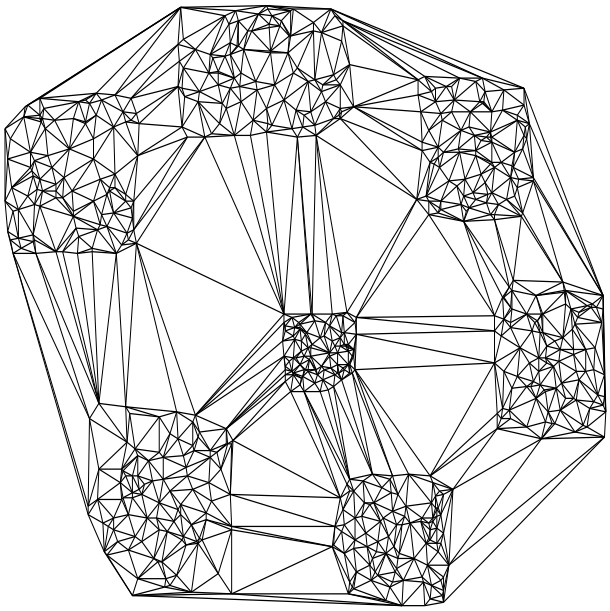
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Presidente Prudente

Triangulação de Delaunay no Plano: Algoritmos de Inserção



Anderson Gregório da Silva
Monografia para Conclusão de Curso

Orientador: Marco Antônio Piteri

PRESIDENTE PRUDENTE
JANEIRO DE 2008

ANDERSON GREGÓRIO DA SILVA

TRIANGULAÇÃO DE DELAUNAY NO PLANO: ALGORITMOS DE INSERÇÃO

Monografia apresentada ao curso de Bacharelado em Ciência da Computação da Faculdade de Ciências e Tecnologia/UNESP - Câmpus de Presidente Prudente, como requisito para a conclusão do curso.

PRESIDENTE PRUDENTE
2008

TERMO APROVAÇÃO

ANDERSON GREGÓRIO DA SILVA

TRIANGULAÇÃO DE DELAUNAY NO PLANO: ALGORITMOS DE INSERÇÃO

Monografia aprovada como requisito parcial para conclusão do curso de Bacharelado em Cîencia da Computação, da Faculdade de Cîencia e Tecnologia/UNESP, pela seguinte banca examinadora:

Orientador: Prof. Dr. Marco Antônio Piteri
Departamento de Matemática, Estatística e Computação - UNESP

Prof. Dr. Ronado Celso Messias
Departamento de Matemática, Estatística e Computação - UNESP

Prof. Dr. Messias Meneguetti Junior
Departamento de Matemática, Estatística e Computação - UNESP

Presidente Prudente, 30 de Janeiro de 2008

Agradecimentos

A realização do presente curso foi possível devido à colaboração de muitas pessoas que me auxiliaram durante os 5 anos de curso. Manifesto assim minha gratidão:

primeiramente a Deus, que sempre me ajudou a não desistir dessa longa caminhada, e que sempre me acompanha durante a execução de minhas tarefas.

Resumo

Esse trabalho apresenta uma discussão teórica e os resultados obtidos a partir da implementação de algoritmos de inserção aplicados ao problema de triangulação de Delaunay no plano. A representação de todas as fases do processo construtivo é baseada na estrutura de dados topológica *winged edge modificada*. Dentre todas as triangulações existentes e associadas a um mesmo conjunto de pontos, a triangulação de Delaunay é aquela que não só maximiza o menor dos ângulos, mas garante sob certas condições, a unicidade da triangulação. Nesse sentido, a triangulação de Delaunay associada a um conjunto de pontos em R^2, apresenta características de regularidade local e global, razão pela qual é indispensável para um amplo conjunto de aplicações, como por exemplo, modelagem digital de terrenos e geração automática de malhas de elementos finitos.

Os resultados previamente obtidos serviram como ponto de partida para que um projeto associado ao diagrama de Voronoi, que é o dual da triangulao de Delaunay, pudesse ser obtido de maneira indireta.

Palavras-Chave:

Triangulação de Delaunay

Estruturas de dados topológicas

winged-edge modificada

Abstract

That work presents a theoretical discussion and the results obtained from the implementation of insert algorithms applied to the Delaunay triangulation problem in the plane. The representation of all of the phases of the constructive process is based on the *modified winged edge* topological data structure.

Among all of the existent triangulations and associated to a same set of points, the Delaunay triangulation is that not only maximizes the smallest of the angles (MaxMin Criterium), but it guarantees under certain conditions, the unicity of the triangulation. In that sense, the triangulation of Delaunay associated to a set of points in R^2, presents characteristics of local and global regularity, reason for the which is indispensable for a wide group of applications, as for instance, terrain digital modelling and automatic finite elements meshes generation. The results previously obtained served as starting point so that a project associated to the diagram of Voronoi, that is the dual of the Delaunay triangulation, it could be obtained in an indirect way.

Keywords

Delaunay Triangulation

Topological data structures

modified winged-edge

Notação

$\det(A)$	determinante correspondente à matriz A;
(x, y)	par ordenado de pontos do espaço euclideano 2D;
$\mathbf{G}(V, E)$	grafo $\mathbf{G}$ com conjunto de vértices V e arestas E;
R^n	espaço euclideano de dimensão n;
P	conjunto finito de pontos no espaço euclideano de dimensão R^n;
$\mathbf{p}_{ix}$	abscissa do ponto $\mathbf{p}_i$;
$\mathbf{p}_{iy}$	ordenada do ponto $\mathbf{p}_i$;
$Conv(P)$	fecho convexo do conjunto P;
$\Delta(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$	triangulo formado pelos pontos $\mathbf{p}_i, \mathbf{p}_j$ e $\mathbf{p}_k$;
$Area(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$	área orientada do $\Delta(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$;
$dist(\mathbf{q}, \mathbf{p})$	distância euclideana entre os pontos $\mathbf{p}$ e $\mathbf{q}$;
$\overline{\mathbf{ab}}$	segmento definido pelos pontos $\mathbf{a}$ e $\mathbf{b}$;
$V(P)$	Diagrama de Voronoi associada ao conjunto de pontos P;
$V(\mathbf{p}_i)$	região de Voronoi associada ao ponto $\mathbf{p}_i$;
$D(P)$	Triangulação de Delaunay associada ao conjunto de pontos P;
Γ	Representação do Parabolóide no espaço euclidiano R^3.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Capítulo 2

exemplo

Essa monografia apresenta uma fundamentação teórica e os resultados práticos obtidos segundo a proposta inicial presente no ante-projeto de pesquisa elaborado no início do ano de 2007.

O projeto aborda um tema clássico na área de geometria computacional e consiste no estudo e implementação de técnicas algorítmicas para resolução de problemas de natureza geométrica e topológica. No escopo da área de geometria computacional trabalhamos especificamente na resolução de problemas associados a subdivisões planares, mais especialmente aquelas em que todas as regiões são triangulares. Esse problema também é conhecido por *triangulação*.

Do ponto de vista teórico o problema de triangulação pode ser tratado em muitas dimensões, porém suas aplicações mais imediatas estão restritas ao espaço euclidiano de dimensão 2 e 3. A importância do assunto é tanta que existe uma gama significativa de problemas que podemos citar em que a representação de um objeto 2D ou 3D se mostra de suma importância para melhor visualização ou até mesmo para resolução do problema em si, por exemplo, a representação primária de um objeto dado um conjunto finito de pontos obtidos por amostragem, através de imagens de satélite, ressonância magnética, ultra-sonografia ou por meio de um mecanismo de leitura automático, como por exemplo, um scanner 3D gera um subconjunto de ordem 3 e dá origem a um conjunto de triângulos que irá aproximar a superfície do objeto que está sendo modelado. Existem outras aplicações que envolvem o esquema de triangulação, em que o problema de triangulação aparece somente como um subproblema a ser resolvido, são eles:

1 Visão por computador/Robótica;

2 Reconstrução de Curvas e Superfícies (modelagem digital de terreno, arque-

ologia, medicina);

3 Reconstrução de objetos tridimensionais a partir de seções transversais;

4 Visualização de superfícies matemáticas;

5 Geração de malhas de elementos finitos.

2.1 Organização do Trabalho

O capítulo 3 contém toda a fundamentação teórica necessária para entender as técnicas algorítmicas que foram usadas para viabilizar as etapas relativas as diferentes implementações realizadas. Discute sobre as primitivas geométricas, a estrutura de dados topológica e os operadores de Euler que foram implementados por meio de operadores topológicos.

Capítulo 3

Fundamentação Teórica

3.1 Triangulação

Antes de iniciarmos ativamente os estudos sobre a estrutura de dados e as primitivas que utilizamos nas nossas implementações daremos uma introdução básica sobre o que seria uma triangulação, sua importância, os diversos tipos e sobre que tipo de dados aplicamos nossos algoritmos.

Triangulação é um problema fundamental em geometria computacional, pois o primeiro passo para se trabalhar com objetos geométricos complexos é fazer a decomposição em objetos mais simples, o objeto geométrico mais simples que conhecemos no plano é o triângulo e no espaço o tetraedro. Algumas aplicações clássicas conhecidas que utilizam triangulação podem ser, por exemplo, análise de elementos finitos e computação gráfica. Uma triangulação também pode ser referenciada na literatura como uma *malha de elementos finitos* [25] ou *malha de elementos triangulares*.

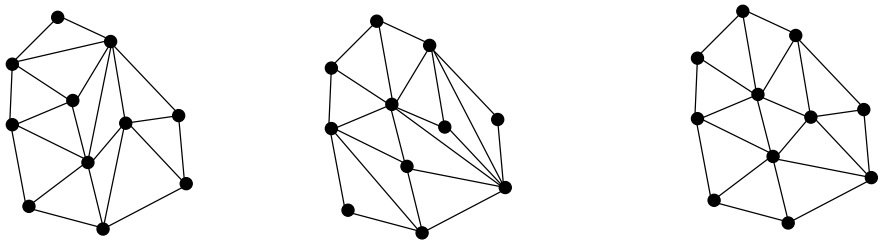
Quando trabalhamos no plano, geralmente utilizamos um conjunto de pontos dado como entrada, tais pontos podem ser obtidos por meio de um escaneamento feito em uma superficie planar. Porém, quando realizamos isso geralmente há ocorrência de pontos diferentes com mesma coordenada, e o ideal é termos pontos únicos dados no plano, para isso o que geralmente se faz é uma permutação entre os dados de entrada, e isso faz com que a ordem de processamento dos pontos não seja a mesma ordem do scan, esse princípio denotaremos por *randomização*[15], permitindo que a probabilidade de termos dois pontos de mesma coordenada diminua.

Uma triangulação pode ser entendida como ligação de cada ponto do conjunto através de uma aresta (também denotado como **corda**) formando um

triângulo. Existirão arestas que serão compartilhados por dois triângulos, a triangulação ocorre quando todos os pontos constituem os *vértices* dos triângulos formados.

Um problema geométrico interessante é o *Fecho Convexo*, trata-se do menor conjunto convexo que contém todos os pontos do conjunto dado, ver figura 3.2 representado por linhas sólidas mais espessas. É importante mencioná-lo pois ele contém todos os pontos de um conjunto de pontos já triangularizados, existem diversos algoritmos capazes de computá-lo, mas se conseguirmos a triangulação do conjunto automaticamentes obteremos o *Fecho Convexo* relativo àquele conjunto de pontos.

A triangulação de um conjunto de pontos não possui unicidade. Em outras palavras, a partir de um mesmo conjunto de pontos é possível obter diferentes triangulações cada uma delas com $2(n-1)k$ triângulos e $3(n-1)-k$ arestas, onde k é o número de pontos pertencentes ao (*Fecho Convexo*) da triangulação. No caso a forma dos triângulos também podem influenciar na aplicação que se deseja fazer, é necessário evitar triângulos muito finos(ângulos internos muito agudos) na triangulação como na figura 3.1 a) e b), o ideal é termos triângulos o mais próximo possível dos triângulos equiláteros com os ângulos se aproximando de 60^{o}, figura 3.1 c).



a) b) c)

Figura 3.1: Possíveis triangulações para um mesmo conjunto de pontos.

Foi provado que quando um quarto ponto se encontra no exterior do círculo definido por três pontos, significa que os ângulos internos do triângulo que definiu esse círculo serão os máximos que podem ser obtidos para esse elemento na triangulação, se todos os elementos da triangulação obedecerem essa propriedade significa que é a melhor triangulação para aquele conjunto de pontos dado como entrada, essa triangulação especial é única sob certas condições, e recebeu o nome de *Triangulação de Delaunay* figura 3.2 representado por linhas sólidas mais finas. Falaremos de suas propriedades, relações e casos degenerados no capítulo **??**.

O Diagrama de Voronoi é outro problema geométrico que mencionaremos

durante a descrição, este por sua vez tem relação direta com a triangulação de Delaunay, figura 3.2 representado por linhas tracejadas, pois é o dual da triangulação, foi observando esse diagrama que Delaunay obteve sua triangulação e a relação entre os dois problemas geométricos, se conseguir resolver um dos problemas o outro poderá ser obtido de maneira indireta. Explicaremos mais detalhadamente as relações no capítulo **??**.

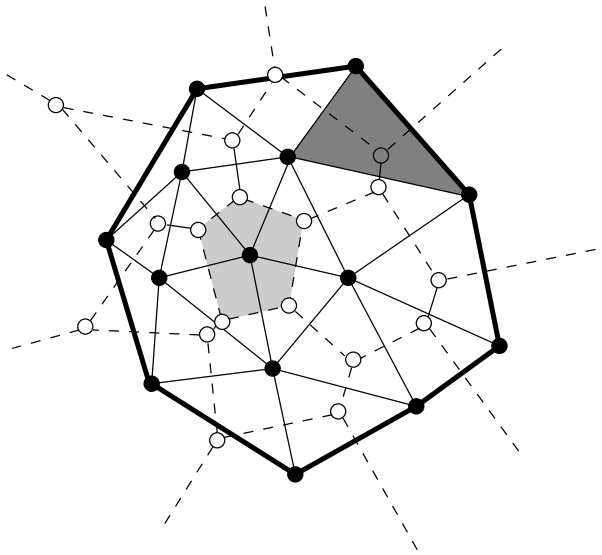


Figura 3.2: Relacionamento entre a *Triangulação de Delaunay*, o *Diagrama de Voronoi* e o *Fecho Convexo* para um mesmo conjunto de pontos. A área hachurada mais clara trata-se de uma região de Voronoi, e a área hachurada mais escura é um triângulo de Delaunay.

3.2 Primitivas Geométricas

Nessa pequena introdução falaremos sobre a importância das primitivas geométricas numa triangulação e denotaremos algumas propriedades relevantes.

Pontos cocirculares e polígonos inscritos possuem muitas propriedades especiais que permitem uma construção eficiente de uma triangulação equiangular [29]. Algumas dessas propriedades serão descritas a partir dos *LEMAS*.

Suponhamos que temos n pontos cocirculares, ligados por n *arestas* para construir um polígono convexo inscrito. As n *arestas* são cordas do círculo inscrito, e chamaremos estas de **cordas exteriores**. Os demais segmentos entre os pares dos n pontos são as **cordas interiores**. Uma corda interior que

omite exatamente um *vértice*, ligando o *vértice* i ao $(i + 2)$ de n é chamada de **corda intermediária**. Uma observação trivial, mas vale ressaltar, é que qualquer triangulação deve conter no mínimo duas **cordas intermediárias**.

Um ângulo está inscrito no círculo se o *vértice* pertencer ao círculo e os dois raios interseseccionam o círculo. Cada ângulo de cada triângulo da triangulação cocirculares é um ângulo inscrito. Um fato geometricamente clássico é que para um ângulo inscrito no círculo o arco sob o ângulo é duas vezes maior que o grau angular. Portanto todos os ângulos inscritos subtendidos sobre o mesmo arco tem o mesmo tamanho.

Em qualquer triangulação de um *polígono* de n lados inscrito num círculo, as **cordas exteriores** são as *arestas* daquela triangulação. Cada **corda exterior** aparece num triângulo simples e para esse triângulo, o ângulo oposto à *aresta* exterior é sempre igual à metade do arco do lado de fora da *aresta*. Devido essa razão temos:

Lema 1 *: Cada triangulação de um polígono de n lados inscrito terá entre seus ângulos o mesmo conjunto de n ângulos, isto é, os n ângulos opostos às n cordas do lado de fora.*

Cada **corda interior** de uma triangulação de um *polígono* de n lados inscrito pertencerá a dois triângulos, e os tamanhos dos dois ângulos opostos às cordas nos dois triângulos são determinados pelo comprimento da corda e não pela triangulação em si. Isto é, os dois ângulos são suplementares (soma igual a 180°) e cada um equivale à metade do arco correspondente subtendido pela corda (um arco maior, um arco menor).

Visto que todos ângulos da triangulação do *polígono* de n lados inscrito (exceto os *n-conjuntos* ligados associados com as cordas exteriores) são dados em pares de suplementares, um acutângulo e outro obtusângulo, para maximizar lexicograficamente os acutângulos da triangulação é necessário maximizar todos os ângulos. Mas desde que os acutângulos para as cordas interiores forem ordenados de acordo com o comprimento da corda. Temos entao:

Lema 2 *: Uma triangulação de um polígono de n lados inscrito terá maximização lexicograficamente dos ângulos dispostos em sequência crescente se e somente se maximizarmos uma sequência crescente de arestas e isso acontece se maximizamos os arcos ao longo das* arestas *do triângulo.*

Visto que podemos ignorar as cordas exteriores ligadas criando a sequência lexicograficamente crescente de cordas, iremos nos preocupar com a trian-

gulação que maximiza a sequência crescente de cordas interiores. Já que qualquer corda interior em uma triangulação que não é uma corda intermediária deve ter a menor corda conectando pontos interiores com menor arco, e segue:

Lema 3 *: O comprimento mínimo de uma corda interior da triangulação de um polígono é uma corda intermediária.*

Como queremos maximizar a *aresta*(corda) interior mais curta, devemos tentar somar poucas cordas intermediárias tantos quanto possível, e então adicionaremos somente aquela mais comprida. Como temos que adicionar no mínimo duas cordas intermediárias, nossa estratégia deve ser somar as duas maiores, se possível.

Lema 4 *: Qualquer triangulação equiangular de pontos cocirculares tem exatamente duas cordas intermediárias.*

Prova O gráfico de qualquer triangulação tendo três ou mais cordas intermediárias é uma árvore, com no mínimo um *vértice* de grau três. Esse *vértice* corresponde ao triângulo formado por três cordas interiores. Seja **e** a *aresta* mais curta de um certo triângulo. O *vértice* **v** oposto a **e** é agudo. Removendo **e**, e trocando a porçao de triangulação pelo menor arco subinclinado por um conjunto de cordas anexados a **v**, é fácil ver que todas as novas cordas são mais compridas que a corda que foi trocada, implicando numa sequência crescente de cordas.

A seguir trataremos das primitivas em si que usaremos, explicando em detalhes o uso delas na triangulação, e como são contruídas a partir de uma teoria matemática *euclidiana*.

3.3 As primitivas

3.4 CounterClockWise

A seguir introduziremos a noção de orientação de uma sequência de $n+1$ pontos em R^n. Sob essa concepção, seremos capazes de implemetarmos operações primitivas para $n+1$ pontos.

A orientação de uma sequência de pontos $(\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ em R^n é negativa ou positiva sem os $n+1$ pontos serem *co-hiperplanares* no qual a orientação é

indefinida. O caso excepcional é uma degeneralidade que pode ser ignorada se os pontos são perturbados. Definiremos que a orientação da seqüencia depende somente das posições relativas dos pontos mutuamente e não nas suas posições absolutas.

Se a dimensão $n = 1$, então a orientação de $(\mathbf{p}_1, \mathbf{p}_2)$ é positiva se $\mathbf{p}_1 > \mathbf{p}_2$ e negativa se $\mathbf{p}_1 < \mathbf{p}_2$ comparar com a figura 3.3. Se $n = 2$, então$(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ tem orientação positiva se três pontos rotacionam se à esquerda no plano, isto é, $\mathbf{p}_3$, fica à esquerda da linha que passa por $\mathbf{p}_1$,$\mathbf{p}_2$ nesta ordem. Se $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ faz uma rotação à direita, então a orientação é negativa. Note que a orientação de $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ é a mesma de $(\mathbf{p}_1, \mathbf{p}_2)$ como visto. De fato, a linha que passa por $\mathbf{p}_2$ e $\mathbf{p}_3$ pode ser identificada com R^1 tão logo escolhamos a direção da linha. Essa direção é dada pela localização de $\mathbf{p}_1$; que vai da esquerda para direita a partir de $\mathbf{p}_1$.

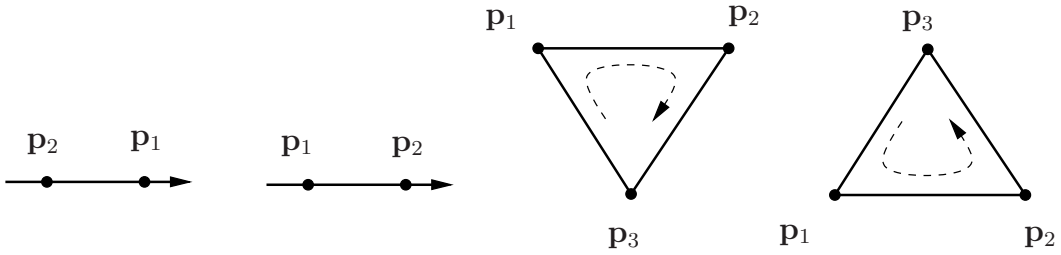


Figura 3.3: Ilustração dos possíveis resultados da primitiva *Ccw(.)*

Para $n > 2$, a orientação de $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$, é dada a partir da resolução de uma determinante, no qual pode ser usada a técnica da eliminação de Gauss, Laplace ou outras técnicas para resolvê-lo.

Seja $\mathbf{p}_1 = (\mathbf{p}_{1x}, \mathbf{p}_{1y})$, $\mathbf{p}_2 = (\mathbf{p}_{2x}, \mathbf{p}_{2y})$, $\mathbf{p}_3 = (\mathbf{p}_{3x}, \mathbf{p}_{3y})$ três pontos no plano, consideraremos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ indefinidos se forem coplanares. No caso indefinido, o ponto $\mathbf{p}_3$ é uma combinação linear de $\mathbf{p}_1$ e $\mathbf{p}_2$, $\mathbf{p}_3 = \lambda_1\mathbf{p}_1 + \lambda_2\mathbf{p}_2$ com $\lambda_1 + \lambda_2 = 1$. Dessa maneira λ_1, λ_2 existirão se e somente se o determinante de

$$Ccw(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = \begin{pmatrix} 1 & \mathbf{p}_{1x} & \mathbf{p}_{1y} \\ 1 & \mathbf{p}_{2x} & \mathbf{p}_{2y} \\ 1 & \mathbf{p}_{3x} & \mathbf{p}_{3y} \end{pmatrix} = 0 \tag{3.1}$$

Se não acontecer isso então $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ orientam-se para esquerda ou direita e o uso do determinante dará a orientação de $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$.

