Generalizações do Diagrama de Voronoi
construídas através de Cônicas
no Plano Projetivo Orientado
e suas Visualizações

Guilherme Albuquerque Pinto

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Generalizações do Diagrama de Voronoi construídas através de Cônicas no Plano Projetivo Orientado e suas Visualizações

Guilherme Albuquerque Pinto[1]

Março de 1998

Banca Examinadora:

- Prof. Dr. Pedro Jussieu de Rezende (IC–UNICAMP) (Orientador)
- Prof. Dr. Luiz Henrique de Figueiredo (LNCC–CNPq)
- Prof. Dr. Cid Carvalho de Souza (IC–UNICAMP)
- Prof. Dr. Alexandre Xavier Falcão (IC–UNICAMP) (Suplente)

[1]Apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (proc. 139084/96-7) e da Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP (proc. 96/09738-5)

Generalizações do Diagrama de Voronoi construídas através de Cônicas no Plano Projetivo Orientado e suas Visualizações

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Guilherme Albuquerque Pinto e aprovada pela Banca Examinadora.

Campinas, 17 de março de 1998.

Prof. Dr. Pedro Jussieu de Rezende
(IC–UNICAMP) (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Tese de Mestrado defendida e aprovada em 13 de março de 1998 pela Banca Examinadora composta pelos Professores Doutores

Prof. Dr. Luiz Henrique de Figueiredo

Prof. Dr. Cid Carvalho de Souza

Prof. Dr. Pedro Jussieu de Rezende

Resumo

Esta dissertação discute diagramas de Voronoi no plano projetivo orientado, um espaço geométrico que propicia vantagens computacionais tanto na representação quanto na construção dos diagramas. Esses diagramas, resumidamente, agregam informação de proximidade para um conjunto de objetos no espaço e estão entre as estruturas mais estudadas na Geometria Computacional com aplicações em diversas ciências.

Apresentamos um algoritmo incremental simples, baseado somente no conceito de orientação, para construir o diagrama de pontos e, também, o de pontos com peso aditivo. Esse último e algumas outras generalizações do diagrama de Voronoi possuem arcos de cônicas entre suas arestas. Para obter a visualização dos diagramas, estudamos as cônicas naquele espaço e propomos uma representação para arcos que unifica as três classes de cônicas afins no plano. O trabalho se completa com a implementação do algoritmo e o aprimoramento de dois visualizadores para os modelos plano e esférico do plano projetivo orientado, com a inclusão de cônicas, que permite a completa visualização dos diagramas e das vantagens desse espaço geométrico.

Abstract

This dissertation discusses Voronoi diagrams on the oriented projective plane, a geometric space which gives computational advantages in the representation as well as in the construction of the diagrams. These diagrams, in short, aggregate proximity information for a set of objects in the space and are among the best studied structures in Computational Geometry with applications in several sciences.

We present a simple incremental algorithm, based only on the concept of orientation, to construct the diagram of points and, also, the additively weighted diagram. The latter and some other generalizations of the Voronoi diagram include conic arcs among their edges. To achieve the visualization of the diagrams, we study the conics in that space and propose a representation for arcs that unifies the three classes of affine conics in the plane. The work also includes the implementation of the algorithm and the update of two visualizers for the planar and spherical models of the oriented projective plane, in order to include conics, that allow for the complete visualization of the diagrams and the advantages of this geometric space.

Agradecimentos

Aos professores da minha graduação Arnaldo Vieira da Rocha Filho e Mauro Pereira de Mello, e ao meu ex-chefe Luis Fernando Seixas de Oliveira pelo apoio inicial;

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (processo 139084/96-7) e à Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP (processo 96/09738-5) pelo apoio financeiro *sine qua non*;

A César Nivaldo Gon pela valiosa herança dos visualizadores do projeto GeoPrO;

A Alexandre Volpim e Frederico Guth pelo gratificante trabalho conjunto;

Ao meu orientador professor Pedro Jussieu de Rezende, um agradecimento especial, pelas muitas contribuições para este trabalho e para a minha formação; e sobretudo pela caríssima amizade.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O diagrama de Voronoi foi uma das primeiras estruturas geométricas estudadas sob aspectos computacionais devido à sua grande importância para diversas ciências [4]. Antes mesmo do termo Geometria Computacional ser popularizado, vários algoritmos já haviam sido obtidos para calculá-lo. Desde então, muitos algoritmos ótimos foram descobertos e, ao menos para o caso planar, todos os aspectos combinatoriais do diagrama foram estabelecidos.

Atualmente, a pesquisa tem se voltado para as generalizações do diagrama e para técnicas de implementação robusta. A implementação de algoritmos geométricos é, em geral, dificultada por casos especiais e por problemas numéricos. Como veremos, o diagrama de Voronoi construído no plano Euclidiano possui regiões e arestas que se estendem até o infinito. Essas arestas representam um caso especial a ser tratado na construção e representação explícita dos diagramas. Nesse contexto, este trabalho propõe uma nova maneira de representar o diagrama onde as arestas infinitas não trazem qualquer problema. Essa representação é conseguida pela simples construção do diagrama, não no plano Euclidiano $\mathbb{E}^2$, mas no plano projetivo orientado $\mathbb{T}^2$, que possui todas as características interessantes do plano Euclidiano mais as vantagens do plano projetivo clássico $\mathbb{P}^2$.

Esta foi a nossa motivação inicial para o tema. Entretanto, outros aspectos teóricos interessantes foram observados, ainda na definição da proposta de trabalho, que se mostraram realmente válidos e fazem parte das contribuições da dissertação. Um exemplo é a conexidade do diagrama de pontos com peso aditivo no $\mathbb{T}^2$ (seção 4.3), ao contrário do plano Euclidiano, onde ele pode ser desconexo. Esse fato também contribui para diminuir as dificuldades na representação do diagrama.

Antes de prosseguirmos com a introdução à dissertação, traçaremos um histórico da fase inicial da pesquisa com o intuito de contextualizar melhor o que será apresentado em todo o texto.

Dentro daquele espírito de buscar novas ferramentas para implementação robusta de

algoritmos geométricos, tínhamos a intenção de estudar técnicas de perturbação simbólica para tratamento de degenerações em algoritmos [15]. Na literatura, é possível encontrar essas técnicas aplicadas a primitivas envolvendo pontos, retas e círculos. Pretendíamos estudá-las para primitivas envolvendo cônicas, que são naturalmente o próximo passo. Além dessa parte mais conceitual, a pesquisa envolveria trabalho no desenvolvimento do ambiente de visualização distribuída GeoPrO [18], que provê suporte para o $\mathbb{T}^2$, e é também um esforço para prover ferramentas que facilitem a implementação, demonstração e depuração de algoritmos geométricos.

Boa parte do estudo dirigido, que leva à proposta de trabalho, foi dedicado a uma revisão bibliográfica para identificar os problemas e algoritmos que, de alguma maneira, envolvessem cônicas e que pudessem, eventualmente, se beneficiar da aplicação das técnicas de perturbação simbólica. Entramos em contato, assim, com as generalizações do diagrama de Voronoi que possuem cônicas. Em um dado momento, percebemos que poderia ser muito interessante explorar a construção de diagramas de Voronoi no $\mathbb{T}^2$ e que não havia muitos indícios da utilidade ou viabilidade da aplicação de perturbação simbólica a primitivas envolvendo cônicas.

A meta do trabalho tornou-se, então, o estudo dos diagramas no $\mathbb{T}^2$ e as suas visualizações com o auxílio dos visualizadores do ambiente GeoPrO. O estudo envolveu a análise das características dos diagramas e a busca por algoritmos que os construíssem de maneira uniforme e transparente, para que o $\mathbb{T}^2$ não se tornasse um novo problema para os implementadores. A visualização envolveu o estudo das cônicas no $\mathbb{T}^2$ para encontrar uma representação viável para arcos; e a inclusão do suporte a cônicas nos visualizadores do GeoPrO a partir da representação.

Uma das funções mais interessantes de ferramentas de visualização é o auxílio à depuração na implementação de algoritmos. Como sabíamos que os diagramas estavam bem definidos no $\mathbb{T}^2$ (faltava encontrar os algoritmos), resolvemos pesquisar a representação de cônicas e aprimorar os visualizadores antes de pesquisar e implementar os algoritmos. Esta estratégia foi muito bem sucedida e os algoritmos foram implementados com uma ajuda visual que acelerou o processo e foi responsável pela rápida identificação de uma peculiaridade no caso base (seção 5.3.2).

1.1 Organização da Dissertação

O capítulo 2 faz uma introdução a diagramas de Voronoi e algumas de suas generalizações. Pretende dar uma visão geral da bibliografia da área, apresentar algumas aplicações e ressaltar os aspectos que são diretamente influenciados pela construção no $\mathbb{T}^2$. O capítulo 3 contém uma breve introdução ao $\mathbb{T}^2$. São apresentados os conceitos mais importantes e alguma notação usada posteriormente.

A parte teórica principal da dissertação é composta pelo capítulo 4, que apresenta o estudo sobre cônicas e diagramas no $\mathbb{T}^2$, e pelo capítulo 5, que apresenta em detalhes o algoritmo implementado. O capítulo 6 finaliza a dissertação com a parte prática das implementações no ambiente GeoPrO e algumas imagens da visualização dos diagramas nos modelos plano e esférico do $\mathbb{T}^2$.

Esta dissertação pretende apresentar todos os aspectos relevantes no que diz respeito a diagramas de Voronoi e ao $\mathbb{T}^2$. No entanto, assumimos familiaridade com conceitos e notação básicos de análise de algoritmos, programação estruturada de computadores, geometria Euclidiana [32, 10] e topologia de espaços métricos [10].

Capítulo 2

Diagrama de Voronoi e Generalizações

The usefulness of the Voronoi diagram for solving a large number of problems, the fact that it can be constructed efficiently, and maybe also its aesthetically pleasing appearance subsequently kindled the interest of many researchers.

— H. Edelsbrunner e R. Seidel [12]

A estrutura geométrica a que chamamos de diagrama de Voronoi [4, 6] pode ser encontrada na literatura também com os nomes de *Dirichlet* ou *Thiessen Tessellations*. Quando é construída no interior de um polígono simples recebe os nomes de *medial axis* ou *internal skeleton*. Conceitualmente, dado um conjunto de objetos em um espaço, o diagrama é a divisão do espaço em regiões, cada uma associada a um objeto do conjunto, tal que todos os pontos de uma região estão mais próximos do seu objeto associado do que de todos os outros.

2.1 Diagrama de Pontos

Seja $\mathcal{S}$ um conjunto finito de pontos, chamados sítios, no plano Euclidiano $\mathbb{E}^2$. Vamos assumir, por simplicidade, que os sítios estão em posição geral, ou seja, não há quatro sítios co-circulares em $\mathcal{S}$. Para $p,q \in \mathcal{S}$ definimos

$$H_{pq} = \{x \in \mathbb{E}^2 | d(x,p) \leq d(x,q)\},$$

onde d é a distância Euclidiana usual. A região de Voronoi de um sítio p é, então,

$$R_p = \bigcap_{q \in \mathcal{S} \setminus \{p\}} H_{pq}.$$

Note que H_{pq} é um semiplano limitado por uma reta; portanto, R_p é sempre convexa.

De acordo com [37] vamos definir o diagrama de Voronoi de $\mathcal{S}$, DVor($\mathcal{S}$), como sendo o conjunto de pontos do E^2 que pertencem a *mais de uma* região de Voronoi. Uma aresta do diagrama é a interseção de duas regiões (um segmento de reta ou uma semi-reta) e um vértice é a interseção de três regiões (a posição geral impede que mais de três regiões tenham interseção). A figura 2.1(a) mostra, como exemplo, um diagrama para um conjunto de dez sítios.

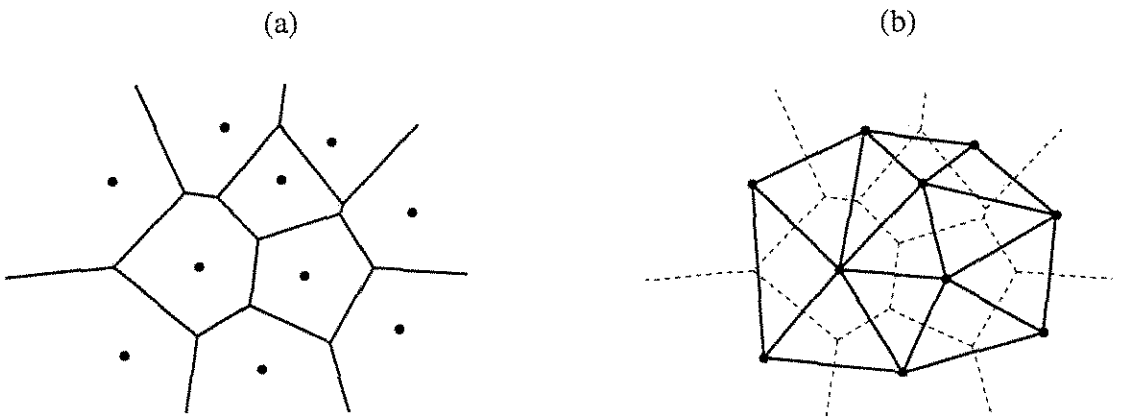

Figura 2.1: O diagrama de Voronoi e a triangulação de Delaunay

As regiões infinitas do DVor são aquelas associadas aos sítios da envoltória convexa de $\mathcal{S}$ (veja [32]). As arestas infinitas são as interseções entre pares dessas regiões. É razoável esperar que a representação, topológica e geométrica, e a manipulação dessas arestas não seja uniforme em relação às arestas finitas. Isso porque a representação mais usual para a topologia de subdivisões planares é a por contorno (por exemplo [22]), onde as arestas são representadas explicitamente e as regiões implicitamente pelas relações de incidência das arestas.

Uma maneira interessante de evitar esses problemas é trabalhar com o dual do DVor [22]. Esse dual, chamado diagrama de Delaunay, é obtido pela conexão, por um segmento de reta, dos sítios cujas regiões possuem uma aresta em comum, figura 2.1(b). Quando o conjunto $\mathcal{S}$ está em posição geral, o dual é chamado de triangulação de Delaunay. Essa triangulação possui propriedades matemáticas interessantes [32] e pode ser vista como uma construção implícita do DVor onde não é necessário calcular e trabalhar com os vértices do diagrama.

2.1.1 Complexidade

Dentre as questões importantes do ponto de vista computacional a respeito do DVor está a quantidade de arestas e vértices, que influi diretamente sobre a quantidade de memória necessária para armazenar o diagrama e o tempo necessário para calculá-lo. A dualidade da triangulação de Delaunay permite obter facilmente essas cotas [21]: $e = 3(n-1) - h$ e $v = 2(n-1) - h$, onde e é o número de arestas, v o número de vértices, n o número de sítios e h o número de sítios na envoltória convexa. Como a quantidade h é desconhecida antes da construção do diagrama, muitos artigos apresentam as cotas $e \leq 3n - 6$ e $v \leq 2n - 4$.

Antes de considerarmos os algoritmos para calcular o diagrama, vale a pena comentar sobre quão difícil é calculá-lo. No modelo computacional RAM (*random-access machine*) [32], são necessárias $\Omega(n \log n)$ operações para construir o diagrama. Essa cota vem da possibilidade de se obter a envoltória convexa com $O(n)$ operações do DVor e de se obter a ordenação de um conjunto de números reais com $O(n)$ operações da envoltória convexa. É sabido que no modelo RAM a ordenação requer $\Omega(n \log n)$ operações [32]. Felizmente, vários algoritmos conseguem construir o DVor em $O(n \log n)$ operações.

2.1.2 Algoritmos e representações

Podemos dividir os algoritmos para diagramas de Voronoi em três classes, segundo a técnica empregada: incremental [19, 3, 22], divisão e conquista [26, 22, 37, 40, 33] e varredura do plano (*plane-sweep*) [16, 34]. Das três técnicas, somente a incremental não é capaz de construir o diagrama no plano em $O(n \log n)$ operações; no pior caso, ela pode atingir $O(n^2)$ operações. Isso porque há configurações para as quais é sempre possível inserir um novo sítio cuja região possua arestas em comum com todas as regiões do diagrama já construído, de forma que a atualização demanda no mínimo n operações, onde n é o número de sítios. Apesar desta desvantagem, os algoritmos incrementais são em geral muito fáceis de implementar e muito adequados quando a aplicação necessita atualizar o diagrama com a inserção de novos sítios.

No capítulo 5 apresentaremos um algoritmo incremental que atinge o objetivo de construir o diagrama de pontos e de pontos com peso aditivo no $\mathbb{T}^2$ de maneira uniforme e transparente. A idéia geral está baseada, como a maioria dos outros algoritmos incrementais para o plano, no trabalho de Green e Sibson [19]. Vamos apresentar agora, em mais detalhes, esse algoritmo para dar uma intuição maior sobre a técnica incremental.

Dado um DVor de $n - 1$ sítios, a inclusão de um novo sítio k consiste de duas fases:

1. Encontrar, no diagrama, o sítio p mais próximo de k. A nova região de p com certeza possui uma aresta comum com a região de k, ou seja, eles serão vizinhos no novo diagrama.

2. A partir de p, encontrar os outros vizinhos de k e atualizar o diagrama.

As duas fases podem ser implementadas de várias maneiras. No artigo de Green e Sibson, o sítio p é encontrado por um caminhamento no diagrama a partir de um dado sítio, sempre para o seu vizinho que esteja mais próximo de k até que o sítio p seja atingido. A sugestão é que o caminhamento comece por um sítio aproximadamente central, o que garante custo esperado $O(\sqrt{n})$ para a primeira fase. Os outros vizinhos são encontrados como sugere a figura 2.2. A partir de p, o vizinho seguinte é encontrado, em sentido horário, pela identificação da aresta da região de p que é atravessada pelo bissetor do segmento entre p e k, assim sucessivamente até que o vizinho seguinte volte a ser p.

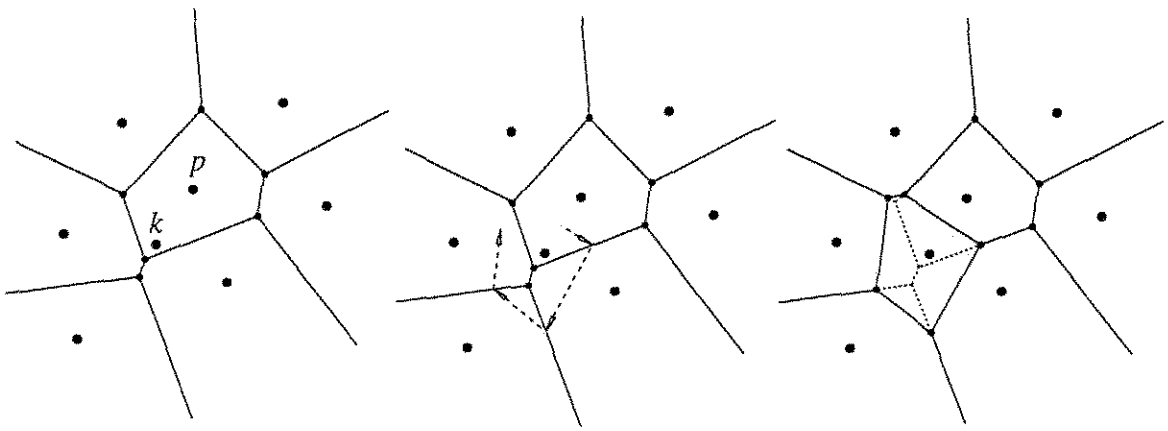

Figura 2.2: Algoritmo de Green e Sibson para construção incremental do DVor

A descrição desse procedimento oferece oportunidade para verificarmos que as regiões e arestas infinitas constituem um caso especial. Sempre que o novo sítio k for inserido na envoltória convexa do conjunto, haverá arestas infinitas que não são tratadas pelo procedimento geral. A solução do artigo original é, simplesmente, eliminar as arestas infinitas construindo o diagrama restrito a um polígono convexo.

No que se refere à representação da topologia do diagrama, duas outras soluções são comuns: a introdução de arestas virtuais conectando arestas infinitas adjacentes [40, 37] e de um vértice virtual conectando todas as arestas infinitas [34, 22].

2.2 Generalizações

Na literatura costuma-se chamar de generalizações todos os diagramas de Voronoi que não são o DVor (de pontos no $\mathbb{E}^2$). As generalizações podem ser obtidas, basicamente, de três maneiras [4, 6]: trocando a função de distância d na definição; trocando os sítios por objetos mais complexos e trocando o espaço geométrico onde o diagrama é construído. A

construção dos diagramas no $\mathbb{T}^2$ a ser apresentada no capítulo 4 poderia, em si, ser chamada de uma generalização. No entanto, veremos que todas as propriedades geométricas dos diagramas no $\mathbb{E}^2$ se mantêm; por isso o diagrama de pontos com peso aditivo, por exemplo, está sendo chamado de generalização construída no $\mathbb{T}^2$.

Apresentaremos o diagrama de vizinho mais distante e o diagrama na esfera, que estão relacionados com a construção do DVor no $\mathbb{T}^2$. Depois veremos com mais detalhes o diagrama de pontos com peso aditivo que, das generalizações contendo cônicas, foi a que obteve os maiores benefícios da construção no $\mathbb{T}^2$ e que é construída pelo algoritmo do capítulo 5.

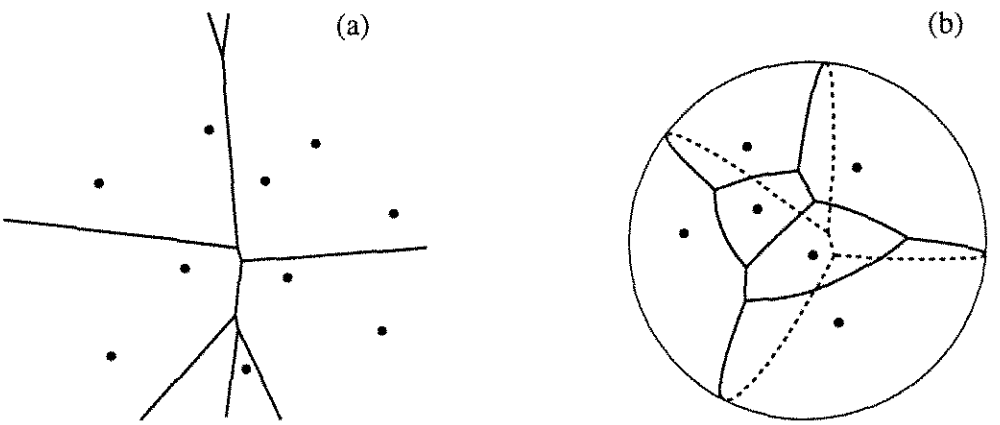

Figura 2.3: Diagrama de vizinho mais distante e diagrama na esfera

O DVor, pela sua definição, é chamado de diagrama de vizinho mais próximo. Uma das primeiras generalizações estudadas [7, 4] foi o diagrama de vizinho mais distante que divide o plano em regiões associadas aos sítios, tal que para todos os pontos de uma região o sítio associado é o mais distante. A figura 2.3(a) mostra esse diagrama para o mesmo conjunto de sítios da figura 2.1. Esse diagrama pode ser obtido apenas mudando-se o sentido da desigualdade na definição de H_{pq}:

$$H_{pq} = \{x \in \mathbb{E}^2 | d(x,p) \geq d(x,q)\}.$$

Somente os sítios da envoltória convexa possuem regiões nesse diagrama, o que é bastante intuitivo. As cotas para os números de arestas e vértices também podem ser facilmente obtidas [4]: $e = 2h - 3$ e $v = h - 2$, onde h é novamente o número de sítios na envoltória convexa.

Como podemos obter a envoltória convexa deste diagrama em tempo linear, ele requer $\Omega(n \log n)$ operações no modelo RAM. Há pelo menos dois algoritmos, ambos baseados na técnica de divisão e conquista, que conseguem construí-lo em $O(n \log n)$ operações [32, 22].

Outra generalização imediata do DVor é a sua construção na esfera do $\mathbb{E}^3$ [28, 3]. Um ponto interessante aqui é que não existem regiões e arestas infinitas, de modo que a representação é sempre uniforme, veja figura 2.3(b). Na esfera, as cotas exatas para o número de arestas e vértices não dependem de h [3]: $e = 3n - 6$ e $v = 2n - 4$.

O artigo [3] apresenta um algoritmo incremental, baseado no original de Green e Sibson, para construir esse diagrama que não precisa de procedimentos especiais quando o novo sítio é inserido na envoltória convexa. Vale a pena notar entretanto que, apesar da semelhança, não é possível usar essa construção para obter o DVor. Não podemos projetar os sítios sobre a esfera, calcular o diagrama e projetá-lo de volta para o plano.

2.2.1 Diagrama de pontos com peso aditivo

No diagrama de pontos com peso aditivo, DVorW, cada sítio $p \in \mathcal{S}$ recebe um peso $w(p) \in \mathbb{R}$, que é subtraído na definição de H_{pq}:

$$H_{pq} = \{x \in \mathbb{E}^2 | d(x,p) - w(p) \leq d(x,q) - w(q)\}.$$

Deste modo, H_{pq} torna-se um semiplano limitado por um ramo de hipérbole [16] e R_p torna-se estrelada (*star-shaped*) em relação ao sítio e não mais necessariamente convexa. Uma região é estrelada se ela possui pelo menos um ponto tal que os segmentos ligando este ponto a todos os outros pontos da região estão totalmente contidos nela.

Alguns artigos que tratam desse diagrama somam o peso ao invés de subtraí-lo na definição. Isso não altera as propriedades pois podemos multiplicar os pesos por -1 para obter o diagrama gerado pela outra definição, o que equivale a inverter a ordenação relativa dos pesos dos sítios. Outro fato relevante sobre a definição é que se somamos uma constante a todos os pesos o diagrama não se altera. Esse fato permite um artifício no cálculo dos vértices do diagrama (seção 5.3.1).

A figura 2.4(a) mostra um exemplo com sete sítios onde os números indicam seus respectivos pesos. O DVorW é topologicamente mais complexo do que o DVor pois podem haver regiões (de sítios dentro da envoltória convexa) com apenas duas arestas e duas regiões com mais de uma aresta em comum. No entanto, as cotas para os números de arestas e de vértices encontradas na literatura [34] são quase as mesmas: $e \leq 3n - 5$ e $v \leq 2n - 4$.

Quando os pesos são todos positivos podemos imaginar os sítios como sendo círculos de raio igual aos seus pesos. Se, então, os círculos não se interceptam o DVorW é equivalente ao diagrama de Voronoi de círculos, veja figura 2.4(b). Essa equivalência será explorada no capítulo 5.

