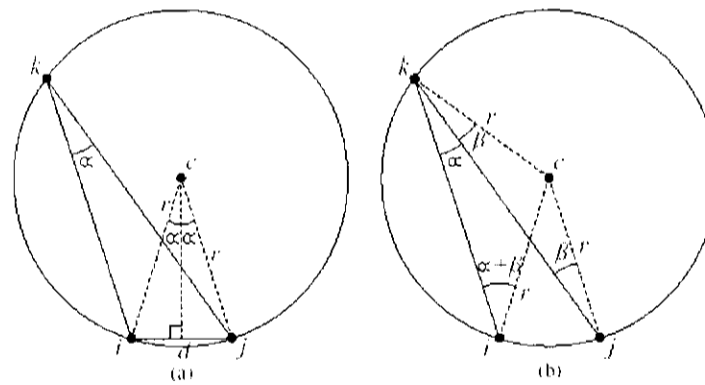


Figura 3.1: Diagrama de ilustração das relações geométricas: (a) $d = 2r \sin \alpha$; (b) $\angle icj = 2\angle ikj$.

Da Figura 3.1 (a) temos que

$$\sin \alpha = \frac{\text{catetooposto}}{\text{hipotenusa}} \Rightarrow \sin \alpha = \frac{\frac{d}{2}}{r} \Rightarrow$$

$$\sin \alpha = \frac{d}{2r}.$$

Se a menor aresta do triângulo tem comprimento d , então α é o menor ângulo. Ao inserirmos o circuncentro do triângulo, obtemos um ângulo oposto a menor aresta d de 2α .

Se B é um limite superior para a razão entre o circunraio e o comprimento da menor aresta de todos os triângulos da malha, então não existe ângulo menor que $\arcsin \frac{1}{2B}$, e vice versa (Shewchuk, 1999). Ou seja, $\sin \alpha = \frac{d}{2r}$ então $\alpha = \arcsin \frac{d}{2r}$ e sendo B o limite superior temos, $\alpha > \arcsin \frac{1}{2B}$, com $B = \limsup \frac{r}{d}$.

O algoritmo de Chew tem $B = 1$, conseqüentemente os ângulos dos triângulos estão limitados entre 30 e 120 graus (Shewchuk, 1999).

Deste modo, qualquer triângulo que apresentar a razão entre o circunraio e o comprimento da menor aresta maior que um ($\frac{r}{d} > 1$), ou seja, o triângulo cujo circunraio for maior que o comprimento da menor aresta ($r > d$) é dividido pela inserção do circuncentro como um novo vértice. A propriedade de Delaunay é mantida, eliminando assim o triângulo de qualidade ruim, pois o circuncírculo do triângulo deve permanecer vazio.

3.1.3 Garantia de Parada

O comprimento de todas as novas arestas criadas pela inserção do circuncentro serão maiores ou iguais ao circunraio, conseqüentemente toda nova aresta tem comprimento maior que a menor aresta do triângulo de qualidade inferior. Assim, se h é o comprimento da menor aresta da triangulação, chegará um exato momento em que o circunraio de todo triângulo será menor que h e não haverá mais espaço para a inserção de vértices, acarretando no término do algoritmo.

Dado o valor h escolhido pelo usuário, que deve ser menor ou igual ao comprimento da menor aresta da triangulação, para evitar problemas com a inserção de vértices fora da fronteira, o algoritmo de Chew subdivide todas as arestas da fronteira com comprimento maior que h , em subsegmentos cujo comprimento está no intervalo de $[h, \sqrt{3}h]$. Posteriormente, é realizado o refinamento dos triângulos internos, inserindo os circuncentros dos triângulos com circunraio maior que h a cada iteração. Pode ser mostrado que com

esta subdivisão das arestas da fronteira, os circuncentros dos triângulos estão sempre no interior da fronteira, o que assegura que realmente o algoritmo irá terminar.

O algoritmo de Chew gera uma malha com densidade quase uniforme, como mostra a Figura 3.2, o que pode gerar uma malha muito refinada. Isto pode ser visto como uma desvantagem, pois mesmo que um triângulo se aproxime de um triângulo equilátero, se seu circunraio for maior que h ele é dividido com a inserção de seu circuncentro. O algoritmo de Ruppert, que descreveremos a seguir tenta sanar esta desvantagem, mantendo na malha os triângulos que se aproximam de triângulos equiláteros.

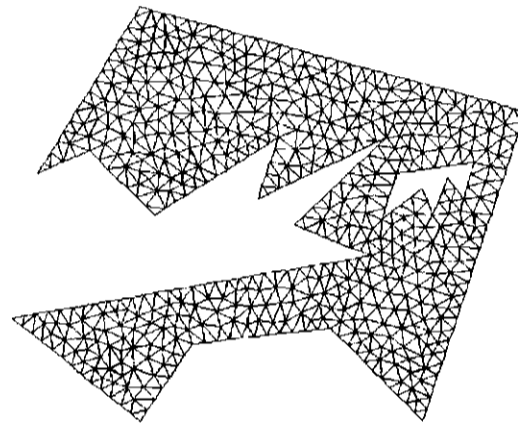


Figura 3.2: Malha gerada pelo Algoritmo de Refinamento de Chew.

3.2 Algoritmo de Ruppert

O algoritmo de refinamento de Jim Ruppert (Ruppert, 1995) para geração de uma malha bidimensional de boa qualidade faz uso iterativo da Triangulação de Delaunay. A cada iteração ocorre a inserção de vértices adicionais, mantendo válida a propriedade de Delaunay, até se obter uma triangulação que satisfaça as condições desejadas de qualidade.

O refinamento pode ser feito numa Triangulação de Delaunay com ou sem restrição, embora a apresentação de Ruppert do seu algoritmo esteja baseada na triangulação sem restrição. A entrada do algoritmo é um PSLG. Assumindo a triangulação de Delaunay apenas dos vértices, ignorando a entrada de segmentos, alguns segmentos que fazem parte

do PSLG podem não aparecer na triangulação, estes segmentos chamamos de segmentos ausentes. Um aspecto interessante do algoritmo de Ruppert é que os segmentos ausentes acabam sendo inseridos como consequência natural do algoritmo (Shewchuk, 1999).

O algoritmo de Ruppert é uma extensão do primeiro algoritmo de Chew, a sua principal operação também é a inserção do circuncentro de triângulo de qualidade inferior. A principal diferença é a variação na densidade dos triângulos, enquanto o algoritmo de Chew gera uma malha mais homogênea, o algoritmo de Ruppert utiliza uma densidade variada, o que acaba gerando uma quantidade menor de triângulos, como pode ser observado na Figura 3.3.

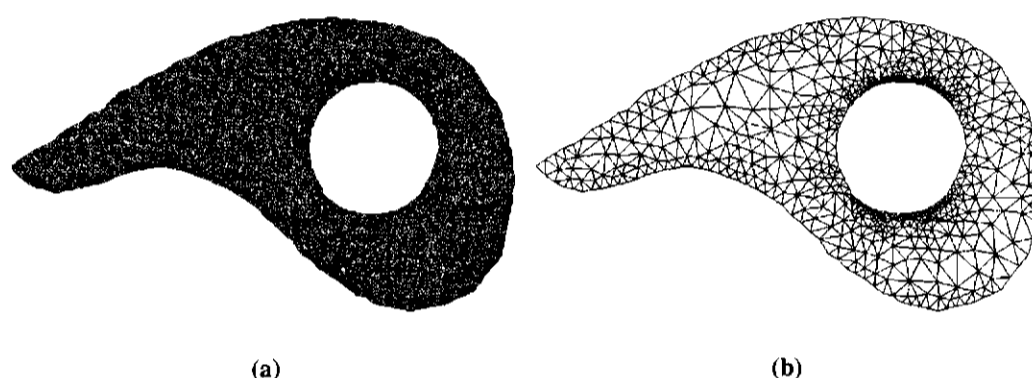


Figura 3.3: Malha gerada pelos algoritmos de refinamento. (a) Algoritmo de Refinamento de Chew e (b) Algoritmo de Refinamento de Ruppert.

Segundo Shewchuk o primeiro algoritmo de geração de malha que fornece a prova teórica da qualidade da malha procurada na prática é possivelmente o algoritmo de Ruppert. Temos a seguir uma seção que contém parte deste fundamento teórico, os demais detalhes podem ser obtidos em (Ruppert, 1995) e (Shewchuk, 1999).

3.2.1 Algoritmo

O objetivo desta seção é apresentar algumas definições, lemas e teoremas que descrevem o fundamento teórico do algoritmo de Ruppert.

Definição 3.6 (Círculo diametral) Círculo diametral de um segmento é o menor círculo

que envolve um segmento.

Definição 3.7 (Segmento invadido) Um segmento é dito segmento invadido se um vértice estiver estritamente dentro do seu círculo diametral.

O segmento que não aparece como uma aresta da triangulação (segmento ausente) é sempre um segmento invadido.

Critérios para a Inserção de Vértices

Primeiro Critério para a Inserção de Vértices

Todo segmento invadido que aparece é dividido em dois subsegmentos através da inserção de um vértice no seu ponto médio, até que não encontre nenhum segmento invadido, como pode ser observado na Figura 3.4 (Shewchuk, 1999).

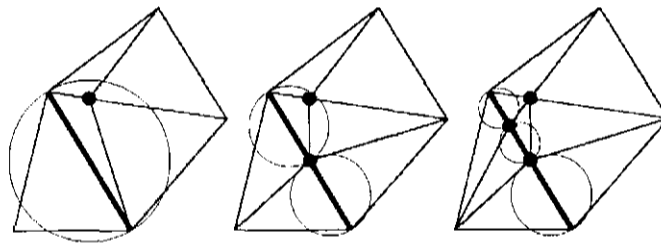


Figura 3.4: Divisão de segmento invadido.

Segundo Critério Para a Inserção de Vértices

Cada triângulo fino é dividido pela inserção de um vértice no seu circuncentro. Um triângulo é considerado fino quando a razão entre o circunraio do triângulo pelo comprimento da menor aresta do mesmo for maior que um dado limite B .

Contudo, se o novo vértice gerar um segmento invadido, cancela-se sua inserção “por um instante”. Primeiramente divide-se todos os segmentos invadidos que apareceria com a sua inserção. Depois, caso o triângulo fino ainda exista, o novo vértice é finalmente inserido. Caso contrário, ou seja, caso o triângulo fino tenha sumido com a divisão dos segmentos invadidos, a inserção do vértice no circuncentro fica definitivamente cancelada.

Na Figura 3.5, adaptação de (Shewchuk, 1999), a inserção do circuncentro gera um segmento invadido, o que cancela a sua inserção, fazendo primeiramente a divisão do segmento que se tornaria segmento invadido. Note que o triângulo anteriormente fino não existe mais. Já na Figura 3.6, também adaptada de (Shewchuk, 1999), temos um exemplo do caso onde mesmo após a divisão do segmento, o triângulo fino ainda existe, devendo agora ser inserido o seu circuncentro.

Uma consequência da discussão acima é que segmentos invadidos têm prioridade sobre triângulos finos durante o refinamento.

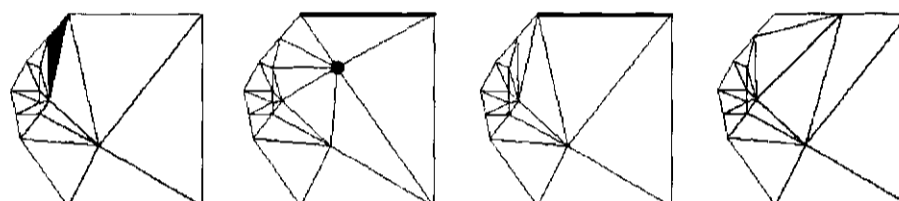


Figura 3.5: Caso em que após a divisão do segmento invadido não é necessário a inserção do circuncentro.

