João Carlos de Brito Dinis

Mestre em Engenharia Informática

Construção e Edição de Diagramas de Voronoi na Esfera

Dissertação para obtenção do Grau de Doutor em Informática

Orientadora: Profª Doutora Margarida Paula Neves Mamede

Júri:

Presidente:	Prof. Doutor Nuno Manuel Robalo Correia
Arguentes:	Prof. Doutor Manuel Abellanas Oar
	Profª Doutora Ana Paula Nunes Gomes Tomás
Vogais:	Doutora Ana Mafalda de Oliveira Martins
	Prof. Doutor Manuel João Toscano Próspero dos Santos
	Profª Doutora Margarida Paula Neves Mamede

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro, 2013

Construção e Edição de Diagramas de Voronoi na Esfera

Aos meus filhos,
Baltasar e Gaspar.

Agradecimentos

À Professora Doutora Margarida Mamede, pela forma calorosa com que aceitou a orientação desta tese, pela constante disponibilidade, apoio e encorajamento manifestado ao longo do todo o trabalho e pelo entusiasmo demonstrado numa área do meu interesse.

Ao Professor Doutor José Manuel Rebordão, director do Laboratório de Lasers, Óptica e Sistemas, da Faculdade de Ciências da Universidade de Lisboa, no qual desempenho funções, pela cedência de tempo e meios materiais, ingredientes fundamentais para a execução deste trabalho.

E à minha esposa Elsa e aos meus filhos Baltasar e Gaspar, pelo apoio incondicional e paciência em me acompanharem nesta maratona. Eles estavam tão ansiosos como eu para me ver chegar à meta.

Resumo

Muitos objectos de estudo em ciências da Terra e do Espaço são representados por pontos na superfície de uma esfera, identificando, por exemplo, medidas de temperatura nos oceanos ou posições de estrelas na esfera celeste. Nestes contextos, o diagrama de Voronoi esférico é a ferramenta natural para o processamento e a análise de relações de proximidade entre os pontos.

Presentemente, o diagrama de Voronoi esférico pode ser obtido por uma de duas formas: construindo o invólucro-convexo tridimensional dos pontos na superfície da esfera ou adaptando à esfera o algoritmo incremental de construção da triangulação de Delaunay no plano. Porém, os algoritmos usados na prática possuem uma complexidade temporal quadrática, no pior caso, e os que calculam o invólucro-convexo são susceptíveis de produzir um resultado errado, ao não incluir todos os pontos na estrutura construída.

Como alternativa, é proposta uma adaptação ao domínio esférico do algoritmo de construção do diagrama de Voronoi planar pelo método do varrimento. O novo algoritmo constrói um diagrama em tempo $O(n \log n)$, onde n é o número de pontos, o que é óptimo no pior caso. Adicionalmente, são propostos dois algoritmos de edição de diagramas de Voronoi esféricos, um de inserção e outro de remoção de um ponto, igualmente baseados no método do varrimento. Em conjunto, os novos algoritmos implementam uma estrutura de dados dinâmica, apropriada para cenários em que a informação é variável no tempo.

Mostra-se que, para além de eficientes, os três algoritmos são fáceis de implementar e robustos a configurações de pontos degeneradas. Estas propriedades são verificadas experimentalmente, com dados sintéticos e reais. A aplicabilidade dos algoritmos desenvolvidos é ainda exemplificada através de um problema de redução de um catálogo de estrelas.

Termos-chave: diagramas de Voronoi esféricos; edição de diagramas de Voronoi; técnica do varrimento do plano; geometria computacional; processamento de catálogos de estrelas.

Abstract

Many objects studied in Earth and Space sciences are represented as points on the surface of a sphere, locating, for instance, temperature measurements in the oceans or stars in the celestial sphere. In these contexts, the spherical Voronoi diagram is the natural tool for processing and analysing proximity relations among the points.

Presently, there are two ways of obtaining a spherical Voronoi diagram: by computing the three-dimensional convex-hull of the points on the sphere surface or by adapting to the spherical domain the incremental algorithm that builds the Delaunay triangulation in the plane. However, the algorithms used in practice have a quadratic worst-case running time and those that compute the convex-hull can produce an erroneous output, by not including all points in the result.

As an alternative, it is proposed an adaptation to the spherical domain of the plane sweep algorithm that computes the planar Voronoi diagram. The novel algorithm runs in $O(n \log n)$ time, where n is the number of points, which is optimal in the worst case. Furthermore, two algorithms are proposed for updating a spherical Voronoi diagram, one for inserting and the other for deleting a point, which are also based on the plane sweep technique. Together, the new algorithms implement a dynamic data structure, suitable for settings where the information vary along time.

It is shown that, besides being efficient, the three algorithms are easy to implement and cope very well with degenerated data sets. These properties are verified experimentally, with synthetic and real-world data. Moreover, the applicability of the developed algorithms is exemplified with a star catalogue reduction problem.

Keywords: spherical Voronoi diagram; update of Voronoi diagrams; plane sweep technique; computational geometry; processing of star catalogues.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Algoritmos

Lista de Abreviaturas, Siglas e Símbolos

CGAL
Biblioteca de Algoritmos de Geometria Computacional.

Hull
Algoritmo Hull, de Kenneth L. Clarkson.

Qhull
Algoritmo Qhull, de Bradford Barber.

STRIPACK
Algoritmo de construção da triangulação de Delaunay esférica.

UCAC4
Catálogo astrográfico do Observatório Naval Norte Americano (quarta edição).

$\langle a \rangle$
Arco da frente de onda gerado pelo local a.

$\langle a,b \rangle$
Intersecção entre os arcos da frente de onda $\langle a \rangle$ e $\langle b \rangle$.

$C_P(q)$
Círculo vazio máximo centrado em q, em relação a P.

C_{abc}
Círculo circunscrito aos pontos a, b e c.

P_{abc}
Plano coplanar aos pontos a, b e c.

$V(P)$
Diagrama de Voronoi de um conjunto de pontos P.

$V(p)$
Região de Voronoi do local p.

dist(a, b)
: Distância entre os pontos a e b.

W
: Frente de onda.

H
: Linha de varrimento (no plano) ou círculo de varrimento (na esfera).

$\mathcal{M}_{ab}$
: Mediatriz do segmento de recta (ou arco de círculo máximo) $\overline{a\,b}$.

$h(a, b)$
: Semi-plano delimitado por $\mathcal{M}_{ab}$, contendo a.

$\overline{a\,b}$
: Segmento de recta (no plano) ou
arco de círculo máximo (na esfera) que liga a a b.

$\overline{a - b}$
: Comprimento de $\overline{a\,b}$.

$a \triangleright b$
: Semi-recta (no plano) ou semi-círculo máximo (na esfera)
com origem em a e que passa por b.

$\triangle a\,b\,c$
: Triângulo definido pelos vértices a, b e c.

$\vec{a}$
: Vector que liga a origem ao ponto a.

$\|\vec{a}\|$
: Norma do vector $\vec{a}$.

$\vec{a}^{\,\perp}$
: Vector $\vec{a}$ rodado de $\pi/2$ no sentido directo.

$\overrightarrow{b - a}$
: Vector que liga a a b.

$\angle(a, b, c)$
: Ângulo interno entre $\overrightarrow{a - b}$ e $\overrightarrow{c - b}$.

Capítulo 1

Introdução

Os diagramas de Voronoi são uma importante estrutura de dados geométrica, com muitos campos de aplicação. O interesse dos diagramas de Voronoi reside na capacidade que estes têm de capturar a estrutura espacial de um conjunto de dados, estabelecendo relações entre objectos geométricos.

Os diagramas de Voronoi são simples de definir. Imagine-se um conjunto de pontos no plano. O diagrama de Voronoi do conjunto de pontos é uma sub-divisão do plano em polígonos convexos, em que cada polígono delimita a região do plano mais próxima de cada ponto [2, 3, 20, 47]. A Figura 1.1 ilustra um exemplo.

Apesar da inerente simplicidade da sua definição, os diagramas de Voronoi são estruturas ricas em informação. Em primeira análise, observa-se que os diagramas de Voronoi caracterizam, sob a forma de um polígono, a região de influência de cada ponto. A forma e a extensão de cada polígono é uma medida da distribuição dos pontos em redor do ponto respectivo e das distâncias relativas entre eles. Por outro lado, o polígono de cada ponto também caracteriza as relações topológicas ente pontos. Num diagrama, dois pontos dizem-se vizinhos se os polígonos respectivos partilham uma aresta. Desta forma, os vizinhos de um ponto formam um sub-conjunto particular: são os pontos que lhe estão mais próximos qualquer que seja a direcção tomada em seu redor.

As relações de vizinhança podem ser agregadas em formas mais complexas, estendendo as relações anteriores a todo o plano. Por exemplo, pode definir-se o grau de vizinhança topológica entre quaisquer dois pontos pelo número mínimo de regiões que é necessário atravessar para ir de um ponto ao outro ponto. Opcionalmente, pode também definir-se o grau de vizinhança geométrica entre dois pontos quaisquer pelo número de arestas intersectadas pelo segmento de recta que une os dois pontos. Em resumo, um diagrama de Voronoi sintetiza as relações de proximidade entre quaisquer dois pontos.

Por seu lado, as arestas de um diagrama providenciam um ponto de vista alternativo. Representam as posições do plano mais afastadas dos pontos. Em particular, os vértices (extremos das arestas) representam as posições do plano localmente mais afastadas de pontos.

As relações de vizinhança descritas, sejam de proximidade ou de afastamento, são apenas exemplos do tipo de informação que se pode extrair dos diagramas. A riqueza dos diagramas

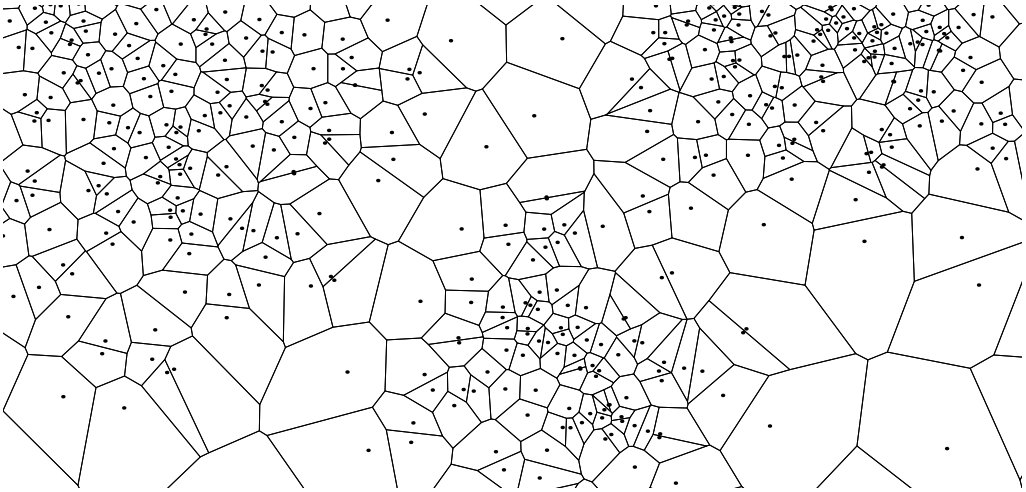


Figura 1.1: Diagrama de Voronoi de um conjunto de pontos no plano.

de Voronoi reside na facilidade com que se elaboram relações de proximidade sobre objectos geométricos. Daí a sua aplicação em muitos domínios. Apontam-se alguns exemplos.

Os polígonos de cada local indicam o número de vizinhos próximos (pelo número de arestas) e a proximidade relativa (pela área da região). Esta informação foi aplicada a problemas de *clustering*, detecção de colisões, interpolação de dados, simplificação de modelos digitais de terrenos, etc.

As arestas e os vértices de um diagrama de Voronoi estão equidistantes a dois ou mais pontos, indicando as posições do plano menos ocupadas. Esta propriedade tem sido aplicada à resolução de problemas de planeamento de percursos em robótica, posicionamento de postos de visualização, determinação de bacias hidrográficas, entre outros.

1.1 Diagramas de Voronoi esféricos

Inicialmente, os diagramas de Voronoi foram definidos para pontos num plano, com a medida de proximidade dada pela distância Euclidiana. Mais tarde, foram generalizados para objectos com diferentes formas, de diferentes espaços e com diferentes métricas [2]. Uma generalização possível, denominada de *diagrama de Voronoi esférico*, obtém-se considerando pontos na superfície da esfera, sendo a medida de proximidade dada pelo comprimento do arco geodésico que liga duas posições [47]. O resultado é um diagrama semelhante ao diagrama planar, formando uma subdivisão da esfera em polígonos esféricos, cujas arestas são arcos de círculo máximo (ver Figura 1.2).

Os diagramas de Voronoi esféricos têm aplicação em domínios onde a informação é inerentemente esférica, tais como as ciências da Terra e do Espaço. São várias as ciências que se dedicam ao estudo da Terra, como, por exemplo, a cartografia, a geodesia, a geologia, a geofísica, a oceanografia e a meteorologia. Esta lista, representativa das ciências da Terra, tem por ponto comum gerar e processar informação que abarca toda a esfera em que vivemos. Exemplos de informação recolhida pelas ciências da Terra são medidas da temperatura

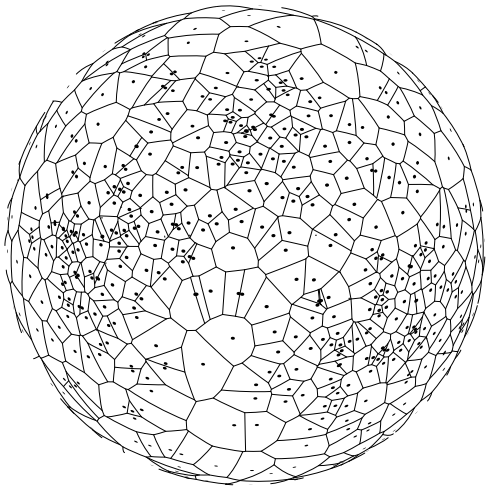


Figura 1.2: Diagrama de Voronoi de um conjunto de pontos na esfera.

do ar, posições de marcos geodésicos, localizações de recursos minerais, medidas do campo gravítico ou medidas de salinidade da água do mar. No domínio do Espaço, as estrelas são representadas, na forma mais simples, como pontos numa esfera imaginária. Também aqui é natural o recurso ao diagrama de Voronoi esférico como ferramenta de análise.

Ao longo do tempo foram desenvolvidos vários algoritmos para a construção de diagramas de Voronoi. Há, no entanto, um algoritmo que se destaca pela simplicidade e elegância: o algoritmo de Fortune [34]. Trata-se de um algoritmo que se aplica à construção de diagramas de Voronoi no plano, para objectos formados por pontos ou por segmentos de recta. Na forma original, este algoritmo aplica-se apenas a diagramas gerados segundo a norma Euclidiana. Mais tarde, o algoritmo foi generalizado para uma gama mais alargada de métricas, tais como L_1 e L_∞ [22].

Neste algoritmo, a construção do diagrama faz-se pela técnica do varrimento do plano que, resumidamente, pode ser descrita da seguinte forma. O plano é varrido por uma linha recta que se desloca segundo uma direcção arbitrária; por exemplo, varrendo o plano com uma linha horizontal, que se desloca de cima para baixo, segundo a vertical. Durante o varrimento, observa-se que apenas uma parte do diagrama (em construção) é intersectado pela linha de varrimento. A estratégia é construir, em cada instante, apenas a parte do diagrama que está relativamente próxima da linha de varrimento. A construção do diagrama faz-se pela ordem induzida pelo varrimento.

A eficiência do algoritmo deriva da forma como a linha de varrimento restringe a complexidade do problema inicial. Numa solução de força-bruta, por exemplo, seria necessário determinar cada aresta individualmente, comparando cada local com todos os outros. No algoritmo de varrimento, cada elemento do diagrama é determinado comparando apenas, entre si, os elementos do diagrama que estão próximos da linha de varrimento.

Para além da eficiência, o algoritmo de Fortune possui duas vantagens adicionais: é de fácil implementação e resolve naturalmente configurações degeneradas. Como se verá mais adiante, a implementação depende apenas de primitivas geométricas de grau reduzido (cál-

culo do centro e do raio de um círculo circunscrito a três pontos e posição de um ponto em relação à recta definida por dois pontos) e de estruturas de dados relativamente simples (árvore binária ordenada e fila com prioridade), contribuindo para a simplicidade da sua implementação. Configurações degeneradas ocorrem quando, na execução de um algoritmo, uma configuração de um conjunto de pontos despoleta uma excepção que pode conduzir a um resultado errado ou, na pior das hipóteses, a uma terminação abrupta da sua execução. Casos típicos são configurações em que três ou mais pontos são colineares ou quatro ou mais pontos são co-circulares. Habitualmente, estes obstáculos são evitados restringindo a aplicação de um algoritmo a conjuntos de dados em "posição geral", isto é, sem exibirem configurações degeneradas. Outras soluções passam por estender o caso geral de um algoritmo com rotinas de processamento adicional, específicas para resolver estes casos. O algoritmo de Fortune, como é característica da técnica do varrimento, resolve configurações degeneradas de uma forma elegante, sem despoletar excepções nem necessitar de processamento adicional.

Actualmente, o diagrama de Voronoi esférico é construído por uma de duas maneiras: indirectamente, por construção de um invólucro-convexo [9, 4, 18, 48] ou, directamente, por adaptação do algoritmo incremental de construção da triangulação de Delaunay no plano [38, 40, 20].

Na primeira opção, o diagrama de Voronoi esférico é obtido pela equivalência entre este e o invólucro-convexo tridimensional dos pontos na superfície da esfera. A construção do invólucro-convexo pode ser feita em tempo $O(n \log n)$, no pior caso, onde n é o número de pontos [49]. Mas, na prática, são preferidos algoritmos mais lentos mas mais simples de implementar, executando em tempo $O(n \log n)$, no caso esperado, e em tempo $O(n^2)$, no pior caso [48]. Esta solução comporta uma desvantagem. Porque os pontos se localizam na superfície da esfera, todos os pontos devem pertencer ao invólucro-convexo. Porém, imprecisões numéricas podem posicionar, erradamente, um ponto no interior do invólucro-convexo, produzindo um resultado errado. Esta particularidade impõe uma exigência acrescida na implementação desta solução.

Na segunda opção, a construção do diagrama de Voronoi esférico é obtida pela equivalência entre um diagrama e o seu grafo dual, a triangulação de Delaunay, adaptando o algoritmo de construção da triangulação no plano à esfera. A construção incremental da triangulação de Delaunay faz-se em duas partes: primeiro é procurado o triângulo que contém o ponto a inserir e depois este é inserido, actualizando a triangulação. Este algoritmo constrói uma triangulação em tempo $O(n \log n)$, no caso esperado, e em tempo $O(n^2)$, no pior caso. O custo temporal depende da distribuição espacial dos pontos e da ordem pela qual são inseridos. O custo mais favorável assume que a procura se faz em tempo $O(\log n)$, o que apenas é garantido para pontos distribuídos uniformemente.

Em contraste, o algoritmo de Fortune executa em tempo $O(n \log n)$, no pior caso. Como se verá mais adiante, o desempenho deste algoritmo não depende de nenhuma distribuição espacial de pontos.

O desafio que se propõe nesta dissertação é a adaptação do algoritmo de Fortune à cons-

trução do diagrama de Voronoi esférico. A adaptação não se revelou trivial. O equivalente esférico de uma linha recta (no plano) é um círculo máximo. Logo, pode imaginar-se uma adaptação da técnica do varrimento em que a superfície da esfera é varrida por um círculo máximo rodando em torno de um eixo. Porém, tal varrimento não produz o mesmo efeito que o varrimento linear no plano, onde há uma clara separação entre a parte do domínio que já foi varrida e a parte que está por varrer. Em particular, num varrimento esférico em torno de um eixo, os pólos do eixo estariam continuamente a ser varridos. A solução encontrada passa por considerar uma forma alternativa do algoritmo de Fortune, em que a construção do diagrama de Voronoi no plano se faz pelo varrimento do plano por um círculo de raio crescente [22]. Nesta formulação, o diagrama é construído adicionando elementos sucessivamente mais distantes do centro do círculo de varrimento. Tal como no varrimento linear, também no varrimento circular há uma clara separação entre a parte do domínio que já foi varrida (a parte interior ao círculo) e a parte que está por varrer (a parte exterior ao círculo). De resto, as duas abordagens são muito semelhantes, possuindo as mesmas características: a construção do diagrama por varrimento circular é eficiente, de fácil implementação e robusta a configurações degeneradas.

A formulação com varrimento circular torna exequível a adaptação do algoritmo de Fortune à superfície da esfera. Qualquer que seja o ponto escolhido para centro do varrimento, um círculo da raio crescente varre a superfície da esfera de uma forma inequívoca, visitando cada lugar uma única vez. O círculo de varrimento começa num ponto, aumentando até atingir a forma de um círculo máximo, após o qual reduz de tamanho até convergir de novo para um outro ponto (antípoda do primeiro).

A técnica do varrimento circular revelou-se uma ferramenta interessante em si mesma. Ao varrer o espaço em torno de um ponto, o varrimento circular está a executar uma pesquisa ordenada do espaço em torno do centro de varrimento. De facto, uma vantagem do varrimento circular é a de possibilitar a construção parcial de um diagrama de Voronoi; por exemplo, determinando apenas o polígono de Voronoi de um ponto. Esta capacidade intrínseca do varrimento circular levou à concepção de dois algoritmos de edição de diagramas de Voronoi: um para a inserção de um novo ponto e outro para a remoção de um ponto.

A edição de um diagrama de Voronoi é tão fundamental como a sua construção. É razoável assumir que o conjunto de pontos para o qual se quer construir um diagrama nem sempre é conhecido, à partida, na sua totalidade. Por exemplo, pode imaginar-se a geração de um conjunto de pontos inscrito a um quadrado, segundo uma distribuição de Poisson, em que os pontos são determinados iterativamente, com o próximo ponto posicionado no lugar mais afastado de todos os outros. Opcionalmente, também se poderia obter um resultado semelhante gerando primeiro um conjunto de pontos em posições aleatórias, do qual são removidos pontos que estejam demasiado próximos de algum ponto vizinho. Em qualquer dos casos, é vantajoso que a construção do diagrama de Voronoi se faça de forma incremental, evitando a reconstrução completa do diagrama em cada iteração de qualquer dos processos atrás descritos. Em suma, os algoritmos de edição enriquecem e ampliam a utilidade dos

diagramas de Voronoi.

Os algoritmos de edição operam de forma muito semelhante à do algoritmo de Fortune, com um varrimento circular em torno do ponto a inserir ou a remover. Obtiveram-se dois algoritmos computacionalmente eficientes e de fácil implementação. Como se baseiam na técnica de varrimento circular, os novos algoritmos de edição são aplicáveis a diagramas de Voronoi no plano e na esfera.

1.2 Redução de um catálogo de estrelas

Com o propósito de exemplificar a aplicabilidade dos algoritmos de construção e de edição de diagramas de Voronoi esféricos, foi desenvolvido um algoritmo de redução de catálogos de estrelas. A redução de um catálogo consiste na remoção selectiva de estrelas de um catálogo com o objectivo de reduzir o espaço ocupado em memória, mas retendo a sua capacidade descritiva. Esta necessidade surge no contexto de uma solução de navegação autónoma de naves espaciais, onde a orientação de uma nave é determinada mapeando o padrão de estrelas observadas por um câmara num catálogo de referência. Devido à escassez de recursos computacionais, habitual em plataformas espaciais, é imperativo reduzir o espaço ocupado pelo catálogo de estrelas. Daí a necessidade de se remover estrelas selectivamente, garantindo, no entanto, que o catálogo reduzido constitui um sub-conjunto representativo do catálogo de referência.

O algoritmo de redução proposto constitui uma oportunidade para aplicar os algoritmos de construção e de edição de diagramas de Voronoi esféricos. Na base do algoritmo está a construção e edição (sob forma de remoção de pontos) de diagramas de Voronoi esféricos, representando cada ponto uma estrela. A qualidade de um catálogo reduzido depende do número de estrelas presentes em recortes circulares, centrados em posições arbitrárias, tal como é observado por uma câmara. Logo, para se avaliar a qualidade de um catálogo, houve necessidade de desenvolver um algoritmo de recorte de catálogos de estrelas, sabendo-se que um catálogo de estrelas é habitualmente formado por um conjunto numeroso de pontos na superfície da esfera. Este recorte foi implementado com recurso a uma estrutura de dados denominada de *hierarquia de Voronoi* que, como o nome indica, é baseada em diagramas de Voronoi. Os algoritmos de redução e de recorte de um catálogo foram implementados e testados experimentalmente.

1.3 Contribuições originais

Os principais resultados desta tese são o desenvolvimento e análise de algoritmos de construção e edição de diagramas de Voronoi esféricos. Foram também obtidos resultados complementares, que derivam da análise dos algoritmos desenvolvidos e da sua aplicação a um caso prático. Nomeadamente, são contribuições originais deste trabalho:

- O desenvolvimento de um novo algoritmo de construção do diagrama de Voronoi de pontos na superfície da esfera. A eficiência e a facilidade de implementação justificam que este algoritmo seja o melhor para a construção do diagrama de Voronoi esférico;
- O desenvolvimento de dois novos algoritmos de edição de diagramas de Voronoi de pontos no plano ou na esfera, um de inserção de um ponto num diagrama e outro de remoção de um ponto de um diagrama. Em conjunto, estes algoritmos tornam a implementação do diagrama de Voronoi numa estrutura de dados dinâmica;
- A adaptação do método do varrimento do plano ao varrimento da superfície da esfera. Este resultado, corolário do desenvolvimento do algoritmo de construção do diagrama de Voronoi esférico, potencia a adaptação de outros algoritmos à esfera, desde que baseados na estratégia do varrimento do plano;
- O desenvolvimento de um algoritmo de redução de um catálogo de estrelas. O algoritmo de redução processa um catálogo de estrelas, equalizando a distribuição espacial das estrela. É uma exemplificação da aplicabilidade dos novos algoritmos;
- A construção de uma estrutura de dados de indexação espacial de pontos na superfície da esfera, especializada para a selecção de pontos em recortes circulares. Faz parte do desenvolvimento da solução de redução de catálogos, exemplificando também a aplicabilidade dos novos algoritmos.

Duas das contribuições enunciadas foram apresentadas em conferências internacionais. Nomeadamente, o novo algoritmo de construção do diagrama de Voronoi esférico (descrito no capítulo 5) está publicado em [29]:

> Dinis, J., and Mamede, M. Sweeping the sphere., In *Proceedings of the 2010 International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'10)* (June 2010), IEEE, pp. 151–160.

e os novos algoritmos de edição do diagrama de Voronoi esférico (descritos no capítulo 6) estão publicados em [30]:

> Dinis, J., and Mamede, M., Updates on Voronoi diagrams. In *Proceedings of the 2011 International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'11)* (June 2011), IEEE, pp. 192–199.

1.4 Estrutura do documento

Este documento divide-se em duas partes. A primeira parte expõe o novo algoritmo de construção de diagramas de Voronoi esféricos e os dois novos algoritmos de edição de diagramas de Voronoi, analisando-os em detalhe. A segunda parte é dedicada ao processamento de catálogos de estrelas, com o objectivo de exemplificar a utilidade dos algoritmos desenvolvidos.

A **parte I** está organizada do seguinte modo.

O **capítulo 2** define os diagramas de Voronoi, no plano e na esfera, enumerando as propriedades mais relevantes. Neste capítulo são também descritos os algoritmos mais usuais de construção e edição de diagramas planares e esféricos.

O **capítulo 3** apresenta as triangulações de Delaunay, estruturas geométricas intimamente ligadas aos diagramas de Voronoi. Define as suas propriedades e descreve os algoritmos mais usuais para a sua construção e edição. Mais à frente, estes algoritmos são comparados com os algoritmos equivalentes desenvolvidos neste trabalho.

No **capítulo 4** é apresentado um resumo dos algoritmos de construção do diagrama de Voronoi planar pelo método do varrimento. Primeiro, na secção 4.1, é revisto o varrimento linear, que serve de base ao algoritmo de Fortune, apresentado em detalhe na secção 4.2. Na secção 4.3 é descrita uma forma alternativa deste algoritmo, baseada no varrimento circular. É uma adaptação do algoritmo de Dehne e Klein [21], aqui descrito com base na figura da frente de onda.

O **capítulo 5** descreve o algoritmo de varrimento esférico em detalhe, fazendo um amplo uso da matéria do capítulo 4. A secção 5.1 estuda o varrimento da esfera por elipses esféricas, necessárias para definir a frente de onda, que é a base do algoritmo de construção do diagrama de Voronoi esférico, detalhado em pormenor na secção 5.2. A secção 5.3 relaciona o novo algoritmo com o diagrama de Voronoi planar obtido por projecção estereográfica, tal como estabelecido por Brown [9].

O **capítulo 6** apresenta e analisa dois novos algoritmo de edição de diagramas de Voronoi. Na secção 6.1 são estudados os algoritmos de edição dos diagramas de Voronoi esféricos, enquanto que os algoritmos de edição no plano são analisados na secção 6.2. A secção 6.3 apresenta uma forma alternativa do algoritmo de inserção, específica para a implementação eficiente de interpolação por vizinhos naturais.

O **capítulo 7**, por fim, apresenta resultados experimentais que comparam os novos algoritmos com algoritmos apresentados nos capítulos 2, 3 e 4.

A **parte II** consiste no desenvolvimento de um algoritmo de processamento de catálogos de estrelas, denominado de *redução de catálogo*.

O **capítulo 8** define a redução de um catálogo e descreve o problema que a motiva. O algoritmo de redução é descrito na secção 8.1, enquanto que o método de aferição de resultados é definido na secção 8.2.

O **capítulo 9** apresenta a estratégia utilizada para a implementação da medição do desempenho de um catálogo, baseada na operação de recorte num conjunto de pontos. Na secção 9.1 é introduzida uma estrutura de dados para pesquisas de localização, denominada de hierarquia de Voronoi. Na secção 9.2 é detalhada a operação de recorte num catálogo de estrelas.

O **capítulo 10** apresenta e analisa em detalhe os resultados do teste do algoritmo de redução de catálogos de estrelas.

A dissertação finaliza com possíveis seguimentos ao trabalho iniciado com o desenvolvimento do método de varrimento na esfera, apontando algumas linhas de trabalho futuro.

Parte I

Diagramas de Voronoi

Capítulo 2

Diagrama de Voronoi

Um diagrama de Voronoi é uma decomposição de um espaço métrico, definido por uma função distância e por um conjunto de locais. Neste contexto, um local é um objecto geométrico compacto que pode assumir variadas formas. Exemplos de um local no plano são um ponto, um segmento de recta, um círculo, um polígono fechado, etc. Estes exemplos encontram correspondência na esfera, desde que se faça uma adaptação geométrica apropriada, como é o caso da troca de segmento de recta por segmento de arco de círculo máximo.

A escolha da forma dos locais e de uma função distância define um tipo de diagrama de Voronoi [2, 47]. Este trabalho incide sobre dois tipos de diagramas, ambos com locais pontuais: o diagrama de Voronoi planar, definido pela distância Euclidiana, e o diagrama de Voronoi esférico, definido pela distância geodésica na esfera. Nas secções que se seguem, são apresentadas as principais propriedades destes dois tipos de diagramas.

2.1 Diagrama no plano

O diagrama de Voronoi planar definido por um conjunto de pontos no plano e pela distância Euclidiana é o exemplo mais comum e mais simples de diagrama. Constitui uma subdivisão do plano num conjunto de regiões poligonais, em que cada região está associada a um ponto. Mais exactamente, cada região é formada pelo lugar dos pontos do plano que estão mais próximos do local que a gera do que de qualquer outro local. A Figura 2.1 ilustra um exemplo.

Formalmente, o diagrama de Voronoi planar é definido do seguinte modo.

Definição 1 (Diagrama de Voronoi)**.** *Seja $P = \{p_1, p_2, \ldots, p_n\}$ (com $n \geq 2$) um conjunto de pontos em $\mathbb{R}^2$, denominados por locais, e seja $dist(x, y)$ a distância Euclidiana entre dois pontos x e y. A região de Voronoi $V(p_i)$ de um local $p_i \in P$ é dada por: $V(p_i) = \{x \in \mathbb{R}^2 \mid dist(x, p_i) \leq dist(x, p_j), j = 1, \ldots, n\}$. O diagrama de Voronoi de P forma uma subdivisão planar constituída por n regiões, uma por cada local de P. Isto é, $V(P)$, é dado por: $V(P) = \bigcup_{i=1}^{n} V(p_i)$.*

As regiões de um diagrama cobrem todo o plano, existindo regiões de área finita e outras de área infinita. Os locais que pertencem ao invólucro-convexo de P têm regiões de Voronoi

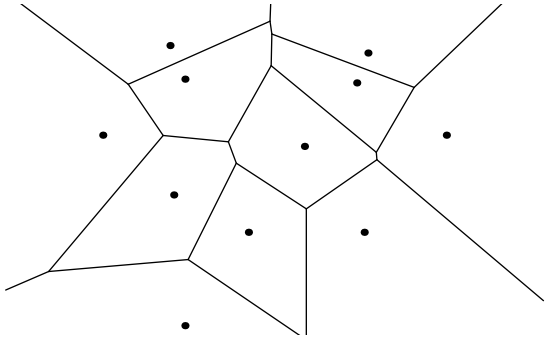


Figura 2.1: Diagrama de Voronoi planar.

de área infinita, delimitadas por linhas poligonais. Por oposição, as regiões dos locais que *não* pertencem ao invólucro-convexo são delimitadas por polígonos.

Uma forma alternativa de caracterizar uma região de Voronoi é através de um processo de intersecção de semi-planos. Sejam p_i e p_j dois locais de P e $h(p_i, p_j)$ o semi-plano definido pela mediatriz do segmento de recta que une os dois locais e que contém p_i. Desta forma, a região $\mathrm{V}(p_i) = \bigcap h(p_i, p_j) : p_j \in P \setminus \{p_i\}$. Por resultar da intersecção de $n-1$ semi-planos, a fronteira de uma região de Voronoi tem no máximo $n-1$ arestas.

Topologicamente, um diagrama é formado por um conjunto de locais, arestas e vértices. Uma aresta pertence sempre a duas regiões, separando as regiões de dois locais vizinhos. As arestas podem ser segmentos de recta, semi-rectas ou mesmo rectas. As regiões dos locais não pertencentes ao invólucro-convexo são delimitadas por segmentos de recta. As regiões geradas pelos locais pertencentes ao invólucro-convexo são, regra geral, delimitadas por duas semi-rectas e, eventualmente, por um ou mais segmentos de recta. No caso particular em que todos os locais estão colineares, cada região é delimitada apenas por uma recta ou por um par de rectas (paralelas entre si). Um vértice é partilhado por três ou mais arestas. Por construção, um ponto de uma aresta é equidistante a dois locais, enquanto que um vértice é equidistante a três ou mais locais.

O número exacto de arestas de cada região de Voronoi depende da forma como os pontos estão distribuídos. No máximo, uma região de Voronoi tem $n-1$ arestas. Em média, o número de arestas de cada região é igual a seis, independentemente do número de locais [3]. Mais exactamente, para $n \geq 3$, o número de vértices n_v e o número de arestas n_e de um diagrama de Voronoi verificam as seguintes propriedades [50, 20]:

$$\begin{aligned} n_v &\leq 2\,n - 5, \\ n_e &\leq 3\,n - 6. \end{aligned} \tag{2.1}$$

Estes resultados decorrem da aplicação da fórmula de Euler, aplicável por intermédio de um homeomorfismo entre uma sub-divisão planar e um poliedro convexo. Segundo a fórmula de Euler, para qualquer poliedro convexo tridimensional com n_v vértices, n_e arestas e n_f faces tem-se:

$$n_v - n_e + n_f = 2. \tag{2.2}$$

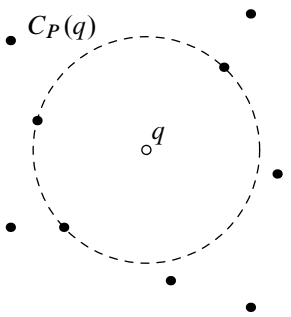


Figura 2.2: Círculo vazio máximo de um ponto arbitrário em relação a um conjunto de pontos P.

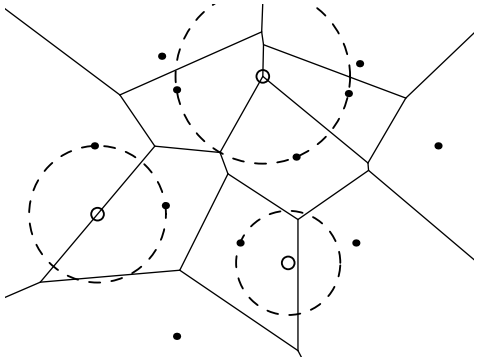


Figura 2.3: Círculos vazios máximos num diagrama de Voronoi. Pontos em três casos: interior a uma região, sobre uma aresta e coincidente com um vértice.

Para que se possa estabelecer um homeomorfismo, é necessário ligar de alguma forma as arestas de comprimento infinito. Uma forma expedita é adicionando um vértice auxiliar v_∞ no "infinito", ao qual todas as arestas infinitas são ligadas. Assim, sabendo que cada local gera uma face, obtém-se a relação:

$$(n_v + 1) - n_e + n = 2. \tag{2.3}$$

Desta forma, cada aresta liga dois vértices e cada vértice pertence a três ou mais arestas. Ou seja:

$$2\,n_e \geq 3(n_v + 1). \tag{2.4}$$

Pelas expressões (2.3) e (2.4), obtêm-se as propriedades indicadas em (2.1). Observe-se que cada aresta é partilhada por duas regiões, o que justifica o número médio de seis arestas por região.

O resultado da expressão (2.1) indica que o número de arestas é linear em n, apesar de o número de mediatrizes possíveis entre n locais ser quadrático. Como é óbvio, nem todas as mediatrizes possíveis geram uma aresta no diagrama de Voronoi. Porém, este resultado assegura que o número de arestas e o número de vértices de um diagrama de Voronoi são lineares no número de locais.

Um modo interessante de caracterizar as arestas e os vértices de um diagrama de Voronoi é por intermédio de círculos vazios máximos.

Definição 2 (Círculo vazio máximo)**.** *Dado um ponto $q \notin P$ qualquer, o* círculo vazio máximo *de q em relação ao conjunto P, designado por $C_P(q)$, é dado pelo maior círculo centrado em q que não contém nenhum ponto de P no seu interior (ver Figura 2.2).*

Por definição, um círculo vazio máximo contém pelo menos um ponto de P na sua fronteira. Consequentemente, no diagrama $\mathrm{V}(P)$ verificam-se as seguintes propriedades:

1. Um ponto q pertence ao interior de V(p_i) se, e só se, $C_P(q)$ apenas contém o local p_i na sua fronteira.

2. Um ponto q pertence a uma aresta de $V(P)$ se, e só se, $C_P(q)$ contém exactamente dois locais na sua fronteira.

3. $V(p_i)$ e $V(p_j)$ partilham uma aresta se existe um ponto q tal que $C_P(q)$ apenas contém p_i e p_j na sua fronteira.

4. Um ponto q é um vértice de V(P) se, e só se, $C_P(q)$ contém três ou mais locais na sua fronteira.

O número de arestas incidentes num vértice, que é igual ao número de locais na fronteira do círculo vazio máximo respectivo, designa-se por *grau* de um vértice. Vértices com grau superior a três designam-se por *vértices múltiplos*.

Estas propriedades estão ilustradas na Figura 2.3. A propriedade dos círculos vazios máximos permite identificar as arestas e os vértices do diagrama de Voronoi e é central na construção do diagrama por varrimento.

2.2 Diagrama na esfera

O segundo tipo de diagrama de Voronoi abordado neste trabalho é o diagrama formado por pontos na superfície da esfera e pela distância geodésica, designado por *diagrama de Voronoi esférico* e ilustrado na Figura 2.4.

Em termos gerais, as diferenças entre um diagrama na esfera e um diagrama no plano são consequência da diferente função distância. Na esfera, a distância entre dois pontos é dada pelo comprimento do arco geodésico que os une [47], que é parte de um círculo máximo na esfera. A mediatriz do arco geodésico que une dois pontos é igualmente um círculo máximo. Deste modo, o diagrama de Voronoi esférico é uma subdivisão da superfície da esfera em regiões, delimitadas por segmentos de arcos de círculo máximo.

As propriedades do diagrama de Voronoi esférico são idênticas às do diagrama planar, com excepção de alguns detalhes intrínsecos ao domínio esférico. Como indicado atrás, a mediatriz do arco geodésico que une dois locais é um círculo máximo que divide o domínio em dois hemisférios, em que cada um representa o lugar dos pontos mais próximos de cada local. Agora, todas as regiões têm área finita e todas as arestas são finitas, sendo cada região delimitada por polígonos esféricos. Os vértices mantêm a mesma propriedade: são equidistantes a três ou mais locais.

Os diagramas de Voronoi esféricos são igualmente caracterizados pelos círculos vazios máximos. O número de locais tangentes a um círculo vazio máximo indica se o centro do círculo pertence ao interior de uma região, a uma aresta ou a um vértice, tal como no caso planar. Mas neste caso, um círculo também é definido pela intersecção de um plano com a esfera. Em particular, a intersecção da esfera pelo plano definido por três locais na superfície

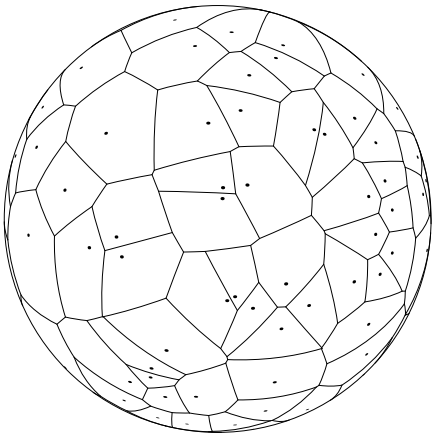


Figura 2.4: Diagrama de Voronoi esférico.

da esfera (quando considerados pontos em $\mathbb{R}^3$) define um círculo que corresponde ao círculo vazio máximo circunscrito aos três locais, em que o centro deste último é um vértice do diagrama. No caso de um vértice de grau superior a três, todos os locais são necessariamente coplanares.

O número de elementos de um diagrama de Voronoi esférico difere ligeiramente do equivalente planar. Agora o domínio é fechado, não sendo necessário recorrer ao artifício de considerar um vértice no infinito. Neste caso, o diagrama de Voronoi é homeomorfo a um poliedro convexo. Portanto, pela fórmula de Euler, tem-se que:

$$n_v - n_e + n = 2\,, \tag{2.5}$$

em que n_v, n_e e n são o número de vértices, o número de arestas e o número de locais, respectivamente. Seguindo o mesmo argumento que no plano, tem-se que cada vértice tem três ou mais arestas incidentes e cada aresta é partilhada por dois vértices. Ou seja:

$$2\,n_e \geq 3\,n_v\,. \tag{2.6}$$

Juntando as duas relações, conclui-se que num diagrama de Voronoi esférico o número de vértices n_v e o número de arestas n_e verificam as relações:

$$\begin{aligned} n_v &\leq 2n - 4, \\ n_e &\leq 3n - 6. \end{aligned} \tag{2.7}$$

As igualdades verificam-se quando todos os vértices têm grau três [47].

2.3 Implementação de diagramas

Os diagramas de Voronoi são, habitualmente, implementados por uma de duas estruturas de dados: a *quad-edge* [39] ou a DCEL [20].

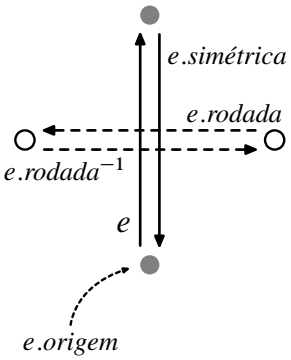


Figura 2.5: Ilustração de uma *quad-edge*, estrutura que agrupa quatro arestas orientadas: e, $e.simétrica$, $e.rodada$ e $e.rodada^{-1}$.

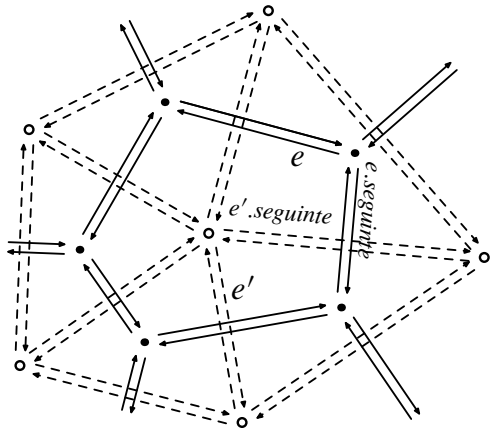


Figura 2.6: Representação de um diagrama de Voronoi por uma estrutura *quad-edge*.