Para $n + 1$ pontos $(\mathbf{p}_1, \ldots, \mathbf{p}_n)$ o teste de orientação vai indicar como os pontos estão orientados no hiperplano e nesse caso geral a primitiva *Ccw* é dada pelo sinal do determinante:

$$\begin{vmatrix} 1 & \mathbf{p}_{12} & \cdots & \mathbf{p}_{1n} & \mathbf{p}_{11} \\ 1 & \mathbf{p}_{22} & \cdots & \mathbf{p}_{2n} & \mathbf{p}_{21} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & \mathbf{p}_{n+1,2} & \cdots & \mathbf{p}_{n+1,n} & \mathbf{p}_{n+1,1} \end{vmatrix} \tag{3.2}$$

Teste do plano: Seja $T = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ e ht é o único plano que contém todos três pontos de T. Esse plano pode ser orientado se substituirmos o conjunto T por uma seqüencia T, por exemplo, $T = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$. Assim, um lado de ht pode ser o positivo e o outro o negativo, e os referimos como os lados da seqüência T.

$T = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$: $\mathbf{p}_4$ fica no lado positivo (esquerda) se e somente se $\det(Ccw(.)) > 0$ e orienta-se para o lado negativo (direita) se e somente se $\det(Ccw(.)) < 0$

Prova Verificaremos para $\mathbf{p}_1 = (0, 0)$, $\mathbf{p}_2 = (1, 0)$ e $\mathbf{p}_3 = (0, 1)$, geometricamente é obvio que $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ orientam-se para esquerda (lado positivo), de fato.

$$\det \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = 1 \tag{3.3}$$

Portanto girar continuamente $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ para qualquer outra posição à esquerda de T sem termos três pontos colineares desde que o determinante mude continuamente com as coordenadas, o determinante permanecerá sempre positivo durante a rotação e consequentemente positivo em relação à T. Simetricamente toda rotação à direita (negativa) implicará num determinante menor que 0.

3.5 InCírculo

Esta primitiva tem por finalidade verificar se o ponto $\mathbf{p}_4$ está no interior do círculo definido pelos pontos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$. Consideraremos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ degenerados se $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ coplanares ou $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ cocirculares. Para mostrar se os pontos são cocirculares faremos $\mathbf{p}'_1 = (\mathbf{p}_{1x}, \mathbf{p}_{1y}, \mathbf{p}_{1z})$, com $\mathbf{p}_{1z} = \mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2$ e assim por diante. Os pontos serão cocirculares se $\mathbf{p}'_1$, $\mathbf{p}'_2$, $\mathbf{p}'_3$, $\mathbf{p}'_4$ forem coplanares. Em outras palavras, $\mathbf{p}'_4$ é combinação linear de $\mathbf{p}'_1$, $\mathbf{p}'_2$, $\mathbf{p}'_3$ [3]. O resultado de *InCirculo* é obtido, solucionando o seguinte determinante:

$$InCirculo\ (\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) = \begin{vmatrix} \mathbf{p}_{1x} & \mathbf{p}_{1y} & \mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2 & 1 \\ \mathbf{p}_{2x} & \mathbf{p}_{2y} & \mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2 & 1 \\ \mathbf{p}_{3x} & \mathbf{p}_{3y} & \mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2 & 1 \\ \mathbf{p}_{4x} & \mathbf{p}_{4y} & \mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2 & 1 \end{vmatrix}$$

Se:

InCirculo $(\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) = 0$ então os pontos são cocirculares;

InCirculo $(\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) > 0$ então $\mathbf{p}_4$ pertence ao exterior do círculo;

InCirculo $(\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) < 0$ então $\mathbf{p}_4$ pertence ao interior do círculo; Veja a representação na figura 3.5

Essa primitiva é baseada no seguinte lema:

Lema 5 *: O teste InCirculo(.) é equivalente a:*

$$InCirculo(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \begin{vmatrix} \mathbf{p}_{1x} & \mathbf{p}_{1y} & \mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2 & 1 \\ \mathbf{p}_{2x} & \mathbf{p}_{2y} & \mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2 & 1 \\ \mathbf{p}_{3x} & \mathbf{p}_{3y} & \mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2 & 1 \\ \mathbf{p}_{4x} & \mathbf{p}_{4y} & \mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2 & 1 \end{vmatrix}. \tag{3.4}$$

Prova Para provar a validade desse resultado teremos que olhar para equação do parabolóide Γ representado no R^3 de equação 3.5 e ilustrado na figura 3.4.

Seja H um plano não-vertical de equação 3.6, a projeção O_{xy} de $H \bigcap \Gamma$ no plano da origem a um círculo de equação 3.7. Ver figura 3.4 que contém a projeção mencionada.

Seja $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y)$ elevaremos o ponto $\mathbf{p}$ e obtemos um $\mathbf{p}'$ de equação 3.8. Essa transformação $\mathbf{p} \longrightarrow \mathbf{p}'$ é chamada **mapa de elevação**. A elevação dos pontos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ e $\mathbf{p}_4$ resulta respectivamente nas equações 3.9, 3.10, 3.11 e 3.12. Assim se *InCirculo* $(\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) = 0$, significa que $\det(Ccw(.)) = 0$ e o ponto $\mathbf{p}'_4 \in H$ e consequentemente $\mathbf{p}_4$ é cocircular à circunferência definida por $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, caso *InCirculo* $(\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) > 0$ então $\det(Ccw(.)) > 0$ e o ponto $\mathbf{p}'_4$ está acima do plano H e $\mathbf{p}_4$ fora do círculo $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, por último se *InCirculo* $(\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ \mathbf{p}_4) < 0$ então $\det(Ccw(.)) < 0$ e o ponto $\mathbf{p}'_4$ abaixo do plano H e $\mathbf{p}_4$ no interior do círculo $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$.

$$z = x^2 + y^2 \tag{3.5}$$

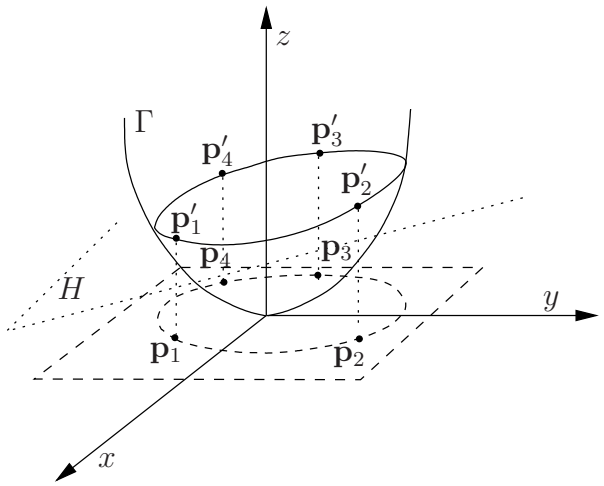


Figura 3.4: Parabolóide Γ e a projeção da área interceptada pelo plano H no R^2

$$z = \alpha x + \beta y + \gamma \tag{3.6}$$

$$x^2 + y^2 = \alpha x + \beta y + \gamma \tag{3.7}$$

$$\mathbf{p}' = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_x{}^2 + \mathbf{p}_y{}^2) \tag{3.8}$$

$$\mathbf{p}'_1 = (\mathbf{p}_{1x}, \mathbf{p}_{1y}, \mathbf{p}_{1x}{}^2 + \mathbf{p}_{1y}{}^2) \tag{3.9}$$

$$\mathbf{p}'_2 = (\mathbf{p}_{2x}, \mathbf{p}_{2y}, \mathbf{p}_{2x}{}^2 + \mathbf{p}_{2y}{}^2) \tag{3.10}$$

$$\mathbf{p}'_3 = (\mathbf{p}_{3x}, \mathbf{p}_{3y}, \mathbf{p}_{3x}{}^2 + \mathbf{p}_{3y}{}^2) \tag{3.11}$$

$$\mathbf{p}'_4 = (\mathbf{p}_{4x}, \mathbf{p}_{4y}, \mathbf{p}_{4x}{}^2 + \mathbf{p}_{4y}{}^2) \tag{3.12}$$

Convém lembrar que o teste do *Incirculo*, em uma situação geral, serve também para orientação de um ponto em relação à uma esfera no R^3 e também no hiperplano. Dada uma sequência de $n+1$ pontos $(\mathbf{p}_2, \ldots, \mathbf{p}_{n+1})$, se $\mathbf{p}_1$ pertencer ao exterior da esfera, dizemos que está orientado positivamente caso

contrário, está orientado negativamente, e o sinal do determinante indica, assim como no caso do círculo, a orientação de $\mathbf{p}_1$ e esse determinante no espaço R^n está ilustrado como abaixo:

$$\begin{vmatrix} \mathbf{p}_{11} & \cdots & \mathbf{p}_{1n} & \mathbf{p}_{11}^2 + \ldots + \mathbf{p}_{1n}^2 & 1 \\ \mathbf{p}_{21} & \cdots & \mathbf{p}_{2n} & \mathbf{p}_{21}^2 + \ldots + \mathbf{p}_{2n}^2 & 1 \\ \vdots & & \vdots & \vdots & \vdots \\ \mathbf{p}_{n+2,1} & \cdots & \mathbf{p}_{n+2,n} & \mathbf{p}_{n+2,1}^2 + \ldots + \mathbf{p}_{n+2,n}^2 & 1 \end{vmatrix} \tag{3.13}$$

É interessante observar que essa mesma idéia foi utilizada para se relacionar a *Triagulação de Delaunay* no plano, com o *Fecho convexo* do mesmo conjunto de pontos projetados sobre um parabolóide.

A resolução do determinante pode ser obtida através da relação:

$$
\begin{array}{ll}
-((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) & +((\mathbf{p}_{1x}) * (\mathbf{p}_{2y}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) \\
+((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3x})) & -((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2y}) * (\mathbf{p}_{3x})) \\
-((\mathbf{p}_{1x}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3y})) & +((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2x}) * (\mathbf{p}_{3y})) \\
+((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) & -((\mathbf{p}_{1x}) * (\mathbf{p}_{2y}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
-((\mathbf{p}_{1y}) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) & +((\mathbf{p}_{2y}) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
+((\mathbf{p}_{1x}) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) & -((\mathbf{p}_{2x}) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
-((\mathbf{p}_{1y}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{4x})) & +((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2y}) * (\mathbf{p}_{4x})) \\
+((\mathbf{p}_{1y}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4x})) & -((\mathbf{p}_{2y}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4x})) \\
-((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x})) & +((\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3y}) * (\mathbf{p}_{4x})) \\
+((\mathbf{p}_{1x}) * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{4y})) & -((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{2x}) * (\mathbf{p}_{4y})) \\
-((\mathbf{p}_{1x}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4y})) & +((\mathbf{p}_{2x}) * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) * (\mathbf{p}_{4y})) \\
+((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4y})) & -((\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) * (\mathbf{p}_{3x}) * (\mathbf{p}_{4y}));
\end{array}
$$

É importante observar que podemos utilizar outros métodos para resolução do mesmo, inclusive o desenvolvimento de *Laplace*, um método muito utilizado na álgebra para o cálculo de determinantes.

Utilizando o desenvolvimento de *Laplace* para calcular *InCirculo*($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$), obtemos a seguinte relação:

InCirculo ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) =

$$
\begin{array}{l}
((\mathbf{p}_{1x}) * ((\mathbf{p}_{2y} * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) + (\mathbf{p}_{3y} * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) + (\mathbf{p}_{4y} * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2)) \\
-(\mathbf{p}_{4y} * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) - (\mathbf{p}_{3y} * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2)) - (\mathbf{p}_{2y} * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)))) \\
+((\mathbf{p}_{1y}) * (-((\mathbf{p}_{2x} * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) + (\mathbf{p}_{3x} * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) + (\mathbf{p}_{4x} * (\mathbf{p}_{2x}^2 \\
+\mathbf{p}_{2y}^2)) - (\mathbf{p}_{4x} * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) - (\mathbf{p}_{3x} * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2)) - (\mathbf{p}_{2x} * (\mathbf{p}_{4x}^2 \\
+\mathbf{p}_{4y}^2))))) + ((\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) * ((\mathbf{p}_{2x} * \mathbf{p}_{3y}) + (\mathbf{p}_{3x} * \mathbf{p}_{4y}) + (\mathbf{p}_{4x} * \mathbf{p}_{2y}) \\
-(\mathbf{p}_{4x} * \mathbf{p}_{3y}) - (\mathbf{p}_{3x} * \mathbf{p}_{2y}) - (\mathbf{p}_{2x} * \mathbf{p}_{4y}))) - ((\mathbf{p}_{2x} * \mathbf{p}_{3y} * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) \\
+(\mathbf{p}_{3x} * \mathbf{p}_{4y} * (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2)) + (\mathbf{p}_{4x} * \mathbf{p}_{2y} * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)) - (\mathbf{p}_{4x} * \mathbf{p}_{3y} \\
*(\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2)) - (\mathbf{p}_{3x} * \mathbf{p}_{2y} * (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2)) - (\mathbf{p}_{2x} * \mathbf{p}_{4y} * (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2)));
\end{array}
$$

Temos ainda que esta mesma primitiva *InCirculo*($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$), pode ser resolvida utilizando o cálculo de área orientada de um triângulo. Isto é, ao invés das relações citadas acima para o cálculo do determinante, podemos usar:

$$
\begin{array}{lrcll}
\textit{InCirculo}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = & (\mathbf{p}_{1x}^2 + \mathbf{p}_{1y}^2) & * & \textit{Area}\,(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) & - \\
 & (\mathbf{p}_{2x}^2 + \mathbf{p}_{2y}^2) & * & \textit{Area}\,(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4) & + \\
 & (\mathbf{p}_{3x}^2 + \mathbf{p}_{3y}^2) & * & \textit{Area}\,(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4) & - \\
 & (\mathbf{p}_{4x}^2 + \mathbf{p}_{4y}^2) & * & \textit{Area}\,(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) & ;
\end{array}
$$

onde *Area* $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ é uma função auxiliar que retorna a área do triângulo

orientado formado pela terna $(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$, isto é, a área é positiva se o triângulo está orientado em sentido anti-horário, e é exatamente no sinal que esta função retorna que está nosso maior interesse, pois ele influenciará no valor final da *InCirculo*. A função auxiliar *Area* utiliza a relação da equação 3.14.

$$Area\ (\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k) = 1/2 * [(\mathbf{p}_{jx} - \mathbf{p}_{ix}) * (\mathbf{p}_{ky} - \mathbf{p}_{iy})] - [(\mathbf{p}_{jy} - \mathbf{p}_{iy}) * (\mathbf{p}_{kx} - \mathbf{p}_{ix})]; \tag{3.14}$$

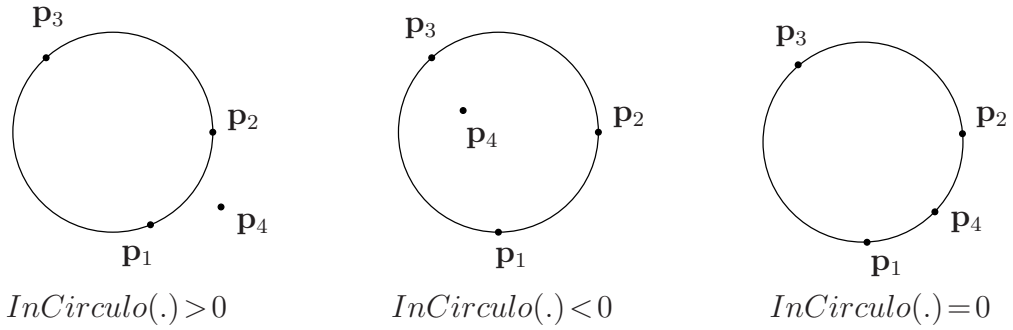


Figura 3.5: Ilustração dos resultados da primitiva *InCículo*.

Uma propriedade importante da *Triangulação de Delaunay* é que a *circunferência* (que circunscreve cada triângulo) não possui nenhum outro ponto da triangulação em seu interior. Assim, através da primitiva InCirculo, conseguimos determinar se o triângulos contruídos durante o processo de discretização da geometria, satisfazem esta importante propriedade.

Embora não seja necessário utilizar as primitivas abaixo, aproveitamos a oportunidade e fizemos um estudo teórico sobre os princípios matemáticos associados à elas e julgamos oportuno descrevê-los para subsidiar futuras implementações.

3.6 Circuncentro de um Elemento Triangular

A primitiva circuncentro, da mesma maneira que as outras primitivas, também é muito importante à *Triangulação de Delaunay*, pelo fato de, por meio dela, identificarmos uma circunferência definida por três pontos e com auxílio da primitiva *Incírculo* poderemos conseguir triângulos com ângulos internos máximos, que se trata exatamente de como trabalha os cálculos para a triangulação feita por *Delaunay*. A seguir definiremos mediatriz, circuncentro e também como calculá-lo.

Mediatriz:

Definição 1 *Consideramos um segmento $\overline{AB}$ e seja M o ponto médio do segmento. A reta perpendicular ao segmento $\overline{AB}$ e que passa por M é denominada* ***mediatriz****.*

Circuncentro:

Definição 2 *Dado um $\Delta(\boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3)$ se traçarmos as mediatrizes dos três lados elas intercectam-se num mesmo ponto* $\boldsymbol{c}$*, que está eqüidistante dos vértices do triângulo. Este ponto* $\boldsymbol{c}$ *é chamado de* ***circuncentro do triângulo****.*

Tomando $\mathbf{c}$ como centro e o raio como a distância de $\mathbf{c}$ a um dos *vértices*, obteremos a circunferência que circunscreve o $\Delta(\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3)$. Esta descrição pode ser observado na figura 3.6

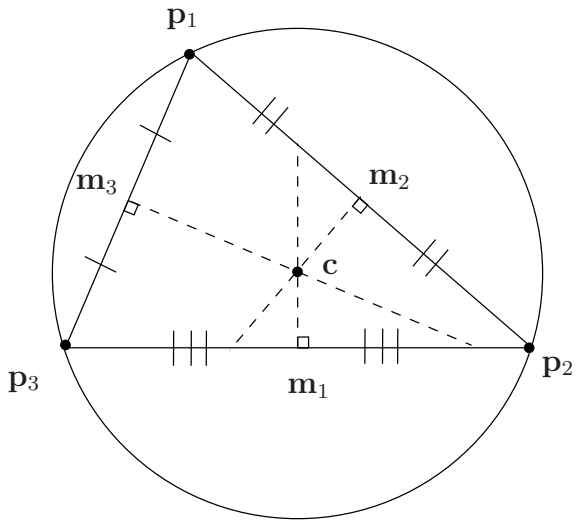


Figura 3.6: Representação do Circuncentro de um elemento triangular e da circunferência por ele determinada.

Prova Seja o $\Delta\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$

Hipótese: $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ mediatrizes de $\overline{\mathbf{p}_2\mathbf{p}_3}$, $\overline{\mathbf{p}_1\mathbf{p}_3}$, e $\overline{\mathbf{p}_1\mathbf{p}_2}$ respectivamente:

1) $\mathbf{m}_1 \cap \mathbf{m}_2 \cap \mathbf{m}_3 = \{\mathbf{c}\}$

2) $\overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_2} \equiv \overline{\mathbf{cp}_3}$

Seja $\mathbf{c}$ o ponto tal que:

$\mathbf{m}_2 \cap \mathbf{m}_3 = \{\mathbf{c}\}$

$\mathbf{c} \in \mathbf{m}_2 \Longrightarrow \overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_3}$

$\mathbf{c} \in \mathbf{m}_3 \Longrightarrow \overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_2} \qquad \Longrightarrow \mathbf{cp}_1 \equiv \mathbf{cp}_2 \equiv \mathbf{cp}_3$

Logo:

1) $\mathbf{m}_1 \cap \mathbf{m}_2 \cap \mathbf{m}_3 = \{\mathbf{c}\}$

2) $\overline{\mathbf{cp}_1} \equiv \overline{\mathbf{cp}_2} \equiv \overline{\mathbf{cp}_3}$

As coordenadas de $\mathbf{c}$ são obtidas conforme as equações descritas a seguir:

$$\mathbf{c}_x = \frac{\begin{vmatrix} (x^2_{\mathbf{p}_3\mathbf{p}_2} + y^2_{\mathbf{p}_3\mathbf{p}_2}) & y_{\mathbf{p}_3\mathbf{p}_2} \\ (x^2_{\mathbf{p}_1\mathbf{p}_2} + y^2_{\mathbf{p}_1\mathbf{p}_2}) & y_{\mathbf{p}_1\mathbf{p}_2} \end{vmatrix}}{2 \begin{vmatrix} x_{\mathbf{p}_3\mathbf{p}_2} & y_{\mathbf{p}_3\mathbf{p}_2} \\ x_{\mathbf{p}_1\mathbf{p}_2} & y_{\mathbf{p}_1\mathbf{p}_2} \end{vmatrix}} \tag{3.15}$$

$$\mathbf{c}_y = \frac{\begin{vmatrix} x_{\mathbf{p}_3\mathbf{p}_2} & (x^2_{\mathbf{p}_3\mathbf{p}_2} + y^2_{\mathbf{p}_3\mathbf{p}_2}) \\ x_{\mathbf{p}_1\mathbf{p}_2} & (x^2_{\mathbf{p}_1\mathbf{p}_2} + y^2_{\mathbf{p}_1\mathbf{p}_2}) \end{vmatrix}}{2 \begin{vmatrix} x_{\mathbf{p}_3\mathbf{p}_2} & y_{\mathbf{p}_3\mathbf{p}_2} \\ x_{\mathbf{p}_1\mathbf{p}_2} & y_{\mathbf{p}_1\mathbf{p}_2} \end{vmatrix}} \tag{3.16}$$

Onde:

$$x_{\mathbf{p}_1\mathbf{p}_2} = \mathbf{p}_{1x} - \mathbf{p}_{2x}, \quad x_{\mathbf{p}_3\mathbf{p}_2} = \mathbf{p}_{3x} - \mathbf{p}_{2x} \tag{3.17}$$

e

$$y_{\mathbf{p}_1\mathbf{p}_2} = \mathbf{p}_{1y} - \mathbf{p}_{2y}, \quad y_{\mathbf{p}_3\mathbf{p}_2} = \mathbf{p}_{3y} - \mathbf{p}_{2y} \tag{3.18}$$

O raio da cincunferência que circunscreve o triângulo segue:

$$\mathbf{R}_{circ} = \frac{1}{2}\sqrt{\frac{(\mathbf{d}_1 + \mathbf{d}_2)(\mathbf{d}_2 + \mathbf{d}_3)(\mathbf{d}_3 + \mathbf{d}_1)}{\mathbf{c}}} \tag{3.19}$$

de maneira que:

$$\mathbf{d}_1 = (\mathbf{p}_3 - \mathbf{p}_1)(\mathbf{p}_2 - \mathbf{p}_1) \quad (3.20)$$
$$\mathbf{d}_2 = (\mathbf{p}_3 - \mathbf{p}_2)(\mathbf{p}_1 - \mathbf{p}_2) \quad (3.21)$$
$$\mathbf{d}_3 = (\mathbf{p}_1 - \mathbf{p}_3)(\mathbf{p}_2 - \mathbf{p}_3) \quad (3.22)$$
$$\mathbf{c} = \mathbf{d}_1\mathbf{d}_2 + \mathbf{d}_2\mathbf{d}_3 + \mathbf{d}_3\mathbf{d}_1 \quad (3.23)$$
$$\mathbf{p}_i = (\mathbf{p}_{ix}, \mathbf{p}_{iy}) \quad (3.24)$$

3.7 Incentro de um Elemento Triangular

Nessa sessão definiremos o Incentro e como obter a circunfência inscrita num triângulo arbitrário. Começaremos definindo bissetriz de um triângulo:

Bissetriz:

Definição 3 *O segmento com extemidades num vértice e no lado oposto que divide o ângulo desse vértice em dois ângulos congruentes é denominada* ***bissetriz do triângulo***

Incentro:

Definição 4 *As três bissetrizes internas de um triângulo interceptam-se num mesmo ponto que está a igual distância dos lados do triângulo. A este ponto de intersecção damos o nome de* ***incentro do triângulo****.*

Podemos assim obter a maior circunferência que pode ser desenhada dentro de um triângulo, esta circunferência é tangente a seus lados (figura 3.7). O centro da circunferência é determinado pelo incentro do triângulo, que representaremos por $\mathbf{c}$. O ponto final do segmento interseccionado com o lado oposto do triângulo é o ponto que tangência entre a circunferência e o triângulo.