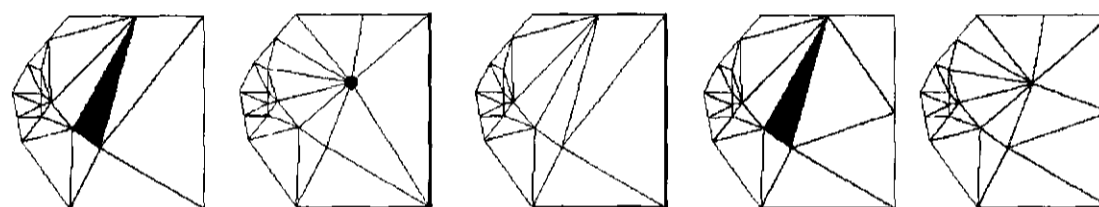


Figura 3.6: Exemplo onde mesmo após a divisão do segmento invadido é necessário a inserção do circuncentro.

3.2.2 Refinamento Dentro do PSLG

Lema 3.1 *Seja T uma triangulação de Delaunay de um PSLG. Suponha que T não tem nenhum segmento invadido. Seja v o circuncentro de algum triângulo t de T . Então v está no interior do PSLG.*

Prova:

Esta prova é feita por contradição.

Algoritmo 2 Algoritmo de Ruppert

Dada um conjunto de pontos P e um conjunto de segmentos S ;

- 1: Construir a triangulação de Delaunay a partir de P ;
- 2: **enquanto** existir segmento invadido **faça**
- 3: Inserir o ponto médio de tal segmento;
- 4: **fim do enquanto**
- 5: **enquanto** existir triângulo fino **faça**
- 6: $x \leftarrow$ circuncentro de um triângulo fino t ;
- 7: **se** x invadir os segmentos s_1, \dots, s_k **então**
- 8: **para** (todo $s_i \in s_1, \dots, s_k$) **faça**
- 9: Inserir o ponto médio de s_i ;
- 10: Atualizar a triangulação;
- 11: **fim do para**
- 12: **se** t ainda existir **então**
- 13: Inserir x ;
- 14: **fim do se**
- 15: **caso contrário**
- 16: Inserir x ;
- 17: **fim do se**
- 18: **fim do enquanto**

Primeiramente supomos que v está fora de T .

Chamamos de c o centróide de t . Como c está dentro de t (por definição) e t é um triângulo de T , podemos afirmar que c está dentro de T . Assim a linha cv cruza algum subsegmento s , como na Figura 3.7, adaptação de (Shewchuk, 1999).

O circuncírculo de t contém uma parte de s , pois cv está inteiramente contida no circuncírculo de t , mas por Delaunay, o circuncírculo não pode conter os vértices de s .

Como s separa os vértices que estão dentro do PSLG dos que estão fora, a parte do circuncírculo que está estritamente dentro do PSLG, está também estritamente dentro do círculo diametral de s .

Dois vértices de t podem ser os vértices de s , mas resta um vértice dentro do círculo diametral, o que representa uma contradição, pois s não é um segmento invadido. Logo, podemos concluir que v está dentro de T . ■

Assim sendo, compreendemos a importância de segmento invadido ter prioridade sobre triângulos finos e porque um circuncentro só pode ser inserido quando não gerar

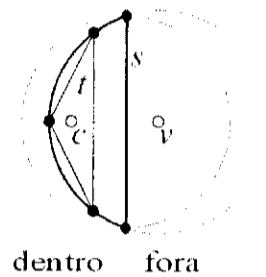


Figura 3.7: Ilustração da idéia principal para provar que o circuncentro é inserido dentro da triangulação.

segmentos invadidos, ou seja, a subdivisão dos segmentos garante que o circuncentro dos triângulos estarão dentro do PSLG.

Uma mudança na implementação do algoritmo de Ruppert permite utilizar um critério de limite superior definido pelo usuário, especificando a área do triângulo ou o comprimento da menor aresta. Este limite superior pode ser fornecido pelo usuário através de uma função que varia de acordo com a localização do triângulo ou da aresta. Os triângulos que excederem ao limite superior desta função são divididos, sendo eles triângulos finos ou não.

3.2.3 Convergência do Algoritmo e Boa Qualidade

Com o intuito de provarmos a convergência do algoritmo de Ruppert, descrevemos algumas definições adicionais, lemas e teoremas. As provas dos lemas e teoremas não descritas aqui podem ser encontradas em (Ruppert, 1995) e (Shewchuk, 1999).

Definição 3.8 (Incidentes) Dado um PSLG X , dois segmentos ou um vértice e um segmento de X são ditos incidentes se eles se intersectam em suas extremidades.

Definição 3.9 (Local feature size) Dado um PSLG X , o local feature size de um ponto p , $lfs(p)$, é o raio do menor disco centralizado em p que intersecta dois vértices ou segmentos não incidentes de X .

A Figura 3.8 (Ruppert, 1995) ilustra o *local feature size* de diferentes pontos, onde o raio do disco D_i é $lfs(p_i)$.

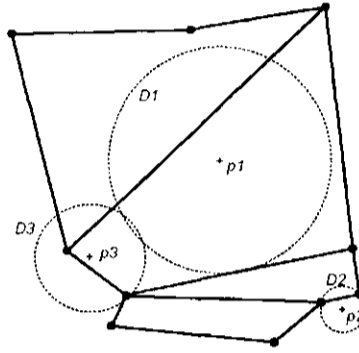


Figura 3.8: Local feature size (lfs) de diferentes pontos.

Lema 3.2 Para algum PSLG X , e quaisquer dois pontos u e v no plano,

$$lfs(v) \leq lfs(u) + |uv|.$$

Prova:

O disco com raio $lfs(u)$ centrado em u intersecta dois vértices ou segmentos não adjacentes de X . O disco com raio $lfs(u) + |uv|$ centrado em v contém o disco anterior, desta forma também intersecta dois vértices ou segmentos não adjacentes de X . Portanto, o menor disco centrado em v que intersecta dois vértices ou segmentos não adjacentes de X não tem raio maior que $lfs(u) + |uv|$. ■

Definição 3.10 (Vértice da malha) Um vértice da malha é qualquer vértice que foi inserido com sucesso na malha (incluindo os vértices de entrada).

Definição 3.11 (Vértice rejeitado) Um vértice rejeitado é um vértice que foi considerado para inserção mas rejeitado porque geraria um segmento invadido.

Definição 3.12 (Raio de inserção) Raio de inserção (r_v) é um valor atribuído a cada vértice da malha ou vértice rejeitado v , que equivale ao comprimento da menor aresta conectada a v imediatamente depois de v ser introduzido na triangulação.

Podemos considerar três casos de raio de inserção (r_v):

1. Se v é um vértice de entrada então r_v é a distância Euclidiana entre v e o vértice de entrada vizinho de v , mais próximo à v , (Figura 3.9 (a) (Shewchuk, 1999)).
2. Se v é um vértice inserido como o ponto médio de um segmento invadido, então r_v é a distância entre v e o vértice mais próximo à v que está dentro do círculo diametral, (Figura 3.9 (b) (Shewchuk, 1999)). Se o círculo diametral for vazio (os circuncentros que invadiram o segmento não foram inseridos), r_v é o raio do círculo diametral do segmento invadido, (Figura 3.9 (c) (Shewchuk, 1999)).
3. Se v é um vértice inserido como o circuncentro de um triângulo fino, então r_v é o circunraio do triângulo, (Figura 3.9(d) (Shewchuk, 1999)).

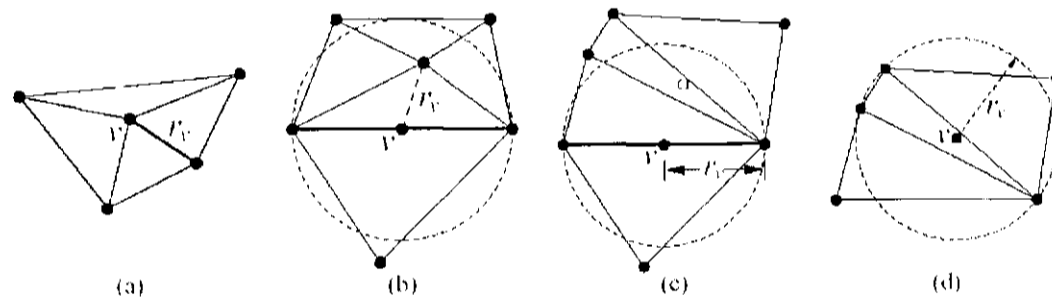


Figura 3.9: Diferentes casos de raio de inserção.

Definição 3.13 (Vértice pai) Vértice pai de um vértice v , denotado por $p(v)$, é intuitivamente o vértice responsável pela inserção de v . Casos:

- Se v é um vértice de entrada, ele não tem $p(v)$.
- Se v é um vértice inserido como o ponto médio de um segmento invadido, então $p(v)$ é o vértice que está dentro do círculo diametral, se houver mais de um, escolher o mais perto; $p(v)$ pode ser um vértice rejeitado e não precisa necessariamente pertencer à malha.

- Se v é um vértice inserido (ou rejeitado) como o circuncentro de um triângulo fino t , então $p(v)$ é o extremo da menor aresta de t que foi inserido mais recentemente. Caso os dois vértices da menor aresta sejam vértices de entrada, escolher um arbitrariamente.

Sendo assim temos que cada vértice de entrada é a raiz de uma árvore de vértices.

Lema 3.3 *Seja v um vértice, se $p(v)$ é o seu pai, então $r_v \geq \text{lfs}(v)$ ou $r_v \geq Cr_{p(v)}$, onde:*

- $C = B$ se v é um circuncentro de um triângulo fino.
- $C = \frac{1}{\sqrt{2}}$ se v é o ponto médio de um segmento invadido e $p(v)$ é o circuncentro de um triângulo fino.
- $C = \frac{1}{2 \cos \alpha}$ se v e $p(v)$ estão num segmento incidente separado por um ângulo α (com $p(v)$ dentro do círculo diametral do segmento cujo ponto médio é v), com $45^\circ \leq \alpha \leq 90^\circ$.
- $C = \sin \alpha$ se v e $p(v)$ estão num segmento incidente separado por um ângulo $\alpha \leq 45^\circ$.

Pelo Lema 3.3, considerando a relação entre o raio de inserção do vértice e o raio de inserção do seu pai, concluímos que nenhum descendente de um vértice da malha tem um raio de inserção menor que o raio de inserção de seu pai.

Teorema 3.1 *Seja lfs_{\min} a menor distância entre duas entidades (segmentos não adjacentes ou vértices não incidentes) de uma entrada PSLG. Suponhamos que quaisquer dois segmentos adjacentes estão separados por um ângulo de no mínimo 60° , e um triângulo é considerado fino se o seu circunraio dividido pelo comprimento da sua menor aresta for maior que B , onde $B \geq \sqrt{2}$. O algoritmo de Ruppert terminará com todas as arestas da triangulação maiores que lfs_{\min} .*

Como já foi visto anteriormente, a inserção do circuncentro é sempre feita no interior da triangulação (Lema 3.1). Pelo Lema 3.3 e Teorema 3.1 é provado que a cada “descendência” de vértices, o valor do raio de inserção não diminui. Desta forma, arestas menores que as já existentes não são introduzidas, e isso garante que o refinamento de Delaunay termina. Quando isto acontece todos os triângulos da malha possuem ângulos maiores que 20.7° .