O DVorW possui uma característica que dificulta ainda mais sua representação: ele pode ser desconexo [37], veja figura 2.4(c). Veremos no capítulo 4 que o DVorW construído

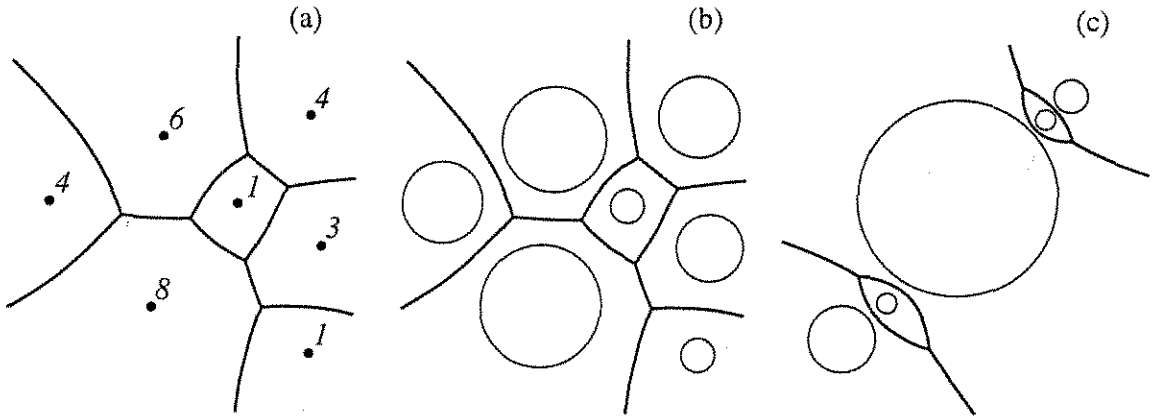

Figura 2.4: Diagrama de pontos com peso aditivo

no $\mathbb{T}^2$ é sempre conexo. Sempre que o diagrama de vizinho mais próximo é desconexo o de vizinho mais distante (definido com $d(x,p)+w(p)$) possui regiões desconexas [33]. Isso dá uma pista sobre o porquê dele ser conexo no $\mathbb{T}^2$.

É fácil ver que o DVorW se reduz ao DVor quando todos os pesos são iguais. Portanto também são necessárias $\Omega(n \log n)$ operações para construí-lo. Dois algoritmos fazem isso em $O(n \log n)$ operações: por divisão e conquista [40] e por varredura do plano [16].

A presença das cônicas Existem pelo menos mais três generalizações do DVor que possuem cônicas: o diagrama de segmentos de reta [26, 16, 9, 40], com arcos de parábola; o diagrama de pontos com peso multiplicativo [5], com arcos de círculo; e o diagrama de arcos de círculos [40], com arcos de elipse e de hipérbole. Para todos estes diagramas, as arestas que se estendem até o infinito são ou segmentos de reta, como no DVor, ou arcos de hipérbole, como no DVorW. O capítulo 4 discute com mais detalhes a construção do diagrama de segmentos de reta no $\mathbb{T}^2$. Infelizmente, esse diagrama pode ser desconexo tanto no $\mathbb{E}^2$ quanto no $\mathbb{T}^2$.

2.3 Aplicações

Podemos dizer que todas as aplicações dos diagramas de Voronoi se beneficiam, de uma forma ou de outra, das informações de proximidade que eles carregam. O diagrama define, por exemplo, quais são os "vizinhos naturais" de cada sítio, o que é útil na interpolação de dados discretos.

Vamos separar as aplicações em duas principais classes: quando os diagramas são usados como um passo para localização de pontos e quando o interesse reside no diagrama em si como recurso de simulação ou visualização.

Uma das primeiras motivações para o estudo do DVor foi, certamente, o problema de localização de pontos, que pode ser exemplificado com o problema das agências dos correios: dada a localização de todas as n agências dos correios em uma cidade e a localização de uma pessoa, dizer qual a agência mais próxima da pessoa. A solução trivial é computar todas as n distâncias e guardar a menor, o que pode ser feito em $O(n)$ operações. No entanto, se a pergunta for muito freqüente, vale a pena investir em uma estrutura de dados que permita responder à questão em menos operações. O DVor, aliado a técnicas de localização de pontos em subdivisões planares (por exemplo [11]), reduz o número de operações para $O(\log n)$. Várias aplicações em geografia e banco de dados, por exemplo, podem ser modeladas a partir desse problema simples [29, 4].

Entre as aplicações do segundo tipo, uma bastante citada, que motiva principalmente o estudo de diagramas para objetos mais complexos do que pontos, é o planejamento de movimento em robótica [40]. Construindo-se o diagrama de Voronoi dos obstáculos que um robô deve evitar, pode-se planejar um movimento, pelas arestas do diagrama, que maximize a menor distância do robô para os obstáculos.

Outra aplicação muito interessante é a simulação do crescimento de cristais [4]. Se várias fontes de cristal começam a crescer radialmente, ao mesmo tempo e à mesma velocidade numa superfície plana, o resultado final será exatamente o DVor. Como em uma situação prática os cristais não começam a crescer exatamente no mesmo momento, o DVorW é proposto como um modelo mais realista onde os pesos dos sítios refletem as diferenças de tempo do início do crescimento.

Para finalizar citamos ainda que o DVor pode ser usado, na própria geometria computacional, para encontrar: todos os vizinhos mais próximos; o par mais próximo; uma triangulação; a envoltória convexa e a árvore espalhada mínima [32]. O diagrama de vizinho mais distante pode ser usado para encontrar: todos os vizinhos mais distantes e o menor círculo envolvente [7].

Capítulo 3

O Plano Projetivo Orientado

Let us mention one more way of generalization: changing the underlying space. Ehrlich and Im Hof investigated the behavior of Voronoi regions in Riemann manifolds. Brown, Paschinger, and Yap observed that Voronoi diagrams on the sphere and on the torus, respectively, are closely related to their equivalents in the euclidean space of the same dimension. Diagrams on three-dimensional polyhedral surfaces and on the three-dimensional cone have been treated by Mount and by Dehne and Klein, respectively.

— F. Aurenhammer [4]

Uma desvantagem do plano Euclidiano $\mathbb{E}^2$ para computações geométricas é a ausência de pontos no infinito: duas retas possuem sempre uma interseção, a menos que sejam paralelas. A solução clássica para esse problema é o plano projetivo $\mathbb{P}^2$. Topologicamente, ele é composto de uma cópia do $\mathbb{E}^2$ mais uma reta no infinito. Assim, as retas que não se interceptam no $\mathbb{E}^2$ o fazem no $\mathbb{P}^2$ em um ponto no infinito. O problema é que, para uma dada reta, os pontos no infinito nas duas direções são o mesmo. Essa identificação de pontos do infinito em direções opostas transforma o $\mathbb{P}^2$ em um plano onde não se pode definir coerentemente conceitos básicos da computação geométrica, como orientação e convexidade.

No plano projetivo orientado $\mathbb{T}^2$ [38, 10] os pontos no infinito em direções opostas *não* estão identificados. Topologicamente, ele é composto de duas cópias do $\mathbb{E}^2$ mais uma reta no infinito, sendo equivalente à esfera do $\mathbb{E}^3$. O $\mathbb{T}^2$ une as vantagens do $\mathbb{P}^2$ com as do $\mathbb{E}^2$ para computações geométricas. Vamos apresentar alguma notação e os conceitos que serão usados nos capítulos posteriores. De tudo que será visto neste capítulo, somente as seções 3.2.1 e 3.2.2 não podem ser encontradas em [38] ou [10].

3.1 Conceitos Projetivos

No $\mathbb{T}^2$ nós utilizamos coordenadas homogêneas com sinal: $p = [x, y, w]$ *não* é idêntico ao ponto $[-x, -y, -w]$, que é denotado por $\neg p$. Eles são chamados de pontos antípodas. O conjunto de pontos com coordenada $w > 0$ é chamado de aquém e o conjunto com $w < 0$ de além. Os pontos do aquém e do além são referidos como pontos próprios. O conjunto dos pontos com $w = 0$ (com a exceção da tripla inválida $[0, 0, 0]$), que são referidos como pontos impróprios, é a reta no infinito Ω.

Há pelo menos dois modelos geométricos para o $\mathbb{T}^2$: o modelo plano, com o mapeamento usual para coordenadas Cartesianas $[x, y, w] \mapsto (x/w, y/w)$, e o modelo esférico com o mapeamento $[x, y, w] \mapsto (x, y, w)/\sqrt{x^2 + y^2 + w^2}$. Esses modelos estão claramente relacionados por projeção central, veja figura 3.1(a). Note que, no modelo plano, pontos antípodas são coincidentes mas podem ser imaginados como estando em lados opostos do plano (o aquém e o além). Todas as figuras seguem a convenção: os pontos são vazios e as linhas tracejadas quando estão no além para o modelo plano, e quando estão escondidos para o modelo esférico.

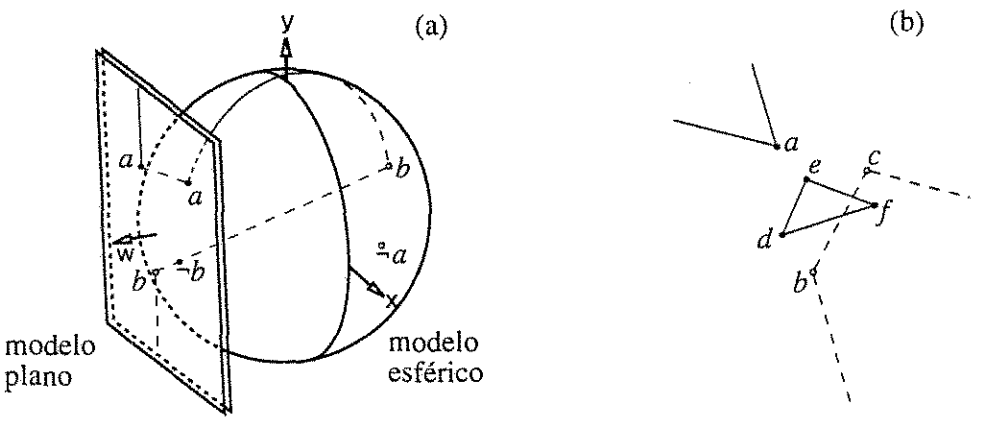

Figura 3.1: Modelos e convenções para o $\mathbb{T}^2$

As retas no $\mathbb{T}^2$ têm a mesma equação homogênea das retas em $\mathbb{P}^2$, ou seja, $ax + by + cw = 0$, onde a,b e c são constantes reais. Se multiplicarmos a equação por -1, obteremos a reta com orientação oposta $-ax - by - cw = 0$, que é composta pelo mesmo conjunto de pontos. A figura 3.2(b) mostra duas retas r e s no modelo esférico. Note que elas são grandes círculos da esfera e que separam o $\mathbb{T}^2$ em dois subespaços, ao contrário de retas no $\mathbb{P}^2$. Com isso nós podemos propriamente dizer quais são os lados direito e esquerdo da reta, baseado na sua orientação. A figura 3.2(a) mostra uma reta no modelo plano do $\mathbb{T}^2$ com seu lado esquerdo ressaltado. Note que a projeção central provoca uma aparente inversão no modelo plano.

O segmento entre dois pontos não coincidentes no $\mathbb{T}^2$ é definido, com a ajuda do modelo esférico, como sendo o conjunto de pontos do menor arco, dos dois arcos definidos pelos dois pontos, no grande círculo que os contém. Assim está bem definido no $\mathbb{T}^2$ o segmento entre um ponto próprio e outro impróprio. Na figura 3.1 podemos ver alguns segmentos nos modelos esférico, parte (a), e plano, parte (b). Note que o segmento entre um ponto do aquém e outro do além possui uma aparência incomum no modelo plano (veja segmento ab).

3.1.1 Operações com pontos e retas

A determinação de uma reta por dois pontos é chamada operação de *join* e denotada pelo símbolo $\vee$. A figura 3.2(a) mostra o *join* dos pontos a e b, $a \vee b$, que resulta na reta orientada de a para b. Note que no $\mathbb{T}^2$ duas retas passam por dois pontos não coincidentes ou antípodas. A operação $b \vee a$ gera a reta orientada opostamente.

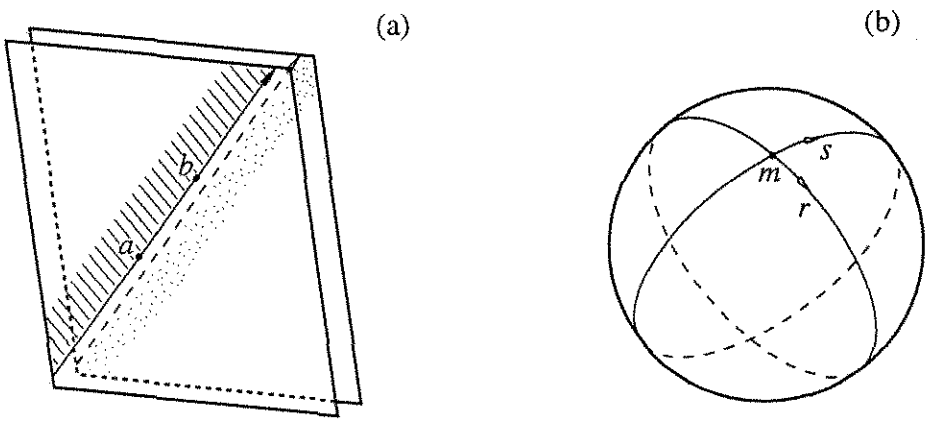

Figura 3.2: As operações *join* e *meet*

De maneira dual define-se o ponto determinado por duas retas com a operação *meet*, denotada pelo símbolo $\wedge$. No $\mathbb{T}^2$ duas retas não coincidentes ou antípodas sempre se interceptam em dois pontos. A figura 3.2(b) mostra como decidir qual dos dois pontos é $r \wedge s$, de acordo com a ordem dos operandos. Se $m = r \wedge s$, e nós movemos uma seta ao longo de r, concordando com sua orientação, então m é o ponto onde essa seta atravessa s passando do lado esquerdo para o direito.

A posição relativa de pontos e retas é decidida pelo operador $\diamond$. Essa operação resulta no sinal da substituição das coordenadas do ponto p na equação da reta r. Com isso

podemos dizer que:

$$p \text{ está } \left\{ \begin{array}{c} \text{à esquerda de} \\ \text{sobre} \\ \text{à direita de} \end{array} \right\} r \text{ se } p \diamond r = \left\{ \begin{array}{c} +1 \\ 0 \\ -1 \end{array} \right\}.$$

Na prática, quando a reta r é dada como $a \vee b$, podemos saber em que lado de r está um ponto p calculando diretamente o tradicional determinante que decide a orientação dos três pontos a, b e p no plano. Esse teste constitui uma das duas primitivas do algoritmo do capítulo 5:

Procedimento 1 `antiHorario( a, b, p )` *retorna* `true` *se* $p \diamond (a \vee b) = +1$ *e* `false` *caso contrário:*

```
real abp = ax(byc_w-cyb_w) + ay(b_wcx-c_wbx) + a_w(bxcy-cxby);
return ( abp > 0 );
```

3.2 Plano Euclidiano de Dois Lados

Até agora estivemos definindo conceitos invariantes sob projetividades. No entanto, os problemas em geometria computacional são normalmente definidos no $\mathbb{E}^2$, de modo que suas soluções dependem de conceitos como perpendicularidade, distância e outros conceitos que não são projetivos. Todos podem ser definidos no $\mathbb{T}^2$ se nós atribuímos um significado especial para a reta no infinito Ω. Assim, o $\mathbb{T}^2$ é usado para emular o $\mathbb{E}^2$ [38, capítulo 17], recebendo o nome de plano Euclidiano de dois lados.

Perpendicularidade Considere a reta Ω orientada tal que o aquém seja seu lado esquerdo e uma reta $r \neq \Omega$ qualquer. Nós dizemos que $\text{dir}(r) = r \wedge \Omega$, ou seja, a direção de r é o ponto onde r cruza Ω passando do aquém para o além. Dois pontos a e b do $\mathbb{T}^2$ são polares entre si quando suas coordenadas satisfazem $a_x b_x + a_y b_y + a_w b_w = 0$, ou seja, no modelo esférico, seus correspondentes vetores $(x, y, w)/\sqrt{x^2 + y^2 + w^2}$ são ortogonais. Chamamos $\text{norm}(r)$ o ponto impróprio polar a $\text{dir}(r)$ e contido no lado esquerdo de r (veja figura 3.5(b)). Dizemos que duas retas r e s são perpendiculares se $\text{dir}(r) = \text{norm}(s)$ ou $\text{dir}(s) = \text{norm}(r)$ (figura 3.5(b)).

3.2.1 Distância de dois lados

A distância entre dois pontos próprios pode ser definida pela expressão

$$\text{d}_{\mathbb{T}^2}(a, b) = \frac{\sqrt{(a_x b_w - b_x a_w)^2 + (a_y b_w - b_y a_w)^2}}{a_w b_w},$$

que é a versão com sinal da distância Euclidiana usual. Essa fórmula assume valores numéricos negativos quando aplicada a pontos de lados opostos do plano, e valores positivos para pontos do mesmo lado. No entanto, para definir as cônicas e os diagramas precisamos operar e comparar esses valores consistentemente. Isso é possível quando evitamos a divisão e associamos cada valor da distância a um ponto $[x, w]$ da reta projetiva orientada $\mathbb{T}^1$, fazendo $d_{\mathbb{T}^2} : \mathbb{T}^2 \times \mathbb{T}^2 \to \mathbb{T}^1_+$, definida como $d_{\mathbb{T}^2}(a, b) = [\sqrt{(a_x b_w - b_x a_w)^2 + (a_y b_w - b_y a_w)^2}, a_w b_w]$. A figura 3.3 mostra o $\mathbb{T}^1$ no modelo plano com as coordenadas w normalizadas para facilitar a comparação. Note que o numerador de $d_{\mathbb{T}^2}$ é sempre positivo, de modo que o contra-domínio da distância corresponde a uma metade do aquém mais uma metade do além. Esse subconjunto do $\mathbb{T}^1$ será chamado de $\mathbb{T}^1_+$ por analogia ao $\mathbb{R}_+$ que é o contra-domínio da distância Euclidiana. Desse modo, a distância assume pontos no aquém do $\mathbb{T}^1$ quando aplicada a pontos do mesmo lado do $\mathbb{T}^2$, e pontos no além do $\mathbb{T}^1$ para pontos em lados opostos do $\mathbb{T}^2$.

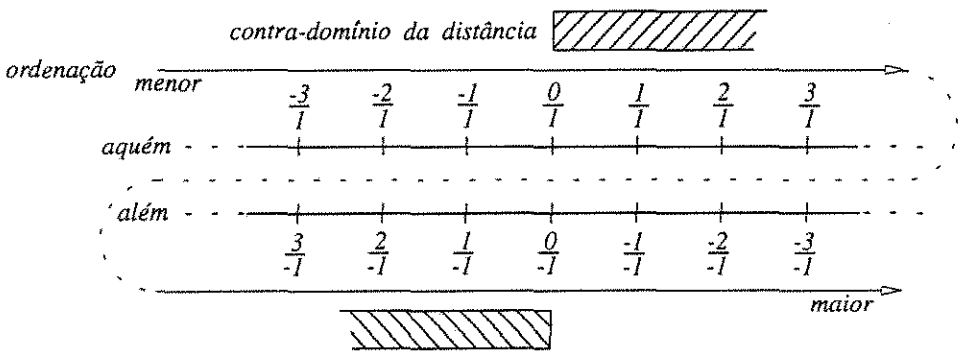

Figura 3.3: Ordenação dos pontos do $\mathbb{T}^1$

Para comparar dois pontos $a,b \in \mathbb{T}^1$ usamos a regra: $a <_{\mathbb{T}^1} b$ se e somente se a_w e b_w têm sinais opostos e $a_w > 0$; ou a_w e b_w têm sinais iguais e $a_x b_w < b_x a_w$. A figura 3.3 indica o sentido da ordenação.

A figura 3.4 mostra um exemplo de quatro segmentos com extremos em um mesmo ponto a nos modelos plano e esférico. O modelo esférico permite uma visualização melhor para os segmentos e seus comprimentos relativos. Note que, para todo $a,b \in \mathbb{T}^2$, $d_{\mathbb{T}^2}(a, b) = d_{\mathbb{T}^2}(b, a) = [x, w]$ e $d_{\mathbb{T}^2}(\neg b, a) = d_{\mathbb{T}^2}(b, \neg a) = [x, -w]$.

Operações Vamos definir duas operações: a soma de dois pontos do $\mathbb{T}^1_+$ e a soma de um ponto do $\mathbb{T}^1_+$ com um real. A primeira é uma função $+ : \mathbb{T}^1_+ \times \mathbb{T}^1_+ \to \mathbb{T}^1$, definida como $a + b = [a_x b_w + b_x a_w, a_w b_w]$ se a_w e b_w têm sinais opostos ou a_w e b_w são positivas; e como $a + b = [-a_x b_w - b_x a_w, -a_w b_w]$ se a_w e b_w são negativas. Podemos usar modelo plano do $\mathbb{T}^1$, na figura 3.3, para avaliar qualitativamente o resultado dessa operação: se os dois operandos estão no aquém, a soma resulta em um ponto do aquém *maior ou igual*

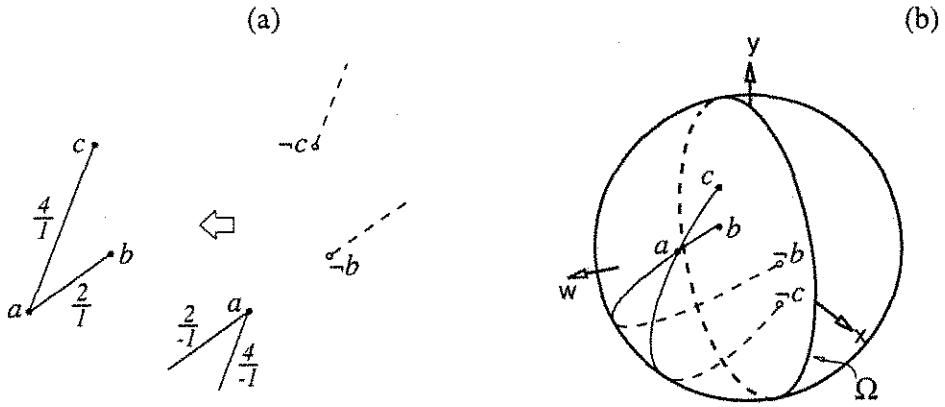

Figura 3.4: Comparando distâncias no $\mathbb{T}^2$

ao *maior* dos operandos; se estão no além, a soma resulta em um ponto do além *menor ou igual* ao *menor* dos operandos; e se estão em lados opostos, a soma resulta em um ponto do além *maior ou igual* ao operando do além.

A segunda é uma função $+ : \mathbb{T}^1_+ \times \mathbb{R} \to \mathbb{T}^1$, definida como $a + v = [a_x + va_w, a_w]$. Também podemos fazer uma avaliação qualitativa: o resultado da operação é um ponto do mesmo lado do operando do $\mathbb{T}^1_+$, menor que ele se $v < 0$ e maior se $v > 0$.

As cônicas e os diagramas no $\mathbb{T}^2$ serão definidos com essas operações, mas nós não precisaremos operar ou comparar distâncias no algoritmo que os constrói.

3.2.2 Distância relativa entre pontos próprios e impróprios

Podemos definir no $\mathbb{T}^2$ um conceito bastante intuitivo do $\mathbb{E}^2$: se nós varremos o plano desde o infinito com uma reta, numa dada direção, e a reta encontra o ponto a antes do ponto b, dizemos que b está mais perto do infinito *naquela direção* do que a.

Definição 1 *Sejam a e b dois pontos próprios, no mesmo lado do $\mathbb{T}^2$, e c um ponto impróprio (figura 3.5(a)). Seja $r_a = a \vee c$ e $r_a^\perp = a \vee \mathrm{norm}(r_a)$. Dizemos que*

$$\textit{se } b \diamond r_a^\perp = \left\{ \begin{array}{c} +1 \\ -1 \end{array} \right\} \textit{então} \left\{ \begin{array}{c} a \\ b \end{array} \right\} \textit{está mais perto de } c \textit{ do que} \left\{ \begin{array}{c} b \\ a \end{array} \right\}$$

e que a está tão perto de c quanto b se $b \diamond r_a^\perp = 0$.

Podemos também dizer, no contexto da definição 1, quão mais perto digamos, b está de c do que a está de c, simplesmente calculando a distância de b para a reta $r_a^\perp$. Chamamos esse conceito de distância relativa entre pontos próprios e impróprios. Muito embora pareça estranho comparar $\mathrm{d}_{\mathbb{T}^2}(a, c)$ e $\mathrm{d}_{\mathbb{T}^2}(b, c)$ quando ambas são infinitas (podendo assim

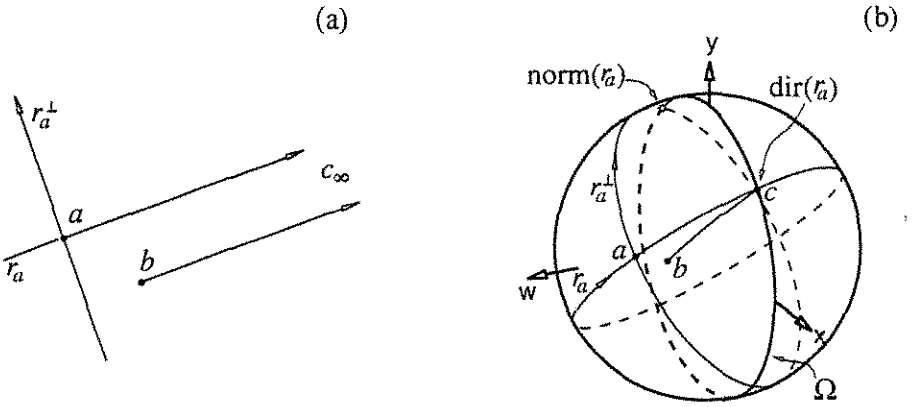

Figura 3.5: Distância relativa entre pontos próprios e impróprios

serem consideradas iguais), a comparação não leva a qualquer inconsistência quando nós usamos pontos impróprios em algoritmos e estruturas de dados.

Capítulo 4

Cônicas e Diagramas no Plano Projetivo Orientado

We have found it convenient to consider such diagrams as being drawn on the sphere rather than on the plane; topologically that is equivalent to augmenting the Euclidean plane by a dummy point at infinity. This allows us to represent such things as infinite edges and faces in the same way as their finite counterparts.

— L. Guibas e J. Stolfi [22]

A visualização dos diagramas depende diretamente de uma representação geométrica consistente para arcos de cônicas no $\mathbb{T}^2$. Normalmente, os artigos em geometria computacional não discutem esses detalhes de implementação, especialmente para algoritmos envolvendo objetos que não são lineares. A seção 4.1 apresenta o resultado da pesquisa que fizemos para obter uma representação para cônicas e arcos de cônicas. É interessante ver que o $\mathbb{T}^2$, ao invés de impor complicações, possibilitou uma simplificação do problema.

Essa discussão sobre cônicas pretende também aumentar a intuição do leitor sobre a topologia do $\mathbb{T}^2$ para ajudar o entendimento da construção dos diagramas que as seções 4.2, 4.3 e 4.4 definem e discutem. Apresentamos as principais características e, para o diagrama de pontos com peso aditivo, demonstramos sua conexidade no $\mathbb{T}^2$ o que constitui um resultado de importância teórica e prática.

4.1 Cônicas

É comum encontrar em livros de geometria, computação gráfica e modelagem geométrica representações de curvas classificadas como paramétricas ou implícitas [25]. Esses dois

tipos de representação têm objetivos complementares: com a paramétrica, podemos obter pontos sobre a curva a partir de um valor do parâmetro; com a implícita, podemos saber se um ponto qualquer do plano está sobre a curva a partir de suas coordenadas. Estes tipos de representação, entretanto, não dão intuição sobre a geometria da curva e, freqüentemente, estão distantes dos dados de entrada que o implementador de algoritmos geométricos tem em mãos. Buscamos, ao contrário, uma representação para cônicas e arcos que pode ser chamada de geométrica. Ela descreve a curva a partir de um conjunto de pontos, cada qual com uma função pré-estabelecida. Algumas das características desejáveis de tal representação são:

1. Capacidade de representar todos os possíveis casos sem ambigüidades;
2. Ser invariante sob transformações (projetivas, afins, similares);
3. Ausência de dados redundantes, o que pode tornar a representação inconsistente com uma pequena perturbação dos pontos originais;
4. Simplicidade.