A quad-edge é uma estrutura de dados que implementa simultaneamente uma sub-divisão e o seu dual que, no caso presente, é o diagrama de Voronoi e a triangulação de Delaunay. O nome deriva da forma como as arestas são representadas. Num diagrama de Voronoi, uma aresta é incidente a dois vértices (extremos da aresta) e a duas faces (geradas por dois locais). Uma quad-edge é uma estrutura que agrupa quatro arestas orientadas correspondendo a uma aresta do diagrama (que liga dois vértices) e à aresta dual (que liga dois locais na triangulação). A Figura 2.5 ilustra uma quad-edge. Para simplificar o discurso, designe-se uma aresta orientada apenas por aresta. Uma quad-edge é representada por um vector de quatro arestas, permitindo o acesso directo entre uma aresta e as restantes arestas do grupo. Cada aresta contém ainda um ponteiro para uma aresta *seguinte* e um ponteiro para o elemento origem da aresta (que pode ser um vértice ou um local do diagrama de Voronoi). A aresta seguinte de uma aresta é aquela que se encontra numa rotação segundo o sentido directo em torno da sua origem. A Figura 2.6 exemplifica a representação de um diagrama de Voronoi por uma quad-edge, assinalando também as arestas seguintes de duas arestas, e e e', com origem num vértice e num local, respectivamente. Os ponteiros $e.seguinte$ implementam uma lista simplesmente ligada das arestas em torno de uma origem. Desta forma, a vizinhança em torno de um local é obtida por um percurso nesta lista. Se e' é uma aresta incidente a um local, a origem de $e'.simétrica$ identifica um local vizinho, enquanto que a origem de $e'.rodada$ (ou $e'.rodada^{-1}$) identifica um vértice da fronteira da região respectiva. Dada a simetria da quad-edge, os locais e vértices adjacentes a um vértice são obtidos de um modo semelhante.

A simetria intrínseca da quad-edge faz desta uma estrutura de dados simples e flexível, características desejáveis na implementação de um algoritmo. Porém, por nenhuma razão em particular, a escolha de estrutura de dados para implementar diagramas de Voronoi, para os

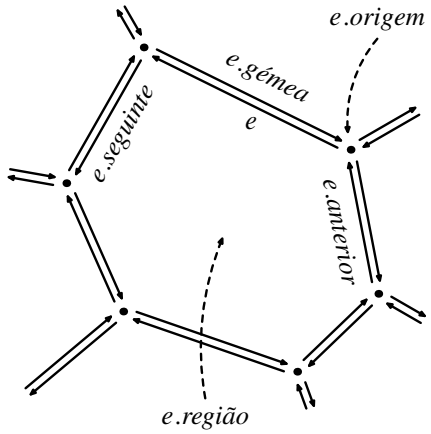


Figura 2.7: Ilustração de uma meia-aresta numa DCEL.

algoritmos desenvolvidos neste trabalho, recaiu sobre a DCEL. Esta é uma estrutura de dados igualmente simples e flexível, específica para a implementação de subdivisões em superfícies bidimensionais, embora sem possuir a simetria que caracteriza a quad-edge. Note-se, no entanto, que nenhum dos algoritmos depende de características únicas da DCEL, podendo todos ser facilmente adaptados à quad-edge.

A DCEL, contracção de *lista duplamente ligada de arestas*[1], é uma estrutura de dados capaz de representar um grafo planar, seja no plano ou numa superfície em 3D (como a esfera). Tal como o nome indica, é uma estrutura desenhada em torno das arestas de um diagrama. Cada região é descrita por uma lista circular duplamente ligada de arestas. Como uma aresta pertence a duas regiões, uma aresta do diagrama é repartida por duas instâncias de meias-arestas, gémeas uma da outra, pertencendo a listas de arestas distintas.

A DCEL está ilustrada nas Figuras 2.7 e 2.8. Um diagrama é representado por um vector de meias-arestas E, conjuntamente com dois vectores adicionais: um vector de índices R que associa a cada região uma meia-aresta (arbitrária) da lista correspondente e um vector de índices T que associa a cada vértice uma das meias-arestas que o têm por origem. As coordenadas são guardadas em dois vectores: um para as coordenadas dos locais P e outro, opcional, para as coordenadas dos vértices V.

Cada célula de uma lista, que abusivamente se denomina por meia-aresta (ilustrada na figura 2.7), contém índices (em E) para o elemento seguinte e para o elemento anterior, um índice (em E) para a aresta gémea (que liga a região à região vizinha), um índice (em R e P) que identifica o local da região a que pertence e um índice (em T e V) que identifica o vértice origem.

Uma operação de consulta numa DCEL traduz-se num percurso pelas suas meias-arestas. Destacam-se duas operações. A fronteira de uma região i é obtida percorrendo a lista circular que tem início em $e_0 = R[i]$. As arestas e_k desta lista permitem conhecer, por exemplo, as coordenadas dos vértices respectivos ($V[e_k$.origem]) ou os índices dos locais vizinhos (e_k.gémea.região). Um passeio num diagrama, com início numa aresta arbitrária e, é obtido

[1]Do acrónimo em Inglês *doubly connected edge list*.

Estrutura **DCEL**:

vector: E	{meias-arestas}
vector: R	{aresta incidente à região i}
vector: T	{aresta incidente ao vértice i}
vector: P	{coordenadas dos locais}
vector: V	{coordenadas dos vértices}

Estrutura **meia-aresta**:

inteiro: seguinte	{índice em E}
inteiro: anterior	{índice em E}
inteiro: gémea	{índice em E}
inteiro: região	{índice em R e P}
inteiro: origem	{índice em T e V}

Com os seguintes invariantes:

$$\begin{cases} R[i].\text{região} = i \\ T[j].\text{origem} = j \end{cases}$$

Figura 2.8: Representação de um diagrama por listas duplamente ligadas de arestas (DCEL).

transitando entre regiões (via e.gémea) seguido de um percurso nas meias-arestas da nova lista circular, à procura da região seguinte.

A triangulação de Delaunay é obtida de forma implícita por intermédio do vector T. O triângulo centrado num vértice k é obtido percorrendo as três arestas que têm k como origem (começando por $e = T[k]$), registando os locais incidentes às meias-arestas.

As regiões de área limitada são naturalmente descritas por uma lista duplamente ligada circular. Para as regiões ilimitadas, existentes apenas nos diagramas planares, há que resolver a interrupção entre as arestas definidas por semi-rectas. Na secção 2.1, este obstáculo foi resolvido com a introdução de um vértice auxiliar v_∞, localizado no infinito, ao qual todas as arestas infinitas ligam. No entanto, esta solução tem a desvantagem de introduzir, no caso geral, um vértice múltiplo. Como se verá mais à frente, o algoritmo de Fortune produz diagramas contendo apenas vértices de grau três. A solução passa por adicionar um local fictício p_∞, posicionado no infinito, e fechar cada região ilimitada com a adição de uma aresta que separa o local gerador respectivo do local p_∞. Desta forma, cada região ilimitada é aumentada com a adição de uma aresta e dois vértices (auxiliares), tendo o local p_∞ por vizinho. Este esquema tem a vantagem de normalizar as transformações a operar num diagrama. Em particular, a operação de rotação de uma aresta (descrita abaixo) é válida também para uma aresta auxiliar, simplificando a edição de diagramas de Voronoi planares.

Os algoritmos de edição (descritos no Capítulo 6) transformam um diagrama, representado por uma DCEL, aplicando apenas três operações básicas: a rotação de uma aresta e a inserção e remoção de um par de arestas.

A rotação de uma aresta está ilustrada na Figura 2.9, onde cada aresta é representada pelo par de arestas gémeas. Quando a aresta e é rodada, duas arestas adjacentes (b e d) são religadas ao extremo oposto de e, enquanto que as outras duas arestas adjacentes (a e c) se mantêm inalteradas. A rotação de uma aresta faz com que e seja expulsa das regiões a que pertence (c.f. Figura 2.9 da esquerda para a direita). Por outro lado, a rotação de uma aresta incidente a uma região absorve essa aresta para a fronteira da região (c.f. Figura 2.9 da direita para a esquerda).

A inserção de um par de arestas é utilizada para criar uma região, no início da operação de inserção de um novo local. Dada uma aresta e, a inserção de um par de arestas em e consiste

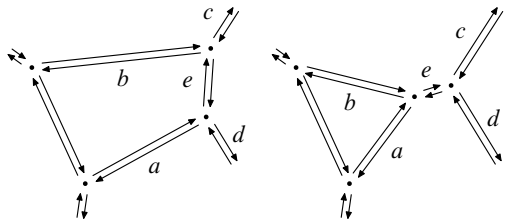


Figura 2.9: Rotação de uma aresta e numa DCEL.

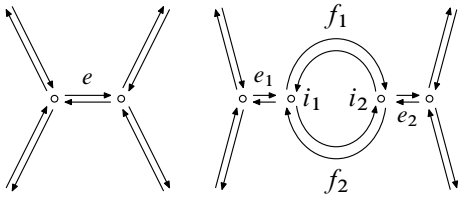


Figura 2.10: Inserção/remoção de um par de arestas numa DCEL.

em trocar e por quatro arestas e dois vértices, tal como ilustrado na Figura 2.10 (da esquerda para a direita). A nova região consiste em apenas dois vértices (i_1 e i_2) e duas arestas (f_1 e f_2).

A operação inversa, a remoção de um par de arestas, é utilizada para eliminar uma região, na operação de remoção de um local. A remoção de um par de arestas consiste em substituir as duas arestas que formam a região e as duas arestas incidentes, e_1 e e_2, por uma única aresta e (ilustrado na Figura 2.10, da direita para a esquerda).

As operações descritas transformam um diagrama por via de alterações topológicas (com novas ligações entre arestas) e alterações geométricas (com vértices em novas posições). No entanto, durante uma operação de edição, apenas são consideradas as alterações topológicas. O cálculo das posições dos vértices modificados (que é opcional) só é feito após finalizada a sequência de alterações topológicas.

2.4 Algoritmos de construção e de inserção

Os diagramas de Voronoi e as triangulações de Delaunay são estruturas de tal forma interligadas que um algoritmo de construção de uma das estruturas também constrói, implicitamente, a outra. Nesta secção são apresentados algoritmos de construção de diagramas de Voronoi planares e esféricos. Os algoritmos de construção da triangulação de Delaunay são apresentados mais à frente, na secção 3.2.

O primeiro algoritmo de construção do diagrama de Voronoi planar com complexidade temporal O(n log n) no pior caso foi desenvolvido por Shamos e Hoey [54], sendo baseado na estratégia de divisão e conquista. Dado um conjunto S de n locais no plano, este é separado em dois sub-conjuntos S_e e S_d, com um número de elementos aproximadamente igual, cujos diagramas $V(S_e)$ e $V(S_d)$ são construídos recursivamente e depois unidos para formar o diagrama $V(S)$. A sub-divisão consiste em dividir o conjunto de locais pela linha vertical que passa pela mediana de S, segundo as abcissas. A união dos dois diagramas $V(S_e)$ e $V(S_d)$ faz-se determinando a sequência de mediatrizes que separam regiões de locais de S_e de regiões de locais de S_d. Esta sequência forma uma linha poligonal monótona em y e é determinada iterativamente, como ilustra a Figura 2.11. O ponto de partida é dado

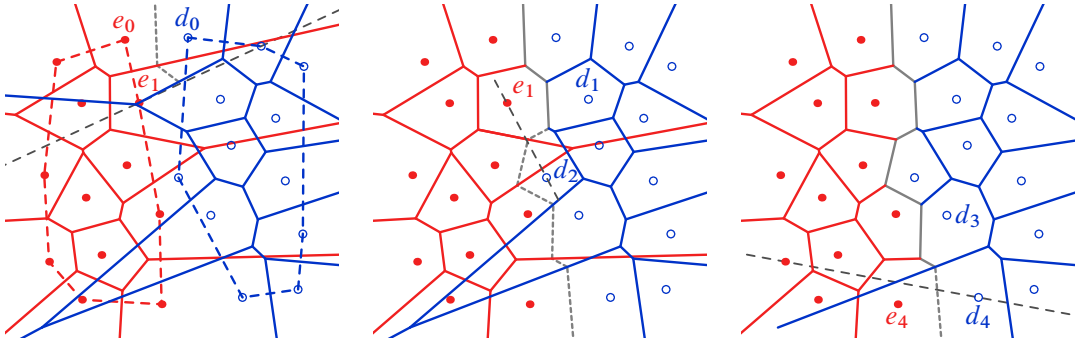


Figura 2.11: Construção do diagrama de Voronoi pelo método de divisão e conquista.

por uma das duas arestas que pertencem ao invólucro-convexo de S e com extremos em cada um dos sub-conjuntos. Conhecidos os invólucros convexos de S_e e S_d, basta adicionar duas arestas para se obter o invólucro-convexo de S, sendo estas facilmente determinadas em tempo linear. Destas duas arestas, seja $\overline{e_0\, d_0}$ a aresta com o extremo de maior ordenada, com $e_0 \in S_e$ e $d_0 \in S_d$. A mediatriz de $\overline{e_0\, d_0}$, quando percorrida de cima para baixo, intersecta arestas dos diagramas de cada sub-conjunto. Se a mediatriz de $\overline{e_0\, d_0}$ intersecta primeiro uma aresta de $V(S_e)$ (como ilustrado na Figura 2.11), então é adicionado um vértice na posição da intersecção, é adicionada uma aresta sobre a mediatriz de $\overline{e_0\, d_0}$ e é iniciada uma nova mediatriz, agora do segmento $\overline{e_1\, d_0}$, sendo e_1 o vizinho de e_0 com quem partilha a aresta intersectada. No caso de ser uma aresta de $V(S_d)$ a ser primeiro intersectada pela mediatriz de $\overline{e_0\, d_0}$, é executado um procedimento semelhante, sendo ao invés iniciada a mediatriz do segmento $\overline{e_0\, d_1}$. A nova mediatriz começa a traçar um segmento de aresta de $V(S)$ até que intersecta uma aresta de $V(S_e)$ ou uma aresta de $V(S_d)$. Continuando, é adicionado um novo vértice, é determinada uma nova mediatriz e o processo é iterado até se completar o traçado da linha poligonal. A última aresta determinada coincide com a mediatriz da segunda aresta que une os invólucros convexos de S_e e S_d, atrás referidos (aresta $\overline{e4\, d4}$ na Figura 2.11). A linha poligonal é determinada em tempo linear no número de pontos das duas sub-divisões. Logo, a construção do diagrama de Voronoi pelo método de divisão e conquista faz-se em tempo $O(n \log n)$, o que é óptimo. Apesar da complexidade óptima, este algoritmo é reportado como sendo de difícil implementação [48, 39], relevante apenas do ponto de vista histórico. Por outro lado, a eficiência deste algoritmo é penalizada pela repetida adição e remoção de algumas arestas na junção recursiva dos diagramas.

O algoritmo de Green e Sibson [38] constrói o diagrama de Voronoi planar, sendo baseado na estratégia de construção incremental. Conceptualmente, é muito simples. A construção faz-se iterativamente, adicionando um local de cada vez. Imagine-se que se tem um diagrama $V(S)$ de n locais. A adição de um novo local s começa pela localização do local s_0 que lhe está mais próximo, com o qual partilha necessariamente uma aresta (ver Figura 2.12). A segunda aresta é determinada prolongando a mediatriz do segmento $\overline{s\, s_0}$, que intersecta, pelo menos, uma das arestas de $V(s_0)$. O ponto de intersecção com a aresta identifica um vértice de $V(S \cup \{s\})$, enquanto que o local s_1, com o qual s_0 partilha a aresta intersectada,

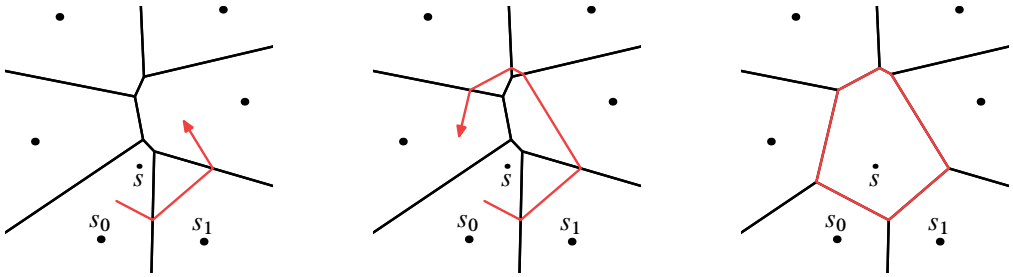


Figura 2.12: Construção do diagrama de Voronoi pelo método incremental.

permite identificar uma nova mediatriz, agora entre s e s_1. O resto da fronteira da nova região é determinado iterando este procedimento, até se regressar ao ponto de partida ou não se registar nenhuma intersecção da mediatriz corrente com qualquer aresta de $V(S)$ (que se verifica no caso em que $V(s)$ é uma região não-finita). A adição do local s termina com a remoção da parte de $V(S)$ interior à nova região.

Os autores indicam duas formas de identificar o local mais próximo de s, ponto de partida para a inserção. Na forma mais simples, é feito um passeio no diagrama corrente, percorrendo regiões sucessivamente mais próximas do novo local, com custo $O(\sqrt{n})$, o que conduz a um algoritmo com complexidade temporal de $O(n^{3/2})$. Como forma de acelerar o processo, é também sugerido que a pesquisa se faça passeando em gerações sucessivas de diagramas, guardando uma cópia do diagrama corrente após cada inserção. Potencialmente, um passeio num diagrama mais "antigo" aproxima-se mais rapidamente do alvo, servindo o resultado intermédio como ponto de partida para o passeio num diagrama mais "recente". É sugerido que tal algoritmo possa ter uma complexidade de $O(n \log n)$ no tempo, à custa de um maior consumo de memória. Tal como o algoritmo por divisão e conquista, o algoritmo incremental tem a desvantagem de produzir múltiplas versões da região de cada local, implicando a adição e posterior remoção de arestas que não fazem parte do diagrama final.

O algoritmo de Fortune será apresentado na secção 4.2.

A construção do diagrama de Voronoi pelo método incremental é facilmente adaptável ao domínio esférico, notando apenas que segmentos de recta no plano correspondem, na esfera, a arcos de círculo máximo. Também é possível aplicar o método incremental à construção da triangulação de Delaunay, o que é descrito na secção 3.2.

O primeiro algoritmo de construção de diagramas de Voronoi esféricos foi desenvolvido por Brown [9] e faz uso da ligação entre o invólucro-convexo e a triangulação de Delaunay: o invólucro-convexo de pontos na superfície da esfera é topologicamente equivalente à triangulação de Delaunay esférica. Mais concretamente, uma face do invólucro-convexo corresponde a um triângulo da triangulação de Delaunay e, consequentemente, a um vértice do diagrama de Voronoi. Logo, o diagrama de Voronoi esférico pode ser obtido calculando o invólucro-convexo de pontos em 3D. Existem vários algoritmos para este fim. Brown [9] sugere o algoritmo de divisão e conquista de Preparata e Hong [49, 50], com uma comple-

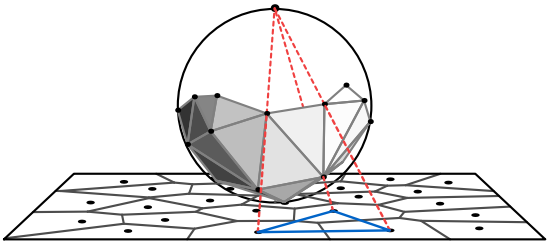


Figura 2.13: Construção do diagrama de Voronoi por projecção na esfera.

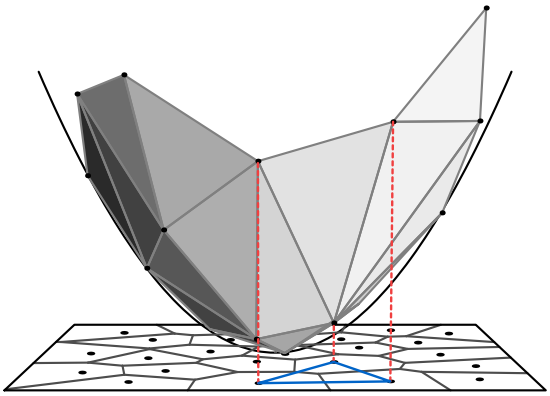


Figura 2.14: Construção do diagrama de Voronoi por projecção no parabolóide.

xidade $O(n \log n)$ no pior caso, sendo n o número de pontos. Uma alternativa assimptoticamente mais lenta, mas de implementação mais simples, é dada pelo algoritmo de Clarkson e Shor [18], cuja complexidade é de $O(n \log n)$ no caso esperado. Neste último algoritmo, o invólucro-convexo tridimensional é calculado incrementalmente, adicionando os locais um a um.

Outro resultado importante, também devido a Brown [9], relaciona pontos na superfície da esfera com a sua projecção estereográfica num plano (ver Figura 2.13). O diagrama de Voronoi de pontos no plano é topologicamente equivalente ao invólucro-convexo dos pontos na esfera visíveis do plano. Mais especificamente, os locais que definem uma face do invólucro-convexo, visível do plano, também definem um vértice no diagrama de Voronoi (ou, de modo equivalente, um triângulo na triangulação de Delaunay). De facto, é interessante notar que Brown fez uso desta propriedade para construir o diagrama de Voronoi no plano. Na *et al.* [46], baseando-se na relação inversa, mostraram como construir o diagrama de Voronoi esférico combinando dois diagramas de Voronoi planares dos locais projectados. O algoritmo resultante funciona para quaisquer locais compactos. Para locais pontuais, a complexidade temporal é de $O(n \log n)$.

A correspondência entre o diagrama de Voronoi planar e o invólucro-convexo de pontos na superfície da esfera, descoberta por Brown [9], é um exemplo de conexão entre uma estrutura num espaço (2D) e uma superfície imersa num espaço de ordem superior (3D). Mais tarde, Edelsbrunner e Seidel [32] descobriram uma segunda correspondência entre diagramas de Voronoi e uma superfície imersa num espaço de ordem superior, sob a forma de uma projecção de pontos num parabolóide. Esta correspondência foi usada por Preparata e Shamos [50] para desenvolver um algoritmo de construção do diagrama de Voronoi. Neste algoritmo, cada ponto $p = (x, y)$ do plano é projectado no parabolóide definido por $z = x^2 + y^2$. Ou seja, a cada ponto p do plano é associado um ponto $p^* = (x, y, x^2 + y^2)$ posicionado na superfície do parabolóide (ver Figura 2.14). A projecção no parabolóide permite a construção do diagrama de Voronoi pelo mesmo método que a projecção estereográfica, uma vez que a parte do invólucro-convexo, visível do plano, equivale topologicamente ao diagrama de Voronoi no

plano. Também neste caso se observa que cada face do invólucro-convexo corresponde a um vértice do diagrama de Voronoi no plano (que equivale também a um triângulo da triangulação de Delaunay).

As correspondências entre diagramas de Voronoi no plano e superfícies imersas em $\mathbb{R}^3$, seja na esfera ou no parabolóide, têm a propriedade de se generalizar para $\mathbb{R}^n$. A relevância desta propriedade é notória dada a existência de algoritmos de construção de invólucros-convexos em dimensão arbitrária [12, 16, 5].

A inserção de um local num diagrama pode ser feita exactamente como indicam os algoritmos incrementais, atrás descritos. Se a estrutura de dados a actualizar é um diagrama de Voronoi, a inserção pode ser feita pelo algoritmo de Green e Sibson [38] que, como foi referido, aplica-se ao plano e à esfera. Se a estrutura de dados a actualizar é um invólucro-convexo (por via de uma das correspondências indicadas entre um diagrama de Voronoi e um invólucro-convexo) a inserção do novo local pode ser feita com o algoritmo incremental de Clarkson e Shor [18]. Em qualquer dos casos, a inserção é a operação natural destes algoritmos, não sendo necessária qualquer adaptação para este fim.

2.5 Algoritmos de remoção

A operação de remoção não é tão simples como a de inserção, estando a investigação recente focada principalmente no domínio planar. A remoção de um local de um diagrama é feita em duas fases. Na primeira fase, que é trivial, são removidas as arestas do diagrama que formam a fronteira da região do local a remover, criando um buraco. Na segunda fase, bastante mais complexa, o buraco é preenchido com novas arestas que estendem a fronteira das regiões vizinhas, completando a remoção.

Existem várias soluções para a segunda fase. O algoritmo de Aggarwal *et al.* [1] tem um grande interesse teórico porque executa em tempo linear, no pior caso. No entanto, é demasiado intrincado e, por isso, de difícil implementação.

O algoritmo de Chew [14] constrói o diagrama de Voronoi de um conjunto de pontos localizados nos vértices de um polígono convexo, tendo sido adaptado pelo autor para remover um local de um diagrama de Voronoi. Trata-se de um algoritmo aleatório incremental de complexidade linear, no caso esperado. Considerem-se os locais $P = p_1, \ldots, p_m$, vizinhos do local removido s, ordenados segundo a ordem induzida pelas arestas de $V(s)$. Sejam $r_1, \ldots, r_m$ os mesmos locais segundo uma ordem aleatória. Para cada local r_i ($i > 1$) é determinado um local $a_i \in \{r_1, \ldots, r_{i-1}\}$ vizinho de r_i, isto é, um local a_i com quem r_i partilha uma aresta em $V(\{r_1, \ldots, r_i\})$. A determinação de a_i é fácil de elaborar se se inverter o sentido do problema, imaginando que se eliminam os locais de $V(P)$, um de cada vez. Por exemplo, se p_3 é eliminado (c.f. Figura 2.15), então p_2 e p_4 são vizinhos em $V(P \setminus \{p_3\})$; se p_2 é eliminado, então p_1 e p_4 são vizinhos em $V(P \setminus \{p_2, p_3\})$, e assim sucessivamente. Dada a relativa simplicidade da estrutura do diagrama de Voronoi de P, uma lista duplamente ligada dos locais p_i é suficiente para registar a relação de vizinhança dos locais vizinhos ainda não

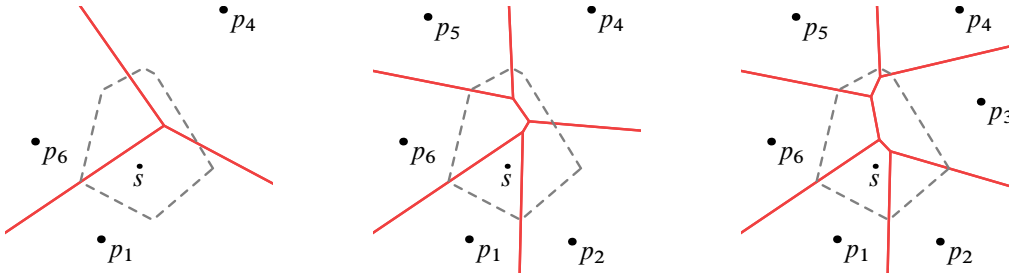


Figura 2.15: Remoção de um local de um diagrama pelo algoritmo de Chew.

"eliminados". No caso geral, em que $V(s)$ é uma região limitada, há sempre dois locais candidatos para vizinho de r_i, qualquer um elegível como a_i. Se $V(s)$ é uma região não-limitada, a sua fronteira é agora uma linha poligonal (não-fechada). Neste caso, existem dois locais que, apesar de serem adjacentes em P, não são necessariamente vizinhos em $V(P)$; após a remoção de s, estes podem ficar separados por uma região infinita. Este caso pode ser identificado assinalando, na lista duplamente ligada de locais de P, a adjacência que liga os locais inicialmente separados por uma região infinita, e nunca os escolhendo como vizinhos um do outro.

Uma vez determinado o vizinho a_i, o diagrama $V(P)$ é construído iterativamente. Primeiro é criado $V(r_1)$. Depois, cada r_i é inserido em $V(\{r_1, \ldots, r_{i-1}\})$, seguindo a fronteira da sua região que começa com uma aresta entre r_i e a_i. Ou seja, é construída a região de cada r_i recortando o diagrama de Voronoi formado pelos locais já adicionados, de uma forma idêntica à utilizada no algoritmo de inserção. A Figura 2.15 ilustra este algoritmo.

A inserção dos locais vizinhos segundo uma ordem aleatória garante uma complexidade temporal de $O(m \log m)$, no caso esperado. A principal desvantagem desta abordagem é a ineficiência dos passos intermédios: durante o processo, são calculadas e adicionadas arestas que não pertencem ao diagrama final.

Apesar de não ser explicitado na literatura, o algoritmo de Chew [14] é facilmente adaptável ao domínio esférico. Tal como o algoritmo de inserção incremental, basta substituir, na descrição do algoritmo, segmentos de recta por arcos de círculo máximo.

Capítulo 3

Triangulação de Delaunay

Um diagrama de Voronoi não só agrega as relações de vizinhança espacial entre locais, como também define a forma como essas relações são representadas. Independentemente da forma dos locais ou da função distância considerada, um diagrama de Voronoi associa a cada local uma região, cuja fronteira é representada explicitamente, tendo as regiões ligadas entre si. No entanto, as mesmas relações de vizinhança podem ser descritas de uma forma alternativa. A triangulação de Delaunay é uma estrutura de dados que agrega a mesma informação que o diagrama de Voronoi, mas representando explicitamente as ligações entre locais, sendo a região de cada local apenas representada de forma implícita. As duas estruturas são duais uma da outra. Dois locais são vizinhos no diagrama de Voronoi se e só se estão ligados por uma aresta na triangulação de Delaunay. Consequentemente, a triangulação de Delaunay é formada pelo conjunto das arestas incidentes a locais vizinhos (no diagrama de Voronoi). A Figura 3.1 ilustra a triangulação correspondente ao diagrama da Figura 2.1.

Por terem a mesma capacidade descritiva, a triangulação de Delaunay e o diagrama de Voronoi têm um uso semelhante como ferramentas de análise de dados espaciais. Consoante o tipo de informação pretendida, pode ser mais natural sintetizar as relações geométricas numa ou noutra estrutura. Dois exemplos ilustram a diferença.

O primeiro exemplo é dado pela modelação de terrenos por conjuntos de pontos. Um modelo digital de terreno é uma representação de uma superfície tridimensional (normalmente a

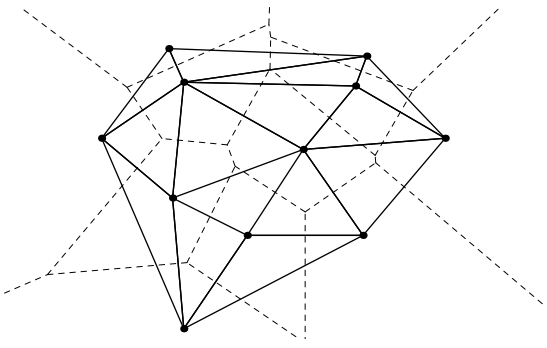


Figura 3.1: Triangulação de Delaunay planar.

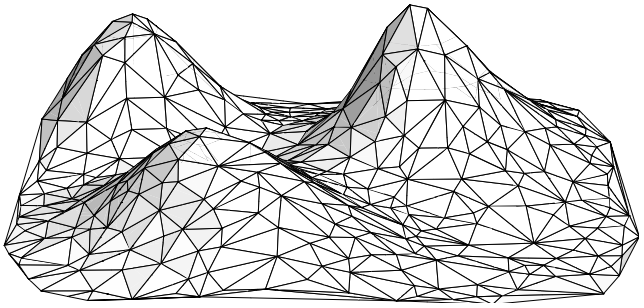


Figura 3.2: Exemplo de um modelo digital de terreno.

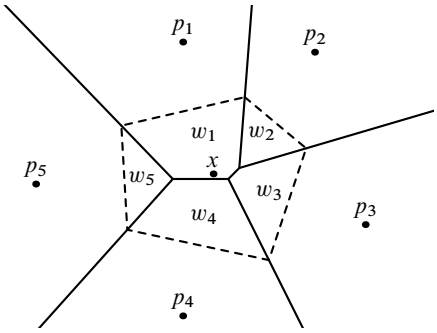


Figura 3.3: Exemplo de interpolação por vizinhos naturais.

superfície terrestre), criada a partir do conhecimento da elevação num número finito de posições, formando uma rede irregular de pontos. Esta rede irregular é, habitualmente, agrupada numa triangulação de Delaunay, preferida pela propriedade intrínseca que esta tem de maximizar os ângulos internos dos triângulos. Neste contexto, a triangulação é construída a partir das projecções dos pontos na superfície horizontal. A agregação das elevações numa triangulação é vantajosa porque facilita a construção de uma representação gráfica (ilustrada na Figura 3.2), assim como a identificação de características morfológicas (cristas, vales, encostas, planaltos, etc.). Neste contexto, a triangulação de Delaunay é, obviamente, a ferramenta adequada.

O segundo exemplo é dado por um método de interpolação. Interpolar é estimar o valor de uma medida (temperatura, por exemplo) numa posição arbitrária, a partir de medidas conhecidas em posições próximas. A interpolação por vizinhos naturais calcula uma estimativa sobre o diagrama de Voronoi dos pontos em que são conhecidas medidas. Neste método, a interpolação é feita inserindo um local num diagrama de Voronoi, na posição a interpolar, o que gera uma região cuja área é ganha à custa da redução da área das regiões vizinhas. Os locais cuja área foi reduzida são os vizinhos naturais do ponto a interpolar. O valor da interpolação é a média ponderada das medidas dos vizinhos naturais, com o peso de cada vizinho dado pela área *roubada* relativa (exemplificado pelas áreas dos polígonos $w_1, \ldots, w_5$ na Figura 3.3). Para este fim, é necessário calcular áreas e intersecções de regiões de Voronoi, sendo o diagrama de Voronoi a escolha preferível. O método de interpolação por vizinhos naturais é apresentado em detalhe na secção 6.3.

Dado que o diagrama de Voronoi e a triangulação de Delaunay são estruturas duais, construir uma equivale a construir a outra, mesmo que implicitamente. Naturalmente, um algoritmo está intimamente ligado à estrutura de dados que constrói. Por exemplo, a construção incremental da triangulação de Delaunay (descrita na secção 3.2) é feita apenas por manipulações do elemento mais básico da triangulação: o triângulo. Em contraste, o algoritmo de construção do diagrama de Voronoi pelo varrimento do plano (descrito na secção 4.2) identifica, essencialmente, arestas entre locais vizinhos. Por este motivo, é interessante comparar os algoritmos de construção e edição dos diagramas de Voronoi com os algoritmos preferenciais

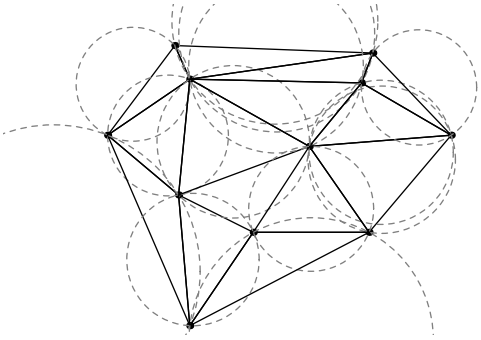


Figura 3.4: Propriedade dos círculos circunscritos aos vértices de triângulo numa triangulação de Delaunay.

de construção e edição da triangulação, seja no plano ou na esfera.

Vamos ver primeiro as propriedades das triangulações de Delaunay e depois detalhar os algoritmos de construção e edição respectivos.

3.1 Propriedades

No plano, uma *triangulação* de um conjunto de pontos P é uma subdivisão do invólucro-convexo em triângulos de tal modo que dois triângulos apenas se intersectam nos lados e os vértices de todos os triângulos coincidem com P. No caso geral, um conjunto de pontos aceita um número elevado de triangulações. No entanto, de entre todas as triangulações possíveis, existe um tipo de triangulação que se destaca: a *triangulação de Delaunay*. Este tipo de triangulação tem a propriedade de maximizar o ângulo mínimo (nos vértices) de todos os triângulos na triangulação. O resultado é uma triangulação que escolhe, sempre que possível, triângulos com uma forma mais próxima de um triângulo equilátero, evitando, sempre que possível, triângulos longos e estreitos. Esta propriedade torna a triangulação de Delaunay especial, potenciando o seu uso em muitos contextos.

Outra propriedade desta estrutura relaciona-a com os diagramas de Voronoi: os círculos circunscritos aos vértices de cada triângulo não contêm nenhum ponto de P no seu interior (ilustrado na Figura 3.4). Desta forma, pelas propriedades dos círculos vazios máximos, obtém-se uma equivalência entre vértices no diagrama e triângulo na triangulação.

Para finalizar, enunciam-se algumas equivalências entre diagramas de Voronoi e triangulações de Delaunay. Em diagramas de Voronoi em que todos os vértices têm grau três, tem-se que:

- um vértice no diagrama corresponde a um triângulo da triangulação, com o vértice localizado no centro do círculo circunscrito aos três locais que definem o triângulo;

- dois locais estão ligados por uma aresta na triangulação se e só se partilham uma aresta do diagrama;

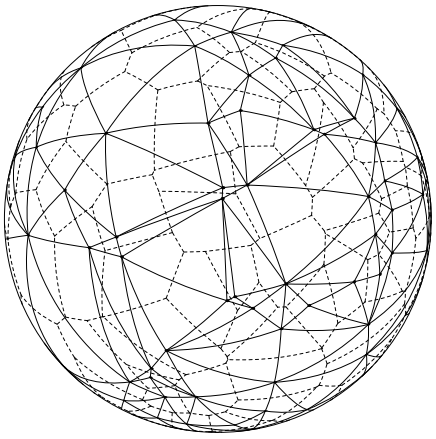


Figura 3.5: Triangulação de Delaunay esférica.

- cada aresta do diagrama de comprimento infinito corresponde a uma aresta da triangulação que pertence ao invólucro-convexo dos locais.

A presença de vértices de grau múltiplo (superior a três) num diagrama implica uma restrição adicional. Extrapolando a equivalência entre vértices e triângulos, um vértice de grau k corresponde a um polígono (na triangulação) de k lados e vértices co-circulares. Para completar a triangulação é suficiente triangular o polígono referido (por uma triangulação arbitrária). Nestes casos, a triangulação de Delaunay não é única. Porém, prova-se facilmente que todas as triangulações possíveis (do polígono) possuem os mesmos ângulos internos [20]. Isto é, é garantida a maximização do menor ângulo nos vértices, qualquer que seja a triangulação escolhida.

A triangulação de Delaunay esférica possui as mesmas propriedades e a mesma correspondência descritas para o caso planar, com excepção de duas diferenças:

- os triângulos (esféricos) são formados por arcos de círculo máximo;
- todas as arestas (do diagrama) têm comprimento finito.

De resto, a triangulação representa uma subdivisão do domínio esférico, tal como o diagrama de Voronoi. A Figura 3.5 ilustra a triangulação de Delaunay esférica dual do diagrama de Voronoi da Figura 2.4.

3.2 Algoritmos de construção e de inserção

Vamos agora descrever a escolha usual para a construção da triangulação de Delaunay. Inicialmente, este algoritmo foi desenvolvido para a construção da triangulação no plano [38, 20] tendo sido, posteriormente, adaptado ao domínio esférico [52]. Representa a escolha mais usual por ser um algoritmo conceptualmente simples e de fácil implementação. É também uma opção eficiente, competindo, nesse aspecto, com o algoritmo desenvolvido neste trabalho.

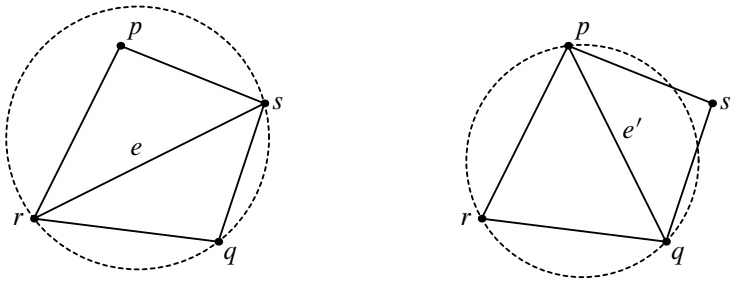


Figura 3.6: Rotação de uma aresta numa triangulação. No quadrilátero definido por dois triângulos adjacentes, a rotação de uma aresta equivale à troca de uma diagonal do quadrilátero pela outra diagonal. Permite a legalização de uma aresta. Em (a), a aresta e é ilegal; após rotação, em (b), a aresta e' é legal.

A construção da triangulação de Delaunay faz-se por um algoritmo aleatório incremental. A inserção de um local numa triangulação é feita em duas etapas. Primeiro é procurado o triângulo que contém o novo local, assumindo-se, por agora, que o local está contido no interior do triângulo. Depois, o novo local é inserido no triângulo, adicionando três arestas entre o novo local e os vértices do triângulo, gerando três novos triângulos. Por fim, modifica-se a triangulação em torno do novo local, por forma a garantir o critério de Delaunay na triangulação resultante. Vamos primeiro ver como é feita a operação de inserção, deixando a procura do triângulo que contém um local para mais tarde.

Seja *triângulo de Delaunay* um triângulo (de uma triangulação) que verifica o critério de Delaunay: o círculo circunscrito aos seus vértices não contém nenhum local no seu interior. Com a inserção de um ponto num triângulo, são gerados três novos triângulos que, regra geral, não são triângulos de Delaunay. Para repor o critério de Delaunay, a triangulação é modificada em torno do novo local através de uma sequência de rotações de arestas, que fazem parte de uma operação denominada de *legalização de aresta*.

A operação de legalização de arestas é a operação central da construção de uma triangulação de Delaunay. Uma aresta diz-se ilegal se está em conflito com algum dos locais dos triângulos adjacentes.

Definição 3 (Aresta ilegal)**.** *Seja $e = \overline{r\,s}$ uma aresta incidente aos triângulos $\triangle q\ r\ s$ e $\triangle s\ r\ p$ (ver Figura 3.6). A aresta e diz-se ilegal se o local p está em conflito com o triângulo $\triangle q\ r\ s$, isto é, se p está contido no interior do círculo C_{qrs} circunscrito aos vértices q, r e s. Se e é ilegal, então q está igualmente em conflito com o triângulo $\triangle s\ r\ p$.*

A posição relativa de um ponto p em relação ao círculo circunscrito a três pontos q, r, e s é dada pelo sinal do determinante:

$$\text{lado\_do\_círculo}(p, C_{qrs}) = \begin{vmatrix} q_x & q_y & q_x^2 + q_y^2 & 1 \\ r_x & r_y & r_x^2 + r_y^2 & 1 \\ s_x & s_y & s_x^2 + s_y^2 & 1 \\ p_x & p_y & p_x^2 + p_y^2 & 1 \end{vmatrix}. \tag{3.1}$$

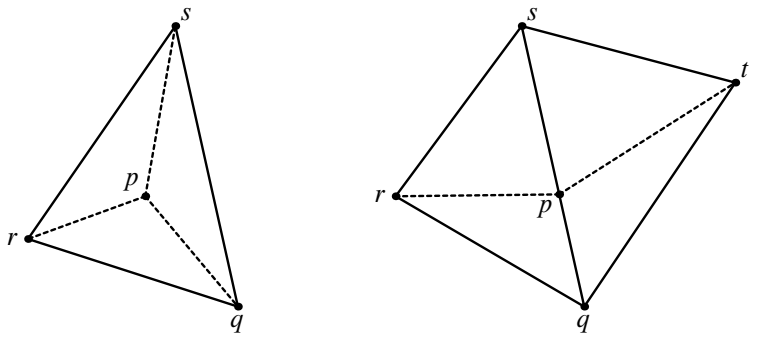


Figura 3.7: Inserção de um local no triângulo inicial: (a) se p está contido num triângulo, (b) se p está sobre uma aresta.

Em particular, p pertence ao interior de C_{qrs} se, e só se, lado_do_círculo$(p, C_{qrs}) > 0$.

Com esta definição, a triangulação de Delaunay pode ser redefinida como uma triangulação em que todas as arestas são legais.

Se os locais p, q, r e s formam um quadrilátero convexo, então é possível trocar uma diagonal pela diagonal transversa sem invalidar a triangulação, isto é, trocar $\overline{r\,s}$ por $\overline{p\,q}$ sem que a nova aresta intersecte as restantes. Esta operação denomina-se de *rotação de aresta* (ilustrada na Figura 3.6). Por outro lado, num quadrilátero definido nas condições indicadas, tem-se que uma das arestas $\overline{r\,s}$ ou $\overline{p\,q}$ é legal, enquanto que a outra pode ser ilegal. Logo, se os triângulos incidentes a uma aresta formam um quadrilátero convexo e se a aresta é ilegal, então uma rotação da aresta legaliza-a. Por outro lado, se os triângulos adjacentes a uma aresta formam um quadrilátero não-convexo, então a aresta que forma a única diagonal é necessariamente legal.

Numa triangulação, a legalização de uma aresta por via de uma rotação pode ter o efeito secundário de tornar ilegal qualquer uma das arestas adjacentes (que compõem o quadrilátero). A ilegalidade destas arestas é igualmente resolvida com rotações podendo, eventualmente, ilegalizar outras arestas. A continuação deste processo pode causar uma cascata de rotações. No entanto, prova-se que o processo de legalização de arestas não só não despoleta um ciclo infinito de rotações, como é convergente, transformando uma triangulação qualquer numa triangulação de Delaunay (terminado o processo) [20]. Em particular, a estratégia de legalização sucessiva de arestas possibilita a inserção eficiente de um local numa triangulação (como veremos a seguir).

Como atrás indicado, a inserção de um ponto numa triangulação começa por adicionar triângulos entre o novo local e os locais vizinhos. Normalmente, este passo inicial destrói (localmente) a condição de Delaunay, que é reposta modificando a triangulação em redor do novo local.