É possível ainda provar que cada aresta de saída da malha tem comprimento proporcional ao *local feature size* (*lfs*) de seus vértices, ou seja, provar que o comprimento da aresta está intimamente relacionado com os elementos a sua volta; a fim de satisfazer usuários interessados em malhas que possuem triângulos com densidades diferentes, de acordo com as regiões, e não em malhas uniformes. Além disso, esse tipo de malha ajuda a reduzir o número de triângulos.

3.3 Ângulos Pequenos

A presença de um ângulo pequeno na entrada do algoritmo, neste caso num PSLG, requer um procedimento especial a fim de que na malha final não apareçam ângulos pequenos. Sabendo que ângulos pequenos de entrada não podem ser removidos, o principal propósito é triangular um PSLG sem criar nenhum ângulo pequeno que não esteja presente na entrada. Nenhum algoritmo fornece esta garantia para todos PSLGs. Este procedimento ainda é um fato obscuro na literatura. Descrevemos nesta seção a forma com que Ruppert, Shewchuk e Miller abordam este fato e também o tratamento de ângulos pequenos adotado em nossa implementação.

3.3.1 Abordagem de Ruppert

Para triangular algum PSLG sem introdução adicional de ângulos pequenos, Ruppert sugere que um vértice no canto de um ângulo pequeno pode ser coberto com uma faixa fina de triângulos bem formados (Definição 3.1). Esta técnica utiliza *shield edges* (Ruppert, 1995) da seguinte maneira:

Primeiro realiza-se o cálculo do *local feature size* para todo vértice de entrada v .

Todo vértice v com ângulo pequeno é cercado com um círculo que intersecta as arestas de entrada, como mostra a Figura 3.10. O raio do círculo é $\frac{lf_s(v)}{3}$, assim os círculos ao redor de vértices diferentes não se intersectam e nem estão muito perto.

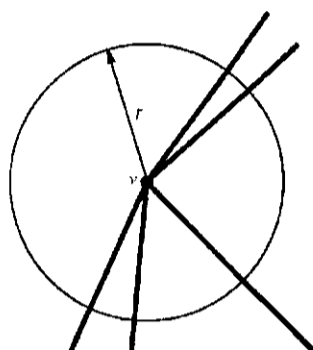


Figura 3.10: Círculo ao redor do vértice de entrada.

Sendo α o menor ângulo de entrada do PSLG, os ângulos em v maiores que 2α são divididos para que fiquem entre α e 2α , introduzindo *shield edges* ao redor do círculo, e *spoke edges* de v para o círculo, Figura 3.11. Estas arestas e os novos vértices, se tornam parte do PSLG de entrada, aparecendo na malha de saída e reduzindo o *local feature size* no PSLG.

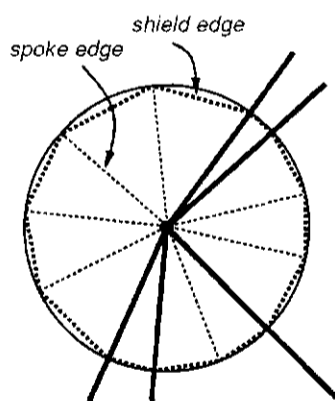


Figura 3.11: Ilustração das *shield edges* e *spoke edges*.

Cada *shield edge* é dividida, havendo para isto duas opções:

1. Coloca-se arestas entre os vértices que dividiram a *shield edge* e v , como mostra a Figura 3.12.

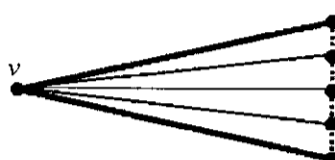


Figura 3.12: Primeiro método de divisão da *shield edge*.

2. Já nesta opção, a idéia é que a *shield edge* seja dividida um número constante de vezes, e pode-se usar o número constante de camadas para preencher com triângulos, Figura 3.13. Esta construção é mais complicada, mas pode evitar que o menor de todos os ângulos de entrada seja dividido.

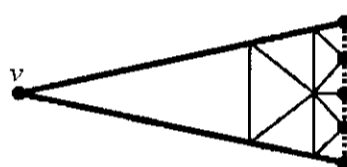


Figura 3.13: Segundo método de divisão da *shield edge*.

Assim, a região dentro das *shield edges* é triangulada por uma sequência finita de tiras similares, com cada tira sucessivamente menor que a tira anterior por um fator constante perto de um. Essas tiras são finas e compostas por triângulos bem formados, Figura 3.14 (Shewchuk, 1999).

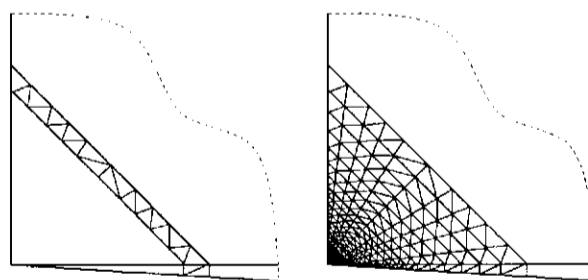


Figura 3.14: Triangulação da região dentro de *shield*, com uma sequência finita de tiras similares sucessivamente menor de triângulos bem formados.

Segundo Shewchuk (Shewchuk, 1999), Ruppert afirma de forma errada que a região escondida atrás de *shield edges* sempre tem uma triangulação finita de boa qualidade. A tira está limitada o suficiente para admitir uma triangulação de qualidade na entrada de ângulos pequenos.

3.3.2 Abordagem de Shewchuk

A alternativa sugerida por Shewchuk (Shewchuk, 1999), está baseada no Quitter, um algoritmo de refinamento de Delaunay que fornece uma flexibilidade e que possui término garantido.

O Quitter está baseado no refinamento de Delaunay com círculos concêntricos. Quando um segmento s é invadido pelo circuncentro de um triângulo fino, toma-se uma decisão entre dividir s com um vértice v , ou deixar s inteiro.

A decisão não é trivial e depende de alguns princípios:

- Se nenhum vértice de s possuir um ângulo de entrada menor que 60° (Figura 3.15 (a)), ou se ambos possuírem (Figura 3.15 (b)), então s é dividido.
- Caso contrário, seja a o vértice do ângulo pequeno. Definimos o grupo de segmentos incidentes à a que estão separados de s , ou de algum outro membro do grupo por um ângulo menor que 60° , como sendo o grupo de segmento de s , Figura 3.16 (a).

Note que esta definição do grupo de segmento não implica que todos segmentos incidentes num vértice de entrada são sempre parte do mesmo grupo. Pode ocorrer de ter dois grupos de segmentos incidentes que compartilham um mesmo vértice, separado um do outro por ângulos maiores que 60° , como na Figura 3.16 (b).

Neste caso, o segmento s é dividido pela inserção do vértice v (seu ponto médio), somente se uma ou mais das três circunstâncias acontecer:

1. Dado o grupo G de segmento de s , $G = s, s_1, \dots, s_n$, cada segmento do grupo que tem comprimento maior ou igual a $|s|$ é dividido com a inserção do ponto

médio. Seja r_{min} o menor raio de inserção dos pontos médios e r_g o raio de inserção do avô g de v . Se $r_{min} \geq r_g$, então v é inserido.

Se todos os segmentos do grupo têm o mesmo comprimento, então r_{min} depende do menor ângulo.

2. Se um dos segmentos do grupo de segmentos de s tem um comprimento que não seja potência de dois, então v é inserido.
3. Se nenhum antepassado de v se encontrar no mesmo segmento, então v é inserido. Os vértices do segmento não contam.

Esta circunstância tenta distinguir se um segmento é invadido por causa de atributos pequenos de entrada, ou porque produz um ângulo pequeno.

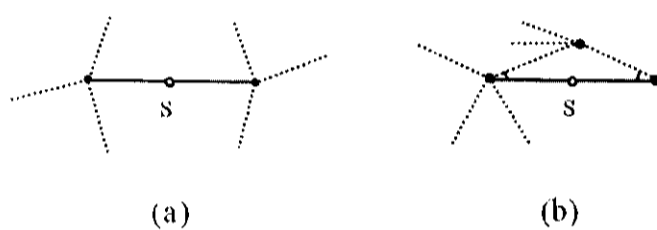


Figura 3.15: Divisão do segmento s : (a) Nenhum vértice de s possui um ângulo de entrada menor que 60° ; (b) Ambos os vértices de s possuem ângulo de entrada menor que 60° .

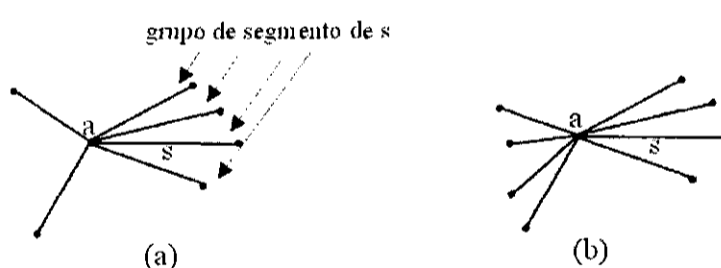


Figura 3.16: Grupo de segmento: (a) Grupo de segmento de s ; (b) Dois grupos de segmentos distintos que compartilham o mesmo vértice.

Se não houver nenhum ângulo de entrada menor que 60° , o algoritmo Quitter trabalha exatamente como o algoritmo de Ruppert.

Uma desvantagem do Quitter é a exigência de mais memória, porque cada vértice da malha deve armazenar seu raio de inserção e um ponteiro para seu pai, ou para seu avô, caso seu pai tenha sido rejeitado. Mas o autor sugere algumas modificações, para não haver necessidade de mais memória. Os detalhes destas modificações podem ser encontrados em (Shewchuk, 1999), assim como o esboço da prova do teorema que garante a convergência do algoritmo de Quitter.

Segundo Shewchuk, os triângulos finos na malha final ocorrem somente perto dos ângulos de entrada menores que 60° , na prática, somente perto dos ângulos de entrada muito menores que 60° .

Miller afirma que a análise do esquema de Shewchuk vem sem garantias de classificação e assim, sem nenhuma afirmação de otimização (Miller and Walkington, 2003).

3.3.3 Abordagem de Miller

Em (Miller and Walkington, 2003), encontramos uma alteração do algoritmo de Ruppert que gera uma malha de Delaunay com todos os ângulos maiores que 26.45° , exceto os ângulos dos triângulos cujo menor aresta está “oposta” a um ângulo de entrada $\theta < 36.53^\circ$; neste caso, o ângulo de saída será maior do que $\arctan\left(\frac{\sin \theta}{2 - \cos \theta}\right)$, onde θ é o ângulo mínimo de entrada. Todos os ângulos da malha gerada pelo algoritmo proposto por Miller são menores que 137.1° .

A entrada do algoritmo é um *Planar Straight Line Graph* (PSLG) arbitrário.

Dado um conjunto de pontos de entrada P e um conjunto de segmentos de entrada S , o algoritmo mantém um conjunto de pontos executáveis e um conjunto de segmentos atuais, que são inicializados com o conjunto P e S , respectivamente.

Quando um segmento pertencente ao conjunto de segmentos atuais for dividido com a inserção do seu ponto médio, durante a execução do algoritmo, este segmento é removido do conjunto de segmentos atuais e as duas novas metades do segmento são adicionadas neste conjunto. Um outro tipo de ponto, além do ponto médio do segmento, que pode ser acrescentado a triangulação é o circuncentro do triângulo.