Prova Sendo $\Delta\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ de lados $\overline{\mathbf{p}_2\mathbf{p}_3} = l, \overline{\mathbf{p}_1\mathbf{p}_3} = m, \overline{\mathbf{p}_1\mathbf{p}_2} = n$;

Hipótese:

$\overline{\mathbf{p}_1\mathbf{b}_1}, \overline{\mathbf{p}_2\mathbf{b}_2}, \overline{\mathbf{p}_3\mathbf{b}_3}$, são as bissetrizes internas:

Tese:

1) $\overline{\mathbf{p}_1\mathbf{b}_1} \cap \overline{\mathbf{p}_2\mathbf{b}_2} \cap \overline{\mathbf{p}_3\mathbf{b}_3} = \{\mathbf{c}\}$

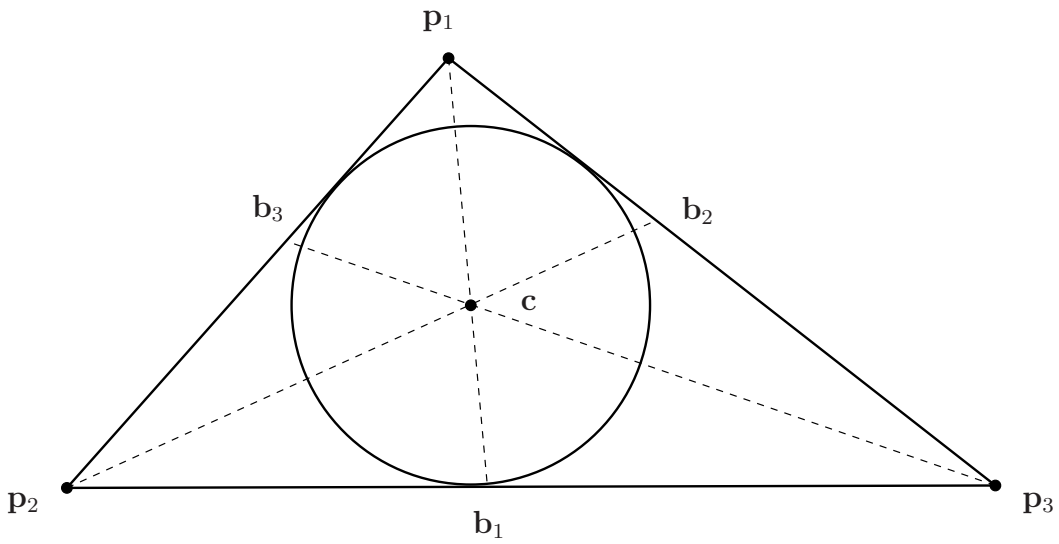


Figura 3.7: Representação do Incentro e da circunfêrencia definida em seu interior.

2) $\mathbf{d}_{c,l} = \mathbf{d}_{c,m} = \mathbf{d}_{c,n}$

Demonstração:

Seja $\mathbf{c}$ o ponto tal que:

$\overline{\mathbf{p}_2\mathbf{b}_2} \cap \overline{\mathbf{p}_3\mathbf{b}_3} = \{\mathbf{c}\}$

$\mathbf{c} \in \overline{\mathbf{p}_2\mathbf{b}_2} \Longrightarrow \mathbf{d}_{c,l} = \mathbf{d}_{c,m}$

$\mathbf{c} \in \overline{\mathbf{p}_3\mathbf{b}_3} \Longrightarrow \mathbf{d}_{c,l} = \mathbf{d}_{c,n} \Longrightarrow \mathbf{c} \in \overline{\mathbf{p}_1\mathbf{b}_1}$

Logo:

1) $\overline{\mathbf{p}_1\mathbf{b}_1} \cap \overline{\mathbf{p}_2\mathbf{b}_2} \cap \overline{\mathbf{p}_3\mathbf{b}_3} = \{\mathbf{c}\}$

2) $\mathbf{d}_{c,l} = \mathbf{d}_{c,m} = \mathbf{d}_{c,n}$

Tomando P o perímetro do *elemento triangular* e R_{ik} o comprimento do segmento de reta definido por dois *vértices* arbitrários do triângulo i e k.

$$\mathbf{c}_x = \frac{R_{\mathbf{p}_3\mathbf{p}_1} x_{\mathbf{p}_2} + R_{\mathbf{p}_1\mathbf{p}_2} x_{\mathbf{p}_3} + R_{\mathbf{p}_2\mathbf{p}_3} x_{\mathbf{p}_1}}{P} \tag{3.25}$$

$$\mathbf{c}_y = \frac{R_{\mathbf{p}_3\mathbf{p}_1} y_{\mathbf{p}_2} + R_{\mathbf{p}_1\mathbf{p}_2} y_{\mathbf{p}_3} + R_{\mathbf{p}_2\mathbf{p}_3} y_{\mathbf{p}_1}}{P} \tag{3.26}$$

O raio da circunferência é obtido pela seguinte relação:

$$\mathbf{r}_d = \sqrt{\frac{(s - R_{\mathbf{p}_3\mathbf{p}_2})(s - R_{\mathbf{p}_1\mathbf{p}_2})(s - R_{\mathbf{p}_1\mathbf{p}_3})}{s}} \tag{3.27}$$

onde $s = \frac{P}{2}$.

3.8 Medidas de Qualidade

Sabemos que o triângulo equilátero é o melhor triângulo em termos de qualidade, suas medidas angulares são as ideais que podem ser obtidas para um elemento triangular. Consequentemente a qualidade de uma malha de elementos triangulares refere-se à quantidade de triângulos cujos ângulos têm valores próximos a 60^o. Evidentemente quando é dado uma nuvem de pontos arbitrária os triângulos devem ser construídos de modo que seus vértices contenham os pontos dados como entrada, como apresentamos na figura 3.1 existem vários modos de se fazer a triangulação de um mesmo conjunto de pontos, a Triangulação de Delaunay possui os melhores triângulos dentre todas as outras possíveis triangulações, em termos da qualidade dos triângulos obtidos.

Dependendo da aplicação é importante fazer a avaliação da malha de elementos triangulares para que possam ser obtidos resultados e a partir desses resultados possa ser feita uma melhor distribuição dos pontos dados como entrada, de modo que a regularidade dos elementos triangulares seja aceitável para aplicação.

Existem vários meios de se fazer análise dos elementos de uma malha dentre elas podemos citar:

1 Análise por ângulo mínimo: nessa abordagem é feito o calculo independente de cada ângulo de cada elemento na malha e limita esse o menor dos ângulos por uma constante, o cálculo da qualidade dos elementos fica sendo as medidas obtidas em relação à constante que limita o ângulo mínimo figura 3.8

2 Razão entre o raio da circunferência inscrita e circunscrita de um elemento triangular: Nessa abordagem podemos calcular o incentro e o circuncentro e o raio através das primitivas mencionadas para que seja possível obter medidas da circunferência para que seja possível realizar um teste de qualidade como desse tipo. Figura 3.9

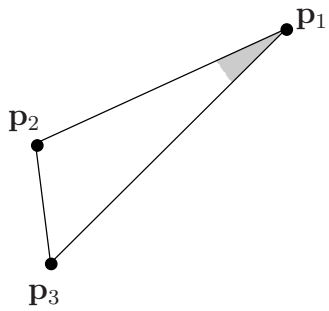


Figura 3.8: Medida de qualidade considerando o ângulo.

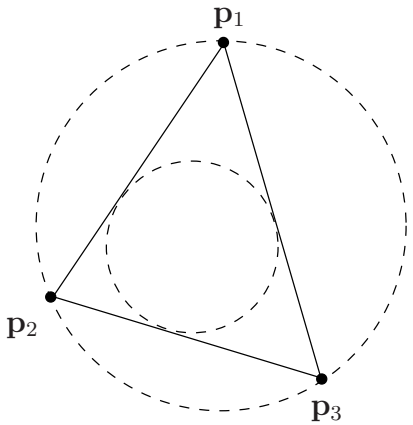


Figura 3.9: Medida de qualidade considerando o incírculo e o circuncírculo.

3.9 Centro de gravidade de um Elemento Triangular

O baricentro também é importante de ser mencionado, pois por meio dele conseguimos o centro de massa em qualquer triângulo. Em seguida definiremos mediana e como se calcula o baricentro do triângulo.

Mediana:

Definição 5 *A mediana de um triângulo é um segmento com extemidades num vértice e no ponto médio do lado oposto.*

Baricentro:

Definição 6 *As três medianas de um triângulo interceptam-se num mesmo ponto que divide cada mediana em duas partes, tal que, a parte que contém o vértice é o dobro da outra. Este ponto de intersecção é chamado de* ***baricentro do triângulo****, também conhecido como* ***centro de gravidade*** *do mesmo e o indicaremos por* $\mathbf{c}$ *(figura 3.10).*

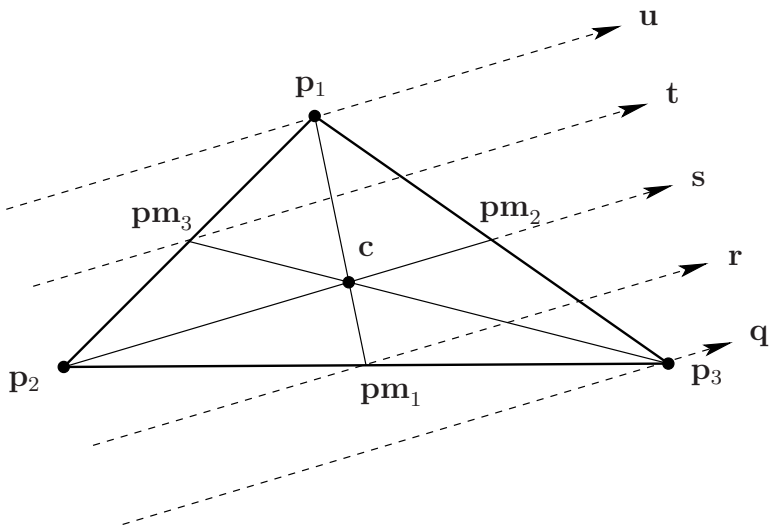


Figura 3.10: Representação do Baricentro.

Prova Consideremos, no $\Delta \mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$, os pontos $\mathbf{pm}_1$,$\mathbf{pm}_2$, e $\mathbf{pm}_3$ como pontos médios de $\overline{\mathbf{p}_2\mathbf{p}_3}$, $\overline{\mathbf{p}_1\mathbf{p}_3}$, e $\overline{\mathbf{p}_1\mathbf{p}_2}$, respectivamente. Vamos demonstrar que existe um ponto $\mathbf{c}$ que está em $\overline{\mathbf{p}_1\mathbf{pm}_1}$, $\overline{\mathbf{p}_2\mathbf{pm}_2}$, e $\overline{\mathbf{p}_3\mathbf{pm}_3}$, respectivamente, e $\mathbf{p}_1\mathbf{c}=\frac{2}{3}\mathbf{p}_1\mathbf{pm}_1$, $\mathbf{p}_2\mathbf{c}=\frac{2}{3}\mathbf{p}_2\mathbf{pm}_2$ e $\mathbf{p}_3\mathbf{c}=\frac{2}{3}\mathbf{p}_3\mathbf{pm}_3$.

Sejam $\mathbf{r}$, $\mathbf{s}$ e $\mathbf{t}$ retas paralelas que dividem o lado $\overline{\mathbf{p}_1\mathbf{p}_3}$ em quatro segmentos congruentes, com $\mathbf{s} = \overline{\mathbf{p}_2\mathbf{pm}_2}$. Consideremos as retas $\mathbf{u}$ e $\mathbf{q}$, ambas paralelas a $\mathbf{r}$, $\mathbf{s}$ e $\mathbf{t}$, passando por $\mathbf{p}_1$ e $\mathbf{p}_3$, respectivamente.

A reta $\mathbf{t}$ divide o segmento $\mathbf{p}_1\mathbf{p}_2$ em dois segmentos congruentes, e, portanto, o ponto, $\mathbf{pm}_3$ está na reta $\mathbf{t}$; além disso, as retas $\mathbf{t}$, $\mathbf{r}$, $\mathbf{s}$ e $\mathbf{q}$ dividem a mediana $\mathbf{p}_3\mathbf{pm}_3$ em três segmentos congruentes, e portanto, se $\mathbf{c}$ é o ponto de intersecção das medianas $\mathbf{p}_2\mathbf{pm}_2$ e $\mathbf{p}_3\mathbf{pm}_3$, temos $\mathbf{p}_3\mathbf{c} = \frac{2}{3}\mathbf{p}_3\mathbf{pm}_3$.

Do mesmo modo, com retas paralelas a $\mathbf{p}_1\mathbf{pm}_1$ mostramos que se $\mathbf{c}'$ é a intersecção das medianas $\overline{\mathbf{p}_3\mathbf{pm}_3}$ e $\overline{\mathbf{p}_1\mathbf{pm}_1}$, então $\mathbf{p}_3\mathbf{c}' = \frac{2}{3}\mathbf{p}_3\mathbf{pm}_3$.

Como sabemos agora que a mediana $\overline{\mathbf{p}_1\mathbf{pm}_1}$ passa por $\mathbf{c}$, podemos concluir que $\mathbf{p}_1\mathbf{c} = \frac{2}{3}\mathbf{p}_1\mathbf{pm}_1$ e $\mathbf{p}_2\mathbf{c} = \frac{2}{3}\mathbf{p}_2\mathbf{pm}_2$.

Tal posição geométrica é obtida através da média aritmética das coordenadas do *elemento triangular*, conforme a relação descrita abaixo:

$$\mathbf{c}_x = \frac{\mathbf{p}_{1x} + \mathbf{p}_{2x} + \mathbf{p}_{3x}}{3} \tag{3.28}$$

$$\mathbf{c}_y = \frac{\mathbf{p}_{1y} + \mathbf{p}_{2y} + \mathbf{p}_{3y}}{3} \tag{3.29}$$

3.10 Estrutura de Dados

3.10.1 Introdução

A maneira como representamos os dados em um sistema computacional de qualquer natureza determina a complexidade e a eficiência das operações realizadas por algoritmos associados a ele. Em vista disso a escolha de uma estrutura de dados adequada é necessária para suportar o conjunto de operações que caracterizam as funcionalidades do sistema, o seu sucesso e conseqüentemente o seu desempenho.

Existem diferentes estruturas de dados estáticas que são capazes de representar a topologia de objetos geométricos, e consequentemente possuem todos os elementos necessários para que seja feita uma análise sobre tais objetos, porém o pré-processamento e pós-processamento sobre essas estruturas, torna-se muito custoso, computacionalmente falando, além do fato de que em uma triangulação não saberemos a priori quantos elementos(*vértices*, *arestas*,*faces*) a malha conterá ao todo.

O uso de *estruturas de dados dinâmicas* possibilita reservar novas áreas de memória em tempo de execução, de acordo com a necessidade de uma aplicação. O advento de *malhas adaptativas* realçou ainda mais a fragilidade patente nos esquemas de representação baseados em estruturas estáticas.

3.10.2 Fundamentação

Primeiramente traremos alguns conceitos teóricos que fornecerão o rigor necessário para fundamentar as demais seções do trabalho.

Um grafo $\mathbf{G}(V, E)$ é um conjunto finito não-vazio V de *vértices*, e um conjunto E de *aretas*, que são formadas a partir de pares não-ordenados de vértices distintos de V. Uma aresta $\mathbf{e} = (\mathbf{u}, \mathbf{v})$ é constituida pelos vértices $\mathbf{u}$ e $\mathbf{v}$, que são seus extremos.

Todo grafo admite uma *representação geométrica* no plano, de modo que cada vértice corresponda a um ponto distinto nesse plano numa posição qualquer, enquanto cada aresta $\mathbf{e} = (\mathbf{u}, \mathbf{v})$ corresponde a uma curva unindo os pontos associados. Assim, ao se referir a um grafo, faz-se igualmente uma referência à sua representação.

Por outro lado, um *grafo* $\mathbf{G}$ é imersível numa dada superfície $\mathbf{S}$ se existir uma representação geométrica $\mathbf{R}$ em $\mathbf{S}$, de tal forma que duas curvas quaisquer de $\mathbf{R}$ não se interceptem a não ser em seus extremos. Dessa maneira, um grafo é dito *planar* se ele admite uma representação $\mathbf{R}$ no plano. Assim, uma aresta arbitrária poderá sempre ser representada em $\mathbf{R}$ através de um segmento de reta. Ao ser desenhado no plano, os segmentos de $\mathbf{R}$ que representam arestas em $\mathbf{G}$ decompõem o plano em regiões que são referenciadas por *faces*. Dessa forma a representação $\mathbf{R}$ de todo *grafo planar* induz no plano uma partição que é conhecida por *subdivisão planar*.

O número de *vétices*, *arestas* e *faces* em qualquer subdivisão planar, dados respectivamente pr nv, ne e nf, se relacionam combinatoriamente através da relação de Euler dada pela equação 3.30.

$$nv + nf - ne = 2 \tag{3.30}$$

Num *grafo planar* cada face é formada por, no mínimo, três arestas, cada aresta possui exatamente dois vértices e é adjacente a exatamente duas faces. Logo, $2ne \geq 3nf$, e, ao se substituir esta desigualdade na equação 3.30, obtém-se as inequações 3.31, 3.32 e 3.33, que são sempre válidas. Se for acrescentada a restrição de que cada *vértice* possui pelo menos três *arestas* incidentes, então,

as inequações 3.34, 3.35 e 3.36 também são válidas.

Um *grafo* $\mathbf{G}(V, E)$ é dito *conexo* quando existe uma sequência de arestas entre dois vértices quaisquer de V. Dado um conjunto $\mathbf{S}$ e um subconjunto $\mathbf{S}'(\mathbf{S}' \subseteq \mathbf{S})$, diz-se que $\mathbf{S}'$ é *maximal* em relação a uma determinada propriedade $\mathbf{M}$ quando $\mathbf{S}'$ satisfaz a propriedade $\mathbf{M}$ e não existe outro subconjunto $\mathbf{S}''$ que contém $\mathbf{S}'$ e também satisfaz a propriedade $\mathbf{M}$. Muitos outros resultados teóricos sobre teoria dos grafos podem ser encontrados em [32].

$$ne \leq 3nv - 6 \tag{3.31}$$

$$nf \leq \frac{2}{3}ne \tag{3.32}$$

$$nf \leq 2nv - 4 \tag{3.33}$$

$$nv \leq \frac{2}{3}ne \tag{3.34}$$

$$ne \leq 3nf - 6 \tag{3.35}$$

$$nv \leq 2nf - 4 \tag{3.36}$$

Seja $V = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\} \subseteq R^2$, $n \geq 3$, um conjunto finito de pontos distintos e não todos colineares e E o conjunto de todos os segmentos possíveis entre vértices de V. Formalmente, uma *triangulação* de V é uma grafo planar $\mathbf{G}(V, E')$, onde E' é um *subconjunto maximal* de E, com a propriedade de que duas arestas quaisquer de E' não se interceptam, a não ser em seus vértices [20]. Nessas condições, todas as faces limitadas da representação geométrica de $\mathbf{G}(V, E')$ são triangulares e se referenciará a elas simplesmente por triângulos, ou ainda, elementos triangulares.

3.10.3 Estrutura de Dados baseada em Fronteira

Os três tipos de objetos *face*, *aresta* e *vértice*, possuem informações associadas a eles que formam uma componente básica definida como *modelo de fronteira*. Na

coleta das informações geométricas tanto das *faces*, equações de curva e coordenadas do *vértice*, o modelo de fronteira deve representar como as *faces*, *arestas* e *vértices* estão relacionados entre si. Para encapsular toda a informação das entidades sob a geometria de um modelo de fronteiras, é comum usarmos as informações de suas interconectividades sob o termo *topologia*.

Todos os modelos de fronteiras representam *faces* em termos de nós explícitos numa estrutura de dados de fronteiras. Isso, possibilita muitas alternativas para representação da geometria e da topologia de um modelo de fronteiras, algum dos quais são descritos a seguir. Para deixar mais simples e mais claro, devemos ilustrá-los baseados na representação do bloco mostrado na figura 3.11.

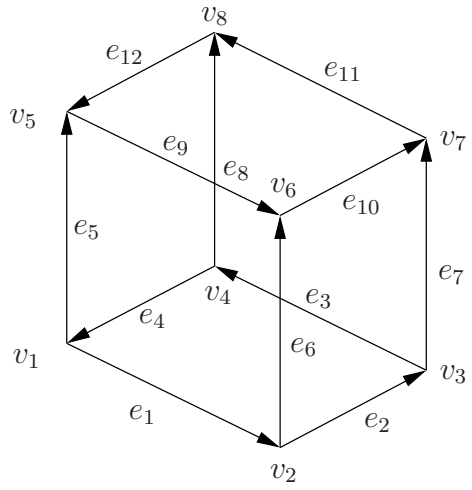


Figura 3.11: Orientação das *arestas* relacionada com o *loop* exterior do objeto.

Modelo de fronteira baseado em polígonos.