Vamos ver primeiro os casos em que o local p a inserir está contido no invólucro-convexo da triangulação T. Se p está contido num triângulo $\triangle q\,r\,s$ (ver Figura 3.7a), então começa-se por repartir esse triângulo adicionando arestas entre p e os três vértices do triângulo. Se p se encontra posicionado numa aresta (que não pertence ao invólucro-convexo), então

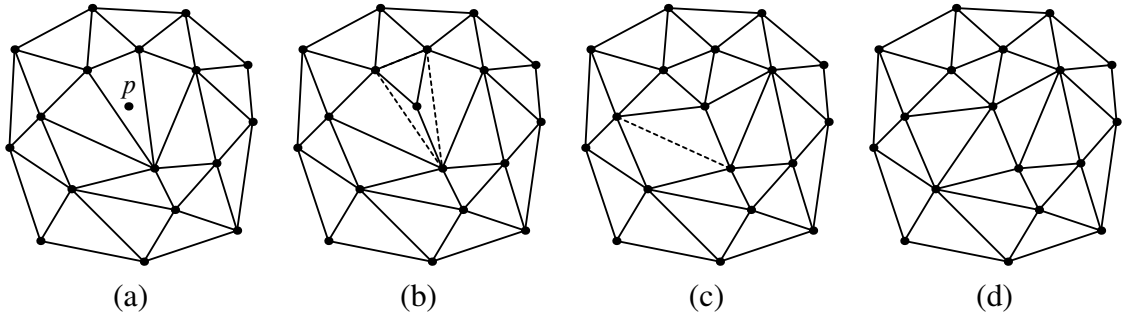


Figura 3.8: Inserção de um local na triangulação. As arestas ilegais estão assinaladas a tracejado. (a) Triangulação inicial; (b) após inserção das três arestas iniciais; (c) depois da rotação de duas arestas ilegais; (d) após a rotação da última aresta ilegal.

adicionam-se apenas duas arestas entre p e cada vértice oposto dos triângulos adjacentes (ver Figura 3.7b). Este passo inicial adiciona duas ou três arestas, que são necessariamente legais. No primeiro caso (Figura 3.7a), os três quadriláteros adjacentes às novas arestas são obrigatoriamente não-convexos, o que justifica a legalidade das arestas indicadas. No segundo caso (Figura 3.7b), a rotação de qualquer uma das novas arestas iria gerar um triângulo com os vértices colineares que, obviamente, não pode pertencer à triangulação de Delaunay. Portanto, as novas arestas são necessariamente legais. Por outro lado, as arestas opostas a p pertencem a novos triângulos, podendo estar agora ilegais. Por esse motivo, a legalidade destas arestas deve ser testada e, eventualmente, rodadas arestas.

Vamos ver um exemplo do primeiro caso, ilustrado na Figura 3.8. De início, p tem três arestas incidentes e três arestas opostas (c.f. Figura 3.8b). É determinada a legalidade de cada uma das arestas opostas, rodando-as, se necessário, para as legalizar. A rotação de uma destas arestas incrementa o número de arestas incidentes a p (ver Figura 3.8c). Ao mesmo tempo, cada rotação gera dois novos triângulos, ambos adjacentes à aresta rodada. Estes triângulos têm duas arestas incidentes a p, enquanto que a terceira é agora oposta a p, cuja legalidade deve também ser testada. Este processo é iterado, causando a rotação de arestas ilegais, ao mesmo tempo que incrementa o número de arestas incidentes a p (ver Figura 3.8d). Observe-se que, neste processo, apenas são rodadas arestas opostas a p, que passam a incidentes a p (e nunca o contrário). Dado que, numa triangulação, o grau de um vértice é finito, o processo de legalização de arestas em torno da inserção de um novo local é feito num número finito de passos.

O segundo caso (c.f. Figura 3.7b) difere apenas no número inicial de arestas potencialmente ilegais (quatro em vez de três), sendo processado de forma semelhante.

Uma triangulação é construída iterativamente, aplicando o algoritmo de inserção a cada ponto. A triangulação dos primeiros três pontos é trivial. Apenas para os restantes é necessário determinar o triângulo (da parte já construída) que contém cada ponto.

Este algoritmo de inserção é bastante simples e capaz de resolver a maioria dos cenários de inserção de um local numa triangulação. Apenas ficaram por tratar os casos em que a inserção é feita sobre ou fora do invólucro-convexo da triangulação corrente, em que não existe

um triângulo a conter o novo local. Estes casos podem ser resolvidos de uma forma muito expedita com o auxílio de um triângulo suficientemente grande para englobar o conjunto de todos os pontos. Neste caso, a construção é iniciada com o triângulo auxiliar, garantindo que qualquer local está contido num triângulo. No final do processo iterativo, basta descartar as arestas incidentes aos vértices do triângulo auxiliar, para se obter a triangulação pretendida.

O algoritmo atrás descrito insere um local numa triangulação em tempo e espaço lineares no número de arestas incidentes ao local inserido. Efectivamente, o algoritmo realiza um percurso em largura na triangulação em redor do local inserido, suportado, por exemplo, por uma fila de arestas. O percurso começa com a inserção na fila das três ou quatro arestas opostas a p. Depois, enquanto a fila não está vazia, é retirada uma aresta, a sua legalidade é testada e, se é ilegal, é rodada e são inseridas na fila as duas novas arestas (opostas a p). O processamento de cada aresta apenas envolve um número constante de operações. No caso mais geral, ao todo são processadas $3+2\,(k-3) = 2\,k-3$ arestas, sendo k o número de arestas incidentes a p (as três primeiras são inseridas directamente e cada uma das restantes implica o escalonamento de duas arestas na fila). No máximo, a fila terá $3 + (k-3) = k$ arestas (porque cada iteração só pode aumentar a fila em um elemento; remove uma aresta e adiciona duas arestas). No outro caso (novo local sobre aresta), são processadas $4+2\,(k-4) = 2\,k-4$ arestas, sendo k o comprimento máximo da fila.

Falta apenas descrever como é feita a procura do triângulo que contém um ponto. Descreve-se a seguir a solução mais comum.

Durante a construção incremental da triangulação de Delaunay, a maior parte dos triângulos inseridos são posteriormente eliminados pela inserção de novos triângulos. A ideia é construir simultaneamente uma estrutura de pesquisa D, implementada por um grafo orientado e acíclico. Nesta estrutura, as "folhas" são os triângulos na triangulação corrente e os "nós internos" correspondem a triângulos que foram entretanto eliminados. O grafo D é inicializado com o triângulo auxiliar que engloba o conjunto de pontos. Depois, este é aumentado de duas formas. Na repartição de um triângulo, há a troca de um triângulo por três novos triângulos, sendo adicionados ao nó correspondente em D três nós "filhos" (ver Figura 3.9a e b). Na rotação de uma aresta, há a troca de dois triângulos por outros dois novos triângulos, sendo adicionados dois nós "folha" para os novos triângulos, que são ligados simultaneamente aos dois nós entretanto eliminados (ver Figura 3.9b e c). Pela forma como é construído, cada nó interno de D tem no máximo três "filhos" e qualquer nó tem no máximo dois "pais".

Uma pesquisa em D é feita tal como uma pesquisa numa árvore, executando um percurso da raiz para uma folha, identificando o triângulo que contém o ponto de pesquisa. O custo de uma pesquisa é proporcional ao número de nós visitados que, por sua vez, depende da forma adquirida pelo grafo durante a construção da triangulação. No pior dos casos, obtém-se uma estrutura em que uma pesquisa percorre todos os nós internos. Porém, a probabilidade destes casos degenerados é baixa se os locais forem inseridos por uma ordem aleatória.

Durante a construção de uma triangulação de Delaunay pelo algoritmo incremental, prova-se que, no caso esperado, o número de triângulos gerados não supera $9n+1$ [20]. Ou seja, uma pesquisa em D faz-se, em média, em tempo $O(\log n)$, ocupando esta estrutura $\Theta(n)$ espaço.

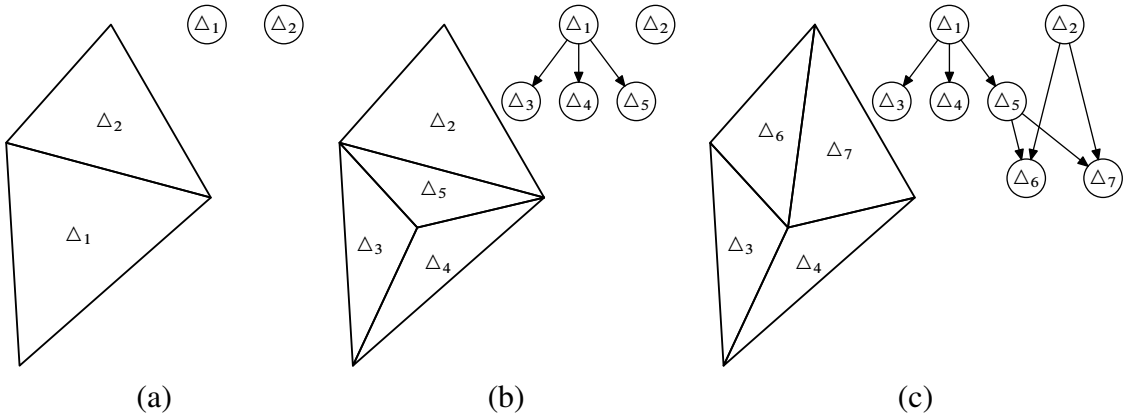


Figura 3.9: Construção de um grafo de pesquisa numa triangulação. Na inserção de um ponto num triângulo são adicionados três sucessores directos ao nó do triângulo repartido (de (a) para (b)). Na rotação de uma aresta são adicionados dois sucessores directos a ambos os nós dos triângulos incidentes à aresta rodada (de (b) para (c)).

Como corolário, tem-se que o algoritmo incremental constrói a triangulação de Delaunay, no caso esperado, em tempo $O(n \log n)$ e espaço $\Theta(n)$ [20].

A estratégia de procura apresentada é crucial para que o algoritmo incremental de construção da triangulação de Delaunay tenha um custo no tempo e no espaço óptimos, no caso esperado. Na secção 7.4 são descritas outras estratégias de procura que mantêm o mesmo custo computacional.

3.3 Algoritmo de remoção

Vamos ver agora como é feita a remoção de um local de uma triangulação. A remoção de um local é feita em duas etapas: primeiro são removidos os triângulos incidentes ao local a remover p, criando um "buraco" poligonal B, que é necessário triangular numa segunda etapa. A remoção dos triângulos é trivial; basta remover as arestas incidentes a p, o que gera um buraco formado pelas arestas opostas a p. A triangulação do buraco pode ser feita com uma estratégia semelhante à seguida na inserção: triangular o polígono de uma forma arbitrária, seguida de uma fase de legalização de arestas, até se obter de novo uma triangulação de Delaunay. Embora simples, esta solução tem um custo quadrático no número de arestas do buraco poligonal.

Uma solução mais eficiente é dada pelo algoritmo de Devillers [24], que remove um local de uma triangulação de Delaunay com um custo $O(k \log k)$ no tempo, sendo k o grau do vértice removido. Aplica-se a triangulações de Delaunay planares, mas é facilmente adaptável ao domínio esférico. A abordagem faz-se em torno do conceito de *orelha* de um polígono. Num polígono B, três vértices consecutivos $p_i\ p_{i+1}\ p_{i+2}$ formam uma orelha em B se o segmento $\overline{p_i\ p_{i+2}}$ está contido em B e não cruza a sua fronteira.

O algoritmo de Devillers segue a estratégia da adição sucessiva de orelhas a B, determinando, em cada iteração, uma orelha que pertence necessariamente à triangulação de Delau-

nay. A adição de cada orelha encurta B em uma aresta. Naturalmente, a triangulação termina quando B fica reduzido a apenas três arestas.

Resta saber como determinar qual das orelhas de B é, garantidamente, um triângulo de Delaunay. A solução para este problema faz uso da dualidade entre a triangulação de Delaunay em d dimensões e o invólucro-convexo em $d+1$ dimensões [33] e recorre a um algoritmo de enumeração de faces de um poliedro [53]. Vamos ver cada uma destas ferramentas em detalhe.

A dualidade entre uma triangulação e um invólucro-convexo é obtida por intermédio do parabolóide Π, definido por $z = x^2 + y^2$, já usado atrás, para a construção do diagrama de Voronoi. Associando cada ponto $p = (x, y) \in P$ ao ponto $p^* = (x, y, x^2 + y^2)$ na superfície do parabolóide Π, a triangulação de Delaunay de P é dada pela projecção (no plano) do invólucro-convexo dos pontos p^*. Mais concretamente, cada face triangular do invólucro-convexo (vista de baixo) corresponde exactamente a um triângulo de Delaunay. A dualidade entre estruturas também se estende às propriedades da triangulação de Delaunay. Sejam $p, q, r, s \in P$. Tem-se que p está contido no círculo C_{qrs} circunscrito a $q\ r\ s$ se e só se p^* está abaixo do plano P_{qrs} que passa por $q^*\ r^*\ s^*$, isto é, $\Pi \cap P_{qrs}$ projecta-se no círculo C_{qrs}. Tem-se também que a "distância vertical com sinal" entre p^* e P_{qrs} é dada pela potência do ponto p em relação a C_{qrs}, em que:

$$\text{potência}(p, C_{qrs}) = -\frac{\begin{vmatrix} q_x & q_y & q_x^2+q_y^2 & 1 \\ r_x & r_y & r_x^2+r_y^2 & 1 \\ s_x & s_y & s_x^2+s_y^2 & 1 \\ p_x & p_y & p_x^2+p_y^2 & 1 \end{vmatrix}}{\begin{vmatrix} q_x & q_y & 1 \\ r_x & r_y & 1 \\ s_x & s_y & 1 \end{vmatrix}}. \tag{3.2}$$

Mais precisamente, a potência$(p, C_{qrs}) < 0$ se p^* está abaixo de P_{qrs} e potência$(p, C_{qrs}) > 0$ se p^* está acima de P_{qrs}. Note-se que o determinante de 3×3 determina a orientação relativa de $q\ r\ s$ e o determinante de 4×4 determina a inclusão de p no círculo circunscrito a $q\ r\ s$. Um determinante de 3×3 negativo indica que $q\ r\ s$ não formam uma orelha ($\overline{q\ s} \not\subset B$).

O algoritmo de enumeração de faces de um poliedro serve de base a um algoritmo de construção de invólucros-convexos. Em termos gerais, o algoritmo funciona do seguinte modo. Imagine-se um observador que se desloca ao longo de uma linha recta que intersecta um poliedro, começando no ponto de intersecção. Inicialmente, o observador apenas consegue ver uma face. Mas, à medida que o observador se afasta do poliedro, vão sendo descobertas outras faces, uma a uma, até que, no infinito, o observador consegue ver "metade" do poliedro. A outra "metade" é descoberta de uma forma semelhante. A enumeração das faces obtidas desta forma, pela ordem com que são descobertas, tem a propriedade de produzir uma sequência de faces, em que cada face está sempre ligada às anteriores. Na enumeração, as faces descobertas podem ser de dois tipos: ou ligam uma aresta conhecida a um vértice novo, ou ligam dois

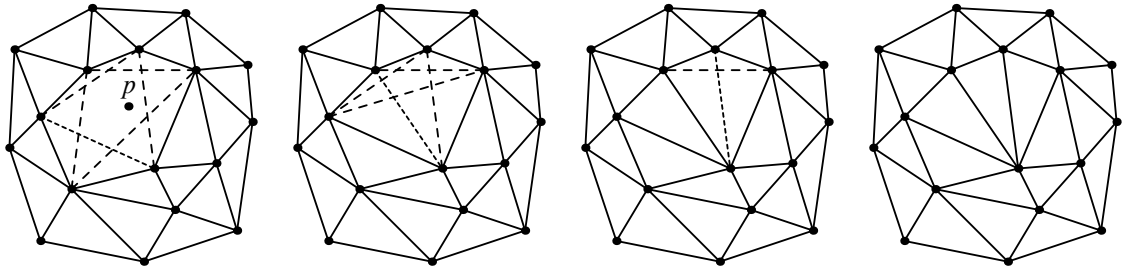


Figura 3.10: Remoção de um local de uma triangulação. As orelhas estão assinaladas a tracejado. Em cada iteração, a orelha com menor prioridade está assinalada com um tracejado mais curto.

vértices conhecidos anteriormente.

Munidos destas duas ferramentas, a determinação da orelha que é simultaneamente um triângulo de Delaunay é resolvida por partes. Primeiro, o problema da triangulação do buraco criado pela remoção de p é transformado no problema equivalente do fecho do buraco no invólucro-convexo criado pela remoção de p^*. Por seu lado, este problema é resolvido pelo processo de enumeração de faces. Imagine-se, por um momento, que o buraco no invólucro-convexo já se encontra fechado. Imagine-se também que um observador se desloca segundo a linha vertical que passa por p^*. Quando em p^*, o observador vê apenas a parte do invólucro-convexo interior ao bordo do buraco (que está estritamente acima de p^*). Subindo ao longo da linha vertical, as faces que compõem o invólucro-convexo do buraco são enumeradas pela ordem com que desaparecem da vista do observador, o que acontece exactamente quando o observador está posicionado na intersecção da linha vertical com o plano definido por cada face. Neste procedimento, apenas são encontradas faces do segundo tipo (ligam dois vértices conhecidos anteriormente), uma vez que não há lugar à descoberta de novos vértices.

Com as ferramentas descritas, fica agora trivial escolher a orelha que, garantidamente, faz parte da triangulação do buraco B. Cada orelha do polígono B define, pela projecção no parabolóide, um plano que está necessariamente acima de p^*. Pelo algoritmo de enumeração de faces, o plano mais próximo de p^*, segundo a vertical, corresponde a uma face do invólucro-convexo de $P^* \setminus \{p^*\}$ que, por sua vez, corresponde a uma orelha de B que é necessariamente um triângulo de Delaunay em $P \setminus \{p\}$. Pelas propriedades do mapeamento no parabolóide, o plano mais próximo de p corresponde à orelha $\triangle qrs$ de menor potência de p em relação a C_{qrs}.

Na forma completa, o algoritmo de Devillers é descrito do seguinte modo (ver Figura 3.10). Primeiro, removem-se os triângulos incidentes a p, formando um buraco B de forma poligonal. De seguida, é calculada a prioridade de cada orelha de B, inserindo as orelhas numa fila (com prioridade). A triangulação de B faz-se iterativamente. Seja uma orelha formada pelos três vértices $p_{i-1}\ p_i\ p_{i+1}$ identificada apenas pelo vértice do meio p_i. Retira-se da fila a orelha p_i com menor prioridade e insere-se a aresta $\overline{p_{i-1}\ p_{i+1}}$ na triangulação, o que encurta o buraco B em uma aresta. Ao mesmo tempo, esta operação invalida as duas orelhas

adjacentes, p_{i-1} e p_{i+1}, que adquirem uma nova forma. A prioridade destas duas orelhas é recalculada, actualizando-se também as respectivas posições na fila. O processo é iterado até que restem apenas três orelhas na fila, que correspondem aos três vértices do último triângulo descoberto.

Resta analisar o custo de uma remoção. Inserem-se as k orelhas iniciais na fila, removem-se $k-3$ orelhas e, por cada uma das remoções, excepto a última, há duas actualizações. Logo, o número total de operações na fila é de $k + (k-3) + 2\,(k-4) = 4\,k - 11$. Obviamente, a fila tem no máximo k elementos, com um custo de $O(\log k)$ por operação, o que justifica a complexidade temporal. O autor nota que, na prática, o custo computacional é dominado pelo custo do cálculo das prioridades. Estes são em número de $k + 2\,(k-4) = 3\,k - 8$ (k iniciais mais dois por cada orelha actualizada).

O autor não o refere, mas este algoritmo é facilmente adaptável à remoção de locais numa triangulação esférica. Primeiro, observe-se que, dado o isomorfismo existente entre a triangulação de Delaunay esférica e o invólucro-convexo tridimensional (dos mesmos pontos), não é necessário recorrer à projecção no parabolóide (ou similar) para transformar o problema numa operação sobre um poliedro. Logo, basta aplicar directamente o algoritmo de enumeração de faces de um poliedro para efectuar a remoção de um ponto da triangulação. Neste caso, imagina-se que o observador percorre uma linha recta ao longo da direcção radial que passa por p. As orelhas que formam a triangulação do buraco poligonal são igualmente dadas pela ordem com que os planos correspondentes são encontrados quando o observador se desloca de p para a origem. Concluindo, a remoção na esfera faz-se com o mesmo algoritmo que remove um ponto no plano, com excepção da prioridade. Esta é dada pela distância, ao longo da linha recta referida, entre p e o plano definido por cada orelha $\triangle q\,r\,s$. Ou seja:

$$\text{prioridade}(p, C_{qrs}) = \frac{\Delta}{\Delta' - \Delta}, \tag{3.3}$$

onde

$$\Delta = \begin{vmatrix} q_x & q_y & q_z & 1 \\ r_x & r_y & r_z & 1 \\ s_x & s_y & s_z & 1 \\ p_x & p_y & p_z & 1 \end{vmatrix} \quad \text{e} \quad \Delta' = \begin{vmatrix} q_x & q_y & q_z \\ r_x & r_y & r_z \\ s_x & s_y & s_z \end{vmatrix}. \tag{3.4}$$

Capítulo 4

Construção do diagrama planar

O algoritmo de varrimento para a construção do diagrama de Voronoi esférico é semelhante aos algoritmos de varrimento que constroem o diagrama de Voronoi planar. Por este motivo, neste capítulo são revistas as necessárias definições e resultados dos varrimentos planares. A secção 4.1 apresenta o varrimento linear, usado no algoritmo de Fortune [34], descrito na secção 4.2. A secção 4.3 descreve uma variante do algoritmo anterior em que o varrimento é feito por uma linha circular [21].

Os algoritmos de varrimento para a construção do diagrama de Voronoi podem ser compreendidos através da noção de círculo vazio máximo que, como já foi referido, é um círculo de raio máximo, centrado numa dada posição, que não contém nenhum local no seu interior. Recorde-se que o número de locais na sua fronteira caracteriza o ponto que é centro do círculo vazio máximo. Este ponto pertence ao interior de uma região de Voronoi se a fronteira contém apenas um local; o ponto pertence a uma aresta se a fronteira contém exactamente dois locais; e o ponto é um vértice de uma aresta se a fronteira contém três ou mais locais. Tanto o varrimento linear como o varrimento circular constroem o diagrama de Voronoi percorrendo, implicitamente, todos os círculos vazios máximos.

4.1 Varrimento linear

O algoritmo de Fortune constrói o diagrama de Voronoi pela técnica de varrimento. Um algoritmo desenhado segundo esta técnica *varre* conceptualmente o plano, transformando um problema a duas dimensões em dois problemas a uma dimensão, que se espera sejam de mais fácil resolução.

Inicialmente, esta técnica foi aplicada num problema de intersecções de segmentos: dado um conjunto de segmentos de recta, determinar se quaisquer dois se intersectam. É um problema que se coloca, por exemplo, quando se pretende determinar se um polígono é simples (isto é, um polígono que não se auto-intersecta) ou determinar se dois polígonos se intersectam.

Este problema foi resolvido pelo algoritmo de Shamos e Hoey [55]. Nele, o plano é var-

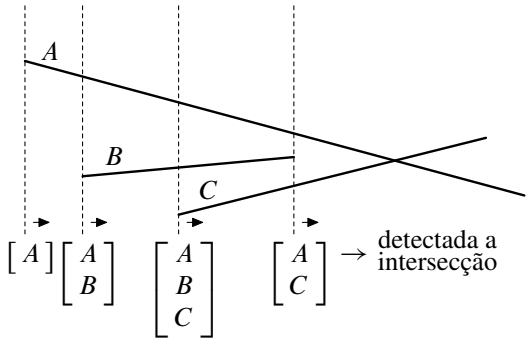


Figura 4.1: Detecção da existência de intersecção entre segmentos de recta pelo método do varrimento do plano.

rido por uma linha recta vertical, que se imagina a percorrer o plano num movimento contínuo da esquerda para a direita, tal como ilustrado na Figura 4.1. Para simplificar, assume-se que não existem segmentos verticais. O algoritmos baseia-se na observação de que, em cada instante do varrimento, as posições em que os segmentos de recta intersectam a recta vertical definem uma ordem nos segmentos tal que, se dois segmentos não se intersectam, então a sua ordem relativa não se altera durante todo o varrimento. Mas se um par de segmentos se intersecta, então a sua ordem relativa altera-se quando a linha de varrimento cruza o ponto de intersecção. O algoritmo procede do seguinte modo. Uma linha vertical *varre* o plano da esquerda para a direita, parando em cada extremo de cada segmento (previamente ordenados). Em cada posição (de paragem) do varrimento é mantida uma lista com os segmentos intersectados pela linha de varrimento, ordenada pela relação de ordem entre segmentos (e esquematizada por uma coluna de letras entre parênteses rectos na Figura 4.1). Quando se cruza o extremo esquerdo de um segmento, este é inserido na lista por ordem, sendo removido da lista quando se cruza o extremo direito do segmento. Caso dois segmentos se intersectem, prova-se que, em algum instante, estes ficam adjacentes na lista de segmentos. Deste modo, sempre que um segmento é inserido na lista, é verificada a sua intersecção com os segmentos que lhe são adjacentes e, quando um segmento é removido, é verificada a intersecção entre o novo par de segmentos adjacentes. O algoritmo termina assim que é detectada uma intersecção. O resultado é um algoritmo eficiente, repartido por duas partes lineares: o varrimento linear segundo a direcção horizontal e a manutenção da lista de segmentos definida pela linha vertical. Facilmente se prova que, se a estrutura de segmentos é implementada por uma árvore equilibrada, a detecção de intersecções tem um custo assimptótico no tempo de $O(n \log n)$, onde n é o número de segmentos. Em contraste, uma solução por força bruta teria um custo quadrático. Posteriormente, este algoritmo foi estendido para a determinação de todas as intersecções entre um conjunto de segmentos [6], notando que se pode manter a ordem correcta entre segmentos durante todo o varrimento se se trocar a ordem de um par de segmentos quando a linha de varrimento cruza um ponto de intersecção.

Mas o que interessa reter do algoritmo de Shamos e Hoey não é a resolução do problema da determinação de intersecções mas a técnica de varrimento introduzida para a sua resolução. É uma técnica que se aplica naturalmente a problemas de natureza geométrica, permitindo a

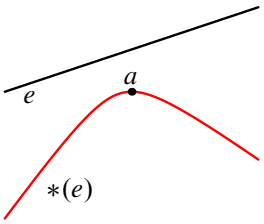


Figura 4.2: Transformação$-*$ de um segmento de recta.

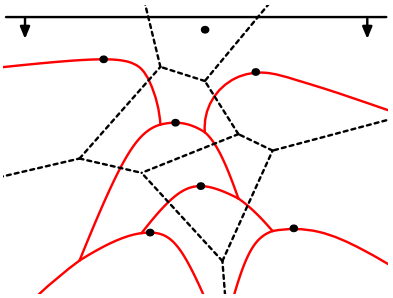


Figura 4.3: Aplicação da transformação$-*$ para um varrimento linear.

divisão de um problema bidimensional em dois problemas lineares para que se obtenha uma solução simultaneamente eficiente e elegante. Foi apenas uma questão de tempo para que esta técnica fosse aplicada a outros problemas e, em particular, à construção do diagrama de Voronoi, conseguido por Steven Fortune [34].

4.2 Algoritmo de Fortune

Embora atractiva pela sua eficácia, não é possível aplicar directamente a técnica de varrimento do plano à construção do diagrama de Voronoi. A dificuldade prende-se com o facto de um local estar sempre contido no interior da região de Voronoi que gera. Por este motivo, o varrimento de um plano por uma linha recta intersecta primeiro a fronteira da região de Voronoi (desconhecida nesse instante) *antes* do local respectivo. Esta dificuldade foi ultrapassada por Fortune [34]. Assumindo que a linha de varrimento percorre o plano de cima para baixo, a solução encontrada foi deformar o diagrama pela aplicação de uma transformação geométrica, denominada de transformação$-*$, definida por $*(p) = (p_x, p_y - \overline{p-a})$, sendo a o local mais próximo de um ponto arbitrário p. A Figura 4.2 exemplifica a transformação$-*$ de um segmento de recta. Esta transformação possui a dupla função de posicionar cada local na fronteira da região (deformada) respectiva, ao mesmo tempo que mantém a estrutura topológica do diagrama (ver Figura 4.3). Desta forma fica fácil determinar a posição da linha de varrimento quando inicia o varrimento de uma região deformada: é igual à ordenada do local respectivo.

Posteriormente, Guibas e Stolfi [41] desenvolveram uma forma alternativa de apresentar o mesmo algoritmo, trocando a transformação$-*$ por uma segunda linha de varrimento. É uma variante mais simples de ilustrar, sendo presentemente a forma usual de descrever o algoritmo de Fortune, que se apresenta a seguir.

4.2.1 Frente de onda parabólica

Seja H uma linha recta horizontal, de ordenada h, que varre o plano verticalmente, de cima para baixo. Para cada local $a = (a_x, a_y)$ acima ou sobre H, considere-se o lugar dos pontos

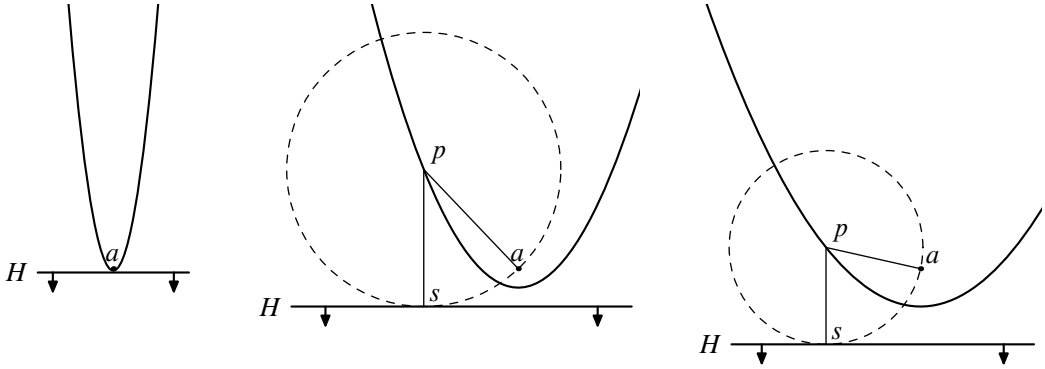


Figura 4.4: Parábola definida por um local a e a linha de varrimento H. Um ponto p pertence à parábola se está equidistante a a e a H (com a distância indicada pelo círculo a tracejado).

p equidistantes a a e a H. Para cada ponto p, existe um ponto $s \in H$ tal que $\overline{p-a} = \overline{s-p} = p_y - h$. Estes pontos formam uma parábola (ver Figura 4.4). A parábola começa degenerada numa semi-recta vertical, quando H cruza a, alargando à medida que H progride no varrimento. A Figura 4.4 ilustra a variação da forma da parábola com a posição da linha de varrimento.

Vamos agora ver em detalhe como uma parábola varre o plano e como várias parábolas se intersectam entre si. Primeiro consideram-se apenas os casos de parábolas geradas por locais estritamente acima da linha de varrimento. Mais à frente serão analisados os casos em que uma ou mais parábolas são degeneradas.

Seja $s = (s_x, s_y)$ um ponto da linha de varrimento H e $a = (a_x, a_y)$ um local tal que $s_y < a_y$. O ponto p da parábola gerada por a e H cuja abcissa é s_x é $p = (s_x, s_y + \tau(a,s))$, onde $\tau(a,s)$, que é a distância de p a H, é dada por:

$$\tau(a,s) = \frac{(a_y - s_y)^2 + (a_x - s_x)^2}{2 \cdot (a_y - s_y)} \,. \tag{4.1}$$

A parábola é definida pela directriz H e pelo foco a. Por construção, a parábola varre monotonamente todo o plano.

Considere-se um segundo local, $b = (b_x, b_y)$, igualmente acima de H. Os dois locais geram duas parábolas que se intersectam em um ou dois pontos. Seja $i = (i_x, i_y)$ um ponto da intersecção das parábolas. Então tem-se necessariamente que $\overline{i-a} = \overline{i-b} = i_y - h$, ou seja, i pertence à mediatriz do segmento de recta $\overline{a\,b}$. Se $a_y = b_y$ (ilustrado na Figura 4.5(a)), as duas parábolas intersectam-se apenas num ponto, localizado na recta vertical $x = (a_x + b_x)/2$. Caso contrário, intersectam-se exactamente em dois pontos, $\langle a,b \rangle$ e $\langle b,a \rangle$, que estão localizados em lados opostos da parábola gerada pelo local de menor ordenada. Por exemplo, se $a_y < b_y$ (tal como está desenhado na Figura 4.5(b)), as duas intersecções localizam-se nos semi-planos opostos definidos pela recta vertical $x = a_x$. Isto é, $\forall h \leq a_y : \langle b,a \rangle_x \leq a_x \leq \langle a,b \rangle_x$. Um resultado semelhante é obtido para $a_y > b_y$.

No caso geral ter-se-ão três ou mais locais acima de H, gerando cada um uma parábola.

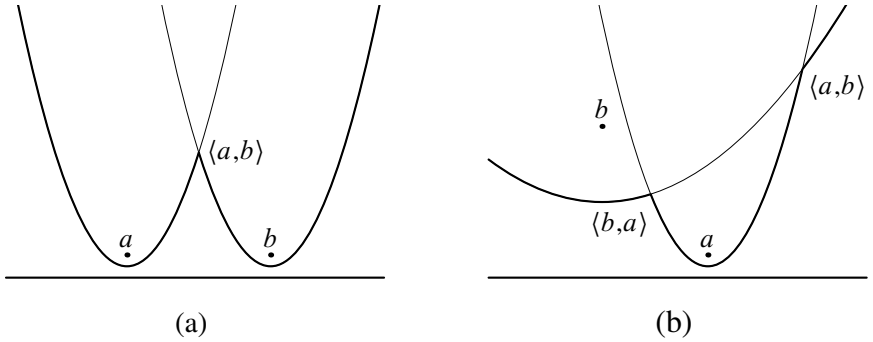


Figura 4.5: Intersecção de duas parábolas.

Para a construção do diagrama de Voronoi, interessa definir a curva W formada pelo envelope inferior de todas a parábolas, denominada de *frente de onda* (ilustrado na Figura 4.6). Trata-se de uma curva contínua, não fechada, formada por uma sequência alternada de arcos de parábola e intersecções de arcos.

A frente de onda possui a propriedade relevante de percorrer, implicitamente, os círculos vazios máximo do diagrama de Voronoi em construção. Por definição, um círculo $C(i)$ centrado num ponto arbitrário i de um arco da frente de onda e que seja tangente ao local que o gera é também tangente a H. Mas porque W é definida pelo envelope inferior de todas as parábolas, $C(i)$ não pode conter nenhum local de P acima de H no seu interior, ou i não seria um ponto de W. Por outro lado, $C(i)$ também não pode conter nenhum local abaixo de H, porque o círculo está no semi-plano superior a H. Logo $C(i)$ é necessariamente o círculo vazio máximo de i em P. Ou seja, a frente de onda, conjuntamente com a linha de varrimento, percorre todos os círculos vazios máximos de P em todo o plano. E pelas propriedades dos círculos vazios máximos, cada arco de W percorre o interior da região do local que o gera e as intersecções de arcos percorrem as arestas do diagrama de Voronoi. Em particular, dois locais a e b têm uma aresta comum no diagrama de Voronoi se, e só se, existe uma posição de H tal que W contém uma intersecção de arcos gerados por a e por b.

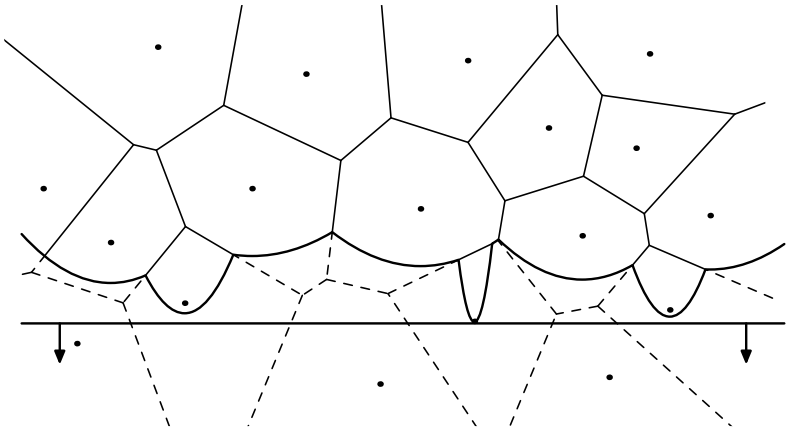


Figura 4.6: Frente de onda parabólica na construção de um diagrama de Voronoi pelo varrimento do plano.

Observe-se que a linha de varrimento não é mais do que a representação gráfica da transformação $-*$ da frente de onda. Mais exactamente, $W = *^{-1}(H)$. Ou seja, nesta formulação do algoritmo de Fortune, o varrimento de um diagrama (deformado) com uma linha recta é trocado pelo varrimento do diagrama (não deformado) com a transformada da linha de varrimento, que é a frente de onda parabólica. Embora equivalente, a visualização do varrimento pela frente de onda é preferível por tornar a descrição do algoritmo mais clara e intuitiva.

Resta apenas analisar os casos em que a linha de varrimento está posicionada sobre um ou mais locais, gerando parábolas degeneradas em semi-rectas verticais. Nestes casos, a equação (4.1) não se aplica, uma vez que o ponto a de H (e só este) corresponde a uma infinidade de pontos p. Por outro lado, duas parábolas degeneradas não se intersectam. Em particular, num cenário em que todas as parábolas são degeneradas (o que acontece quando todos os locais são colineares com H), o envelope inferior das parábolas não forma uma frente de onda contínua, dificultando a sua representação. Este obstáculo é ultrapassado forçando uma ordem estrita entre locais, por forma a normalizar a intersecção entre quaisquer parábolas (degeneradas ou não).

Parábolas geradas por dois locais com ordenadas diferentes intersectam-se sempre em dois pontos, inicialmente coincidentes, mas que depois se afastam em direcções opostas à medida que percorrem a mediatriz entre os locais respectivos. Por outro lado, e como atrás observado, parábolas geradas por locais de igual ordenada (mas não degeneradas) intersectam-se apenas num ponto. Porém, se se considerar que, para dois locais de igual ordenada, o local de menor abcissa está *acima* do local de maior abcissa, então tem-se que, conceptualmente, as parábolas geradas por estes dois locais também se intersectam em dois pontos. Isto é, apesar da linha de varrimento não se deslocar efectivamente entre os dois locais, considera-se que, quando se inicia a parábola (degenerada) do local de maior abcissa, a parábola do local de menor abcissa já assume uma forma não degenerada. Na prática, obtém-se uma configuração semelhante à de dois locais cuja ordenada difere de δy, quando δy tende para 0. Desta forma, durante o varrimento, nunca ocorrem duas parábolas degeneradas em simultâneo. Consequentemente, a frente de onda é sempre dada por uma linha curva única e contínua.

Segundo este esquema, duas parábolas intersectam-se sempre em dois pontos. Para duas parábolas geradas por locais de igual ordenada, os dois pontos percorrem uma linha recta vertical (mediatriz dos dois locais), coincidindo inicialmente em $+\infty$, seguindo uma na direcção de $-\infty$, ao passo que a outra segue um percurso imaginário para cima de $+\infty$. Na prática apenas uma das intersecções percorre efectivamente a mediatriz. Porém, a normalização da intersecção de parábolas sob esta forma simplifica a construção do algoritmo.

A Figura 4.7 ilustra a frente de onda para três locais a, b, c de igual ordenada, onde se observa que as intersecções $\langle b,a\rangle$ e $\langle c,b\rangle$ percorrem a parte imaginária das mediatrizes (a tracejado) no sentido ascendente. Note-se também que, neste cenário, a parábola de um local a jusante no varrimento só pode intersectar a parte inferior da frente de onda, sendo a restante parte inacessível a novos arcos.

Resolvidos os casos degenerados, pode agora delinear-se as propriedades da frente de

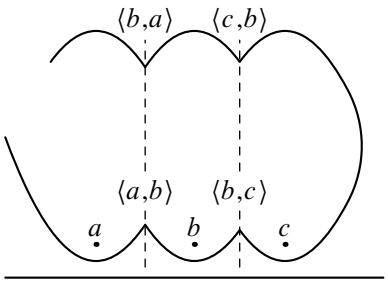


Figura 4.7: Normalização da intersecção de parábolas.

onda. A frente de onda é unívoca em x (cada ponto da linha de varrimento define um único círculo vazio máximo) e o varrimento é monótono em y (um círculo vazio máximo é examinado uma única vez). Ao longo do varrimento, a frente de onda muda continuamente de forma, registando-se a adição e remoção de arcos. Tal como atrás observado, as intersecções de arcos identificam todas as relações topológicas de um diagrama. Em particular, a existência de uma intersecção de (dois) arcos revela uma aresta (no diagrama) entre os locais que os geram. Desta forma, para a construção do diagrama, é suficiente registar as mudanças topológicas na frente de onda.

4.2.2 Eventos-local e eventos-círculo

A topologia da frente de onda altera-se em dois tipos de eventos.

O primeiro tipo de modificação topológica, denominado de *evento-local*, ocorre quando a linha de varrimento cruza um local (ver Figura 4.8). Neste evento, é criado um novo arco $\langle a \rangle$, gerado pelo novo local a. Pela definição de envelope inferior, o novo arco é necessariamente adicionado à frente de onda. O novo arco começa degenerado, descrevendo uma semi-recta vertical, que intersecta o arco da frente de onda imediatamente acima da posição do novo local. Simultaneamente, são criadas duas novas intersecções, $\langle b,a \rangle$ e $\langle a,b \rangle$. Inicialmente, as duas intersecções estão coincidentes. Com o avançar do varrimento estas afastam-se, à medida que percorrem a aresta entre o novo local e o local gerador do arco intersectado.

O segundo tipo de modificação topológica, denominada de *evento-círculo*, ocorre quando duas intersecções de arcos, $\langle a,b \rangle$ e $\langle b,c \rangle$, se juntam, ao mesmo tempo que um arco $\langle b \rangle$ desaparece da frente de onda (ver Figura 4.9). A junção de duas intersecções coincide com a identificação de um vértice, equidistante a três locais (a, b e c), dando início a uma nova intersecção de arcos $\langle a,c \rangle$ que percorre uma nova aresta do diagrama. Em suma, num evento-círculo é determinada a ligação entre três arestas, duas que terminam e uma que se inicia.

O algoritmo de Fortune constrói um diagrama de Voronoi processando uma sequência de eventos-local e eventos-círculo. Os eventos-local adicionam arcos e intersecções de arcos à frente de onda que, posteriormente, alargam, causando o desaparecimento de outros arcos, no decurso de eventos-círculo. Seja a *prioridade de um evento* dada pela ordenada de H que determina a adição ou remoção de um arco da frente de onda. A cada local, em número de n, corresponde um evento-local. São todos conhecidos à partida tendo a prioridade igual à

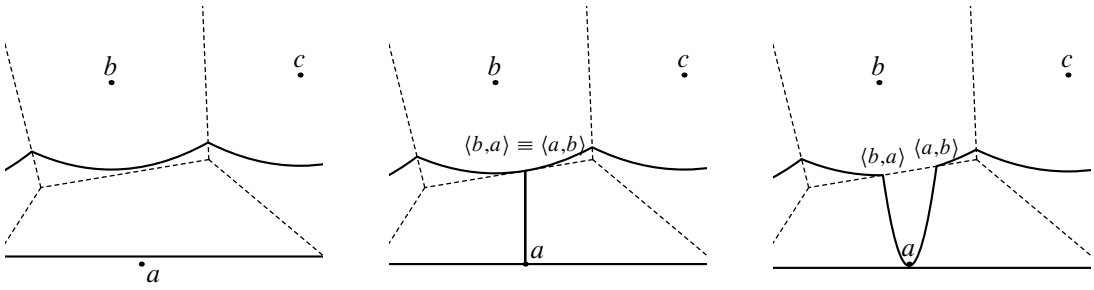


Figura 4.8: Ocorrência de um evento-local.

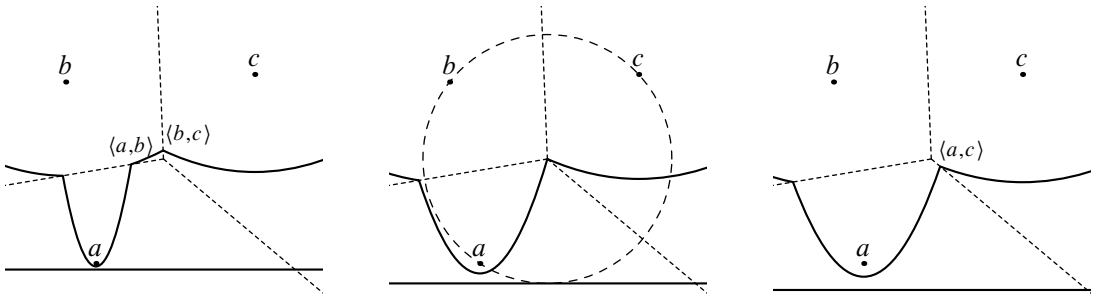


Figura 4.9: Ocorrência de um evento-círculo.

ordenada de cada local. A cada vértice, em número de $2n$ (aproximadamente), corresponde um evento-círculo. A prioridade de um evento-círculo é dada pela posição de H cuja distância ao vértice coincide com o raio do círculo circunscrito aos três locais que o definem. Ou seja, a prioridade de um evento-círculo é dada pela menor ordenada do círculo que caracteriza o evento.

As prioridades dos eventos determinam as posições da linha de varrimento que causam mudanças topológicas na frente de onda. Os eventos-local, atribuídos a locais, são todos escalonados no início do varrimento. Os eventos-círculo, associados aos arcos, não são conhecidos previamente, sendo escalonados à medida que a frente de onda é modificada. Desta forma, o varrimento contínuo do plano traduz-se por um processamento discreto de eventos por ordem decrescente de prioridade.