O algoritmo de refinamento de Delaunay adaptativo proposto por Miller tem duas operações principais, que são as seguintes:

1. Se s é um segmento atual e existe um ponto executável que invade s , ou seja, s é um segmento invadido por um ponto executável, então divide s .
2. Se a , b e c são pontos executáveis, o circuncírculo do triângulo $\triangle abc$ não contém nenhum ponto executável, $\angle acb < k$, sendo k o parâmetro do ângulo de saída, dado pelo ângulo de saída mínimo; o circuncentro p do triângulo está dentro da triangulação, e um dos casos acontece:
 - (a) Tanto a como b são ponto médio em segmentos de entrada não disjuntos distintos, compartilhando o vértice x de entrada, e $\angle axb > \frac{\pi}{3}$, como na Figura 3.17.
 - (b) a e b não são ponto médio em segmentos de entrada.

Então o algoritmo tenta adicionar o ponto p ; contudo, se o ponto p invadir qualquer segmento atual, o algoritmo não insere o ponto p e divide os segmentos atuais que são invadidos por p .

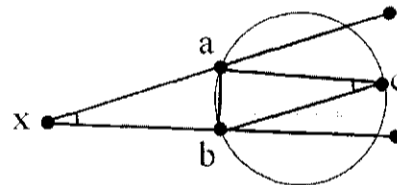


Figura 3.17: O circuncírculo do $\triangle abc$ não contém nenhum ponto executável, $\angle acb < k$ e a e b são pontos médios em segmentos de entrada não disjuntos distintos, compartilhando o vértice x de entrada, e $\angle axb > \frac{\pi}{3}$.

Sendo assim, o algoritmo removerá os ângulos menores que um certo k , exceto quando se tem a menor aresta de um triângulo oposta a um ângulo de entrada muito pequeno, neste caso, os ângulos de saída pequenos são ignorados.

A saída do algoritmo será o conjunto de pontos executáveis, o conjunto de segmentos atuais e a triangulação de Delaunay deste conjunto de pontos executáveis.

Este algoritmo proposto por Miller só é necessário quando o ângulo mínimo de entrada for menor que 36.53° . O mesmo removerá os ângulos pequenos onde for possível, isto é, afastado dos ângulos pequenos da entrada. No entanto, para ângulos de entrada maiores, o algoritmo original de Ruppert com círculos concêntricos gera a mesma saída com otimização garantida; é isso que nos garante a Afirmação 3.1.

Afirmação 3.1 *Suponhamos que se tenha a garantia de que se o algoritmo de refinamento de Delaunay adaptativo proposto por Miller é executado em algum PSLG, com ângulo mínimo de entrada θ , com um parâmetro do ângulo de saída k , e que:*

- (a) *o algoritmo termina,*
- (b) *nenhum ângulo de saída da malha é menor que k , e*
- (c) *nenhum ângulo é maior que $\pi - 2w$, onde $0 < k \leq w \leq \frac{\pi+k}{4}$.*

Então, se o algoritmo de refinamento de Delaunay (original de Ruppert) é executado em algum PSLG, com ângulo mínimo de entrada θ , usando um parâmetro do ângulo de saída denomindo h , com $h = k$, então:

- (a) *o algoritmo termina,*
- (b) *nenhum ângulo de saída da malha é menor que h , e*
- (c) *nenhum ângulo é maior que $\pi - 2w$, onde $0 < h \leq w \leq \frac{\pi+h}{4}$.*

A prova pode ser encontrada em (Miller and Walkington, 2003). Através desta afirmação, notamos que o algoritmo de refinamento de Delaunay de Ruppert é tão bom quanto sua versão adaptativa proposta por Miller. Sendo necessário o algoritmo de refinamento de Delaunay adaptativo proposto por Miller apenas quando o ângulo mínimo de entrada for menor que 36.53° .

Os lemas, teoremas e corolários, que provam a teoria do algoritmo de refinamento de Delaunay adaptativo proposto por Miller, podem ser encontrados em (Miller and Walkington, 2003).

3.3.4 Abordagem Adotada

Depois do levantamento bibliográfico realizado e do conhecimento obtido sobre as vantagens e desvantagens das abordagens dos principais autores neste assunto, decidimos adotar uma maneira própria para tratamento dos ângulos pequenos de entrada, que pode ser descrito nos seguintes passos:

1. Dado um PSLG de entrada, medir os ângulos internos.
2. Ao encontrar um ângulo pequeno, num determinado vértice v , entre duas arestas e_1 e e_2 , verificar se existe mais aresta incidente neste vértice, além de e_1 e e_2 .
 - Se existir, como na Figura 3.18 (b), inserir estas outras arestas no conjunto de arestas incidentes a v , que chamaremos de E .
 - Senão, como na Figura 3.18 (a), E conterá apenas e_1 e e_2 , ou seja, $E = \{e_1, e_2\}$.
3. Calcular o tamanho de cada aresta de E .
4. Seja c_{min} o comprimento da menor aresta, dividir cada aresta de E , com um vértice a uma distância $\frac{c_{min}}{2}$ de v , como ilustra a Figura 3.19.
5. A triangulação e o refinamento acontece normalmente.

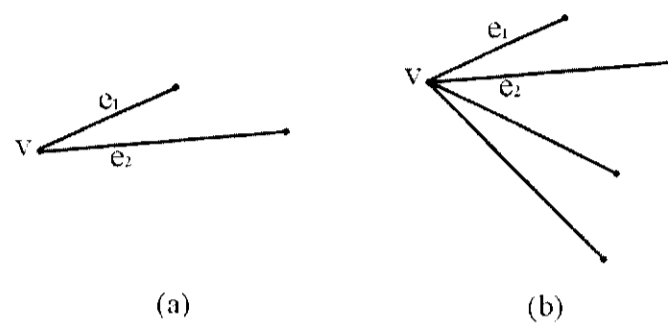


Figura 3.18: (a) Ângulo pequeno em v entre e_1 e e_2 ; (b) Existem mais arestas incidentes a v , além de e_1 e e_2 .

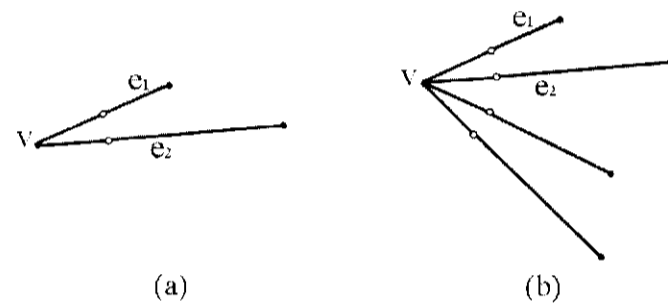


Figura 3.19: Inserção dos novos vértices de acordo com o comprimento de e_1 (c_{min}).

Por que funciona?

O Algoritmo de Refinamento de Ruppert exige ângulos de entrada maiores que 60° , para que seu funcionamento ocorra normalmente e sem risco de “não parar”.

Ao tratar dos ângulos pequenos como descrito acima, a inserção dos novos vértices praticamente garante a geração dos triângulos nestas pontas, como mostra a Figura 3.20. Estes triângulos serão ignorados nos próximos passos, e os novos ângulos do PSLG, exceto nestes triângulos, serão maiores que 60° , o que faz com que o Algoritmo de Refinamento trabalhe normalmente.

Prova matemática:

Esta prova está baseada em propriedades geométricas, como ilustra a Figura 3.21.

Dado três vértices a, b e c , onde $\angle abc = \alpha$ e $0 < \alpha < \frac{\pi}{3}$.

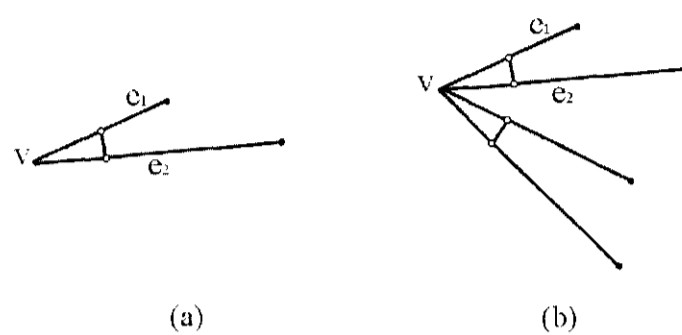


Figura 3.20: Inserção dos triângulos nas pontas garantem novos ângulos, externos aos triângulos, maiores que 60° .

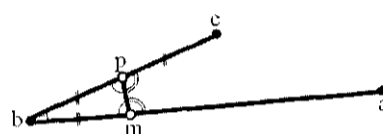


Figura 3.21: Ilustração da prova matemática do tratamento de ângulos pequenos.

Seja $dist\ bc = c_{min}$, $dist\ ba \geq c_{min}$, p o ponto médio da aresta bc e m um ponto inserido na aresta ba , onde $dist\ bm = \frac{c_{min}}{2}$.

O triângulo Δbpm é isósceles, assim:

$$\begin{aligned} \angle bmp = \angle bpm = \theta, e \\ \alpha + \theta + \theta = \pi \Rightarrow \alpha = \pi - 2\theta. \end{aligned}$$

Temos:

$$\begin{aligned} 0 < \alpha < \frac{\pi}{3} \\ 0 < \pi - 2\theta < \frac{\pi}{3} \\ -\pi < -2\theta < \frac{\pi}{3} - \pi \\ -\pi < -2\theta < -\frac{2\pi}{3} \\ \frac{2\pi}{3} < 2\theta < \pi \\ \frac{\pi}{3} < \theta < \frac{\pi}{2} \quad (I) \end{aligned}$$

Seja $\angle cpm = \beta$, temos:

$$\beta + \theta = \pi \Rightarrow \beta = \pi - \theta \text{ ou } \theta = \pi - \beta \quad (II)$$

De (I) e (II) temos:

$$\begin{aligned} \frac{\pi}{3} &< \pi - \beta < \frac{\pi}{2} \\ -\pi + \frac{\pi}{3} &< -\beta < \frac{\pi}{2} - \pi \\ \frac{-2\pi}{3} &< -\beta < \frac{-\pi}{2} \\ \frac{\pi}{2} &< \beta < \frac{2\pi}{3} \end{aligned}$$

Os novos ângulos serão maiores que 90° , o que garante o bom funcionamento do Algoritmo de Refinamento de Ruppert.

Note que esta prova também é válida, para o caso ilustrado na Figura 3.20 (b), uma vez que o mesmo ocorrerá para todo triângulo novo, gerando novos ângulos maiores que 90° .

Existe caso onde a inserção dos vértices a uma distância $\frac{c_{\max}}{2}$ de v não garante a geração do triângulo na ponta como desejado, Figura 3.22. Deve-se então antes de inserir os novos vértices, verificar se a formação do triângulo é possível, por se tratar de uma Triangulação de Delaunay, basta verificar se existe algum vértice no interior do circuncirculo do triângulo desejado. Caso exista, adota-se uma nova distancia, $\frac{c_{\min}}{4}$. Este procedimento é repetido até ser possível a formação do triângulo.

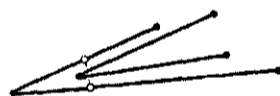


Figura 3.22: Exemplo onde a inserção dos vértices não garante a formação do triângulo na ponta.

Quando existe duas arestas compridas com um ângulo pequeno de entrada entre elas, essa abordagem pode gerar um triângulo grande, desproporcional ao restante da malha. Uma solução é repetir a idéia de divisão dentro deste triângulo. Sendo do mesmo tamanho as duas arestas que compartilham o vértice com ângulo pequeno, o ponto médio destas arestas será inserido, gerando triângulos menores dentro do triângulo grande.