Algumas representações na literatura Há muitas maneiras de se representar cônicas e arcos a partir de um conjunto de pontos. A escolha certamente depende da aplicação. Por exemplo, em um trabalho em computação gráfica, Herman [24] apresenta uma representação apropriada para o processo de composição de imagens a partir de modelos tri-dimensionais *(viewing pipeline)*. As cônicas são descritas por um conjunto de pontos característicos que são invariantes sob projetividades, reduzindo o custo do processo. Ele também mostra como obter da representação uma equação paramétrica tal que arcos podem ser descritos como um subintervalo do domínio do parâmetro. Entretanto, a representação não é muito simples e cada classe de cônicas afins (elipses, parábolas e hipérboles) possui uma representação diferente.

Para modelagem geométrica, Farin [13] mostra como representar arcos de cônicas como curvas NURBS *(Non-Uniform Rational B-Splines)*. Essa representação é única para as três classes e invariante sob transformações afins. Embora conveniente para modelar curvas, essa representação é muito distante do nosso problema.

Em geometria computacional, até onde pudemos verificar, o trabalho de Held [23] é o único envolvendo cônicas que discute como representá-las. No problema de construir *offsets* de contornos formados por segmentos de reta e arcos de círculos, arcos de cônicas são usados como um passo para a construção. Held usa uma representação que é propriamente uma parametrização de arcos cujo parâmetro é a distância para contorno. Essa representação é certamente útil para construir *offsets*, mas muito específica para ser usada em outras aplicações.

A principal observação, que nos guiou na busca pela representação, foi a presença dos focos das cônicas nos sítios dos diagramas. Por exemplo, no DVorW, o foco de todos os arcos de hipérbole são os próprios sítios. No diagrama de segmentos de reta, os focos das parábolas são os extremos dos segmentos. Assim, seria interessante que a representação incluisse os focos entre o conjunto de pontos que descreveria as cônicas.

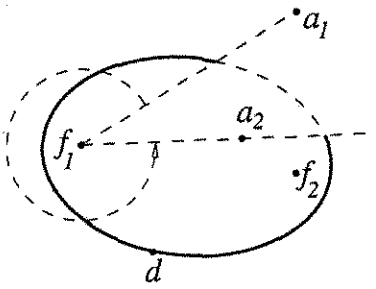

Figura 4.1: Uma representação para arcos de elipse no $\mathbb{E}^2$

Considere a seguinte representação para arcos de elipses no $\mathbb{E}^2$: dois pontos f_1 e f_2 como focos; um ponto d sobre a elipse para determiná-la; e dois outros pontos a_1 e a_2 tal que a interseção dos raios, partindo de f_1 com direção a a_1 e a_2, com a elipse, defina os extremos do arco. Como há dois arcos complementares satisfazendo esta representação, convencionamos que o arco correto é o definido pela rotação do raio de f_1 para a_1, em sentido anti-horário, até que ele seja igual ao de f_1 para a_2. Essa representação simples descreve apenas elipses. As seções seguintes vão mostrar que exatamente este conjunto de pontos representa satisfatoriamente arcos das três classes de cônicas no $\mathbb{T}^2$.

4.1.1 Cônicas algébricas no plano projetivo orientado

Os livros sobre cônicas [36] geralmente dão ênfase à forma implícita, ou seja, à definição algébrica das cônicas. Uma elipse no $\mathbb{E}^2$, por exemplo, é definida como o conjunto de pontos que satisfaz a uma equação polinomial do segundo grau cujos coeficientes satisfazem certas restrições.

Quando precisamos transportar fórmulas do $\mathbb{E}^2$ para o $\mathbb{T}^2$, basta substituir as coordenadas Cartesianas por homogêneas. É interessante observar que, com este processo, sempre que um ponto p satisfizer uma equação homogênea, o seu antípoda $\neg p$ também o fará. O resultado disso é que as cônicas algébricas no $\mathbb{T}^2$ são todas topologicamente iguais à hipérbole no $\mathbb{E}^2$, ou seja, constituem-se de duas componentes conexas que separam o plano em três partes. Para exemplificar, considere o círculo $x^2 + y^2 - 1 = 0$. Em coordenadas homogêneas temos $x^2 + y^2 - w^2 = 0$. O conjunto de pontos que satisfaz esta equação possui duas componentes conexas, veja figura 4.2(a). Se olhamos para esse

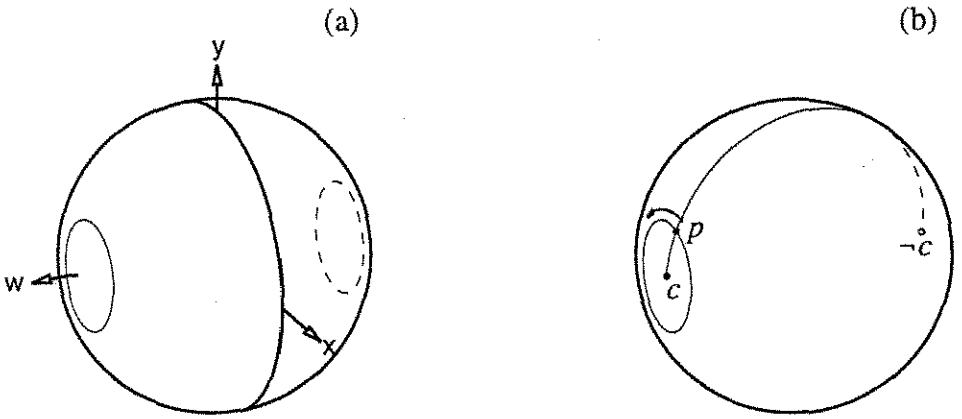

Figura 4.2: Um círculo algébrico no $\mathbb{T}^2$ e um círculo geométrico

círculo do ponto de vista geométrico não podemos dizer propriamente que ponto é o seu centro. Mas a questão, na realidade, é que uma representação para arcos de círculos precisaria distinguir essas duas componentes. Uma maneira é usar a definição de círculo da geometria esférica [17]: um círculo é o conjunto de todos os pontos para os quais a distância para um centro c iguala um raio $\ell/1$, figura 4.2(b). Este é o mesmo conjunto de pontos para o círculo com centro em $\neg c$ e raio $\ell/-1$. Essa distinção pelo centro pode ser observada também pelo conceito de orientação: se movemos um ponto p sobre o círculo, em sentido anti-horário, visto de fora da esfera, o segmento cp rotaciona em sentido anti-horário ao redor de c, mas $\neg cp$ rotaciona em sentido horário ao redor de $\neg c$. Podemos, então, considerar as duas componentes conexas do círculo algébrico como círculos antípodas, cada qual com duas orientações possíveis.

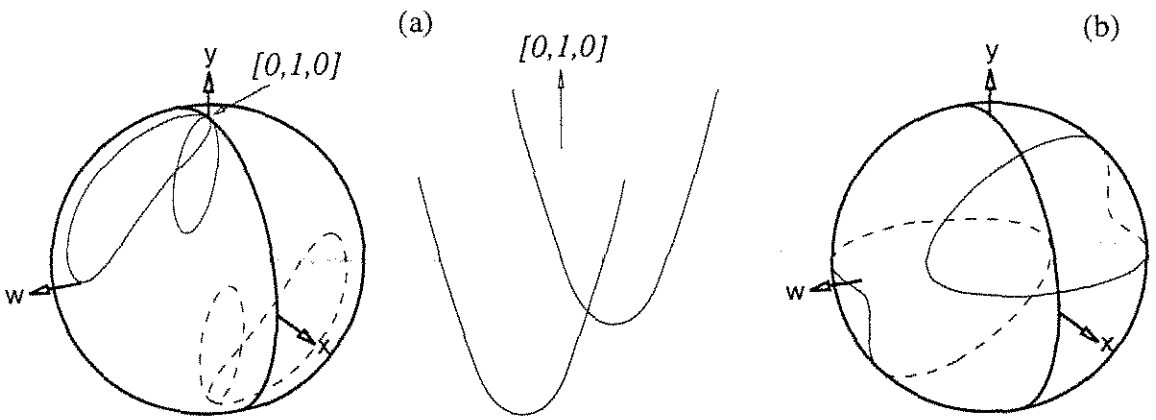

Figura 4.3: Cônicas algébricas no $\mathbb{T}^2$

Para aumentar a intuição sobre a topologia do $\mathbb{T}^2$, a figura 4.3(a) mostra um par de parábolas, cada uma com duas componentes conexas, nos modelos esférico e plano,

$yw = x^2$ e $(y - 100w)w = (x - 100w)^2$ que diferem entre si somente por uma translação pelo vetor $[100, 100, 1]$. Note que elas são tangentes a Ω em dois pontos antípodas, $[0, 1, 0]$ e $[0, -1, 0]$, que são as interseções de suas retas focais com Ω. A figura 4.3(b) mostra uma hipérbole que essencialmente difere das outras cônicas apenas por cruzar a reta do infinito.

Veremos agora que, não sem surpresa, podemos usar a definição de cônicas na esfera para distinguir as duas componentes conexas das cônicas algébricas.

4.1.2 Cônicas na esfera do espaço Euclidiano

Em geometria esférica, cônicas podem ser definidas como o conjunto de todos os pontos para os quais a soma das distâncias para dois focos dados é constante [35, capítulo X]. Essa é a mesma definição de elipses no $\mathbb{E}^2$ e, de fato, cônicas esféricas se assemelham a elipses (note a figura 4.3). Elas podem ser definidas, também, como a interseção da esfera com cones de segundo grau cujos vértices estão no centro da esfera. É fácil ver que a interseção de tais cones com um plano é sempre uma cônica planar. Vemos, portanto, que as cônicas no plano se projetam na esfera como cônicas esféricas. Esse fato não é de todo trivial pois podemos observar que a projeção dos focos das cônicas planares sobre a esfera *não* resulta nos focos das cônicas esféricas projetadas.

Essa discussão foi motivada não apenas pela busca da definição de cônicas da próxima seção, mas também pela necessidade de implementar as funções de desenho de cônicas no modelo esférico, que serão discutidas no capítulo 6.

4.1.3 Definição de cônicas no plano de dois lados

Apesar de não haver correspondência entre os focos das cônicas planares e esféricas, nós podemos aplicar a mesma definição de cônicas esféricas para cônicas no $\mathbb{T}^2$. É sabido que uma parábola pode ser considerada, em todos os aspectos, como sendo uma elipse com um de seus focos no infinito [36, pág. 202]. No $\mathbb{T}^2$ vamos levar este foco para além do infinito e considerar a hipérbole resultante como uma elipse com focos em lados opostos do plano.

Definição 2 *No plano Euclidiano de dois lados, uma cônica é o conjunto de todos os pontos para os quais a* soma *das distâncias para dois focos f e f' dados é constante. Isso gera uma elipse se f e f' estão do mesmo lado, uma parábola se um dos focos está em Ω e uma hipérbole se eles estão em lados opostos; ou, alternativamente, isso gera uma elipse se $d_{\mathbb{T}^2}(f, f') <_{\mathbb{T}^1} [1, 0]$, uma parábola se $d_{\mathbb{T}^2}(f, f') = [1, 0]$ e uma hipérbole se $d_{\mathbb{T}^2}(f, f') >_{\mathbb{T}^1} [1, 0]$*

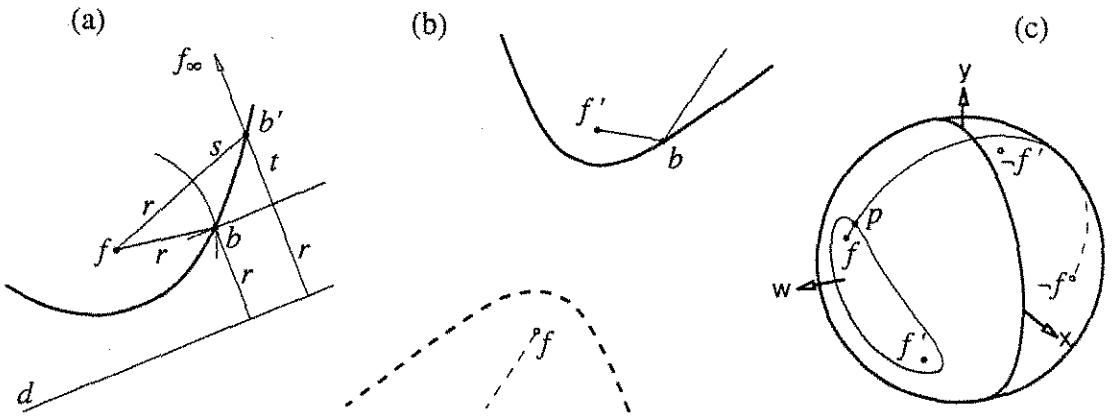

Figura 4.4: Definição de cônicas no T^2

É interessante mostrar como esta definição gera parábolas e hipérboles. Para parábolas nós usamos a distância relativa entre pontos próprios e impróprios como se segue. Considere a parábola da figura 4.4(a): quando movemos um ponto de b para b' a distância relativa para o foco impróprio decresce de t. Da definição de parábolas no E^2, nós sabemos que $r + t = r + s$, de modo que $s = t$ e assim a distância para o foco próprio cresce de t. Para hipérboles note que uma das distâncias está sempre no além do T^1 (na figura 4.4(b) a distância $d_{\mathsf{T}^2}(b, f)$) e a outra no aquém. Assim o efeito de adicionar as duas distâncias é subtrair seus valores absolutos, o que é a definição de hipérboles no E^2.

Essa definição também distingue as duas componentes conexas das cônicas algébricas. Como na discussão sobre círculos, elas agora são cônicas antípodas, cada uma com duas possíveis orientações. A figura 4.4(c) mostra uma elipse definida pelos focos f e f' e alguma soma constante $\ell/1$, que é o mesmo conjunto de pontos da elipse definida pelos focos $\neg f$ e $\neg f'$ e soma constante $\ell/-1$. Diremos que a primeira está orientada em sentido anti-horário e a segunda em sentido horário. Note que se movemos um ponto p em sentido anti-horário sobre a elipse (visto de fora da esfera), o segmento fp rotaciona em sentido anti-horário ao redor de f, mas $\neg fp$ rotaciona em sentido horário ao redor de $\neg f$. A elipse antípoda é definida pelos focos f e f' e soma constante $\ell/-1$. A elipse orientada opostamente em relação a esta última é definida pelos focos $\neg f$ e $\neg f'$ e soma constante $\ell/1$.

A figura 4.5 mostra, como um exemplo adicional, uma elipse, uma parábola e uma hipérbole nos modelos plano e esférico. Podemos ver que a parábola tangencia Ω e que a hipérbole intercepta Ω em dois pontos, mas as três são obtidas a partir da mesma definição.

Vamos citar três características das cônicas dadas pela definição 2 que serão úteis no estabelecimento da representação, especialmente para arcos:

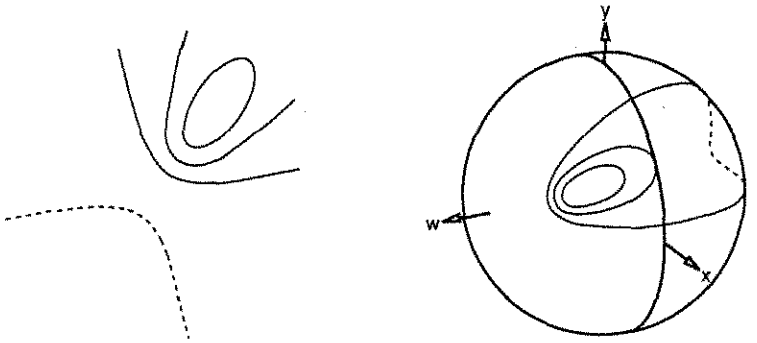

Figura 4.5: Cônicas no $\mathbb{T}^2$

1. Se removemos uma cônica do $\mathbb{P}^2$ restam dois subespaços, um topologicamente equivalente a um disco aberto e o outro a uma fita de Möbius. Se removemos uma cônica do $\mathbb{T}^2$ restam dois subespaços, ambos equivalentes a um disco aberto;

2. Chamaremos de interior de uma cônica o subespaço que contém ao menos um dos focos. Os segmentos ligando qualquer ponto sobre a cônica aos seus focos estão inteiramente contidos no interior da cônica, ao contrário do que acontece com hipérboles no $\mathbb{E}^2$;

3. Da propriedade anterior, vemos que o interior de uma cônica é sempre estrelado em relação aos focos. Além disso, se o interior é convexo, então a cônica estará orientada em sentido anti-horário (como definido anteriormente).

4.1.4 Representação de cônicas e arcos

Nós representamos uma cônica pelas coordenadas homogêneas com sinal de três pontos no $\mathbb{T}^2$: os dois focos f_1 e f_2 e um ponto d sobre a cônica. Isso é suficiente para determinar univocamente a cônica. A classe afim é dada implicitamente pela localização relativa dos focos. Para todas as classes, para trocar a orientação da cônica, basta trocar os focos por seus antípodas, como vimos na seção anterior. Isso significa simplesmente multiplicar suas coordenadas por -1. Para obter a cônica antípoda, nós trocamos o ponto d por seu antípoda.

Como foi adiantado no início do capítulo, nós representamos um arco de cônica com os mesmos três pontos (f_1, f_2 e d para representar a cônica em si) e mais dois pontos a_1 e a_2 usados na determinação dos extremos do arco.

Como as cônicas são estreladas em relação aos focos, todas as retas passando por um foco interceptam a cônica em dois pontos. Seja $r_1 = f_1 \vee a_1$ e $r_2 = f_1 \vee a_2$. Nós definimos o primeiro extremo do arco como sendo a interseção de r_1 com a cônica quando a reta atravessa do interior para o exterior da cônica (veja figura 4.6 para um arco de hipérbole). O

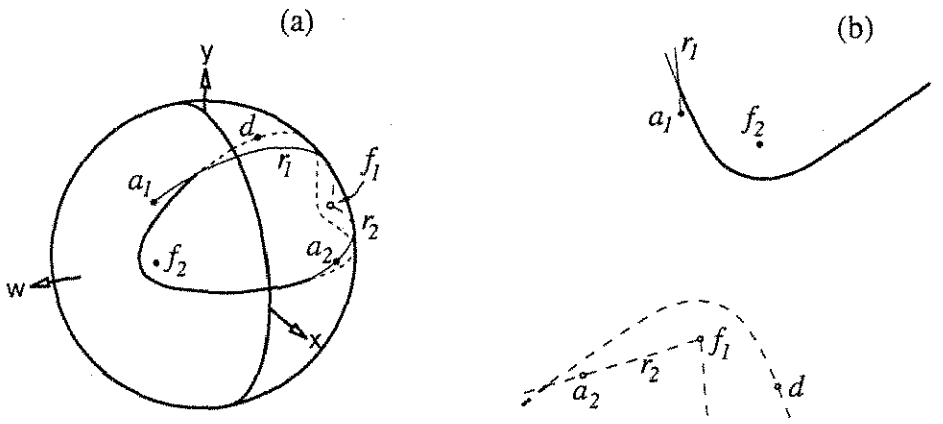

Figura 4.6: Representação de um arco de hipérbole

segundo extremo é, como esperado, a interseção de r_2 com a cônica nas mesmas condições. Novamente há dois arcos complementares satisfazendo essa representação. Convencionamos que o correto é o definido pela rotação de r_1, em sentido anti-horário (visto de fora da esfera), até que ela seja igual a r_2. Note que, em função dessa convenção, podemos obter o arco complementar, a partir de uma representação, de duas maneiras: trocando a orientação da cônica (veja figura 4.7) ou trocando a_1 por a_2. Para conseguir o arco antípoda devemos trocar d, a_1 e a_2 por seus antípodas.

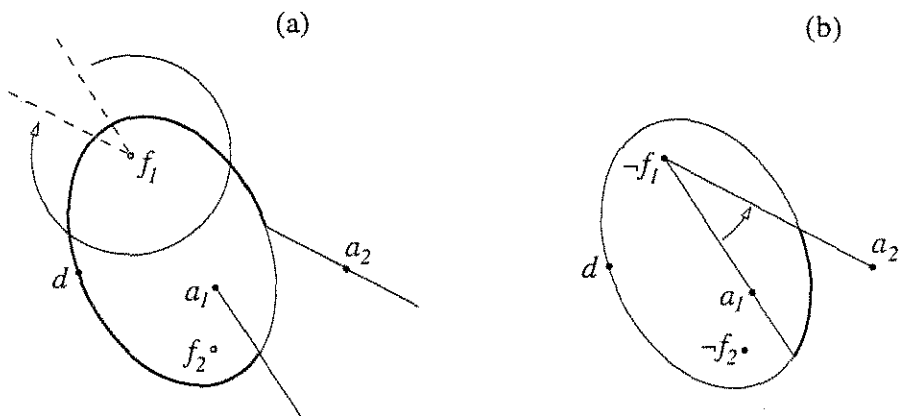

Figura 4.7: Obtendo o arco complementar trocando a orientação da cônica

Casos degenerados Essa representação pode descrever sem redundância qualquer arco de cônica no T^2. Mais precisamente, uma pequena perturbação dos pontos *não* é capaz de tornar a representação inconsistente. Entretanto, as coordenadas dos pontos f_1, f_2, d, a_1 e a_2 podem, de fato, assumir valores que não representam propriamente uma cônica ou um arco:

1. Quando $a_1(a_2)$ e f_1 são coincidentes ou antípodas, $r_1(r_2)$ não está definida;
2. Quando ambos f_1 e f_2 estão no infinito;
3. Quando f_1 e f_2 são antípodas;
4. Quando d está no infinito e f_1 e f_2 estão do mesmo lado. Nesse caso, nós podemos considerar a cônica como sendo igual a Ω.

   Quando d está no infinito e f_1 e f_2 em lados opostos, temos ainda uma hipérbole bem definida, mas este caso requereria um tratamento computacional especial.

4.2 Diagrama de Pontos

Depois de todos estes conceitos, podemos finalmente discutir diagramas de Voronoi no $\mathbb{T}^2$. Seja $\mathcal{S}$ um conjunto finito de pontos do $\mathbb{T}^2$, em posição geral. Para cada $p,q \in \mathcal{S}$ definimos:

$$H_{pq} = \{x \in \mathbb{T}^2 | \mathrm{d}_{\mathbb{T}^2}(x,p) \leq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(x,q)\}.$$

Novamente, a região de Voronoi de um sítio p é $R_p = \bigcap_{q \in \mathcal{S}\setminus\{p\}} H_{pq}$. A figura 4.8(a) mostra o $\mathrm{DVor}_{\mathbb{T}^2}$ de $\mathcal{S} = \{p,q\}$. Não é difícil ver que H_{pq} é um semiplano limitado por uma reta do $\mathbb{T}^2$. A figura 4.8(b) mostra o diagrama depois que três outros sítios foram incluídos em $\mathcal{S}$. As arestas destacadas são o contorno de R_p. Como $\mathrm{d}_{\mathbb{T}^2}$ se reduz à distância Euclidiana se nos restringimos a um dos lados do $\mathbb{T}^2$, a parte do $\mathrm{DVor}_{\mathbb{T}^2}$ contida no aquém é idêntica ao DVor.

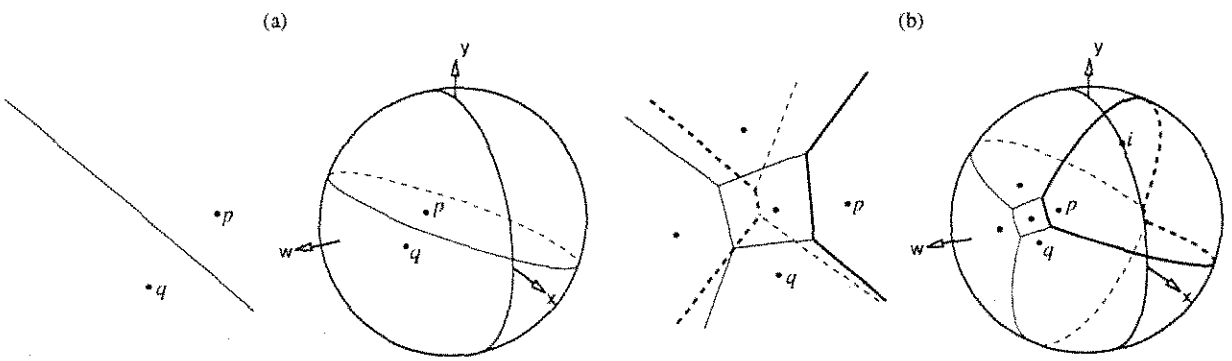

Figura 4.8: Diagrama de pontos no $\mathbb{T}^2$

Ao contrário do que acontece no $\mathbb{E}^2$, agora todas as arestas (mesmo as infinitas) são limitadas por dois vértices e todas as regiões são limitadas por arestas que efetivamente são bissetores de sítios de $\mathcal{S}$. Note que cada vértice do $\mathrm{DVor}_{\mathbb{T}^2}$ é o centro de um círculo que passa por três sítios e possui seu interior vazio.

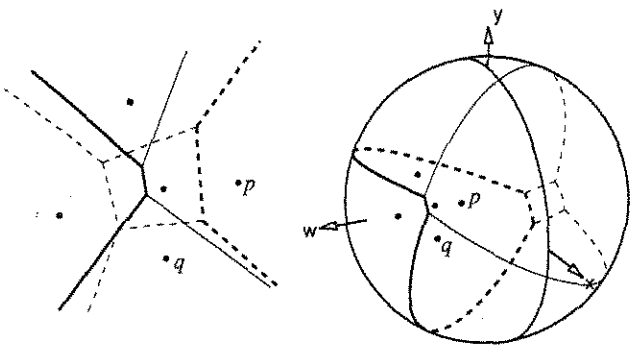

Figura 4.9: Diagrama de pontos de vizinho mais distante no $\mathbf{T}^2$

A figura 4.8(b) permite também ver que as regiões dos sítios na envoltória convexa de $\mathcal{S}$ contêm pontos impróprios. Esses pontos não podem ser diretamente substituídos na definição de H_{pq}, mas é fácil ver que o conceito de distância relativa entre pontos próprios e impróprios (seção 3.2.2) pode ser aplicado. Por exemplo, apesar de as distâncias do ponto i para todos os sítios de $\mathcal{S}$ serem todas infinitas, o sítio p é o que está mais próximo de i.

A observação mais interessante é que as regiões do $\mathrm{DVor}_{\mathbb{T}^2}$ cobrem todo o $\mathbf{T}^2$. Podemos então fazer o seguinte raciocínio intuitivo: como $\neg x$ é o ponto mais distante de x, o sítio mais próximo de $\neg x$ será o sítio mais distante de x. Assim, o diagrama de vizinho mais próximo pode ser usado para descobrir o vizinho mais distante.