Um modelo de fronteira que tem somente *faces* planas é chamado de *modelo poliedral*. Pelo fato de todas as *arestas* do poliedro serem segmentos de linhas retas, é possível contrair a estrutura de dados de fronteira de forma considerável.

A simples variação das *faces* estão representadas como polígonos, cada polígono consiste numa sequência de três coordenadas. Um sólido consiste na coleção de *faces* agrupadas em termos de, digamos, uma tabela de identificadores de *faces* ou o equivalente, como uma lista encadeada com nós das *faces*. Algumas vezes as informação agrupada das *faces* são eliminadas e o relacionamento entre elas ficam implícitos. Esse modo de representação é utilizada em *metafiles* gráficos em sistemas gráficos.

Modelo de fronteiras baseada em *vértice*.

Num modelo de fronteira baseado em polígonos as coordenadas do *vértice* aparecem tanto quanto os próprios *vértices* aparecem numa *face*. Essa redundância pode ser eliminada introduzindo os *vértices* como entidades independentes da estrutura de dados de fronteira. Nesse caso, os identificadores dos *vértices* (ou os ponteiros para os nós dos *vértices*) estão associados com as *faces*. Essa aproximação nos leva a vários modelos de fronteiras baseada em *vértices*.

Note que a simples representação da figura 3.12 lista os *vértices* de cada *face* em uma ordem consistente (sentido horário) como o visto do lado de fora do cubo. Essa orientação robusta é útil em muitos algoritmos. Por exemplo, em linhas ocultas ou numa superfície remota, permite a eliminação de *faces traseiras* na base das *faces* normais através dos ponteiros consistentemente apontando para elas. No caso da figura 3.12 as *faces* f_1, f_4 e f_5 podem ser imediatamenete descartadas pelas linhas ocultas e um algoritmo de remoção de superfície.

A representação da figura 3.12 não inclui todas informações da superfície; como todas as *faces* são planares, suas geometrias são completamente definidas através das coordenadas dos *vértices*. Por outro lado, se a equação da *face* é necessária para cálculo numérico (digamos, para sombreamento) essa informação também podem ser incorporadas junto às *faces*.

Em geral, muitas escolhas, sobre quais informações devem ser mantidas explicitas e quais deverão estar implícitamente (isto é, necessárias para ser calculadas) tem que ser definidas ao implementarmos um modelo de fronteiras. Na figura 3.12, a inclusão explícita das coordenadas dos *vértices* podem também dar-nos as equações das *faces* (a aproximação oposta, de guardar as equações das *faces* e obter as coordenadas do *vértice* leva a um modelo de *half-space* apropriados para objetos convexos somente).

Se algumas informações geométricas redundantes estão guardadas (como as equações das *faces* no exemplo acima), alguns problemas numéricos sutís podem surgir. Suponhamos que devido a pequenas imprecisões nos cálculos de pontos flutuantes, os *vértices* de uma *face* não fique exatamente no plano definido pela equação da *face*. Quais informações seriam corretas?

Modelo de fronteira baseada em *arestas*

Se superfícies curvas estão presentes num modelo de fronteiras, torna-se útil incluir também os nós das *arestas* explicitamente numa estrutura de dados de

vertice	*coordenadas*
v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

face	*vertices*
f_1	$v_1\ v_2\ v_3\ v_4$
f_2	$v_6\ v_2\ v_1\ v_5$
f_3	$v_7\ v_3\ v_2\ v_6$
f_4	$v_8\ v_4\ v_3\ v_7$
f_5	$v_5\ v_1\ v_4\ v_8$
f_6	$v_8\ v_7\ v_6\ v_5$

Figura 3.12: Modelo de fronteira baseado em *vértice* segundo o modelo 3.11.

fronteiras para ser capaz de guardar as informações de intersecção das *faces* curvas (Os nós das *arestas* também são úteis para registrar relacionamentos topológicos).

Um modelo de fronteiras baseado em *arestas* representa uma fronteira de *face* em uma sequência fechada de *arestas*, ou por um pequeno *loop*. Os *vértices* das *faces* estão representados através das *arestas*. Essa aproximação nos leva ao modelo da figura 3.13.

A estrutura de dados indica uma orientação para cada *aresta*, digamos, a *aresta* $\mathbf{e}_1$ é considerada orientada positivamente do *vértice* $\mathbf{v}_1$ para $\mathbf{v}_2$ e novamente as *faces* estão consistentemente orientadas, isto é, suas *arestas* estão dispostas no sentido horário como o visto do lado exterior do bloco. Note que cada *aresta* aparece em duas *faces*, uma na orientação positiva e outra na negativa.

aresta	*vertices*
e_1	$v_1\ v_2$
e_2	$v_2\ v_3$
e_3	$v_3\ v_4$
e_4	$v_4\ v_1$
e_5	$v_1\ v_5$
e_6	$v_2\ v_6$
e_7	$v_3\ v_7$
e_8	$v_4\ v_8$
e_9	$v_5\ v_6$
e_{10}	$v_6\ v_7$
e_{11}	$v_7\ v_8$
e_{12}	$v_8\ v_5$

vertice	*coordenadas*
v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

face	*arestas*
f_1	$e_1\ e_2\ e_3\ e_4$
f_2	$e_9\ e_6\ e_1\ e_5$
f_3	$e_{10}e_7\ e_2\ e_6$
f_4	$e_{11}e_8\ e_3\ e_7$
f_5	$e_{12}e_5\ e_4\ e_8$
f_6	$e_{12}e_{11}e_{10}e_9$

Figura 3.13: Modelo de fronteira baseado em *aresta* de acordo com a figura 3.11.

A estrutura de dados *winged-edge*: A inclusão explícita de nós para cada tipo de entidade (*face*, *aresta* e *vértice*) abre as portas para elaboração de um modelo de fronteiras baseada em *arestas* mais completo. Por exemplo, para algoritmos que ocultam removem e sombreiam superfície, a informação da vizinhança *face-face* pode ser adicionado à estrutura de dados da figura 3.13, associando *arestas* com identificadores de duas *faces* separadas.

A tão comentada *winged-edge* dá o exemplo inicial da maneira de como elaborar um modelo baseado em *arestas*. Dando um passo mais longe por representar também as informações do *loop* nos nós das *arestas*.

Pelo fato de cada *aresta* **e** aparecer em duas *faces*, mais duas outras $\mathbf{e}'$ e $\mathbf{e}''$ aparecem depois de **e** nessas *faces*. Além disso, pelo fato da orientação consistente das *faces*, **e** fica uma vez numa orientação positiva e outra vez numa orientação negativa.

A estrutura *winged-edge* tem a vantagem dessas propriedades estruturais pelo fato de associar os identificadores das próximas e duas *arestas* com um nó da *aresta*. Convencionalmente esses dados são denotados pela *ncw* e pela *nccw* a próxima *aresta* na outra *face*.

aresta	*vstart*	*vend*	*ncw*	*nccw*
e_1	v_1	v_2	e_2	e_5
e_2	v_2	v_3	e_3	e_6
e_3	v_3	v_4	e_4	e_7
e_4	v_4	v_1	e_1	e_8
e_5	v_1	v_5	e_9	e_4
e_6	v_2	v_6	e_{10}	e_1
e_7	v_3	v_7	e_{11}	e_2
e_8	v_1	v_8	e_{12}	e_3
e_9	v_5	v_6	e_6	e_{12}
e_{10}	v_6	v_7	e_7	e_9
e_{11}	v_7	v_8	e_8	e_{10}
e_{12}	v_8	v_5	e_5	e_{11}

vertice	*coordenadas*	*face*	*first edge*	*sinal*
v_1	$x_1\ y_1\ z_1$	f_1	e_1	$+$
v_2	$x_2\ y_2\ z_2$	f_2	e_9	$+$
v_3	$x_3\ y_3\ z_3$	f_3	e_6	$+$
v_4	$x_4\ y_4\ z_4$	f_4	e_7	$+$
v_5	$x_5\ y_5\ z_5$	f_5	e_{12}	$+$
v_6	$x_6\ y_6\ z_6$	f_6	e_9	$-$
v_7	$x_7\ y_7\ z_7$			
v_8	$x_8\ y_8\ z_8$			

Figura 3.14: A estrutura de dados *winged-edge* seguindo a figura 3.11.

Por conveniência da representação por *aresta*, as *faces* somente precisam incluir os identificadores de uma *aresta* arbitrária com um bit que indica a orientação. A figura 3.14 dá um exemplo da *winged-edge*. Os sinais + e - denotam a orientação da *aresta*.

Começando pela *aresta* diretamente associada ao *loop*, todas *arestas* podem ser corrigidas segundo os ponteiros da *ncw* e *nccw*. Por exemplo, no caso da figura 3.14, a fronteira da *face* f_5 é extraída iniciando de $\mathbf{e}_{12}$. A próxima *aresta* é dada por $ncw(\mathbf{e}_{12}) = \mathbf{e}_5$. Pelo fato de $\mathbf{e}_5$ ser passeado na sua orientação negativa (que podem ser observados nos identificadores dos *vértices*), a próxima *aresta* é dada por $nccw(\mathbf{e}_5) = \mathbf{e}_4$. Do mesmo modo pegamos nccw($\mathbf{e}_4$) = $\mathbf{e}_8$ que

é a próxima *aresta*. Passeando pela orientação positiva, a próxima *aresta* é $\text{ncw}(\mathbf{e}_8) = \mathbf{e}_{12}$ novamente, assim todas *arestas* da *face* que será extraída são plenamente conhecidas.

No caso mais geral, os nós das *arestas* da estrutura *winged-edge* também incluem os identificadores *fcw* e *fccw* da *face* vizinha, então analogamente a *ncw* e *nccw*, os identificadores *pcw*, *pccw* das *arestas* anteriores nessas *faces*. Os indicadores da orientação na figura 3.14 se tornam redundantes. O resultado completo da *winged-edge* está ilustrado na figura 3.15.

Pelo fato da *winged-edge completa* incluir os identificadores de uma *aresta* adjacente para cada nó do *vértice*, todas *arestas* que se encontram no *vértice* podem ser removidas por uma algoritmo parecido com o de remoção de *loop*.

A estrutura *winged-edge* e suas variantes não estão restringidas a representação de superfícies de objetos tridimensionais. Em particular, essas representações podem ser aplicadas em qualquer dimensão. A chave para aplicação é o caráter cíclico de vizinhança de qualquer *vértice* ou *face* em uma topologia que pode ser representado usando uma lista simplesmente encadeada com as primitivas "nccw" e "pccw".

Faces com muitas fronteiras

A estrutura de dados descritas até agora leva em conta que cada *face* está simplesmente conectadas, isto é, é apenas uma curva de fronteira. Porém, algumas vezes parece conveniente fazer a inclusão de *faces* que têm muitas fronteiras, ver figura 3.15.

Duas alternativas são apresentadas. A primeira se aproxima do modelo de *faces* com fronteiras complexas, conectando o segmento de fronteira por meio de *arestas* auxiliares. Infelizmente, nesses casos especiais as *arestas* devem ser marcadas como vazias quando as linhas forem desenhadas. Contudo, na estrutura *winged-edge* isso não é necessário porque as *arestas* auxiliares são tratadas para aparecer duas vezes na mesma *face* e podem ser reveladas comparando-se os dados das *fcw* e *fccw*.

O outro modo é adicionarmos um *loop* separando que modela uma fronteira simples no modelo de fronteira, e associa cada *face* com uma lista de *loops*. As *faces* referenciadas podem ser subtituídas pelos *loops* referenciados associando a eles, por exemplo, os indicadores de *face fcw* e *fccw* nos nós das *arestas* da *winged-edge* completa que podem ser trocados com os indicadores de *loop* análogo, *lcw* e *lccw*.

aresta	*vstart*	*vend*	fcw	$fccw$	*ncw*	*pcw*	*nccw*	*pccw*
e_1	v_1	v_2	f_1	f_2	e_2	e_4	e_5	e_6
e_2	v_2	v_3	f_1	f_3	e_3	e_1	e_6	e_7
e_3	v_3	v_4	f_1	f_4	e_4	e_2	e_7	e_8
e_4	v_4	v_1	f_1	f_5	e_1	e_3	e_8	e_5
e_5	v_1	v_5	f_2	f_5	e_9	e_1	e_4	e_{12}
e_6	v_2	v_6	f_3	f_2	e_{10}	e_2	e_1	e_9
e_7	v_3	v_7	f_4	f_3	e_{11}	e_3	e_2	e_{10}
e_8	v_1	v_8	f_5	f_4	e_{12}	e_4	e_3	e_{11}
e_9	v_5	v_6	f_2	f_6	e_6	e_5	e_{12}	e_{10}
e_{10}	v_6	v_7	f_3	f_6	e_7	e_6	e_9	e_{11}
e_{11}	v_7	v_8	f_4	f_6	e_8	e_7	e_{10}	e_{12}
e_{12}	v_8	v_5	f_5	f_6	e_5	e_8	e_{11}	e_9

vertice	$first\ edge$	*coordenadas*
v_1	e_1	$x_1\ y_1\ z_1$
v_2	e_2	$x_2\ y_2\ z_2$
v_3	e_3	$x_3\ y_3\ z_3$
v_4	e_4	$x_4\ y_4\ z_4$
v_5	e_9	$x_5\ y_5\ z_5$
v_6	e_{10}	$x_6\ y_6\ z_6$
v_7	e_{11}	$x_7\ y_7\ z_7$
v_8	e_{12}	$x_8\ y_8\ z_8$

$face$	$first\ edge$
f_1	e_1
f_2	e_9
f_3	e_6
f_4	e_7
f_5	e_{12}
f_6	e_9

Figura 3.15: A estrutura de dados *winged-edge* completa com relação a figura 3.11.

Estrutura de Dados Topológicas

Essa estrutura (descrita em detalhes na seção anterior) e o conceito de estruturas topológicas foi desenvolvida originalmente por Baumgart [1, 2]em Stanford, no contexto da *Visão Computacional* para representação da topologia de superfície de modelos poliedrais.

Uma das principais utilidades dessa estrutura é representar polígonos e auxiliar em pesquisas na área de *Robótica*, *Modelagem Geométrica* e *Geometria Computacional*, e seu uso para *malha de elementos finitos* é muito recente.

Uma estrutura de dados topológica permite extrair do modelo representado todas as relações de adjacência entre as diferentes entidades topológicas presentes no modelo em tempo constante ou linearmente proporcional à incidência da entidade envolvida, ou seja, de forma independente da dimensão do problema, na medida em que todas as operações são realizadas localmente.

Um importante avanço nessa estrutura foi a introdução dos *Operadores de Euler*, permitindo um alto grau de abstração com a utilização de cinco operadores possibilitando a criação e manipulação de várias componentes do modelo.

O principal objetivo aqui é realçar a *winged-edge*, considerada a pioneira das estruturas de dados topológicas. Ela é contruída sobre a componente *aresta*, as informações mais relevantes dessa estrutura estão concentradas nessa entidade, por isso é freqüentemente referenciada nas literaturas como uma estrutura de dados topológica baseada em *arestas* assim como a *half-edge*. Existem outras estruturas nas quais a componente *face* é mais relevante.

O fato da concentração das informações na *aresta* tem um papel importante. Pois ao analizarmos uma *subdivisão planar* qualquer, notamos que o número de *arestas* e *faces* incidentes num *vértice* podem variar, isso vale também para o número de *arestas* e *vértices* em uma *face* qualquer, no entanto o número de *faces* incidentes numa *aresta* é invariante.

É importante ressaltar que a *windge-edge* não representa uma *face* com mais de uma componente conexa. Isso até seria possível com a introdução de "falsas" *arestas* ligando os *ciclos exteriores* aos *ciclos interiores* Figura 3.16, mas tomando certos cuidados ao nível de implementação. No entanto com a introdução da entidade *ciclo* inventada por Braid[5] em 1980, pode-se representar e manipular *faces* com mais de uma componente conexa[1]. Por conseguinte, poderemos manipular situações que ocorrem em polígonos com

[1]mais de uma compenente conexa trata-se da existência de buracos(holes) fazendo parte da malha de elementos triangulares

gênero[2] maior que zero e situações irreais fisicamente, como o que acontece em sistemas que trabalham com o processo de modelagem. Desde então a estrutura passou a ser referenciada como *winged-edge modificada*. Seu diagrama está representado na Figura 3.19. Não fazemos o tipo de manipulação com a malha possuindo mais de uma componente conexa, porém é possível fazer adaptações em implementar em projetos futuros para esse tipo de tratamento.

Em uma *subdivisão planar* as *faces* encontram se consistentemente orientadas: as *faces* interiores no sentido anti-horário (CounterClockWise) enquanto as exteriores no sentido horário (ClockWise), assim para percorrer as *faces*, visitamos cada *aresta* duas vezes, uma no sentido positivo $\mathbf{v}_1$ para $\mathbf{v}_2$ e outra no sentido contrário. Figura 3.17.

A estrutura da *face*, qualquer um dos vários *loops* são guardados visando manter sua consistência. A estrutura do *loop* que pentencem à mesma *face* estão encadeados em um único sentido, e na estrutura da *aresta*, ao invés de *face*, existem ponteiros para os dois *loops*.

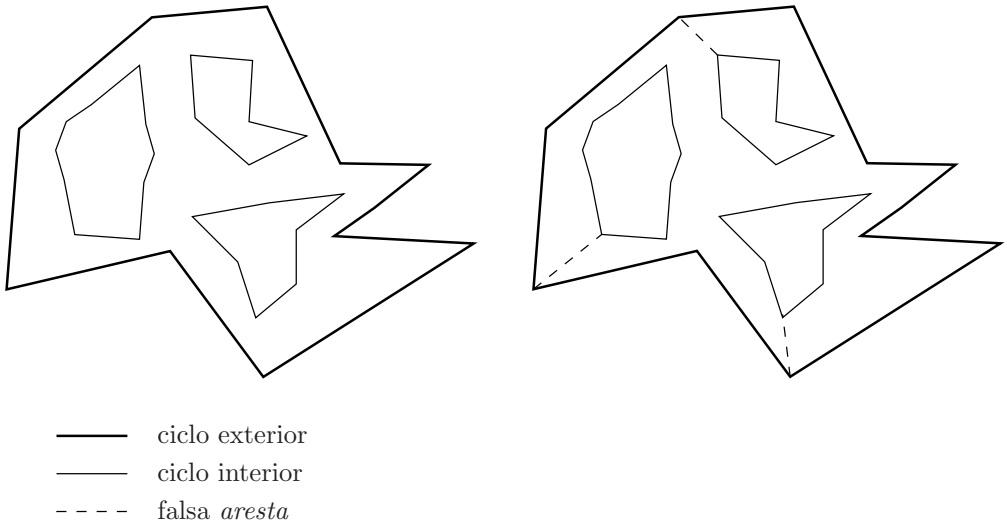


Figura 3.16: Subdivisão planar arbitrária, onde a *face* interior possui três ciclos internos e o domínio e formado por 4 componentes conexas.

Na orientação de sentido positivo haverá duas *arestas*, **pccw** e **nccw**(ciclo1), que serão visitadas antes e depois de **e** no sentido anti-horário. E ao percorrer a mesma *aresta* no sentido negativo (ciclo 2) serão visitadas as *arestas* **pcw** e **ncw**, que são as *arestas* posterior e anterior respectivamente no sentido horário. Essa orientação pode ser visualizada na figura 3.18

[2]O conceito de gênero pode ser entendido como o número de buracos que perfuram completamente a superfície.

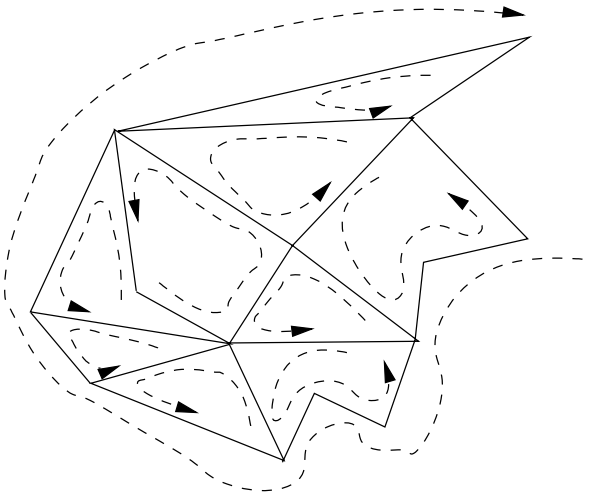


Figura 3.17: Subdivisão planar arbitrária, com todas as *faces* consistentemente orientadas.

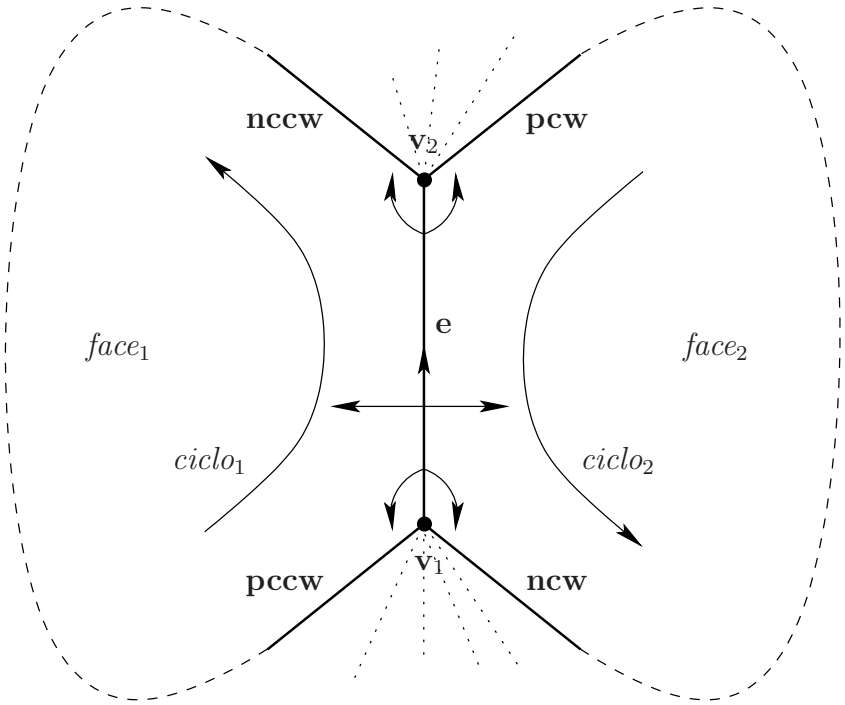


Figura 3.18: Esquema da estrutura de dados *winged-edge modificada* para uma *aresta* qualquer.

Na representação completa da *winged-edge modificada*, essas 4 *arestas* são representadas explicitamente, bem como seus *vértices* extremos $\mathbf{v}_1$ e $\mathbf{v}_2$, mais o $ciclo_1$ e $ciclo_2$. A rigor, somente duas *arestas*, por exemplo, **nccw** e **ncw**, seriam suficientes para se obter todas as informações desejadas, mas isto acarreta um custo adicional em relação ao desempenho de algumas operações, visto que o percurso pelos *ciclos* seria realizado em um único sentido.