Estrutura de Dados Topológica

O conceito de estrutura de dados topológica foi introduzida por Baumgart (Baumgart, 1975), no contexto da Visão Computacional, para a representação da topologia de superfícies de diversos modelos poliedrais, topologicamente equivalentes à esfera. A partir daí tem sido largamente utilizada nas áreas de Modelagem Geométrica e Geometria Computacional.

Estruturas de dados topológica determinam uma classe de estrutura de dados particularmente importante para representação de malhas. Elas visam a representação da subdivisão do espaço enquanto mantém a informação topológica (Nonato et al., 2002). Permitem extrair do modelo representado todas as relações de adjacência entre as diferentes entidades presentes no modelo (Chiyokura, 1988; Mäntyla, 1988).

Quando se utiliza uma estrutura de dados topológica para geração ou manipulação de malhas, as principais características exigidas na estrutura são a capacidade de armazenar a malha geométrica permitindo uma manipulação dinâmica, inserção e remoção das entidades que formam esta malha, e otimização nas buscas por informações topológicas, de

vizinhanças e singularidade de vértices.

Existem várias estruturas de dados topológicas empregadas na geração e representação de malhas 2D, dentre elas *winged-edge* de Baumgart (Baumgart, 1975), *quad-edge* de Guibas e Stolfi (Guibas and Stolfi, 1985) e *half-edge* de Mäntyla (Mäntyla, 1988).

Para manipular a estrutura de dados *quad-edge* Guibas e Stolfi (Guibas and Stolfi, 1985) apresentaram um operador topológico único, o *Splice*. Baumgart (Baumgart, 1975) e Mäntyla (Mäntyla, 1988) utilizam um conjunto de operadores, chamados de operadores de Euler, para modificar as entidades topológicas contidas nas estruturas de dados topológicas *winged-edge* e *half-edge*, respectivamente (Nonato et al., 2002).

Na Seção 4.1 apresentamos a estrutura de dados *Singular Handle-Edge*, podendo também ser referenciada como SHE (Nonato et al., 2002). Esta estrutura de dados é totalmente orientada a objetos, foi baseada nas idéias de Nonato, Castelo e Oliveira (Nonato et al., 2002) e desenvolvida pelo nosso grupo.

Nonato, Castelo e Oliveira (Nonato et al., 2002) apresentam o molde matemático para inserção e remoção em malhas bidimensionais que permite um controle consistente da topologia da malha durante sua construção. O controle topológico durante a inserção e remoção de elementos é feita através da incorporação dos conceitos da Teoria de Morse em um conjunto de operadores para manipulação da malha (Operadores de Morse), além disso, utiliza a estrutura de dados SHE para aplicação prática destes operadores. Na Seção 4.2 fazemos uma breve descrição dos Operadores de Morse.

4.1 Estrutura de Dados SHE

A estrutura de dados topológica Singular Handle-Edge (SHE) é utilizada para manipulação de malhas triangulares bidimensionais não estruturadas, permitindo uma maior otimização na geração e utilização desse tipo de malha. Antes de detalharmos as entidades de representação da SHE, descrevemos algumas definições.

4.1.1 Definições Básicas

Definição 4.1 (Aresta interior) Uma aresta e (simplexo de dimensão 1 Definição 2.4) de uma triangulação T é uma aresta interior de T se e é comum a dois triângulos de T .

Definição 4.2 (Aresta de bordo) Uma aresta e de uma triangulação T é uma aresta de bordo de T se e pertence a um único triângulo de T .

Definição 4.3 (Curva de bordo) Seja T uma triangulação e $E = (e_1, \dots, e_n)$ uma seqüência de arestas de bordo distintas em T tal que e_i e e_{i+1} ($e_{n+1} = e_1$), $i = 1, \dots, n$, têm um vértice em comum. E é curva de bordo de T se, quando se caminha da aresta e_i para e_{i+1} , os triângulos que contém e_i e e_{i+1} estão sempre localizados à esquerda (ou, alternativamente, a direita) de e_i e e_{i+1} para todo $i = 1, \dots, n$.

Definição 4.4 (Curva de bordo externa) Uma curva de bordo $E = (e_1, \dots, e_n)$ é uma curva de bordo externa se para cada e_i em E , existe um número real $\varepsilon > 0$ tal que um disco centralizado em qualquer ponto interior de e_i com raio ε não intersecta nenhum buraco de T .

Definição 4.5 (Curva de bordo interna) Uma curva de bordo $E = (e_1, \dots, e_n)$ é uma curva de bordo interna se ela limita um buraco de T .

Definição 4.6 (Vértice singular) Um vértice é singular se seu link (Definição 2.10) não é homeomorfo a um círculo ou a um segmento.

Definição 4.7 (Característica de Euler) Seja T uma triangulação e nv , ne , nt , nc e nh são, respectivamente, o número de vértices, arestas, triângulos, componentes conexas e buracos em T , a característica de Euler é dada pela seguinte fórmula:

$$Euler(T) = nv(T) - ne(T) + nt(T) = nc(T) - nh(T).$$

Um aspecto importante da SHE é a sua capacidade de manipular vértices singulares e de manter uma representação explícita das curvas de bordo de uma triangulação, tais representações explícitas são essenciais para uma implementação de Operadores de Morse e simplifica a armazenagem de condições de contorno que tipicamente aparecem em simulações numéricas (Nonato et al., 2002).

4.1.2 Entidades de Representação da SHE

A SHE é organizada em sete entidades de representação, descritas a seguir:

- **sheVertex:** Esta classe representa cada vértice de uma malha triangular. Esta representação é feita armazenando as coordenadas geométricas, a característica de pertencer ao bordo ou não, informações sobre singularidade e um identificador. Um ponto de destaque nesta estrutura é a informação de singularidade em vértice, que permite percorrer todo o link do vértice, mesmo este sendo um vértice singular.
- **sheBoundary:** Esta classe representa cada aresta de bordo de uma malha triangular. Outra informação armazenada aqui é uma referência à componente de bordo que ela pertence.
- **sheBoundaryCp:** Esta classe representa a(s) componente(s) de bordo de uma malha, cada componente conexa da malha possui sua lista de componentes de bordo (sheBoundaryCp), que por sua vez possui um ponteiro para uma lista de sheBoundary, que são as arestas da referida borda.
- **sheSing:** Cada sheVertex possui um ponteiro para uma lista de sheSing sendo armazenado nesta lista as arestas de bordo onde o vértice singular é pé, ou seja, percorrendo um ciclo anti-horário ao redor do vértice, é a primeira aresta encontrada. Quando a lista de sheSing possui mais de um elemento, podemos dizer que o vértice é singular. Quando o vértice não é singular, se o mesmo pertencer ao bordo, ele terá na sua lista de sheSing a aresta de bordo, caso contrário, terá uma aresta arbitrária.

- **sheHalfEdge:** Esta classe representa cada semi-aresta dos triângulos da malha bidimensional. Nesta entidade é armazenada a informação da célula a que ela pertence, da semi-aresta vizinha e do vértice pé da semi-aresta. Arestas pertencentes a dois triângulos, são representadas por dois objetos, um para cada triângulo. As semi-arestas de bordo não têm semi-aresta vizinha.
- **sheCell:** Esta classe representa cada triângulo da malha. As informações guardadas por ela são as referências para as semi-arestas que pertencem à célula, uma referência para a malha que contém a célula, e um identificador.
- **sheShell:** Cada grupo de triângulos conexos é representado por esta classe. É nesta entidade onde são armazenadas as listas de células, e de componentes de bordo, além de um identificador, número de células, número de componentes de bordos e uma referência para a malha. Agrupando os triângulos desta maneira, por componentes conexas, temos que para cada Shell, existe apenas um componente de bordo externo, dentre todos bordos que este pode conter, sendo os outros componentes as bordas dos "buracos" na Shell.

Uma classe importante dentro desta estrutura é a classe SHE, que é o nível mais alto da representação e representa toda a malha, com todos os shells, bordos, triângulos, arestas e vértices. Nesta classe se encontram as operações principais, como inserção e remoção de triângulos.

A Figura 4.1 ilustra uma malha triangular com as entidades da estrutura de dados topológica SHE.

Através da SHE podemos obter informações topológicas como o número de componentes, células, bordos e vértices singulares, além de conseguirmos nos referenciar a estas entidades mais rapidamente.

Para uma melhor compreensão das relações entre as classes apresentamos um diagrama de classes na Figura 4.2.

Por conveniência de implementação os vértices estão conectados com a SHE, pois na construção da malha podemos ter vértices não associados a células, não pertencendo

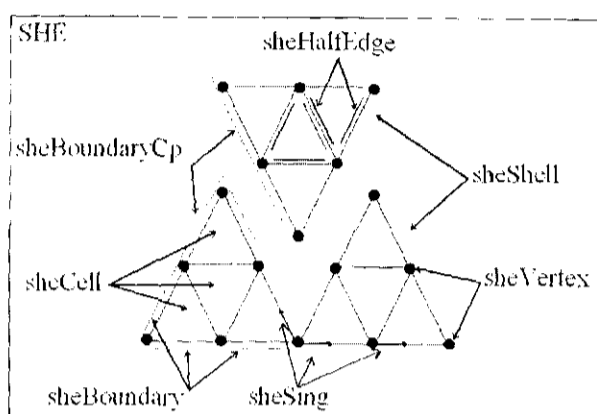


Figura 4.1: Malha triangular e as entidades da estrutura de dados topológica SHE.

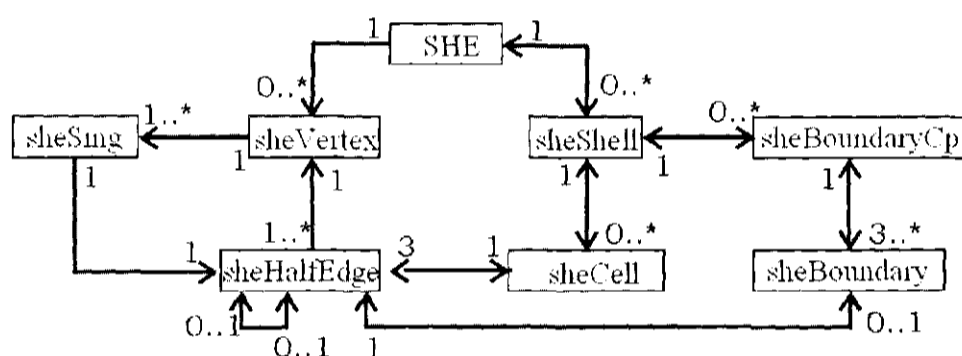


Figura 4.2: Diagrama de classes da estrutura de dados topológica SHE.

assim a nenhuma shell, mas isso se deve apenas para facilitar a implementação das listas. A implementação da estrutura de dados SHE utiliza a linguagem de programação C++.

4.2 Operadores de Morse

As operações mais utilizadas em uma estrutura de dados topológica que armazena uma malha triangular são as de inserção e remoção de triângulos, porém estas operações não mudam apenas a geometria de uma malha, mas a sua topologia, requerendo assim uma atenção especial, pois após tais operações, todas as informações sobre a topologia devem ser válidas, ou seja, deve-se manter a integridade da malha, e ainda, tais operações não podem ser lentas.