De fato, $\mathrm{d}_{\mathbb{T}^2}(x,p) \leq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(x,q)$ se e somente se $\mathrm{d}_{\mathbb{T}^2}(\neg x,p) \geq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(\neg x,q)$, o que é a definição do diagrama de vizinho mais distante:

$$\begin{array}{rcl} [x_1,w_1] & \leq_{\mathbb{T}^1} & [x_2,w_2] \ \textit{implica que} \\ x_1(w_2) & \leq & x_2(w_1) \\ x_1(-w_2) & \geq & x_2(-w_1) \\ [x_1,-w_1] & \geq_{\mathbb{T}^1} & [x_2,-w_2] \end{array}$$

A conclusão é que os dois diagramas são antípodas no $\mathbb{T}^2$. A figura 4.9 mostra o diagrama de vizinho mais distante para o mesmo conjunto $\mathcal{S}$. Note que ele é composto pelos pontos antípodas do $\mathrm{DVor}_{\mathbb{T}^2}(\mathcal{S})$. A característica mais interessante desse diagrama de vizinho mais distante no $\mathbb{T}^2$ é que *todos* os sítios possuem regiões, não só os da envoltória convexa.

Complexidade Sejam n o número de sítios, e o número de arestas e v o número de vértices do $\mathrm{DVor}_{\mathbb{T}^2}(\mathcal{S})$. Temos as seguintes propriedades, que revelam a *similaridade* entre o $\mathrm{DVor}_{\mathbb{T}^2}$ e o diagrama de Voronoi na esfera do $\mathbb{E}^3$ (seção 2.2):

1. Para $n = 2$: o diagrama é uma reta do $\mathbb{T}^2$;

2. Para $n = 3$: $v = 2$ e $e = 3$. Os vértices são antípodas; as arestas *não* são segmentos do $\mathbb{T}^2$ e as regiões *não* são convexas (elas contêm os vértices antípodas);

3. Para $n > 3$: cada aresta é um segmento do $\mathbb{T}^2$ e cada região é convexa.

   Como n é o número de faces da subdivisão planar induzida pelo $\mathrm{DVor}_{\mathbb{T}^2}$ e cada vértice tem valência 3, pela fórmula de Euler ($e + 2 = v + f$), $v = 2n - 4$ e $e = 3n - 6$.

4.3 Diagrama de Pontos com Peso Aditivo

O $\mathrm{DVorW}_{\mathbb{T}^2}$ é definido pela modificação de H_{pq}, como na seção 2.2:

$$H_{pq} = \{x \in \mathbb{T}^2 | \mathrm{d}_{\mathbb{T}^2}(x,p) - w(p) \leq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(x,q) - w(q)\}.$$

O antípoda do diagrama definido com $\mathrm{d}_{\mathbb{T}^2}(x,p) - w(p)$ é o diagrama de vizinho mais distante definido com $\mathrm{d}_{\mathbb{T}^2}(x,p) + w(p)$, pois $\mathrm{d}_{\mathbb{T}^2}(x,p) - w(p) \leq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(x,q) - w(q)$ se e somente se $\mathrm{d}_{\mathbb{T}^2}(\neg x,p) + w(p) \geq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(\neg x,q) + w(q)$:

$$\begin{aligned}
[x_1, w_1] + (-v_1) &\leq_{\mathbb{T}^1} [x_2, w_2] + (-v_2) \\
[x_1 - v_1 w_1, w_1] &\leq_{\mathbb{T}^1} [x_2 - v_2 w_2, w_2] \quad \textit{implica que} \\
(x_1 - v_1 w_1)(w_2) &\leq (x_2 - v_2 w_2)(w_1) \\
(x_1 - v_1 w_1)(-w_2) &\geq (x_2 - v_2 w_2)(-w_1) \\
[x_1 - v_1 w_1, -w_1] &\geq_{\mathbb{T}^1} [x_2 - v_2 w_2, -w_2] \\
[x_1, -w_1] + (v_1) &\geq_{\mathbb{T}^1} [x_2, -w_2] + (v_2)
\end{aligned}$$

A figura 4.10(a) mostra o $\mathrm{DVorW}_{\mathbb{T}^2}$ de $\mathcal{S} = \{p, q\}$. Agora H_{pq} é um semiplano limitado por uma hipérbole de $\mathbb{T}^2$, como definida anteriormente, com focos em p e $\neg q$ ou $\neg p$ e q. A figura 4.10(b) mostra o diagrama depois que três outros sítios foram incluídos em $\mathcal{S}$.

Essa construção no $\mathbb{T}^2$ traz não apenas a vantagem da representação uniforme das arestas (toda aresta é um arco de hipérbole do $\mathbb{T}^2$) mas também a conexidade do diagrama. A próxima seção apresenta uma demonstração desse fato. Note que o exemplo da figura 4.10(b) é desconexo no $\mathbb{E}^2$.

Podemos ver também que, como no caso no $\mathrm{DVor}_{\mathbb{T}^2}$, o número de faces da subdivisão planar induzida é n e, supondo posição geral, cada vértice tem valência 3. Desse modo, apesar de o $\mathrm{DVorW}_{\mathbb{T}^2}$ possuir uma complexidade topológica maior, ele possui as mesmas cotas exatas independentes do número de sítios na envoltória convexa: $v = 2n - 4$ e $e = 3n - 6$.

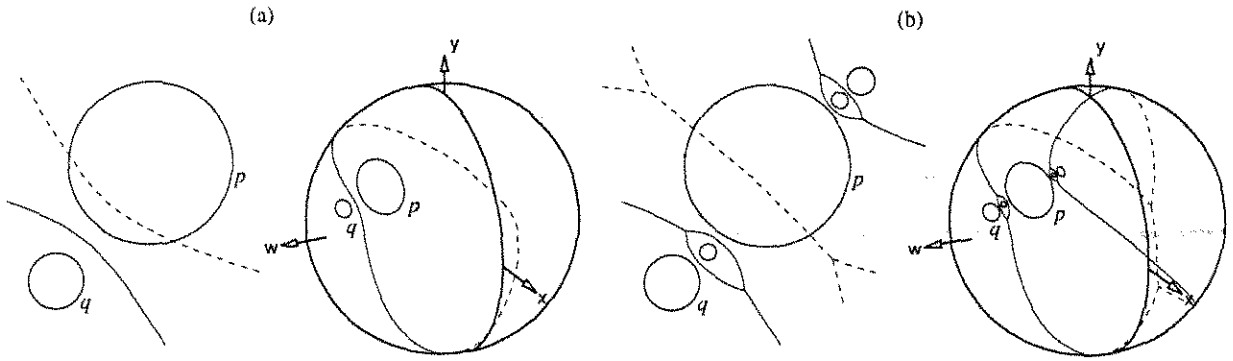

Figura 4.10: Diagrama de pontos com peso aditivo no $\mathbb{T}^2$

4.3.1 Conexidade do diagrama

A demonstração da conexidade do $\text{DVorW}_{\mathbb{T}^2}$ é uma simples extensão da demonstração dada por Sharir em [37] de que, no $\mathbb{E}^2$, todas as componentes conexas do diagrama são ilimitadas, ou seja, possuem arestas infinitas. Vamos apresentar essa demonstração para o $\mathbb{E}^2$ e depois discutir a sua extensão para o $\mathbb{T}^2$.

Teorema 1 *Todas as componentes conexas do DVorW são ilimitadas.*

Prova As regiões de Voronoi cobrem todo o $\mathbb{E}^2$ e são estreladas em relação aos seus correspondentes sítios. Com isso, suponha que o DVorW contém uma componente conexa K limitada. Então a porção E de uma vizinhança suficientemente pequena de K que está no exterior de K tem que estar contida na região de um único sítio, digamos m, veja figura 4.11(a). No caso contrário, haveria uma aresta do DVorW cruzando a porção E e contradizendo a suposição de que K é uma componente conexa. Mas como E envolve K inteiramente, tem que haver um ponto y de E tal que o segmento ligando y a m cruza a componente K, contradizendo o fato de que a região de m é estrelada. □

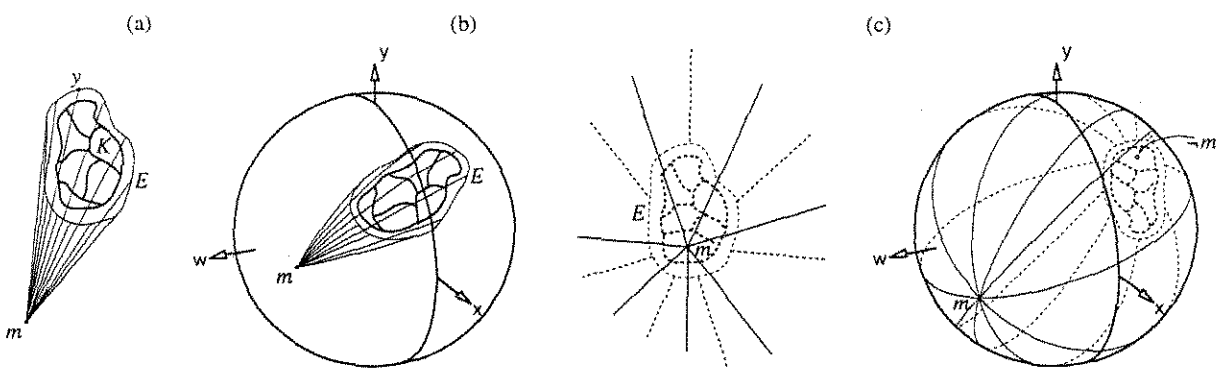

Figura 4.11: Ilustrando a prova da conexidade do $\text{DVorW}_{\mathbb{T}^2}$

A razão pela qual essa demonstração pode ser usada para mostrar a conexidade do $\mathrm{DVorW}_{\mathbb{T}^2}$ é que, simplesmente, não pode haver arestas ou componentes ilimitadas no $\mathbb{T}^2$, dada a sua topologia compacta. No entanto, precisamos distinguir dois casos: quando o antípoda do sítio m não está contido no interior de K e quando está. Para o primeiro caso, a demonstração é idêntica. Veja a figura 4.11(b) para um exemplo de componente conexa onde as arestas são infinitas.

Quando $\neg m$ está contido no interior de K, é fácil ver que o feixe de segmentos ligando os pontos de E a m não cruza K, veja figura 4.11(c) com um exemplo nos modelos plano e esférico. Entretanto, o feixe de segmentos cobre todo o $\mathbb{T}^2$ com exceção da componente K. Assim, haverá algum segmento que cruza alguma outra componente conexa do $\mathrm{DVorW}_{\mathbb{T}^2}$ contradizendo, novamente, o fato de que a região de m é estrelada.

4.4 Diagrama de Segmentos de Retas

O diagrama de Voronoi de segmentos de reta $\mathrm{DVorL}_{\mathbb{T}^2}$ pode ser definido por:

$$H_{L_1 L_2} = \{x \in \mathbb{T}^2 | \min\{\mathrm{d}_{\mathbb{T}^2}(x,p) | p \in L_1\} \leq_{\mathbb{T}^1} \min\{\mathrm{d}_{\mathbb{T}^2}(x,p) | p \in L_2\}\}.$$

onde L_1 e L_2 são dois segmentos de $\mathcal{S}$. Esse diagrama possui arcos de parábola; no entanto, é fácil ver que se os segmentos são finitos, então as arestas que se estendem até o infinito são todas segmentos de reta. O diagrama antípoda é o de vizinho mais distante definido com $\max\{\mathrm{d}_{\mathbb{T}^2}(x,p) | p \in L\}$, pois $\min\{\mathrm{d}_{\mathbb{T}^2}(x,p) | p \in L_1\} \leq_{\mathbb{T}^1} \min\{\mathrm{d}_{\mathbb{T}^2}(x,p) | p \in L_2\}$ se e somente se $\max\{\mathrm{d}_{\mathbb{T}^2}(\neg x,p) | p \in L_1\} \geq_{\mathbb{T}^1} \max\{\mathrm{d}_{\mathbb{T}^2}(\neg x,p) | p \in L_2\}$.

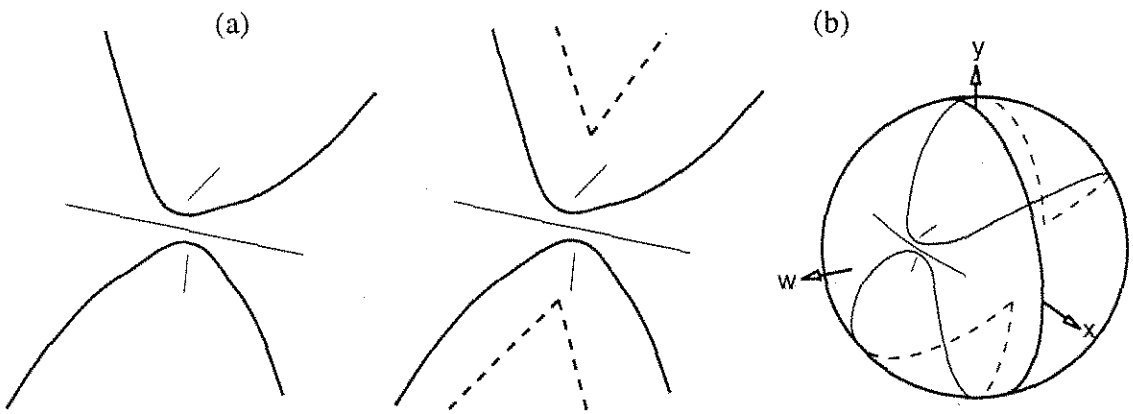

Figura 4.12: Diagrama de segmentos de reta no $\mathbb{T}^2$

Essa parte do diagrama contida no além é sempre um subconjunto das arestas do diagrama de vizinho mais distante do conjunto dos pontos extremos dos segmentos que estão na envoltória convexa, considerados como sítios. Se nenhum segmento contribui

com mais de um extremo para a envoltória convexa, então a parte contida no além será exatamente o diagrama de Voronoi de vizinho mais distante desses extremos.

A figura 4.12(a) mostra um DVorL (no plano Euclidiano) de três segmentos que é desconexo. Infelizmente, o diagrama continua desconexo no T^2, veja figura 4.12(b). Esse exemplo mostra, mais precisamente, uma configuração de três segmentos para os quais não existe um ponto que esteja equidistante dos três, nem no E^2 nem no T^2. Apesar de o diagrama continuar desconexo, a construção no T^2 continua interessante pela representação uniforme das arestas.

O algoritmo a ser apresentado no próximo capítulo *não* é capaz de calcular esse diagrama porque uma das primitivas geométricas é calcular os circuncentros de três dados sítios que são, justamente, os pontos do plano equidistantes dos sítios. A conclusão dessa dissertação apresenta, entre outras opções de trabalhos futuros, uma discussão sobre uma possível implementação dessa construção no T^2.

Capítulo 5

A Descrição do Algoritmo

Incremental algorithms for computing Voronoi diagrams in the plane have the advantage of being simple to implement. Even though for certain situations their complexities can be undesirably high, it has been shown that if the points in the set are introduced in random order then the Voronoi diagram or a Delaunay triangulation for the set can be computed in expected O(n log n) time and expected O(n) space, where n is the number of points in the set.

— J. Bernal [6]

A construção dos diagramas no $\mathbb{T}^2$ proporciona uniformidade na representação. No entanto, o interesse por ela depende certamente da possibilidade de se obter algoritmos que trabalhem no $\mathbb{T}^2$ como um espaço também uniforme, não fazendo distinção entre aquém e além. Seria muito ruim se precisássemos calcular a parte do diagrama no aquém separadamente da parte no além e depois "costurar" os dois diagramas na reta do infinito.

Este capítulo apresenta, em detalhes, o algoritmo incremental implementado para a construção dos diagramas no $\mathbb{T}^2$ que atinge esse objetivo. O $\mathbb{T}^2$ aparece somente no mais baixo nível das primitivas geométricas. A caracterização geral é a mesma para as duas implementações: $\mathrm{DVor}_{\mathbb{T}^2}$ e $\mathrm{DVorW}_{\mathbb{T}^2}$. Elas diferem apenas na primitiva de cálculo dos dois circuncentros de três dados sítios e em alguns procedimentos especiais necessários para o $\mathrm{DVorW}_{\mathbb{T}^2}$.

Os algoritmos foram implementados em C++ com o auxílio da biblioteca LEDA [27] para tipos básicos como listas e tabelas de hashing, e tipos numéricos para computação exata. A topologia dos diagramas é armazenada com a estrutura *quad-edge* [22] e a geometria com a representação do capítulo 4. Algumas imagens de diagramas construídos

com este algoritmo, nos modelos plano e esférico do $\mathbb{T}^2$, serão apresentadas no capítulo 6 como resultado da implementação dos visualizadores com suporte a cônicas no $\mathbb{T}^2$.

5.1 Caracterização Geral

A idéia básica do algoritmo incremental é a de Green e Sibson [19] apresentada no capítulo 2. Ele consiste de duas fases: a localização do sítio mais próximo ao novo sítio k e a construção de sua região R_k para a atualização do diagrama.

Vamos introduzir uma definição de diagramas de Voronoi apresentada em [40]. Chamaremos de disco de liberdade (*clearance disk*) um disco fechado tangente a dois ou mais sítios do conjunto $\mathcal{S}$. Um disco fechado é tangente a um ponto se o ponto está contido na sua fronteira, e tangente a outro disco fechado quando a interseção entre eles é um único ponto. Os diagramas podem, então, ser definidos como o conjunto de todos os centros de discos de liberdade de $\mathcal{S}$.

No $\mathbb{T}^2$, o diagrama de Voronoi de três sítios ($\mathrm{DVor}_{\mathbb{T}^2}$ ou $\mathrm{DVorW}_{\mathbb{T}^2}$) possui sempre dois vértices. Esses vértices são exatamente os centros dos dois discos de liberdade que tangenciam os três sítios. Eles também são os dois pontos do $\mathbb{T}^2$ que estão equidistantes dos três sítios. Chamaremos esses pontos equidistantes a três dados sítios de circuncentros (*circuncenters*). Uma das primitivas geométricas é justamente o cálculo dos dois circuncentros de três dados sítios. Assim, para os dois diagramas, mostraremos como calcular esses circuncentros no $\mathbb{T}^2$.

Agora considere um diagrama $\mathcal{D}$ de um conjunto $\mathcal{S}$ de sítios. A inclusão de um novo sítio k irá adicionar alguns vértices ao diagrama (os vértices de R_k) e, talvez, subtrair outros. Nosso trabalho será, então, encontrar os novos vértices. Sabemos, pela definição acima, que os centros de todos os discos de liberdade de $\mathcal{S}$ estão sobre $\mathcal{D}$. Vemos então que os novos vértices, que são centros de discos de liberdade que tangenciam k e dois sítios de $\mathcal{S}$, estarão necessariamente sobre alguma aresta de $\mathcal{D}$. Chamaremos as arestas de $\mathcal{D}$ que contiverem pelo menos um novo vértice de contribuintes. Uma aresta de $\mathcal{D}$ pode contribuir com 0, 1 ou 2 novos vértices. Isto porque todo novo vértice sobre uma aresta é um circuncentro de k com os dois sítios cuja interseção das regiões a definem em $\mathcal{D}$ e há somente dois circuncentros.

A operação geométrica básica do algoritmo é, então, dizer se uma aresta de $\mathcal{D}$ é ou não contribuinte dado o novo sítio k. Mostraremos como efetuar esta operação apenas com a primitiva geométrica de orientação, descrita na seção 3.1.1. Como todos os novos vértices estarão sobre alguma aresta de $\mathcal{D}$, uma maneira de obtê-los é testar seqüencialmente todas as arestas. Veremos que podemos otimizar essa tarefa, na segunda fase do algoritmo, encontrando as arestas contribuintes consecutivamente dada uma aresta contribuinte inicial. Assim, na primeira fase do algoritmo, procuraremos não pelo sítio mais próximo a

k, mas por alguma aresta contribuinte.

5.1.1 Considerações

Antes de fazer essa caracterização pelas arestas, tentamos usar os sítios e os vértices para adaptar o algoritmo ao $\mathbb{T}^2$. Para o caso do $\mathrm{DVor}_{\mathbb{T}^2}$, é possível obter um algoritmo chamando os vértices que desaparecem, na inclusão de k, de vértices conflitantes. Assim, na primeira fase, era preciso achar algum vértice conflitante. Na atualização, calculávamos os novos vértices olhando para as arestas cujas origens fossem um vértice conflitante e o destino não. Essa caracterização, em essência, é a apresentada em [3]. No entanto, para o $\mathrm{DVorW}_{\mathbb{T}^2}$, pode acontecer de a inclusão de k tornar todos os vértices de $\mathcal{D}$ conflitantes e, neste caso, o algoritmo falhava.

A escolha da estrutura *quad-edge* (uma das estruturas que atribuem orientações às arestas) foi muito oportuna, pois pudemos dizer, então, que cada aresta orientada contribui com 0 ou 1 vértice, o que simplificou ainda mais a segunda fase do algoritmo para o $\mathrm{DVorW}_{\mathbb{T}^2}$. A representação geométrica das arestas, que é obtida simultaneamente com a representação topológica durante a segunda fase do algoritmo, fica referenciada por um apontador na estrutura *quad-edge*.

5.1.2 Procedimento geral de orientação relativa

Esta seção descreve o procedimento comum às duas implementações, que é usado na decisão sobre a contribuição das arestas. Ele é implementado com três chamadas ao procedimento `antiHorario()` descrito na seção 3.1.1.

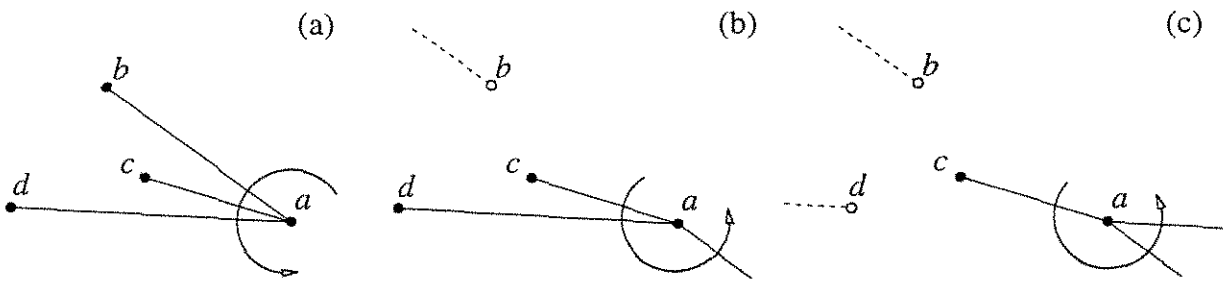

Figura 5.1: Procedimento para orientação relativa de 4 pontos

Procedimento 2 `ordemCircular( b, c, d, a )` *retorna* `true` *se a ordem circular de* `b, c` *e* `d`*, em sentido anti-horário ao redor de* `a`*, é* `bcd` *e* `false` *caso contrário:*

```
bool abc = antiHorario( a, b, c );
bool abd = antiHorario( a, b, d );
```

```
bool acd = antiHorario( a, c, d );
return ((abc && (!abd || acd)) || (!abd && acd));
```

Como exemplo, a figura 5.1(a) e (b) mostra duas instâncias verdadeiras do procedimento `ordemCircular( b, c, d, a )`. A figura 5.1(c) mostra uma instância falsa.

5.2 Diagrama de Pontos

O algoritmo para o $\mathrm{DVor}_{\mathbb{T}^2}$, como descrito nesta seção, assume que $\mathcal{S}$ não contém quatro pontos co-circulares. Ao começar a desenvolver o algoritmo, nós previámos que $\mathcal{S}$ continha apenas sítios no aquém. Para construir o diagrama para $\mathcal{S}$ com sítios apenas no além, bastaria tomar os antípodas dos sítios e depois tomar o diagrama antípoda. Foi muito interessante observar, entretanto, que a implementação funciona para os dois casos sem nenhuma modificação. Quando $\mathcal{S}$ possui sítios no aquém *e* no além o diagrama ainda é bem definido, mas possui vértices no infinito, o que constitui um caso especial a ser tratado no procedimento de identificação de arestas contribuintes.

O capítulo 6 apresenta uma imagem do diagrama com sítios no aquém e no além gerada com uma implementação que trata esse caso especial.

5.2.1 Calculando os circuncentros

Os circuncentros de três pontos a, b e c no $\mathbb{T}^2$, usados na construção do $\mathrm{DVor}_{\mathbb{T}^2}$, são dois pontos antípodas cir e $\neg cir$, veja figura 5.2(b) e (c). Dadas as coordenadas homogêneas de a, b e c, podemos calcular as coordenadas de cir pela interseção de dois dos bissetores, somente com operações inteiras, sem nenhuma divisão. Isto implica que se as coordenadas de a, b e c forem racionais, então as coordenadas de cir também o serão. É interessante notar isto, pois, no $\mathrm{DVorW}_{\mathbb{T}^2}$, as coordenadas dos circuncentros são em geral irracionais, mesmo quando todas as coordenadas dos sítios são racionais. Isso implica num custo maior para a computação exata de predicados envolvendo esses circuncentros irracionais.

A figura 5.2(a) mostra o cálculo implementado para o $\mathrm{DVor}_{\mathbb{T}^2}$:

Procedimento 3 `circunCentros( a, b, c, cir1, cir2 )` *calcula os dois circuncentros dos sítios* `a,b` *e* `c` *e os retorna nos pontos* $\mathtt{cir}_1$ *e* $\mathtt{cir}_2$*:*

1. Calcule os pontos médios Mab e Mbc.
2. Calcule os pontos no infinito $Pab = \mathrm{norm}(a \vee b)$ e $Pbc = \mathrm{norm}(b \vee c)$.
3. Calcule a interseção das retas $(Mab \vee Pab)$ e $(Mbc \vee Pbc)$.

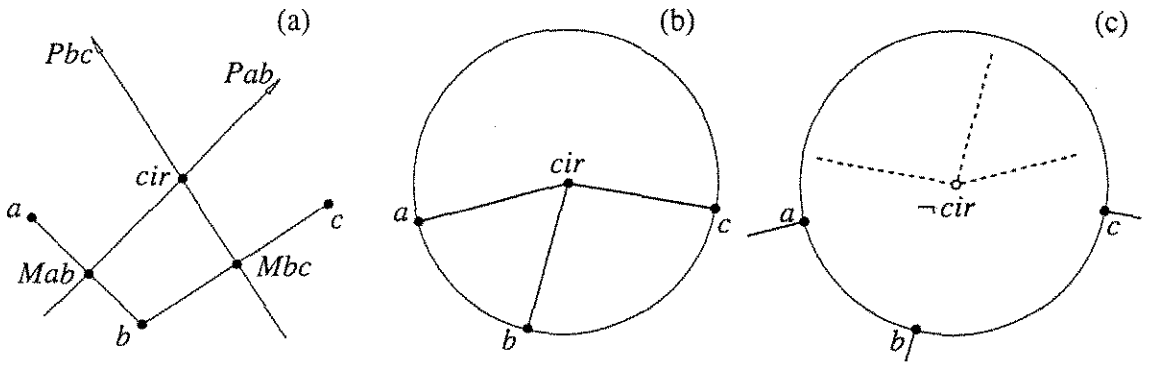

Figura 5.2: Circuncentros de três pontos no $\mathrm{DVor}_{\mathbb{T}^2}$

5.2.2 Construção do caso base

O caso base implementado para o $\mathrm{DVor}_{\mathbb{T}^2}$ é $n = 3$, onde n é o número de sítios, exatamente como descrito na seção 5.3.2 para o $\mathrm{DVorW}_{\mathbb{T}^2}$. No entanto, apesar de o diagrama estar bem definido, a representação geométrica das arestas por meio de segmentos do $\mathbb{T}^2$ não é possível quando $n = 3$ (veja seção 4.2). Vale a pena, então, discutir o caso base $n = 4$ que apresenta no $\mathbb{T}^2$ uma simetria muito interessante em topologia. Isto torna a sua construção quase totalmente mecânica (chamamos neste contexto, de mecânico, um procedimento que não envolve instruções condicionais).