No caso geral, são escalonados mais eventos-círculo do que o número de vértices do diagrama. As condições que determinam o escalonamento de um evento-círculo podem não se manter até ao momento em que o evento seria processado, invalidando-o. No caso de um evento-local, o novo arco é inserido num arco da frente de onda, podendo invalidar um evento-círculo que lhe estivesse associado. No caso de um evento-círculo, é iniciada uma nova intersecção que invalida necessariamente os eventos-círculo que os arcos adjacentes possam ter. Nestes casos, os eventos invalidados, que se denominam por *falsos eventos*, são descartados, sendo necessário gerar novos eventos para os arcos cujas intersecções adjacentes se alteraram.

Os locais devem ser processados pela ordem induzida pela linha de varrimento, isto é, por ordem decrescente de ordenada. Para respeitar a normalização de intersecções de parábolas, também é necessário que locais de igual ordenada sejam processados por ordem crescente de abcissa.

Definição 4 (Ordem entre locais no varrimento linear)**.** *Sejam $a = (a_x, a_y)$ e $b = (b_x, b_y)$ dois locais. A relação de ordem entre locais é definida por:*

$$a > b \Leftrightarrow a_y < b_y \vee (a_y = b_y \wedge a_x > b_x). \tag{4.2}$$

4.2.3 Estruturas de dados

O algoritmo de varrimento é suportado por três estruturas de dados: uma árvore binária de pesquisa para guardar a frente de onda, uma fila com prioridade adaptável para guardar eventos-círculo e um vector ordenado com os locais.

Para o processamento de um evento-local há que encontrar o arco da frente de onda intersectado por uma linha vertical, para uma determinada posição da linha de varrimento. Esta operação pode ser executada eficientemente guardando a frente de onda numa árvore binária de pesquisa, ordenada por abcissas. Sendo a frente de onda uma sequência alternada de arcos e intersecções, tem-se que, na árvore, os arcos estarão guardados nos nós folha enquanto que as intersecções estarão guardadas nos nós interiores [20] (propriedade demonstrada no apêndice B). Deste modo, encontrar o arco intersectado por uma recta vertical equivale a operar uma pesquisa na árvore binária, que identifica um nó folha.

Para implementar esta operação há que comparar, em cada nó interno, a linha vertical que passa por um local $s = (s_x, s_y)$ com a intersecção $\langle a,b \rangle$ de dois arcos (gerados pelos locais a e b e pela linha de varrimento posicionada em s_y). Assumindo que $a > b$, tem-se que $\langle a,b \rangle_x \geq a_x$. Logo, apenas o caso em que $s_x > a_x$ não é trivial, sendo resolvido comparando a distância, segundo a vertical que passa por s, da linha de varrimento a cada parábola. Esta distância é dada pela função $\tau(a, s)$ (c.f. Eq. (4.1)) que, recorde-se, não é válida para arcos de parábola degenerados, isto é, quando $s_y = a_y \vee s_y = b_y$, o que obriga a um processamento adicional.

Definição 5 (Ordem na frente de onda no varrimento linear)**.** *A comparação de um local s com uma intersecção $\langle a,b \rangle$, tal que $s > a \wedge s > b$, depende da ordem dos locais que a definem.*

Para $a > b$, a comparação é dada por:

- *Se $s_y = a_y = b_y$, então ambas as parábolas são degeneradas e $\langle a,b \rangle$ percorre uma parte imaginária da mediatriz entre a e b, tendo-se $s_x < \langle a,b \rangle_x$;*

- *Se $s_y = a_y \neq b_y$, então apenas a parábola de a é degenerada e tem-se que $s_x > \langle a,b \rangle_x$ porque locais de igual ordenada estão ordenados por abcissa (i.e., $s_x > a_x$);*

- *Senão, nenhuma parábola é degenerada e a comparação é dada por:*

$$s_x \leq \langle a,b \rangle_x \Leftrightarrow s_x \leq a_x \vee \tau(a,s) \leq \tau(b,s). \tag{4.3}$$

Para $b > a$, a comparação é dada por:

- *Se $s_y = b_y$, então ou $a_y = b_y$ e ambas as parábolas são degeneradas e $\langle a,b \rangle$ percorre a parte real da mediatriz entre a e b, ou $a_y \neq b_y$ e $\langle a,b \rangle_x = b_x$. Em qualquer dos casos tem-se que $s_x > \langle a,b \rangle_x$ porque $s > b$;*

- *Senão, $s_y < b_y$ e a comparação é dada por:*

$$s_x \leq \langle a,b \rangle_x \Leftrightarrow s_x \leq b_x \wedge \tau(a,s) \leq \tau(b,s). \tag{4.4}$$

Em qualquer dos casos, a abcissa do maior dos locais define o semi-plano onde $\langle a,b \rangle_x$ reside (simplificando a comparação em 4.3 e 4.4). Por conveniência, denomina-se o maior local de uma intersecção por *local principal* da intersecção.

Os eventos são repartidos por duas estruturas de dados, consoante se trata de eventos-local ou eventos-círculo. Os dois tipos de eventos poderiam ser guardados numa só fila de eventos. No entanto, é vantajoso separá-los em duas estruturas de dados. A razão é puramente prática. Aos eventos-local é imposta uma ordem mais estrita (c.f. (4.2)) do que a dos eventos-círculo, cuja prioridade é somente definida por um número. Por outro lado, os eventos-local necessitam de ser todos ordenados, ao contrário dos eventos-círculo, em que alguns não chegam a ser processados. Logo os eventos-local podem ser determinados eficientemente percorrendo um vector com os locais, previamente ordenados.

Os eventos-círculo são gerados e processados (ou descartados) ao longo do varrimento, sendo guardados numa fila com prioridade adaptável [37]. Cada elemento da fila (um evento) guarda um apontador para um nó folha da árvore (o arco associado ao evento). A operação mais habitual será a de remover o elemento da fila com maior prioridade, que identifica o próximo evento-círculo a processar. No entanto, no caso de um falso evento, há que remover o elemento da fila associado a um nó folha da árvore. Esta operação é implementada com recurso a um vector auxiliar que identifica a posição na fila de um evento (variável ao longo do varrimento).

No final do varrimento, após esgotados os eventos-local e círculo, a frente de onda contém um arco por cada região ilimitada. Em particular, as intersecções de arcos identificam as arestas de comprimento infinito do diagrama. Resta apenas fechar cada uma destas regiões por adição de uma aresta auxiliar entre o local respectivo e um local auxiliar (p_∞). O fecho é feito de forma trivial ligando as arestas apontadas por nós internos consecutivos num percurso ordenado na árvore binária. Deste modo, todas as regiões são representadas por uma região fechada, sendo o local auxiliar p_∞ vizinho de todas as regiões ilimitadas. Como foi referido, esta uniformização das regiões é necessária para facilitar uma posterior edição do diagrama, seja pela inserção ou pela remoção de locais.

4.2.4 Cálculo de prioridades

O desenrolar do algoritmo depende de cálculos envolvendo as coordenadas de locais. Nos parágrafos que se seguem, sejam $a = (a_x, a_y)$, $b = (b_x, b_y)$ e $c = (c_x, c_y)$ três locais que geram três arcos consecutivos na frente de onda, sendo $\langle a,b \rangle$ e $\langle b,c \rangle$ as respectivas intersecções de arcos. Seja $s = (s_x, s_y)$ um local adicional, tal que $s_y > a_y \wedge s_y > b_y$.

A prioridade do evento-local de a é simplesmente:

$$\Pi_{\text{local}}(a) = a_y. \tag{4.5}$$

No processamento de um evento-local, há que fazer uma pesquisa na árvore binária que depende das relações de ordem indicadas nas expressões (4.3) e (4.4). Após simplificação, o termo não trivial, comum a ambas as expressões, fica reduzido a:

$$\tau(a,s) \leq \tau(b,s) \Leftrightarrow \vec{u} \cdot \vec{v} + v_x\, v_y\, (a_y - b_y) \leq 0 \text{ onde}$$
$$\begin{cases} \vec{u} = (\|a-s\|^2, \|b-s\|^2), \\ \vec{v} = (b_y - s_y, a_y - s_y). \end{cases} \tag{4.6}$$

Um evento-círculo é gerado para um arco cujas intersecções adjacentes convirjam (para um vértice). Ou seja, as intersecções adjacentes ao arco $\langle b \rangle$, $\langle a,b \rangle$ e $\langle b,c \rangle$, convergem para um vértice se o ângulo interno $\angle(a,b,c) < \pi$. Ou seja, um evento é gerado se:

$$\overrightarrow{a-b}^{\perp} \cdot \overrightarrow{c-b} < 0 \tag{4.7}$$

onde $\vec{v}^{\perp} = (-v_y, v_x)$ é o vector $\vec{v} = (v_x, v_y)$ rodado de $\pi/2$ no sentido directo. A prioridade de um evento-círculo é dada pela ordenada mínima do círculo circunscrito aos três locais a, b, c. Isto é:

$$\Pi_{\text{círculo}}(a,b,c) = b_y + v_y - \|\vec{v}\| \text{ onde}$$
$$\begin{cases} \vec{ab} = (a_x - b_x, a_y - b_y, \|a-b\|^2), \\ \vec{cb} = (c_x - b_x, c_y - b_y, \|c-b\|^2), \\ \vec{u} = \vec{ab} \times \vec{cb}, \\ \vec{v} = \vec{u}\,/(2\,u_z). \end{cases} \tag{4.8}$$

Da enumeração dos cálculos apresentados conclui-se que o algoritmo de Fortune calcula o diagrama de Voronoi com recurso apenas às quatro operações aritméticas ($+$, $-$, $\times$, $\div$) mais a raiz quadrada.

4.2.5 Algoritmo de varrimento linear

O algoritmo de Fortune está descrito no Algoritmo 1. A frente de onda é implementada com uma árvore vermelha-preta e a fila com prioridade com um *heap* binário e um vector

Algoritmo 1 Construção do diagrama de Voronoi V de um conjunto P de locais no plano, por varrimento linear.

```
 1: diagrama: V ← ∅                                   {O diagrama de Voronoi.}
 2: árvore: w ← ∅                          {Árvore vermelha-preta, guarda a frente de onda.}
 3: vector: a ← P.ordenar()  {Locais ordenados por ordenada (≥), e por abcissa (<).}
 4: inteiro: k ← 0                          {Posição em a do próximo evento-local.}
 5: fila: q ← ∅                    {Fila com prioridade, guarda os eventos-círculo.}
 6: número: α, β                                                    {Prioridades.}
 7: enquanto k < |P| ∨ q ≠ ∅ fazer
 8:     α ← (k < |P|) ? a[k].y : −∞
 9:     β ← (q ≠ ∅) ? q.prioridade_máxima() : −∞
10:     se α ≥ β então
11:         processar_evento_local(V, w, q, a[k])
12:         k++
13:     senão
14:         w_0 ← q.remover_máximo()                          {w_0 identifica o nó arco.}
15:         processar_evento_círculo(V, w, q, w_0)
16:     fim
17: fim
18: fechar_regiões_ilimitadas(V, w)
```

auxiliar. O locais são ordenados previamente e guardados num vector (linha 3). Os eventos são processados iterativamente (linhas 7–17). Inicialmente só existem eventos-local. O processamento de qualquer um dos eventos pode implicar o escalonamento de um ou dois eventos-círculo. O processamento de um evento-local (linha 11) começa por determinar o nó folha que identifica o arco da frente de onda verticalmente acima do novo local. Se este arco tem um evento-círculo associado, então é necessariamente um falso evento, que é removido da fila de eventos. Depois, o nó folha é substituído por uma sub-árvore com cinco nós, representando um troço da frente de onda formado por três arcos e duas intersecções de arcos. Dois destes arcos são gerados pelo mesmo local do arco do nó substituído. Para cada um destes arcos, é verificado se as intersecções adjacentes convergem entre si e, se for o caso, é escalonado um evento-círculo (que é inserido na fila) e associado ao nó da árvore respectivo. As intersecções adjacentes ao arco do novo local divergem entre si, pelo que este arco não pode gerar nenhum evento. Por fim, é adicionada uma nova aresta ao diagrama, comum ao novo local e ao local do nó substituído. Um evento-círculo identifica um nó folha associado a um arco que desaparece da frente de onda (linha 15). Percorrendo a árvore binária identificam-se os nós internos associados às intersecções do arco e os nós folha associados aos arcos adjacentes ao arco que desaparece. No processamento deste evento, os três nós do arco e intersecções adjacentes são substituídos por um único nó que representa a intersecção de arcos entre os arcos adjacentes (atrás referidos). Para cada um destes arcos (adjacentes à intersecção de arcos), é verificado se as intersecções de arcos convergem para um vértice e, se for o caso, é igualmente escalonado um evento-círculo. Finalmente, é adicionada uma nova

aresta ao diagrama, comum aos locais que definem a nova intersecção de arcos. A construção do diagrama de Voronoi termina quando se esgotam os eventos. Por fim, resta apenas fechar as regiões ilimitadas (linha 18).

Proposição 1. *O algoritmo de Fortune constrói o diagrama de Voronoi de n locais em tempo* $O(n \log n)$ *usando espaço* $\Theta(n)$.

Demonstração. A ordenação prévia dos eventos-local preenche um vector com n elementos. No que respeita à frente de onda, o primeiro evento-local inicia-a com um arco. Cada um dos restantes $n-1$ eventos-local adiciona dois arcos à frente de onda, o que limita o número de arcos a um máximo de $2n-1$. Com excepção do primeiro e último arcos, cada arco pode ter um evento associado, limitando a fila de eventos a um máximo de $2n-3$ elementos (para $n > 1$). Resumindo, as estruturas de dados usam um espaço linear, justificando a complexidade espacial. Consequentemente, cada operação primitiva na frente de onda ou na fila com prioridade tem um custo $O(\log n)$ (quando implementadas pelas estruturas de dados atrás indicadas). O processamento de um evento-local necessita de uma pesquisa e de quatro inserções na frente de onda e, no máximo, três operações na fila de eventos (remoção de um evento falso e inserção de dois novos eventos). O processamento de um evento-círculo implica duas operações na frente de onda (para remover um arco e uma intersecção) e, no máximo, quatro operações na fila de eventos (remoção de dois eventos falsos e inserção de dois novos eventos-círculo). Concluindo, o processamento de cada evento necessita de um número constante de operações primitivas e o número total de eventos é $O(n)$ (num total de n eventos-local e $2n-5$ eventos-círculo), justificando a complexidade temporal. □

4.3 Varrimento circular

O algoritmo de Fortune segue a estratégia de varrer o plano com uma linha recta. Infelizmente não é possível aplicar esta estratégia à superfície da esfera, cujo domínio limitado não permite a construção de parábolas como no plano. A solução para este obstáculo passa por redesenhar o algoritmo de Fortune numa forma equivalente que facilmente se adapte ao domínio esférico. De facto, essa versão existe na forma de um varrimento circular. Foi desenhada por Dehne e Klein [21] para a construção do diagrama de Voronoi na superfície de um cone, embora também se aplique ao plano.

Na versão planar, o plano é varrido por um círculo de raio crescente, centrado num ponto arbitrário, variando o raio desde zero até infinito. Essencialmente, o varrimento circular possui as mesmas propriedades do varrimento linear, construindo o diagrama de Voronoi de uma forma muito semelhante. A seguir descrevem-se as principais diferenças entre os dois tipos de estratégia, com especial relevo para as peculiaridades inerentes ao varrimento circular.

Figura 4.10: Elipse definida por um círculo e um local.

4.3.1 Elipses planares

O varrimento circular é centrado num ponto arbitrário. Porque simplifica a exposição, escolhe-se a origem para centro do varrimento. Seja H o círculo de varrimento centrado em $o = (0, 0)$ e de raio r. Seja $a = (a_\rho, a_\theta)$ um local, em coordenadas polares. Tal como no caso linear, um evento-local ocorre quando H cruza um local. Com o local contido no interior do círculo, interessa considerar o lugar geométrico dos pontos equidistantes de H e de a. Isto é, para um ponto qualquer s de H, interessa conhecer o ponto p tal que $\overline{p-a} = \overline{p-s}$. Uma vez que $\overline{o-p} + \overline{p-s} = r$, tem-se que $\overline{o-p} + \overline{p-a} = r$. Ou seja, os pontos equidistantes a a e a H formam uma elipse, com focos em o e a e eixo maior igual a r (ver Figura 4.10).

Sejam $s = (s_\rho, s_\theta)$ um ponto do círculo de varrimento H e $a = (a_\rho, a_\theta)$ um local tal que $s_\rho > a_\rho$. O ponto p da elipse gerada por a e H cujo ângulo é s_θ é $p = (\epsilon(a, s), s_\theta)$, onde $\epsilon(a, s)$, a distância de p à origem, é dada por:

$$\epsilon(a, s) = \frac{1}{2} \cdot \frac{s_\rho^2 - a_\rho^2}{s_\rho - a_\rho \cdot \cos(s_\theta - a_\theta)} \,. \tag{4.9}$$

A forma da elipse muda continuamente com o varrimento. Começa por ser um segmento de recta que liga os focos e depois converge para uma forma circular (ver Figura 4.10). Isto é, a excentricidade da elipse começa por ser igual a um, decrescendo com o avançar do varrimento, tendendo para zero à medida que o raio tende para infinito.

Tal como no varrimento linear, também aqui interessa processar os eventos-local por uma ordem estrita, definida pela seguinte relação de ordem entre locais.

Definição 6 (Ordem entre locais no varrimento circular)**.** *Sejam $a = (a_\rho, a_\theta)$ e $b = (b_\rho, b_\theta)$ dois locais. A relação de ordem entre locais é definida por:*

$$a > b \Leftrightarrow a_\rho > b_\rho \vee (a_\rho = b_\rho \wedge a_\theta > b_\theta) \,. \tag{4.10}$$

Por forma a estudar a frente de onda, vamos ver como duas elipses se intersectam. Seja b um segundo local, tal que $a > b$ (c.f. Figura 4.11). As duas elipses intersectam-se exactamente em dois pontos, $\langle a,b \rangle$ e $\langle b,a \rangle$, que estão equidistantes a ambos os locais. Ou seja, $\langle a,b \rangle$ e $\langle b,a \rangle$ pertencem à mediatriz do segmento de recta que une a a b. Uma vez que $a_\rho \geq b_\rho$, a mediatriz cruza necessariamente o segmento de recta $\overline{o\,a}$ (porque $\overline{o\,b} \leq \overline{o\,a}$) e as

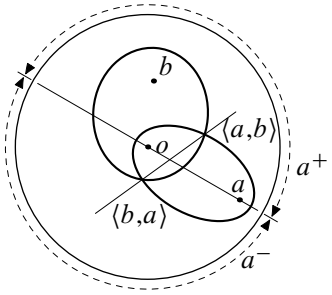


Figura 4.11: Intersecção de duas elipses no plano.

intersecções estão em semi-planos opostos definidos pela recta $o\,a$. Em resumo, $\langle a,b\rangle_\theta \in a^+$ e $\langle b,a\rangle_\theta \in a^-$, onde a^+ e a^- são os sectores de círculo definidos por:

$$a^+ = [a_\theta, a_\theta + \pi] \quad \text{e} \quad a^- = [a_\theta - \pi, a_\theta] \,. \tag{4.11}$$

Se $b > a$ obtém-se um resultado semelhante, onde $\langle a,b\rangle \in b^-$ e $\langle b,a\rangle \in b^+$.

Note-se que o modo como duas intersecções percorrem uma mediatriz no varrimento circular é, essencialmente, igual ao caso linear. O percurso inicia-se num ponto médio da mediatriz, com as duas intersecções coincidentes, que depois avançam em direcções opostas.

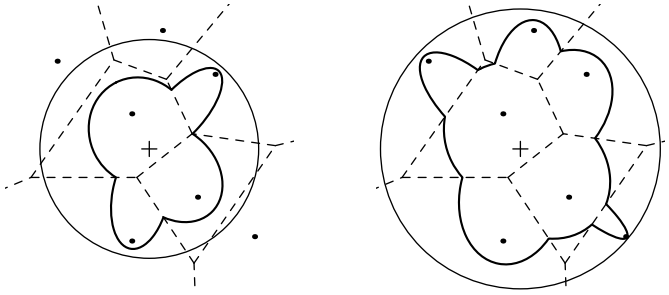
4.3.2 Frente de onda elíptica

A definição e as propriedades da frente de onda do varrimento circular são também semelhantes às do varrimento linear. Em primeiro lugar, a frente de onda é uma sequência de arcos de elipse formada pelo envelope exterior de todas as elipses associadas a um dado círculo de varrimento, tal como ilustrado na Figura 4.12. Em segundo lugar, durante o processo de varrimento, a frente de onda visita todos os pontos do plano uma única vez, percorrendo implicitamente todos os círculos vazios máximos do diagrama de Voronoi. Em particular, o desaparecimento de um arco da frente de onda identifica um vértice do diagrama, tal como no varrimento linear. Adicionalmente, o varrimento é monótono em ρ e a frente de onda é unívoca em θ. De resto, o varrimento circular segue a mesma estratégia que o varrimento linear; constrói um diagrama de Voronoi processando uma sequência de eventos-local e eventos-círculo (agora ordenados em ρ).

Os eventos-local e eventos-círculo sofrem apenas pequenas alterações com a transposição para o varrimento circular. Um evento-local ocorre sempre que o raio do círculo de varrimento iguala a distância do local à origem. A prioridade do evento-círculo é a distância à origem do ponto do círculo circunscrito a três locais que está mais longe da origem.

4.3.3 Linearização da frente de onda e eventos-rotação

Apesar das muitas semelhanças entre os dois métodos, a implementação da frente de onda apresenta um obstáculo: esta é agora uma linha fechada que, por ser cíclica em θ, não é



(a) (b)

Figura 4.12: Frente de onda elíptica no varrimento circular.

implementável através de uma árvore binária de pesquisa. Uma forma de solucionar este problema é fazer um *corte* na frente de onda pela parte positiva do eixo dos xx e representar o arco cortado em duplicado, uma vez no início e outra no fim da sequência de arcos e intersecções de arcos. O caso em que o semi-eixo cruza uma intersecção é resolvido cortando um dos arcos adjacentes. Desta forma, garante-se que a representação da frente de onda começa e acaba no mesmo arco, sendo facilmente implementável através de uma árvore binária de pesquisa. Para simplificar a exposição, de agora em diante o semi-eixo divisor será denominado simplesmente por eixo dos xx.

Consideremos, então, que a frente de onda é guardada numa árvore binária de pesquisa, ordenada em θ, descrevendo um domínio circular. Nesta configuração, uma intersecção de arcos $\langle a,b\rangle$ divide o domínio em dois sectores circulares: $[0, \langle a,b\rangle_\theta]$ e $]\langle a,b\rangle_\theta, 2\pi[$. A comparação de uma direcção arbitrária s_θ (que passa pelo local $s = (s_\rho, s_\theta)$) com $\langle a,b\rangle_\theta$ determina o sector ao qual s_θ pertence e é feita do seguinte modo. Quando $a_\rho \geq b_\rho$, sabe-se que $\langle a,b\rangle_\theta \in a^+$ e, em particular, é possível determinar se a^+ contém o eixo dos xx, o que implica analisar três casos distintos (ilustrados na Figura 4.13).

Definição 7 (Ordem na frente de onda no varrimento circular)**.** *A comparação de um local s com uma intersecção $\langle a,b\rangle$ depende da ordem dos locais que a definem. Analisemos primeiro o caso em que $a > b$.*

- *Se $a_\theta \leq \pi$, o que significa que o eixo dos $xx \notin a^+$, então s_θ pode pertencer a um de três sectores:*

$$[0, a_\theta], a^+, [a_\theta + \pi, 2\pi[.$$

  Neste caso, apenas a^+ coloca dificuldades porque, se s_θ pertence ao primeiro ou terceiro sector, então é menor ou maior que $\langle a,b\rangle_\theta$, respectivamente. O caso de a^+ pode ser convertido numa comparação de distâncias da origem a cada elipse, na direcção s_θ, ou seja:

$$s_\theta \leq \langle a,b\rangle_\theta \Leftrightarrow \epsilon(a,s) \geq \epsilon(b,s). \tag{4.12}$$

- *Quando $a_\theta > \pi$ então $xx \in a^+$, havendo que considerar duas situações.*

  - *Se $\langle a,b\rangle_\theta \in [a_\theta, 2\pi[$, então ou se tem $s_\theta \in [0, a_\theta]$, o que implica que $s_\theta \leq$*

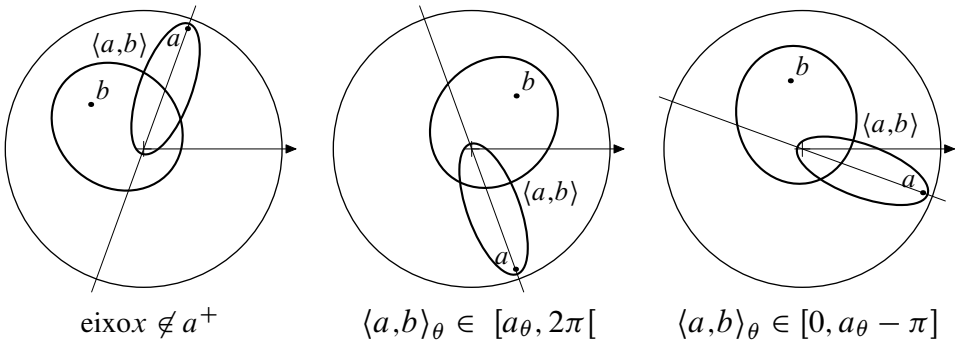


Figura 4.13: Ordenação na frente de onda elíptica: comparação com uma intersecção $\langle a,b\rangle_\theta$ de elipses no plano, (para $a_\rho \geq b_\rho$).

$\langle a,b\rangle_\theta$, *ou então* $s_\theta \in]a_\theta, 2\pi[$. *O último caso é igualmente resolvido pela equação da elipse:* $s_\theta \leq \langle a,b\rangle_\theta$ *se, e só se,* $\epsilon(a,s) \geq \epsilon(b,s)$.

– *No outro caso, quando* $\langle a,b\rangle_\theta \in [0, a_\theta - \pi]$, *se* $s_\theta \in [a_\theta - \pi, 2\pi[$, *então* $s_\theta \geq \langle a,b\rangle_\theta$; *senão,* $s_\theta \in [0, a_\theta - \pi[$, *e* $s_\theta \leq \langle a,b\rangle_\theta$ *se, e só se,* $\epsilon(a,s) \geq \epsilon(b,s)$.

O caso em que $a < b$ *origina um conjunto semelhante de regras.*

Tal como no varrimento linear, denomina-se por local principal de uma intersecção o maior dos dois locais. Note-se que no caso do varrimento circular não é necessário distinguir os casos em que $s_\rho = a_\rho$ ou $s_\rho = b_\rho$ (c.f. Definição 5). As intersecções entre duas elipses são sempre "reais", mesmo quando $a_\rho = b_\rho$, ao contrário do que acontece com a intersecção de parábolas no varrimento linear. Consequentemente, as regras de ordenação da frente de onda são bastante mais simples.

A complexidade das regras atrás descritas deve-se à complicação introduzida pela linearização da frente de onda. No entanto, esta complexidade é aparente. A correcta aplicação de regras é simplificada registando, ao longo do varrimento, a ordem relativa entre o ângulo de uma intersecção e o ângulo do local principal respectivo. A manutenção desta ordem depende da forma como é mantida a linearização da frente de onda quando muda o arco dividido pelo eixo dos xx.

No caso geral, é de esperar que o arco dividido pelo eixo dos xx se altere durante o varrimento. Esta mudança acontece quando uma aresta do diagrama de Voronoi (traçada pela intersecção de arcos correspondente) intersecta a parte positiva do eixo dos xx (ver Figura 4.14). Nesta situação, há que modificar a frente de onda de forma a transferir um arco e uma intersecção de um extremo da frente de onda para o extremo oposto. Este procedimento é incorporado no algoritmo de Fortune por adição de um terceiro tipo de evento, denominado por *evento-rotação*. A prioridade de um evento-rotação é determinada calculando o círculo centrado no eixo dos xx que é simultaneamente tangente aos dois locais da intersecção associada ao evento. Apenas o primeiro e o último arco da frente de onda linearizada podem ter um evento-rotação associado (caso as intersecções adjacentes convirjam para o eixo dos xx). Tal como os eventos-círculo, os eventos-rotação escalonados podem ser descartados antes de serem processados, por interposição de um novo arco (gerando um falso evento).

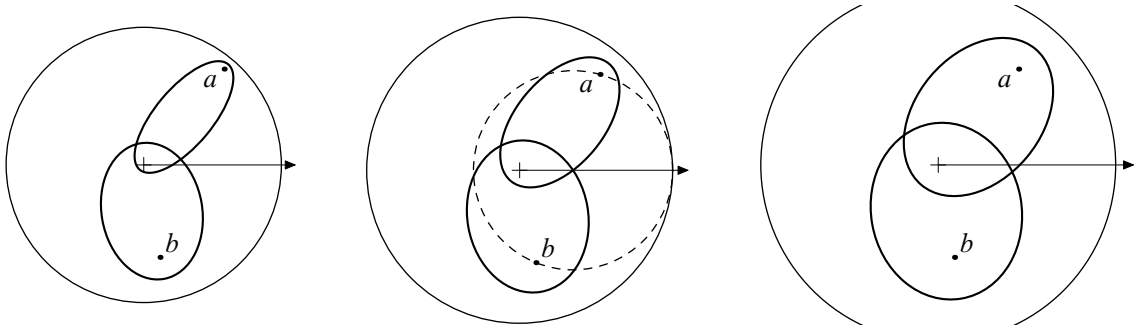


Figura 4.14: Ocorrência de um evento-rotação. Acontece quando muda o arco divido pelo eixo dos xx.

A ordem relativa entre uma intersecção e o local principal respectivo também é fácil de determinar. Seja a o local principal de uma intersecção $\langle a,b\rangle$. Desde o momento em que $\langle a,b\rangle$ é gerada, tem-se que $\langle a,b\rangle_\theta > a_\theta$ e, se $\langle a,b\rangle$ não intersecta o eixo dos xx, esta ordem não se altera durante o resto do varrimento. Mas se $\langle a,b\rangle$ for objecto de um evento-rotação, então a ordem relativa com o local principal inverte-se nesse instante, passando a ter-se $\langle a,b\rangle_\theta < a_\theta$. Além disso, esta inversão só pode acontecer uma vez no percurso de uma intersecção. Deste modo, é fácil determinar a ordem relativa entre uma intersecção e o local principal. Basta associar um atributo a cada intersecção de arcos, indicando a ordem relativa corrente, que apenas é actualizada no processamento de um evento-rotação. Com a adição deste atributo, fica trivial a determinação de pertença de uma intersecção a um sector (seja $[0, a_\theta - \pi]$ ou $[a_\theta, 2\pi[$ na aplicação das regras de comparação de uma intersecção com uma direcção arbitrária (c.f. Definição 7).

4.3.4 Cálculo de prioridades

A descrição do algoritmo de varrimento circular foi feita considerando uma representação dos locais em coordenadas polares. No entanto, uma representação Euclidiana é mais vantajosa, permitindo efectuar o cálculo de prioridades apenas com recurso às operações aritméticas habituais ($+$, $-$, $\times$, $\div$) mais a raiz quadrada.

Sejam $a = (a_x, a_y)$, $b = (b_x, b_y)$ e $c = (c_x, c_y)$ três locais. A relação de ordem entre dois locais a e b (que é trivial em coordenadas polares, c.f. Definição 6) é agora dada por:

$$a > b \Leftrightarrow \begin{cases} \|a\| > \|b\| \\ \|a\| = \|b\| \wedge \begin{cases} b_y > 0 \wedge (a_y < 0 \vee \vec{a}^{\perp} \cdot \vec{b} < 0) \\ \vee \\ b_y < 0 \wedge (a_y < 0 \wedge \vec{a}^{\perp} \cdot \vec{b} < 0) \\ \vee \\ b_y = 0 \wedge (a_y < 0 \vee b_x \geq 0) \end{cases} \end{cases} \tag{4.13}$$

A prioridade de um evento-local é dada pela distância do local à origem.

$$\Pi_{\text{local}}(a) = \|\vec{a}\| \tag{4.14}$$

A operação de pesquisa na árvore binária depende da relação de ordem indicada na expressão (4.12). Dado um local $s = (s_x, s_y) = (s_\rho, s_\theta)$, tal que $s > a \wedge s > b$, a relação de ordem entre a direcção s_θ e a direcção da intersecção $\langle a,b\rangle_\theta$, para um raio de varrimento de s_ρ, fica reduzida a (assumindo $a > b$):

$$\begin{gathered} s_\theta \leq \langle a,b\rangle_\theta \Leftrightarrow (1-\|\vec{u}\|^2)(1-\vec{u}\cdot\vec{w}) \leq (1-\|\vec{v}\|^2)(1-\vec{v}\cdot\vec{w}) \text{ onde} \\ \begin{cases} \vec{u} = \vec{a}/\|s\| \\ \vec{v} = \vec{b}/\|s\| \\ \vec{w} = \vec{s}/\|s\|. \end{cases} \end{gathered} \tag{4.15}$$

Um evento-círculo é gerado para um arco sempre que as intersecções de arcos adjacentes convirjam entre si, exactamente como no varrimento linear (ver inequação 4.7). A prioridade de um evento-círculo é dada por:

$$\begin{gathered} \Pi_{\text{círculo}}(a,b,c) = \|\vec{b} + \vec{v}\| + \|\vec{v}\| \text{ onde} \\ \begin{cases} \vec{ab} = (a_x - b_x, a_y - b_y, \|a-b\|^2), \\ \vec{cb} = (c_x - b_x, c_y - b_y, \|c-b\|^2), \\ \vec{u} = \vec{ab} \times \vec{bc}, \\ \vec{v} = \vec{u}\,/(2\,u_z). \end{cases} \end{gathered} \tag{4.16}$$

Por fim, um evento-rotação é gerado sempre que o caminho da primeira ou da última intersecção de arcos da frente de onda cruza a parte positiva do eixo dos xx. Seja $\langle a,b\rangle$ a primeira intersecção de arcos da frente de onda. É associado um evento-rotação ao arco $\langle a\rangle$ se $b > a \wedge b_y > 0 \wedge a_x < b_x$. A geração de um evento-rotação para o último arco da frente de onda produz um resultado idêntico. Em qualquer dos casos, a prioridade de um evento-rotação é dada por:

$$\begin{gathered} \Pi_{\text{rotação}}(a,b) = x + \|\vec{a} - (x,0)\| \text{ onde} \\ x = \left(\|b\|^2 - \|a\|^2\right) / \left(2(b_x - a_x)\right). \end{gathered} \tag{4.17}$$

4.3.5 Algoritmo de varrimento circular

O algoritmo de varrimento circular está resumido no Algoritmo 2. Apenas difere do Algoritmo 1 no cálculo de prioridades e pela presença de eventos-rotação. Os eventos-local são guardados num vector ordenado de locais (linha 3). Na fila de eventos (linha 5) são guardados os eventos-círculo e os eventos-rotação (estes últimos em número máximo de dois).

Algoritmo 2 Construção do diagrama de Voronoi V de um conjunto P de locais no plano, por varrimento circular.

```
 1: diagrama: V ← ∅                                   {O diagrama de Voronoi.}
 2: árvore: w ← ∅                                     {Árvore preta-vermelha, guarda a frente de onda.}
 3: vector: a ← P.ordenar()                           {Locais ordenados por módulo (≤), e por ângulo (<).}
 4: inteiro: k ← 0                                    {Posição em a do próximo evento-local.}
 5: fila: q ← ∅                                       {Fila com prioridade, para eventos-círculo e rotação.}
 6: número: α, β                                      {Prioridades.}
 7: enquanto k < |P| ∨ q ≠ ∅ fazer
 8:    α ← (k < |P|) ? a[k].ρ : +∞
 9:    β ← (q ≠ ∅) ? q.prioridade_mínima() : +∞
10:    se α ≤ β então
11:       processar_evento_local(V, w, q, a[k])
12:       k++
13:    senão                                         {w0 identifica o nó arco}
14:       (w0, ev) ← q.remover_mínimo()              {ev indica o tipo de evento}
15:       se ev é evento-círculo então
16:          processar_evento_círculo(V, w, q, w0)
17:       senão
18:          processar_evento_rotação(V, w, q, w0)
19:       fim
20:    fim
21: fim
22: fechar_regiões_ilimitadas(V, w)
```

Um evento na fila tem a prioridade dada por um número real (que é um raio do círculo de varrimento) e dois atributos associados: um ponteiro para um arco da árvore binária e um identificador de tipo de evento (círculo ou rotação). Ao se retirar um evento da fila (linha 14), o tipo de evento determina o processamento a operar. No entanto, para que se possa usar a mesma disciplina na fila de eventos (comparando apenas uma prioridade), pode acontecer que seja processado um evento-círculo para o arco dividido; caso possível se existirem um ou dois eventos-rotação escalonados para a mesma prioridade que o evento-círculo do arco dividido. Neste caso, há apenas que forçar o processamento de um dos eventos-rotação (escolhendo o de menor prioridade, se existirem dois) antes de continuar com o normal processamento do evento-círculo (linha 16). No processamento de um evento-rotação (linha 18), há apenas que transferir um arco e uma intersecção de arcos de um extremo da árvore binária para o extremo oposto e actualizar a ordem relativa da intersecção de arcos transferida. Note-se que as modificações topológicas na frente de onda apenas ocorrem nos eventos-local e eventos-círculo. Os eventos-rotação têm a única tarefa de sincronizar a árvore binária com a alteração que ocorre na frente de onda. O varrimento termina com o escoar da fila de eventos, com um arco da frente de onda a percorrer cada região de área infinita. Como no Algoritmo 1, estas regiões são fechadas por adição de arestas auxiliares (linha 22).

Proposição 2. *O algoritmo de varrimento circular constrói o diagrama de Voronoi de n locais em tempo* $O(n \log n)$ *usando espaço* $\Theta(n)$.

Demonstração. A análise da complexidade segue o mesmo raciocínio utilizado na Proposição 1, diferindo apenas no que respeita ao escalonamento e processamento de eventos-rotação. No máximo são escalonados e processados $O(n)$ eventos-rotação, limitados ao número de arestas intersectadas pelo eixo dos xx. Logo, o espaço ocupado pelas estruturas de dados é igualmente $\Theta(n)$, com um custo $O(\log n)$ por operação primitiva. De resto, observa-se que o processamento de cada evento-rotação necessita de um número constante de operações nas estruturas de dados. Concluindo, o varrimento circular usa as mesmas estruturas de dados, ocupando basicamente o mesmo espaço e, logo, com o mesmo custo assimptótico por operação primitiva. □

Comparando as duas estratégias, o varrimento circular tem um custo computacional mais elevado que a alternativa linear. Duas razões justificam este facto. Primeiro, a relação de ordem usada na frente de onda do varrimento circular tem um maior custo computacional. Segundo, o varrimento circular necessita de escalonar, descartar e processar eventos-rotação.

O varrimento circular tem a vantagem de permitir a construção parcial de um diagrama de Voronoi, em torno de um dado ponto [21]. Esta possibilidade é útil, por exemplo, para calcular uma medida local de um conjunto de dados, parando o processo de varrimento quando é conhecida uma parte significativa do diagrama. Note-se que não há restrições na escolha do centro de varrimento, podendo este ser posicionado até fora do invólucro-convexo do conjunto de locais.

Como foi referido no início desta secção, o principal interesse na estratégia de varrimento circular é a capacidade de adaptação ao domínio esférico, como se verá no próximo capítulo.

Capítulo 5

Construção do diagrama esférico

Este capítulo é dedicado ao estudo do algoritmo que constrói o diagrama de Voronoi de pontos na superfície da esfera. A construção é feita aplicando a técnica de varrimento, adaptando à esfera o algoritmo de varrimento circular do plano. No plano, o varrimento faz-se aumentando o raio do círculo de varrimento de zero até infinito. Na prática, o círculo é aumentado até um raio arbitrariamente grande, o suficiente para esgotar a fila de eventos. Na esfera, aplica-se a mesma estratégia. A superfície da esfera é varrida por um círculo de raio crescente, centrado num ponto arbitrário, a partir do qual diverge. Dado que a esfera forma um domínio compacto, o círculo de raio crescente degenera num ponto quando o raio iguala π. Nesse instante, o círculo de varrimento completa o varrimento da esfera. Porém, o varrimento de 0 a π não é suficiente para que a frente de onda, sempre *atrasada* em relação à linha de varrimento, complete o varrimento da esfera. Para que isso aconteça, há que prosseguir com o varrimento, aumentando o círculo de varrimento para um raio superior a π. Nestas condições, a forma do círculo de varrimento aparenta fazer um segundo varrimento da superfície da esfera, agora convergindo para o centro de varrimento. É este duplo varrimento que permite que a frente de onda percorra toda a superfície da esfera e que o diagrama de Voronoi esférico seja construído.

O varrimento tem início numa posição arbitrária designada por *centro do varrimento*. O raio do círculo de varrimento, que é o comprimento de um arco geodésico, começa em zero e aumenta à medida que o varrimento avança. Mais uma vez, quando um local é atravessado pelo círculo de varrimento, interessa considerar o lugar geométrico dos pontos equidistantes ao local e ao círculo, que forma também uma elipse (esférica). De igual modo, o envelope exterior de todas as elipses constitui a frente de onda. É uma curva fechada, composta por arcos de elipse cujas intersecções traçam as arestas do diagrama de Voronoi esférico. Neste caso, dado que a esfera é um domínio fechado, a frente de onda desaparece quando o varrimento termina. A versão esférica da frente de onda mantém as mesmas propriedades que a frente de onda planar. Mais precisamente, cada arco varre o interior da região do local que a gera e as intersecções de arcos percorrem as arestas do diagrama de Voronoi.

Por ser conveniente, será usada a terminologia usual em geografia para designar partes da esfera, nomeadamente, latitude, longitude, pólo Norte, pólo Sul, paralelo de latitude, meridiano de longitude, meridiano principal e antípoda de um local. Em geral, um ponto p na

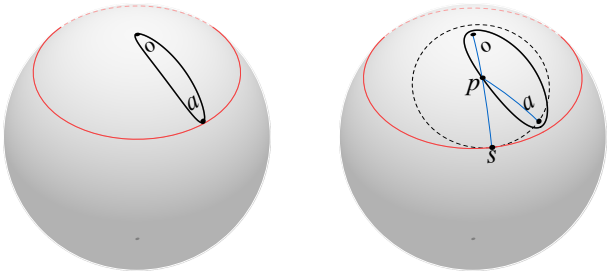


Figura 5.1: Elipse esférica definida por um círculo e um local.

esfera é definido pelo par $p = (p_\rho, p_\lambda)$, em que p_ρ é a co-latitude e p_λ é a longitude. Quando conveniente, um ponto também será designado pelo vector unitário em 3D $\vec{p} = (p_x, p_y, p_z)$. Sem perda de generalidade, e porque simplifica a exposição do algoritmo, assume-se que o varrimento é centrado no pólo Norte; isto é, o centro do círculo de varrimento é $o = (0, 0)$. Deste modo, o círculo de varrimento varre paralelos de latitude e o raio do círculo é igual à co-latitude do paralelo.

Os locais são processados por ordem crescente de co-latitude. Mas, tal como no algoritmo de Fortune, há que impor uma ordem total nos locais para assegurar a correcta ordem dos arcos na frente de onda.

Definição 8 (Ordem entre locais no varrimento esférico)**.** *Sejam a e b dois locais na superfície da esfera. A relação de ordem entre locais, é dada por:*

$$a < b \quad \Leftrightarrow \quad a_\rho < b_\rho \vee (a_\rho = b_\rho \wedge a_\lambda < b_\lambda). \tag{5.1}$$

5.1 Elipses esféricas

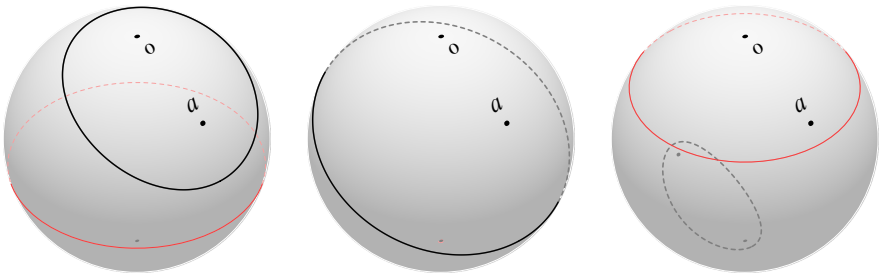
A construção do diagrama de Voronoi esférico é obtida por intermédio de uma frente de onda esférica. Dado que a frente de onda é formada por arcos de elipse, são primeiro analisadas as propriedades da elipse gerada por um local e pelo círculo de varrimento e o modo como a elipse varre a esfera. Esta construção define a frente de onda mais simples, formada por uma única elipse.

Seja $a = (a_\rho, a_\lambda)$ um local na esfera, que não o pólo Sul. Seja H o círculo de varrimento, centrado em $o = (0, 0)$ e de raio r. Tal como no caso planar, também aqui interessa considerar o lugar dos pontos equidistantes a a e a H. Ou seja, para um ponto qualquer s de H, interessa conhecer o ponto p tal que $\overline{p-s} = \overline{p-a}$. Mas $\overline{p-o} + \overline{p-s} = r$. Logo, tem-se que:

$$\overline{p-o} + \overline{p-a} = r, \tag{5.2}$$

o que significa que o lugar dos pontos equidistantes a a e a H define uma elipse esférica, tendo o e a como focos e r como eixo maior (ver Figura 5.1).