Uma vantagem na utilização da topologia é conseguir independência geométrica ou seja a estrutura de dados que não depende da geometria (reta, grau 2, grau 3) de uma *aresta*. Assim poderemos adicionar novas geometrias ao sistema sem necessidade de alterar o código existente apenas escrevendo os procedimentos para a geometria relativa. Outro fator importante, fazendo implementações baseada na topologia, é conseguir representar operações de natureza geométrica com o máximo de robustez e eficiência para o sistema.

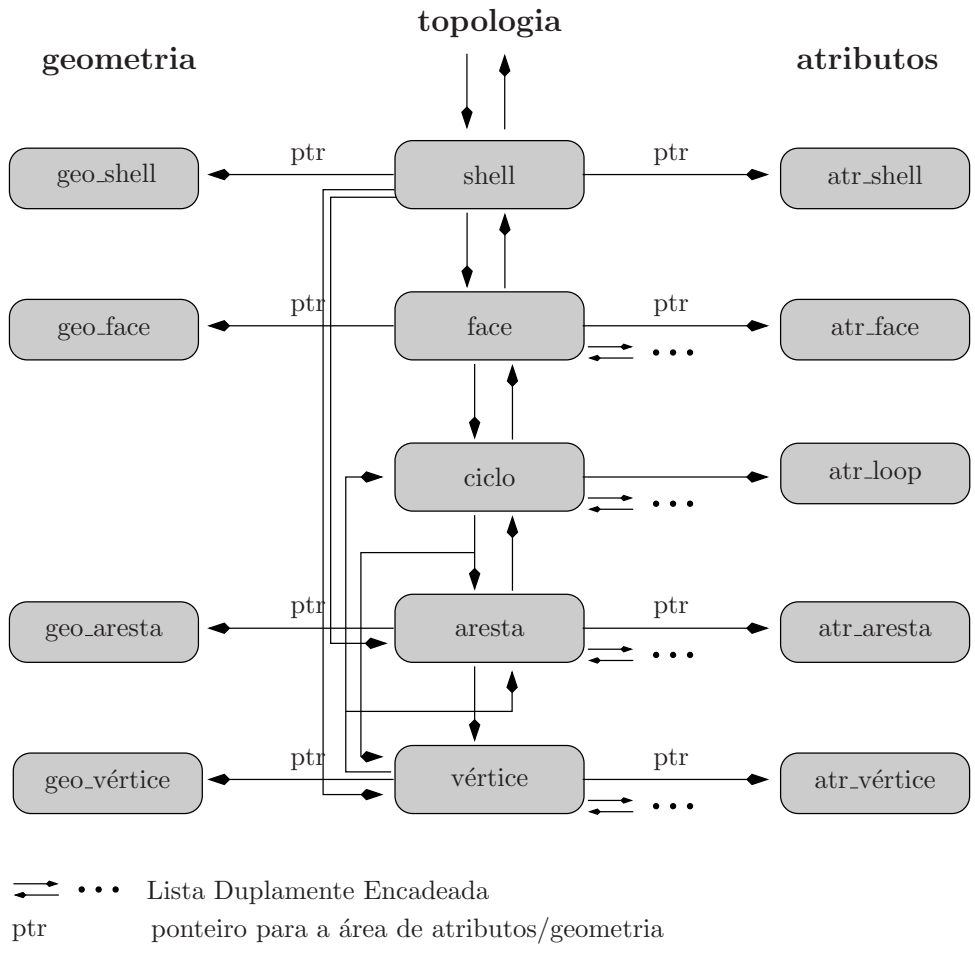


Figura 3.19: **Modelo hierárquico topológico** associado à estrutura *winged-edge modificada*.

3.11 Primitivas Topológicas

3.11.1 Operadores de Euler

As operações em *subdivisão planar* são feitas através de operadores topológicos denominados *Operadores de Euler* em Baumgart[1], esses operadores mantêm a consistência na contrução de *malha de elementos finitos* Piteri[26], e envolvem a complexidade de ordem geométrica e topológica subjacente à criação e manutenção.

Esses operadores obedecem à formula de Euler 3.30, e são utilizados visando manter e também modificar a estrutura de um polígono representado apenas mudando sua topologia, isto é, em qualquer polígono o número de *faces*(f), *arestas* (e) e *vé rtices*(v) são relacionados de acordo com a fórmula.

Verificávamos que a consistência topológica da *malha de elementos finitos* era considerada responsável por altos custos em termos de dados de entrada, devido à forma mecânica como era realizada.

De acordo com Braid [5] cinco operadores topológicos são necessários para manipulação dos modelos, porém para maior prática no desenvolvimento de um sistema, mais operadores podem ser implementados. As seis entidades envolvidas no modelo planar utilizado são: *vértices*, *arestas*, *faces*, *ciclos*(*loops*), e corpos ou shell que são baseados na estrutura de dados *winged-edge modificada*. Em nossas implementação utilizamos primitivas topológicas para executar atualizações durante o processo construtivo da malha e garantindo sua consistência e robustez, o uso desses operadores são melhor explicados em [26].

3.11.2 As Relações entre as Entidades Topológicas

Há relações de adjacência entre as entidades topológicas *vértice*, *aresta* e *face* encontradas em uma *subdivisão planar*, a Figura 3.20 representa essa situação. Cada uma dessas relações é representada por um par de letras, em que a segunda identifica o conjunto de entidades incidentes ou adjacentes à primeira. Assim, **VF** e **FF** significam, respectivamente, o conjunto de *faces* incidentes a um *vértice* dado e o conjunto de *faces* adjacentes a uma *face* dada.

Segue as nove relações de adjacência possíveis em toda estrutura topológica:

- **VV**: dado um *vértice*, encontrar os *vértices* adjacentes a ele. Dois *vértices* são adjacentes se estão conectados por uma *aresta*;

- **VE**: dado um *vértice*, encontrar as *arestas* adjacentes. Uma *aresta* é adjacente a um *vértice* se o *vértice* é seu extremo;
- **VF**: dado um *vértice*, encontrar o *ciclo* de *faces* adjacentes a ele. Face é adjacente se o *vértice* está contido na *face* com um único *loop* ou compartilha no mínimo uma das *arestas* adjacente ao *vértice*;
- **EV**: dada uma *aresta*, obter dois *vértices* conectados pelas *arestas*;
- **EE**: dada uma *aresta*, encontrar as *arestas* adjacentes a cada *vértice* adjacente;
- **EF**: dada uma *aresta*, encontre duas *faces* que a compartilham;
- **FV**: dada uma *face*, encontre os *vértices* que representam os *loops* das *faces*;
- **FE**: dada uma *face*, obtenha as *arestas* que representam os *loops* das *faces*;
- **FF**: dada uma *face*, encontrar as *faces* adjacentes à elas. Duas *faces* são adjacentes se compartilham uma *aresta*.

Todas as nove relações podem ser obtidas de forma muito simples na estrutura *winged-edge modificada*, que foi a adotada na implementação, com o auxílio de apenas duas primitivas topológicas, descritas abaixo ilustradas na Figura 3.23.

ccw_nev(v, e, ne) Dado um *vértice* **v** e uma *aresta* **e**, esta primitiva retorna a *aresta* **ne**, que é a próxima *aresta* incidente a **v** no sentido anti-horário depois de **e**.

ccw_nel(l, e, ne) Dado um *ciclo* **l** e uma de suas *arestas* **e**, esta primitiva retorna a *aresta* **ne**, que é a *aresta* seguinte a **e** quando se percorre o *ciclo* **l** no sentido anti-horário.

A seguir, apresentam-se formalmente todas as componentes no modelo hierárquico topológico:

- **vértice:** representa um simplexo de dimensão zero, que é um ponto no espaço euclidiano.
- **aresta:** representa um segmento de curva limitado por dois *vértices*, não necessariamente distintos, e *homeomorfo* ao intervalo [0,1]. Na implementação que estamos realizando, uma *aresta* será sempre um segmento de reta, podendo pertencer a um ou no máximo dois *ciclos*.

- **ciclo:** é um conjunto conexo, fechado e ordenado de *vértices* e *arestas* alternadamente, sem auto-interseções e que define a fronteira de uma *face*. Em casos degenerados pode ser constituído por uma *face* com um único *vértice*.

- **face:** corresponde a uma área finita de uma superfície orientável, que neste contexto será sempre o plano euclidiano. É definida em termos de seus *ciclos* e toda *face* possui pelo menos um; os outros se existirem, definem "buracos" na *face*.

- **shell:** é composta por um conjunto finito de *faces* conectadas, consistentemente orientadas e que representam a fronteira fechada de uma área. Em situações de singularidades, uma *shell* pode ser formada por um único *vértice*. A rigor, não precisariamos desta entidade, pois trabalhamos apenas com a versão 2D, mas a utilizamos, pois é útil no sentido de introduzir mais um nível de abstração no modelo.

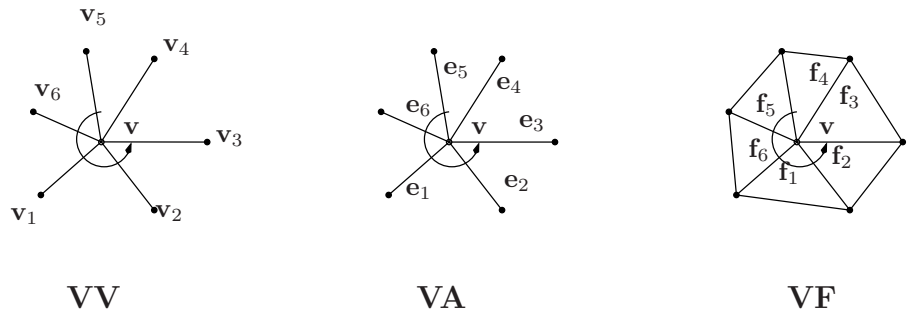


Figura 3.20: Relações de adjacências entre *aresta*, *vértice* e *face*, entidades presentes na estrutura.

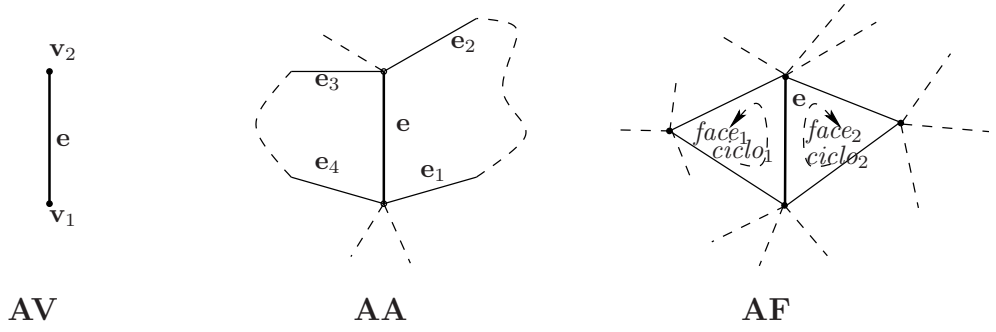


Figura 3.21: Relações de adjacências entre *aresta*, *vértice* e *face*, entidades presentes na estrutura de *diagrama planar*.

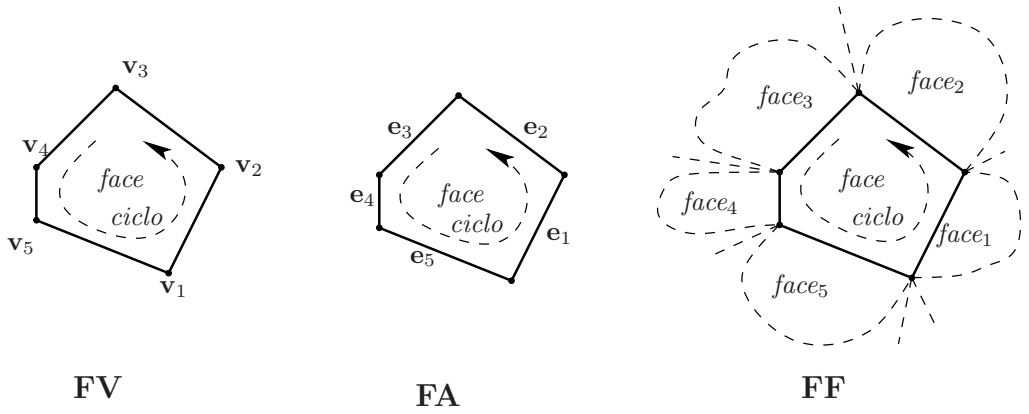


Figura 3.22: Relações de adjacências entre *aresta*, *vértice* e *face*, entidades presentes na estrutura de *diagrama planar*.

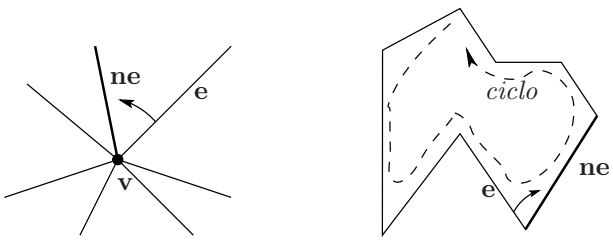


Figura 3.23: Primitivas utilizadas para se obter todas as relações de adjacências entre as entidades *aresta*,*vértice* e *face*.

- **modelo:** no momento da edição de um domínio em particular, é constituído apenas pelo nível geométrico, definido por um conjunto de curvas. Quando há a conversão para um nível topológico, o modelo passa a ser representado por uma única *shell*, que é composta por uma única *face*, definida por um número arbitrário de *ciclos.*

Com essa sistemática seremos capazes de desenvolver algoritmos eficientes de intersecção, visualização entre outros, além de aumentar o nível de abstração na descrição dos objetos.

Cada uma das entidades no modelo é representado em **C** por uma variável estruturada do tipo *struct*. Toda a informação relativa aos atributos e as de natureza geometrica correspondem a apenas um componente da topologia. Figura 3.19.

Ao se trabalhar com *malhas de elementos triangulares* é preciso considerar a necessidade de constantes atualizações sobre a *malha*, principalmente ao executar operações de inserção e remoção das diferentes entidades. Nesse sentido as entidades são representadas utilizando listas duplamente encadeadas.

Em nosso sistema foram implementadas primitivas topológicas que são responsáveis pelas constantes atualizações da topologia da malha no passo construtivo. As características mencionadas da *winged-edge modificada* facilitaram a implementação garantindo a consistência de toda arquitetura do sistema.

3.11.3 GG - Grabriel graph

O grafo de Gabriel é um grafo $\mathbf{G}(V, E)$ que conecta um conjunto de pontos no plano euclidiano. Dois pontos $\mathbf{p}_1$ e $\mathbf{p}_2$ são conectados por uma aresta no grafo de Gabriel se o círculo contendo o diâmetro $\overline{\mathbf{p}_1\mathbf{p}_2}$ não possui nenhum outro ponto no seu interior figura 3.24 a), caso haja um terceiro ponto no interior da circunferência figura 3.24 b), a aresta não fará parte do grafo de Gabriel. Em qualquer outra dimensão o grafo de Gabriel conecta dois pontos formando o diâmetro de uma esfera vazia. O grafo de Gabriel foi apresentada por K. Ruben Gabriel e intitulada por Robert R. Sokal em 1969 [30]. Esse grafo é um sub-grafo da Triangulação de Delaunay e com a triangulação dada pode ser obtido em tempo linear [23]. Esse gráfico contém a MST e o RNG na instância de um esqueleto-beta. Sua aplicabilidade também se encontra na área de redes, na montagem de redes wireless. Um exemplo de grafo pode ser visto na figura 3.25.

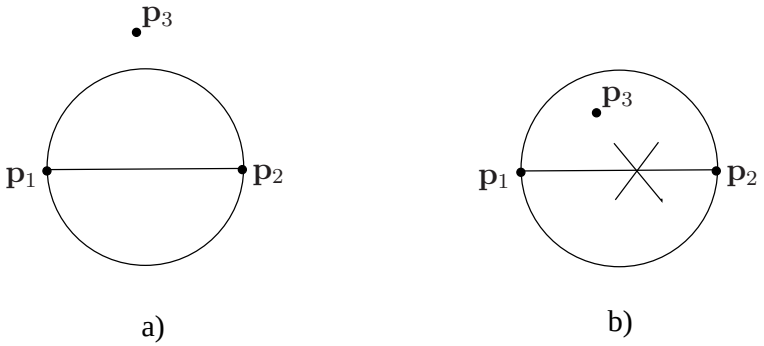


Figura 3.24: a) circunferência que contém o diâmetro de $\overline{\mathbf{p}_1\mathbf{p}_2}$, com um terceiro ponto $\mathbf{p}_3$ no exterior da circunferência. b) circunferência que contém o diâmetro de $\overline{\mathbf{p}_1\mathbf{p}_2}$, porém com um terceiro ponto $\mathbf{p}_3$ no interior da circunferência

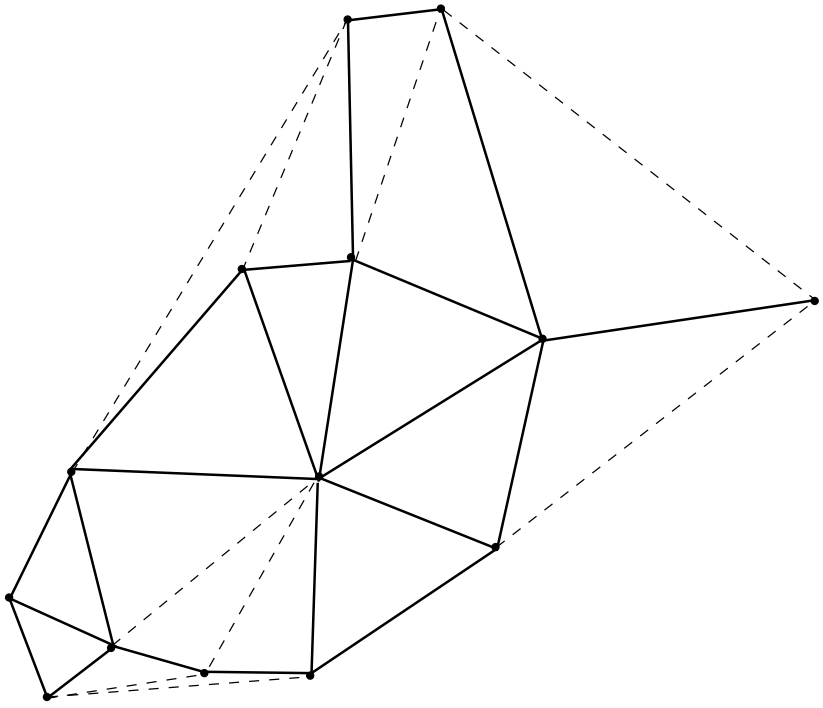


Figura 3.25: Gabriel Graph de um determinado conjunto de pontos em destaque sobre a Triangulação de Delaunay - linha pontilhadas

3.11.4 MST - Minimum Spanning Tree

A árvore de espalhamento mínimo trata-se de um sub-grafo da Triangulação de Delaunay e está contida no Grafo de Gabriel, que não possui direção e conecta todos os vértices da triangulação através de arestas. Um único grafo pode ter várias MST se acontecer das arestas que foram marcada como tendo algum peso possuirem pesos iguais, contudo se todas arestas possuirem pesos diferentes, é garantido que haverá somente uma MST associada à Triangulação de Delaunay.

A aplicação imediata da MST é em redes, no qual podemos citar a conecção entre os roteadores externos que podem ter um custo para transferência dos dados e o que se deseja é que esse custo seja o mais otimizado possível. Com a Triangulação de Delaunay já obtida o tempo esperado para se obter uma das MST referente àquela triangulação é linear($O(n)$). A Figura 3.26 mostra uma MST de um conjunto de pontos.

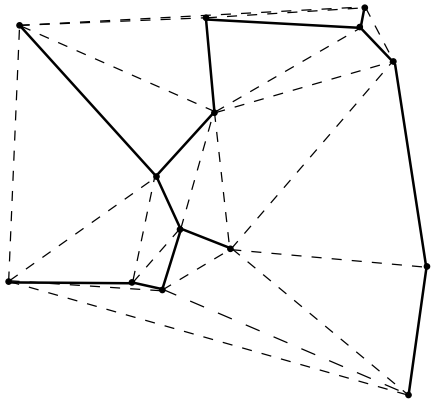


Figura 3.26: MST de um conjunto de pontos sobre a Triangulação de Delaunay - em linhas pontilhadas

3.11.5 Modelagem Digital de Terrenos

Um Modelo digital de Terrenos associado a um espaço físico, permite que seja feito vários tipos de simulação sobre ele, tais como treinamento em simulação de vôo, extração de curvas de níveis, estradas, túneis, barragem etc. Porém somente a existência de um conjunto de pontos amostrados sobre a superfícia não oferece nenhum tipo de informação adicional de natureza topológica(incidência, adjacência, conectividade, relações de proximidade) e também geométrica(declividade, orientação do terreno). Isso torna necessário a existência de uma triangulação na medida em que ela estabelece a conexão

entre os pontos criando duas novas entidades, as arestas e as faces. Com uma estrutura de dados topológica associada à triangulação é possível obter muitas informações sobre o terreno, e permitindo gerar a abstração necessária para criar um modelo digital de terrenos. O processo basicamente acontece da seguinte maneira:

1 Primeiro obtém-se uma nuvem de pontos no espaço através de um scaneamento sobre o terreno, geralmente feito com um avião que possui algum tipo de laser que permite obter um ponto espacial com precisão. Figura 3.27 a);

2 Faz-se a projeção desses pontos sobre o plano. Figura 3.27 b);

3 Com os pontos projetados no plano aplicamos nosso algoritmo para triangularizar os pontos. Figura 3.27 c);

4 Finalmente retornamos os pontos projetados à sua altura espacial inicial e obtemos a representação tridimensional do terreno utilizando do modelo já representado no plano. Figura 3.27 d)

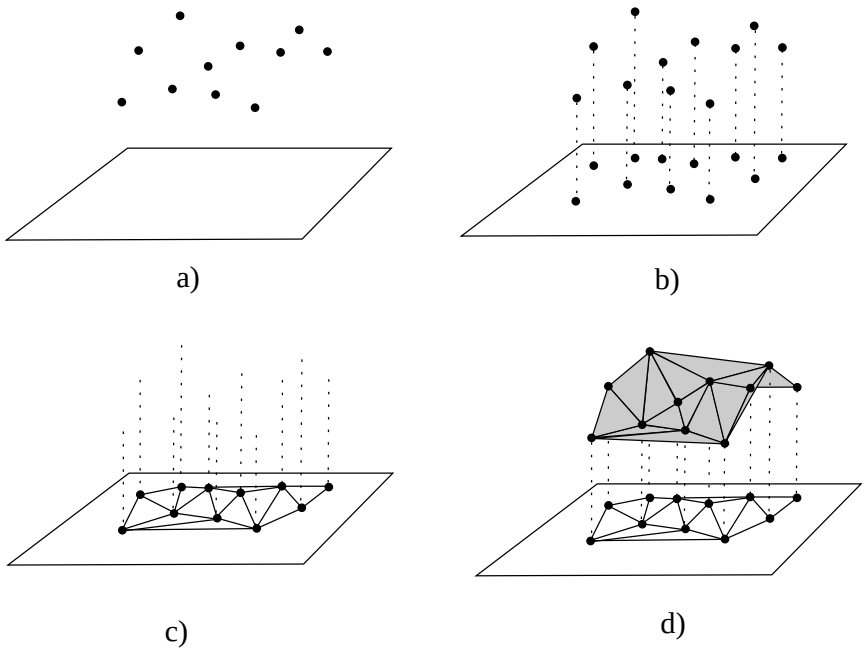


Figura 3.27:

Capítulo 4

Como Obter a Triangulação de Delaunay

4.1 Introdução

A grande aplicabilidade da Triangulação de Delaunay tem levado vários pesquisadores a desenvolver algoritmos, por vezes muito complexos, que levam a uma triangulação eficiente em pontos dados aleatoriamente sobre o plano. Tais algoritmos tem função de computar a triangulação tanto no plano bidimensional como também no espaço.