No $\mathbb{E}^2$, o DVor de 4 sítios possui duas configurações topológicas distintas [21]. Uma com 2 vértices e 5 arestas e outra com 3 vértices e 6 arestas, figura 5.3(a). A última configuração ocorre quando um dos sítios não pertence à fronteira da envoltória convexa do conjunto.

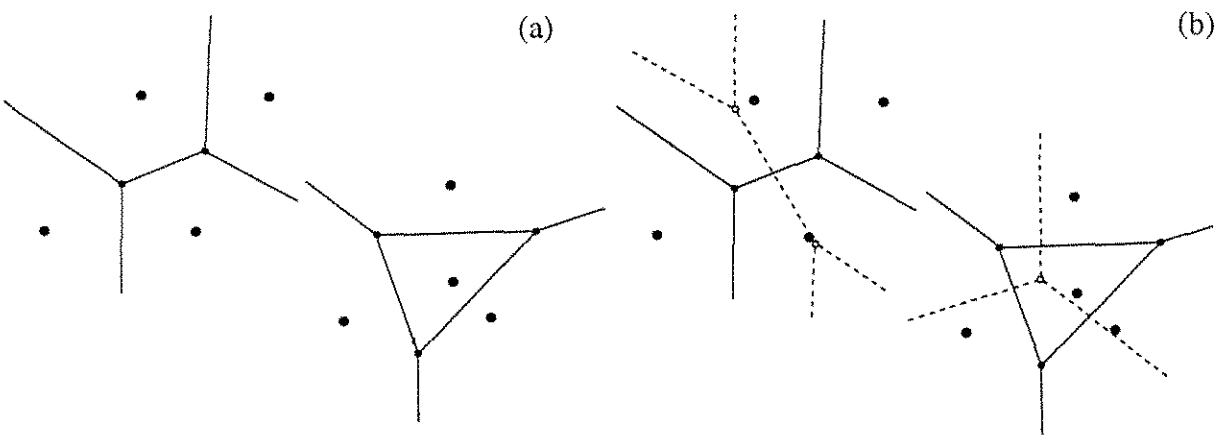

Figura 5.3: Caso base $n = 4$ no $\mathrm{DVor}_{\mathbb{T}^2}$

No $\mathbb{T}^2$, entretanto:

1. Há sempre $3n - 6 = 6$ arestas e $2n - 4 = 4$ vértices.

2. Toda região é limitada por 3 arestas e cada vértice está conectado aos outros três por uma aresta, figura 5.3(b).

3. Cada combinação dos quatro sítios tomados três a três gera um dos vértices.

Assim, o trabalho consiste em descobrir, dada uma combinação dos quatro sítios tomados três a três, qual de seus dois circuncentros é o vértice que ela define. Uma vez calculados os 4 vértices, o diagrama é construído pela conexão apropriada, com a estrutura *quad-edge*, de cada vértice aos outros três.

O vértice definido pela combinação de três sítios é justamente aquele que é centro de um disco de liberdade (que não contém o quarto sítio). O teste do parabolóide [22] pode ser usado para decidir isto, mas como já temos calculadas as coordenadas dos vértices, pode ser mais prático comparar o quadrado das distâncias entre o vértice e um dos sítios que o definiram, e entre o vértice e o quarto sítio.

5.2.3 Arestas contribuintes

No $\mathrm{DVor}_{\mathbb{T}^2}$, uma aresta não orientada pode contribuir, no máximo, com um vértice. Isto porque toda aresta é um segmento de reta (que não contém pontos antípodas) e os dois circuncentros candidatos são antípodas. Na inclusão do sítio k, o procedimento para decidir se uma dada aresta orientada e, definida pelos sítios e_{esq} e e_{dir}, com origem no vértice e_{orig} e destino em e_{dest}, é contribuinte ou não, consiste do cálculo dos circuncentros de k, e_{esq} e e_{dir}, e duas chamadas ao procedimento `ordemCircular()`, uma para cada circuncentro:

Procedimento 4 `arestaNaoContribui( e )` *retorna* `true` *se a aresta* `e` *não contribui com vértices para* R_k *e* `false` *caso contrário:*

```
point a, b;
circunCentros( eesq, edir, k, a, b );
if ( ordemCircular( eorig, a, edest, eesq ) ||
     ordemCircular( eorig, b, edest, eesq ) ) return false;
return true;
```

A figura 5.4 mostra o mesmo diagrama da figura 2.2. Na parte (a), a aresta orientada e, com origem e_{orig} e destino e_{dest}, é contribuinte pois o circuncentro *cir* contido no aquém está sobre ela. Como a aresta é um segmento do $\mathbb{T}^2$ e as regiões de Voronoi são estreladas em relação aos sítios correspondentes, a ordem circular anti-horária ao redor de e_{esq} é, necessariamente, e_{orig} *cir* e_{dest}. Assim, `ordemCircular(`$e_{orig}, cir, e_{dest}, e_{esq}$`)`

retorna `true`. O procedimento `ordemCircular()` aplicado ao outro circuncentro $\neg cir$ retorna `false`. Note que não precisa haver qualquer distinção entre arestas contidas no aquém e as contidas no além ou as que atravessam o infinito, como mostram as partes (b) e (c) da figura. As duas partes da figura mostram arestas não contribuintes pois o procedimento `ordemCircular()` não é satisfeito por nenhum dos dois circuncentros.

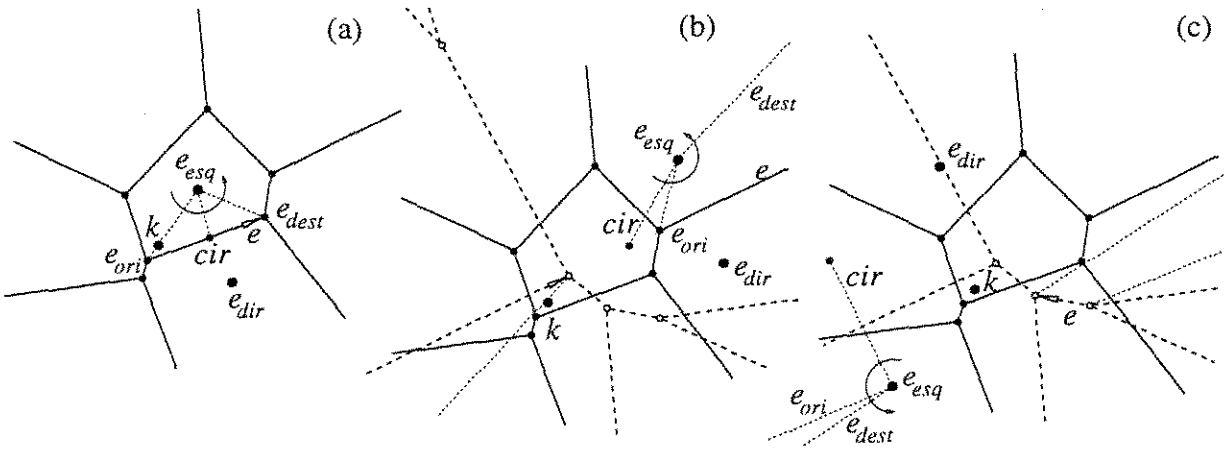

Figura 5.4: Procedimento para decidir se aresta é contribuinte no $\mathrm{DVor}_{\mathbb{T}^2}$

5.2.4 Atualização do diagrama

Dada uma aresta orientada contribuinte inicial e, as outras arestas são encontradas com um percurso no diagrama, em sentido anti-horário, com auxílio dos operadores *onext* e *rprev* da estrutura *quad-edge*: $onext(e)$ devolve a aresta seguinte com a mesma origem de e, em sentido anti-horário; $rprev(e)$ devolve a aresta anterior com a mesma face direita de e, em sentido anti-horário, veja figura 5.5(a).

Considere a figura 5.5(b). Em sentido anti-horário, o contorno da região do novo sítio atravessa a aresta contribuinte 1 e entra na sua face esquerda. O próximo passo é, como no algoritmo original de Green e Sibson, saber por qual outra aresta, o contorno da região sai da face. Na iteração do algoritmo, após uma aresta contribuinte e, tomaremos a sua aresta $onext(e)$ e percorreremos a face esquerda de e com o operador *rprev* sempre testando as arestas. Repetiremos o procedimento até que a próxima aresta contribuinte seja 1 novamente. A figura 5.5(b) mostra, com um traço mais largo, as arestas não orientadas percorridas por este procedimento na atualização da região do novo sítio.

O trecho de código a seguir implementa em C/C++ o procedimento discutido acima:

Código 1

```
primeiraAresta = e = encontraAlgumaArestaContribuinte();
```

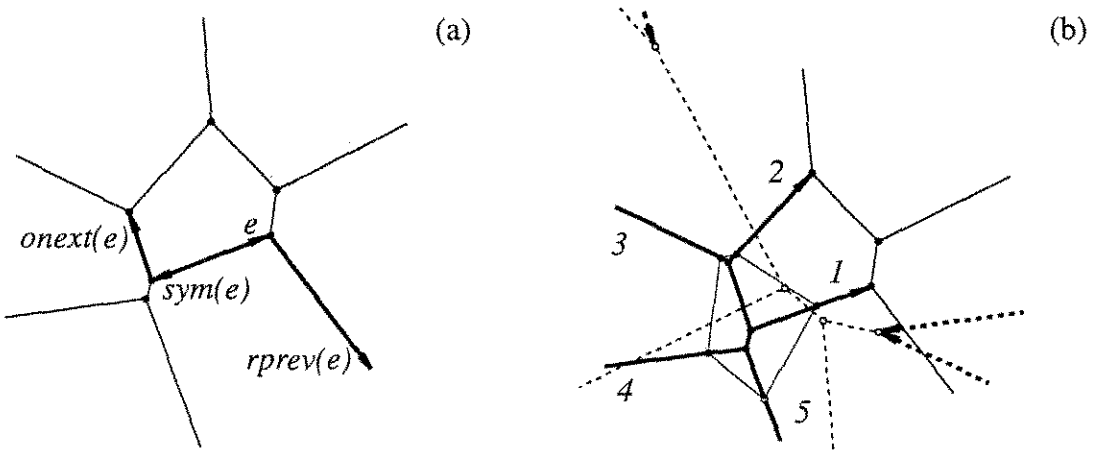

Figura 5.5: Encontrando os vizinhos do novo sítio

```
do {
   processaArestaContribuinte( e );
   e = onext( e );
   while ( e != primeiraAresta && arestaNaoContribui( e ) )
         e = rprev( e );
} while ( e != primeiraAresta );
```

Nesse código, a função `encontraAlgumaArestaContribuinte()` implementa a primeira fase do algoritmo. Como discutimos no capítulo 2, esta função não afeta a complexidade de pior caso do algoritmo. O caminhamento sugerido em [19] pode ser usado, pois a região do sítio mais próximo a k tem que ter uma aresta contribuinte.

A função `processaArestaContribuinte()` pode fazer a atualização topológica e geométrica do diagrama à medida em que as arestas contribuintes são encontradas. No entanto, é mais *simples* apenas guardar as arestas contribuintes e os vértices que elas definem numa lista simples e efetuar a atualização depois. A figura 5.6 mostra como foi implementada esta última abordagem:

1. percorra a lista de arestas contribuintes desconectando suas origens (parte (a)).
2. percorra a lista de vértices construindo a nova região (parte (b)).
3. percorra a lista de arestas contribuintes conectando suas origens aos vértices correspondentes (parte(c)) e apague as arestas que ficaram desconectadas.

Esse procedimento para atualização topológica revela um problema. Se a função `encontraAlgumaArestaContribuinte()` houvesse retornado não a aresta orientada 1, mas sim a sua simétrica $sym(1)$ (veja figura 5.5(a) para exemplo de aresta simétrica), as

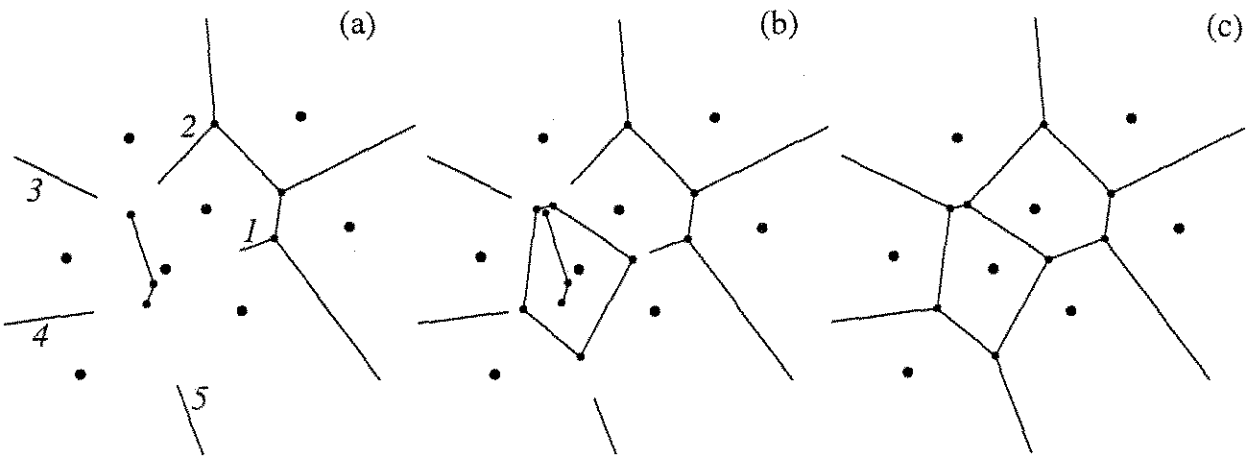

Figura 5.6: Manipulação topológica na atualização do $\mathrm{DVor}_{\mathbb{T}^2}$

arestas do diagrama percorridas pelo algoritmo seriam as mostradas pela figura 5.7(a). Nesse caso, deveríamos alterar não a origem mas o destino das arestas contribuintes durante a atualização. Felizmente, há uma maneira bastante prática para resolver esse problema. Note que, para o $\mathrm{DVor}_{\mathbb{T}^2}$, as arestas contribuintes ligam sempre um vértice que permanece no diagrama a um que desaparece. Se a atualização é feita na origem das arestas contribuintes, devemos garantir que a aresta inicial está orientada de um vértice que desaparece para um que permanece, compare figura 5.5(b). É interessante observar que não é preciso olhar para os vértices para decidir qual das duas arestas orientadas deve ser escolhida. A figura 5.7(b) mostra como usar o procedimento `ordemCircular()` para essa tarefa:

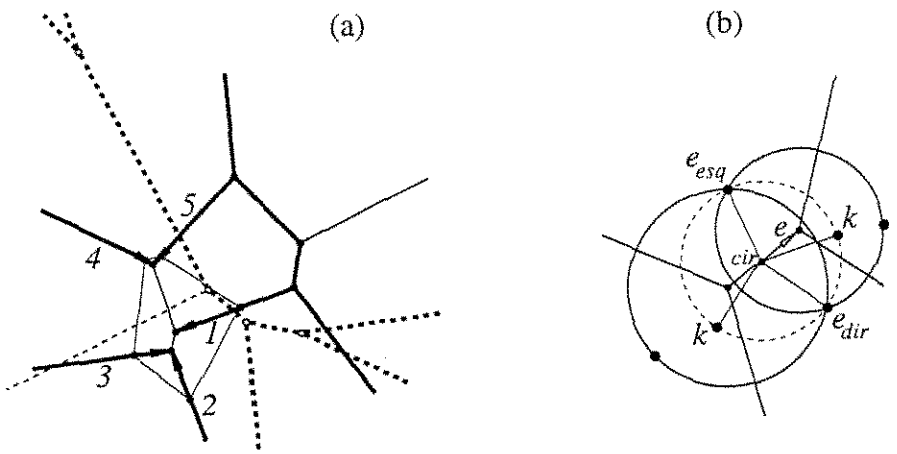

Figura 5.7: Escolhendo a aresta orientada correta

Procedimento 5 *Sejam* e_{esq} *e* e_{dir} *os sítios que definem a aresta contribuinte e à esquerda e à direita, e cir o novo vértice que* k, e_{esq} *e* e_{dir} *definem. Se* `ordemCircular(`e_{esq}`,` k`,` e_{dir}`,` cir`)` *retorna* `true` *escolha e, caso contrário escolha* $sym(e)$.

A correção deste procedimento baseia-se no fato de que e_{esq} e e_{dir} dividem a fronteira do disco de liberdade com centro em cir em dois arcos complementares, veja figura 5.7(b). O novo sítio k está sobre um desses arcos. A figura mostra duas posições possíveis para k. Note que se `ordemCircular(`e_{esq}`,` k`,` e_{dir}`,` cir`)` retorna `true`, então o vértice que desaparece é a origem de e, e_{orig}, pois nenhum dos discos de liberdade com centro entre cir e e_{dest} contém k, ao passo que todos os discos de liberdade com centro entre cir e e_{orig} contêm k.

Esse procedimento mereceu atenção especial pois veremos que, no $\mathrm{DVorW}_{\mathbb{T}^2}$, além do problema da atualização topológica, o código 1 falha em encontrar todas as arestas contribuintes, caso a aresta contribuinte inicial não esteja corretamente orientada.

5.3 Diagrama de Pontos com Peso Aditivo

A implementação para o $\mathrm{DVorW}_{\mathbb{T}^2}$ assume que o conjunto $\mathcal{S}$ não contém quatro sítios co-circulares, figura 5.8 (a) e (b). O cálculo dos circuncentros implementado não trata três sítios cujos discos correspondentes são tangentes a uma reta e três sítios colineares, Fig. 5.8(c) e (d). Esses casos, entretanto, podem ser implementados com uma rotina especial, sem afetar o algoritmo. Por fim, a implementação não trata sítios contidos em outros sítios, Fig. 5.8(e). Discutiremos, ao final da seção, as implicações no algoritmo caso esta restrição precise ser retirada. Apesar desses casos especiais, o algoritmo é *on-line* e admite que os círculos se interceptem.

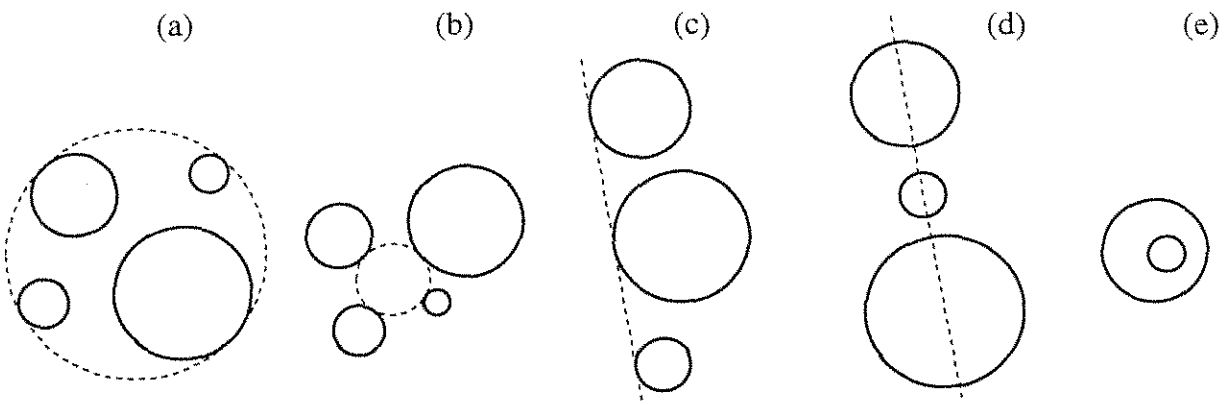

Figura 5.8: Configurações degeneradas para a implementação do $\mathrm{DVorW}_{\mathbb{T}^2}$

5.3.1 Calculando os circuncentros

No $\mathbb{E}^2$, três discos podem possuir 0, 1 ou 2 circuncentros [34], figura 5.9(a). No $\mathbb{T}^2$, há sempre 2 circuncentros, figura 5.9(b).

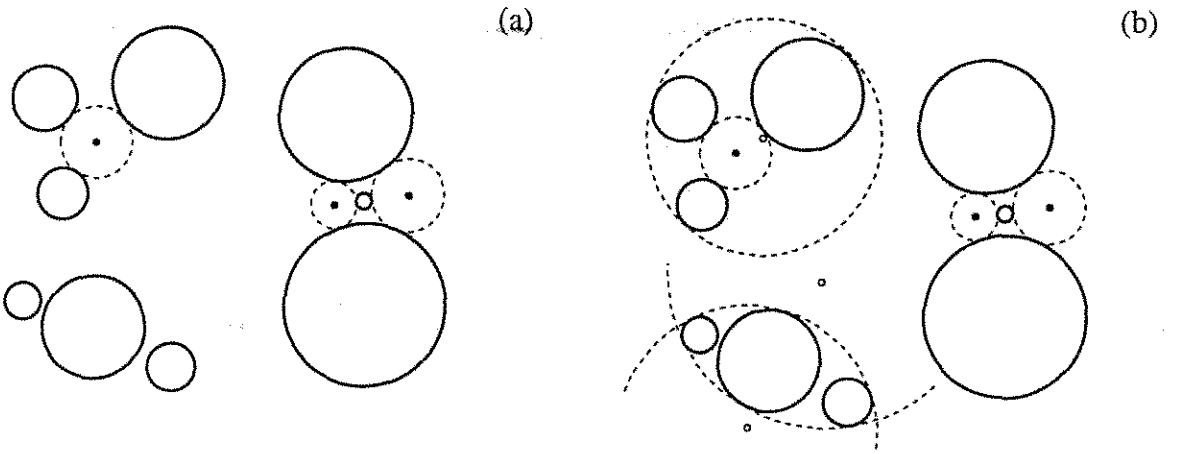

Figura 5.9: Circuncentros de três sítios no $\mathrm{DVorW}_{\mathbb{T}^2}$

Dadas as coordenadas homogêneas $[x, y, w; p]$, onde p é o peso, de três sítios a, b e c, podemos calcular as coordenadas dos dois circuncentros sem nenhuma divisão, mas não podemos evitar a raiz quadrada. Em geral, as coordenadas podem ser irracionais mesmo quando todas as coordenadas dos sítios são inteiras. A figura 5.10(a) mostra um exemplo simples em que $a = [1, 2, 1; 3]$, $b = [-1, 0, 1; 1]$ e $c = [2, -3, 1; 1]$. Os dois circuncentros possuem coordenadas $[3/2 - \sqrt{11}/2, -1/2 - \sqrt{11}/2, 1; -2 + \sqrt{11}]$ e $[-3/2 - \sqrt{11}/2, +1/2 - \sqrt{11}/2, -1; -2 - \sqrt{11}]$.

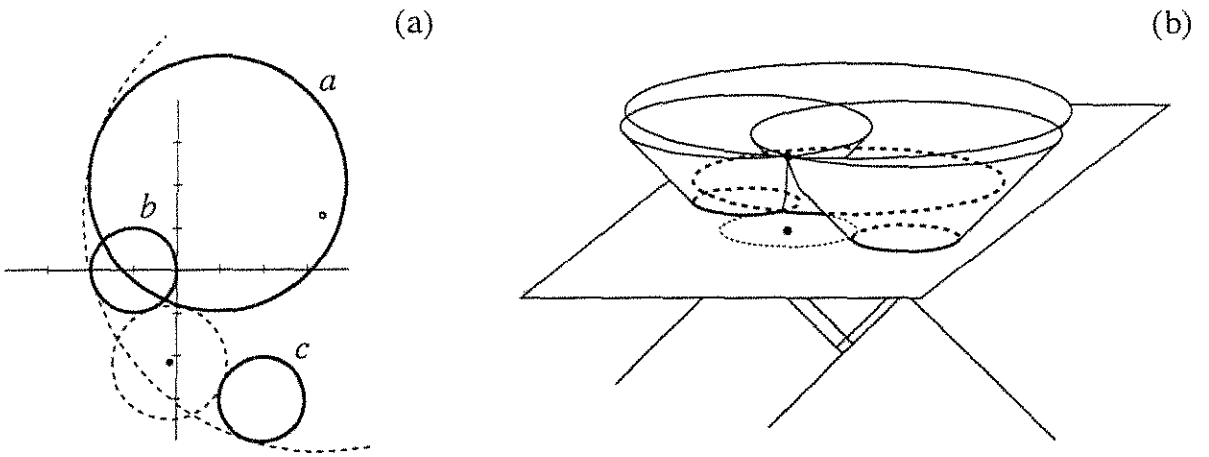

Figura 5.10: Cálculo dos circuncentros no $\mathrm{DVorW}_{\mathbb{T}^2}$

Os circuncentros podem ser encontrados, como no $\mathrm{DVor}_{\mathbb{T}^2}$, com a interseção de dois dos bissetores dos sítios, que nesse caso são hipérboles. O problema é que duas hipérboles podem se interceptar em quatro pontos. Restaria o trabalho de identificar quais dois dos quatro pontos seriam os circuncentros. O cálculo implementado foi sugerido por Held [23] e encontra diretamente os dois pontos corretos. A idéia é imaginar os três discos se expandindo com o mesmo incremento no raio até que eles se interceptem num único ponto. Se imaginamos que o plano que contém os discos é o plano $z = 0$ do $\mathbb{E}^3$, e imaginamos que o incremento no eixo z é o mesmo do raio, podemos ver que isto equivale a fazer a interseção de três cones retos "assentados" sobre os discos. Esses cones são exatamente os usados em [16] e [34] para caracterizar geometricamente o algoritmo de varredura do plano para diagramas de Voronoi. Se nenhum círculo está contido em outro, esses três cones possuem exatamente dois pontos de interseção, que projetados de volta para o plano $z = 0$ são os circuncentros procurados. A figura 5.10(b) mostra uma das interseções. Nesta figura, a outra interseção está abaixo do plano.

Apresentaremos o cálculo com coordenadas cartesianas por simplicidade. Dadas as coordenadas cartesianas $(x, y; p)$ dos sítios a, b e c, as equações dos cones serão:

$$(x - a_x)^2 + (y - a_y)^2 = (a_p + z)^2 \tag{5.1}$$
$$(x - b_x)^2 + (y - b_y)^2 = (b_p + z)^2 \tag{5.2}$$
$$(x - c_x)^2 + (y - c_y)^2 = (c_p + z)^2 \tag{5.3}$$

Resolvemos esse sistema, primeiro para z (o deslocamento dos raios) e depois para x e y (esses cálculos podem ser simplificados se transladamos o sítio a para a origem antes de resolver o sistema):

1. Subtraia 5.2 de 5.1 e 5.3 de 5.1 obtendo, respectivamente:

$$2(b_x - a_x)x + 2(b_y - a_y)y + 2(b_p - a_p)z + b_p^2 - a_p^2 + a_x^2 + a_y^2 - b_x^2 - b_y^2 = 0 \tag{5.4}$$
$$2(c_x - a_x)x + 2(c_y - a_y)y + 2(c_p - a_p)z + c_p^2 - a_p^2 + a_x^2 + a_y^2 - c_x^2 - c_y^2 = 0 \tag{5.5}$$

2. Obtenha de 5.4 e 5.5 expressões de x e y em função de z. Substitua estas expressões em 5.1 e resolva a equação do segundo grau em z resultante.