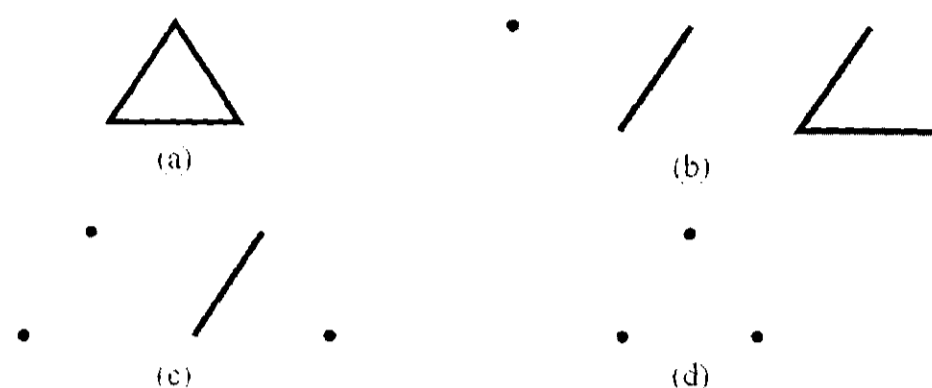


Figura 4.3: Diferentes casos de handle (a) (-1)-handle, (b) 0-handle, (c) 1-handle e (d) 2-handle

Para garantir a integridade topológica, a estrutura de dados SHE utiliza os operadores de Morse que serão descritos a seguir. A definição e lema a seguir são fundamentais para definirmos os operadores (Nonato et al., 2002).

Definição 4.8 *Seja T uma triangulação em \mathbb{R}^2 . Um triângulo t que deve ser inserido em T é definido como:*

1. *(-1)-handle de T se $t \notin T$ ou se todas arestas de t estão em T , Figura 4.3 (a).*
2. *0-handle de T se um vértice, ou uma aresta, ou duas arestas de t estão em T , Figura 4.3 (b).*
3. *1-handle de T se dois vértices, ou uma aresta e um vértice (oposto a aresta) de t estão em T , Figura 4.3 (c).*
4. *2-handle de T se três vértices de t estão em T , Figura 4.3 (d).*

Lema 4.1 *Seja t um k -handle de uma triangulação T . Então*

$$Euler(T \cup t) = Euler(T) - k.$$

Este lema mostra como a adição de um k -handle altera a característica de Euler de uma triangulação. A prova deste lema pode ser encontrada em (Nonato et al., 2002).

Os operadores de Morse são baseados no resultado do Lema 4.1 e estão classificados de acordo com a alteração topológica que provocam em uma malha.

Para a nomenclatura dos operadores são utilizadas as letras: M - *make* (criar) e K - *kill* (excluir), e as letras das entidades: T - triângulo, E - *edge* (aresta), C - componente conexa, S - vértice singular e H - *hole* (buraco).

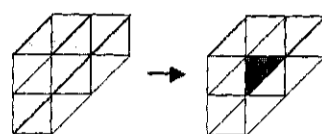
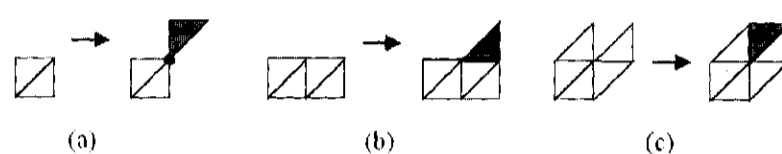
Para o caso bidimensional, temos os seguintes operadores para inserção de triângulos com as seguintes classificações:

- MO_{-1} : MEEETKH, MTC;
- MO_0 : MST, MET, MEET;
- MO_1 : MSSTH, MSETH, MSSTKC, MSETKC;
- MO_2 : MSSSTHH, MSSSTHKC, MSSSTKCC.

E também os operadores inversos, para remoção de triângulos com as seguintes classificações:

- MO_{-2} : KSSTH, KSETH, KSSTMC, KSETMC;
- MO_{-1} : KSSSTHH, KSSSTHMC, KSSSTMCC;
- MO_0 : KST, KET, KEET;
- MO_1 : KEEETMH, KTC.

Sendo os números subscritos, os valores obtidos da mudança da característica de Euler, após a aplicação do operador. Com a inserção de apenas um triângulo na malha por vez, temos os seguintes valores possíveis para a mudança na característica de Euler: -1, 0, 1 e 2 já com a remoção temos os possíveis valores: -1, -2, 0 e 1.

Figura 4.4: Operador MO_{-1} (MEEETKH).Figura 4.5: Operadores MO_0 (a)MST, (b)MET e (c)MEET.

As Figuras 4.4, 4.5, 4.6 e 4.7 ilustram resumidamente a ação de cada operador, exceto o operador MTC, que insere na malha um triângulo isoladamente. Por exemplo, o operador MSSTH na Figura 4.6 (a) cria dois vértices singulares, um triângulo e um buraco, já o operador MSSTKC na Figura 4.6 (c), cria dois vértices singulares, um triângulo e junta as duas componentes conexas, excluindo uma.

O uso dos operadores de Morse para a construção da malha nos assegura o controle total sobre a topologia da malha, ou seja, podemos obter facilmente o número de componentes conexas e de buracos na malha após cada inserção de triângulo. Desta forma, o usuário pode especificar a topologia desejada, estabelecendo um valor para o número de componentes conexas e de buracos da malha final. As componentes conexas e buracos não desejados são removidos usando os operadores inversos apropriados.

Existem duas importantes proposições sobre os operadores de Morse, que citamos a seguir, suas provas podem ser obtidas em (Nonato et al., 2002).

Proposição 4.1 *Qualquer triangulação pode ser gerada com quatro operadores de Morse.*

Proposição 4.2 *Não é possível gerar uma triangulação com buracos (pela inserção de triângulos) sem adicionar vértices singulares na triangulação durante o processo.*

A estrutura de dados topológica Singular Handle-Edge (SHE), que por sua vez utiliza os operadores de Morse para sua manipulação, foi utilizada na implementação deste

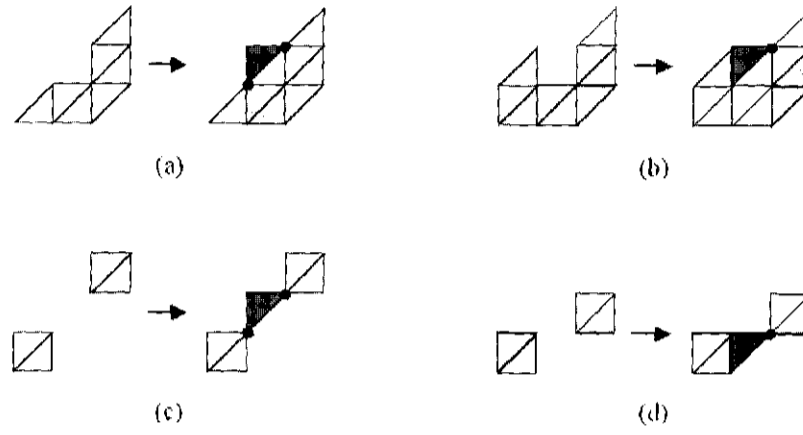


Figura 4.6: Operadores MO_1 (a)MSSTH, (b) MSETH, (c)MSSTKC e (d)MSETKC.

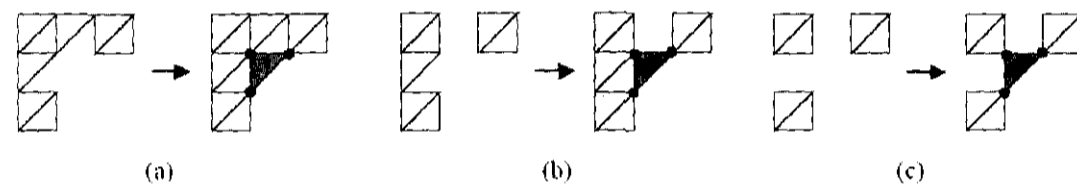


Figura 4.7: Operadores MO_2 (a)MSSSTHH, (b)MSSSTHKC, e (c)MSSSTKCC.

trabalho. Maiores detalhes e outras aplicações dos operadores de Morse podem ser encontrados em (Nonato et al., 2003).

O Sistema

Este projeto está vinculado a outros¹ que totalizam um conjunto completo de softwares para simulação de escoamentos em domínios complexos, ambiente Umflow-2D.

Estes softwares são constituídos de três módulos distintos e independentes: um módulo de modelagem (modelador), um módulo de simulação (simulador) e um módulo de visualização (visualizador). A comunicação entre os módulos é feita através de arquivos.

A implementação dos módulos foi feita em Linguagem C++ (Deitel and Deitel, 2002). A linguagem de programação C++ é um superconjunto da linguagem de programação C (seu primeiro nome foi “C com Classes”). A linguagem C recebeu este nome porque ela foi sucessora de uma linguagem chamada B, desenvolvida por Ken Thompson em 1970. B era um tanto quanto restrito e em 1972 Dennis Ritchie e Ken Thompson criaram a Linguagem C para aumentar o poder de B.

O nome C++ foi criado por Rick Mascitti em 1983, o nome representa um incremento

¹Projeto de mestrado: Simulação Numérica de Escoamentos de Fluidos Utilizando Diferenças Finitas Generalizadas, e projeto de iniciação científica: Interface Gráfica.

(operador ++ de incremento no C) na linguagem C. A linguagem não foi chamada de D porque é uma extensão da linguagem C.

A interface está sendo desenvolvida utilizando Qt Designer², um ambiente de programação distribuído gratuitamente com a biblioteca Qt, que além de ser um programa para criar interfaces, ele pode ser utilizado para gerenciar projetos completos em C++.

A biblioteca gráfica utilizada pelo modelador e pelo visualizador é OpenGL (Open Graphics Library), uma interface de software para dispositivos de hardware (Wright and Sweet, 1996). Esta biblioteca gráfica de modelagem e exibição tridimensional, é bastante rápida e portátil para vários sistemas operacionais. Seus recursos permitem ao usuário criar objetos gráficos, além do tratamento de imagens e texturas.

A biblioteca OpenGL foi introduzida em 1992 pela Silicon Graphics, no intuito de conceber uma API (Interface de Programação de Aplicação) gráfica independente de dispositivos de exibição. Com isto, foi estabelecida uma ponte entre o processo de modelagem geométrica de objetos, situadas em um nível de abstração mais elevado, e as rotinas de exibição e de processamento de imagens implementadas em dispositivos (hardware) e sistemas operacionais específicos. Devido às funcionalidades providas pelo OpenGL, tal biblioteca tornou-se um padrão amplamente adotado na indústria de desenvolvimento de aplicações gráficas. Todas as rotinas do OpenGL são implementadas em C, tornando fácil sua utilização em qualquer programa escrito em C ou C++.

As especificações do OpenGL não descrevem as interações entre OpenGL e o sistema de janelas utilizado. Assim, tarefas comuns como criar janelas gráficas, gerenciar eventos provenientes de mouse e teclado, e apresentação de menus ficam a cargo de bibliotecas próprias de cada sistema operacional. Neste contexto, merece destaque a OpenGL Utility Toolkit (GLUT), uma biblioteca que permite criar janelas, menus, manipular eventos, entre outras coisas, independente do sistema operacional utilizado. Aplicações escritas em OpenGL/GLUT podem ser compiladas em diversas plataformas, sem necessidade de alterações em seus códigos.

O sistema implementado neste projeto é constituído pelos módulos de modelagem

²O website oficial do Qt Designer está disponível em: <<http://www.trolltech.com/products/qt/>>

(modelador) e visualização (visualizador).

O simulador requer a definição de um objeto, chamado também por domínio, que neste caso é uma malha não estruturada e que será o molde dentro do qual o fluido será introduzido. Requer também a definição de parâmetros que configuram o tipo do escoamento a ser modelado, entre estes podemos citar: a viscosidade, a tolerância para a solução da equação de Poisson, e muitos outros. Estes dados são introduzidos na interface do modelador.

Após a simulação numérica de um escoamento tem-se a visualização dos resultados por meio do visualizador.