4.2 Algoritmo de Inserção Randomizado

Para implementação adotamos esse algoritmo no qual utilizaremos a estrutura de dados já antes definida *winged-edge modificada*, que nos permitirá articulação e generalidade total para manipulação de algoritmos ligados à *Triangulação de Delaunay*.

Nesse algoritmo incremental, a *Triangulação de Delaunay* é realizada fazendo-se uma geração de pontos randômica[15] e utilizando a técnica de regularização de Lawson para que possa ser obtido seu resultado final.

Para iniciar o processo, três pontos são escolhidos para formar um *supertriângulo*(técnica de Watson [33]), o qual contém todos os pontos que farão parte da triangulação. Inicialmente a *Triangulação de Delaunay* é formada por um único triângulo definido pelos *vértices* $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$ do *supertriângulo* . Ao inserir um novo ponto na triangulação existente, é feito o

teste do *InCírculo*, e qualquer triângulo que conténha o novo ponto no interior do círculo definido é um triângulo que não satisfaz o princípio de Delaunay. Como o *supertriângulo* cerca todos os pontos da triangulação, cada novo ponto deve estar dentro de pelo menos um dos triângulos existentes. Esses triângulos, quando juntos, formam um polígono com todos os seus *vértices* incidindo em sua fronteira. Todos os triângulos são substituídos e o novo ponto forma um novo triângulo com cada par de *vértices* da fronteira do polígono (note que o novo ponto está sempre no interior do polígono). Depois de cada ponto inserido, o ganho total do número de triângulos deve ser exatamente dois triângulos. Isto acontece porque cada polígono formado pelos triângulos tem sempre duas *arestas* a mais que triângulos satisfazendo a relação de Euler 3.30. Então, quando a triangulação de n pontos é finalizada devemos ter $2n + 1$ triângulos (incluindo os triângulos formados com os *vértices* do *supertriângulo*).

Após todos os pontos serem inseridos, para terminar a triangulação é feita a remoção de todos os triângulos que contém um ou mais *vértices* do *supertriângulo*. Qualquer *vértice* dos triângulos removidos que não seja um *vértice* do *supertriângulo* deve estar sobre a fronteira da triangulação final.

Os passos fundamentais desse algoritmo estão detalhadamente explicados a seguir:

1 Definir os *vértices* de um grande triângulo no sentido anti-horário. Por conveniência chamamos esses *vértices* de $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$. Junte as coordenadas desse *vértices* e então todos os pontos a serem triangularizados ficarão no interior do triângulo Figura 4.1. Adicione o grande triângulo à lista dos triângulos formados;

2 Faça a geração randomizada de pontos e insira numa lista duplamente encadeada de vértices. Fazendo a eliminação de pontos de mesma coordenada posteriormente.

3 Insira um novo ponto da lista previamente gerada na triangulação. Faça a localização da face que contém o novo ponto.

4 Se o ponto não cair sobre uma aresta já criada anteriormente execute os passos de **5** a **8**. Caso Contrário execute o passo **9**

5 Crie três novos triângulos. O novo ponto forma novos triângulos com cada par de *vértices* que estão na fronteira do polígono formado pela intersecção dos triângulos.

6 Execute o teste do InCírculo para cada triângulo construído.

7 Se o teste do InCírculo for negativo, faça o *flip-edge* nesse triângulo.

8 Execute o passo recursivamente até que todos triângulos da região modificada satisfaça o critério de Delaunay.

9 Crie 4 novos triângulos formado pela união do novo ponto ao vértice que não pertença a aresta no qual se encontra o novo ponto. Execute os passos de **5** a **8** para cada novo triângulo recém criado.

10 Repita os passos **3** e **4** até que não haja mais pontos a serem processados.

11 Construa a triangularização removendo todos triângulos que contém um ou mais *vértices* do grande triângulo Figura 4.4. Isso pode ser feito varrendo a lista de triângulos e apagando qualquer triângulo no qual o *vértice* é maior que n.

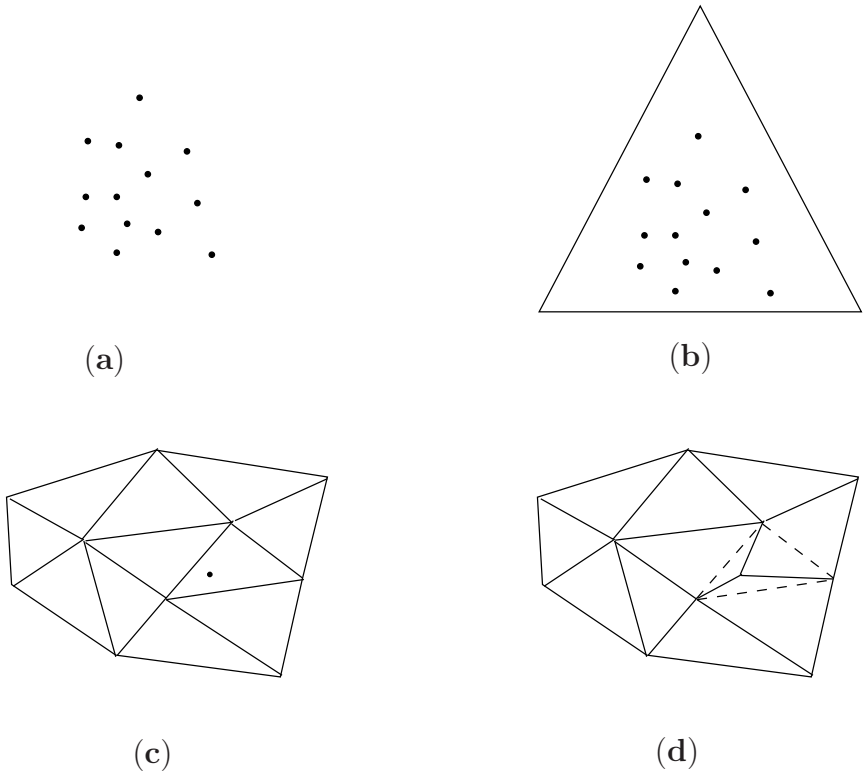
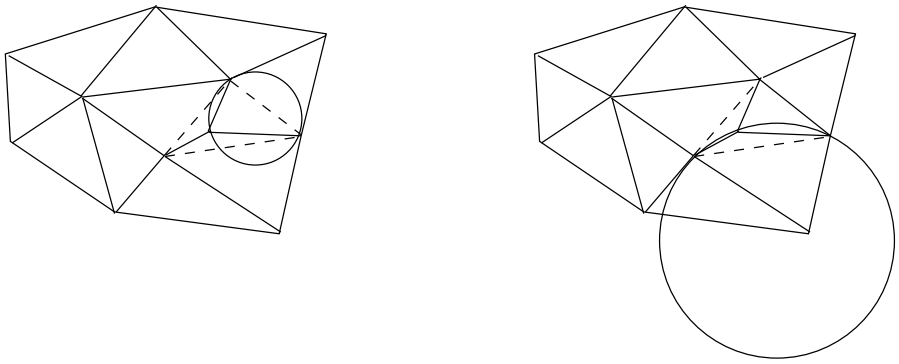


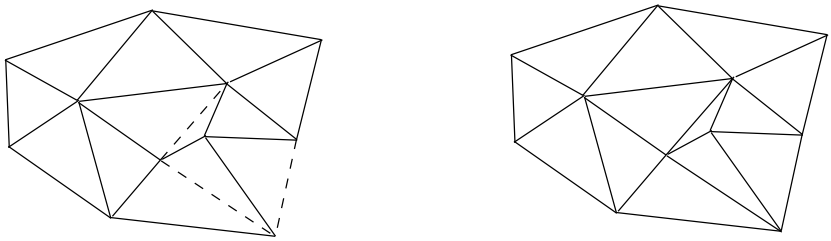
Figura 4.1: a) Conjunto de pontos no plano; b) Triângulo imaginário incidente ao conjunto de pontos; c) Um ponto sendo inserido no interior da pré-triangulação; d) Atualização das arestas incidentes sobre os triângulos na vizinhança.

Como por construção, todos os triângulos criados pelo algoritmo são localmente Delaunay, a triangulação gerada é de fato, a *Triangulação de Delaunay*



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

Figura 4.2: a) e b) Exame das arestas dos triângulos formados.



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

Figura 4.3: a) Execução do *flip* no quadrilátero; b) Triangulação de Delaunay após verificação das arestas restantes.

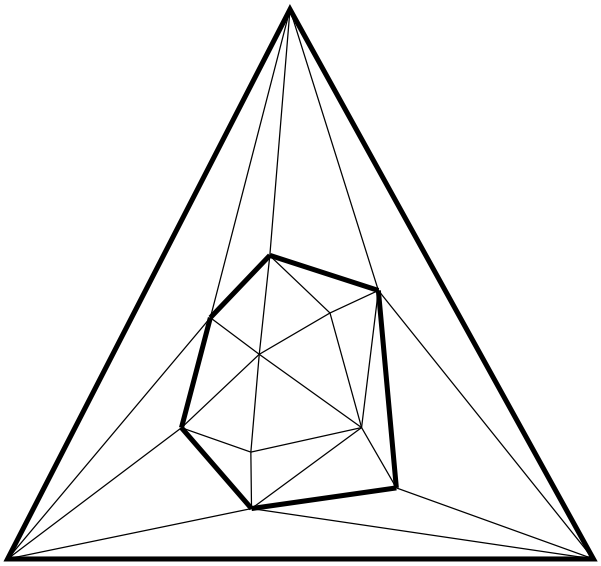


Figura 4.4: Triângulo imaginário incidente sobre a triangulação e o *Fecho Convexo* em destaque.

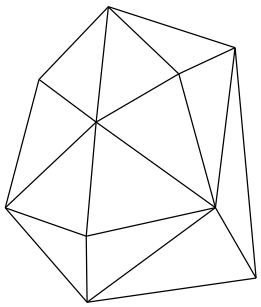


Figura 4.5: Triangulação de Delaunay obtida após remoção das *arestas* incidentes.

Figura 4.5. Como a complexidade do algoritmo é dependente da busca a ser realizada, e como utilizamos no momento a busca por coordenadas baricêntricas seção 4.2.2 a complexidade algoritmica fica limitada em $O(\sqrt{n})$.

Veremos abaixo o pseudo-código desenvolvido do algoritmo a ser implementado:

Algoritmo 1 Algoritmo incremental

Entrada: Um conjunto P de n pontos no plano

Saida: A Triangulação de Delaunay de P. Seja $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$ um conjunto de três pontos que formam uma grande triângulos imaginário que contem P.

Inicializar a triangulação $\mathcal{T}$ que possui apenas um triângulo simples $\mathbf{p}-1$, $\mathbf{p}-2$ e $\mathbf{p}-3$

para r $\Longleftarrow$ 1 **até** n **faça**

Insere-se um novo ponto em $\mathcal{T}$

Encontre o triângulo $p_ip_jp_k$ que contém o ponto p.

Se p está no interior do triângulo $p_ip_jp_k$ **então**

Ligue o ponto p aos três *vértices* formados criando *arestas* e dividindo $p_ip_jp_k$ em três triângulos.

flip $(p, \overline{p_ip_j})$

flip $(p, \overline{p_jp_k})$

flip $(p, \overline{p_kp_i})$

senão

Adicione arestas de p até $\mathbf{p}_k$ e ao terceiro *vértice* $\mathbf{p}_l$ do outro triângulo que é incidente a $\overline{p_ip_j}$, logo dividindo os dois triângulos incidentes a $\overline{p_ip_j}$ em quatro.

flip $(p, \overline{p_ip_l})$

flip $(p, \overline{p_lp_j})$

flip $(p, \overline{p_jp_k})$

flip $(p, \overline{p_kp_i})$

fim Se

fim para

Remova os *vértices* p-1, p-2 e p-3 e também todas *arestas* do supertriângulo incidentes na triangulação

O procedimento auxiliar **flip** verifica se uma *aresta* é ilegal Figura **??**(b). Se a verificação for verdadeira esse procedimento realizará um *flip* e verifica recursivamente mais duas *arestas* potencialmente ilegais.

No pseudo-código a inserção de cada ponto incide sobre um triângulo pertencente a *Triangulação de Delaunay*. Na primeira inserção, o ponto incide dentro do *supertriângulo*, que pode-se dizer que é uma *Triangulação de Delaunay*. Por isso cada ponto inserido estará sempre dentro de um triângulo, e então faz se a ligação do ponto inserido a cada *vértice* deste triângulo, criando

mais três novas *arestas* na triangulação. A Figura 4.6 mostra como é feita a inserção do primeiro ponto, e para isso é preciso que seja feita várias atualizações nos elementos que compõem a estrutura de dados da geometria. Isto é feito para cada ponto inserido na triangulação. Na Figura 4.6(a) ilustramos a inserção do primeiro ponto na triangulação, que incide no *supertriângulo*. Esta primeira inserção difere das demais inserções apenas pelo fato de que o *supertriângulo* tem suas *arestas* orientadas de acordo com o seu índice em ordem crescente, já nas demais inserções é necessário fazermos a verificação da orientação das *arestas*, fora isto, e levando em conta este detalhe, esta representação já é suficiente para entendermos este procedimento. Na Figura 4.6(b) temos o primeiro vértice V, inserido no *supertriângulo*, e também a configuração do *supertriângulo* com relação aos elementos que compõem sua estrutura. Para cada inserção de *aresta*, é necessário que se faça atualização dos ponteiros para que se mantenha a consistência da representação na estrutura. Ao final deste passo teremos criado mais duas *faces* e seus respectivos *loops* como vemos nas Figuras 4.6(c) e 4.6(d).

Após a inserção do ponto e a criação das novas *arestas*, é feita a verificação das *arestas* do triângulo inicial para manter o invariante da *Triangulação de Delaunay*, ou seja, é necessário verificar se estas *arestas* são ilegais, pois o invariante diz que o círculo cinrcunscrito de cada triângulo da triangulação não deve conter outro ponto do conjunto de pontos S da triangulação. Se a *aresta* verificada for ilegal, então é feito o *flip-edge* da *aresta*, sendo necessário assim fazer a verificação de mais duas *arestas* como vimos anteriormente. Neste caso em especial o *flip-edge* é realizado de forma simples, apenas modificando alguns elementos da estrutura. O elemento destas *arestas* que precisa ser atualizado é o $loop_1$ ou o $loop_2$, dependendo da orientação das *arestas*. Isto tem que ser feito, pois estes elementos apontavam para *loops* diferentes antes do *flip*.

A seguir nas Figuras 4.7 e 4.8 listamos exemplos de triangulação gerada pelo algoritmo Incremental de Lawson.

4.2.1 Escolha adequada das coordenadas de um supertriângulo

No algoritmo incremental de Lawson, citamos a necessidade de criar-se um grande triângulo que contenha todos os pontos da triangulação, as coordenadas dos vértices do grande triângulo segundo o criterio devem ser escolhidas como sendo os maiores valores possíveis. Porém o ideal é que escolhamos números adequados para que o supertriângulo contenha todos os pontos que serão processados e que não tenha coordenadas muito exageradas. Então o que faremos será escolher os pontos $\mathbf{p} - 1$, $\mathbf{p} - 2$ e $\mathbf{p} - 3$ e modificar o teste

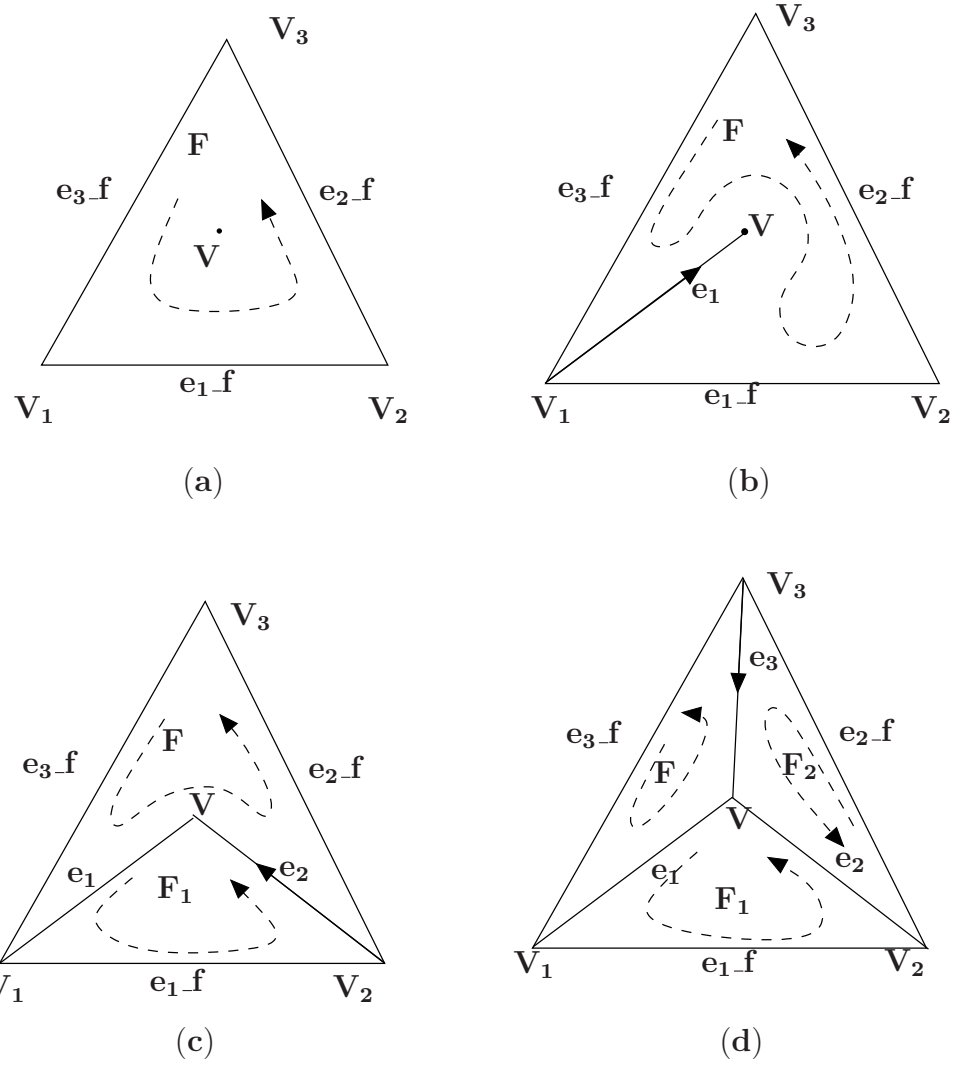


Figura 4.6: Ilustração das atualizações topológicas na inserção do primeiro ponto **V**.

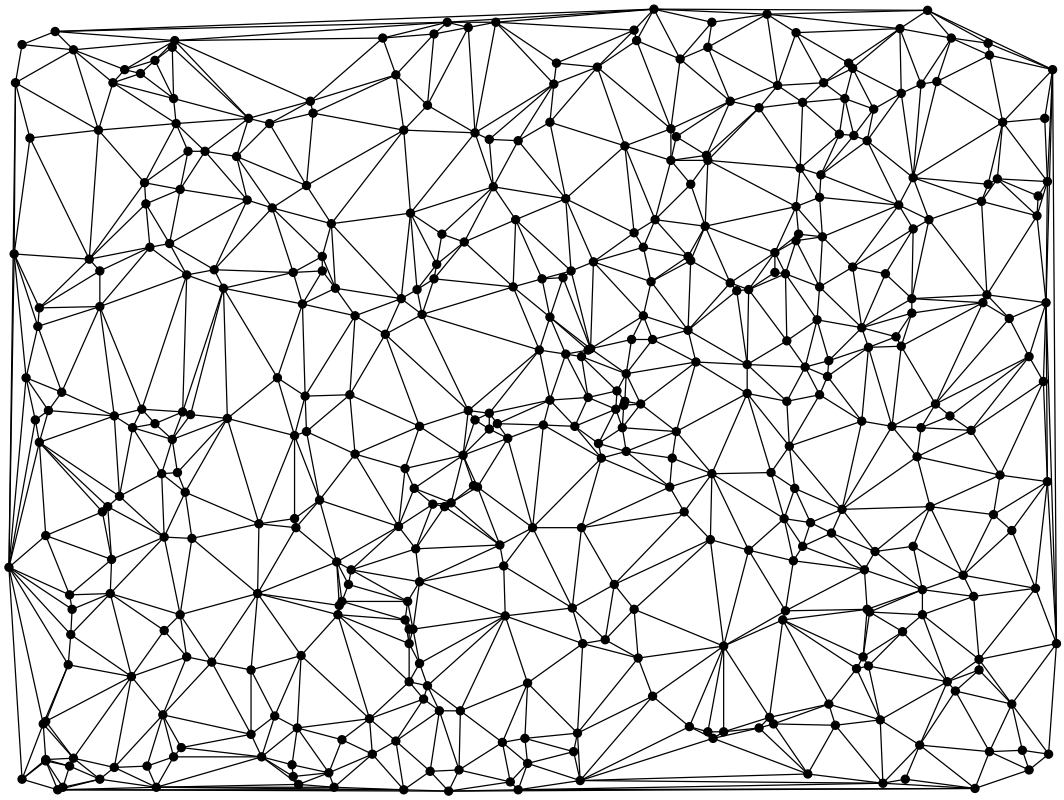


Figura 4.7: Triangulação de 400 pontos aleatorios gerada pelo algoritmo Lawson.