3. Substitua os dois valores de z em 5.4 e 5.5 e resolva o sistema linear para x e y.

É interessante que, se os três discos não se interceptam, então uma interseção dos cones acima do plano $z = 0$ corresponde a um circuncentro do aquém e uma interseção abaixo a um circuncentro do além, conforme figura 5.10. Mas não podemos aplicar esta

regra diretamente para decidir em que lado do plano os pontos do $\mathbb{E}^3$ calculados devem ser projetados. Se os discos se interceptam, pode acontecer de um circuncentro do aquém corresponder a um ponto abaixo do plano $z = 0$. Para corrigir esse problema, basta subtrair dos pesos dos sítios, o menor peso. Assim,

$$Se\ z > 0\ projete\ no\ aquém$$
$$Se\ z < 0\ projete\ no\ além$$

Vale lembrar que os dois circuncentros podem estar ambos no aquém, ambos no além, ou um no aquém e outro no além.

Este artifício de subtrair o menor peso dos demais no cálculo dos circuncentros possibilita, também, que o algoritmo trabalhe com pesos negativos. Deste modo, podemos construir o diagrama para H_{pq} definido com $\mathrm{d}_{\mathbb{T}^2}(x,p) + w(p)$ também de maneira *on-line*. Todo algoritmo para uma função pode ser usado para a outra. Basta que se inverta a ordenação relativa dos pesos. Por exemplo, se o peso do sítio a é três unidades maior que o de b, o algoritmo precisa receber o peso de a três unidades menor que o de b. Se o algoritmo trabalha apenas com pesos positivos, para construir o diagrama para H_{pq} definido com $\mathrm{d}_{\mathbb{T}^2}(x,p) + w(p)$ de maneira *on-line*, é preciso saber, com antecedência, os pesos de todos os sítios para obter a inversão. Se o algoritmo admite pesos negativos, basta manter o peso a_p do primeiro sítio a e atribuir peso $2a_p - k_p$ para cada novo sítio k inserido. Esta conta pode gerar pesos negativos, o que será compensado pelo artifício no cálculo dos circuncentros.

A resolução do sistema como sugerida acima falha quando os três sítios são colineares, figura 5.8(d). Isto porque, neste caso, o valor de z é o mesmo para os dois circuncentros (os raios dos discos de liberdade são iguais). Mas os valores de x e y não são iguais, e como o cálculo de x e y é em função de z, a resposta é necessariamente inconsistente. Uma solução é, então, resolver o sistema primeiro para x e depois para y e z. Se os sítios estiverem na horizontal, é necessário resolver primeiro para y, e depois para x e z. Para o caso da figura 5.8(c) o sistema possui apenas uma solução, que é o circuncentro finito. Mas a abordagem sugerida não é capaz de calculá-lo. Para este caso, uma solução específica precisa ser desenvolvida.

5.3.2 Construção do caso base

O caso base implementado para o $\mathrm{DVorW}_{\mathbb{T}^2}$ é $n = 3$. Ao contrário do $\mathrm{DVor}_{\mathbb{T}^2}$, é possível representar geometricamente as arestas, dada a definição e a representação de cônicas no $\mathbb{T}^2$ do capítulo 4. É interessante que o caso $n = 4$ possui, também, sempre 6 arestas e 4 vértices, mas possui mais de uma configuração topológica, e uma combinação dos quatro sítios tomados três a três pode gerar 0, 1 ou 2 vértices. Construir este caso diretamente seria consideravelmente complicado.

A topologia do caso $n = 3$ é sempre a mesma: um circuncentro está conectado ao outro por três arestas que separam os sítios. Entretanto, a construção também não é totalmente mecânica. A figura 5.11 mostra as três configurações geométricas possíveis. Utilizaremos o seguinte procedimento para criação de uma aresta no $\mathrm{DVorW}_{\mathbb{T}^2}$:

Procedimento 6 `criaAresta( i, f, p, q )` *cria uma aresta topológica com origem em* `i` *e destino em* `f`. *Cria uma aresta geométrica com* $f_1 = \mathtt{p}$, $f_2 = \neg\mathtt{q}$, $d = \mathtt{i}$, $a_1 = \mathtt{i}$ *e* $a_2 = \mathtt{f}$, *de acordo com a representação do capítulo 4.*

Para obter a representação geométrica das arestas usamos os circuncentros como os pontos a_1 e a_2 na representação geométrica. Por exemplo, na figura 5.11(a), o arco de hipérbole h é definido como o caminho traçado pela interseção da hipérbole com a reta $(a \vee c_1)$, quando rotacionamos a reta, em sentido anti-horário ao redor de a, até que ela seja igual a $(a \vee c_2)$. Surge um problema porque não há como definir uma ordem circular de c_1 e c_2 ao redor de a porque todas as regiões do diagrama são compostas por somente duas arestas unindo os dois circuncentros. Assim, precisamos escolher corretamente qual circuncentro será a_1, na representação geométrica, para não tomar os arcos complementares por engano. Esse erro também provocaria falha do procedimento `arestaNaoContribui()` (que depende essencialmente da ordem dos vértices de uma região) na inserção do próximo sítio.

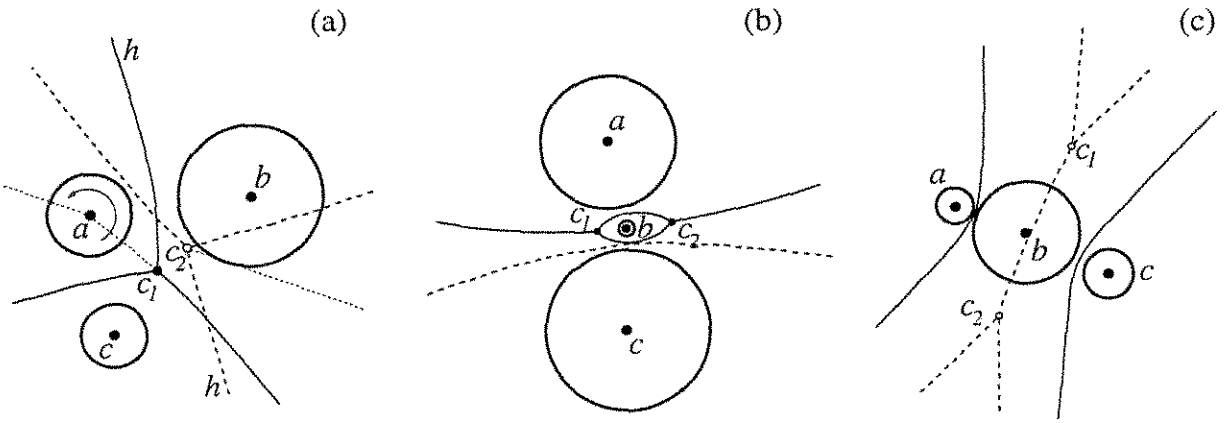

Figura 5.11: Caso base $n = 3$ no $\mathrm{DVorW}_{\mathbb{T}^2}$

O procedimento implementado foi:

Procedimento 7 `casoBase( p, q, r )` *constrói* $DVorW_{\mathbb{T}^2}(S)$ *para* $S = \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$ *e retorna um ponteiro para uma das arestas do diagrama:*

```
point i, f;
circunCentros( p, q, r, i, f );
if ( ordemCircular( p, q, r, i ) ) swap(i, f);
criaAresta(i, f, p, q);
criaAresta(i, f, q, r);
return criaAresta(i, f, r, p);
```

A instrução `if` garante que o arco correto será representado. A correção deste procedimento está baseada, novamente, no fato de que as regiões são estreladas em relação ao sítio. Se estivermos tomando os arcos complementares, trocar c_1 por c_2 nos fará tomar os arcos corretos, veja figura 5.11. Não é difícil ver que, como as arestas separam os sítios e estão conectadas entre os circuncentros e as regiões são estreladas, a ordem circular de a, b e c em relação a um circuncentro será necessariamente contrária à ordem em relação ao outro. Assim, como convencionamos criar arestas entre a e b, b e c e entre c e a nesta ordem, devemos escolher para c_1 o circuncentro que avista a ordem acb, em sentido anti-horário para que os arcos corretos sejam representados.

5.3.3 Arestas contribuintes

No $\mathrm{DVorW}_{\mathbb{T}^2}$, uma aresta não orientada pode contribuir com até dois vértices. No entanto, diremos que uma aresta orientada contribui com 0 ou 1 vértice. A figura 5.12(a) apresenta quatro arestas, e_1, e_2, e_3 e e_4 num diagrama de sete sítios. Note que e_4 é a simétrica de e_3. Na inclusão do sítio k (figura 5.12(b)) a aresta e_1 não contribui, mas e_2, e_3 e e_4 contribuem, respectivamente, com os vértices c_2, c_3 e c_4. A figura 5.12(c) apresenta o diagrama resultante apenas como ilustração.

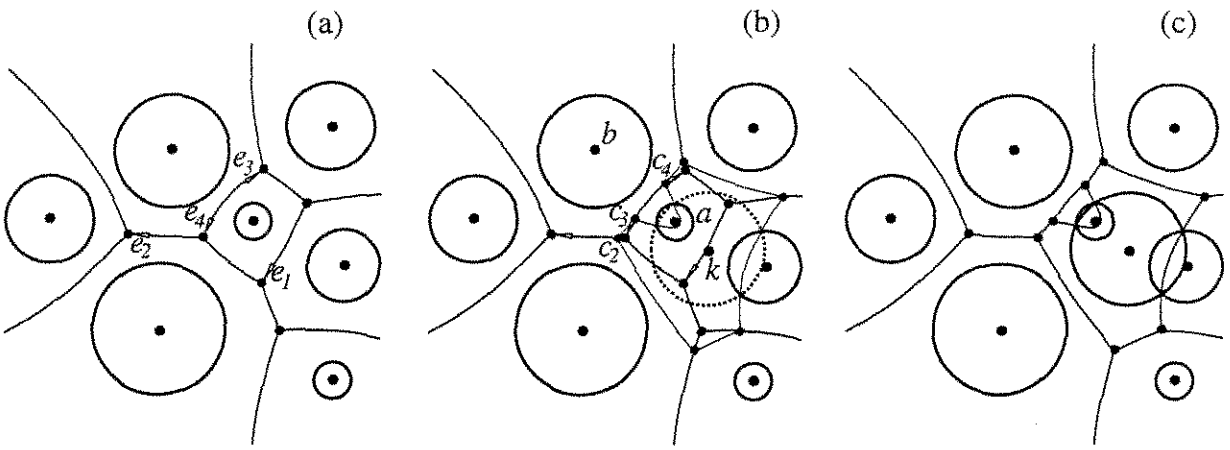

Figura 5.12: Arestas contribuintes no $\mathrm{DVorW}_{\mathbb{T}^2}$

O procedimento para decidir se uma dada aresta orientada e contribui é idêntico ao do $\text{DVor}_{\mathbb{T}^2}$, pois as regiões são estreladas em relação ao seu sítio. Sejam cir_1 e cir_2 os dois circuncentros definidos por e_{esq}, e_{dir} e k. Se `ordemCircular`$(e_{orig}, cir_1, e_{dest}, e_{esq})$ ou `ordemCircular`$(e_{orig}, cir_2, e_{dest}, e_{esq})$ retorna `true` então a aresta e contribui.

Se a aresta não orientada de e contribui com dois vértices, e contribuirá com um vértice, tal que ele esteja entre e_{orig} e o outro vértice. A figura 5.14(a) mostra o porquê desta atribuição. Nela, o contorno de R_k não pode sair da região de e_{esq} por cir_2 pois ele teria que entrar de novo por cir_1 e não conseguiria sair depois. Assim se `ordemCircular`$(e_{orig}, cir_1, cir_2, e_{esq})$ retorna `true`, então e contribui com cir_1 e $sym(e)$ com cir_2 (como na figura 5.12(a) e (b)). Se for `false`, será o caso contrário. Veremos adiante que esta atribuição garante que a atualização topológica e geométrica de e seja correta.

5.3.4 Atualização do diagrama

A atualização do $\text{DVorW}_{\mathbb{T}^2}$ é feita com o mesmo código 1 apresentado na seção 5.2.4. Entretanto, não é imediato perceber que este código corretamente encontra os novos vértices. Isto porque as regiões dos sítios não são necessariamente convexas, e dois sítios podem ser vizinhos por várias arestas, e não por apenas uma como no $\text{DVor}_{\mathbb{T}^2}$. Assim, o contorno da região do novo sítio k possui um comportamento mais complexo. Ele pode:

1. Sair de uma região imediatamente pela mesma aresta não orientada por onde entrou (veja região do sítio a na figura 5.12(b) ou figura 5.13(b));

2. Atravessar uma região mais de uma vez (veja região do sítio b na figura 5.12(b) ou figura 5.13(b)).

Se a função `encontraAlgumaArestaContribuinte()` retornar uma aresta orientada de um vértice que permanece no diagrama para um que desaparece, o procedimento nem sequer encontra todos os vértices da nova região. A figura 5.13(a) mostra as arestas não orientadas percorridas pelo código 1, para o diagrama da figura 5.12, quando a primeira aresta contribuinte é a aresta orientada 1. O procedimento não irá percorrer a aresta entre os sítios a e b que, de fato, contribui com dois vértices.

Para evitar este problema, usamos o mesmo procedimento 5 da seção 5.2.4. É interessante observar que a generalização daquele procedimento do $\text{DVor}_{\mathbb{T}^2}$ para o $\text{DVorW}_{\mathbb{T}^2}$ é imediata e ele pode, então, ser usado para decidir qual aresta orientada devemos usar, caso a função `encontraAlgumaArestaContribuinte()` retorne uma aresta não orientada que contribui com apenas um vértice. Caso a aresta retornada contribua com dois vértices, ambas as arestas orientadas podem ser usadas como a aresta inicial.

Quando utilizamos, como aresta inicial, uma aresta cuja origem é um vértice que desaparece no novo diagrama, garantimos que o percurso que o código 1 executa encontra

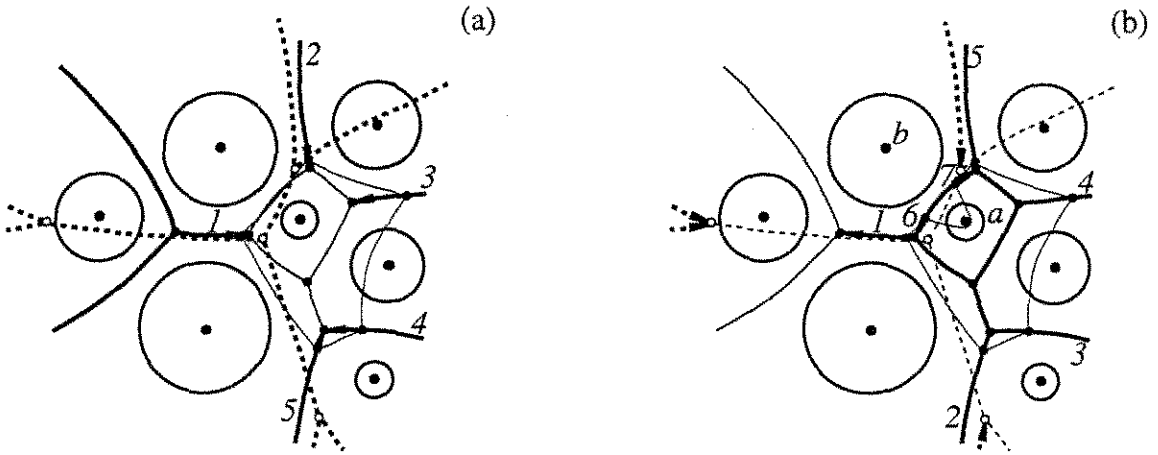

Figura 5.13: Procedimento de atualização do $\mathrm{DVorW}_{\mathbf{T}^2}$

todas as arestas contribuintes, e seus respectivos vértices, na ordem anti-horária correta. A figura 5.13(b) apresenta as arestas não orientadas percorridas quando tomamos a aresta orientada de 1 correta. Como esclarecimento, alertamos que, nesta figura, a sexta aresta contribuinte percorrida é a indicada pelo número 6 próximo à seta que indica o seu *destino*. No entanto, o vértice que ela define é o que está próximo à sua *origem*.

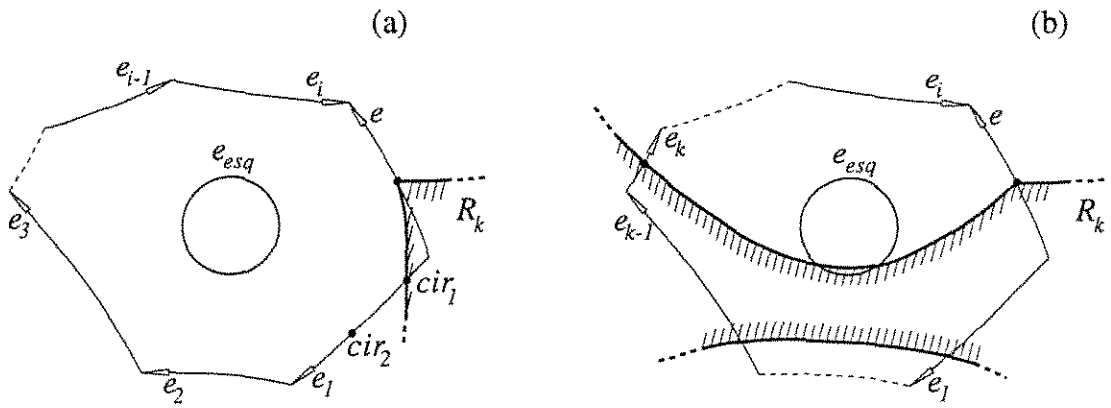

Figura 5.14: Correção da função de percurso

Este exemplo das arestas orientadas simétricas 6 e 7 constitui um caso interessante para mostrarmos a correção do código 1. Note que o código, depois de processar a aresta 6 irá percorrer todas as arestas da região do sítio a até encontrar a aresta 7. O contorno da região do novo sítio entra e imediatamente sai da região de a pela mesma aresta não orientada. É fácil ver que o código falharia caso este contorno pudesse, mais tarde, entrar novamente na região de a. Neste caso, a aresta 7 não seria encontrada, pois o percurso encontraria alguma outra aresta contribuinte antes de 7. Felizmente há uma maneira

interessante de se verificar que isto não é possível.

De maneira geral, se o contorno de R_k sai de $R_{e_{esq}}$ por e_k, veja figura 5.14(b), então nenhuma das arestas $e_1, e_2, ..., e_{k-1}$, que são testadas antes de e_k, pode ser contribuinte e o código irá chegar a e_k. Isso é devido ao fato de que toda parte do contorno cruzando $R_{e_{esq}}$ é uma parte do bissetor de k e e_{esq}. Se qualquer das $e_1, e_2, ..., e_{k-1}$ pudesse ser contribuinte, então $R_{e_{left}}$ ficaria desconexa, o que é impossível. Isso implica também que todas as arestas percorridas, exceto as contribuintes, estão inteiramente contidas em R_k.

Atualização topológica Para a atualização topológica existem dois casos. A caracterização mais intuitiva para os casos é por meio dos vértices que desaparecem na inclusão do sítio k. Se ao menos um vértice do diagrama desaparece, o procedimento é idêntico ao do $\mathrm{DVor}_{\mathbf{T}^2}$. Para esse caso, vamos detalhar apenas a criação das arestas de R_k para mostrar como a representação geométrica é obtida no algoritmo. Nesse caso, R_k contém ao menos três arestas e a criação delas é feita da seguinte maneira. Seja n o número de vértices de R_k, $V(i)$ o i-ésimo vértice e $T(i)$ o sítio da região esquerda da aresta orientada que contribuiu com o vértice $V(i)$:

1. Para $i = 1$ até n crie aresta $E(i) =$ `criaAresta`$(V(i), V(i+1), k, T(i))$, onde $V(n+1) = V(1)$.

2. Para $i = 1$ até n conecte topologicamente o destino de $E(i)$ à origem de $E(i+1)$, onde $E(n+1) = E(1)$.

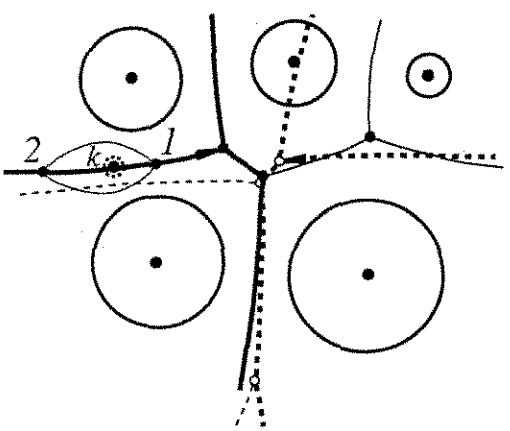

Figura 5.15: Caso especial na atualização do $\mathrm{DVorW}_{\mathbf{T}^2}$

Quando a inclusão de k não faz desaparecer nenhum vértice, a sua região contém apenas duas arestas e dois vértices, e há apenas uma aresta do diagrama que contribui com os dois vértices. A figura 5.15 mostra este caso e apresenta as arestas percorridas pelo código 1, que são exatamente as arestas das duas regiões que compartilham a aresta

contribuinte. Dada uma das arestas orientadas contribuintes e. Sejam cir_e e $cir_{sym(e)}$ os vértices que ela e a sua simétrica definem:

1. Crie as arestas $E(1) = \texttt{criaAresta}(cir_e, cir_{sym(e)}, e_{esq}, k)$,
   $E(2) = \texttt{criaAresta}(cir_e, cir_{sym(e)}, k, e_{dir})$,
   $E(3) = \texttt{criaAresta}(e_{orig}, cir_e, e_{esq}, e_{dir})$,
   $E(4) = \texttt{criaAresta}(cir_{sym(e)}, e_{dest}, e_{esq}, e_{dir})$.

2. Conecte topologicamente: as origens e destinos de $E(1)$ e $E(2)$; a origem de $E(3)$ à origem de e; o destino de $E(3)$ à origem de $E(1)$; o destino de $E(1)$ à origem de $E(4)$ e o destino de $E(4)$ ao destino de e. Apague e.

Vale a pena notar que, apesar da aparente semelhança entre este segundo caso e o caso base, a atualização geométrica neste caso é mecânica, pois a orientação da aresta contribuinte e nos dá automaticamente o vértice correto para a origem das arestas na representação geométrica.

5.3.5 Considerações sobre a implementação

Uma das vantagens mais interessantes dos algoritmos incrementais é a possibilidade de que sejam *on-line*, ou seja, em momento algum, o algoritmo faz qualquer suposição a respeito do próximo sítio a ser incluído. No início da seção, fizemos a restrição de que um sítio não poderia estar inteiramente contido em outro. Se, dependendo da aplicação, esta restrição precisar ser retirada, então duas modificações no algoritmo precisam ser feitas para que ele continue *on-line*:

1. O procedimento de cálculo dos circuncentros precisa verificar se algum dos três sítios está contido em outro, caso em que não há circuncentros. A aresta correspondente *não* será contribuinte.

2. O procedimento de atualização precisa ser capaz de dizer se a inclusão do sítio k reduz o diagrama ao caso base, quando a atualização é diferente. Note que a inclusão de um sítio pode reduzir o diagrama ao caso $n = 1$.

Se a aplicação não necessita de um algoritmo *on-line*, podemos incluir os sítios em ordem decrescente de pesos, facilitando o trabalho. Ou podemos comparar os sítios dois a dois eliminando os que estiverem inteiramente contidos em outros, antes de iniciar a inclusão, pois esses sítios não contribuem para o diagrama.

Capítulo 6

O Ambiente GeoPrO

As anyone who has tried to implement a complex geometric algorithm knows, implementing geometric algorithms is a difficult task. Conventional tools are limited as aids in this process. The programmer spends time with pen and pencil drawing the geometry and data structures the program is developing. This problem could be solved by the use of visualization tools. In the ideal world, this visualization would be used for three purposes: demonstration, debugging, and isolation of degeneracies.

— D. Dobkin [20]

O GeoPrO é um ambiente de visualização distribuída, desenvolvido no Instituto de Computação - UNICAMP [18], projetado para dar suporte a aplicações em geometria computacional. Ele consiste de um *núcleo*, e de interfaces de programação para implementação de aplicações e visualizadores capazes de estabelecer conexão com o núcleo. A idéia é manter no núcleo conjuntos de *objetos* geométricos, chamados *contextos* de visualização, que podem ser compartilhados pelas aplicações e visualizadores. As conexões são feitas por canais TCP/IP na Internet, tornando possível a cooperação de plataformas diferentes em redes heterogêneas.

Todo o funcionamento do ambiente está baseado na troca de mensagens pré-definidas entre o núcleo e as aplicações e visualizadores. Cada instância de aplicação ou visualizador pode conectar-se a apenas um contexto no núcleo, onde deseja trabalhar. No entanto, o núcleo manipula vários contextos; permitindo, assim, que grupos independentes compartilhem seus recursos. Como exemplo, uma típica seqüência de interação entre o núcleo e uma aplicação seria: a aplicação envia uma mensagem de conexão, especificando o contexto no qual deseja trabalhar; o núcleo aceita a conexão enviando uma mensagem de reconhecimento; a aplicação, então, insere ou remove objetos no contexto. Todos os visualizadores que estiverem conectados àquele contexto vão receber uma cópia dos objetos

inseridos e poderão apresentá-los na sua área de desenho.

A principal característica do ambiente é que, uma vez inserido num contexto, um objeto torna-se *persistente*. Mesmo que a aplicação que o inseriu termine ou seja interrompida, o objeto permanece no contexto e, conseqüentemente, visível nos visualizadores conectados a ele. Outra funcionalidade, que foi importante para a visualização dos diagramas, é a *requisição* de objetos pelas aplicações aos visualizadores. Uma aplicação pode precisar, por exemplo, de um conjunto de pontos. Ela envia uma requisição ao contexto e qualquer visualizador conectado àquele contexto pode responder à requisição. Assim, existe um caminho pelo qual um visualizador pode funcionar como interface de entrada de dados.

As implementações do algoritmo O algoritmo foi implementado como classes que são instanciadas em duas aplicações do ambiente GeoPrO: `DVOR` e `DVORW`. Ambas funcionam da seguinte maneira:

1. Cadastram-se no núcleo ao contexto especificado pelo usuário na linha de comando que as executou; e inserem nele, imediatamente, uma requisição por um conjunto de objetos (pontos para o `DVOR` e círculos para o `DVORW`). Ficam aguardando algum visualizador responder à requisição.

2. Após receber o conjunto de objetos, instanciam a classe do algoritmo e inserem os sítios um a um com o método `includeSite()`.

3. Inserem no contexto um conjunto de objetos (segmentos de reta para o `DVOR` e arcos de cônicas e/ou segmentos de reta para o `DVORW`); todos os visualizadores conectados ao contexto podem desenhar o diagrama segundo seus próprios recursos.

6.1 Protocolo de Descrição Geométrica

Os objetos geométricos que são inseridos nos contextos são sempre representados através do Protocolo de Descrição Geométrica (PDG). Este protocolo descreve qualquer objeto tratado pelo núcleo por um conjunto de objetos básicos. Na concepção inicial do ambiente, o PDG era composto de ponto, reta, segmento de reta, polígono, linha poligonal e círculo. Todos estes objetos básicos são, por sua vez, descritos por pontos em coordenadas homogêneas com sinal.