Sejam $s = (s_\rho, s_\lambda)$ um ponto do círculo de varrimento H e $a = (a_\rho, a_\lambda)$ um local



(a) (b) (c)

Figura 5.2: Varrimento da esfera por uma elipse, variando o raio do círculo de varrimento: (a) para $r < \pi$ com a elipse a aumentar de tamanho, (b) para $r = \pi$ em que elipse degenera num círculo máximo e (c) para $r > \pi$ com a elipse a convergir para os antípodas dos focos.

tal que $s_\rho \in]a_\rho, 2\pi - a_\rho[$. O ponto p da elipse gerada por a e H cuja longitude é s_λ é $p = (\mathcal{E}(a,s), s_\lambda)$, onde $\mathcal{E}(a,s)$, a distância de o a p, é dada por (com a função arctan normalizada para um valor positivo):

$$\mathcal{E}(a,s) = \operatorname{arctg}\left(-\frac{\cos(s_\rho) - \cos(a_\rho)}{\operatorname{sen}(s_\rho) - \operatorname{sen}(a_\rho)\cdot\cos(a_\lambda - s_\lambda)}\right). \tag{5.3}$$

Tal como as elipses no plano, as elipses esféricas são curvas fechadas. Têm uma forma delimitada por um sector esférico, cujo ângulo diedro λ é dado por:

$$\lambda = 2\arccos(\sec(a_\rho/2)\cos(r/2)). \tag{5.4}$$

Com $r = a_\rho$, $\lambda = 0$ e a elipse é o arco geodésico que une o a a. À medida que r aumenta, λ também aumenta, correspondendo ao alargar da forma da elipse (ver Figura 5.2a). Quando $r = \pi$, $\lambda = \pi$ e a elipse degenera num círculo máximo, que é a mediatriz do arco geodésico que liga a ao pólo Sul, antípoda de o (ver Figura 5.2b). Aumentando r ainda mais torna λ maior que π. Quando $r = 2\pi - a_\rho$, $\lambda = 2\pi$, que significa que a elipse degenera mais uma vez, agora no arco geodésico que une os antípodas dos focos (ver Figura 5.2c).

Se a se localiza no pólo Sul, então os valores válidos para r restringem-se ao intervalo singular $\{\pi\}$, indicando que a esfera é varrida completamente com um único raio de varrimento. De facto, neste caso especial, qualquer ponto da esfera satisfaz (5.2). Logo, pode considerar-se que a esfera é varrida toda em simultâneo, de uma forma abrupta. No entanto, como se verá mais adiante, a peculiaridade deste varrimento não levanta nenhum obstáculo ao algoritmo.

Concluindo, uma elipse começa degenerada num arco que liga os focos, alarga até degenerar num círculo máximo e depois estreita até terminar num arco que liga os antípodas dos focos. Por este processo, qualquer elipse varre efectivamente toda a esfera quando o raio do círculo varia de a_ρ até $2\pi - a_\rho$.

O processo de varrimento tem uma interpretação geométrica interessante, relacionada com os círculos vazios máximos (ilustrado na Figura 5.3). Como foi anteriormente referido

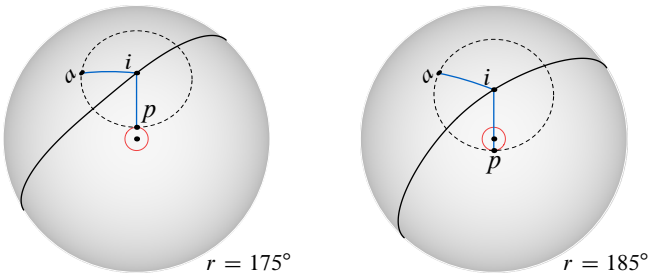


Figura 5.3: Cruzamento do pólo Sul num varrimento circular. A linha a cheio representa a parte visível da elipse. A linha a tracejado é o círculo vazio máximo para um ponto da elipse.

(recorde-se a Figura 5.1), o círculo vazio máximo centrado num ponto p da elipse é tangente a um local a e a um ponto s do círculo de varrimento (de raio r). Se $r < \pi$, o círculo vazio máximo é tangente à parte superior do círculo de varrimento (do lado das co-latitudes menores) e não contém o pólo Sul. Se $r > \pi$, o círculo de varrimento é tangente à parte inferior do círculo de varrimento (do lado das co-latitudes maiores) e contém o pólo Sul. Quando $r = \pi$, o círculo de varrimento degenera num ponto, o pólo Sul, ao qual o círculo vazio máximo é tangente.

O próximo objectivo é analisar como a intersecção de duas elipses percorre a mediatriz gerada por dois locais. Sem perda de generalidade, sejam a e b dois locais distintos tais que $a > b$, com o raio do círculo de varrimento $r \in [a_\rho, 2\pi - a_\rho]$ (como ilustrado na Figura 5.4). As elipses geradas por a e b intersectam-se exactamente em dois pontos, designados por $\langle a,b \rangle$ e $\langle b,a \rangle$, pertencentes à mediatriz definida pelos dois locais. As duas intersecções estão em lados opostos do círculo máximo coincidente com o meridiano a_λ, uma vez que a mediatriz tem a forma de um círculo máximo que intersecta o arco geodésico que une o a a (porque o está mais próximo de b que de a). Em resumo, $\langle a,b \rangle_\lambda \in a^+$ e $\langle b,a \rangle_\lambda \in a^-$, onde:

$$a^+ = [a_\lambda, a_\lambda + \pi] \quad \text{e} \quad a^- = [a_\lambda - \pi, a_\lambda] \,. \tag{5.5}$$

As duas intersecções começam por estar coincidentes, quando $r = a_\rho$. Depois, traçam um percurso em direcções opostas: $\langle a,b \rangle_\lambda$ cresce monotonamente em λ, enquanto que $\langle b,a \rangle_\lambda$ decresce monotonamente em λ (ignorando por agora a descontinuidade na passagem pelo meridiano principal). A separação entre as duas intersecções é máxima quando $r = \pi$ (com as elipses degeneradas em círculos máximos), após o qual voltam a aproximar-se. Quando $r = 2\pi - a_\lambda$, as duas intersecções coincidem de novo, completando o percurso de toda a mediatriz $\mathcal{M}_{ab}$. Observe-se que, se $a_\rho = b_\rho$, $\mathcal{M}_{ab}$ é composta por dois meridianos, com $\langle a,b \rangle_\lambda$ e $\langle b,a \rangle_\lambda$ constantes ao longo do varrimento.

Recorde-se que, para o caso especial em que a se localiza no pólo Sul, a "elipse" gerada por a é toda a esfera. Logo, neste caso, a intersecção entre as duas elipses não é mais que a elipse gerada por b, que coincide, necessariamente, com $\mathcal{M}_{ab}$ (porque o raio do círculo de varrimento é π).

O caso em que $b > a$ produz um resultado semelhante, com $\langle a,b \rangle_\lambda \in b^-$ e $\langle b,a \rangle_\lambda \in b^+$.

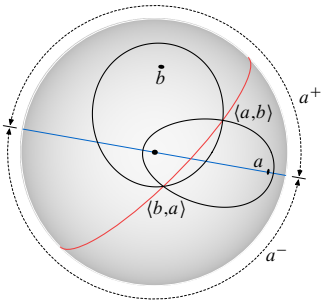


Figura 5.4: Intersecção de duas elipses na esfera.

Neste domínio, recorre-se a duas definições adicionais relativas a intersecções de arcos. Nomeadamente, define-se por *caminho de uma intersecção* o lugar dos pontos percorridos por uma intersecção e define-se por *local principal* de uma intersecção o maior dos locais (pela ordem definida por (5.1)). Vale a pena notar que, apesar de $\langle a,b\rangle$ e $\langle b,a\rangle$ serem definidos por intersecções de elipses, também podem ser vistos como a intersecção de uma elipse com a mediatriz de dois locais.

5.2 Varrimento esférico

O algoritmo de varrimento esférico não difere em muito do algoritmo de varrimento circular do plano. A frente de onda esférica é definida pelo envelope exterior das elipses geradas pelos locais já visitados pela linha de varrimento. A curva resultante é uma linha fechada, unívoca em λ, e composta por uma sequência alternada de arcos de elipse e intersecções de arcos. Mas, mais importante, a frente de onda varre implicitamente os círculos vazios máximos de todos os pontos da esfera e, por este motivo, permite a construção do diagrama de Voronoi esférico.

5.2.1 Frente de onda

A frente de onda tem início quando o círculo de varrimento cruza o local mais a Norte, começando degenerada num arco de círculo máximo. Depois, o processo de varrimento continua em direcção ao Sul, cruzando todos os locais até que o círculo de varrimento atinge o pólo Sul. Com o círculo de varrimento no pólo Sul, a frente de onda é formada pelo envelope exterior de círculos máximos. Prosseguindo com o varrimento, agora de Sul para Norte, permite-se que a frente de onda percorra o resto da esfera. O varrimento esférico termina quando o círculo de varrimento cruza o local mais a Sul pela segunda vez. Como veremos mais à frente (na Proposição 3), isto acontece quando a frente de onda fica reduzida a dois arcos.

A construção de um diagrama de apenas três locais é suficiente para ilustrar o algoritmo de varrimento. O resultado é um diagrama formado por três arestas e dois vértices (correspondendo aos centros dos dois círculos circunscritos a três pontos). Sejam a, b e c três locais tais que $a < b < c$. Os primeiros dois eventos-local (dos locais a e b) geram dois arcos

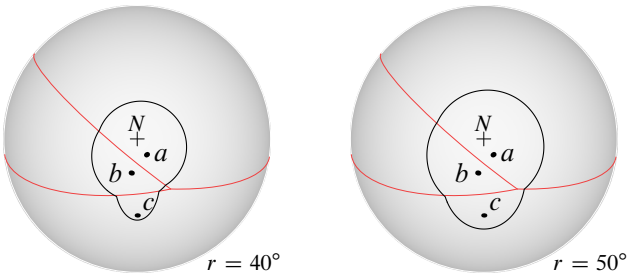


Figura 5.5: Frente de onda elíptica para um conjunto de três locais, antes e depois do primeiro evento-círculo. N indica o pólo Norte.

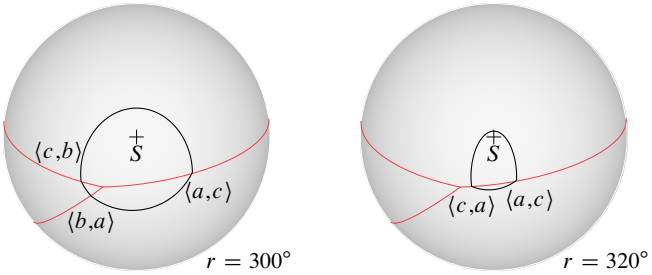


Figura 5.6: Fecho da frente de onda para o mesmo conjunto de três locais, antes e depois do último evento-círculo. S indica o pólo Sul.

na frente de onda, cujas intersecções $\langle a,b\rangle$ e $\langle b,a\rangle$ varrem a mesma aresta (ver Figura 5.5). O terceiro evento-local (do local c) adiciona mais dois arcos (e o início de uma nova aresta, entre c e b) e despoleta o escalonamento de dois eventos-círculo: o evento-círculo $\langle c,a,b\rangle$, escalonado para $r = 45°$, e o evento-círculo $\langle a,b,c\rangle$, escalonado para $r = 314°$ (este já na fase do varrimento de Sul para Norte). O processamento do primeiro evento-círculo gera um vértice e uma nova intersecção de arcos $\langle c,a\rangle$ que varre uma nova aresta. Durante o resto do varrimento da esfera de Norte para Sul não se registam mais alterações. Após cruzar o pólo Sul, o varrimento prossegue de Sul para Norte, altura em que é processado o segundo evento-círculo, que adiciona o segundo vértice, dando início a um segundo varrimento da aresta entre c e a, agora pela intersecção de arcos $\langle a,c\rangle$ (ver Figura 5.6). As duas intersecções de arcos que sobram continuam a varrer a mesma aresta, juntando-se quando o círculo de varrimento *revisita* o primeiro local depois do pólo Sul (local c, neste caso) e o arco $\langle c\rangle$ degenera num arco de círculo máximo, terminando o varrimento da esfera.

Para se provar que, em qualquer cenário, a frente de onda termina com dois arcos e duas intersecções, convém recordar uma propriedade dos diagramas de Voronoi esféricos, demonstrada na secção 2.2. Um diagrama de Voronoi esférico de n locais (com $n \geq 3$) tem $2n-4$ vértices e $3n-6$ arestas, desde que cada vértice tenha grau três.

Proposição 3. *Para $n \geq 2$ locais, a frente de onda termina com exactamente dois arcos (e duas intersecções).*

Demonstração. Os dois primeiros eventos-local conduzem a exactamente dois arcos e duas intersecções. Se $n = 2$, não são gerados mais arcos e as duas intersecções percorrem a

mediatriz dos dois locais, juntando-se de novo quando o varrimento esférico termina. Para $n > 2$, cada evento-local subsequente adiciona dois arcos à frente de onda, enquanto que cada evento-círculo subtrai um arco. Logo, os arcos adicionados pelos $n-2$ eventos-local são contrabalançados pelos arcos subtraídos pelos $2n-4$ eventos-círculo, restando dois arcos na frente de onda que termina. □

As duas intersecções restantes da frente de onda têm de percorrer a mesma aresta do diagrama de Voronoi. Caso contrário, deveria faltar pelo menos um vértice (e um evento-círculo). Esta aresta, que é percorrida em duplicado, foi também adicionada em duplicado ao diagrama. Logo, o diagrama é completado descartando uma das arestas e ligando adequadamente a outra aresta.

Para a implementação do algoritmo são igualmente necessárias três estruturas de dados: uma para guardar os locais ordenados, uma para guardar a frente de onda e uma para guardar a fila de eventos.

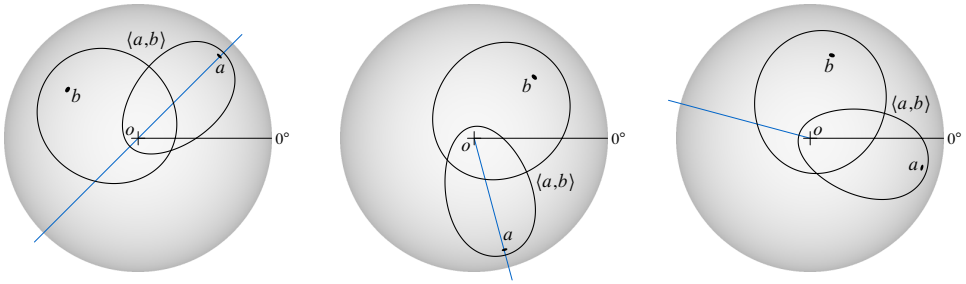
5.2.2 Eventos-rotação

A frente de onda esférica é guardada com a solução desenhada para o varrimento circular planar. Agora o meridiano principal desempenha o mesmo papel que o eixo dos xx desempenha no plano. Divide um arco em duas partes, linearizando a frente de onda, que é guardada numa árvore binária de pesquisa, ordenada em λ. O arco cortado pelo meridiano principal deve ser representado duas vezes, como primeiro e último arco da frente de onda. De igual modo, há que actualizar a árvore binária (via evento-rotação) sempre que uma intersecção cruza o meridiano principal, passando o meridiano principal a dividir um novo arco.

A condição que determina a ocorrência de um evento-rotação é essencialmente a mesma: há que escalonar um evento rotação se o caminho da primeira ou última intersecção da frente de onda intersecta o meridiano principal. Porém, há que notar que na esfera, o caminho de uma intersecção percorre metade de um círculo máximo, o que simplifica as condições para a ocorrência de eventos. Seja $\langle a,b \rangle$ uma intersecção de arcos da frente de onda. Um evento rotação é associado a um dos arcos adjacentes, $\langle a \rangle$ ou $\langle b \rangle$, se:

- $\langle a \rangle$ é o primeiro arco da frente de onda e $b_\rho > a_\rho \wedge b_\lambda < \pi$;
- $\langle b \rangle$ é o último arco da frente de onda e $a_\rho > b_\rho \wedge a_\lambda > \pi$.

A prioridade de um evento-rotação é determinada calculando o círculo que, centrado no meridiano principal, circunscreve os dois locais correspondentes. Observe-se que só há lugar a evento-rotação se uma intersecção cruza efectivamente o meridiano principal. No caso particular em que $a_\rho = b_\rho$ a intersecção percorre um meridiano completo (que pode até coincidir com o meridiano principal), mas não cruza o meridiano principal, não originando nenhum evento-rotação.



meridiano principal $\notin a^+$ $\quad \langle a,b\rangle_\lambda \in [a_\lambda, 2\pi[$ $\quad \langle a,b\rangle_\lambda \in [0, a_\lambda - \pi]$

Figura 5.7: Ordenação na frente de onda elíptica: comparação com uma intersecção $\langle a,b\rangle_\lambda$ de elipses esféricas (para $a > b$).

5.2.3 Ordenação da frente de onda

Vamos agora definir a relação de ordem entre duas longitudes, uma especificada por um local s (que não no pólo Sul) e outra definida pela intersecção de arcos $\langle a,b\rangle$, quando o raio do círculo de varrimento $r = s_\rho$. Esta relação é usada na árvore binária de pesquisa que implementa a frente de onda para encontrar o arco directamente acima de s.

Definição 9. *A comparação de um local s (onde $s_\rho < \pi$) com uma intersecção $\langle a,b\rangle$, tal que $s > a \wedge s > b$, depende da ordem dos locais que a definem.*

Quando $a_\rho \geq b_\rho$, $\langle a,b\rangle_\lambda \in a^+$, podendo ocorrer um de dois casos.

- *Se $s_\rho = a_\rho$, então $\langle a,b\rangle_\lambda = \langle b,a\rangle_\lambda = a_\lambda$, uma vez que os locais de igual latitude são ordenados por longitude, e tem-se $s_\lambda > \langle a,b\rangle_\lambda$.*

- *Senão, tem-se que $s_\rho > a_\rho$, e a elipse gerada por a não é um arco geodésico. Nesta configuração pode acontecer um de três casos (ilustrados na Figura 5.7):*

  - *Se o meridiano principal $\notin a^+$, a esfera pode ser dividida em três sectores: $[0, a_\lambda[$, a^+, e $]a_\lambda + \pi, 2\pi[$. Tal como no caso planar, apenas a^+ causa dificuldades, uma vez que, se s_λ pertence ao primeiro ou terceiro sector, então este é menor ou maior que $\langle a,b\rangle_\lambda$, respectivamente. A comparação com a^+ faz uso de (5.3):*

$$s_\lambda < \langle a,b\rangle_\lambda \Leftrightarrow \mathcal{E}(a,s) > \mathcal{E}(b,s)\ . \tag{5.6}$$

  - *Se a^+ contém o meridiano principal e $\langle a,b\rangle_\lambda \in [a_\lambda, 2\pi[$, então ou $s_\lambda \in [0, a_\lambda[$, que implica que $s_\lambda < \langle a,b\rangle_\lambda$, ou $s_\lambda \in [a_\lambda, 2\pi[$. O último caso é resolvido por aplicação de (5.6).*

  - *Por último, temos o caso em que a^+ contém o meridiano principal e $\langle a,b\rangle_\lambda \in [0, a_\lambda - \pi]$. Se $s_\lambda \in]a_\lambda - \pi, 2\pi[$ então $s_\lambda > \langle a,b\rangle_\lambda$; caso contrário, $s_\lambda \in [0, a_\lambda - \pi]$ que é resolvido por (5.6).*

O caso em que $b_\rho > a_\rho$ é resolvido de um modo semelhante.

Para um local situado no pólo Sul, observe-se que qualquer arco da frente de onda lhe está directamente acima. Logo, quando se processa o evento-local correspondente, e por forma a evitar qualquer indeterminação na definição anterior, é suficiente escolher um arco arbitrário para ser intersectado pelo arco do novo local. De facto, como neste caso o círculo de varrimento degenera num ponto, o pólo Sul, a frente de onda coincide com a região de Voronoi de s. Como consequência, o evento-local despoleta uma cascata de eventos-círculo (um para cada vértice de $V(s)$), todos escalonados para a co-latitude π.

De certo modo, há uma inversão no papel do círculo de varrimento quando este cruza o pólo Sul. Quando a esfera é varrida de Norte para Sul, são visitados todos os locais. Depois, no varrimento de Sul para Norte, apenas são processados eventos-círculo e eventos-rotação. Este duplo varrimento é necessário para que as elipses completem o varrimento da esfera. Apesar desta inversão, não há nenhuma descontinuidade no processo de varrimento ao se cruzar o pólo Sul.

5.2.4 Eventos-local e eventos-círculo

O escalonamento e o processamento dos eventos na esfera não diferem muito do caso planar. No entanto, alguns detalhes merecem ser mencionados.

Um evento-local adiciona um novo arco de elipse à frente de onda e, se não é o primeiro, um arco repetido e duas novas intersecções. As intersecções de arcos seguem em direcções opostas e, a não ser que outros eventos se interponham, voltam a juntar-se quando o varrimento termina. Como esperado, a prioridade de um evento-local é simplesmente a co-latitude do local envolvido.

Um evento-círculo assinala a eliminação de um arco, pela junção de duas intersecções de arcos, e corresponde à intersecção de duas mediatrizes. Determinar se um arco conduz a um evento-círculo não é tão imediato como no caso planar. Agora as duas mediatrizes intersectam-se duas vezes (em posições antípodas) e os caminhos das intersecções têm alcance limitado (que é metade de um círculo máximo).

Sejam $\langle a,b\rangle$ e $\langle b,c\rangle$ as intersecções adjacentes ao arco $\langle b\rangle$. Sejam também v e $r^{a,b,c}$ o centro e o raio do círculo esférico que circunscreve os locais a, b e c (tal que o determinante $|\vec{v},\vec{a}-\vec{b},\vec{b}-\vec{c}\,| > 0$). Se $a_\rho \leq b_\rho$ e $b_\rho > c_\rho$ ou $a_\rho < b_\rho$ e $b_\rho \geq c_\rho$, então as duas intersecções de arcos $\langle a,b\rangle$ e $\langle b,c\rangle$ seguem em direcções opostas, o que significa que $\langle b\rangle$ não desaparece. Qualquer outro caso pode conduzir a um evento-círculo associado a $\langle b\rangle$, que será escalonado (com prioridade $v_\rho + r^{a,b,c}$) se uma das seguintes condições for verificada.

1. $a_\rho = b_\rho = c_\rho$.
   As duas intersecções juntam-se num dos pólos.

2. $a_\rho > b_\rho$, $b_\rho < c_\rho$ e $v_\lambda \in a^+ \cap c^-$.
   As duas intersecções seguem uma contra a outra, juntando-se durante os percursos respectivos.

3. $a_\rho > b_\rho, b_\rho \geq c_\rho$ e $v_\lambda \in a^+ \cap b^+$.
   Os caminhos de cada intersecção (que percorrem longitudes crescentes) juntam-se durante os percursos respectivos.

4. $a_\rho \leq b_\rho, b_\rho < c_\rho$ e $v_\lambda \in b^- \cap c^-$.
   Este é o oposto do caso anterior. Ambas as intersecções percorrem pontos de longitude decrescente.

As condições enunciadas dependem da circularidade da frente de onda. Se o arco dividido tem um evento-círculo associado, então o evento é conhecido pelas duas instâncias do arco. Por este motivo, o primeiro e o último arco podem ter dois eventos associados (um evento-círculo e um evento-rotação).

Existe ainda um restrição adicional, que condiciona o escalonamento de eventos-círculo e eventos-rotação: a prioridade de um evento não pode exceder $2\pi - s_\rho$, onde s é o local mais a Sul, uma vez que o varrimento termina quando o raio do círculo de varrimento atinge aquele valor. Logo, eventos-círculos e eventos-rotação cuja prioridade (calculada) ultrapassa este limiar não são escalonados.

5.2.5 Cálculo de prioridades

O cálculo das prioridades e a definição da relação de ordem usada na frente de onda dependem da escolha do sistema de coordenadas. Até agora, foi adoptado o sistema de coordenadas esférico por ser a escolha natural para lidar com posições na esfera. Tem, no entanto, dois inconvenientes: requer o uso de funções trigonométricas e depende da comparação de valores não exactos (por exemplo, π). Mas tal como no varrimento circular, estes obstáculos são ultrapassados facilmente recorrendo a coordenadas Cartesianas.

Sejam os locais representados por (ou convertidos para) coordenadas Cartesianas, em que um local $a = (a_\rho, a_\lambda)$ é transformado num vector unitário $\vec{a} = (a_x, a_y, a_z)$. Desta forma, a relação de ordem entre dois locais a e b (c.f. Definição 8) é dada por:

$$a > b \Leftrightarrow \begin{cases} a.z < b.z \\ a.z = b.z \wedge \begin{cases} b_y > 0 \wedge (a_y < 0 \vee \vec{a}^{\perp} \cdot \vec{b} < 0) \\ \qquad \vee \\ b_y < 0 \wedge (a_y < 0 \wedge \vec{a}^{\perp} \cdot \vec{b} < 0) \\ \qquad \vee \\ b_y = 0 \wedge (a_y < 0 \vee b_x \geq 0) \end{cases} \end{cases} \tag{5.7}$$

De igual modo, transforme-se a prioridade associada a um raio de círculo de varrimento ρ na expressão equivalente dada por (sendo $\mathrm{sgn}(v)$ o sinal do valor v):

$$\Pi(\rho) = 2\,\mathrm{sgn}(t)\,t^2, \text{ onde } t = \cos(\rho/2). \tag{5.8}$$

Desta forma, as prioridades variam de $+2$ a -2, sendo positivas no varrimento de Norte para Sul, zero no pólo Sul, e negativas quando o círculo de varrimento regressa para Norte. Como consequência, há que trocar a fila com prioridade organizada por mínimos por uma fila com prioridade organizada por máximos na implementação da fila de eventos. Aplicando (5.8) aos três tipos de prioridades, obtêm-se os seguintes resultados.

A prioridade de um evento-local é simplesmente dada por:

$$\Pi_{\text{local}}(a) = 1 + a_z. \tag{5.9}$$

Nas condições que determinam o escalonamento de um evento-círculo, apenas uma comparação não é trivial: a comparação do centro do círculo v com o alcance dos caminhos das intersecções $\langle a,b\rangle$ e $\langle b,c\rangle$. Essas comparações traduzem-se por:

$$\begin{aligned} v_\lambda \in a^+ \cap b^+ &\Leftrightarrow \vec{a}\,' \cdot \vec{v}\,' \geq 0 \wedge \vec{b}\,' \cdot \vec{v}\,' \geq 0 \\ v_\lambda \in b^- \cap c^- &\Leftrightarrow \vec{b}\,' \cdot \vec{v}\,' \leq 0 \wedge \vec{c}\,' \cdot \vec{v}\,' \leq 0 \end{aligned} \tag{5.10}$$

onde

$$\begin{cases} \vec{u} = (\vec{a} - \vec{b}) \times (\vec{c} - \vec{b}), \\ \vec{v} = \vec{u}\,/\|\vec{u}\|, \end{cases} \tag{5.11}$$

$\vec{v}\,' = (v_x, v_y)$, $\vec{a}\,' = (a_x, a_y)^\perp$, $\vec{b}\,' = (b_x, b_y)^\perp$ e $\vec{c}\,' = (c_x, c_y)^\perp$.

A prioridade de um evento-círculo é dada por:

$$\begin{gathered} \Pi_{\text{círculo}}(a,b,c) = 1 + \sigma\, v_z - \sqrt{(1 - v_z^2)(1 - \sigma^2)}, \text{ onde} \\ \begin{cases} \sigma = \vec{v} \cdot \vec{a}. \end{cases} \end{gathered} \tag{5.12}$$

e $\vec{v}$ é dado por (5.11).

Um evento-rotação é gerado sempre que o caminho da primeira ou da última intersecção de arcos da frente de onda cruza o meridiano principal. Se $\langle a,b\rangle$ é a primeira intersecção de arcos da frente de onda, então é associado um evento-rotação ao arco $\langle a\rangle$ se $b > a \wedge b_y > 0$. De igual modo, se $\langle a,b\rangle$ é a última intersecção de arcos da frente de onda, então é associado um evento-rotação ao arco $\langle b\rangle$ se $a > b \wedge a_y < 0$. Em qualquer dos casos, a prioridade de um evento-rotação é dada por:

$$\begin{gathered} \Pi_{\text{rotação}}(a,b) = \operatorname{sgn}(\sigma + \tau)\left(1 + \sigma\tau - \sqrt{(1 - \tau^2)(1 - \sigma^2)}\right), \text{ onde} \\ \begin{cases} \delta = \|(a_x - b_x, a_z - b_z)\|, \\ \sigma = \operatorname{sgn}(b_z - a_z)\ (a_x\, b_z - a_z\, b_x)/\delta, \\ \tau = \operatorname{sgn}(b_z - a_z)\ (a_x - b_x)/\delta. \end{cases} \end{gathered} \tag{5.13}$$

A relação de ordem usada na ordenação da frente de onda também fica mais simples,

Algoritmo 3 Construção do diagrama de Voronoi V de um conjunto P de locais na esfera.

```
 1: diagrama: V ← ∅                                   {O diagrama de Voronoi.}
 2: árvore: w ← ∅                                     {Árvore vermelha-preta, guarda a frente de onda.}
 3: vector: a ← P.ordenar()                           {Locais ordenados por co-latitude (≤),}
 4:                                                   {e por longitude (<).}
 5: inteiro: k ← 0                                    {Posição em a do próximo evento-local.}
 6: fila: q ← ∅                                       {Fila com prioridade, para eventos-círculo e rotação.}
 7: número: α, β                                      {Prioridades.}
 8: enquanto k < |P| ∨ q ≠ ∅ fazer
 9:    α ← (k < |P|) ? Π_local(a[k]) : −∞
10:    β ← (q ≠ ∅) ? q.prioridade_máxima() : −∞
11:    se α ≥ β então
12:       processar_evento_local(V, w, q, a[k])
13:       k++
14:    senão                                         {w_0 identifica o nó arco}
15:       (w_0, ev) ← q.remover_máximo()             {ev indica o tipo de evento}
16:       se ev é evento-círculo então
17:          processar_evento_círculo(V, w, q, w_0)
18:       senão
19:          processar_evento_rotação(V, w, q, w_0)
20:       fim
21:    fim
22: fim
23: V.juntar_par_de_arestas(w)
```

sendo (5.6) substituído por:

$$s_\lambda < \langle a,b\rangle_\lambda \;\Leftrightarrow\; \alpha|b_z - s_z| < \beta|a_z - s_z|, \text{ onde}$$
$$\begin{cases} \alpha = s_x\,(s_x - a_x) + s_y\,(s_y - a_y), \\ \beta = s_x\,(s_x - b_x) + s_y\,(s_y - b_y). \end{cases} \tag{5.14}$$

Concluindo, o uso de coordenadas Cartesianas permite implementar o algoritmo de varrimento esférico recorrendo apenas às operação aritméticas e à raiz quadrada.

5.2.6 Algoritmo de varrimento esférico

O Algoritmo 3 descreve o algoritmo de varrimento da esfera. No geral pouco difere do algoritmo de varrimento circular. As diferenças estão encapsuladas no cálculo de prioridades e processamento de eventos. Agora os locais são ordenados por co-latitude e por longitude (linha 3, de acordo com (5.7)). Os eventos são processados por ordem decrescente de prioridade (linha 11). Esgotados os eventos, a frente de onda fica reduzida a dois arcos, cujas intersecções percorrem duas instâncias da mesma aresta. Resta apenas descartar uma das instâncias, ligando a outra apropriadamente (linha 23).

Proposição 4. *O Algoritmo 3 constrói o diagrama de Voronoi esférico de n locais em tempo* $O(n \log n)$ *e em espaço* $\Theta(n)$.

Demonstração. Todas as estruturas de dados ocupam espaço $\Theta(n)$ uma vez que o número de arcos na frente de onda não pode exceder $2n$. Logo, cada operação primitiva na frente de onda ou na fila de eventos tem um custo $O(\log n)$, se implementadas, respectivamente, por uma árvore preta-vermelha e por um *heap* binário mais um vector (que associa cada evento na fila com a sua posição no heap). O número de eventos-rotação corresponde ao número de regiões atravessadas pelo meridiano principal que, por serem convexas, não pode ultrapassar n, intersectada cada uma em duas arestas. Como cada aresta é partilhada por duas regiões, tem-se um máximo de n arestas intersectadas pelo meridiano principal, correspondendo a n eventos-rotação. Por fim, o processamento de cada evento requer um número constante de operações primitivas sendo o número total de eventos processados $\Theta(n)$. Mais exactamente, tem-se n eventos-local, $2n-4$ eventos-círculo e um máximo de n eventos-rotação. □

5.3 A transformada de inversão

Um resultado importante, devido a Brown [9], relaciona o diagrama de Voronoi de locais na superfície da esfera com o diagrama de Voronoi da projecção estereográfica dos locais num plano, através de uma transformada de inversão. Seja c um ponto da esfera unitária S que é, simultaneamente, a origem e um centro de projecção. Seja T o plano tangente a S que toca o antípoda de c. Para cada ponto $v = (r, \theta, \phi)$, definido em coordenadas polares em relação à origem c, a *transformada de inversão de* v é definida por:

$$v^{-1} = \left(\frac{1}{r}, \theta, \phi\right) . \tag{5.15}$$

Ou seja, o vector v^{-1} tem a mesma direcção de v, mas o inverso da magnitude. Esta transformada tem a propriedade de transformar superfícies esféricas tangentes a c em superfícies planas que não contêm c e vice-versa. Em particular, esta transformada imita a projecção estereográfica, transforma pontos da esfera S em pontos do plano T.

Além do mais, a transformada de inversão estabelece uma relação entre o diagrama de Voronoi esférico de locais em S com o diagrama de Voronoi planar da projecção estereográfica dos locais (de S) em T. Seja γ um círculo vazio máximo que contém pelo menos três locais de S na sua fronteira e P o plano que intersecta S em γ. A transformada de inversão projecta esses locais em T, ao mesmo tempo que transforma o plano P numa esfera P^{-1} que, ao intersectar T, define um círculo γ^{-1}. O facto crucial é que os círculos γ e γ^{-1} contêm os mesmos conjuntos de locais nos seus bordos. Adicionalmente, se γ *não contém* o centro de projecção c, então γ^{-1} é um círculo vazio máximo do diagrama de Voronoi dos pontos mais próximos (em T); e se γ *contém* c, então γ^{-1} é o círculo englobante mínimo do diagrama de Voronoi dos pontos mais afastados (também em T).

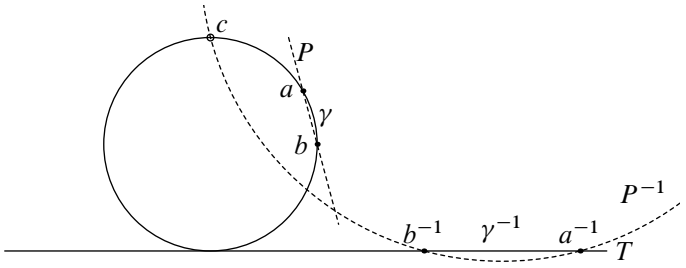


Figura 5.8: Inversão do diagrama dos pontos mais próximos em duas dimensões.

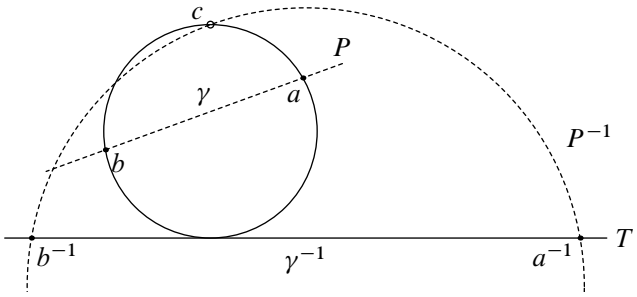


Figura 5.9: Inversão do diagrama dos pontos mais afastados em duas dimensões.

As Figuras 5.8 e 5.9 ilustram estas relações para o caso bidimensional, onde esferas e planos são substituídos por círculos e linhas rectas. Em ambos os casos, o centro de γ^{-1} é um vértice do diagrama de Voronoi em T.

Um corolário destes resultados é que um algoritmo que construa o diagrama de Voronoi esférico também constrói implicitamente dois diagramas de Voronoi planares: um diagrama dos pontos mais próximos e um diagrama dos pontos mais afastados. O algoritmo de varrimento esférico constrói-os à vez: primeiro o diagrama dos pontos mais próximos e depois o diagrama dos pontos mais afastados. Posicionando o centro de projecção c no pólo Sul e o centro de varrimento no pólo Norte, como é habitual, observa-se que:

- um vértice cujo círculo vazio máximo não inclui c é encontrado *antes* do círculo de varrimento cruzar o pólo Sul; e
- um vértice cujo círculo vazio máximo contém c é encontrado *depois* do círculo de varrimento cruzar o pólo Sul.

Este resultado deriva do facto de o círculo vazio máximo centrado num ponto da frente de onda ser tangente ao círculo da varrimento. Concluindo, o algoritmo de varrimento esférico constrói implicitamente dois diagrama de Voronoi planares da projecção estereográfica dos locais: constrói o diagrama dos pontos mais próximos quando o círculo de varrimento varia de 0 a π e constrói o diagrama dos pontos mais afastados quando o círculo de varrimento varia de π a $2\,\pi$.

Capítulo 6

Edição de diagramas de Voronoi

Neste capítulo são descritos dois algoritmos que actualizam um diagrama de Voronoi. Nomeadamente, um algoritmo para inserir um novo local e um algoritmo para remover um local. Ambos os algoritmo são baseados na técnica de varrimento circular, sendo aplicáveis à edição de diagramas de Voronoi planares e esféricos. Na secção 6.1 são descritos os algoritmos de edição de diagramas esféricos, sendo as variantes para o caso planar descritas na secção 6.2.

A edição de um diagrama de Voronoi V, seja por inserção ou remoção de um local s, modifica, regra geral, apenas uma pequena parte de V que, numa remoção, corresponde à parte de V cujos círculos vazios máximos são tangentes a s e, numa inserção, corresponde à parte de V cujos círculos vazios máximos contêm s. Por definição, a parte de V referida é exactamente a região de s, $V(s)$, seja a região a remover ou a região a inserir.

Os algoritmos de edição seguem a mesma estratégia do algoritmo de Fortune. O algoritmo de remoção varre $V(s)$ com uma frente de onda que começa na fronteira da região e termina num arco contido no seu interior. O algoritmo de inserção faz exactamente o oposto: a frente de onda começa por ser um arco com um extremo em s e cresce até que atinge o contorno de $V(s)$. A Figura 6.1 ilustra os dois processos.

A tarefa dos algoritmos de edição é identificar um conjunto de arestas e vértices a adicionar a V (numa remoção) ou a remover de V (numa inserção). Ignorando a fronteira de $V(s)$, que é calculada numa inserção, o referido conjunto forma um grafo acíclico G_s (ver

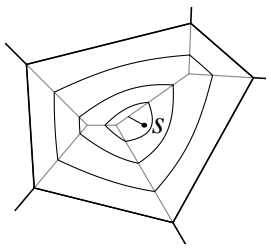


Figura 6.1: Varrimento da região $V(s)$ numa edição.

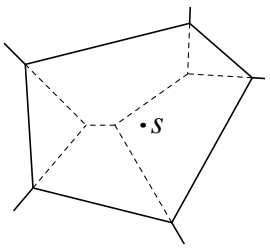


Figura 6.2: Grafo acíclico G_s formado pelas arestas e vértices percorridos durante o varrimento (duas arestas são interiores).

Figura 6.2). Define-se por *arestas interiores* de G_s aquelas cujos extremos estão contidos no interior de $V(s)$. O número de elementos de G_s deriva directamente das propriedades dos diagrama de Voronoi. Sendo m o número de locais vizinhos de s, G_s contém $m-3$ arestas interiores cujos extremos definem $m-2$ vértices, uma vez que a adição (ou remoção) de um local implica uma variação de três arestas e dois vértices no diagrama (c.f. (2.7)).

Os algoritmos modificam um diagrama de Voronoi por aplicação de três operações básicas, descritas na secção 2.3: rotação de arestas e inserção ou remoção de um par de arestas. A remoção de um local é feita contraindo a região respectiva, descartando arestas da sua região para regiões vizinhas através de uma sucessão de rotações de arestas, até ao ponto em que a região fica reduzida a duas arestas. Por fim, esse par de arestas é removido (de uma só vez, eliminando três arestas). Reciprocamente, a inserção de um local aplica a sequência de operações básicas em sentido inverso. Começa com a inserção de um par de arestas, que é o contorno da região inicial do novo local, e que, efectivamente, adiciona três arestas ao diagrama. Depois, a região é expandida por uma sequência de rotações de arestas, absorvendo arestas das regiões vizinhas, até que atinja a forma final.

6.1 Edição na esfera

A edição de um local s é feita varrendo a região $V(s)$ por um círculo centrado no próprio local. Para simplificar a descrição, assume-se que $s = o = (0,0)$, o que pode sempre ser obtido após uma rotação apropriada. A construção da frente de onda segue o mesmo esquema utilizado na construção do varrimento esférico. Dado um círculo de varrimento centrado em o, interessa considerar o lugar dos pontos equidistantes ao círculo e a um local. O resultado é uma hipérbole, sendo agora a frente de onda formada por arcos de hipérbole, onde cada arco é definido pelo círculo de varrimento e por um local.

Seja $a \neq (0,0)$ um local na superfície da esfera e r o raio do círculo de varrimento. Os pontos i tais que:

$$\overline{i-a} = r + i_\rho, \text{ para todo o } r \in [0, a_\rho], \tag{6.1}$$

são o centro de um círculo tangente ao círculo de varrimento (com o no seu interior) que passa por a (como ilustrado na Figura 6.3). A equação (6.1) define um ramo da hipérbole esférica cujos focos são a e o e cuja distância entre vértices é r. De agora em diante, os ramos de hipérbole denominam-se apenas por *hipérboles*.

Vamos agora ver como a hipérbole gerada por um arco varre um hemisfério. Primeiro, recorde-se que as hipérboles esféricas são curvas fechadas (e congruentes com elipses esféricas). A forma de uma hipérbole é delimitada por um sector esférico cujo ângulo diedro é dado por:

$$\lambda = 2\arccos(\csc(a_\rho/2)\ \mathrm{sen}(r/2)). \tag{6.2}$$

Quando $r = 0$, $\lambda = \pi$ e a hipérbole degenera na mediatriz $\mathcal{M}_{ao}$ do arco geodésico que une os focos. À medida que r aumenta, λ diminui, e a hipérbole fecha na direcção de o. Finalmente,

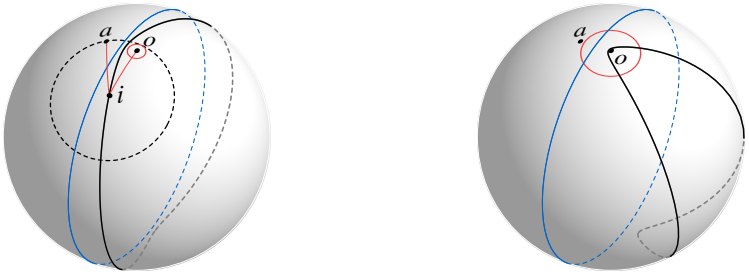


Figura 6.3: Varrimento de um hemisfério por um ramo da hipérbole esférica definida pelo local a e pelo círculo de varrimento (centrado em o). O círculo máximo ilustra $\mathcal{M}_{ao}$. Na figura da esquerda, o círculo a tracejado centrado em i ilustra a Eq. (6.1).

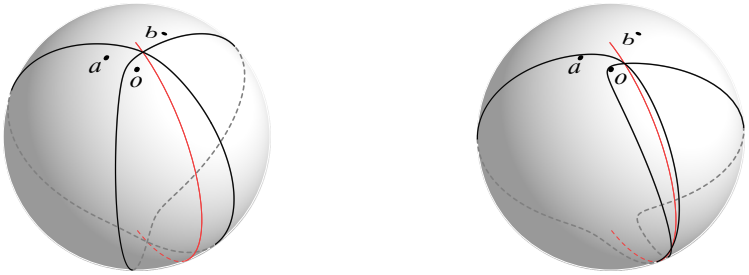


Figura 6.4: Varrimento da meia-mediatriz $\mathcal{M}_{ab}$ pelas intersecções de duas hipérboles (quando r diminui).

quando $r = a_\rho$, $\lambda = 0$ e a hipérbole degenera novamente, agora no arco geodésico que une o ao antípoda de a. Deste modo, a hipérbole varre monotonamente o hemisfério definido pela mediatriz $\mathcal{M}_{ao}$ e que contém o, quando r varia de 0 a a_ρ. Consequentemente, numa remoção, as hipérboles geradas pelos locais vizinhos de s varrem a totalidade do interior de $V(s)$, da fronteira para o interior. O mesmo se aplica na operação de inserção, onde $V(s)$ é varrido do interior para a fronteira, quando r varia de a_ρ até 0.