Com a definição dos dados de entrada no simulador, e com as facilidades para visualização dos resultados, o sistema gráfico e interativo, desenvolvido neste projeto, dá suporte ao simulador³.

Neste capítulo descrevemos a metodologia do sistema dividido em duas partes: Seção 5.1 Modelador e Seção 5.2 Visualizador.

5.1 Modelador

A entrada dos dados do objeto no modelador é feita através de um arquivo no formato `vtk`⁴ e está em desenvolvimento a parte interativa da interface⁵, onde o usuário poderá criar o objeto apenas com clicks do mouse. No arquivo `vtk` contém a definição de um `PSLG Planar Straight Line Graph` e o arquivo pode ser dividido nas seguintes partes:

1. Cabeçalho:

```
vtk DataFile Version 1.0

PSLG Planar Straight Line Graph

ASCII
```

³Desenvolvido no projeto *Simulação Numérica de Escoamentos de Fluidos Utilizando Diferenças Finitas Generalizada*.

⁴<http://www.vtk.org/pdf/file-formats.pdf>

⁵Projeto de iniciação científica: Interface Gráfica

```
DATASET UNSTRUCTURED_GRID
```

2. Pontos: número de pontos, tipo das coordenadas dos pontos e coordenadas dos pontos em cada linha.

```
POINTS 52 float
```

```
0.00000000 0.00000000 0.00000000
```

```
20.00000000 0.00000000 0.00000000
```

```
...
```

```
16.00000000 1.00000000 0.00000000
```

3. Células: número de células, código verificador e em cada linha a quantidade de vértice do polígono e os identificadores destes vértices.

```
CELLS 5 57
```

```
4 0 1 2 3
```

```
...
```

```
12 40 41 42 43 44 45 46 47 48 49 50 51
```

4. Tipo das curvas poligonais: Número de células e o tipo de cada célula. O tipo 7 de células no formato vtk corresponde a polígonos, desta forma, as fronteiras do PSLG são escritas como polígonos

```
CELL_TYPES 5
```

```
7 7 7 7 7
```

As fronteiras do PSLG são escritas obedecendo uma orientação. A fronteira externa está orientada no sentido anti-horário e a interna no sentido horário. Se houver um outro polígono, dentro da fronteira interna, este receberá uma orientação diferente da fronteira interna anterior, neste caso, o sentido anti-horário, e assim sucessivamente, como na Figura 5.1.

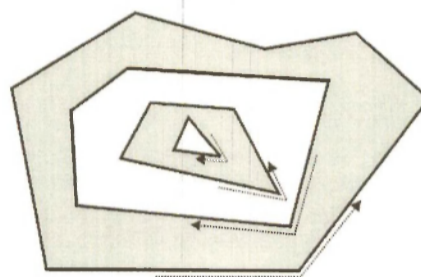


Figura 5.1: Orientação das fronteiras de um PSLG.

O usuário escolhe, através da interface, um arquivo neste formato para ser carregado (Figura 5.2) e qual algoritmo de refinamento será usado no PSLG, carregado pelo arquivo escolhido.

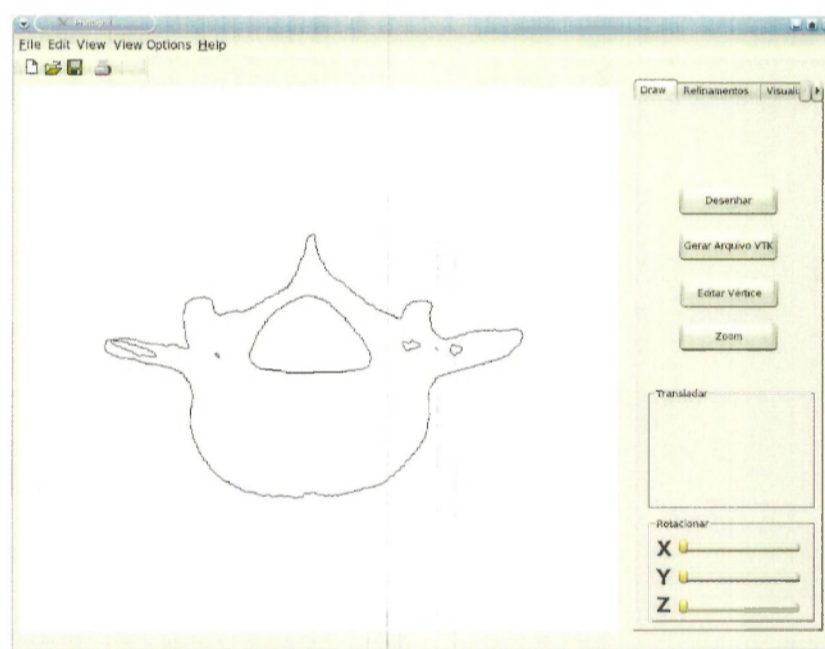


Figura 5.2: Interface com o arquivo .vtk carregado.

Tem-se como opções o Algoritmo de Refinamento de Chew e o Algoritmo de Refinamento de Ruppert, que gera uma malha não estruturada com a Triangulação de Delaunay. Estes algoritmos foram descritos com detalhes no Capítulo 3 e a Figura 5.3 ilustra a malha gerada por eles. A malha gerada pelo Algoritmo de Refinamento de Chew na Figura 5.3 é muito refinada e os triângulos são tão pequenos que para visualizá-los é necessário rea-

lizar um *zoom* na malha, apresentado posteriormente como um recurso do visualizador.

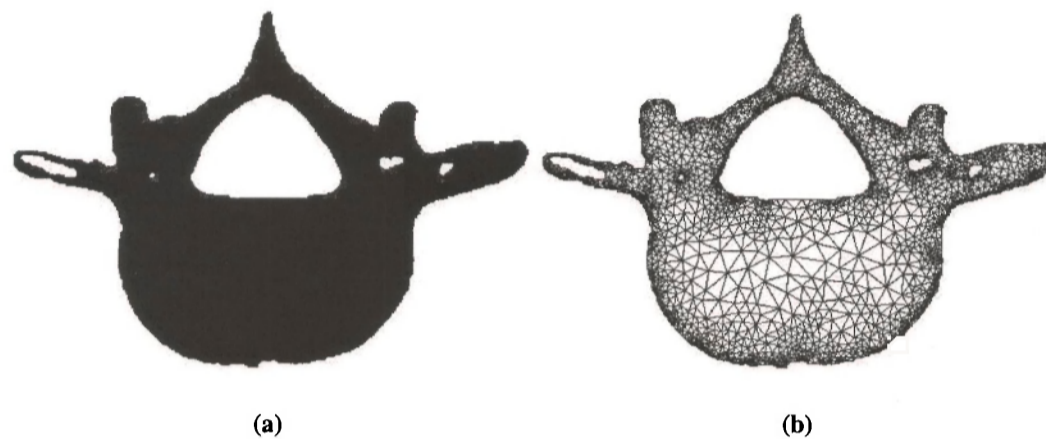


Figura 5.3: (a) Algoritmo de Refinamento de Chew, (b) Algoritmo de Refinamento de Ruppert.

A triangulação de Delaunay utilizada retorna o fecho convexo do conjunto de pontos. Através da orientação dos polígonos é possível descobrir se uma célula é interna ou externa. Antes de refinar a malha triangulada rotulamos cada célula como interna ou externa. O refinamento é aplicado nas células internas, rotulando todas as novas células adicionadas pelo algoritmo. No final o programa exclui as células externas.

Os dados para a simulação são digitados pelo usuário na interface do modelador, Figura 5.4.

Após a entrada do objeto e de todos os dados necessários para a simulação, o modelador cria três arquivos de saída, que serão os arquivos de entrada para o simulador. Os dados são divididos em três arquivos:

1. Arquivo *vtk*: Descreve o objeto (malha refinada), utilizando o formato *vtk*, por exemplo:

- Cabeçalho:

```
vtk DataFile Version 1.0
Mesh from SHE
ASCII
```



Figura 5.4: Interface do modelador para entrada dos dados para simulação.

```
DATASET UNSTRUCTURED_GRID
```

- Pontos: número de pontos, tipo das coordenadas dos pontos e coordenadas dos pontos em cada linha.

```
POINTS 155 float
```

```
1.00000000 3.00000000 0.00000000
```

```
...
0.00000000 0.00000000 0.00000000
```

- Células: número de células, código verificador e em cada linha a quantidade de vértice da célula e os identificadores destes vértices. Note que, por se tratar de uma malha triangular a quantidade de vértice em cada célula será sempre três, diferente do PSLG descrito anteriormente que pode ser variado.

```
CELLS 166 664
3 104 0 1
```

```
...
```

```
3 108 107 106
```

- Tipo das curvas poligonais: Número de células e o tipo de cada célula. O tipo de célula 5 no formato vtk corresponde a triângulos.

```
CELL_TYPES 166
5 5 ... 5
```

2. Arquivo dat: Contém os dados necessários para a simulação, por exemplo:

```
Modelo: NomeExemplo
```

```
Dados do modelo:
```

```
Tempo inicial: 0.00000000
```

```
Tempo final: 0.02000000
```

```
Ciclo inicial: 0
```

```
Ciclo final: 10000000
```

```
Espacamento de tempo para impressao: 0.00010000
```

```
Espacamento de tempo para gravacao automatica: 0.0010
```

```
Escala de comprimento em metros: 0.00200000
```

```
Escala de velocidade em metros por segundo: 3.13000000
```

Densidade: 1.00000000
Força de gravidade: 9.81000000
Gravidade em X: 0.00000000
Gravidade em Y: 1.00000000
Viscosidade: 0.000001000000000000
Numero de Reynolds: 6260.00000000
Numero de Frouden: 0.04475100
Numero de Strouhal: 1.00000000
Incremento de tempo inicial: 0.00000100
Tolerancia para a solucao da equacao de Poisson: 0.0000
00001
Fator de controle de passo -0-: 0.10000000
Fator de controle de passo -1-: 0.50000000
Fator de controle de passo -2-: 0.50000000
Metodo para solucao das equacoes de velocidades: DFG
Tipo de escoamento: Newtoniano

3. Arquivo meshdat: Contém os valores iniciais da velocidade e da pressão no objeto, como também as condições de contorno, escritos da seguinte forma:

Dados de uma malha da SHE
ASCII
DADOS E CONDICÕES DE CONTORNO

VELOCIDADE U	VELOCIDADE V	TIPO DE VERTICE
1.00000000	0.00000000	1
...		

```
2.00000000      0.00000000      2
PRESSAO
3.010
...
2.132
```

Neste arquivo, o campo TIPO DE VERTICE classifica se o vértice faz parte da entrada ou saída de fluido no objeto, ou se está na fronteira, da seguinte forma:

Tipo 0 - Vértice outflow : É o vértice que pertence a fronteira do objeto e no qual há saída de fluido.

Tipo 1 - Vértice inflow : É o vértice que pertence a fronteira do objeto e no qual há entrada de fluido.

Tipo 2 - Vértice boundary : É o vértice que pertence a fronteira e que não é nem outflow e nem inflow.

Tipo 3 - Vértices interno : É o vértice que está no interior do objeto, que simplesmente não é boundary, nem inflow e nem outflow.

5.2 Visualizador

O simulador cria dois arquivos de saída que serão usados na entrada do visualizador. Esses arquivos são:

1. Arquivo vtk: Descreve o objeto (malha) usada na simulação no formato vtk.
2. Arquivo meshdat: Contém os valores da velocidade, os valores da pressão e as condições de contorno do objeto.

Estes arquivos são escritos como os exemplos utilizados na Seção 5.1.