A intenção de ampliar o conjunto de objetos básicos já existia em [18]. A utilização do ambiente GeoPrO para implementação dos algoritmos e visualização dos resultados deste trabalho nos levou a estender o PDG para dar suporte a cônicas: o objeto `conics` consiste de uma lista de cônicas descritas por três pontos, segundo a representação do

capítulo 4; o objeto `conic arcs` consiste de uma lista de arcos de cônica descritos por cinco pontos.

A maioria dos problemas tratados em geometria computational envolve apenas objetos lineares. Entretanto há um crescente interesse por problemas envolvendo objetos mais complexos [20]. A razão é simples: muitos problemas práticos são melhores descritos, ou aproximados, quando usamos objetos não lineares. A descrição de objetos e estruturas não lineares por objetos lineares pode ser muito custosa e ineficiente. As cônicas são o primeiro passo na generalização dos problemas em geometria computacional.

6.2 Gerenciador

O núcleo do GeoPrO é um processo sem interface gráfica. Anteriormente, o usuário que o executava não dispunha de qualquer informação sobre os contextos, aplicações e visualizadores cadastrados. A única maneira de manipular os dados no núcleo era através das interfaces para aplicações e visualizadores. Além disso, ele não provia nenhuma maneira de guardar em disco os objetos dos contextos, de forma que a existência dos objetos se limitava ao tempo de execução do núcleo. Se o núcleo era abortado, por qualquer problema, todo o trabalho era perdido.

Na sua tese, Gon [18] sugeriu várias extensões ao ambiente GeoPrO, inclusive a criação de um módulo gerenciador para solucionar os problemas citados. Nós projetamos e implementamos um gerenciador em linguagem Java que permite:

1. Acesso a todas as informações do núcleo: quais aplicações e visualizadores estão cadastrados; quais contextos existem e quais objetos e requisições tem cada contexto;

2. Criar contextos; destruir contextos; salvar em disco objetos de um contexto; recuperar objetos do disco e destruir objetos. Essas funções, em particular, foram de extrema utilidade na fase de testes das implementações dos algoritmos e tornaram o ambiente bastante mais amigável.

Muitas melhorias são necessárias ainda para tornar o gerenciador mais útil. Especialmente, seria interessante prover níveis de gerenciamento. Atualmente, se o usuário possui a senha, ele pode conectar o gerenciador ao núcleo e ter acesso irrestrito a todas as funções. A idéia é, por exemplo, que um usuário que criou um contexto, mas não seja o dono do núcleo, possa gerenciar somente o seu contexto.

6.2.1 A interface

O projeto do gerenciador seguiu a mesma filosofia das interfaces para aplicações e visualizadores. O processo do gerenciador não precisa ser executado na mesma máquina do

núcleo. Alteramos o núcleo de forma que ele recebe conexões, pela Internet, de gerenciadores, da mesma forma que de aplicações e visualizadores. A segurança é garantida por uma senha que o usuário, que executou um núcleo em uma dada máquina, pode registrar para conexão de gerenciadores.

O gerenciador é composto por três classes principais: o diálogo de conexão; o diálogo de núcleo; e o diálogo de contexto. As duas últimas classes são *threads* que executam independentemente. O usuário executa um processo gerenciador e pode disparar conexões com vários núcleos a partir do diálogo de conexão, veja figura 6.1.

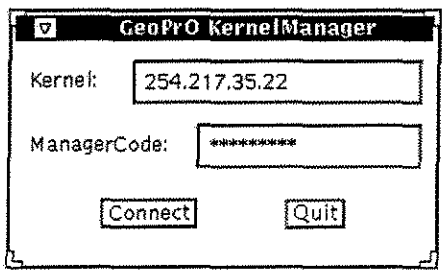

Figura 6.1: Diálogo de conexão na interface OpenLook

Quando uma conexão é aceita, o diálogo de núcleo apresenta os contextos, as aplicações e os visualizadores conectados ao núcleo, veja figura 6.2. O usuário pode então, criar, destruir e abrir os contextos, disparando diálogos de contextos.

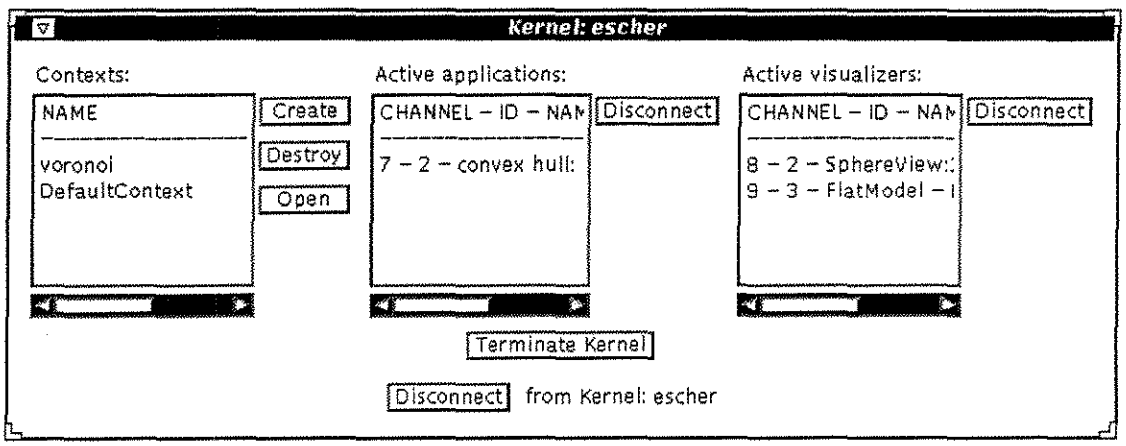

Figura 6.2: Diálogo de núcleo na interface OpenLook

O diálogo de contexto apresenta os objetos e requisições e permite: destruir, salvar e recuperar objetos, veja figura 6.3.

Uma característica importante do projeto do gerenciador é que a conexão com o núcleo é *on-line*, ou seja, as modificações que fizemos no núcleo garantem que qualquer mudança no estado dele é, imediatamente, repassada aos gerenciadores conectados.

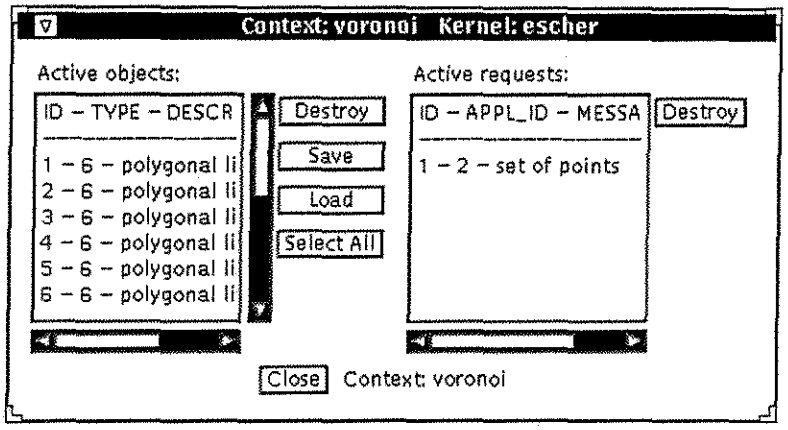

Figura 6.3: Diálogo de contexto na interface OpenLook

6.3 Visualizadores com Suporte a Cônicas

O ambiente GeoPrO conta com dois visualizadores para o $\mathbb{T}^2$, *sphereModel* e *flatModel* [18], para os modelos esférico e plano, respectivamente, que utilizam a biblioteca OpenGL [8] para primitivas básicas de desenho. Nos visualizadores, todos os objetos, com a exceção do ponto, são desenhados através de segmentos de reta, com as primitivas para desenho entre dois pontos com coordenadas homogêneas em três dimensões, que OpenGL provê. Esta seção irá descrever a técnica implementada para aproximar um arco de cônica por segmentos de reta dada a representação vista no capítulo 4.

A *forma paramétrica* para cônicas no modelo plano, seção 6.3.1, foi definida durante as pesquisas para representação das cônicas. Seria possível obter boas aproximações dos arcos no plano com esta forma paramétrica. No entanto, a aproximação na esfera não poderia ser obtida simplesmente pela projeção central dos pontos da aproximação planar sobre a esfera, pois este procedimento é sujeito a distorções e a qualidade do desenho no modelo esférico ficaria comprometida. A primeira tentativa para obter uma aproximação de boa qualidade na esfera foi através do conhecimento, visto no capítulo 4, de que as cônicas na esfera são a interseção desta com cones de segundo grau com vértice no centro da esfera. As equações polinomiais destes cones são exatamente as equações polinomiais homogêneas das cônicas no plano, se considerarmos a coordenada w como sendo a coordenada z do espaço $\mathbb{E}^3$. A partir desta equação, precisaríamos encontrar uma parametrização do cone. Esta última tarefa poderia, certamente, ser feita, mas de maneira alguma seria prática.

A solução foi dada pela união da forma paramétrica do plano com uma heurística para amostragem adaptativa de funções paramétricas, seção 6.3.3. Nesta heurística, a qualidade da aproximação depende de um critério definido *a priori*. Implementamos a forma paramétrica com cálculos homogêneos e uma amostragem adaptativa com o critério calculado, ora sobre o plano (com $[x, y, w] \mapsto (x/w, y/w)$), ora sobre a esfera

(com $[x, y, w] \mapsto (x, y, w)/\sqrt{x^2 + y^2 + w^2}$). Os resultados foram visualmente bastante satisfatórios.

6.3.1 A forma paramétrica

A forma paramétrica de uma cônica no plano com foco na origem, reta focal igual ao eixo das abscissas, e diretriz interceptando a reta focal em $(-K, 0)$ é:

$$\delta(s) = \frac{Ke}{1 - e\cos s}(\cos s, \operatorname{sen} s),$$

onde e é a excentricidade da cônica e o parâmetro s é o ângulo ao redor do foco a partir da reta focal, em sentido anti-horário, veja figura 6.4(a) e [39]. A fórmula homogênea $\delta(s) = [Ke\cos s, Ke \operatorname{sen} s, 1 - e\cos s]$ possui uma característica muito apropriada. Para hipérboles, caso em que $e > 1$, o valor $(1 - e\cos s)$ é negativo justamente quando o ângulo s é menor que a direção de assintocidade da hipérbole, ou seja, a fórmula homogênea devolve os pontos do além de acordo com a definição de cônicas no $\mathbb{T}^2$, veja figura 6.4(b).

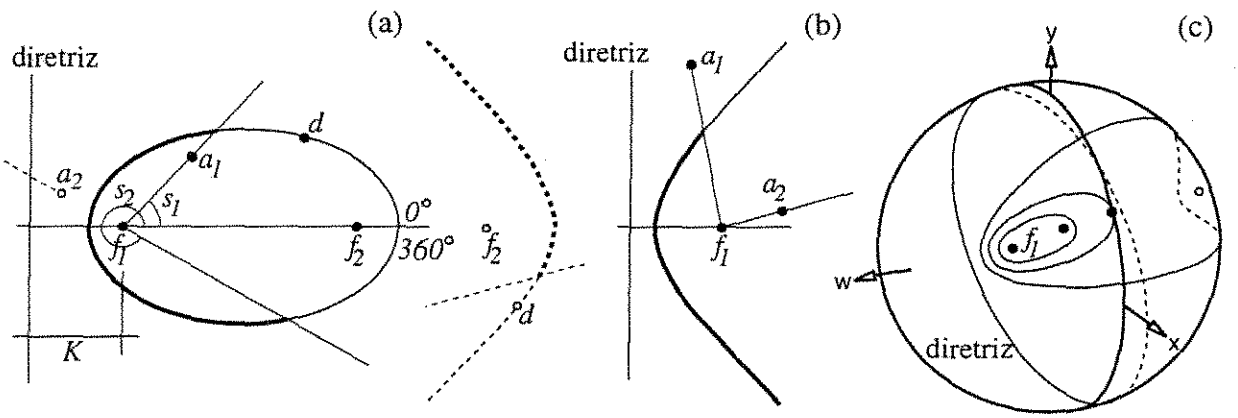

Figura 6.4: Forma paramétrica pela diretriz das cônicas

Os valores K e e desta fórmula podem ser facilmente extraídos da representação do capítulo 4. Note que, na parametrização de um arco, os pontos inicial e final são dados pelos ângulos que as retas $(f_1 \vee a_1)$ e $(f_1 \vee a_2)$ fazem com a reta focal $(f_1 \vee f_2)$.

Esta forma paramétrica pode ser usada somente na posição canônica. No $\mathbb{T}^2$, isto significa: f_1 na origem (ponto $[0, 0, 1]$), reta focal coincidente com o eixo das abscissas e diretriz passando por $[-K, 0, 1]$. Além disto, a cônica deve estar orientada em sentido anti-horário (capítulo 4), o que equivale a dizer que ela deve estar contida no lado da diretriz que também contém os focos, como os exemplos da figura 6.4(c). Descreveremos, agora, o procedimento implementado para reduzir uma cônica genérica à posição canônica no $\mathbb{T}^2$.

6.3.2 Redução de uma cônica genérica à posição canônica

Dados os cinco pontos da representação de um arco, definimos um procedimento que modifica esta representação para garantir que o foco f_1 estará no aquém e a cônica estará orientada em sentido anti-horário. Isto é feito de modo que a representação modificada sempre define o mesmo arco ou o arco antípoda. Para que o arco correto seja desenhado, após calcular um ponto sobre a curva, rotacionamos pelo ângulo entre a reta focal e o eixo das abscissas, transladamos pelo vetor de f_1 e tomamos o ponto antípoda dependendo do resultado do procedimento:

1. Se o foco f_1 estiver no além, tome o antípoda de todos os pontos e troque a_1 por a_2. Faça `flag1=true`.

2. Se a cônica estiver orientada em sentido horário, tome os antípodas de d, a_1 e a_2. Faça `flag2=true`.

3. Calcule o ponto na posição canônica. Rotacione e translade até a posição original. Se `flag1`$\oplus$`flag2=true`, onde $\oplus$ é a operação de *ou exclusivo*, tome o ponto antípoda.

Tanto o item 1 quanto o item 2 devolvem o arco antípoda. Portanto, devemos tomar o ponto antípoda somente se apenas um passo foi necessário, item 3. A figura 6.5(c) mostra um arco de hipérbole na posição canônica após o arco original da figura 6.5(a) passar pelos dois itens.

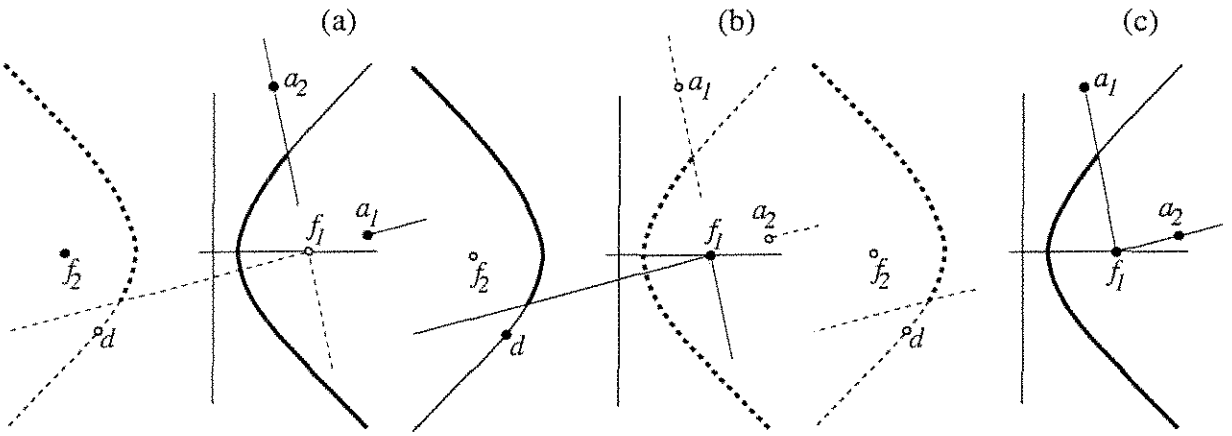

Figura 6.5: Redução a posição canônica para parametrização

6.3.3 Heurística para amostragem adaptativa da forma paramétrica

A heurística implementada é baseada na apresentada por Figueiredo [14]. Considere a figura 6.6(a). Para aproximar o arco entre a e b, amostramos o ponto c cujo parâmetro é um valor aleatório próximo à média dos parâmetros de a e b. Neste momento avaliamos o critério, que na nossa implementação é o ângulo $a\hat{c}b$ ser maior que um valor pré-fixado. Se o critério for satisfeito, a aproximação do arco entre a e b será os segmentos ac e cb. Se o critério não for satisfeito, a heurística é aplicada recursivamente aos arcos entre a e c e entre c e b. As figuras 6.6(b),(c) e (d) mostram a evolução da recursão até a aproximação final.

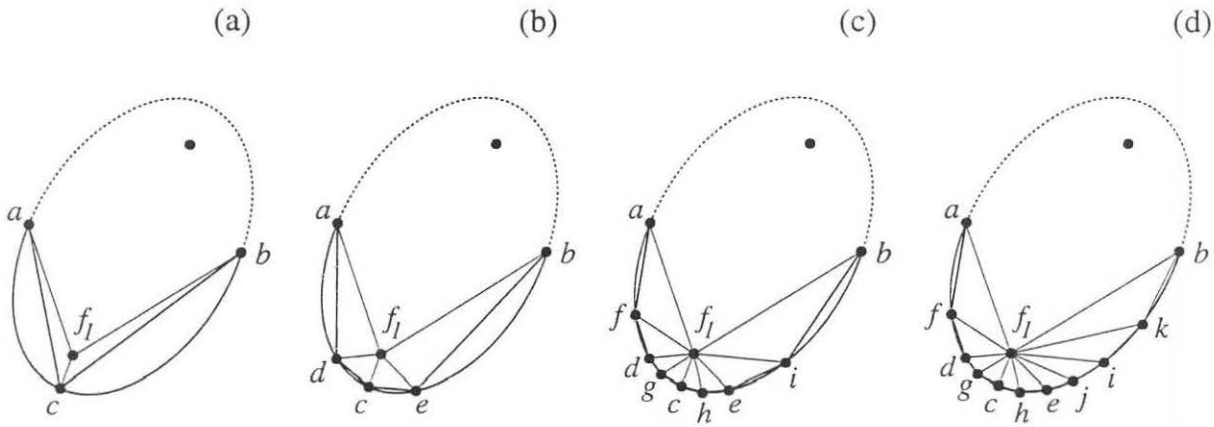

Figura 6.6: Heurística para amostragem adaptativa

Para o *flatModel*, após a parametrização de a, b e c com $\delta(s) = [Ke\cos s, Ke\,\mathrm{sen}\, s, 1 - e\cos s]$, projetamos estes pontos com $[x, y, w] \mapsto (x/w, y/w)$ e calculamos o ângulo $a\hat{c}b$ no plano. Para o *sphereModel*, projetamos os pontos com $[x, y, w] \mapsto (x, y, w)/\sqrt{x^2 + y^2 + w^2}$ e calculamos o ângulo plano $a\hat{c}b$ no espaço $\mathbb{E}^3$, veja figura 6.7(a). Os ângulos 173 e 176 graus, para a esfera e para o plano respectivamente, produziram aproximações muito boas com relativamente poucos segmentos.

Este esquema adaptativo foi também aplicado no desenho de segmentos de reta na esfera. Dados dois pontos a e b, a abordagem tradicional procuraria parametrizar o arco do grande círculo que passa por a e b em uma posição canônica, e depois, levá-lo até a posição correta com duas rotações. As figuras 6.7(b) e (c) mostram como obter diretamente o ponto médio do arco ab sobre a esfera. Se a esfera é unitária, basta tomar o vetor unitário do ponto médio do segmento ab no espaço $\mathbb{E}^3$. Este ponto é, então, usado como o ponto c na heurística.

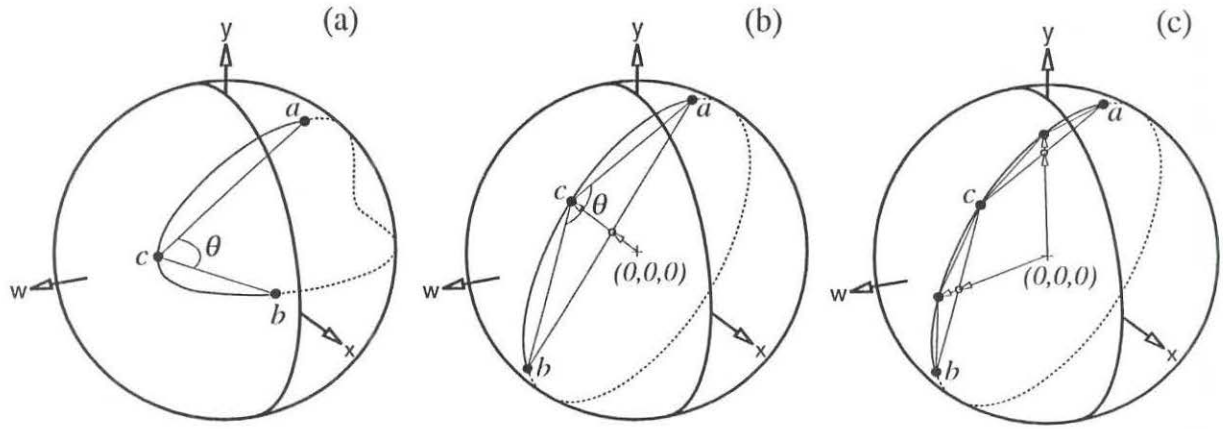

Figura 6.7: Critério da heurística no modelo esférico

6.3.4 Visualização dos Diagramas

Esta seção apresenta algumas imagens de diagramas de Voronoi construídos no $\mathbb{T}^2$ como resultado da implementação do algoritmo do capítulo 5 e dos visualizadores com suporte a cônicas:

1. A figura 6.8 apresenta um $\mathrm{DVor}_{\mathbb{T}^2}$ com sítios apenas no aquém, no modelo esférico. Neste modelo é possível visualizar nitidamente as regiões e arestas que atravessam o infinito;

2. Na figura 6.9 o mesmo diagrama do item 1 aparece no modelo plano. Note a aparente inversão provocada pela projeção central da parte do diagrama contida no além;

3. A figura 6.10 mostra um $\mathrm{DVor}_{\mathbb{T}^2}$ com sítios no aquém e no além. O modelo esférico permite visualizar os vértices de Voronoi contidos na reta do infinito;

4. A figura 6.11 apresenta um $\mathrm{DVorW}_{\mathbb{T}^2}$ no modelo esférico. Note que este diagrama seria desconexo no $\mathbb{E}^2$. O modelo esférico permite visualizar o bissetor dos sítios de menor e maior peso deste conjunto, que atravessa a reta do infinito antes de interceptar outros bissetores;