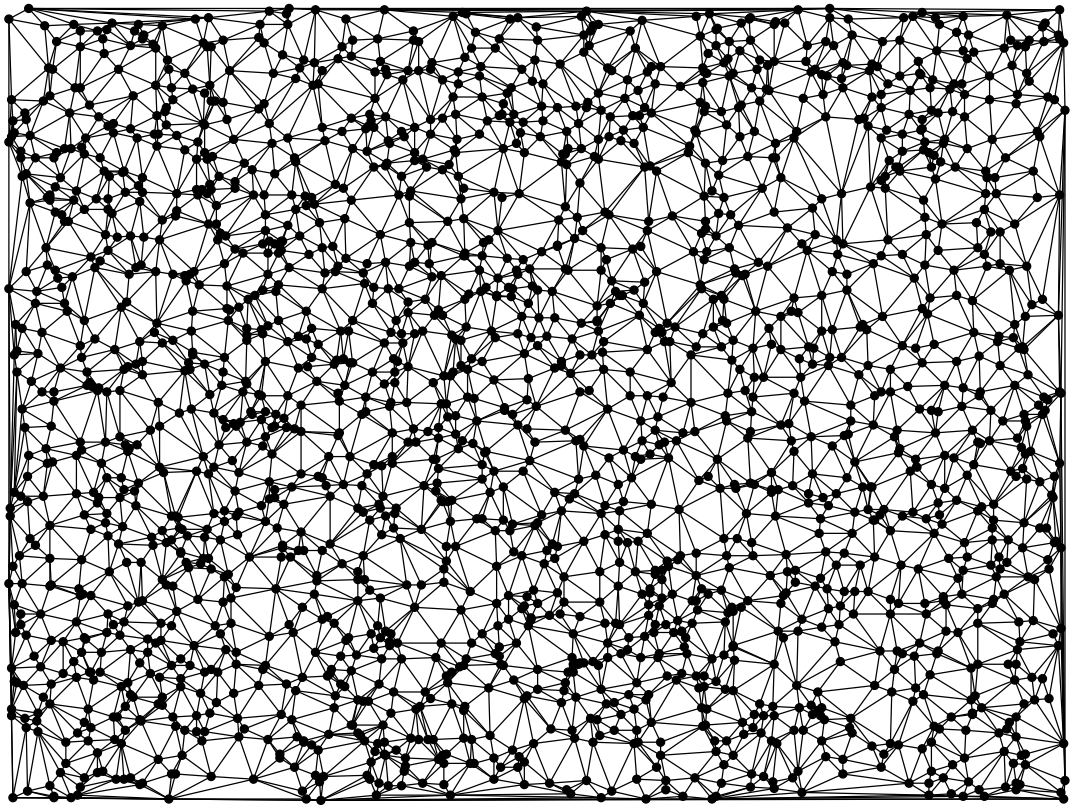


Figura 4.8: Triangulação de aproximadamente 2000 pontos gerada pelo algoritmo Lawson.

feito para arestas ilegais, funcionar como se estivéssemos os escolhido com coordenadas bem grandes. Uma sugestão feita em [3] é que a geração de pontos seja feita no interior de um quadrilátero de lado M e que o supertriângulo deverá envolver o quadrilátero tendo coordenadas de distância $3M$ ($[0, 3M]$, $[3M, 0]$, $[-3M, -3M]$), embora ao invés de $3M$ alguns autores sugerem o uso de uma distância de $6M$ ($[0, 6M]$, $[6M, 0]$, $[-6M, -6M]$) figura 4.9. Uma outra sugestão é utilizar um circulo de raio M, e gerar os pontos no interior do quadrilátero inscrito na circunferência segundo a figura 4.10

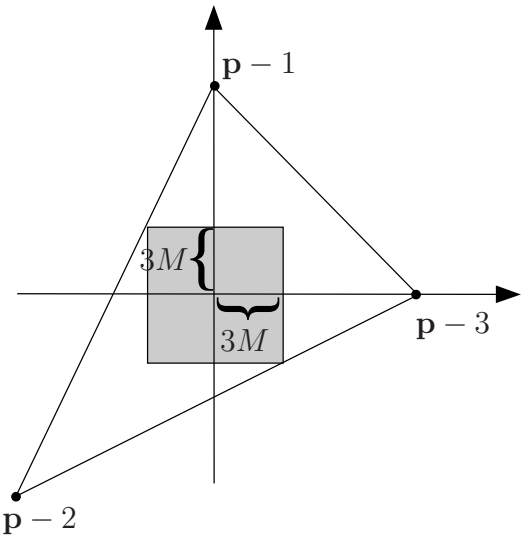


Figura 4.9: Supertriângulo envolvendo o quadrilátero com coordenadas $3M$.

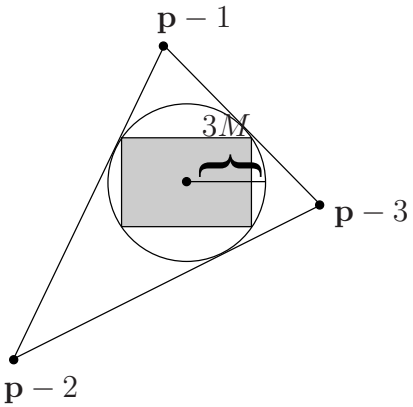


Figura 4.10: Circunferência de raio $3M$ inscrita no Supertriângulo envolve um quadrilátero onde serão gerados os pontos.

4.2.2 Localização do ponto na triangulação

Entre todos os paradigmas computacionais inserção, divisão e conquista e varredura por linhas, o que mais evolui e possui um maior número de representantes é o paradigma de inserção devido à sua grande simplicidade robustez e facilidade de ser generalizado para outras dimensões. O algoritmo incremental que implementamos pertence a essa a classe de inserção, a idéia básica do algoritmo é inserir um ponto por vez na triangulação já existente e ir atualizando de modo que satisfaça o critério de Delaunay, até que se tenha processado todos os pontos. A questão imediata é saber qual o *próximo ponto* entre todos aqueles manipulados será processado. Muitas das variantes algorítmicas associadas a esse paradigma residem no estabelecimento dessa sequência enquanto outras variantes baseiam se na forma de como a triangulação será atualizada e como será feita a operação de localização do elemento triangular. No princípio de randomização fora observado que a alteração na ordem de entrada dos pontos poderia colaborar para minimizar o esforço computacional associado ao algoritmo. Em seguida mostraremos três dos métodos que utilizamos para implementar em nosso algoritmo e como funciona cada método.

Localização Linear

Nesse algoritmo que foi implementado(Lawson), a triangulação que está sendo construída é representada pela já mencionada *winged-edge modificada*. Assim como primeira alternativa seria realizar uma pesquisa linear em cada uma das faces do modelo hierárquico topológico associado a essa estrutura até encontrar definitivamente aquela que contém o ponto em seu interior A busca era feita da seguinte forma, inicialmente inseríamos um ponto aleatório na triangulação, a localização desse ponto era feita de maneira exaustiva, procurando o ponto em cada face, existente, utilizando o fato da conexão de todas as faces, representadas como sendo uma lista duplamente encadeada. Esse método é o mais simples para se localizar um ponto qualquer, mas sabemos que para "passear"pela lista de faces, torna-se muito custoso, em termos de recursos computacionais, e em uma grande triangulação esse tempo de transição entre uma face e outra, torna o algoritmo ineficiente conduzindo-o a ter um tempo quadrático ($O(n^2)$). Para diminuir esse grande prejuízo, temos vários métodos que faz com que a localização do ponto seja feita de maneira rápida e eficiente, dentre eles optamos por implementar dois que serão descritos detalhadamente nas próximas subseções.

Localização do ponto pelo método das coordenadas baricêntricas

Uma segunda alternativa, é partir de um elemento triangular e verificar se ele contém o ponto $\mathbf{p}_r$ mais recentemente introduzido. Se ele contiver, as busca foi concluída, senão, por meio do uso de coordenadas baricêntricas do triângulo e das regiões induzidas pelas retas suportes associadas a cada um de seus lados, é possível classificar o ponto $\mathbf{p}_r$ em relação a esse triângulo e assim, obter a direção a ser seguida, que será através de um de seus triângulos adjacentes. Essa idéia é repetida sistematicamente até que o triângulo que contém o ponto $\mathbf{p}_r$ seja encontrado. Tal situação ocorre quando as coordenadas baricêntricas desse ponto em relação ao triângulo são todas positivas, ver figura 4.11. Por fim, é preciso realçar que as coordenadas baricêntricas podem ser facilmente obtidas a partir de um sistema linear de dimensão 3. Para conjuntos de pontos distribuídos uniformemente pelo domínio, o tempo dessa estratégia e busca está limitado a $O(\sqrt{n})$. Como n pontos são inseridos na triangulação, a complexidade de tempo de processamento do algoritmo fica limitada a $O(n\sqrt{n})$, mais eficiente do que no caso anterior em que o tempo é quadrático.

A grande vantagem dessa busca é que ela pôde ser associada ao fato de que a triangulação representada pela estrutura *winged edge modificada* possui as faces consistentemente orientadas e com isso podemos "caminhar" pelos triângulos utilizando a primitiva $Ccw(.)$ descrita no capítulo 3. O Caminho feito desde a última face da lista de faces $\mathbf{f}_{ult}$ até a próxima face $\mathbf{f}_{prox}$ que contém o próximo ponto a ser processado na triangulação está ilustrado na fugura 4.12

Localização do ponto utilizando a DAG

O conceito da hierarquia entre os elementos define claramente o funcionamento desse método eficiente de busca.

A DAG(Directed Acyclic Graph) é um tipo de estrutura que pode manter um histórico de toda a triangulação desenvolvida. Essa estrutura é semelhante a um diagrama do tipo árvore (mas não pode se dizer que se trata de uma pelo fato de algumas particularidades das estruturas que as diferem) no qual cada nó representa um triângulo presente na fase da construção, quando um novo ponto é adicionado à triangulação os três triângulos resultantes são armazenados nos nós filhos do nó que aponta para o triângulo que foi subdividido.

Quando um flip de uma aresta é realizado os dois triângulos adjacentes conterá os dois novos triângulos.

A DAG será inicializada com um simples nó folha, correspondente ao

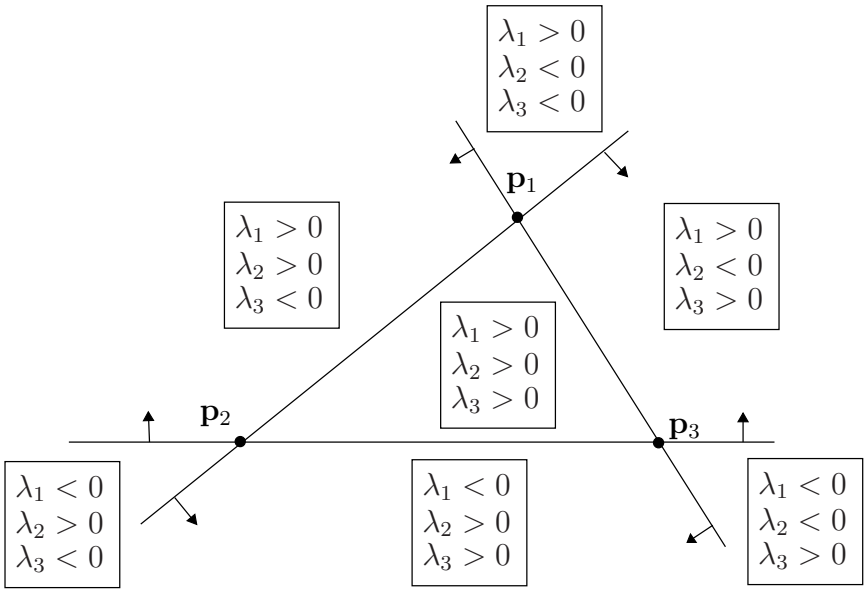


Figura 4.11: Todos os possíveis resultados da orientação através das coordenadas baricêntricas.

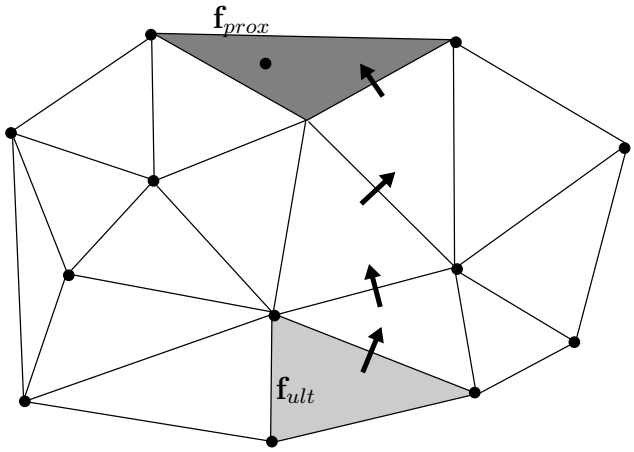


Figura 4.12: Ilustração de como é feita a busca pelas coordenadas baricêntricas. A face hachurada mais clara é o ponto de partida enquanto a mais escura é o ponto onde se deseja chegar, e as setas mostram o caminho a ser percorrido fazendo o teste de sinais ilustrado na figura 4.11.

triângulo inicial p-1, p-2 e p-3. Um exemplo da utilização da estrutura DAG está ilustrado na figura 4.13.

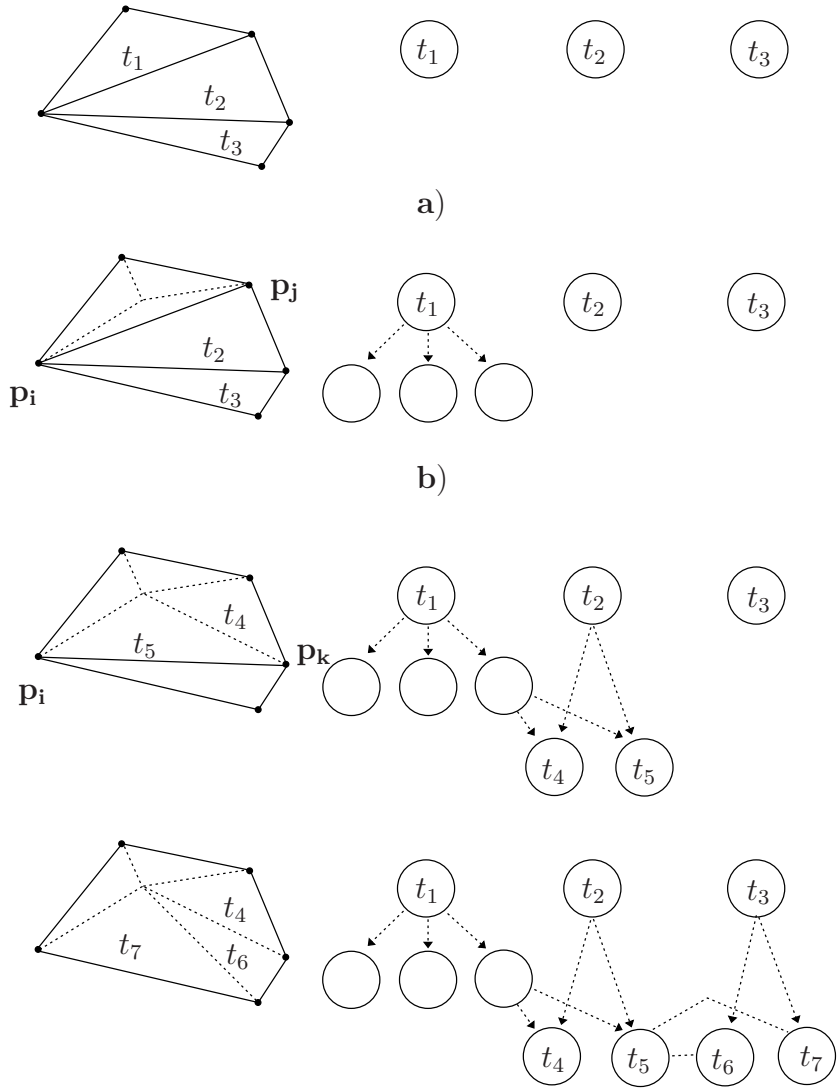


Figura 4.13: Esquema de construção da estrutura Directed Acyclic Graph (DAG).

Agora suponhamos que um ponto seja inserido e dividiremos o triângulo $p_i\ p_j\ p_k$ da triangulação atual em três (ou dois) triângulos. A mudança correspondente na DAG é adicionar três (ou dois) novos nós folhas à DAG, e fazer a folha como nó interno para $p_i\ p_j\ p_k$ com ponteiros para os três nós folhas. Do mesmo modo, se substituirmos os dois triângulos $p_i\ p_j\ p_k$ e $p_i\ p_j\ p_l$ por $p_i\ p_k\ p_l$ e $p_j\ p_k\ p_l$ através da realização de um flip, criaremos nós folhas para os dois triângulos, e os nós de $p_i\ p_j\ p_k$ e $p_i\ p_j\ p_l$ receberão os ponteiros para os dois nós folhas.

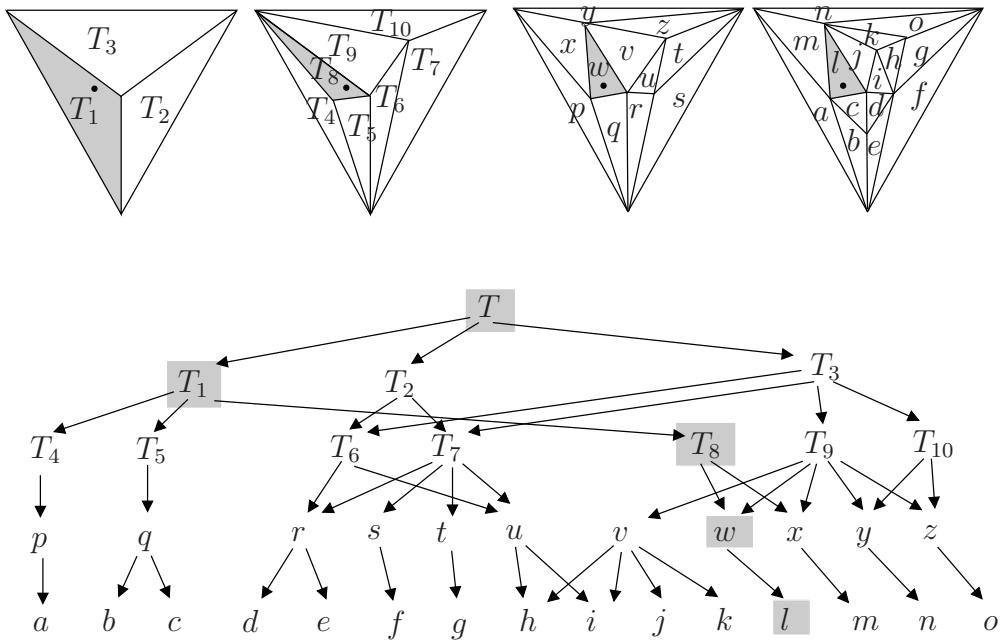


Figura 4.14: Nesse esquema mostramos a eficiência da busca realizada pela DAG iniciada por um ponteiro para um triângulo T(grande triângulo)e percorrendo todos os nós filhos correspondentes até o triângulo onde se localiza o ponto.

Para localizar o próximo ponto p_r para ser adicionado na triangulação recente na DAG, começaremos pelo nó raiz. Verificamos se os três nós filhos da raiz contém o ponto p_r, se a resposta for negativa então saberemos que o ponto pertencerá à face exterior caso contrário desceremos para o filho correspondente a esse nó que contém o ponto p_r, se fizermos isso sucessivamente encontraremos o triângulo que contém o ponto numa das folhas da estrutura. Veja na Figura 4.14 como é feita a localização do ponto a partir de um no raiz, usando o conceito de níveis e de nós filhos para executar a tarefa.

Esse algoritmo de Kirkpatrick [18] consegue encontrar o triângulo que contém o ponto num tempo esperado de $O(\log n)$ e de $O(n)$ no pior caso (quando a DAG está desbalanceada possuindo a estrutura similar a uma lista), isto é, quando se procura o ponto, a descida ocorre nível-a-nível através da estrutura ou seja cada tempo de transição entre os níveis é $O(1)$, ao final da busca o tempo será proporcional ao número de níveis da estrutura.

4.3 Algoritmo Incremental de Bowyer / Watson

Com já foi descrito anteriormente o algoritmo Incremental de Lawson gera a Triangulação de Delaunay segundo as comparações feitas nas arestas e realizando o flip se encontrar uma aresta ilegal.

Nessa outra abordagem algorítmica a Triangulação inicia-se da mesma maneira, ou seja criando um *supertriângulo* que conterá todos os pontos, porém a comparação para verificar se os triângulos da região satisfazem o princípio de Delaunay é feita testando o incírculo nos triângulos ao redor do ponto, fazemos o seguinte:

- A cada novo ponto inserido verifica se o circuncírculo, dos triângulos adjacentes ao que contém o ponto, possuirá esse novo vértice em seu interior;
- Se a resposta for verdadeira então tais triângulos serão apagados;
- Todos os triângulos remanescentes que são Delaunay serão mantidos intactos;

O conjunto de triângulos apagados formarão um polígono convexo ilustrado na Figura 4.15.

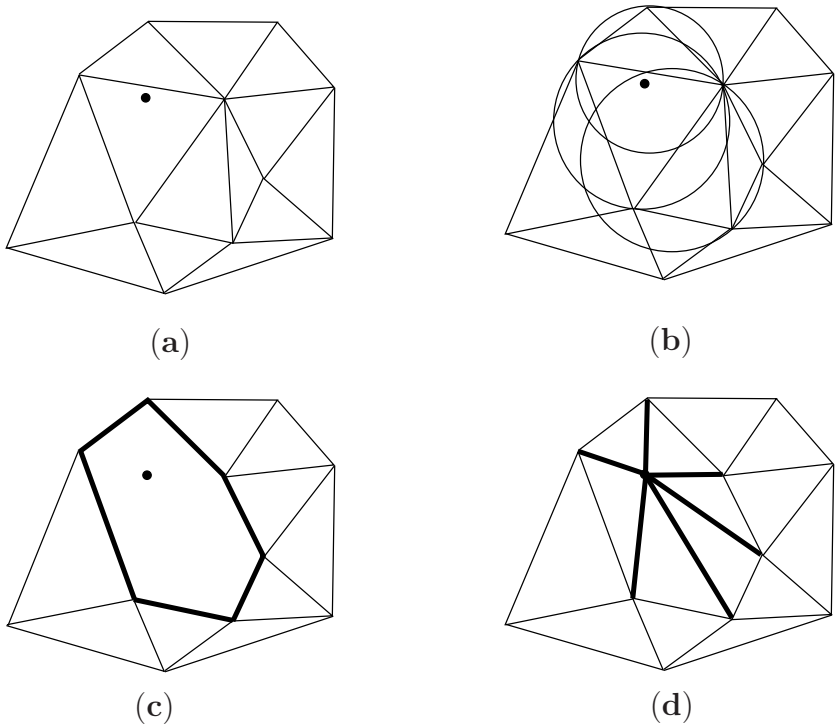


Figura 4.15: a) Inserção do ponto na triângulação; b) Verificação da propriedade Delaunay nos triângulos da vizinhança do ponto; c) Remoção das arestas dos triângulos que não satisfazem Delaunay gerando um *Fecho Convexo* em torno do ponto; d) Criação das *arestas* ligando cada *vértice* do fecho ao ponto.