Há inúmeras estratégias que podem ser utilizadas para converter dados numéricos em imagens computacionais. Os valores da velocidade e da pressão, são valores reais. O visualizador cria uma tabela de cores, para que estes valores representem cores.

A Figura 5.5 exemplifica tipos diferentes de visualização dos dados no objeto, sendo estes, a velocidade na direção u , a velocidade na direção v , e a pressão em cada célula.

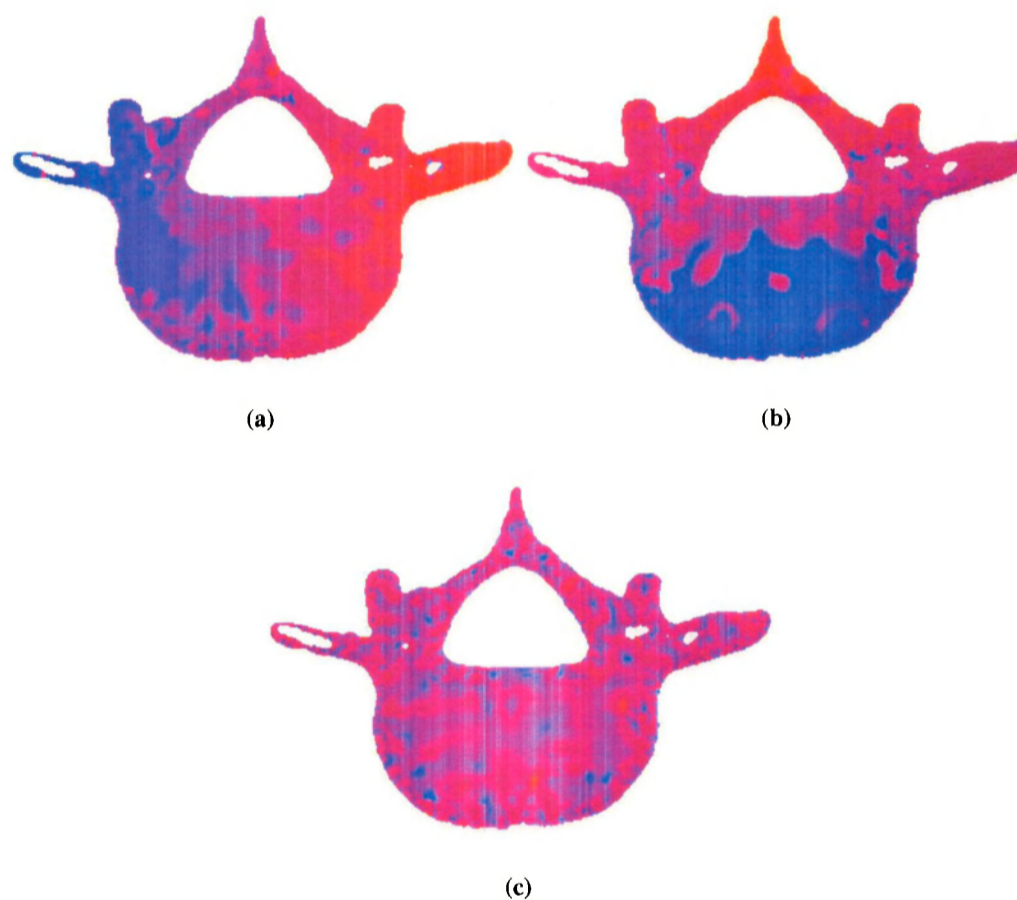


Figura 5.5: (a) Visualização da velocidade na direção u , (b) Visualização da velocidade na direção v , (c) Visualização da pressão.

No visualizador existem outras opções de manuseio do objeto, tais como, *wireframe* (Figura 5.6), *zoom* para aproximar ou distanciar o objeto (Figura 5.7 aproxima a Figura 5.3(a)), translação, etc.

Os triângulos da Figura 5.3(a) só são visíveis através da ferramenta de *zoom*.

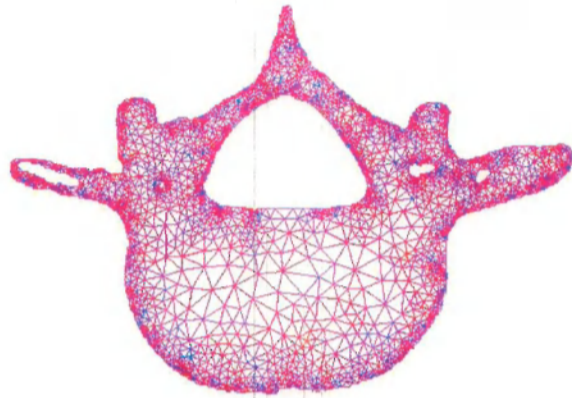


Figura 5.6: Opção de wireframe no visualizador.

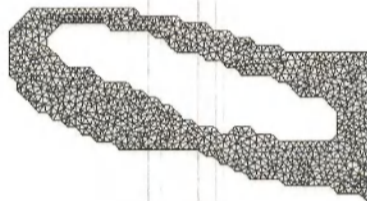


Figura 5.7: Opção de zoom no visualizador: aproximação do canto da Figura 5.3(a).

Está em desenvolvimento a plotagem de isolinhas além de mais técnicas de textura para auxiliar na visualização.

Este sistema representa a fase inicial de um trabalho que se estenderá muito. Futuramente várias opções serão adicionadas, o que o tornará mais completo e abrangente.

Conclusão

Ambientes de simulação de escoamentos incompressíveis com superfícies livres em duas e três dimensões têm sido construído ao longo dos últimos anos pelo grupo de Matemática Aplicada do ICMC. Estes ambientes manipulam apenas malhas estruturadas, o que muitas vezes inviabiliza sua utilização em domínios muito complexos.

O trabalho apresentado nesta dissertação representa o início de uma nova etapa para o grupo. Com a implementação do ambiente Umflow-2D, o grupo ampliou as opções de simulação, sendo agora possível a manipulação de malhas não estruturadas.

A manipulação de malhas não estruturadas torna possível o uso de objetos (domínios) mais complexos, e que eventualmente se aproxime mais de objetos reais.

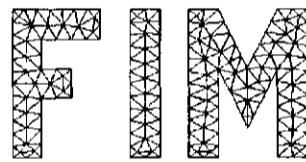
Os algoritmos de refinamento Delaunay implementados permitem que o usuário refine o objeto de acordo com seu interesse. Lembrando que o algoritmo de refinamento de Chew gera uma malha mais refinada, enquanto o algoritmo de refinamento de Ruppert tenta manter na malha os triângulos que se aproximam de triângulos equiláteros, mesmo que estes triângulos não tenham o mesmo tamanho dos demais.

O aprimoramento do ambiente Umflow-2D já é um objetivo do grupo, pois existem vários métodos que podem ser acrescentados com o intuito de melhorar e expandir esta primeira versão.

Alguns métodos que serão acrescentados em cada módulo do sistema:

- Modelador: na interface teremos mais opções de manuseio para o usuário, como por exemplo, a opção de criar um objeto com clicks do mouse; no gerador de malhas outros algoritmos de triangulação poderão ser implementados.
- Simulador: outros métodos de simulação, a simulação com outros tipos de fluidos e também a opção de superfície livre.
- Visualizador: diversas opções e recursos para visualização dos dados.

Característico de um novo trabalho, as opções de extensão do mesmo são promissoras.



Referências Bibliográficas

- Aurenhammer, F. and Klein, R. (2000). *Voronoi diagrams*, pages 201–290. Handbook of Computational Geometry. Elsevier Science Publishing.
- Baumgart, B. G. (1975). A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, pages 589–596.
- Bern, M. W. and Eppstein, D. (1995). Mesh generation and optimal triangulation. In Du, D.-Z. and Hwang, F. K.-M., editors, *Computing in Euclidean Geometry*, number 4 in Lecture Notes Series on Computing, pages 47–123. World Scientific, second edition.
- Bern, M. W., Eppstein, D., and Gilbert, J. R. (1990). Provably good mesh generation. In *IEEE Symposium on Foundations of Computer Science*, pages 231–241.
- Cheng, S. and Dey, T. (2002). Quality meshing with weighted delaunay refinement.
- Chew, L. P. (1989). Guaranteed-quality triangular meshes. Technical Report TR-89-983, Cornell University.
- Chew, L. P. (1993). Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th ACM Symp. Comp. Geometry*, pages 274–280.
- Chiyokura, H. (1988). *Solid Modelling with DESIGNBASE - Theory and Implementation2*. Wesley Publishing Company.

Referências Bibliográficas

- Deitel, H. M. and Deitel, P. J. (2002). *C++ How to Program, Fourth Edition*. Prentice Hall Professional Technical Reference.
- Edelsbrunner, H. (2000). Triangulations and meshes in computational geometry. *Acta Numerica*, pages 133–213.
- Edelsbrunner, H. and Tan, T. S. (1992). An upper bound for conforming delaunay triangulations. In *Symposium on Computational Geometry*, pages 53–62.
- Fortune (1992). Voronoi Diagrams and Delaunay Triangulations. In Du, D.-Z. and Hwang, F., editors, *Computing in Euclidean Geometry*, number 1 in Lecture Notes Series on Computing, pages 193–233. World Scientific, Singapore.
- Guibas, L. and Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2):74–123.
- Lo, S. H. (1985). A new mesh generation scheme for arbitrary planar domains. *Int. J. for Numerical Methods in Engineering*, 21:1403–1426.
- Lohner, R. (1996). Progress in grid generation via the advancing front technique. In *Engineering with Computers*, volume 12, pages 186–210. Springer-Verlag.
- Marcum, D. L. and Weatherill, N. P. (1995). Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9):pp.1619–1625.
- Miller, G. L., Talmor, D., Teng, S.-H., and Walkington, N. (1995). A Delaunay based numerical method for three dimensions: generation, formulation, and partition. In *27th Annu. ACM Sympos. Theory Comput.*, pages 683–692.
- Miller, Gary L., S. E. P. and Walkington, N. J. (2003). When and why ruppert’s algorithm works. In *12th International Meshing Roundtable*, pages 91–102, Santa Fe, New Mexico, USA. Sandia National Laboratories, Carnegie Mellon University, Pittsburgh, PA 15213 U.S.A.
- Mäntyla, M. (1988). *Introduction to Solid Modeling*. W. H. Freeman & Co.

Referências Bibliográficas

- Nonato, L., Castelo, A., and de Oliveira, M. (2002). A topological approach for handling triangle insertion and removal into two-dimensional unstructured meshes. *Submetido para Engineering with Computers*.
- Nonato, L., Castelo, A., Liziér, M. A. S., and de Oliveira, M. (2003). Topological approach for detecting objects from images. In *Vision Geometry XII, 2004 (aceito para publicação)*, San Jose, California. Proceedings of SPIE. SPIE, 2003.
- Owen, S. J. (1998). A survey of unstructured mesh generation technology.
- Ruppert, J. (1995). A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585.
- Shewchuk, J. R. (1999). Lecture Notes on Delaunay Mesh Generation. Department of Electrical Engineering and Computer Science - Berkeley, CA94720.
- Srinivasan, V., Nackman, L. R., Tang, J.-M., and Meshkat, S. (1990). Automatic mesh generation using the symmetric axis transformation of polygonal domains. Technical Report RC 16132, Comp.Science, IBM Research Division, Yorktown Heights, NY.
- Teng, S.-H. and Wong, C. W. (2000). Unstructured mesh generation: Theory, practice, and perspectives. *Int. J. Computational Geometry and Applications*, 10(3):227–266.
- Wright, R. S. and Sweet, M. (1996). *OpenGL Superbible*. Waite Group Press, Corte Madera, CA.