5. Na figura 6.12 o mesmo diagrama do item 4 pode ser visto no modelo plano.

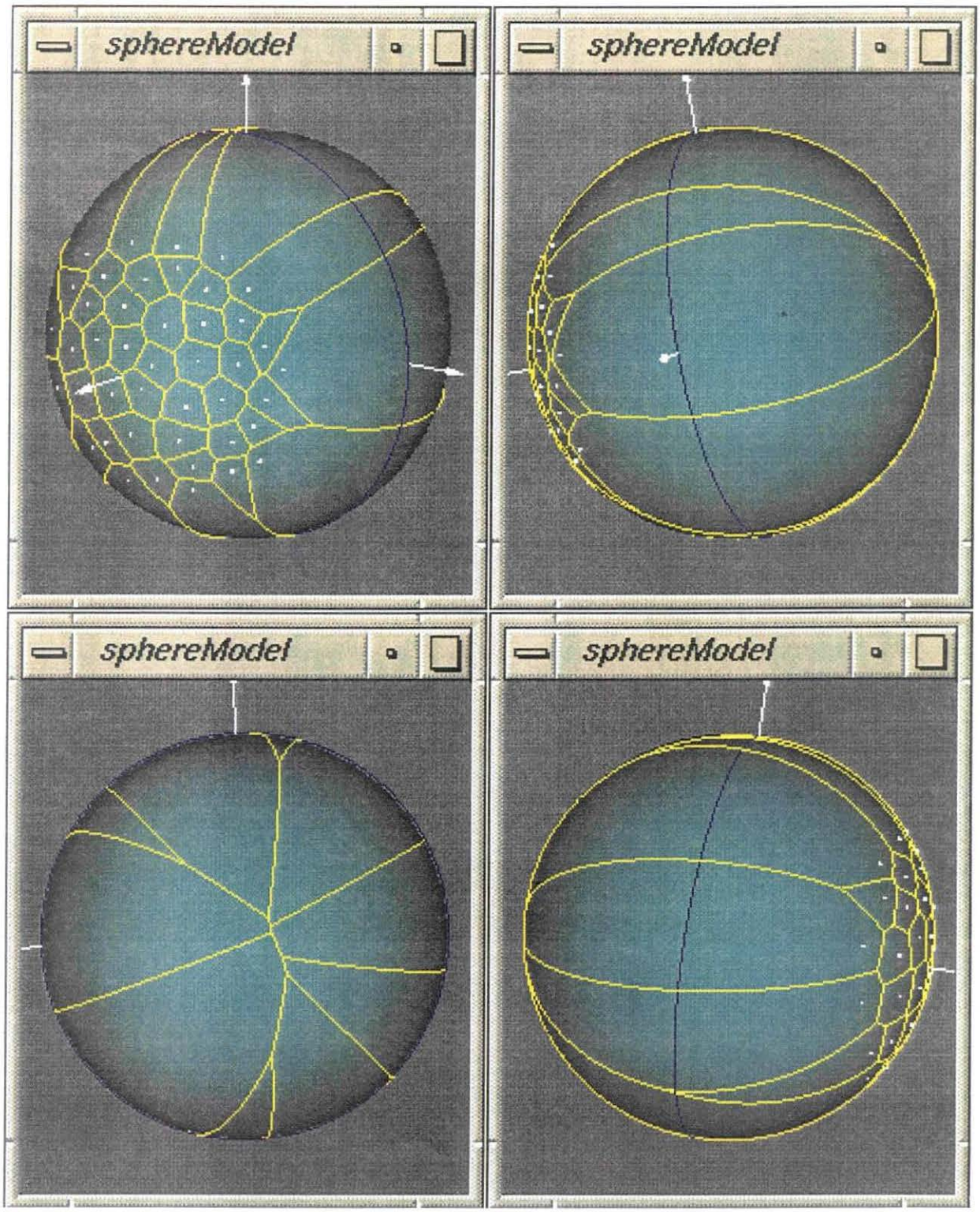

Figura 6.8: Visualização do $DVor_{\mathbb{T}^2}$ no modelo esférico

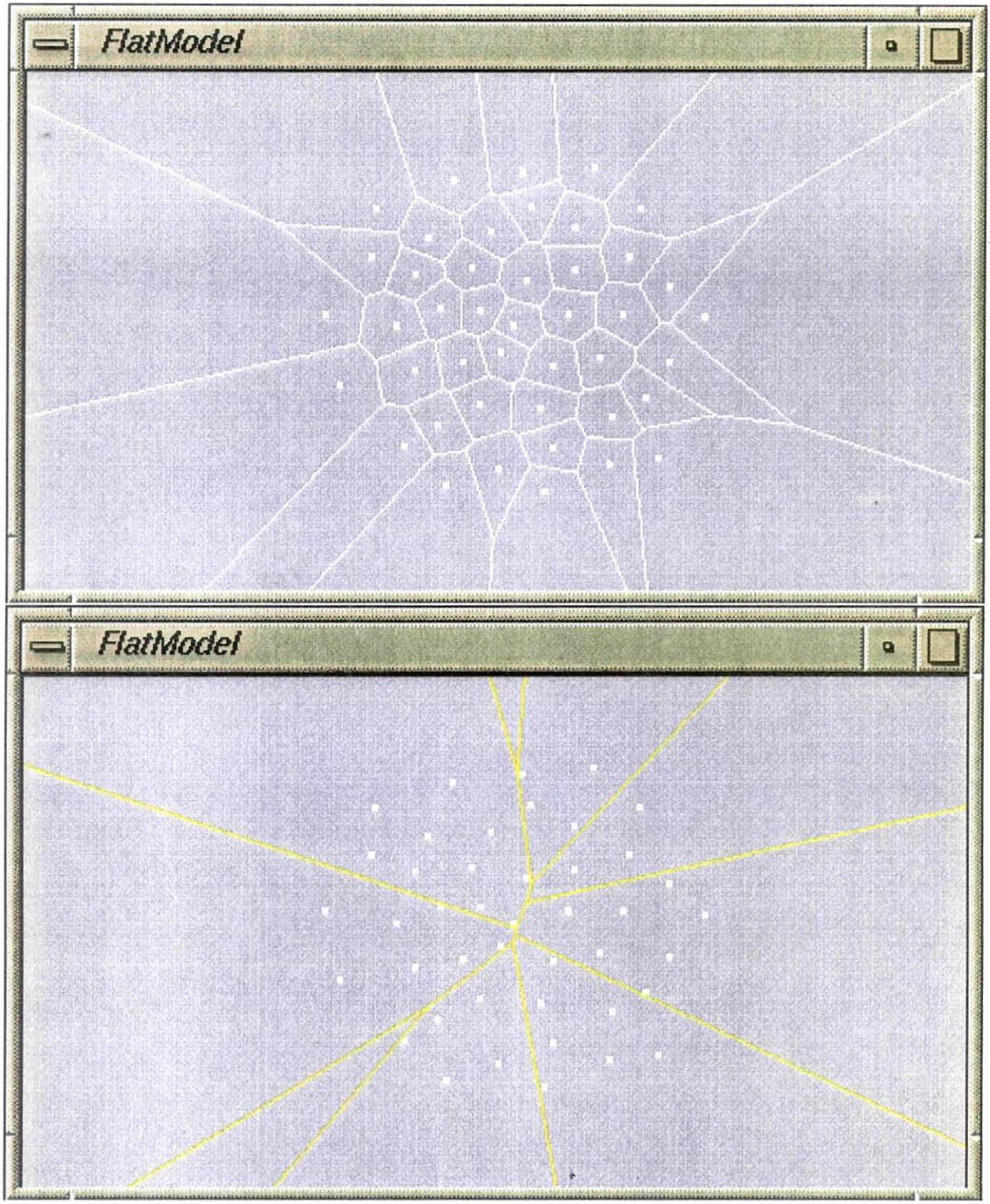

Figura 6.9: Visualização do $\mathrm{DVor}_{\mathbb{T}^2}$ no modelo plano

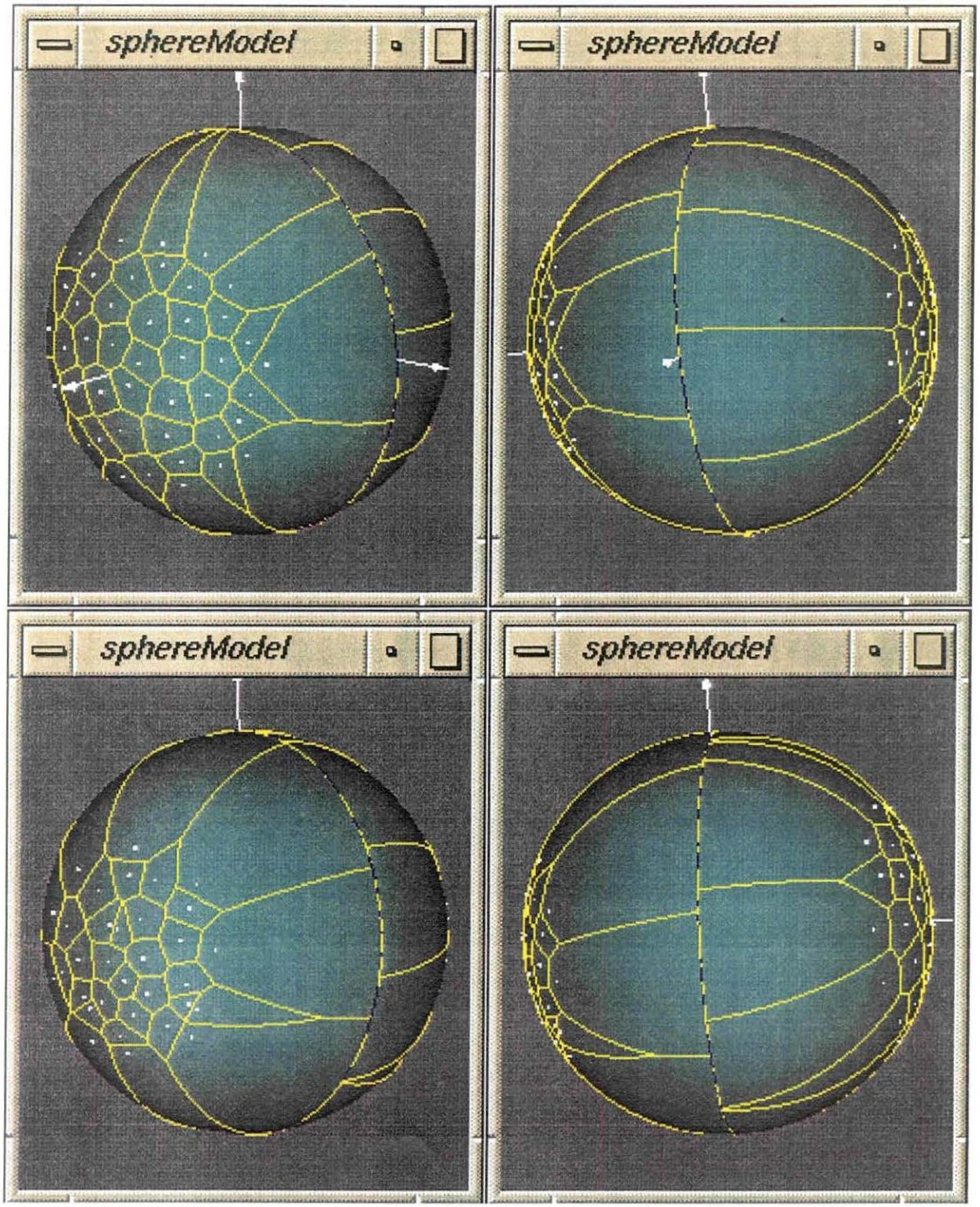

Figura 6.10: Visualização do $\mathrm{DVor}_{\mathbb{T}^2}$ no modelo esférico (sítios no aquém e além)

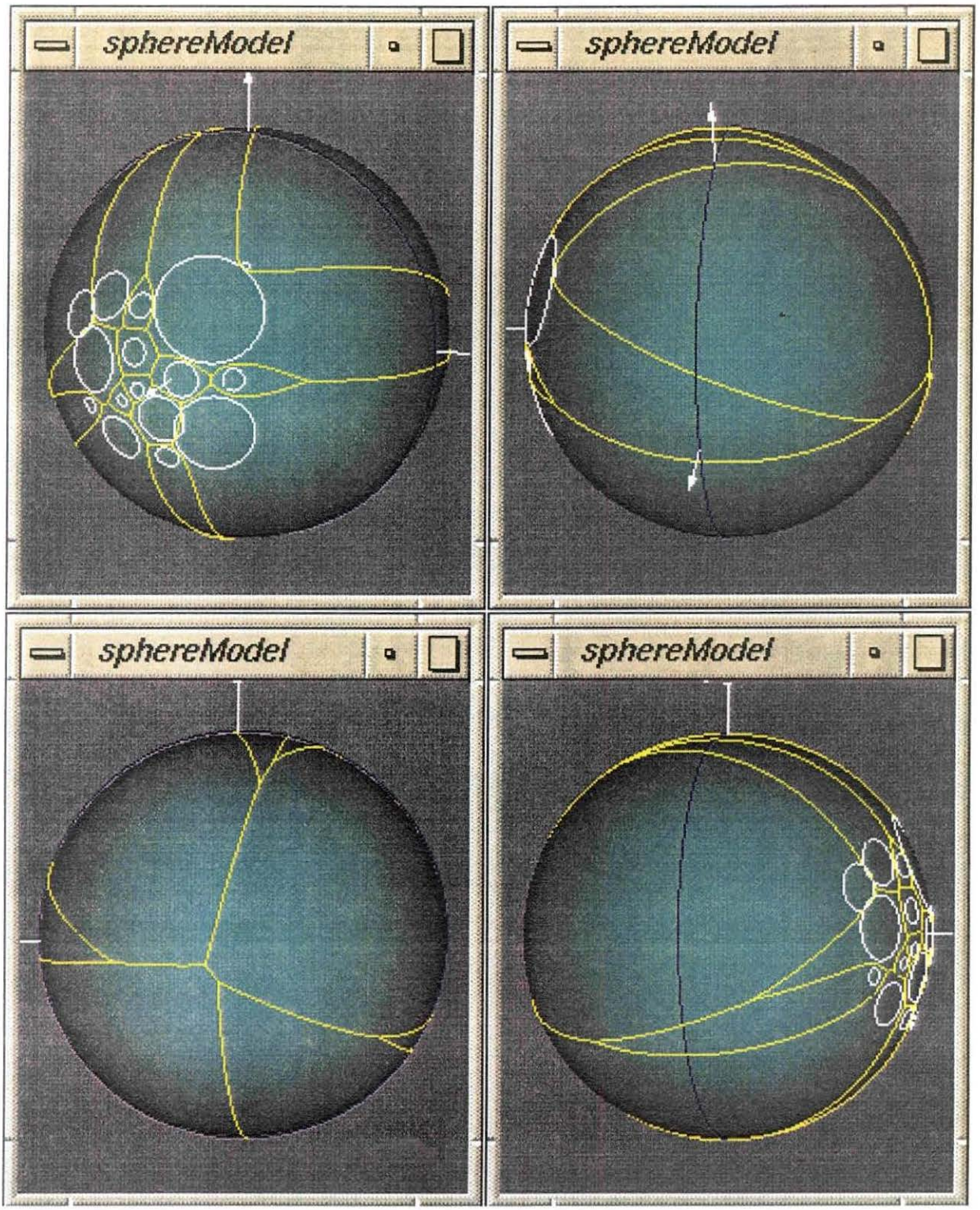

Figura 6.11: Visualização do $\mathrm{DVorW}_{\mathbb{T}^2}$ no modelo esférico

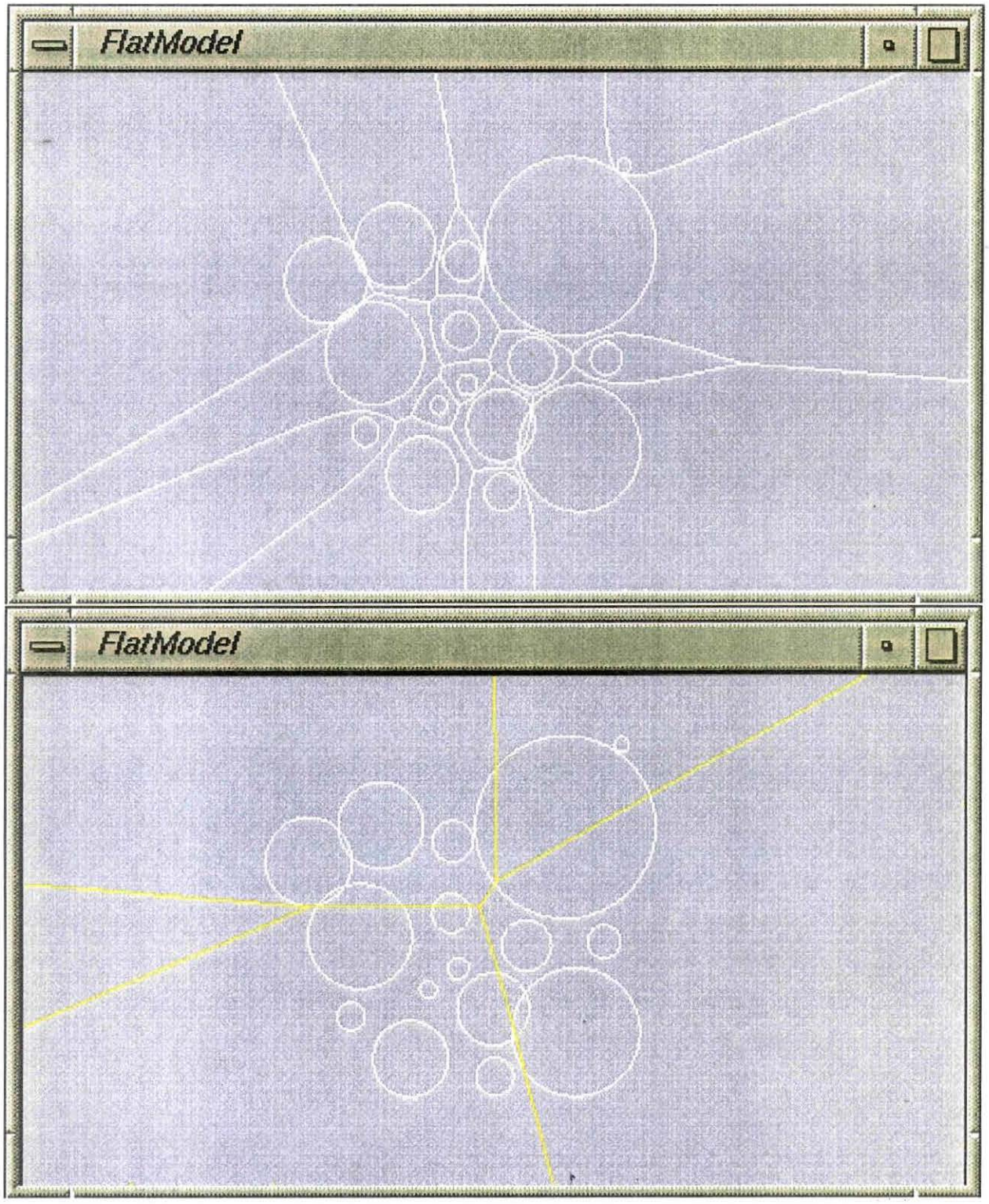

Figura 6.12: Visualização do $\mathrm{DVorW}_{\mathbf{T}^2}$ no modelo plano

Capítulo 7

Conclusão

Neste trabalho de Mestrado mostramos que o plano projetivo orientado $\mathbb{T}^2$ propicia uniformidade na representação e construção dos diagramas de Voronoi. Todas as arestas dos diagramas podem ser topologica e geometricamente representadas uniformemente, sem deixar de ter o significado de interseção de duas regiões.

Vale a pena mencionar que a distância com sinal do plano Euclidiano de dois lados (seção 3.2.1), apesar de ter um comportamento incomum, pôde ser usada consistentemente para definir as cônicas e os diagramas, se mostrando uma ferramenta útil para algoritmos geométricos.

Contribuições teóricas

1. No $\mathbb{T}^2$, o $\mathrm{DVor}_{\mathbb{T}^2}$ e o $\mathrm{DVorW}_{\mathbb{T}^2}$ possuem um número exato de arestas e vértices, $e = 3n - 6$ e $v = 2n - 4$, graças à topologia equivalente à da esfera do $\mathbb{E}^3$. Para o $\mathrm{DVorW}_{\mathbb{T}^2}$, a construção traz um benefício adicional: ele é sempre conexo, ao contrário do caso Euclidiano.

   O diagrama de vizinho mais próximo é antípoda ao de vizinho mais distante. Portanto, um algoritmo que constrói um também constrói o outro. Conseguimos obter um algoritmo incremental que mostra que a construção pode ser implementada sem que seja necessário diferenciar a parte do diagrama contida no além da contida no aquém. Os conceitos do $\mathbb{T}^2$ ficam encapsulados nas primitivas geométricas e o algoritmo segue como se estivéssemos trabalhando no $\mathbb{E}^2$. Esta abordagem foi muito bem sucedida. Obtivemos, assim, algoritmos para os diagramas de vizinho mais distante sem precisar raciocinar sobre as características geométricas ou combinatoriais desses diagramas. Para o $\mathrm{DVorW}_{\mathbb{T}^2}$, o algoritmo é, sem dúvida, bem mais simples do que os dois publicados anteriormente [34, 33] (para o diagrama de vizinho mais distante). Esta parte do trabalho está reunida no artigo [31].

2. Para descrever a geometria das arestas dos diagramas no $\mathbb{T}^2$, estudamos as cônicas e propusemos uma representação simples e eficaz que unifica as três classes de cônicas afins no plano. Esta parte do trabalho foi publicada em [30].

Contribuições práticas Era nosso objetivo inicial alcançar a visualização interativa dos diagramas construídos no $\mathbb{T}^2$. Os trabalhos práticos foram:

1. A implementação do algoritmo para o DVor e o DVorW como aplicações do ambiente GeoPrO em C++.

2. O aprimoramento do protocolo de descrição geométrica do ambiente GeoPrO para inclusão de cônicas em C++. Este trabalho envolveu a alteração de todas as interfaces e do núcleo.

3. O projeto e implementação do gerenciador do GeoPrO em Java. Esse módulo permite, entre outras funções, salvar e recuperar objetos em disco e retirar, diretamente, objetos de um determinado contexto do núcleo. Essas funções foram bastante úteis durante os testes da implementação do algoritmo e possibilitam um uso mais dinâmico do ambiente.

   A implementação do gerenciador também gerou, como subproduto, a interface para visualizadores em Java, que já está sendo usada num projeto de iniciação científica.

4. O aprimoramento do visualizador do modelo esférico *sphereModel* para desenhar cônicas, que permitiu a visualização completa dos diagramas. A classe C++ implementada foi usada também para o aprimoramento do visualizador planar, objeto de outro projeto de iniciação científica.

7.1 Trabalhos Futuros

Acreditamos que há muito ainda a investigar sobre construção de diagramas de Voronoi no $\mathbb{T}^2$. Uma direção que parece muito promissora é a construção dos diagramas de ordem k, como descrito adiante. Por outro lado, vemos a utilização do $\mathbb{T}^2$ como, certamente, uma solução de compromisso. É preciso raciocinar em uma nova topologia e realizar contas com coordenadas homogêneas com sinal, o que pode ser visto como uma desvantagem.

Algoritmos ótimos Uma questão que surge naturalmente é se seria possível adaptar algoritmos ótimos para as construções. Para o DVor, o algoritmo da "bolha ambulante" para a triangulação de Delaunay, descrito em [22], poderia ser adaptado. Os próprios autores citam que a inversão do predicado do parabolóide pode ser feita para calcular

a triangulação de Delaunay de vizinho mais distante. Para o DVorW, entretanto, os algoritmos de varredura do plano [16] e divisão e conquista [40] não parecem, à primeira vista, capazes de computar o diagrama de maneira transparente como foi obtido com o algoritmo incremental.

Diagramas de ordem k Uma generalização imediata do DVor é o diagrama de k vizinhos mais próximos. Por exemplo, o diagrama de ordem 2 divide o plano em regiões, cada uma associada a uma combinação dos sítios tomados dois a dois, tal que para todos os pontos de uma dada região os dois sítios mais próximos são a combinação associada à região. Esse diagrama pode ser definido com $H_{pq,rs} = \{x \in \mathbb{T}^2 | \mathrm{d}_{\mathbb{T}^2}(x,p) + \mathrm{d}_{\mathbb{T}^2}(x,q) \leq_{\mathbb{T}^1} \mathrm{d}_{\mathbb{T}^2}(x,r) + \mathrm{d}_{\mathbb{T}^2}(x,s)\}$. Se fizermos a mesma análise que fizemos para o $\mathrm{DVor}_{\mathbb{T}^2}$ e do $\mathrm{DVorW}_{\mathbb{T}^2}$, veremos que, genericamente, o diagrama de ordem k é antípoda ao de ordem $(n-k)$. Nesse contexto, o diagrama de vizinho mais próximo de ordem $(n-1)$ pode ser visto como o diagrama de vizinho mais distante de ordem 1 que, como mostramos, é antípoda ao de vizinho mais próximo de ordem 1.

Se os algoritmos da literatura para construção dos diagramas de ordem k puderem ser adaptados para o $\mathbb{T}^2$, para construir todos os $2n-2$ diagramas, da ordem 1 até a ordem $(n-1)$, vizinho mais próximo e vizinho mais distante, bastará construir os diagramas de vizinho mais próximo da ordem 1 até a ordem $(\lfloor n/2 \rfloor)$.

Diagrama de segmentos de reta O algoritmo implementado não pôde ser usado para implementar a construção do $\mathrm{DVorL}_{\mathbb{T}^2}$ porque uma das primitivas é o cálculo dos circuncentros, que podem não existir para três segmentos. No entanto, muitos algoritmos da literatura utilizam uma definição alternativa de diagrama estendido. Nessa definição, os extremos dos segmentos são considerados sítios distintos dos segmentos propriamente ditos. Esse diagrama estendido contém propriamente o DVorL, é sempre conexo e mais simples de representar. Seria interessante buscar uma adaptação para o $\mathbb{T}^2$.

Bibliografia

[1] P. F. Ash e E. D. Bolker. Recognizing dirichlet tessellations. *Geometriae Dedicata*, 19:175–206, 1985.

[2] P. F. Ash e E. D. Bolker. Generalized dirichlet tessellations. *Geometriae Dedicata*, 20:209–243, 1986.

[3] J. M. Augenbaum e C. S. Peskin. On the construction of the voronoi mesh on a sphere. *J. Comput. Phys.*, 59:177–192, 1985.

[4] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, Setembro 1991.

[5] F. Aurenhammer e H. Edelsbrunner. An optimal algorithm for constructing the weighted voronoi diagram in the plane. *Pattern Recognition*, 17(2):251–257, 1984.

[6] J. Bernal. Bibliographic notes on voronoi diagrams. Relatório Técnico 5164, National Institute of Standards and Technology, 1993.

[7] B. K. Bhattacharya e G. T. Toussaint. On geometric algorithms that use the furthest-point voronoi diagram. Em G. T. Toussaint, editor, *Computational Geometry*, páginas 43–61. Elsevier Science Pub., 1985.

[8] OpenGL Architecture Review Board. *OpenGL Programming Guide*. Addison-Wesley, 1993. 7ª reimpressão.

[9] C. Burkinel, K. Mehlhorn, e S. Schirra. How to compute the voronoi diagram of line segments: Theoretical and experimental results. *Lecture Notes in Computer Science*, 855:227–239, 1994.

[10] P. J. de Rezende e J. Stolfi. *Fundamentos de Geometria Computacional*. IX Escola de Computação – SBC, Recife, Brasil, 1994.

[11] H. Edelsbrunner, L. Guibas, e J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340, 1986.

[12] H. Edelsbrunner e R. Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

[13] G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, Inc, 1988.

[14] L. H. de Figueiredo. Adaptive sampling of parametric curves. Em A. Paeth, editor, *Graphics Gems V*, pages 173–178. Academic Press, 1995.

[15] CG Impact Task Force. Application challenges to computational geometry. Relatório Técnico TR-521-96, Princeton University, 1996. `http://graphics.lcs.mit.edu/~seth/pubs/taskforce/techrep.html`.

[16] S. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 3:153–174, 1987.

[17] D. Gans. *An Introduction to Non-Euclidean Geometry*. Academic Press, Inc, 1973.

[18] C. N. Gon. Computação exata em geometria projetiva orientada e tratamento de degenerações. Tese de Mestrado, Instituto de Computação – UNICAMP, Junho 1996.

[19] P. J. Green e R. Sibson. Computing dirichlet tessellations in the plane. *Computer Journal*, 21(2):168–173, 1977.

[20] ACM-NSF Computational Geometry Working Group. Strategic directions in computational geometry. A ser publicado em ACM Comput. Surveys, `http://www.cs.brown.edu/people/rt/sdcr/report/report.html`, 1996.

[21] L. Guibas. Fundamentals of geometric algorithms. Em V. Demichelis, editor, *Computational Geometry*, pages 119–174. World Scientific Press, 1993. Também em CS368: Geometric Algorithms, Stanford University.

[22] L. Guibas e J. Stolfi. Primitives for the manipulation of general subdivisions and the computations of voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, Abril 1985.

[23] M. Held. On the computational geometry of pocket machining. *Lecture Notes in Computer Science*, 500, 1991.

[24] I. Herman. The use of projective geometry in computer graphics. *Lecture Notes in Computer Science*, 564, 1991.

[25] C. M. Hoffmann. *Geometric & Solid Modeling: an introduction*. Morgan Kaufman Pub. , Inc., 1989.

[26] D. T. Lee e R. L. Drysdale III. Generalization of voronoi diagrams in the plane. *SIAM J. Comput.*, 10(1):73–86, Fevereiro 1981.

[27] K. Mehlhorn, S. Näher, e C. Uhrig. *The LEDA User Manual – Version R 3.4*, 1996. `http://www.mpi-sb.mpg.de/LEDA/www/papers.html`.

[28] R. E. Miles. Random points, sets and tessellations on the surface of a sphere. *Sankhya: The Indian Journal of Statistics, Series A*, 33:145–174, 1971.

[29] A. Okabe, B. Boots, e K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

[30] G. A. Pinto e P. J. de Rezende. Representation of conics in the oriented projective plane. Anais do *X SIBGRAPI*, páginas 71–78. IEEE, 1997.

[31] G. A. Pinto e P. J. de Rezende. Additively weighted voronoi diagram on the oriented projective plane. Em preparação, 1998.

[32] F. P. Preparata e M. Shamos. *Computational Geometry: an introduction*. Springer-Verlag, 1985.

[33] D. Rappaport. A convex hull algorithm for discs, and applications. *Computational Geometry: Theory and Applications*, 1:171–187, 1992.

[34] H. Rosenberger. Order-k voronoi diagrams of sites with additive weights in the plane. *Algorithmica*, 6:490–521, 1991.

[35] G. Salmon. *A Treatise on the Analytic Geometry of Three Dimensions*. Chelsea Publishing Company, New York, N.Y., 1914. 7ª edição.

[36] G. Salmon. *A Treatise on Conic Sections*. Chelsea Publishing Company, New York, N.Y., c.1914. 6ª edição.

[37] M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM J. Comput.*, 14(2):448–468, Maio 1985.

[38] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, Inc, 1991. 1ª edição.

[39] W. F. Taylor. *The Geometry of Computer Graphics*. Wadsworth & Brooks, 1992.

[40] C. K. Yap. An $O(n \log n)$ algorithm for the voronoi diagram of a set of simple curve segments. *Discrete & Computational Geometry*, 2:365–393, 1987.