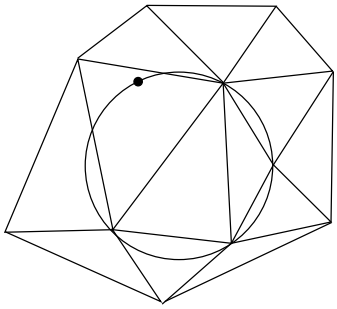


Figura 4.16: Caso degenerado de aproximação, no qual um ponto compartilha o mesmo circuncirculo relativo a dois triângulos distintos.

O algoritmo de Bowyer/Watson conectará cada vértice do polígono ao novo vértice através de uma aresta. Tais arestas seguirão o seguinte lema:

Lema 6 *: Seja v um vértice inserido recentemente e t um triângulo que foi apagado pelo fato do de seu circuncirculo conter o vértice v, sendo w um vértice de t. Entao vw satisfaz Delaunay.*

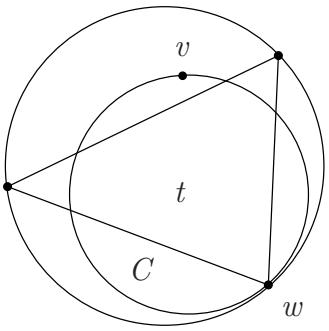


Figura 4.17: Ilustração do circuncírculo que contém o vértice v.

Prova Veja a Figura 4.17. O circuncírculo de t contém somente o vértice v e nenhum outro. Seja C o círculo que passa por v e w, tangente ao circuncírculo de t em w. C está vazio implica que vw satisfaz Delaunay.

Todos as novas arestas criadas tem como extremo o vértice v. Essa afirmação deve se confirmar para qualquer algoritmo de inserção incremental, porque se uma aresta (que não tem v como extremo) não é Delaunay antes de inserir v, não será depois de inserir v.

Na forma mais simples, o algoritmo de Bowyer/Watson não é robusto contra erro de arredondamento. A Figura 4.16 ilustra um exemplo degenerado em

que dois triângulos têm o mesmo circuncírculo devido ao erro de arredondamento somente um triângulo é deletado, e o triângulo restante permanece entre o novo vértice e o novo triângulo.

Ao final do passo essa triângulação resultante será exatamente a mesma da que foi gerada pelo algoritmo de Lawson (que fora implementado).

4.4 Uma outra abordagem algorítmica Incremental

Nessa seção descreveremos um algoritmo incremental para construção de uma região convexa de um conjunto P de n pontos. Esse algoritmo se encontra explicado em [6]. O método incremental consiste em adicionar pontos p_i consecutivamente um a um no fecho, e atualizá-lo através de incrementos no algoritmo.

Nesse instante para uma melhor compreensão, as arestas, faces e os vértices serão identificados por cores. Identificaremos que as faces **vermelhas** são aquelas que seriam iluminadas se um ponto p_i(não pertencente ao fecho $Conv(P_{i-1})$) estivesse emitindo luz. As faces e arestas **azuis** seriam as que ficariam na sombra segundo essa configuração e as **violetas** seriam tangentes pelos raios de luz.

Os pontos em P serão processados na ordem lexicográfica de suas coordenadas. Para simplificar a descrição do algoritmo, assumiremos abaixo que o conjunto de pontos esteja numa posição randômica. Denotaremos por $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots, \mathbf{p}_n\}$ os pontos de P indexados na ordem lexicográfica. Seja P_i o conjuntos dos primeiros i pontos de P.

A dinâmica geral do algoritmo segue:

1 Ordenar os pontos de P em ordem crescente lexicográfica das coordenadas.

2 Inicializar o *Fecho Convexo* para o simplexo $Conv(P_{n+1})$, o *Fecho Convexo* dos primeiros $n+1$ pontos de P.

3 No passo incremental, o *Fecho Convexo* de $Conv(P_i)$ é construído sob o conhecimento do *Fecho Convexo* $Conv(P_{i-1})$ e o ponto p_i será inserido.

4.4.1 Detalhes do passo incremental

Devido à vantagem da ordem lexicográfica dos pontos de P, o ponto p_i nunca pertencerá ao *Fecho Convexo* $Conv(P_{i-1})$, mas será portanto um vértice do fecho $Conv(P_i)$. Toda a eficiência do algoritmo incremental baseia-se no fato de que o grafo incidente do fecho mais recente pode ser atualizado em um passo incremental sem procurar por todas as arestas da lista do fecho.

Para representar esse passo incremental, dividiremos em 4 fases:

Fase 1 : Primeiro identificaremos uma superfície de $Conv(P_{i-1})$ que é **vermelha** em relação a p_i.

Fase 2 : As arestas que são visíveis, ditas **vermelhas**, de $Conv(P_{i-1})$ são passeadas.

Fase 3 : Usando a informação coletada na Fase 2, identificaremos todas as outras faces **vermelhas** ou **violetas** de $Conv(P_{i-1})$. Para cada dimensão k, $n-3 \geq k \geq 0$, uma lista L_k das $k-faces$ **vermelhas** é computada, além da lista P_k de $k-faces$ **violetas**.

Fase 4 : O grafo incidente é atualizado.

Fase 1: Para localizar uma face **vermelha** inicial em $Conv(P_{i-1})$, usaremos a vantagem da ordem lexicográfica dos pontos de P. Porque nessa ordem p_{i-1} será sempre um vértice de $Conv(P_{i-1})$ e haverá no mínimo uma superfície de $Conv(P_{i-1})$ contendo p_{i-1} que é **vermelha** em relação a p_i. Entretanto, façamos com que F_{i-1} seja o conjunto de superfícies de $Conv(P_{i-1})$ que contém p_{i-1} como vértice. Assim passeamos pelas arestas **vermelhas** e atualizamos a topologia do fecho na lista do fecho que será passeada no próximo passo incremental.

Fase 2: Na segunda fase, usaremos a conectividade do conjunto de arestas **vermelhas** inicial que fora encontrada na **fase 1**, visitamos todas as arestas vistas de p_i, no qual predomina a cor **vermelha**, e suas $n-2$ faces, que colorimos de **vermelhas** se são incidentes, ou **violeta** se são incidente a uma superfície azul.

Fase 3: Conhecemos todas as faces **vermelhas** e **violetas**, e as arestas **vermelhas**. Nessa fase, todas as faces **vermelhas** e **violetas** restantes serão coloridas, e suas listas atualizadas em ordem decrescente das dimensões. Por

indução assumiremos que toda face **vermelha** e **violeta** de dimensão $k' \geq k+1$ já foi identificada e colorida, e que as listas $R_{k'}$ e $P_{k'}$ já foram devidamente atualizadas. Processaremos as k faces da seguinte maneira. Cada sub-face de uma face de P_{k+1} que ainda não está colorida será colorida de **violeta** e adicionada à lista P_k. Consequentemente, cada sub-face de R_{k+1} que ainda não é colorida é adicionada à lista.

Fase 4: Para atualizar o grafo incidente, procederemos assim. Todas as faces **vermelhas** serão removidas da lista incidente através do uso da primitiva *Ccw(.)* utilizando como referência a reta que passa pelos pontos que formam retas tangente ao fecho com o vértice que será inserido no fecho. Após a realização da remoção atualizamos as arestas em relação à face. Então o a lista do fecho incidente é atualizada.

4.4.2 Uso do fecho convexo para gerar uma triangulação

O algoritmo que nós apresentaremos aqui utiliza o método incremental para calcular a triangulação de um conjunto de pontos. Esse algoritmo usa o mesmo esquema do algoritmo incremental que calcula o *Fecho Convexo* de um conjunto de pontos. O algoritmo primeiramente ordena os pontos lexicograficamente e em ordem crescente (x,y) e então assegura a triangulação do conjunto de pontos atual obtida através da adição dos pontos um a um na ordem pré estabelecida.

Façamos $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ um conjunto de n pontos no plano. Então assumimos que os pontos estão numa posição qualquer. Além disso, suponhamos que P já esteja ordenado em ordem lexicográfica em (x,y), sendo então $\mathbf{p}_1 < \mathbf{p}_2 < \ldots < \mathbf{p}_n$, o algoritmo não somente preserva a triangulação T_{i-1} construída para um conjunto $P_{i-1} = \{\mathbf{p}_i, ..., \mathbf{p}_{i-1}\}$ dos pontos já processados, como também a fronteira da triangulação do fecho P_{i-1}. A triangulação atual é mantida numa estrutura de dados que armazena o grafo da triangulação incidente. O *Fecho Convexo* P_{i-1} é mantido usando uma lista duplamente encadeada circular de vértices, com um ponteiro p para os vértices da lista que foram inseridos por último.

No primeiro passo, construímos um triângulo formado por três pontos $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ e juntamo-os à lista L $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$, com um ponteiro p apontando para o nó que armazena $\mathbf{p}_3$.

Para descrição do passo incremental atual quando $\mathbf{p}_i$ é o ponto a ser inserido na triangulação, usaremos a mesma idéia da sessão anteriror. Uma aresta E de $Conv(P_{i-1})$ é **vermelho** a respeito de $\mathbf{p}_i$ se uma linha isola $\mathbf{p}_i$ de de $Conv(P_{i-1})$ ou seja se por exemplo $\mathbf{p}_i$ fosse uma laterna somente as arestas **vermelhas** seriam as mais claras, caso contrário a aresta E é pode ser definida como **azul**

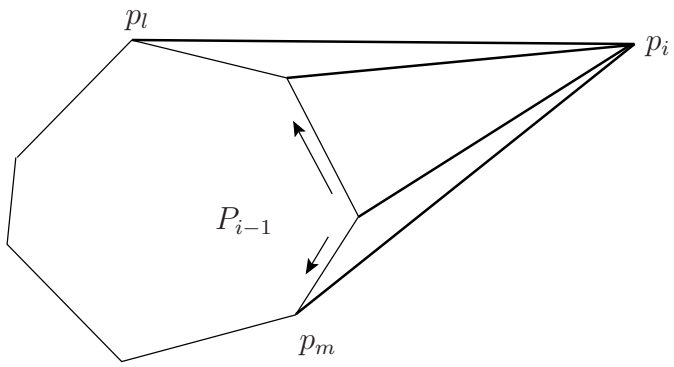


Figura 4.18: Ilustração da triangulação incremental.

a respeito de $\mathbf{p}_i$. Já um vértice de $Conv(P_{i-1})$ é **vermelho** se ele é incidente a dois outros vértices **vermelhos**, **azul** se é incidente a dois outras arestas azuis, e é **violeta** se é incidente a ambas as cores **vermelho** e **azul**. Recapitulando, o vértice p_{i-1} é necessariamente incidente no mínimo a uma aresta **vermelha** e que o conjunto de arestas na fronteira de $Conv(p_{i-1})$ que são **vermelhas** em relação a p_i estão também conectadas. Começando pelo ponto p_{i-1}, o algoritmo passeia pelas arestas **vermelhas** de $Conv(p_{i-1})$, e para cada aresta aresta da mesma cor, adiciona o triângulo $Conv(E, p_i)$ para a triangulação atual. Na sublista de arestas **vermelhas** L são substituídas pelas duas arestas p_ip_m e p_ip_l que conecta p_i aos dois vértices p_m e p_l em $Conv(Pi-1)$ ver Figura 4.18.

Porém apesar da simplicidade algorítmica dessa abordagem, esse algoritmo apresenta muitas singularidades(casos degenerados) e precisaria de muito mais esforço para que fossem tratados todos os casos, assim não demos uma atenção muito especial a esse algoritmo, não oferecendo o nível de robustez que é exigido.

Apesar da instabilidade do algoritmo, será utilizado com propósito didático, por se tratar da constução do fecho convexo, poderá auxiliar no oferecimento de minicursos ou disciplinas relacionadas a geometria computacional ou problemas geométricos e topológicos.

4.5 Sistema

Nessa seção apresentaremos o sistema denominado **Sistema de Automatização da Triangulação de Delaunay** de modo a fornecer uma visão do que fora desenvolvido, e ao mesmo tempo enfatizar a importância de levar adiante a continuação do projeto.

O sistema é um processador bidimensional de algoritmos de Triangulação de Delaunay implementado sobre a estrutura de dados *winged-edge modificada* proposta feita por Braid [5] e construído sobre a interface oferecida pelo compilador Borland Builder C++ 6.0. Basicamente é dada um conjunto de n pontos no plano e o software se responsabiliza por gerar a triangulação de Delaunay sobre o conjunto de pontos dado como entrada mantendo a consistência da estrutura de dados ao qual manipula.

Capítulo 5

Conclusão

Esse trabalho teve como propósito o estudo e implementação de abordagens algorítmicas para um dos problemas mais importantes da área de *Geometria Computacional*, nomeadamente, a triangulação de um conjunto de pontos no espaço euclidiano bidimensional, por meio da técnica de Delaunay. Considerando a amplitude dos algoritmos existentes, optamos por aqueles pertencentes ao paradigma de *inserção*, também referenciado na literatura por *incremental* [3]. As maiores vantagens dos algoritmos pertencentes a essa classe, são: em primeiro lugar, a sua simplicidade algorítmica e, em segundo, a possibilidade deles serem facilmente generalizados para outras dimensões, em particular, o R^3.

5.1 Futuros Trabalhos

Essa seção apresenta um conjunto de funcionalidades que poderiam ser incorporadas ao presente sistema potencializando ainda mais o desenvolvimento de novas aplicações. Para facilitar e orientar as atividades a serem elaboradas, essas serão colocadas em ordem de prioridade.

5.1.1 Estrutura de Dados DAG

A complecidade computacional do processo de triangulação de Delaunay por meio dos métodos do inserção são dependentes da etapa de localização do triângulo que contém o próximo ponto a ser inserido. Nesse trabalho, implementamos duas abordagens para esse propósito, ambas descritas na seção 4.2.2. Entretanto, uma alternativa bem mais eficiênte é o uso da estrutura de dados

DAG(Directed Acyclic Graph)[15].

Embora o estudo dessa complexa estrutura de dados hierárquica tenha sido realizado e possa ser observado na seção 4.2.2, não foi possível finalizar sua implementação. Nesse sentido, essa é uma das tarefas mais imediatas na continuidade desse trabalho e que irá redundar num enorme ganho de qualidade, já que o tempo de busca para cada ponto se reduz a $O(n \log n)$, em contraste com a que usamos atualmente, que possui complexidade $O(\sqrt{n})$.

Vale salientar ainda que a busca ou a localização de pontos numa subdivisão planar é um tópico relevante de pesquisa na área de Geometria Computacional[3],[27], [10] e, encontra inúmeras aplicações, como por exemplo, em Sistemas Geográficos de Informação.

5.1.2 Implementação de outras abordagens algorítmicas

A triangulação de Delaunay pode ainda ser obtida a partir de algoritmos pertencentes aos paradigmas de divisão-e-conquista(divide-and-conquer) [8], [15], [20] e varredura por linhas (plane-sweep) [12], além daqueles baseados no método de inserção que já foi implementado[15], [6]. Seria importante que essas outras abordagens algorítmicas fossem implementadas não somente para robustecer o sistema em termos de técnicas, mas também para viabilizar a implementação de abordagens denominadas híbridas, na medida em que combinam idéias de diferentes métodos. Atualmente, essas abordagens têm se mostrado extremamente competitivas. Um outro aspecto interessante seria a possibilidade de realizar um estudo exaustivo de comparação entre os diferentes algoritmos para uma ampla gama de conjunto de pontos.

5.1.3 Modelagem digital de terrenos

Conforme discutido no capítulo 3, uma aplicação importante da triangulação de Delaunay é junto a área de Modelagem Digital de Terrenos(MDT) [11], [31], [21], [17] onde a partir de um conjunto de pontos amostrados sobre o relevo de uma determinada área, é possível reconstruir a sua respectiva superfície. De posse de um modelo digital do terreno, uma série de operações pode ser realizada sobre o mesmo, como por exemplo, o dimensionamento de volumes de terras a serem removidos/deslocados em construção de túneis e estradas, projetos de construção de barragens e cálculos de áreas alagadas, etc. Esse projeto está muito próximo de ser iniciado, na medida em que um aluno do Programa de Pós-Graduação em Ciências Cartográficas de nossa Unidade, deve elaborar sua dissertação nessa temática e, certamente irá utilizar parte da

implementação que foi desenvolvida nesse trabalho de conclusão de curso.

5.1.4 Reconstrução de Curvas

Uma das abordagens para se reconstruir uma curva no plano [7], faz uso simultâneo das técnicas de Delaunay e do diagrama de Voronoi. Nesse sentido, os resultados já obtidos a partir desse projeto, devidamente integrados ao projeto de um outro aluno que implementou técnicas para a construção do diagrama de Voronoi, certamente estará entre as prioridades de projeto a ser elaborado num futuro próximo. Dessa forma, parte dos resultados obtidos nesse trabalho, também estará colaborando para que outros projetos ser desenvolvidos.

Muitas outras atividades em R^2 poderão ser elaboradas a partir dos resultados já disponíveis, mas julgo que essa pequena relação seja suficiente para mostrar a utilidade o e a abrangência do presente trabalho.

Bibliografia

[1] BAUMGART, B.G. **Geometric modeling for computer vision**. 1974. 463. Relatório técnico - Stanford Artificial Intelligence Laboratory, Computer Sciences Department, Stanford University, CA.

[2] BAUMGART, B.G. **Windge-edge polyhedron representation**. 1972. 320. Relatório Técnico - Computer Science Department, Standford University, Palo Alto, CA.

[3] BERG, M. K.; OVERMARS, M.V.; SCHWARZKOPF, M. O. **Computational geometry**: algorithms and applications. Springer-Verlag, 1977.

[4] BOWYER, A.; WOODWARK, J. **A programmer's geometry**. Butterworths, 1983.

[5] BRAID, I. C.; HILLYARD, R. C.; STROUD, I. A. Stepwise construction of polyhedrain geometric modeling. In: BRODLIE, K. **Mathematical methods in Computer Graphics and Design**. London: Academic Press, 1980. p. 123-141.

[6] BOISSONNAT, J.D.; YVINEC, M. **Algorithmic Geometry**, Cambridge University Press, 1998. 519 p.

[7] DEY, T.K. **Curve and surface reconstruction**:algorithms with mathematical analysis. Cambridge:Cambridge University Press, 2006, 229p.

[8] DWYER, R.A. A faster divide-and-conquer algorithm for constructiong delaunay triangulations. **Algorithmica**, v. 2, p. 137-151, 1987.

[9] EDELSBRUNNER, H. *Geometry and topology for mesh generation*. Cambridge University Press, 2001.

[10] EDELSBRUNNER, H. **Algorithms in Combinatorial Geometry**. Springer-Verlag, Germany, 1987.

[11] EL-SHEIMY, N.; VALEO, C.; HABIB, A. **Digital terrain modeling**: acquisition, manipulation, and applications. Boston: Artech House, 2005, 257p.

[12] FORTUNE, S. A sweepline algorithm for Voronoi Diagrams.**Algorithmica**, v. 2, p. 153-174, 1987.

[13] GLASSNER, A.S. Maintaining winged-edge models. In *Graphics Gems IV*, vol 2, Academic Press, INC., 1991, p. 191-200.

[14] GREEN, P.J.; SIBSON, R. Computing Dirichlet tessellations in the plane. **Computer journal**, v. 2, n. 21, p. 168-173, Cambridge, mai. 1978.

[15] GUIBAS, L.J.; KNUTH, D.; SHARIR, M. Randomized incremental construction of Delaunay ans Voronoi diagrams. **ACM Transactions on Graphics**. v. 4, n. 2, p. 75-123, 1985.

[16] GUIBAS, L.J.; STOLFI, J. Primitives for the manipulation of general subdivisions and computational of Voronoi diagrams.**ACM Transactions on Graphics**, v. 4, n. 2, p. 74-123, abr. 1985.

[17] HJELLE, Ø.; DÆHLEN, M. **Triangulation and applications**. Spring-Verlag Berlin Heidelberg, 2006.

[18] KIRKPATRICK, D.G. Optimal search in planar subdivisions. **SIAM Journal of Computing**, v. 12, n. 1, p. 28-35, 1983.

[19] LAWSON, C. Generation of a Triangulation Grid with Applications to Contour Plotting. 1972. 299. Relatório Técnico - Jet Propulsion Laboratory, California Institute os Technology.

[20] LEE, D.T.; SCHACHTER, B.J. Two algorithms for constructing Delaunay triangulations. **International Journal of Computer and Information Science**, Plenum Press, v. 3, n. 9, p. 219-242, Jun. 1980.

[21] LI, Z.; ZHU, Q.; GOLD, C. **Digital terrain modeling**: principles and methodology. Boca Raton: CRC Press, 2005, 323p.

[22] MUSIN, O.R. Properties of the Delaunay triangulation. In: PROCEEDINGS OF THE ANNUAL SYMPOSIUM ON COMPUTATIONAL GEOMETRY, 13, 1997, Nice, France, p. 424-426.

[23] MATULA, D.W.; SOKAL, R.R. Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane. **Geographic Anal**. v. 12, p. 205-222, 1980.

[24] O'ROURKE, J. Voronoi Diagrams. In: **Computational Geometry in C**. 2 ed. Cambridge: Cambridge University Press, 1994, p. 161-163.

[25] PITERI, M.A.; ALMEIDA, J.P.B.M. Hierarchical 2D mesh generation using a topological data structures. In: OLIVEIRA, E.R.A. BENTO, J. **Education, Practice an Promotion of Computational Methods in Engineering Using Small Computers**. 1995, v. 2, p. 981-986.

[26] PITERI, M.A. **Geração automática de malhas hierárquico-adaptativas em domínios bidimensionais e tridimensionais**. 1999. 227 p. Tese(Doutorado em Engenharia)-Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa.

[27] PREPARATA, F.P.; SHAMOS, M. I. **Computacional geometry**: an introduction. Springer-Verlag, 1985.

[28] SHAMOS, M.; Hoey, D. Closest-point problems. In: PROCEEDING OF THE IEEE ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 60, 1975, Berkeley, California, p. 151-162.

[29] SIBSON, R. Locally equiangular triangulations. **The computer Journal**, v. 21, n. 3, p. 243-245, 1978.

[30] SOKAL, R.R; GABRIEL, K.R. A new statistical approach to geographic variation analysis. **Systematic Zoology**, v. 18, p. 259-270, 1969.

[31] SNOOK, G. **Real Time 3D Terrain Engines using C++ and DirectX9**.Charles River Media, INC., 2003.

[32] SZWARCFITER, J.L. **Grafos e algoritmos computacionais**, Rio de Janeiro, Editora Campus, 1984, 216p.

[33] WATSON, D.F. Computing the n-dimensional Delaunay Tesselation with Application to Voronoi Polytopes.**Computer Journal**, v. 24, n. 2, p. 167-172, 1981.