A intersecção de duas hipérboles, geradas pelos locais a e b, percorrem metade da mediatriz $\mathcal{M}_{ab}$ dos dois locais (c.f. Figura 6.4). Note-se que, devido a (6.1), qualquer ponto de intersecção das duas parábolas pertence a $\mathcal{M}_{ab}$. Quando $r = 0$, os pontos de intersecção são antípodas entre si, uma vez que as hipérboles coincidem com $\mathcal{M}_{ao}$ e $\mathcal{M}_{bo}$, que são círculos máximos. À medida que r aumenta, as intersecções deslocam-se uma em direcção à outra, até ficarem coincidentes. Isto acontece para $r = \min(a_\rho, b_\rho)$, quando uma das hipérboles completa o seu varrimento, degenerando num arco geodésico.

Por fim, vamos ver como duas intersecções de hipérbole convergem para um vértice. Seja c um terceiro local e sejam $\langle a \rangle$, $\langle b \rangle$ e $\langle c \rangle$ os três arcos de hipérbole respectivos, cujo foco comum é o. Sem perda de generalidade, sejam $\langle a,b \rangle$, $\langle b,c \rangle$ e $\langle c,a \rangle$ as correspondentes intersecções de arcos, e sejam v e $r^{a,b,c}$ o centro e o raio do círculo circunscrito aos três locais, respectivamente (ver Figura 6.5). Seja agora a frente de onda definida pelo envelope interior das hipérboles que, neste caso, é formada por três arcos e três intersecções de arcos. As três intersecções de arcos convergem para o interior da frente de onda, percorrendo as mediatrizes definidas por pares consecutivos de locais, que se intersectam em v. No entanto, apenas duas

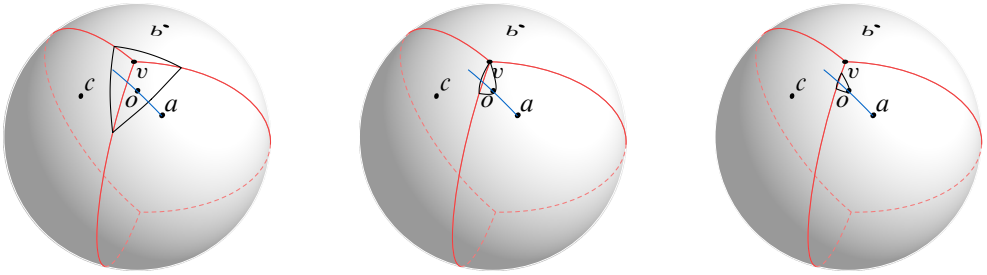


Figura 6.5: Convergência de duas intersecções num vértice. As intersecções $\langle a,b\rangle$ e $\langle b,c\rangle$ juntam-se em v (causando o desaparecimento do arco $\langle b\rangle$).

das intersecções de arcos podem convergir em v, uma vez que a restante intersecção termina, no caso geral, o seu percurso antes de chegar ao vértice. Ou de modo equivalente, apenas um dos três arcos é eliminado da frente de onda no processo de descoberta do vértice. Por exemplo, $\langle a,b\rangle$ e $\langle b,c\rangle$ cruzam-se em v (eliminando o arco $\langle b\rangle$) apenas se $\langle c,a\rangle$ não o faz, o que é facilmente determinado comparando a posição de v em relação ao círculo máximo que passa por o e a, como ilustrado na Figura 6.5. A eliminação de $\langle b\rangle$ ocorre para um raio de círculo de varrimento $r = r^{a,b,c} - v_\rho$.

No caso particular de uma frente de onda com apenas três arcos (atrás analisado), todos os três arcos convergem para o mesmo vértice. No caso geral, em frentes de onda de quatro ou mais arcos, tem-se que cada arco converge efectivamente para um único vértice.

6.1.1 Algoritmo de remoção

Seja s um local a remover de um diagrama de Voronoi V, m o número de vizinhos de s (em V) e u o seu vizinho mais próximo. Recorde-se que $s = (0,0)$ (i.e., posicionado no pólo Norte). O círculo de varrimento (centrado em s), conjuntamente com os m vizinhos de s, definem m hipérboles. Como referido atrás, o envelope interior das hipérboles define a frente de onda, que é uma curva fechada formada por uma sequência alternada de arcos de hipérbole e intersecções de arcos. Inicialmente, para $r = 0$, a frente de onda coincide com $V(s)$. À medida que r aumenta, as hipérboles convergem na direcção de s, causando a eliminação de arcos na frente de onda, o que assinala a presença de novos vértices e novas arestas. O processo termina quando a hipérbole gerada por u termina o seu varrimento, ou seja, quando $r = u_\rho$. Portanto, o algoritmo de remoção pode ser visto como um caso particular do Algoritmo 3 em que apenas há eventos-círculo (c.f. Figura 6.6 que, para simplificar, ilustra a remoção no plano).

O varrimento de remoção induz um percurso do grafo G_s das arestas não-interiores para um ponto interior (que pertence à aresta intersectada pelo semi-círculo máximo $u \triangleright s$). Em cada evento-círculo, é eliminado um arco da frente de onda, que determina um vértice do grafo. Quando restam apenas três arcos na frente de onda, só falta determinar um vértice, o que pode ser obtido pela eliminação de qualquer um dos arcos. O varrimento termina quando a frente de onda fica reduzida a dois arcos, que são removidos numa só operação.

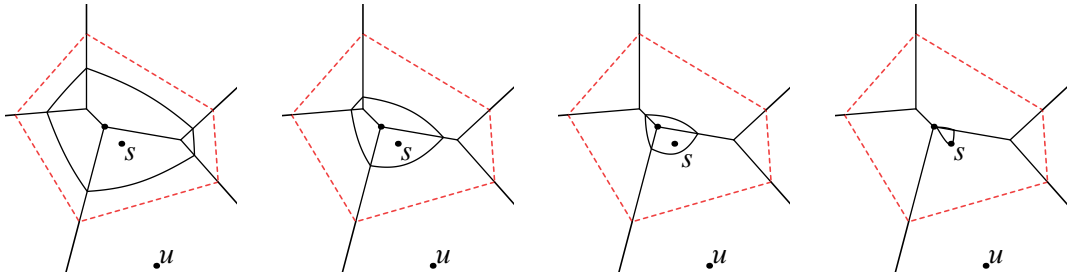


Figura 6.6: Remoção de um local s de um diagrama de Voronoi V

Na implementação da operação de remoção, é indiferente qual dos eventos dos três arcos é processado para a descoberta do último vértice. No entanto, caso se pretendesse realizar uma simulação do processo de varrimento da frente de onda, onde é importante respeitar a correcta ordem dos eventos, seria necessário determinar o arco que efectivamente desaparece no vértice (pela regra indicada na secção anterior).

O algoritmo de remoção, para diagramas com pelo menos três locais, está descrito no pseudo-código Algoritmo 4. Começa por inicializar a frente de onda com $V(s)$ (linha 1). Se a frente de onda apenas tem dois arcos (caso especial de um diagrama de apenas três locais, c.f. secção 5.2.1), então basta removê-los (linhas 2–3). Senão, para cada um dos arcos iniciais, há que calcular e inserir na fila com prioridade o evento-círculo correspondente, ordenado por co-latitude (linhas 5–10). Depois, enquanto há mais que três arcos na frente de onda, é removido o arco com menor prioridade (através de uma rotação de aresta, que deixa um vértice para trás), e são actualizadas as prioridades dos arcos adjacentes (linhas 11–20). Por fim, é feita uma última rotação de aresta, que calcula o último vértice (linhas 21–22), e os dois arcos restantes são descartados (linha 23).

Proposição 5. *O algoritmo 4 remove um local com m vizinhos de um diagrama de Voronoi em tempo $O(m \log m)$ e com espaço $\Theta(m)$.*

Demonstração. É fácil verificar que o comprimento da fila com prioridade é inicialmente de m, nunca excedendo este valor. Logo, o espaço requerido pela fila é $\Theta(m)$. Adicionalmente, cada operação primitiva na frente de onda e na fila com prioridade tem um custo $O(1)$ e $O(\log m)$, respectivamente, se a frente de onda é implementada por uma DCEL, a fila com prioridade é implementada por um *heap* binário e um vector (tal como no Algoritmo 3), e todo o elemento da fila com prioridade tem, para além de um arco da frente de onda, um ponteiro para o elemento correspondente na lista. Além do mais, cada evento-círculo requer um número constante de operações primitivas para ser processado, para um total de $m - 2$ eventos processados (que é o número de vértices). Por fim, o custo de construção da fila com prioridade é $\Theta(m)$. □

À semelhança do que foi feito para o algoritmo de construção do diagrama, também aqui é vantajoso adaptar o cálculo de prioridades a coordenadas Cartesianas. Neste caso, uma

Algoritmo 4 Remoção de um local s de um diagrama de Voronoi esférico V.

```
1:  lista: w ← V.região(s)                          {Frente de onda é V(s).}
2:  se w.comprimento() = 2 então
3:      V.remover_par_de_arestas(w)
4:  senão                                           {w.comprimento() ≥ 3.}
5:      vector: a ← ∅                               {Vector auxiliar.}
6:      para toda a aresta e em w fazer
7:          k ← w.calcular_prioridade(e)
8:          a.insere((k, e))
9:      fim
10:     fila: q ← construir_fila(a)                 {Fila com prioridade.}
11:     enquanto q.comprimento() > 3 fazer
12:         (k, e) ← q.remover_mínimo()
13:         p ← w.anterior(e)
14:         n ← w.seguinte(e)
15:         w.rodar_aresta(e)
16:         k_p ← w.calcular_prioridade(p)
17:         q.actualizar((k_p, p))
18:         k_n ← w.calcular_prioridade(n)
19:         q.actualizar((k_n, n))
20:     fim                                         {Frente de onda tem 3 arcos.}
21:     (k, e) ← q.remover_mínimo()                 {Escolher um dos três eventos.}
22:     w.rodar_aresta(e)
23:     V.remover_par_de_arestas(w)
24: fim
```

prioridade de co-latitude ρ é redefinida por:

$$\Pi''(\rho) = 2\,\mathrm{sen}^2(\rho/2), \tag{6.3}$$

em que as prioridades variam de 0 a $+2$. Sejam $\vec{a}$, $\vec{b}$ e $\vec{c}$ os vectores unitários correspondentes aos locais a, b e c na superfície da esfera, respectivamente. A prioridade de um evento-círculo é dada por:

$$\Pi''_{\text{círculo}}(a,b,c) = 1 - \sigma\, v_z - \sqrt{(1-v_z^2)(1-\sigma^2)}, \quad \text{onde}$$
$$\begin{cases} \vec{u} = (\vec{a} - \vec{b}) \times (\vec{c} - \vec{b}), \\ \vec{v} = \vec{u}\,/\|\vec{u}\|, \\ \sigma = \vec{v} \cdot \vec{a}. \end{cases} \tag{6.4}$$

Curiosamente, este algoritmo é bastante semelhante ao algoritmo de Devillers (descrito na secção 3.3). No algoritmo de remoção na triangulação, a adição de uma orelha equivale à rotação de uma aresta na remoção no diagrama. Ambos os algoritmos começam por considerar a fronteira do *buraco* criado pela remoção de um local, que é depois preenchido iterativamente com a adição de arestas, escolhidas por ordem de prioridade. A principal diferença reside

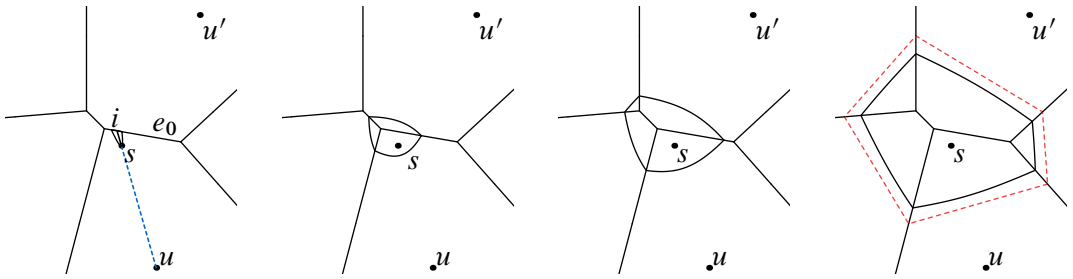


Figura 6.7: Inserção de um local s num diagrama de Voronoi V, sabendo que o local mais próximo é u.

na definição de prioridade. No algoritmo de Devillers, a prioridade é dada pela potência de um local em relação a um círculo. Esta prioridade envolve o cálculo do círculo circunscrito a três pontos, que é tudo o que é preciso para determinar a prioridade no Algoritmo 4. Mas, porque o cálculo da potência tem um custo elevado, o autor refere que soluções quadráticas, mas simples, executam de forma mais rápida [24].

Por ser mais eficiente, o algoritmo foi implementado descartando por completo a fila de eventos. Devido ao reduzido número de locais vizinhos, em média de seis, observou-se que o desempenho melhora quando a operação remover_mínimo é implementada por uma procura sequencial na frente de onda, eliminando todas as inserções e actualizações de eventos-círculo na fila de eventos.

6.1.2 Algoritmo de inserção

O algoritmo de inserção pode ser visto como o algoritmo de remoção executado por ordem inversa. Seja s um local a ser inserido num diagrama de Voronoi V e u o local mais próximo de s (recorde-se que $s = (0,0)$). O círculo de varrimento é centrado em s, mas o seu raio varia agora de u_ρ para 0. Na inserção, o varrimento começa onde o algoritmo de remoção termina. Seja e_0 a aresta de $V(u)$ que é intersectada pelo semi-círculo máximo $u \triangleright s$, i o ponto de intersecção e u' o vizinho de u com quem u partilha a aresta e_0 (c.f. Figura 6.7 que, para simplificar, ilustra a inserção no plano). A frente de onda começa por ter dois arcos, um gerado por u e outro gerado por u', sendo que o primeiro é o arco geodésico que liga s a i. Portanto, as duas intersecções de arcos começam coincidentes em i. À medida de r diminui, os dois arcos de hipérbole *abrem* ao mesmo tempo de as intersecções de arcos percorrem e_0 até que um terceiro arco é inserido na frente de onda. Isto ocorre quando uma das intersecções de arcos cruza um vértice v de V (que é um dos extremos de e_0). Neste instante, há que verificar se v não pertence ao diagrama resultante da inserção de s, o que é feito com recurso aos círculos vazios máximos. Se o círculo vazio máximo $C(v)$ (em V) contém s no seu interior, então diz-se que v *está em conflito com* s e o vértice deve ser removido. Note-se que v está em conflito com s, se $r_v - v_\rho \geq 0$, onde r_v é o raio de $C(v)$. Nesse caso, é escalonado um evento de forma a que a frente de onda ganhe um arco e duas novas intersecções, que irão percorrer duas novas arestas de V. A prioridade desse evento é

dada por $r = r_v - v_\rho$.

O algoritmo de inserção pode ser delineado da seguinte forma. Identificada a aresta inicial, é escalonado um evento-círculo para cada vértice extremo da aresta que está em conflito com s. Os eventos são inseridos numa fila com prioridade. Depois, e enquanto a fila não está vazia, é retirado da fila o evento com maior prioridade, que é processado removendo do diagrama o vértice v correspondente (por via de uma rotação de aresta), seguido do escalonamento de um evento-círculo por cada vértice vizinho de v ainda não visitado e que esteja em conflito com s (no máximo de dois eventos). A remoção termina quando se esgota a fila de eventos.

Basicamente, a tarefa do varrimento circular é identificar e remover os vértices (e arestas) que estão em conflito com o local que é inserido, que caracteriza o grafo G_s referido no início deste capítulo. Mas como G_s é um grafo conexo e acíclico, é fácil calcular a sua extensão com um percurso (parcial) em V, começando numa posição de G_s arbitrária. Um exemplo seria a aresta e_0. Mas, de facto, qualquer vértice de $V(u)$ que esteja em conflito com s serve. Além disso, o diagrama pode ser percorrido por qualquer ordem, uma vez que todos os eventos escalonados são independentes e são necessariamente processados. Ou seja, para a operação de inserção, as prioridades dos eventos são irrelevantes.

O primeiro passo do Algoritmo 5 (para um diagrama com pelo menos três locais) é encontrar a meia-aresta inicial e'_i (linhas 1–6). É uma das arestas de $V(u)$ cuja origem está em conflito com s (o que é verificado, no mínimo, por uma aresta). A fila de eventos é criada (linha 7), na qual são inseridas uma ou duas meias-arestas: e'_i (linha 8) e a sua aresta gémea, desde que a respectiva origem também esteja em conflito com s (linhas 9–12). A frente de onda é inicializada com dois arcos (linha 13). Quando um evento é processado, são encontrados dois novos vértices e, se estes estão em conflito com s, são inseridos na fila os eventos correspondentes (linhas 16–23). Além do mais, um arco e duas intersecções de arcos são adicionadas à frente de onda, o que é feito com uma rotação de aresta no diagrama de Voronoi (linha 24). Este procedimento é repetido até que todos os eventos tenham sido processados. Nesse momento, todas as arestas de $V(s)$ são conhecidas, correspondendo aos arcos da frente de onda quando o raio do círculo de varrimento chega a 0.

Proposição 6. *O Algoritmo 5 insere um local s num diagrama de Voronoi V em tempo $O(k + m)$, usando $O(m)$ espaço, sendo k o número de vizinhos do local u em V e m o número de vizinhos de s no diagrama resultante.*

Demonstração. Dado que os eventos podem ser processados por uma ordem arbitrária, assume-se que são guardados numa fila com disciplina FIFO. Ao todo, são escalonados $m-2$ eventos, em que cada um deles remove um vértice de V. Portanto, o algoritmo requer espaço da ordem de $O(m)$. Além do mais, a aresta inicial e_i é encontrada em $O(k)$ passos, enquanto que a segunda fase do algoritmo requer um tempo da ordem de $O(m)$, dado que o teste de vértice em conflito, o escalonamento de um evento e o posterior processamento são realizados por um número constante de operações. □

Algoritmo 5 Inserção de um local s num diagrama de Voronoi esférico V, u é o local mais próximo de s.

```
 1: para toda a aresta e' em V.região(u) fazer
 2:    se V.em_conflito(e'.origem(), s) então
 3:       e'_i ← e'                          {Encontrada a (meia-)aresta inicial.}
 4:       sair
 5:    fim
 6: fim
 7: fila: q ← ∅                              {Fila com disciplina arbitrária.}
 8: q.insere(e'_i)
 9: e''_i ← V.gémea(e'_i)
10: se V.em_conflito(e''_i.origem(), s) então
11:    q.insere(e''_i)
12: fim
13: V.insere_par_arestas(e'_i, s)            {V(s) começa com dois arcos.}
14: enquanto q ≠ ∅ fazer
15:    e' ← q.remove()
16:    e'_1 ← V.anterior(e')
17:    se V.em_conflito(e'_1.origem(), s) então
18:       q.insere(e'_1)
19:    fim
20:    e'_2 ← V.anterior(V.gémea(e'_1))
21:    se V.em_conflito(e'_2.origem(), s) então
22:       q.insere(e'_2)
23:    fim
24:    V.roda_aresta(e')
25: fim
```

Apesar de o algoritmo de inserção ter sido desenhado de acordo com a técnica de varrimento, a versão final não faz mais que um percurso num grafo. Curiosamente, esta é a estratégia seguida pela operação de inserção do algoritmo incremental de construção da triangulação de Delaunay (descrito na secção 3.2). Observe-se, no entanto, que os algoritmos actualizam estruturas de dados diferentes.

6.2 Edição no plano

A estratégia de varrimento circular tem a propriedade singular de se aplicar ao domínio esférico com a mesma facilidade com que se aplica ao domínio planar. Por este motivo, não é de estranhar que os algoritmos de edição de diagramas de Voronoi esféricos se apliquem, com algumas modificações, aos diagramas planares. Nesta secção são descritas as características do varrimento hiperbólico no plano e o processo de adaptação dos algoritmos de edição ao plano.

A frente de onda hiperbólica foi construída considerando o envelope interior de um con-

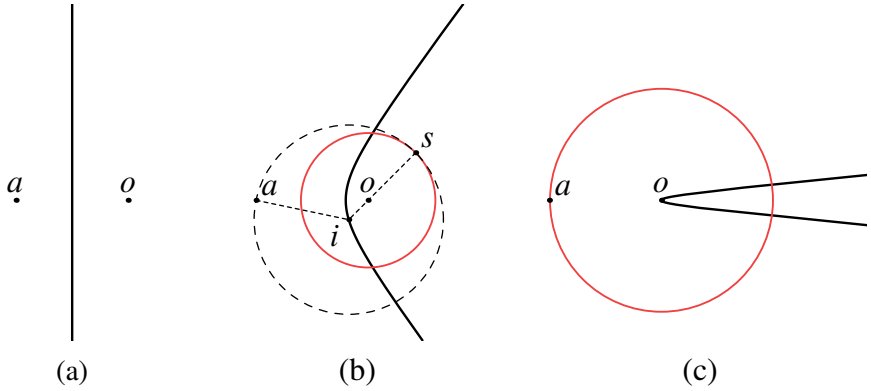


Figura 6.8: Hipérbole definida por um local e um círculo de varrimento.

junto de hipérboles. Cada hipérbole foi definida como o lugar dos pontos equidistantes a um círculo, centrado num local a editar, e a um local vizinho. A forma da hipérbole varia com a distância entre o centro de varrimento e o local gerador e o raio do círculo. Independentemente, cada hipérbole tem a capacidade de varrer um semi-plano definido pela mediatriz entre dois locais: o local gerador e o local a editar.

No plano, cada um destes elementos tem uma forma equivalente. Hemisfério é substituído por semi-plano, círculos máximos por linhas rectas, arcos geodésicos por segmentos de recta e a distância geodésica pela distância Euclidiana. Neste caso, é conveniente especificar os pontos em coordenadas polares, $p = (p_\rho, p_\theta)$. Porque simplifica a análise (tal como na esfera), também aqui se assume que o local a editar s se localiza na origem, isto é, $s = o = (0, 0)$.

Vamos começar por analisar as propriedades das hipérboles no plano, quando definidas no contexto do varrimento circular, seguindo o raciocínio apresentado no caso esférico. Seja $a \neq (0, 0)$ um local do plano e r o raio do círculo de varrimento. Os pontos i tais que:

$$\overline{i - a} = r + i_\rho, \text{ para todo o } r \in [0, a_\rho], \tag{6.5}$$

são o centro de um círculo tangente ao círculo de varrimento (com o no seu interior) que passa por a (ver Figura 6.8b). A equação (6.5) define um ramo de hipérbole cujos focos são a e o e cuja distância entre vértices é r. O ramo de uma hipérbole (aqui também abreviado por hipérbole) é uma curva aberta delimitada por duas assimptotas, cujo ângulo interno é dado por:

$$\lambda = 2 \arccos(r/a_\rho). \tag{6.6}$$

Quando $r = 0$, $\lambda = \pi$ e a hipérbole coincide com $\mathcal{M}_{ao}$ (Figura 6.8a); à medida que r aumenta, λ diminui e as assimptotas convergem em torno de o (Figura 6.8b); e quando $r = a_\rho$, $\lambda = 0$ e a hipérbole degenera na semi-recta com origem em o e na direcção oposta a a (Figura 6.8c).

A intersecção de duas hipérboles varre, igualmente, metade da mediatriz entre dois locais. Porém, dado que se tratam agora de curvas não fechadas, há que analisar três casos distintos.

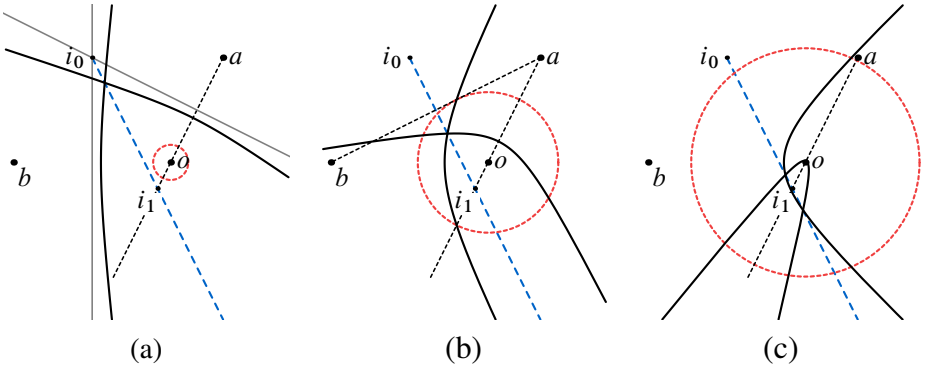


Figura 6.9: Varrimento de meia-mediatriz por *duas* intersecções de hipérboles.

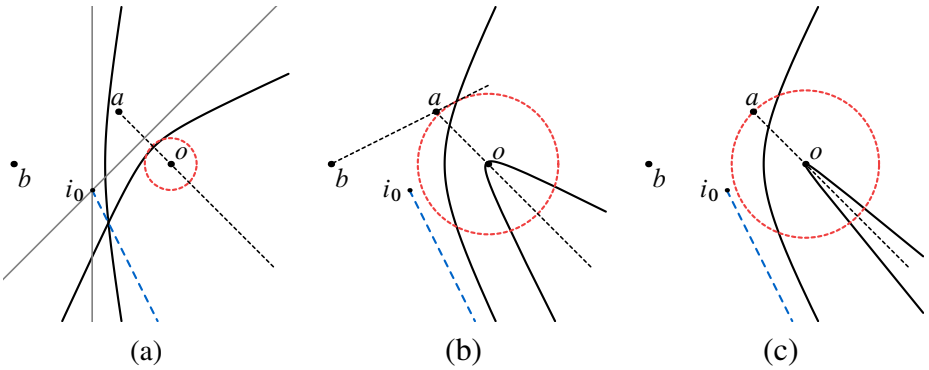


Figura 6.10: Varrimento de meia-mediatriz por *uma* única intersecção de hipérboles.

Seja $b = (b_\rho, b_\theta)$ um segundo local e sejam $\mathcal{M}_{ab}$, $\mathcal{M}_{ao}$ e $\mathcal{M}_{bo}$ as mediatrizes entre os locais (a, b), (a, o) e (b, o), respectivamente.

Consideremos primeiro que a, b e o não são colineares e que a semi-recta $a \triangleright o$ intersecta $\mathcal{M}_{ab}$ (como ilustrado na Figura 6.9) ou, de modo equivalente, em que o ponto da recta $a\,b$ mais próximo de o está contido em $\overline{a\,b}$. Inicialmente, quando $r = 0$, as hipérboles coincidem com $\mathcal{M}_{ao}$ e $\mathcal{M}_{bo}$, intersectando-se em i_0. Com o aumento de r, o ponto de intersecção percorre $\mathcal{M}_{ab}$ até que, quando $r = \text{dist}(o, \overline{a\,b})$, uma assimptota de cada hipérbole fica paralela com $\mathcal{M}_{ab}$ (ver Figura 6.9b), altura em que começa a ser traçada uma segunda intersecção (com início no infinito). As duas intersecções percorrem $\mathcal{M}_{ab}$, convergindo para um ponto. O varrimento de metade de $\mathcal{M}_{ab}$ é concluído quando $r = \min(a_\rho, b_\rho)$, altura em uma das hipérboles completa o varrimento do plano. Nesse instante, as duas intersecções juntam-se em i_1, que é o ponto de intersecção entre $a \triangleright o$ e $\mathcal{M}_{ab}$.

Numa segunda configuração, considerem-se três locais igualmente não colineares, mas em que a semi-recta $a \triangleright o$ não intersecta $\mathcal{M}_{ab}$ (c.f. Figura 6.10) ou, de modo equivalente, que o ponto da recta $a\,b$ mais próximo de o não pertence a $\overline{a\,b}$. O varrimento de $\mathcal{M}_{ab}$ segue um padrão semelhante. Inicialmente, para $r = 0$, as duas hipérboles coincidem com M_{ao} e M_{bo}, intersectando-se em i_0. Com o aumentar de r, o ponto de intersecção varre M_{ab} até que, quando $r = \text{dist}(o, \overline{a\,b})$, uma das assimptotas de cada hipérbole fica paralela com M_{ab}, altura em que o ponto de intersecção (agora localizado no infinito) completa o varrimento de

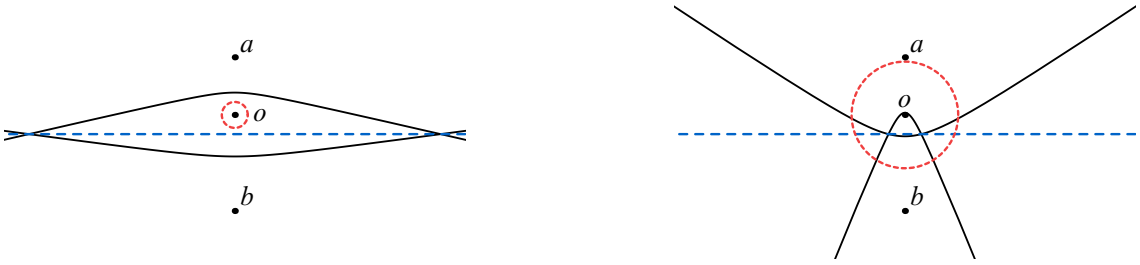


Figura 6.11: Varrimento da mediatriz no caso em que os três locais são colineares.

(metade de) M_{ab}. Note-se que, neste caso, só um dos lados de cada hipérbole é que entra em jogo.

A terceira configuração é obtida com os três locais colineares, com o entre a e b, ou seja, quando o pertence a $\overline{a\,b}$. Agora as duas hipérboles estão viradas uma para a outra, intersectando-se sempre em ambos os lados da recta $\overline{a\,b}$ (ver Figura 6.11). As duas intersecções começam no infinito, juntando-se a meio caminho entre a e b (quando $r = \min(a_\rho, b_\rho)$).

No plano, a frente de onda é igualmente definida pelo envelope interior de hipérboles, tendo o por foco comum. A frente de onda pode agora ser uma curva aberta ou fechada, consoante a configuração das hipérboles que a formam. Possui, no entanto, as mesmas propriedades: as intersecções de arcos percorrem arestas do diagrama de Voronoi, enquanto que as mudanças topológicas na frente de onda assinalam a localização dos vértices. Analisemos primeiro a configuração com frente de onda fechada.

Seja $c = (c_\rho, c_\theta)$ um terceiro local. Sem perda de generalidade, sejam a, b e c três locais nas condições da Figura 6.12. Isto é, com $a_\rho < b_\rho \wedge a_\rho < c_\rho$ e com os três locais orientados em torno de o segundo a ordem dos ponteiros do relógio. Os três arcos de hipérbole são $\langle a\rangle$, $\langle b\rangle$ e $\langle c\rangle$, com as correspondentes intersecções de arcos $\langle a,b\rangle$, $\langle b,c\rangle$ e $\langle c,a\rangle$. Seja $v = (v_\rho, v_\theta)$ e $r^{a,b,c}$ o centro e o raio do círculo circunscrito a a, b e c.

Neste caso, o arco $\langle b\rangle$ só desaparece se as intersecções adjacentes, $\langle a,b\rangle$ e $\langle b,c\rangle$, simultaneamente convergem entre si e cruzam v. A primeira condição é trivial. A segunda condição depende da posição relativa de v em relação à linha recta $\overline{o\,a}$, uma vez que o arco $\langle a\rangle$ termina o varrimento antes do arco $\langle b\rangle$ (porque $a_\rho < b_\rho$), como ilustrado na Figura 6.12. O arco $\langle b\rangle$ é eliminado quando o raio do círculo de varrimento $r = r^{a,b,c} - r_\rho$.

Como se observou no varrimento hiperbólico na esfera, em frentes de onda formadas por quatro ou mais arcos, cada arco converge efectivamente para um vértice distinto. E quando restam apenas três arcos, a determinação do vértice que falta é obtida pela eliminação de qualquer um dos arcos.

O processo de eliminação de arcos é facilmente estendido ao caso de frentes de onda não fechadas, que podem ocorrer em dois tipos de cenários: na remoção de um local pertencente ao invólucro-convexo ou na inserção de um local que passa a fazer parte do invólucro-convexo. Em qualquer dos casos, é possível considerar que a frente de onda é sempre fechada por adição de um arco auxiliar (desenhado a tracejado na Figura 6.13), que liga duas arestas divergentes e infinitas, tal como indicado na secção 2.3. Um arco auxiliar participa num al-

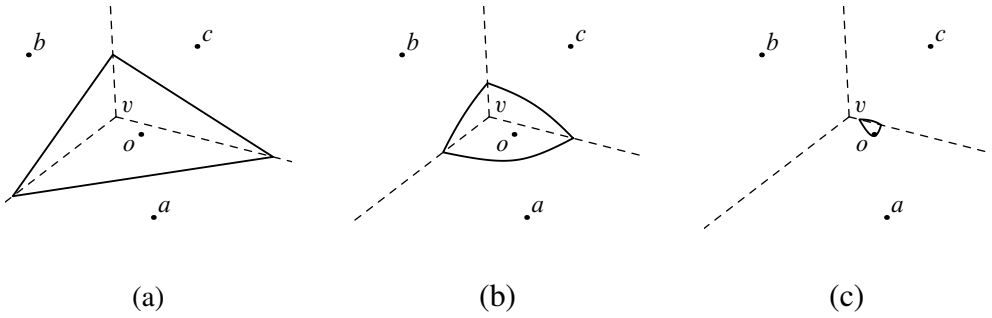


Figura 6.12: Eliminação de um arco por convergência de duas intersecções de hipérboles. Caso com frente de onda fechada.

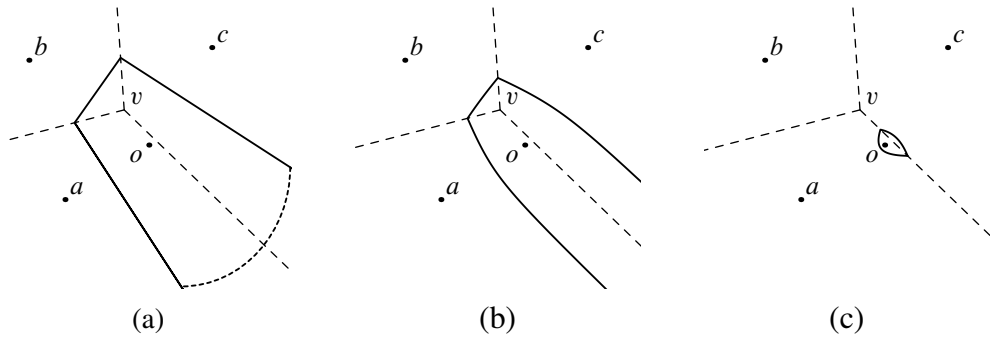


Figura 6.13: Eliminação de um arco por convergência de duas intersecções de hipérboles. Caso com frente de onda aberta.

goritmo de edição tal como um arco normal, podendo ser rodado ou modificado pela inserção ou remoção de um par de arestas. No caso ilustrado na Figura 6.13, em que a semi-recta $a \triangleright o$ intersecta $\mathcal{M}_{ac}$, o arco auxiliar desaparece com $r = \text{dist}(o, \overline{a\,c})$ (ver Figura 6.13b), altura em que as assimptotas dos dois arcos adjacentes ficam paralelas e $\langle a,c \rangle$ começa a traçar $\mathcal{M}_{ac}$.

O truque de adicionar arcos auxiliares a envelopes de hipérboles permite o desenho dos algoritmos de edição de diagramas de Voronoi planares como se todas as regiões envolvidas fossem fechadas, tal como na esfera (c.f. secção 2.3). Em particular, são válidos os mesmos algoritmos desenhados para a edição na esfera (Algoritmos 4 e 5), sendo apenas necessária a adaptação de alguns pormenores, consequência das diferenças entre os dois domínios.

As diferenças nos algoritmos resumem-se ao escalonamento e cálculo de prioridades de eventos-círculo. Na esfera, um arco gera sempre um evento-círculo, uma vez que os caminhos de duas intersecções adjacentes definem um sector de esfera e, por isso, cruzam-se sempre. No plano, um evento-círculo só é escalonado se as intersecções de arcos convergem entre si. O cálculo da prioridade de um evento-círculo, no algoritmo de remoção, e a comparação entre um local e o círculo correspondente a um vértice, no algoritmo de inserção, conduzem necessariamente a diferentes expressões.

Sejam $a = (a_x, a_y)$, $b = (b_x, b_y)$ e $c = (c_x, c_y)$ três locais, especificados em coordenadas Euclidianas, que geram três arcos consecutivos na frente de onda, $\langle a \rangle$, $\langle b \rangle$ e $\langle c \rangle$, respectivamente. A condição que determina a eliminação do arco $\langle b \rangle$ da frente de onda e a

prioridade do evento-círculo correspondente dependem da presença de um arco auxiliar neste trio de arcos. Se nenhum dos três arcos é um arco auxiliar, então o arco $\langle b \rangle$ apenas desaparece se as arestas percorridas pelas intersecções dos arcos adjacentes convergem, ou seja, se $\overrightarrow{a-b}^{\perp} \cdot \overrightarrow{c-b} < 0$. A prioridade do evento-círculo é dada por:

$$\Pi'''_{\text{círculo}}(a,b,c) = \|v\| - \|b+v\| \text{ onde}$$
$$\begin{cases} \vec{ab} = (a_x - b_x, a_y - b_y, \|a-b\|^2), \\ \vec{cb} = (c_x - b_x, c_y - b_y, \|c-b\|^2), \\ \vec{u} = \vec{ab} \times \vec{cb}, \\ \vec{v} = \vec{u}\,/(2\,u_z). \end{cases} \tag{6.7}$$

Se o arco $\langle c \rangle$ é auxiliar (i.e., $c = \varnothing$), então o arco $\langle b \rangle$ apenas desaparece se $a \triangleright o$ intersecta $\mathcal{M}_{ab}$ (o que equivale a determinar se a assimptota de $\langle a \rangle$ converge em torno de o mais rapidamente que a assimptota de $\langle b \rangle$). Esta condição equivale a determinar se $\overrightarrow{a} \cdot \overrightarrow{b-a} > 0$. Neste caso a prioridade é dada por:

$$\Pi'''_{\text{círculo}}(a,b,\varnothing) = \frac{\vec{a}^{\perp} \cdot \vec{b}}{\|b-a\|}\,. \tag{6.8}$$

O caso em que $a = \varnothing$ conduz a uma expressão semelhante.

Resta analisar o caso em que $\langle b \rangle$ é um arco auxiliar ($b = \varnothing$). O arco auxiliar desaparece da frente de onda se $a \triangleright o$ (ou, opcionalmente, $c \triangleright o$) intersecta $\mathcal{M}_{ac}$, o que determina que, durante o varrimento, as assimptotas dos arcos adjacentes ficam paralelas entre si. Esta condição equivale a determinar se $\overrightarrow{a} \cdot \overrightarrow{c-a} < 0$. A prioridade do evento correspondente é dada por:

$$\Pi'''_{\text{círculo}}(a,\varnothing,c) = \frac{\vec{a}^{\perp} \cdot \vec{c}}{\|c-a\|}\,. \tag{6.9}$$

6.2.1 Algoritmo de remoção

O Algoritmo 6 descreve o algoritmo de remoção de um local num diagrama de Voronoi planar. Este algoritmo é idêntico ao algoritmo equivalente no domínio esférico, diferindo no escalonamento de eventos, que agora são condicionais, e na existência de arestas auxiliares. Aos arcos que não geram um evento-círculo, porque não convergem para um vértice, é atribuída uma prioridade de $+\infty$ (linhas 7 e 8). Os eventos são inseridos numa fila de eventos com prioridade (linha 12), sendo processados por ordem crescente de prioridade (linhas 13–22). Enquanto existem mais de três eventos na fila, é removido o evento de menor prioridade (linha 14), que é processado rodando a aresta associada (o que identifica um novo vértice, linha 17) e actualizados os eventos associados aos arcos adjacentes. Para cada um destes arcos, há que determinar o seu desaparecimento e, eventualmente, calcular a prioridade de um novo evento-círculo (linhas 18 e 20). Se existia um evento associado a um arco adja-

Algoritmo 6 Remoção de um local s de um diagrama de Voronoi planar V.

```
 1: lista: w ← V.região(s)                                  {Frente de onda é V(s).}
 2: se w.comprimento() = 2 então
 3:    V.remover_par_de_arestas(w)
 4: senão                                                   {w.comprimento() ≥ 3.}
 5:    vector: a ← ∅                                        {Vector auxiliar.}
 6:    para toda a aresta e em w fazer
 7:       k ← w.calcular_prioridade(e)
 8:       se k < +∞ então                                   {se o arco converge}
 9:          a.insere((k, e))
10:       fim
11:    fim
12:    fila: q ← construir_fila(a)                         {Fila com prioridade.}
13:    enquanto q.comprimento() > 3 fazer
14:       (k, e) ← q.remover_mínimo()
15:       p ← w.anterior(e)
16:       n ← w.seguinte(e)
17:       w.rodar_aresta(e)
18:       k_p ← w.calcular_prioridade(p)
19:       q.inserir_actualizar_ou_remover((k_p, p))
20:       k_n ← w.calcular_prioridade(n)
21:       q.inserir_actualizar_ou_remover((k_n, n))
22:    fim                                                 {Frente de onda tem 3 arcos.}
23:    (k, e) ← q.remover_mínimo()                         {Escolher um dos três eventos.}
24:    w.rodar_aresta(e)
25:    V.remover_par_de_arestas(w)
26: fim
```

cente, então este evento é actualizado para a prioridade do novo evento, caso o novo arco convirja para um vértice, ou então o evento é simplesmente removido da fila de eventos. Se o arco adjacente não tinha um evento associado e se se verifica que o novo arco gera um novo evento, então este é simplesmente inserido. Estas várias opções estão encapsuladas na função inserir_actualizar_ou_remover (linhas 19 e 21). Note-se que, apesar do escalonamento condicionado de eventos, o varrimento percorre (e remove) os $m-2$ vértices do interior do grafo G_s (sendo m o número de locais vizinhos), processando igual número de eventos-círculo.

A adição de arestas auxiliares permite que o varrimento se processe como se não existissem regiões ilimitadas. Na remoção de um local pertencente ao invólucro-convexo, a frente de onda percorre dois vértices auxiliares, extremos da aresta auxiliar. Uma aresta auxiliar pode ser objecto de rotação (tal como as arestas regulares), deixando um vértice auxiliar para trás. Também pode acontecer que uma aresta auxiliar faça parte do par de arestas finais. Em qualquer dos casos, a presença de uma aresta auxiliar apenas altera as condições de desaparecimento de um arco e o cálculo da prioridade respectiva.

O algoritmo de remoção de locais no plano apresenta a mesma complexidade espacial e

temporal que o algoritmo equivalente na esfera. A justificação segue os mesmos argumentos apresentados na Proposição 5 (página 77) notando que a adição de arcos auxiliares não altera o tamanho da frente de onda e o número de eventos processados em mais de uma unidade.

6.2.2 Algoritmo de inserção

O Algoritmo 5 (página 81), que descreve a inserção de um local num diagrama de Voronoi esférico, aplica-se igualmente à inserção de um local num diagrama de Voronoi planar. A inserção no plano apenas difere do equivalente esférico na implementação da função em_conflito(v, s), que agora é estendida para contemplar o caso em que v é um vértice auxiliar. Generalizando, define-se como círculo vazio máximo de um vértice auxiliar v_∞ (extremo de uma aresta infinita) o semi-plano limitado pela recta que passa pelos locais que partilham a aresta que contém v_∞. Como atrás indicado, a operação de rotação de aresta não necessita de qualquer adaptação uma vez que, topologicamente, as arestas auxiliares são indistintas das arestas regulares.

Portanto, o algoritmo de inserção de locais no plano apresenta a mesma complexidade espacial e temporal que o algoritmo equivalente na esfera.

6.3 Interpolação por vizinhos naturais

Nesta secção vamos ver como o algoritmo de inserção pode ser facilmente transformado num algoritmo eficiente de cálculo da interpolação por vizinhos naturais.

A interpolação por vizinhos naturais é um método de interpolação espacial desenvolvido por Sibson [58]. Baseia-se no diagrama de Voronoi de um conjunto de pontos, aplicável a distribuições irregulares de pontos. Constitui uma alternativa a métodos de interpolação simples tais como a interpolação pelo vizinho mais próximo ou a interpolação pela média ponderada dos k vizinhos mais próximos. Métodos mais elaborados dependem habitualmente de parâmetros que, normalmente, derivam de pressupostos sobre a natureza da função de interpolação, restringindo a sua aplicabilidade. Embora possam ter melhor desempenho, o constrangimento imposto pela parametrização descarta-os como métodos universais de interpolação. Em claro contraste, a interpolação por vizinhos naturais não requer qualquer parametrização, nem impõe qualquer constrangimento nas posições dos pontos a interpolar.

Uma forma alternativa de definir a interpolação por vizinhos naturais é como o método que calcula a superfície de área mínima que passa por todos os pontos, tal como a interpolação linear na recta. De facto, no diagrama de Voronoi de pontos na recta, a região de Voronoi de cada local é delimitada pelos pontos médios entre locais consecutivos. Aplicando este método ao caso linear, a interpolação em qualquer posição identifica dois vizinhos naturais que definem pesos que coincidem com a comum interpolação linear. Portanto, a interpolação por vizinhos naturais pode ser vista como uma generalização da interpolação linear em $\mathbb{R}$. Naturalmente, este método é generalizável a $\mathbb{R}^n$, embora aqui seja apenas discutida a sua implementação no plano e na superfície da esfera.

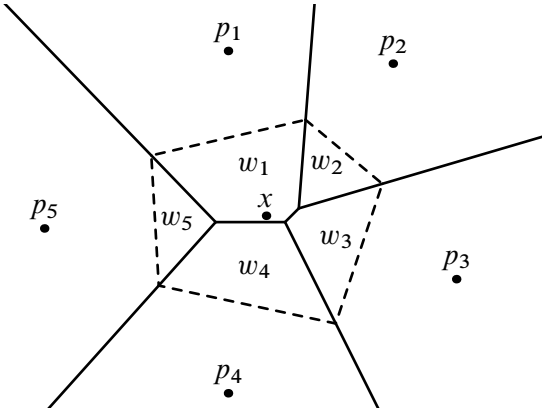


Figura 6.14: Interpolação por vizinhos naturais. A interpolação em x é dada pela média ponderada dos vizinhos naturais, $p_1, \ldots, p_5$, com os pesos $w_1, \ldots, w_5$, respectivamente.

Uma aplicação habitual da interpolação em conjuntos de pontos irregulares é a re-interpolação de dados numa malha regular, como passo intermédio da construção de um mapa de linhas de contorno de valor constante, que possuem a vantagem de evidenciar estruturas espaciais dificilmente visíveis nas medidas pontuais. A interpolação por vizinhos naturais possibilita a construção eficiente de mapas de contornos, calculando uma aproximação linear de medidas pontuais.

A interpolação por vizinhos naturais é feita do seguinte modo. Seja P um conjunto de pontos p_i onde é conhecida uma medida $f(p_i)$ e seja $V(P)$ o diagrama de Voronoi de P. A interpolação num ponto $x \notin P$ é obtida inserindo x no diagrama $V(P)$, o que causa a redução da região de alguns locais de P. Os locais cujas regiões foram reduzidas são os *vizinhos naturais* de x em $V(P)$. Seja então $V'(x)$ a região de Voronoi de x no diagrama $V(P \cup \{x\})$, enquanto que $V(p_i)$ denota a região de p_i em $V(P)$. A interpolação em x é dada por uma média das medidas dos vizinhos naturais, ponderada pela área *roubada* a cada vizinho natural, relativa à área de $V'(x)$. Assumindo, sem perda de generalidade, que p_i, com $i = 1, \ldots, m$, são os vizinhos naturais de x, então a estimativa $F(x)$ de $f(x)$ é dada por:

$$F(x) = \sum_{i=1}^{m} w_i \ f(p_i), \tag{6.10}$$

onde

$$w_i = \frac{\text{área}\,(V'(x) \cap V(p_i))}{\text{área}(V'(x))},$$

tal como ilustrado na Figura 6.14. No plano, este método apenas se aplica a pontos que gerem uma região de Voronoi finita, isto é, pontos contidos no invólucro-convexo de P. Na esfera, aplica-se a todo o domínio. Por outro lado, este método é contínuo em todo o domínio excepto para $x \in P$, em que a estimativa é dada pela medida conhecida.

Uma implementação directa deste método implicaria a inserção de x no diagrama $V(P)$, para determinar $V'(x)$ e identificar os vizinhos naturais, seguida da remoção de x e, finalmente, da intersecção de $V'(x)$ com cada uma das regiões $V(p_i)$ dos vizinhos naturais. Pos-

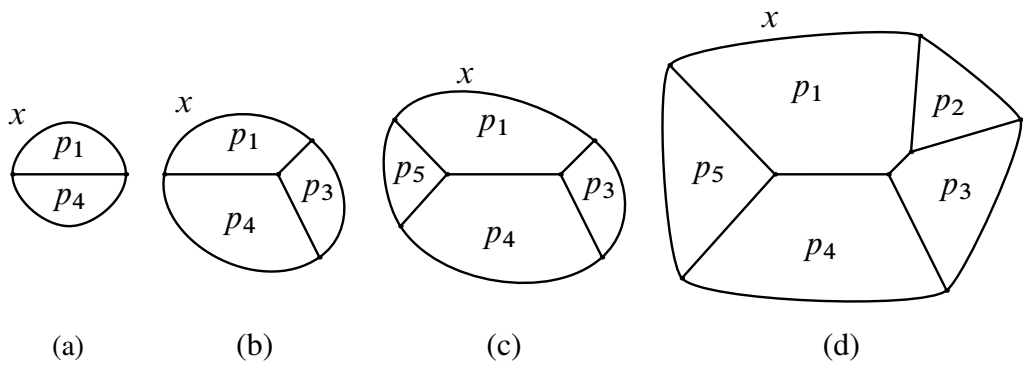


Figura 6.15: Construção da vizinhança natural de x (c.f. Figura 6.14).

sivelmente, seria exequível evitar a operação de remoção, estendendo a implementação do diagrama de Voronoi com um mecanismo que permitisse desfazer (em tempo constante) a última inserção. No entanto, não há forma de evitar a intersecção de polígonos, operação que tem um custo linear no número total de elementos dos dois polígonos [62]. A solução que se propõe a seguir evita esta operação, construindo directamente as intersecções entre polígonos, por adaptação do algoritmo de inserção.

O algoritmo de inserção de um local x num diagrama de Voronoi V começa por inserir um par de arestas (dando início a uma região), seguindo-se uma sequência de rotações de arestas. Durante o varrimento que implementa a inserção, é percorrida a parte de V a remover (vértices e arestas) para dar lugar à região de x. A ideia da adaptação para a construção da vizinhança natural é usar a mesma sequência de operações (com alterações) na construção de um segundo diagrama de Voronoi, preenchido com uma cópia da parte de V que seria removida numa inserção de x. Desta forma, obtém-se um diagrama que apenas contém a vizinhança natural $V_n(x)$ de x, isto é, contém um polígono por cada vizinho natural de x, com a forma de $V'(x) \cap V(p_i)$, delimitada por um polígono com a forma de $V'(x)$. Uma vez construída a vizinhança natural, fica trivial calcular o valor da interpolação em x, bastando calcular a área de cada polígono em $V_n(x)$.

A construção de $V_n(x)$ é feita realizando uma falsa inserção de x em $V(P)$, apenas para se descobrir a sequência de operações da inserção. Essa sequência é então usada, com adaptações, na construção de $V_n(x)$, copiando-se também a parte de $V(P)$ que seria removida com a inserção. As adaptações decorrem da diferença entre construir um diagrama novo e modificar um diagrama existente. A Figura 6.15 ilustra todo o procedimento. O diagrama $V_n(x)$ é inicializado com a inserção do local x. A operação de adição de dois arcos numa aresta de $V(P)$ que separa dois locais, p_i e p_j, é substituída pela adição de duas regiões, as primeiras de $V_n(x)$, associadas aos mesmos locais e delimitadas por três arestas: a aresta entre p_i e p_j, copiada de $V(P)$, e duas entre cada local e o local x (ver Figura 6.15a). A rotação de uma aresta, que acontece quando a frente de onda ganha um arco, gerada por um local p_k, é substituída pela adição de uma nova região a $V_n(x)$, associada a p_k, copiando um vértice e duas arestas de $V(P)$, e adicionando uma aresta entre p_k e x (ver Figuras 6.15b a 6.15d).

Com este procedimento obtém-se um diagrama que é, de certa forma, singular; nenhum

dos locais está contido no interior da região a que está associado. Apesar desta peculiaridade, o diagrama $V_n(x)$ é topologicamente válido, com cada vértice correctamente associado aos três locais que o definem. As vantagens deste método são óbvias: o diagrama $V(P)$ não é modificado de forma alguma e, uma vez calculado $F(x)$, basta descartar $V_n(x)$ para completar o processo.

Capítulo 7

Resultados experimentais

Este capítulo apresenta resultados experimentais obtidos na execução dos sete algoritmos discutidos nos três capítulos anteriores. O objectivo é avaliar o custo dos algoritmos desenvolvidos, analisando o desempenho dos algoritmos de varrimento esférico e dos algoritmos equivalentes no plano e compará-los com o desempenho de algoritmos distribuídos livremente para executar a mesma tarefa.

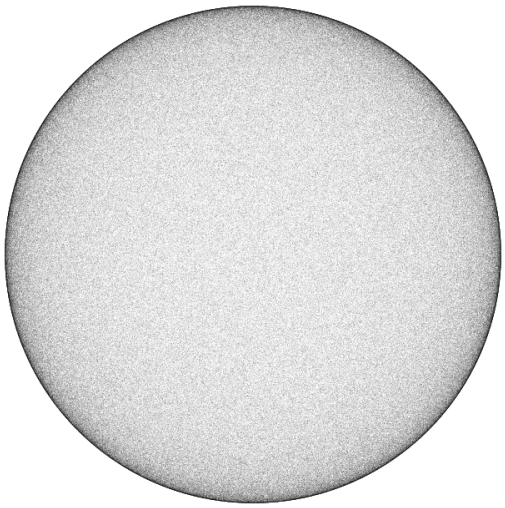
Os testes experimentais processaram os conjuntos de dados apresentados na secção 7.1. Detalhes de implementação dos algoritmos de varrimento são discutidos na secção 7.2, sendo os algoritmos de construção do diagrama de Voronoi (no plano e na esfera) comparados na secção 7.3. O novo algoritmo de construção do diagrama de Voronoi esférico é comparado com algoritmos equivalentes (disponibilizados livremente) na secção 7.4. Por fim, na secção 7.5 são comparados os algoritmos de edição (no plano e na esfera).

Todo o código fonte utilizado nestes testes (tanto o código dos algoritmos deste trabalho como o código das bibliotecas externas) foi compilado com o compilador GNU GCC (versão 4.4.5), com optimizações, capaz de compilar código em C, em C++ e em Fortran. Todos os programas foram executados num computador equipado com um processador Intel Xeon E5160, trabalhando a 3.00 GHz, com 4 GB de memória a 1333 MHz.

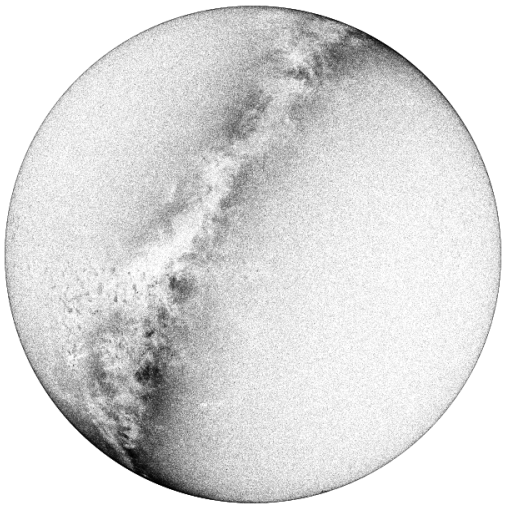
7.1 Conjuntos de dados

Os testes experimentais consistem em executar cada algoritmo com o mesmo conjunto de dados, medindo o tempo de execução e o consumo de memória. Diferentes conjuntos de dados testam os algoritmos de diferentes maneiras. Por este motivo, foram utilizados quatro tipos diferentes de conjuntos de pontos, ilustrados na Figura 7.1.

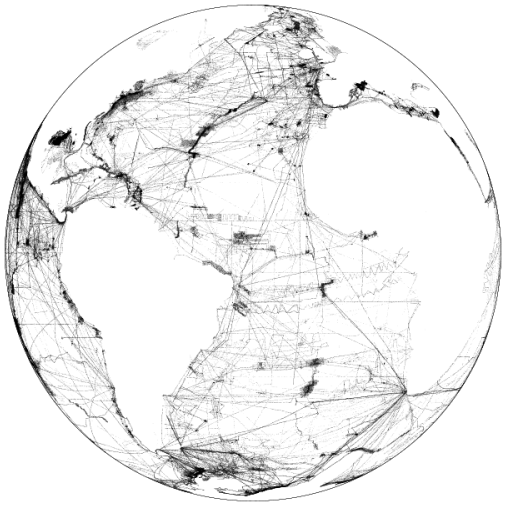
Na forma mais simples, são usados conjuntos de pontos em posições aleatórias (denotados por A), segundo uma distribuição uniforme, tanto no plano como na esfera (c.f. Figura 7.1A). No plano, os pontos são distribuídos no domínio $[0,1] \times [0,1]$. Por imposição da memória RAM disponível (4 GB), os conjuntos de dados foram limitados a um máximo de $2^{24.3}$ pontos ($\approx 21 \times 10^6$ pontos).



(A) Posições aleatórias



(E) Posições estelares



(O) Posições oceânicas

(P) Percursos de navios

Figura 7.1: Conjuntos de dados com 2^{20} pontos (em projecção ortográfica).

No domínio esférico, foram também utilizados conjuntos extraídos de dados reais, com o objectivo de determinar o impacto da não-uniformidade na execução dos algoritmos.

O segundo tipo (denotado por E) contém posições de estrelas, extraídas da quarta edição do catálogo astrográfico CCD do Observatório Naval dos Estados Unidos (UCAC4) [64] (c.f. Figura 7.1E). O UCAC4 é um catálogo de estrelas que preenche toda a esfera celeste, contendo aproximadamente 114×10^6 estrelas, completo até à magnitude 16. Os conjuntos de dados foram gerados por uma selecção aleatória das 21×10^6 estrelas mais brilhantes.

Os restantes tipos de dados foram gerados a partir da base de dados geofísicos marítimos GEODAS, compilada pelo centro de dados geofísicos dos Estados Unidos (NOAA) [35, 56]. Tratam-se de dados adquiridos por sensores de medida a bordo de navios ao longo dos trajectos que percorrem. Cada navio adquire um conjunto elevado de registos. Em cada posição, são adquiridas (habitualmente) várias medidas geofísicas (batimetria, campo gravítico, salini-

dade, etc.), interessando-nos apenas a localização associada. Foram usados os dados digitais registados desde 1972, num total de 2572 percursos de navios, contendo cerca de 48×10^6 posições (não necessariamente distintas). Estas localizações foram agrupadas de duas formas. Nos conjuntos do tipo O, foram seleccionadas localizações de uma permutação aleatória de toda a base de dados (essencialmente cobrindo os *oceanos*, c.f. Figura 7.1O). Os conjuntos do tipo P foram construídos agrupando uma selecção aleatória de percursos inteiros de navios até perfazer o número de posições pretendido (possivelmente truncando o último percurso de navio). Estes últimos conjuntos caracterizam-se pela elevada densidade de pontos ao longo dos percursos dos navios (c.f. Figura 7.1P).

Para cada um dos tipos mencionados, foram gerados dez conjuntos de dados de igual tamanho, para os tamanhos $2^{15.9}, 2^{16.2}, 2^{16.5}, \ldots, 2^{24.3}$. Ou seja, geraram-se 10×28 conjuntos no total. Os resultados apresentados à frente são uma média das dez medições obtidas pela execução com os conjuntos do mesmo tipo e tamanho. Os pontos de cada conjunto de dados são todos distintos e definidos em coordenadas Cartesianas.

7.2 Implementação dos algoritmos

Os algoritmos de varrimento foram implementados na linguagem C, com os cálculos numéricos operados com aritmética de vírgula flutuante, em precisão dupla. Desta forma, os resultados de cálculos numéricos são contaminados por erros devidos a arredondamentos e, no pior caso, por cancelamento catastrófico. Estas imprecisões afectam a implementação dos algoritmos de varrimento de duas formas:

- no cálculo de prioridades, o que pode distorcer a ordem relativa entre eventos;
- no cálculo das condições que determinam o escalonamento de eventos-círculo e eventos-rotação.

Uma forma de atenuar as imprecisões numéricas por completo seria recorrer a operações numéricas de precisão arbitrária [31, 13, 63]. Naturalmente, as coordenadas dos locais são especificadas por números com precisão fixa. Logo, uma maior precisão nos cálculos numéricos permitiria a construção exacta pelos algoritmos de varrimento. Embora exequível, esta solução teria duas desvantagens: maior complexidade na implementação e maior custo computacional.

Por estes motivos, optou-se por realizar os cálculos em vírgula flutuante, como indicado, mas seguindo a estratégia de garantir um resultado topologicamente correcto, mesmo na presença de imprecisões numéricas [60, 61]. Na prática, esta estratégia implica que a lógica da implementação apenas permite a construção de um diagrama topologicamente correcto, que se resume em garantir que (para diagramas de três ou mais locais):

- qualquer região é delimitada por duas ou mais arestas,
- duas regiões que são adjacentes só partilham uma única aresta.

No que respeita aos cálculos numéricos, foi seguida a estratégia de maximizar a exactidão dos resultados perante a precisão disponível. Por exemplo, o cálculo da prioridade de um evento-círculo implica o cálculo do centro e do raio de um círculo circunscrito a três pontos, que é função da área do triângulo formado pelos mesmos pontos. Esta área é calculada pelas coordenadas dos pontos relativas a um dos vértices do triângulo $\triangle abc$, que é usado como origem. As expressões (4.8), (4.16), (5.12), (6.4) e (6.7) calculam a prioridade de um evento-círculo escolhendo o ponto b como origem. No entanto, qualquer um dos vértices do triângulo pode desempenhar a função de origem. Por ser a escolha que maximiza a exactidão do cálculo do círculo, no cálculo do círculo, a origem é atribuída ao canto cujo ângulo interno é mais próximo de $\pi/2$. Em resumo, na implementação é dada prioridade à topologia, colocando a exactidão geométrica em segundo plano. Um aumento da precisão numérica teria o efeito de aproximar o diagrama obtido do diagrama exacto.

7.3 Comparação dos algoritmos de construção por varrimento

Nesta secção são comparados os algoritmos de construção do diagrama de Voronoi pela técnica de varrimento, ou seja, o algoritmo de Fortune, o algoritmo de varrimento circular do plano e o algoritmo de varrimento esférico. O objectivo desta comparação é avaliar o impacto do custo dos eventos-rotação (comparando os dois algoritmos de varrimento planar) e avaliar o custo acrescido pelo processamento de dados tridimensionais (comparando os dois varrimentos circulares).

Foram codificadas as três versões dos algoritmos de varrimento, denominados de *linear*, *circular* e *esférico*, para o algoritmo de Fortune, o varrimento circular e o varrimento esférico, respectivamente. Sempre que possível, foi partilhado o código entre implementações, por forma a tornar os tempos de execução comparáveis. Em particular, as estruturas de dados relevantes são as mesmas nas três implementações. A árvore binária de pesquisa foi implementada por uma árvore vermelha-preta e a fila com prioridade por um *heap* binário e um vector. A principal diferença entre as várias implementações reside no cálculo de prioridades (em particular, o cálculo do círculo circunscrito a três pontos) e na função de ordenação da frente de onda. Estes algoritmos foram testados apenas com conjuntos do tipo A (posições aleatórias).

A Figura 7.2 apresenta os valores médios do tempo de execução normalizado, para os três algoritmos indicados, em função do número de locais n. Em qualquer dos algoritmos, observa-se que o tempo de execução normalizado (tempo$/n$) aumenta com o número de locais, justificável pela complexidade temporal dos algoritmos e pela limitada capacidade da memória *cache*. Com o aumento do volume de dados (e do tamanho das estruturas de dados), reduz-se a taxa de acertos na memória *cache*, aumentando o custo médio de um acesso a memória.

Porém, também se observa que os tempos de execução relativos entre os vários algo-

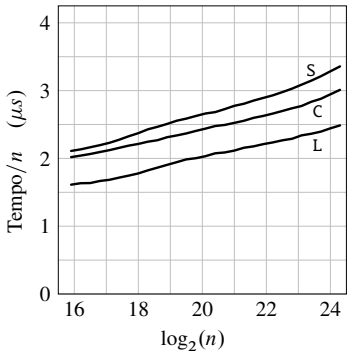


Figura 7.2: Tempos médios de execução para os algoritmos de varrimento *linear* (L), *circular* (C) e *esférico* (S), normalizado pelo número de locais n.

Tabela 7.1: Medidas de desempenho dos algoritmos de varrimento.

algo.	mem ($/n$)	tempo relativo	eventos escalonados ($/n$)	eventos falsos ($/n$)	eventos-rotação ($/\sqrt{n}$)	círculos calculados ($/n$)
L	130 B	1.00×	4.604 ± 0.002	1.604 ± 0.002	–	3.604 ± 0.002
C	130 B	1.21×	4.604 ± 0.002	1.603 ± 0.003	0.907 ± 0.007	3.604 ± 0.002
S	139 B	1.32×	4.606 ± 0.005	1.605 ± 0.006	0.842 ± 0.006	4.047 ± 0.009

ritmos, usando o algoritmo *linear* como referência, são aproximadamente constantes. Os resultados deste teste estão sumariados na Tabela 7.1, apresentando valores médios (e, para alguns parâmetros, o desvio padrão) normalizados por n.

A comparação dos resultados entre os algoritmos *linear* e *circular* mostra que o número de eventos escalonados e eventos falsos é idêntico, tal como é o número de círculos calculados (que tem um peso significativo no custo do cálculo de prioridades). Também se observa que o número de eventos-rotação, que apresenta um valor próximo de $0.91\sqrt{n}$, é desprezável face ao número total de eventos processados. Logo, a diferença nos tempos de execução (relativos) entre os algoritmos *linear* e *circular* deve-se necessariamente ao custo da função de ordenação, mais dispendiosa no varrimento circular.

Quanto ao algoritmo *esférico*, a principal diferença em relação ao algoritmo *circular* é o aumento do número de círculos calculados. Este aumento deve-se à imposição de um horizonte finito, que caracteriza o varrimento esférico. O cálculo de uma prioridade implica o cálculo de um círculo, mesmo que seja para descobrir que o evento está para além do horizonte, nunca sendo escalonado. Este custo extra e a necessidade de guardar uma terceira coordenada justificam o aumento de uso de memória e tempo de execução relativamente ao algoritmo *circular*.

7.4 Comparação dos algoritmos de construção do diagrama esférico

Um diagrama de Voronoi pode ser obtido, indirectamente, construindo uma estrutura de dados dual como a triangulação de Delaunay ou, no caso particular da esfera, o invólucro-convexo tridimensional. Nesta secção, o algoritmo de construção do diagrama de Voronoi esférico é comparado com quatro algoritmos alternativos que calculam o invólucro-convexo de pontos em 3D, cujos códigos-fonte ão disponibilizados livremente. Adicionalmente, também é comparado com um algoritmo incremental de construção da triangulação de Delaunay na superfície da esfera.

A *Computational Geometry Algorithms Library* (CGAL) [11] é uma biblioteca de *software* que aglomera uma extensa colecção de algoritmos do domínio da geometria computacional. O desenvolvimento da biblioteca CGAL teve início em 1996 no seio de um consórcio formado por universidades e centros de investigação, tendo por objectivo a implementação de algoritmos geométricos com especial atenção à correcção e eficiência. Desde então, a CGAL tem sido actualizada frequentemente encontrando-se, em Fevereiro de 2013, na versão 4.1. São usados dois algoritmos desta biblioteca: o *CGAL-triangulation* e o *CGAL-Delaunay*.

O CGAL-triangulation constrói uma triangulação de pontos em 3D [7], sem impor qualquer restrição geométrica, apenas garantindo a validade da triangulação resultante. O CGAL-Delaunay constrói uma triangulação de pontos em 3D, garantindo as restrições de uma triangulação de Delaunay. Note-se que, para este algoritmo em especial e por forma a maximizar o desempenho, a construção foi iniciada com a inserção de um ponto adicional na origem, necessário para evitar um número excessivo de testes de colinearidade de cinco pontos [10]. Após a construção da triangulação, é extraído o invólucro-convexo que, para o caso de pontos na superfície da esfera, coincide com a triangulação de Delaunay esférica. Ambos os algoritmos seguem o método de construção incremental. Primeiro, os locais são ordenados espacialmente, segundo a ordem induzida por uma curva de Hilbert em 3D. Depois, a localização do triângulo onde é inserido o próximo local é feita por um passeio na triangulação, com início no último local inserido. A ordenação espacial garante que, em média, a localização se faz por um passeio curto, crucial para a eficiência do algoritmo. A razão da escolha destes dois algoritmos deve-se às suas especificidades. O CGAL-triangulation foi seleccionado por ser a opção mais rápida para a construção do invólucro-convexo. O CGAL-Delaunay foi seleccionado por ser o único que implementa uma estrutura de dados dinâmica, apropriado para a comparação com os algoritmos de edição.

Na biblioteca CGAL, os algoritmos são parametrizados com um "núcleo" geométrico que permite, por exemplo, escolher entre uma implementação que privilegia a rapidez de execução e uma implementação que privilegia a correcção dos resultados. Para estes testes foi escolhido o núcleo denominado de *Exact_predicates_inexact_constructions_kernel*, por ser o mais usual neste contexto e por garantir, nos algoritmos seleccionados, a construção exacta do invólucro-convexo. Com este núcleo, os cálculos são realizados com números em

vírgula flutuante, recorrendo a precisão arbitrária, quando necessário para garantir a exactidão dos resultados. A CGAL está implementada na linguagem C++.

Ainda no âmbito da biblioteca CGAL, é importante referir que o algoritmo incremental de construção da triangulação de Delaunay no plano foi adaptado ao domínio esférico [10], sendo reportado como mais rápido que o CGAL-Delaunay. No entanto, a implementação ainda não está disponível na biblioteca CGAL.

A *Qhull* [5] é uma biblioteca que implementa o algoritmo *quickhull* de construção do invólucro-convexo em dimensão arbitrária [50, 4]. A primeira versão data de 1993 tendo sido, desde então, apenas actualizada para correcção de erros. A última versão é a 2012.1 (de 2012). Este algoritmo implementa uma estratégia de divisão e conquista, embora adicionando um ponto de cada vez. Na inserção de um novo ponto, há que determinar se este está *acima* ou se está *abaixo* de uma face. Se um ponto está acima de uma face, diz-se que está em *em conflito* com essa face. O ponto localiza-se no exterior do invólucro-convexo se está em conflito com uma ou mais faces. Caso contrário, localiza-se no interior do invólucro-convexo (sendo descartado). Ao se adicionar um local, são removidas todas as faces com que este está em conflito, formando novas faces com as arestas que estão na fronteira entre as faces que *estão* em conflito e as faces que *não estão* em conflito. Os pontos associados às faces removidas são agora redistribuídos pelas faces recém criadas. Resta saber como são determinadas as faces em conflito. O algoritmo quickhull começa por determinar os locais com maior e menor coordenada segundo cada eixo, com os quais é construído o invólucro-convexo inicial. Os restantes locais são associados a uma das faces com que estão em conflito (ou, se não estão em conflito com nenhuma face, descartados por pertencerem ao interior do invólucro-convexo). Depois, para cada face com pelo menos um ponto em conflito, é seleccionado o ponto que lhe está mais distante e inserido no invólucro-convexo. Esta operação envolve percorrer as face vizinhas da face inicial, para se determinar o conjunto de faces em conflito com o novo local e as arestas com que se constroem as novas faces. O processo é iterado enquanto existirem faces com pontos em conflito, isto é, locais por adicionar. No Qhull, os cálculos são feitos com aritmética de vírgula flutuante (com precisão dupla) o que pode conduzir a que um ponto seja erradamente classificado como ponto interior ao invólucro-convexo. Para colmatar estes casos, pontos em que não seja seguro pertencerem ao interior do invólucro-convexo, por estarem muito próximos de uma face, são *fundidos* no que se denomina de *faces espessas*. Como o objectivo é a obtenção do diagrama de Voronoi esférico, foi activada a opção que força a triangulação adicional das faces espessas. O algoritmo Qhull está codificado na linguagem C.

O *Hull* [16] implementa o algoritmo incremental de Clarkson e Shor para a construção do invólucro-convexo numa dimensão arbitrária [17]. Este algoritmo segue uma estratégia semelhante à do algoritmo quickhull, registando relações de conflito entre pontos e faces. Só que, neste caso, é mantida uma estrutura de dados mais complexa, denominada de *grafo de conflitos*, onde cada ponto mantém uma lista de todas as faces com que está em conflito e cada face mantém uma lista de todos os pontos com que está em conflito. Primeiro, são seleccionados três pontos não co-planares para construir o invólucro-convexo inicial e os restantes pontos são adicionados por uma ordem aleatória. Depois, na inserção de um ponto, o grafo de con-

Tabela 7.2: Média e desvio padrão do consumo de memória (em *bytes*) por local para os algoritmos de construção.

Conj. dados	algoritmo varrimento	CGAL Delaunay	CGAL Triangulação	Qhull	Hull	STRIPACK
A	139 ± 1	348 ± 13	499 ± 6	491 ± 5	1237 ± 111	111 ± 4
E	139 ± 1	348 ± 13	499 ± 6	517 ± 43	1236 ± 110	111 ± 4
O	139 ± 1	348 ± 13	505 ± 6	606 ± 39	1271 ± 100	111 ± 4
P	139 ± 1	348 ± 13	531 ± 7	626 ± 2	9567 ± 1703	111 ± 4

flitos identifica todas as faces a remover e, por acréscimo, cada face removida identifica todos os pontos a re-associar às novas faces. Após a inserção de um ponto, o grafo é actualizado e o processo iterado até à exaustão. Neste algoritmo, os cálculos numéricos são feitos de forma exacta, representando as coordenadas dos pontos por números inteiros e recorrendo a rotinas de precisão arbitrária [15]. Adicionalmente, é imposta uma representação das coordenadas por inteiros de 25 *bits*. Foi usada a versão 1.0 (e única disponível), datada de 1995. Reconhecidamente, não é uma implementação optimizada para eficiência, mas interessante por ter sido objecto de comparação habitual com outros algoritmos. O algoritmo está codificado na linguagem C.

O *STRIPACK* [52] implementa uma adaptação à esfera do algoritmo incremental de construção da triangulação de Delaunay no plano (discutido na secção 3.2). Este algoritmo implementa uma estratégia alternativa da procura do triângulo que contém o próximo ponto a inserir na triangulação (discutida na secção 3.2). Os três primeiros locais formam um triângulo esférico inicial. Os restantes locais são associados ao vértice do triângulo que lhes está mais próximo. Nas inserções subsequentes, a procura do triângulo inicial que contém o ponto é feita pesquisando os triângulos incidentes ao (conhecido) vértice mais próximo. Os locais associados aos vértices modificados com a inserção podem agora estar mais próximos de outros vértices, pelo que são analisados e, se necessário, re-associados a um vértice recém inserido. Inicialmente, este algoritmo foi codificado na linguagem FORTRAN 77, em 1993, usando cálculos numéricos com precisão simples. Nestes testes, foi usada uma re-codificação do código inicial na linguagem Fortran 90,[1] datada de 2007, usando cálculos numéricos com precisão dupla. O recurso a cálculos com precisão dupla mostrou-se imprescindível para a obtenção de resultados correctos para a maioria dos conjuntos de dados.

Para cada algoritmo, foi medido o tempo de execução e o consumo de memória na construção do diagrama de Voronoi (ou invólucro-convexo) para cada tipo de conjunto de pontos. A Figura 7.3 apresenta o valor médio dos tempos de execução, normalizado pelo número de pontos. As linhas a tracejado indicam os casos em que um algoritmo falhou em produzir o invólucro-convexo de todos os pontos, atribuindo alguns pontos ao seu interior. As execuções foram interrompidas assim que a memória RAM disponível foi esgotada, o que justifica algumas linhas mais curtas. Os consumos de memória (normalizados por local), que não variam significativamente com o tamanho do conjunto de dados, são sumariados na Tabela 7.2.

[1] Disponível em `http://people.sc.fsu.edu/~jburkardt/f_src/stripack/stripack.html`.

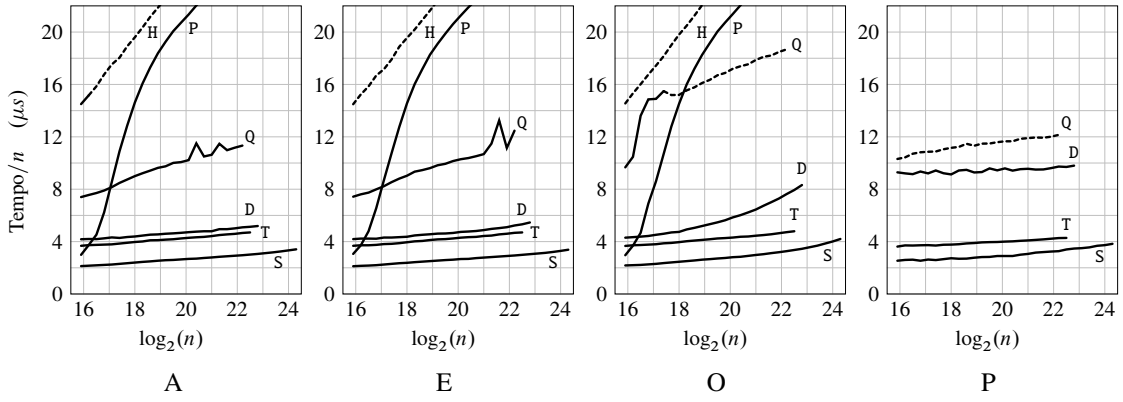


Figura 7.3: Tempos médios de execução para os algoritmos de construção (para n locais). A letra junto a cada curva identifica o algoritmo respectivo: S–algoritmo de varrimento, H–Hull, Q–Quickhull, D–CGAL-Delaunay, T–CGAL-triangulation, P–STRIPACK.

O Hull é o mais lento de todos e o que consome mais memória. O recuso a aritmética de precisão arbitrária parece penalizar o desempenho global do algoritmo. Para satisfazer os requisitos, as coordenadas dos pontos foram convertidas para inteiros de 25-*bits* (descartando colisões de pontos surgidas após a conversão). Com excepção dos conjuntos A de menor tamanho, este algoritmo falha em produzir um resultado correcto (ao não incluir todos os pontos no resultado), que se deve ao limite imposto na precisão das coordenadas. Adicionalmente, o desempenho deste algoritmo decresce abruptamente com os conjuntos de dados P, com um aumento de 200 vezes nos tempos de execução (não apresentados no gráfico) e com um aumento de seis vezes no consumo de memória.

O STRIPACK apresenta um tempo de execução atípico. Comparativamente aos outros algoritmos, é rápido para conjuntos de pontos pequenos mas fica muito mais lento com o aumento do número de pontos. A justificação mais provável para o aumento de custo com o número de pontos deve-se à re-associação de locais por novos vértices, após cada inserção. Até que seja inserido na triangulação, um local é associado a vários vértices, numa sucessão que se aproxima monotonamente do próprio local. Porém, é de esperar que os primeiros vértices desta sucessão estejam distantes do que será o vértice mais próximo do local respectivo, penalizando fortemente o custo de cada inserção. Nos conjuntos de dados P, foram medidos tempos de execução cerca de 25 vezes superiores aos tempos com os restantes conjuntos de dados (não apresentados no gráfico). Como os dados P estão ordenados segundo os percursos dos navios, um local recém inserido tem uma forte probabilidade de passar a ser o vértice mais próximo de muitos locais ainda não inseridos, causando uma avalanche de re-associações em cada inserção. Adicionalmente, foram obtidos resultados incorrectos com os conjuntos de dados P superiores a 10^6 pontos. Relativamente ao consumo de memória, este algoritmo é o que apresenta o menor custo, consequência da simplicidade da estratégia de procura.

O Qhull beneficia da eficiência da estratégia de divisão e conquista do algoritmo *quickhull*, auxiliado pelo recurso a faces espessas. Nos testes efectuados, verificou-se que o mecanismo das faces espessas (e posterior triangulação) maximiza o número de casos em que se obtém

um resultado correcto. Porém, quando este passo adicional é despoletado, observa-se uma duplicação dos tempos de execução (aproximadamente), o que justifica alguns dos máximos locais nos conjuntos A e E. Com os conjuntos mais exigentes (O e P), o Qhull quase não consegue produzir um resultado correcto, ao excluir pontos do invólucro-convexo resultante. As maiores dificuldades são reportadas no conjunto P, em que são descartados cerca de duas ordens de grandeza mais pontos para o interior do invólucro-convexo e geradas faces duas vezes mais espessas que com os conjuntos O, o que justifica as diferenças em tempos de execução (e em correcção, para os conjuntos mais pequenos). Claramente, a falha na geração de um resultado correcto deve-se à limitação do cálculo de primitivas numéricas com precisão fixa e à ineficácia do mecanismo de faces espessas em resolver os casos mais exigentes.

No que respeita à biblioteca CGAL, observa-se que os tempos de execução do CGAL-Delaunay são sempre superiores aos do CGAL-triangulation, sendo as diferenças notórias para os conjuntos de dados O e P. Dada a semelhança entre estes dois algoritmos, a diferença de desempenho só pode ser justificada pela diferença na inserção de um ponto na triangulação, com maior custo na CGAL-Delaunay. Em particular, nos resultados com os conjuntos de dados O, observa-se uma crescente penalização no desempenho à medida que, com o aumento do número de pontos, estes conjuntos de dados se assemelham aos conjuntos de dados P. Estes resultados também mostram como uma forte não-uniformidade na distribuição dos pontos condiciona o desempenho de um algoritmo baseado numa estratégia de construção incremental. Por outro lado, o algoritmo CGAL-triangulation não exibe nenhuma variação apreciável com a distribuição espacial dos pontos. Relativamente ao consumo de memória tem-se o inverso, com o CGAL-triangulation a ultrapassar sempre o CGAL-Delaunay.

O algoritmo de varrimento mostrou-se imbatível no desempenho temporal e com um desempenho espacial muito próximo do melhor. Apesar da extrema não-uniformidade dos conjuntos de dados P e dos maiores conjuntos de dados O causar um aumento do número de falsos eventos, os tempos de execução apenas registam um ligeiro aumento, sem qualquer impacto no consumo de memória. Em particular, a implementação da frente de onda por uma árvore binária equilibrada parece adaptar-se eficazmente às especificidades da distribuição espacial dos conjuntos de pontos. O baixo consumo de memória também se deve às propriedades do método do varrimento, ao manipular, em simultâneo, apenas uma fracção do diagrama.

A construção do diagrama de Voronoi esférico por intermédio da construção de um invólucro-convexo revelou duas dificuldades: é sensível à precisão usada na representação das coordenadas dos pontos (como se observou com os resultados do algoritmo Hull) e requer uma implementação com primitivas geométricas exactas (como se observou com os resultados do algoritmo Qhull).

Por outro lado, a variabilidade nos tempos de acesso a memória causadas pela memória *cache* favorecem as implementações que apresentam um padrão de acessos localizados. É o que acontece com os algoritmos CGAL-triangulation e CGAL-Delaunay (com a estratégia de inserção ao longo de uma curva de Hilbert) e também com o algoritmo de varrimento (ao manipular apenas a parte do diagrama em torno da frente de onda), o que justifica parte do bom desempenho destes algoritmos.

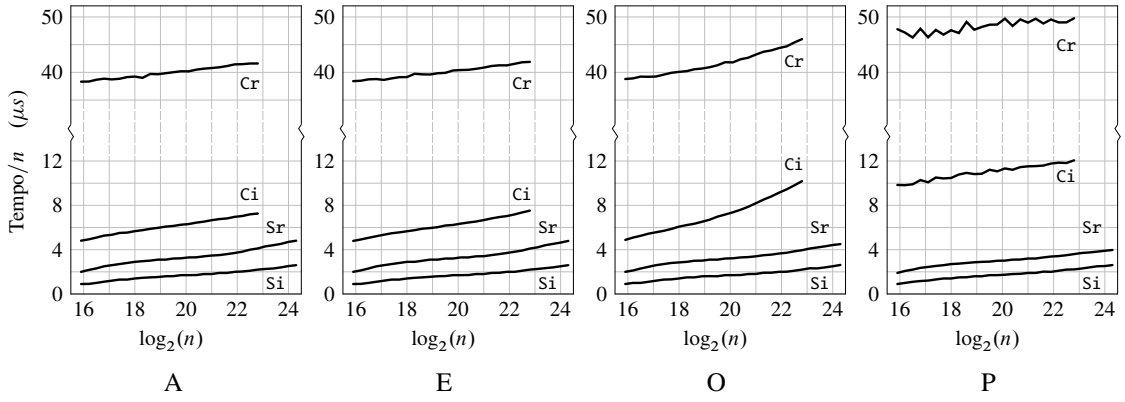


Figura 7.4: Tempos médios de execução por local para os algoritmos de edição (para n locais). Si e Sr indicam os algoritmos por varrimento esférico de inserção e remoção, enquanto que Ci e Cr indicam os algoritmos CGAL de inserção e remoção.

Concluindo, estes testes experimentais mostram que o algoritmo de varrimento é a escolha preferencial para a construção do diagrama de Voronoi esférico.

7.5 Comparação dos algoritmos de edição

Os algoritmos de inserção e remoção foram comparados com os algoritmos equivalentes da biblioteca CGAL, tanto no plano como na esfera. Mais especificamente, foi usada a triangulação de Delaunay, tanto em 2D como em 3D. Em 2D, porque é a estrutura dual do diagrama de Voronoi. Em 3D, porque é a única estrutura dinâmica de construção do invólucro-convexo desta biblioteca, permitindo operações de remoção e inserção de locais.

O teste experimental de um algoritmo de inserção com um conjunto de dados consistiu na construção do diagrama de Voronoi ou da triangulação de Delaunay correspondente, inserindo os locais sucessivamente por uma ordem pré-determinada (resultante de uma permutação arbitrária). Depois de construído o diagrama ou a triangulação do conjunto de dados, o teste experimental de um algoritmo de remoção consistiu na remoção de todos os locais, um a um, respeitando a ordem de uma outra permutação dos locais. Pode-se considerar que uma operação de inserção é composta por duas fases distintas: primeiro a procura da região a modificar, seguida da inserção efectiva (modificando a região). Nos testes experimentais, apenas foram medidos os tempos de execução da actualização. Nas remoções, os locais são acedidos directamente.

Vamos primeiro analisar os resultados obtidos no domínio esférico. A Figura 7.4 apresenta os tempos de execução médios, por local, dos dois algoritmos de inserção e dos dois algoritmos de remoção. Observa-se que todos os algoritmos ficam mais lentos quando o número de locais aumenta, justificado pela limitada capacidade da memória *cache*, o que reduz a respectiva taxa de acertos. De resto, existe uma diferença notável entre os dois tipos de algoritmos, uma vez que os tempos de execução dos algoritmos da CGAL são muito maiores que os dos algoritmos de varrimento. São apontadas duas razões para este facto: a CGAL

Tabela 7.3: Tempos de execução relativos para os algoritmos de edição.

algoritmo	inserção	remoção
varrimento esférico	$1\times$	$1\times$
varrimento planar	1.01 ± 0.04	0.77 ± 0.03
CGAL Delaunay (2D)	1.06 ± 0.02	0.73 ± 0.01

mantém uma estrutura de dados tridimensional e a remoção (cujo desempenho é fraco) segue uma estratégia de remendar o buraco criado pela remoção de um local colando, no seu lugar, a triangulação construída com os vizinhos do local removido. Observa-se também que o desempenho da CGAL é bastante influenciado pela distribuição espacial dos dados, enquanto que os algoritmos de varrimento exibem uma sensibilidade mínima à não-uniformidade espacial.

Em relação ao consumo de memória, não há nada de significativo a relatar. Todos os algoritmos fazem uso de uma estrutura de dados cujo tamanho é limitado pelo número de locais vizinhos do local a ser inserido ou removido.

Foram também medidos os tempos de execução no plano. Nomeadamente, os dos algoritmos da CGAL para editar a triangulação de Delaunay planar e os dos algoritmos de varrimento para editar o diagrama de Voronoi planar. Recorde-se que os algoritmos de varrimento no plano são idênticos aos equivalentes que operam na esfera, excepto na necessidade de contemplar regiões de Voronoi ilimitadas. Todos estes algoritmos foram executados com os mesmos conjuntos de dados A da secção 7.1. Também se observou que os tempos de execução relativos (normalizados pelo número de locais) não variam significativamente. Por este motivo, a Tabela 7.3 apresenta apenas o tempo médio e o desvio padrão dos tempos de execução, relativamente ao tempo médio de execução de uma edição com o algoritmo de varrimento na esfera (executado igualmente com dados em posições aleatórias).

No geral, observa-se que no plano os algoritmos de varrimento e os algoritmos da CGAL têm um desempenho semelhante. Além do mais, o custo de inserção no plano é idêntico ao da esfera. Por outro lado, o custo de remoção é maior na esfera, consequência do maior custo do cálculo de prioridades.

Parte II

Redução de um catálogo de estrelas

Capítulo 8

Redução de um catálogo

Na primeira parte deste trabalho foram analisados em detalhe algoritmos para construção e edição do diagrama de Voronoi de pontos na superfície da esfera. Nesta segunda parte, estes algoritmos são postos à prova como peças fundamentais de um algoritmo de redução de um catálogo de estrelas, com o objectivo de uniformizar a densidade de estrelas por todo catálogo.

O problema da redução de um catálogo de estrelas surge no domínio da orientação de uma nave espacial pelas estrelas. As técnicas de navegação espacial procuram conhecer, ao longo do tempo, a posição e a orientação de uma nave. A orientação de uma nave, designada por *atitude*, define a orientação de um referencial da nave em relação a um referencial externo à nave. Com a notável excepção do Sol, as estrelas estão tão distantes que apresentam sempre a mesma orientação, qualquer que seja a posição da nave espacial no Sistema Solar. Logo, uma forma de determinar a atitude é pelo mapeamento de uma imagem de um campo de estrelas, adquirida pela nave, num catálogo de estrelas.

O procedimento de determinação da atitude é conceptualmente simples [26, 27]. Adquire-se uma imagem de uma parte restrita da esfera celeste, localizam-se as estrelas na imagem, que formam um padrão, e identifica-se o padrão num catálogo. O mapeamento entre as posições na imagem e as posições na esfera, dadas pelo catálogo, permite determinar a atitude da nave. O processo está ilustrado na Figura 8.1. Este método exige que seja observado, em simultâneo, um mínimo de três estrelas.

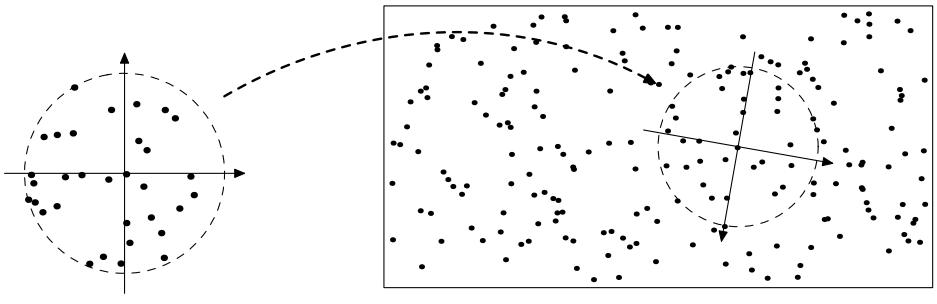


Figura 8.1: Determinação da atitude de uma nave por mapeamento de estrelas.

Um problema na implementação deste método é a dificuldade em armazenar um catálogo de estrelas completo a bordo de uma nave. Seja *campo de visão* um recorte da esfera celeste com forma circular (tal como observado por um telescópio). Por um lado, a exactidão na determinação da atitude é tanto maior quanto menor for o campo de visão observado. Porém, um menor campo de visão requer um maior número de estrelas, ou seja, um catálogo mais volumoso. Por outro lado, as naves espaciais são equipadas com recursos computacionais muito limitados, tanto em capacidade de cálculo como de armazenamento. Consequentemente, o espaço para guardar um catálogo é reduzido. Logo, a exequibilidade da utilização deste método depende da possibilidade de se diminuir o espaço ocupado por um catálogo de estrelas, sem comprometer o mapeamento.

A solução passa por reduzir um catálogo, removendo estrelas que não sejam absolutamente necessárias para a determinação da atitude. Como se observou atrás, apenas se exige um mapeamento mínimo de três estrelas. Logo, o mapeamento pode fazer-se com um catálogo equivalente que contenha, para qualquer região restrita da esfera celeste, centrada numa posição arbitrária, apenas as três estrelas mais intensas. Ou seja, a redução de um catálogo faz-se eliminando estrelas em zonas de densidade elevada que, por estarem próximas de estrelas mais intensas, não são absolutamente imprescindíveis para a determinação da atitude da nave.

A remoção selectiva de estrelas permite construir um catálogo de estrelas que preenche toda a esfera celeste ao mesmo tempo de ocupa pouca memória. De facto, a distribuição de estrelas na esfera celeste é pouco uniforme, com zonas muito densas ao longo do plano da galáxia e zonas pouco densas junto dos pólos. O que se propõe é uniformizar a distribuição de estrelas na esfera, de modo a que o número de estrelas num campo de visão de diâmetro fixo seja aproximadamente constante, numa operação que se denomina de *redução de catálogo*.

8.1 Algoritmo de redução

Nesta secção é proposto e analisado em detalhe um algoritmo de redução de catálogos de estrelas. O objectivo é desenvolver um algoritmo de redução que, dado um catálogo de estrelas inicial, produza um catálogo em que o número estrelas em recortes circulares, de diâmetro fixo, é aproximadamente constante, qualquer que seja a posição da esfera onde é feito o recorte. Quer-se também que o algoritmo de redução retenha, tanto quanto possível, as estrelas mais brilhantes do catálogo inicial. Um catálogo de estrelas, ao representar a esfera celeste, apresenta diferentes densidades de estrelas, consoante a região em análise. Um catálogo reduzido deverá apresentar uma distribuição de estrelas aproximadamente uniforme.

O catálogo inicial escolhido é o catálogo de estrelas UCAC4, já utilizado no capítulo 7 e que, recorde-se, contém aproximadamente 114×10^6 estrelas, estando completo até à magnitude 16.[1] Também aqui se restringiu o catálogo ao subconjunto das 21×10^6 estrelas mais

[1]Magnitude de uma estrela é uma medida da sua intensidade luminosa. É definida por uma escala logarítmica que atribui valores tanto maiores quanto menor for a intensidade de uma estrela.

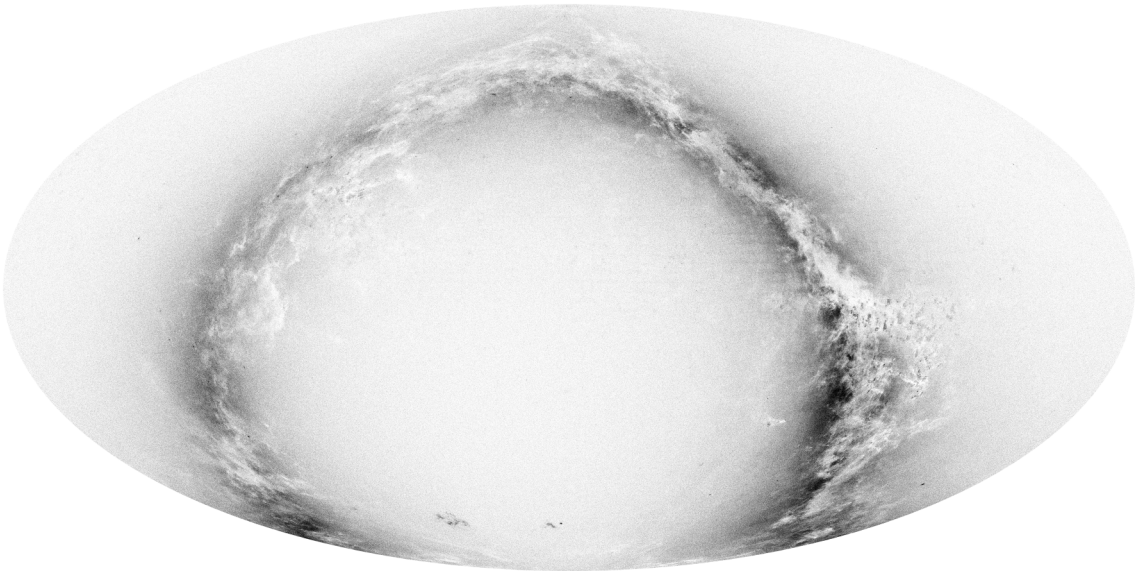


Figura 8.2: Catálogo UCAC4 em projecção de Hammer. O plano da galáxia está alinhado com a zona mais densa.

intensas, por forma a garantir que a redução se processa inteiramente nos 4GB de memória RAM disponíveis na plataforma de teste. A Figura 8.2 ilustra este catálogo em projecção de Hammer. Daqui em diante, UCAC4 designa somente o sub-conjunto referido. A densidade de estrelas média do UCAC4 varia com a declinação, apresentando densidades médias de 110 estrelas/grau^2 nos pólos da galáxia e 1100 estrelas/grau^2 no plano da galáxia. As regiões mais densas apresentam valores superiores a 9000 estrelas/grau^2.

A determinação da atitude de uma nave é feita por um método de mínimos quadrados [57], mapeando as estrelas identificadas numa imagem com as estrelas correspondentes no catálogo. O método exige um mínimo absoluto de três estrelas. No entanto, por se tratar de um problema de aproximação, há vantagem em mapear mais estrelas por forma a reduzir o erro da estimativa da atitude. Dada a variabilidade de densidades de estrelas no catálogo UCAC4, também não é razoável exigir que a redução produza um número de estrelas constante no campo de visão. Uma estrela a mais num campo de visão pode ser indispensável para garantir um mínimo de três estrelas num campo de visão próximo. Pelos motivos apresentados, o requisito do número de estrelas alvo no campo de visão é redefinido para o intervalo compreendido entre 3 a 9 estrelas. Os limites deste intervalo não derivam de nenhum critério adicional, sendo apenas valores considerados razoáveis, correspondendo o limite superior ao triplo do limite inferior.

O diâmetro do campo de visão depende dos constrangimentos da aplicação. Tipicamente, corresponde ao campo de visão da óptica que adquire as imagens das estrelas. Sem nenhum requisito imposto, optou-se por experimentar o algoritmo de redução para dois campos de visão: 15 arcmin e 30 arcmin. No final, serão obtidos dois catálogos reduzidos, um para cada valor de diâmetro. Desta forma, avaliar-se-á o desempenho dos catálogos reduzidos em dois cenários alvo distintos.

O algoritmo de redução baseia-se em diagramas de Voronoi de posições de estrelas. Por este motivo, e consoante o que for mais apropriado no contexto envolvente, a *posição de uma estrela* na esfera celeste é denominada indistintamente por estrela, local ou ponto. Pelo mesmo motivo, usam-se também as seguintes simplificações: um recorte do catálogo, que tem sempre a forma circular, é denominado simplesmente por *recorte*; e os *pontos de um recorte* são as estrelas do catálogo contidas no círculo de recorte.

O desenho do algoritmo de redução é consequência directa do objectivo pretendido: um catálogo reduzido deve conter um número aproximadamente constante de estrelas no campo de visão. Para isso, o algoritmo de redução deve remover do catálogo de trabalho estrelas que estejam circundadas, com relativa proximidade, por outras estrelas. Ou seja, se as distâncias de uma estrela às suas vizinhas são pequenas relativamente ao diâmetro do campo de visão então a remoção dessa estrela deve aproximar o catálogo do resultado pretendido. Esta observação pode ser enunciada de uma forma alternativa: se um recorte centrado numa estrela contém mais estrelas que um limite (a definir), então a remoção dessa estrelas não deve comprometer o mapeamento. Esta condição, embora atraente pela simplicidade, não é suficiente para uma redução eficaz. Na análise de uma estrela, as estrelas que lhe estão mais próximas podem estar todas agrupadas a um canto do campo de visão e, nesse caso, a estrela não deve ser removida, uma vez que é imprescindível para a região que a circunda. Para se definir um critério de remoção, precisa-se de uma medida da distribuição das estrelas em torno de um ponto, que condense a vizinhança de uma estrela em todas as direcções, o que é facilmente obtido com um diagrama de Voronoi.

A região de Voronoi de uma estrela permite definir uma medida da densidade local das estrelas num catálogo. Se as estrelas vizinhas de uma estrela se distribuem uniformemente em seu redor e estão relativamente próximas, então a área do polígono de Voronoi terá um valor relativamente pequeno. Se as estrelas vizinhas estão afastadas ou distribuídas de uma forma assimétrica, a área do polígono de Voronoi terá um valor relativamente grande. Portanto, a análise de uma estrela pelo valor da área do respectivo polígono conduz a um critério simples para o algoritmo de remoção: uma estrela deve ser removida se a área do polígono de Voronoi respectivo é *inferior* a um limiar pré-definido.

O Algoritmo 7 descreve o algoritmo de redução proposto. É uma aplicação directa do critério atrás enunciado. As estrelas são analisadas por ordem crescente de brilho. Para isso, é feita uma ordenação indexada das estrelas por ordem decrescente de magnitude (linha 1). O diagrama de todo o catálogo é construído na linha 3. Depois, cada estrela é analisada por ordem (linhas 4–12). É calculada a área da respectiva região de Voronoi (linha 6) que é comparada com um limiar fixo (linha 7). Se a área é inferior ao limiar, então a estrela é removida do diagrama (linha 8, aumentando as regiões vizinhas); senão, a estrela é adicionada ao catálogo reduzido (linha 10). O cálculo da área de um polígono esférico encontra-se descrito no apêndice A.

Pode-se imaginar o algoritmo de redução como um filtro que remove mais estrelas das regiões mais densas, deixando quase intactas as regiões menos densas. A "intensidade" do filtro depende directamente do limiar de área: quanto maior for o valor do limiar, menor o

Algoritmo 7 Redução de um catálogo de estrelas C, com um limiar σ.

```
1: vector: P ← ordenação(C)                    {Por ordem decrescente de magnitude.}
2: vector: R ← ∅                               {Estrelas do catálogo reduzido.}
3: diagrama: V ← construção_do_diagrama(C)
4: para i de 1 até |C| fazer
5:    j ← P[i]
6:    a ← V.área_da_região(j)
7:    se a < σ então
8:       V.remover_local(j)
9:    senão
10:      R.inserir(j)
11:   fim
12: fim
13: retorna R
```

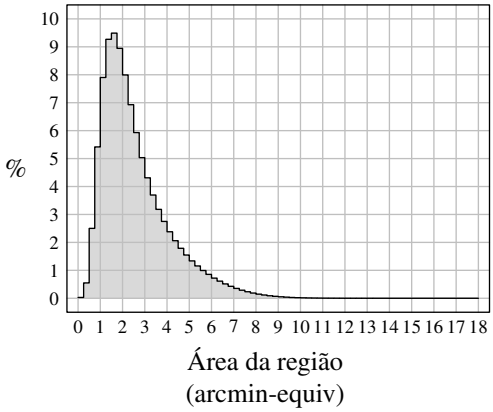


Figura 8.3: Histograma de áreas do catálogo UCAC4. A área média é de 3.0 arcmin-equiv, tendo o maior polígono uma área de 17.19 arcmin-equiv e o menor uma área de 0.06 arcmin-equiv.

número de estrelas removidas. O resultado deverá ser um catálogo de estrelas com uma distribuição aproximadamente uniforme. Note-se que o filtro só opera no sentido de remoção de estrelas. Se uma região do catálogo de trabalho apresenta uma densidade baixa (relativamente ao limiar de área) então é de esperar que essa baixa densidade se mantenha no catálogo reduzido.

O algoritmo de redução é controlado por um único parâmetro: o limiar de área. Variando este parâmetro obtêm-se catálogos reduzidos com diferentes desempenhos. No entanto, não é imediata uma relação entre o limiar de área e o desempenho que o catálogo reduzido terá (para um dado campo de visão). Por este motivo, a determinação do melhor catálogo reduzido é feita por uma procura exaustiva. É construído um conjunto de catálogos reduzidos, para uma gama de valores de limiar de área, escolhendo o que apresenta melhor desempenho para um dado campo de visão alvo. Na secção seguinte é descrito o método de medição do desempenho de um catálogo.

Esta secção termina com uma caracterização do catálogo UCAC4. A Figura 8.3 apresenta

o histograma das áreas dos polígonos de Voronoi, ou simplesmente o *histograma das áreas*, das estrelas do catálogo UCAC4. No histograma, as áreas são indicadas por uma medida alternativa, designada por *arcmin-equiv*, definida para simplificar a comparação entre áreas de polígonos e áreas de campos de visão. Uma área de d arcmin-equiv equivale à área de um círculo na superfície da esfera unitária com d arcmin de diâmetro. O histograma de áreas servirá para guiar a escolha de limiares de área na execução do algoritmo de redução.

8.2 Medição do desempenho de um catálogo

O desempenho de um catálogo será tanto melhor quanto maior for a parte da esfera em que um recorte produza um número de estrelas compatível com os requisitos (3 a 9 estrelas no campo de visão). Logo, o desempenho pode ser medido contando o número de estrelas em recortes feitos em todas as posições da esfera. Mais exactamente, o desempenho η_d de um catálogo para o diâmetro d é dado pelo integral duplo:

$$\eta_d = \frac{1}{4\pi} \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} \cos(\varphi)\, T_d(\varphi, \lambda)\, d\varphi\, d\lambda, \tag{8.1}$$

onde (φ, λ) é uma posição na esfera em coordenadas (latitude, longitude) e T_d é a função:

$$T_d(\varphi, \lambda) = \begin{cases} 1 & \text{se } \#R_d(\varphi, \lambda) \in [3, 9] \\ 0 & \text{caso contrário} \end{cases}, \tag{8.2}$$

em que $\#R_d(\varphi, \lambda)$ é o número de estrelas num recorte de diâmetro d centrado em (φ, λ). O valor de $\eta_d \in [0, 1]$ representa a soma da área (normalizada) das regiões da esfera que satisfazem o requisito indicado. É calculado por um integral duplo que não aceita uma integração analítica. No entanto, é fácil obter uma estimativa de η_d pelo método de Monte Carlo [51]. Usando a técnica de *acerta ou falha*, o valor de η_d é estimado por amostragem da função $T_d(\varphi, \lambda)$ em N pontos de todo o domínio. Seja μ_d dado por:

$$\mu_d = \frac{1}{N} \sum_{i=1}^{N} T_d(\varphi_i, \lambda_i), \tag{8.3}$$

onde (φ_i, λ_i), com $i = 1, \ldots, N$, é uma sequência de posições aleatórias na superfície da esfera, segundo uma distribuição uniforme. A estimativa de η_d é dada por:

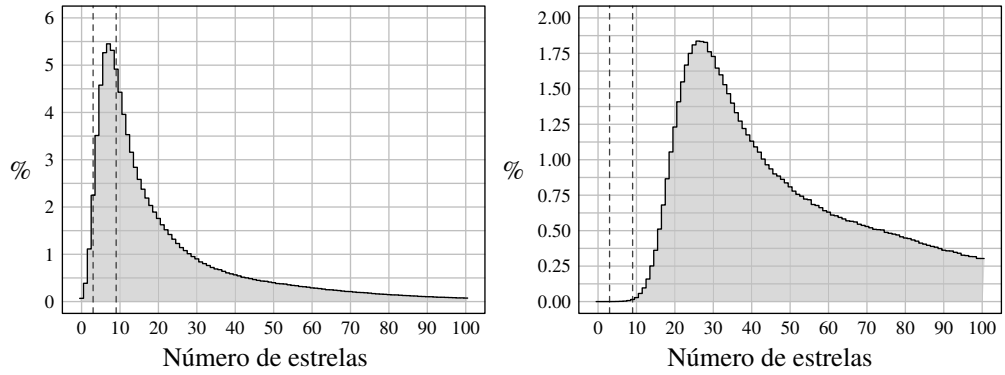
$$\hat{\eta}_d = \mu_d \pm \frac{1}{\sqrt{N}}, \tag{8.4}$$

sendo o segundo termo uma estimativa do erro de $\hat{\eta}_d$.

Na Tabela 8.1 são apresentadas as medidas de desempenho do catálogo UCAC4, para os dois valores do campo de visão considerados: 15 e 30 arcmin e, para comparação, 67.7 arcmin

campo de visão (arcmin)	número de estrelas min.	med.	max.	número médio estrelas zonas polares	plano do equador	desempenho em percent. μ ($\pm$0.1%)
15	0	25	1718	5	58	31.0 %
30	2	100	3114	21	234	$<$ 0.1 %
67.7	52	509	9096	110	1190	–

Tabela 8.1: Densidades de estrelas e desempenho do catálogo UCAC4 para vários campos de visão. Zonas polares e plano do equador referem-se aos pólos e ao equador da galáxia, respectivamente.



a) 15 arcmin b) 30 arcmin

Figura 8.4: Histogramas de desempenho do catálogo UCAC4, para os dois campos de visão.

$= 1$ grau^2. Foram obtidas pelo método de Monte Carlo, a partir de um total de 10^6 amostras. Observa-se que o desempenho piora com o aumento do diâmetro do campo de visão, tendo o catálogo UCAC4 melhor desempenho para 15 arcmin.

Um modo mais ilustrativo de medir o desempenho é com o histograma dos números de estrelas no campo de visão, designado por *histograma de desempenho*, tal como é apresentado na Figura 8.4. Este histograma apresenta o número de amostras obtidas para cada número de estrelas (por recorte). A medida de desempenho (em percentagem) corresponde ao somatório das colunas do histograma entre 3 e 9 estrelas (assinaladas na figura por linhas a tracejado). Ambos os histogramas foram truncados para recortes com contagens superiores a 100 estrelas, para realçar a visualização da parte mais interessante (a parte com menores contagens de estrelas).

O algoritmo de redução, ao remover estrelas do catálogo, irá modificar os histogramas de desempenho, deslocando-os para a esquerda. Se bem sucedido, o algoritmo de redução deve eliminar a parte do histograma superior a 9 estrelas, aumentando apenas a parte entre 3 e 9. Um "excesso" de redução revelar-se-á por um aumento do histograma entre 0 e 2 estrelas. Observa-se que, para o campo de visão de 15 arcmin, há uma parte significativa em que o campo de visão contém menos de 3 estrelas. Portanto, para este campo de visão, não será possível obter um desempenho de cerca de 100%.

O algoritmo de redução depende, essencialmente, de duas operações: a construção do

diagrama de Voronoi esférico e a remoção de um local de um diagrama, ambas desenvolvidas em capítulos anteriores. Para a medição do desempenho, é necessário operar um elevado número de recortes circulares (10^6), num catálogo com um elevado número de estrelas ($\approx 10^7$). O elevado volume de dados e operações dificulta qualquer tentativa de solução pelo método de força bruta. Logo, a opção foi suportar a operação de recorte com uma estrutura de dados, tendo a escolha recaído na hierarquia de Voronoi.

Capítulo 9

Recortes na esfera

Neste capítulo é apresentada a hierarquia de Voronoi, uma estrutura de dados que possibilita pesquisas de localização em conjuntos de pontos e, em particular, possibilita operações de recorte em conjuntos de pontos. Na secção 9.1 é apresentada a hierarquia de Voronoi em detalhe, enquanto que na secção 9.2 é mostrado como a hierarquia pode ser aumentada para suportar as operações de recorte.

9.1 Hierarquia de Voronoi

A hierarquia de Voronoi é uma estrutura de dados desenhada para responder eficientemente a pesquisas de localização [23, 36]. Dado um conjunto de locais $s_i \in S$, com $i = 1 \ldots n$, e um ponto de pesquisa p, a hierarquia de Voronoi permite identificar o local s_j mais próximo de p.

A ideia subjacente ao algoritmo de pesquisa é a de passeio num diagrama. Dado um local arbitrário num diagrama, designado por *local de partida*, é possível definir uma sequência de locais em que cada elemento da sequência está mais próximo do ponto de pesquisa que o local anterior. O processo de identificação da sequência de locais denomina-se por *passeio* num diagrama, que termina quando se encontra o local pretendido. Este último passo equivale a encontrar o local cuja região de Voronoi contém o ponto de pesquisa.

Considere-se um local de partida arbitrário s_1 e um ponto de pesquisa p, tal como ilustrado na Figura 9.1. O passeio de s_1 a p define uma sequência de locais $s_1, s_2, \ldots, s_k$. Esta sequência é obtida determinando um local s_{i+1} mais próximo de p que o local anterior s_i, calculando as distâncias entre p e todos os vizinhos de s_i (com excepção de s_{i-1}, para $i > 1$). O passeio termina quando todos os vizinhos do local s_k distam mais de p que ele próprio.

O Algoritmo 8 sintetiza o passeio num diagrama. Inicialmente, é calculada a distância do ponto de partida s_i à posição final (linha 1). Depois, em cada iteração, é determinado se algum dos vizinhos do local corrente está mais perto de p que o local corrente s_k (linhas 6–13), continuando o passeio no vizinho de s_k mais próximo de p. O processo é iterado (linhas 4–14) até que nenhum local vizinho diste menos de p que o local corrente s_k.

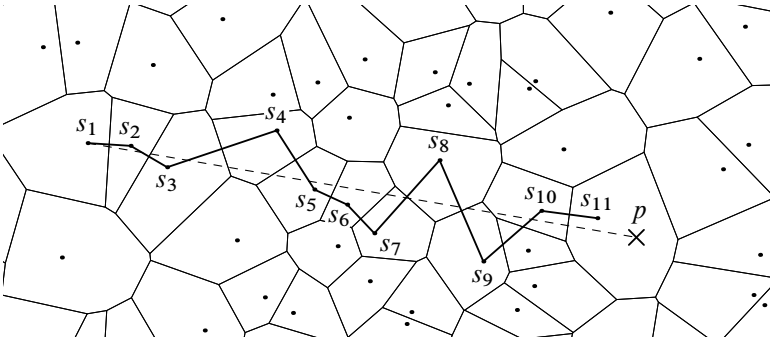


Figura 9.1: Sequência de locais de um passeio num diagrama de Voronoi.

Algoritmo 8 Passeio num diagrama de Voronoi V que, começando em s_i, identifica o local s_k mais próximo do ponto p.

```
1:  número: d ← dist(s_i, p)
2:  inteiro: k ← i
3:  booleano: c ← verdade
4:  enquanto c = verdade fazer
5:      c ← falso
6:      para j ∈ vizinhos(V, k) fazer
7:          e ← dist(s_j, p)
8:          se e < d então
9:              d ← e
10:             k ← j                    {Índice de um local mais próximo de p}
11:             c ← verdade              {Continuar o passeio}
12:         fim
13:     fim
14: fim
15: devolver k                           {Índice do local mais próximo de p}
```

O custo de uma pesquisa é (praticamente) proporcional ao comprimento do passeio, que é igual ao número de arestas intersectadas pelo segmento de recta (ou, na esfera, pelo segmento de círculo máximo) que une s_1 e p (ver Figura 9.1), e que facilmente se prova ter, em média, complexidade $O(\sqrt{n})$ [25]. Este resultado indica que o passeio é uma operação eficiente, mas não óptima, de localização espacial num diagrama. De facto, numa árvore-kd, por exemplo, a pesquisa espacial tem um custo $O(\log n)$ [20]. No entanto, uma árvore-kd não se adapta facilmente ao domínio esférico, o que justifica o recurso à hierarquia de Voronoi.

A baixa eficiência deste algoritmo reside na forma como se processa um passeio. O custo de um passeio é repartido equitativamente ao longo do seu percurso. O que significa que, para $n \gg 1$, a maior parte do custo de uma localização ocorre longe do local de pesquisa. A Figura 9.2 exemplifica um caso destes: o passeio tem início no canto superior esquerdo, executando um total de 57 passos, com passos de comprimento aproximadamente constante. Uma reformulação do conceito de passeio consegue superar esta desvantagem. Se a posição inicial está longe do objectivo, é preferível fazer um passeio a uma escala maior, aproximando-se do

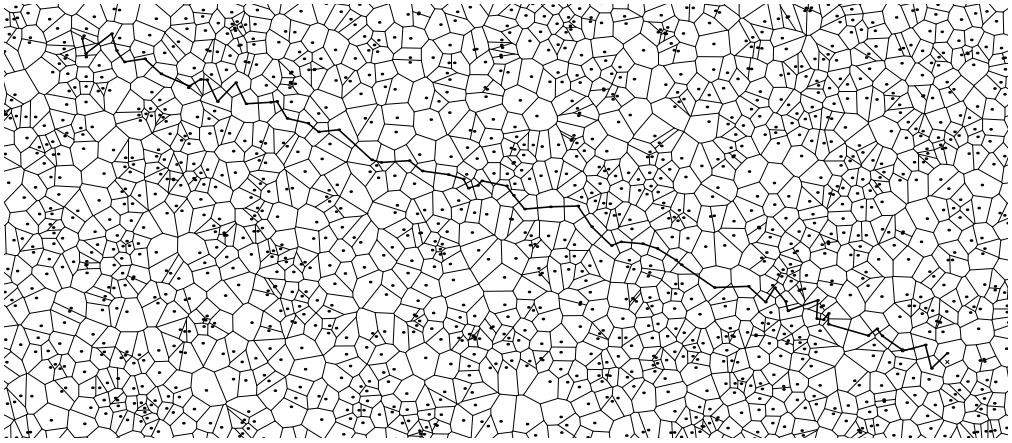


Figura 9.2: Passeio num diagrama de Voronoi (57 passos).

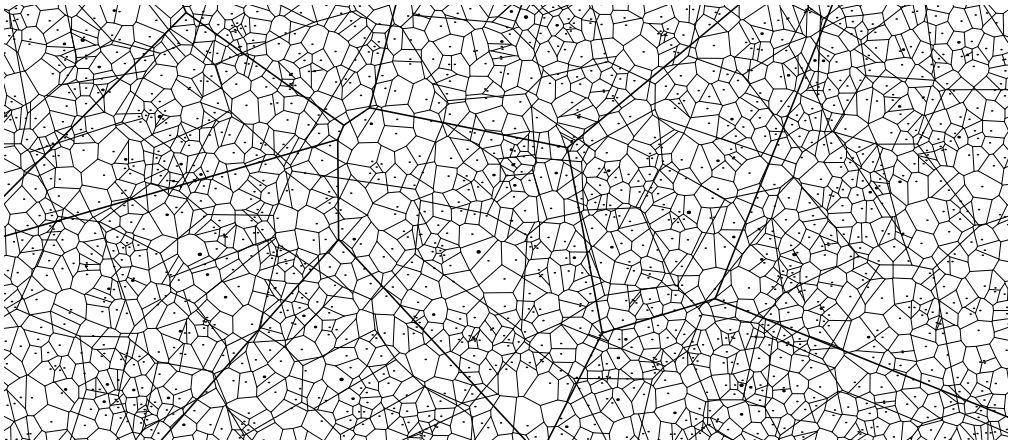


Figura 9.3: Hierarquia de Voronoi.

local pretendido com passos mais largos, encurtando os passos à medida que o percurso se aproxima do objectivo. É aqui que entra a estrutura em hierarquia.

A hierarquia de Voronoi implementa a ideia de um passeio com passos de tamanho variável [23, 36]. Uma *hierarquia* é composta por várias camadas, conceptualmente sobrepostas, em que cada camada é formada por uma selecção dos pontos da camada inferior. A camada 0, também denominada por *base*, é a camada mais baixa e contém todos os n locais. A camada 1 contém um subconjunto de $n_1 = \lfloor n/\alpha \rfloor$ locais da camada 0, seleccionados aleatoriamente, onde α (≥ 2) é um parâmetro denominado por *factor de ramificação*. As restantes camadas são obtidas pelo mesmo processo, preenchendo a camada k com uma selecção aleatória de $\lfloor n_{k-1}/\alpha \rfloor$ locais da camada inferior, até se atingir a camada de *topo*, m, com um número de locais $n_m \leq \alpha$. A construção de cada camada é completada construindo o diagrama de Voronoi V_k respectivo. Os locais seleccionados na construção da hierarquia, que preenchem todas as camadas com excepção da base, denominam-se igualmente por *nós* de uma camada.

As várias camadas estão associadas entre si, ligando os nós de cada camada aos nós respectivos da camada inferior. Esta ligação é feita de cima para baixo, no sentido inverso ao da selecção de nós na construção da hierarquia. A Figura 9.3 ilustra uma hierarquia de

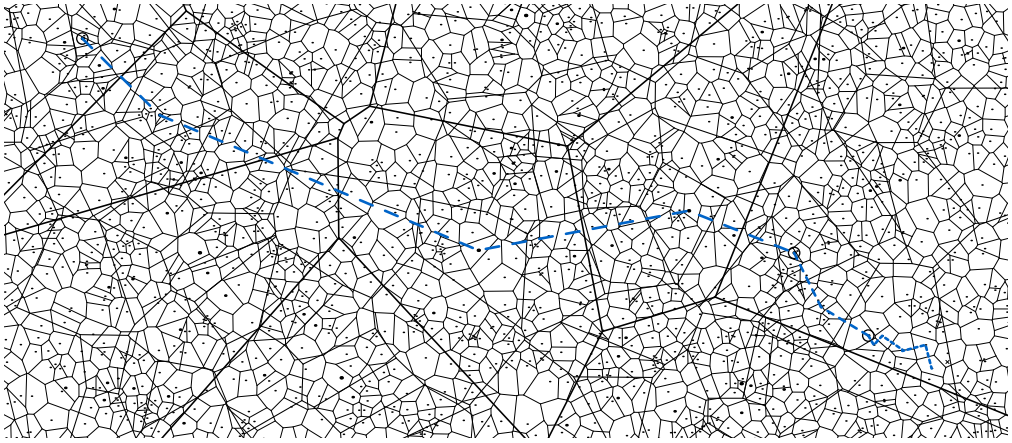


Figura 9.4: Pesquisa na hierarquia de Voronoi (12 passos).

Voronoi com três camadas (desenhados em sobreposição), para o mesmo conjunto de pontos da Figura 9.2. No desenho, o diâmetro de cada ponto varia com as camadas a que pertence: os de maior diâmetro são os nós da camada de topo e os de menor diâmetro pertencem apenas à camada de base.

A estrutura em hierarquia permite solucionar, de uma forma elegante, o problema do passeio demasiado longo quando executado num só diagrama de Voronoi. Agora, uma procura é realizada percorrendo a hierarquia camada a camada, do topo para a base.

Começando num nó arbitrário, percorre-se a camada de topo até se encontrar o nó (do topo) mais próximo do nó de pesquisa. Encontrado esse nó, *desce-se* para a camada inferior, seguindo a ligação entre camadas. A procura recomeça nesta camada, executando um segundo passeio até ao nó mais próximo do ponto de pesquisa (nesta camada). O processo é iterado até se chegar à base, onde é encontrado o local pretendido. Na Figura 9.4 é repetida a operação de pesquisa ilustrada na Figura 9.2, agora numa hierarquia de três camadas. Os círculos indicam descidas de camada ao longo do passeio. Observe-se como o número de passos foi reduzido de 57 para 12, agora com passos de tamanho variável.

Intuitivamente, a procura na hierarquia ganha eficiência dando passos maiores nas camadas superiores, enquanto o passeio se faz longe do ponto de pesquisa, e reduzindo o passo à medida que se desce na hierarquia, quando o passeio se aproxima do objectivo. Deste modo, o custo fica repartido por pequenos passeios nas várias camadas.

A hierarquia de Voronoi é conceptualmente simples e fácil de construir. A selecção de nós de cada camada é obtida escolhendo n_k locais de uma permutação aleatória dos locais da camada inferior. As propriedades da hierarquia decorrem deste processo de selecção. Nomeadamente:

- o número de camadas da hierarquia é igual a $m+1$, sendo o índice da camada de topo $m = 1 + \lfloor \log_\alpha \frac{n}{\alpha+1} \rfloor$, com $n > \alpha$ (e $m \geq 0$);

- se um nó está presente na camada k, então também está presente em todas as camadas inferiores a k;

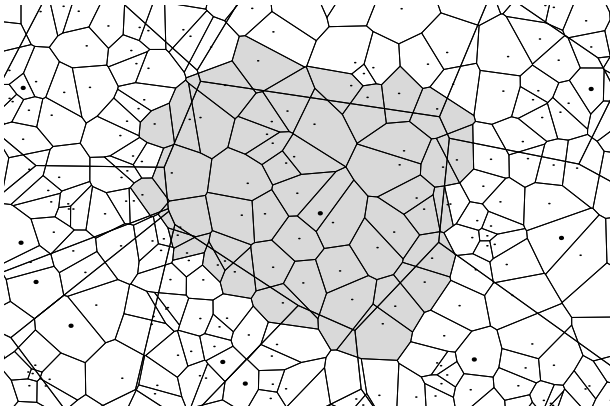


Figura 9.5: Zona de influência de um nó numa hierarquia de Voronoi. A parte a sombreado é a zona de influência do nó da região central.

- a distribuição espacial de nós numa dada camada replica a distribuição espacial dos locais da camada inferior, embora numa escala mais larga;

- o número de pontos de uma camada que estão mais próximos de cada nó da camada acima é, em média, de α.

A última propriedade condiciona o comprimento médio de um passeio numa camada (diferente da do topo). Seja s um nó da camada k e $V(s)$ a respectiva região de Voronoi. A zona de influência do nó s é dada pela união de todas as regiões de Voronoi na camada $k-1$ intersectadas ou contidas em $V(s)$ (ver Figura 9.5). Como o número médio de locais da camada $k-1$ contidos em $V(s)$ é de α, então o comprimento de um passeio numa camada (diferente da do topo) é, em média, igual a $1+\frac{\sqrt{\alpha}}{2}$ passos (assumindo que o passeio tem início no centro da região); onde $\sqrt{\alpha}$ corresponde ao número médio de regiões intersectadas por uma recta arbitrária e o termo 1 corresponde ao passo adicional requerido para determinar o fim do passeio.

Como se verá na secção 9.2, para a operação de recorte é vantajoso *não* construir uma hierarquia completa, preenchendo todas as camadas, sendo preferível descartar a camada de base. Posteriormente, cada ponto da base é associado ao nó da camada 1 que lhe está mais próximo. Desta forma, um recorte é feito na camada 1, identificando o nó mais próximo do local de pesquisa e seleccionando os locais da base associados a esse nó. Vamos primeiro ver, em detalhe, como é utilizada e construída a hierarquia de Voronoi.

O Algoritmo 9 identifica o local da camada 1, de uma hierarquia de Voronoi, mais próximo de um ponto de pesquisa p. Um passeio no diagrama V_m da camada m, com início num local arbitrário (no pseudo-código é indicado o local 1), identifica o local k da camada de topo mais próximo de p (linha 1). Se se acabou de percorrer a camada $i=1$, a pesquisa termina (devolvendo k). Senão, desce-se para a camada inferior ($i-1$), seguindo a correspondência de nós entre camadas guardadas no vector D_i (linhas 4 e 5). O passeio repete-se na nova camada, com o objectivo de encontrar o local k dessa camada mais próximo de p (linha 6). O procedimento é iterado até se chegar à camada 1 (linha 8).

Algoritmo 9 Localização do local do diagrama V_1 mais próximo de p, numa hierarquia de Voronoi $H = (V_{\{1,\ldots,m\}}, D_{\{1,\ldots,m\}})$.

1: $k \leftarrow$ passear$(V_m, 1, p)$
2: $i \leftarrow m$ $\{m \geq 1\}$
3: **enquanto** $i > 1$ **fazer**
4: $\quad j \leftarrow D_i[k]$ {Descer para a camada inferior}
5: $\quad i \leftarrow i - 1$
6: $\quad k \leftarrow$ passear(V_i, j, p)
7: **fim**
8: **devolver** k {Índice do local mais próximo de p em V_1}

Algoritmo 10 Construção de uma hierarquia H de Voronoi de um conjunto S_0.

1: $n \leftarrow |S_0|$
2: $i \leftarrow 0$
3: **enquanto** $n > \alpha$ **fazer** {Construir as camadas}
4: $\quad T \leftarrow$ permutação$(1, \ldots, n)$
5: $\quad n \leftarrow \lfloor n/\alpha \rfloor$
6: $\quad i \leftarrow i + 1$
7: $\quad$ **para** $j = 1$ **até** n **fazer**
8: $\quad\quad D_i[j] \leftarrow T[j]$ {Ligar nó j (camada i) ...}
9: $\quad\quad S_i[j] \leftarrow S_{i-1}[T[j]]$ {... ao nó $T[j]$ (camada $i-1$)}
10: $\quad$ **fim**
11: $\quad V_i \leftarrow$ construir_diagrama(S_i)
12: **fim**
13: $H \leftarrow \left(V_{\{1,\ldots,m\}}, D_{\{1,\ldots,m\}}\right)$
14: $n \leftarrow |S_0|$
15: **para** $j = 1$ **até** n **fazer** {Atribuir locais a regiões}
16: $\quad k \leftarrow$ localizar(H, s_j)
17: $\quad$ adicionar(U_k, s_j) {Associar s_j com a região P_k}
18: **fim**
19: **devolver** $H, U_{\{1,\ldots,\lfloor n/\alpha \rfloor\}}$

O Algoritmo 10 constrói uma hierarquia de Voronoi de um conjunto de pontos S_0. A construção faz-se camada a camada. A camada 1 consiste num subconjunto de $\lfloor n/\alpha \rfloor$ locais extraídos de uma permutação aleatória de todos os locais de S_0 (linha 4) sendo guardado, para cada nó de S_1, a posição e índice do local de respectivos em S_0 (linhas 7–10). A construção da camada 1 é completada com a construção do diagrama de Voronoi de S_1 (linha 11). As restantes camadas são construídas iterando este processo (linha 3–12), enquanto o número de locais da última camada construída for superior a α. Para auxiliar a operação de recorte, a hierarquia de Voronoi é aumentada associando cada local de S_0 à região de V_1 que o contém (linhas 14–17). A hierarquia consiste nos diagramas de Voronoi de cada camada $V_{\{1,\ldots,m\}}$ e nas ligações entre camadas $D_{\{1,\ldots,m\}}$. As associações entre locais de S_0 e a camada 1 são guardadas em $U_{\{1,\ldots,\lfloor n/\alpha \rfloor\}}$.

Vamos agora analisar o custo de construção e pesquisa numa hierarquia de Voronoi.

O custo de uma pesquisa reparte-se pelas pesquisas parciais nas m camadas. Em cada camada, o custo de uma pesquisa é proporcional ao número de locais visitados ao longo do passeio que, como atrás indicado, é de $1 + \frac{\sqrt{\alpha}}{2}$, em média.

Portanto, o custo esperado de uma pesquisa C_{pesq} é de:

$$\begin{aligned} C_{\text{pesq}} &= \left(1 + \frac{\sqrt{\alpha}}{2}\right) \left\lfloor 1 + \log_\alpha \frac{n}{\alpha + 1} \right\rfloor \\ &= O(\sqrt{\alpha}\, \log_\alpha n)\ . \end{aligned} \tag{9.1}$$

Com a estrutura em hierarquia, uma pesquisa tem um custo temporal esperado logarítmico, o que contrasta com o custo $O(\sqrt{n})$ de um passeio num diagrama.

Resta elaborar um critério para a determinação do factor de ramificação. Para optimizar a operação de pesquisa, deve escolher-se um factor de ramificação α que minimize C_{pesq}, encontrando um equilíbrio entre número de camadas e comprimento dos passeios em cada camada. Cada transição entre camadas penaliza a operação de pesquisa com um passo adicional, dando preferência a um número mínimo de camadas. Por outro lado, menos camadas significa passeios mais longos em cada camada, atenuando a vantagem da hierarquia. Um valor óptimo de α pode ser determinado por uma procura exaustiva, calculando C_{pesq} para uma gama de factores de ramificação. Dado que se procura por um mínimo, basta calcular C_{pesq} para valores sucessivos de α (começando em $\alpha = 2$) até que se encontre o primeiro mínimo local. Na prática, não é necessário sondar um intervalo excessivamente grande. Experimentalmente, observou-se que o valor óptimo de α é inferior a 100 para qualquer valor prático de n (i.e., $n < 10^{10}$).

O custo da construção da hierarquia de Voronoi C divide-se em três partes: o custo da selecção de nós C_{sel}, o custo da construção dos diagramas de Voronoi C_{vor}, um por cada camada, e o custo da localização dos pontos da base C_{loc} (para associação de cada ponto a uma região da camada 1).

A selecção dos n_k nós de cada camada implica a construção de uma permutação aleatória dos n_{k-1} locais da camada inferior, que é uma operação linear no tempo. Logo, a selecção dos nós das m camadas tem um custo C_{sel} dado por:

$$\begin{aligned} C_{\text{sel}} &= \sum_{k=0}^{m-1} \frac{n}{\alpha^k} \\ &\leq \sum_{k=0}^{\infty} \frac{n}{\alpha^k} \\ &= \frac{\alpha}{\alpha - 1}\, n \\ &\leq 2\, n\,. \end{aligned} \tag{9.2}$$

A construção dos diagramas de Voronoi das m camadas tem um custo C_{vor} dado por:

$$\begin{aligned} C_{\text{vor}} &= \sum_{k=1}^{m} \frac{n}{\alpha^k} \log\left(\frac{n}{\alpha^k}\right) \\ &\leq \sum_{k=1}^{\infty} \frac{n}{\alpha^k} \log(n) \\ &= \frac{1}{\alpha - 1}\, n \log(n) \\ &\leq n \log(n). \end{aligned} \tag{9.3}$$

O custo esperado da localização C_{loc} dos n pontos da base é dado por:

$$C_{\text{loc}} \approx n\, \sqrt{\alpha} \log_{\alpha}(n)\ . \tag{9.4}$$

Somando as três parcelas, obtém-se o custo médio da construção da hierarquia C_{hier}, que é da ordem de:

$$\begin{aligned} C_{\text{hier}} &\approx 2\,n + n \log(n) + \frac{\sqrt{\alpha}}{\log \alpha}\, n \log(n) \\ &= 2\,n + \left(1 + \frac{\sqrt{\alpha}}{\log \alpha}\right) n \log(n). \end{aligned} \tag{9.5}$$

Conclui-se que a construção tem um custo temporal de $O(\sqrt{\alpha}\, n \log n)$.

O custo espacial da hierarquia C_{esp} é dado pela soma dos custos de cada diagrama de Voronoi, das ligações entre camadas e da associação de cada ponto da base à primeira camada, sendo dado por:

$$\begin{aligned} C_{\text{esp}} &= \sum_{k=1}^{m} \frac{n}{\alpha^k} + \sum_{k=2}^{m} \frac{n}{\alpha^k} + n \\ &\leq \sum_{k=1}^{\infty} \frac{n}{\alpha^k} + \sum_{k=2}^{\infty} \frac{n}{\alpha^k} + n \\ &= \frac{\alpha^2 + 1}{\alpha(\alpha - 1)}\, n \\ &\leq \frac{5}{2}\, n. \end{aligned} \tag{9.6}$$

Concluindo, a hierarquia de Voronoi tem um custo espacial linear em n.

Como estrutura de pesquisa, a hierarquia de Voronoi partilha algumas semelhanças com as árvores-B$^+$ [19]. As pesquisas são feitas comparando uma chave com alguns elementos de cada nível, caminhando do topo para a base. Pela forma como ambas as estruturas de dados estão desenhadas, o número de comparações em cada nível é limitado: corresponde ao factor

de ramificação na hierarquia de Voronoi, que equivale à ordem da árvore-B^+.

9.2 Recorte no catálogo

Continuando o desenvolvimento do algoritmo de redução de catálogos de estrelas, vamos agora descrever a operação de recorte num catálogo e, em particular, justificar a exclusão da camada da base na construção da hierarquia de Voronoi.

Um recorte circular pode ser implementado eficientemente por um percurso num diagrama de Voronoi. Começando na região que contém o centro do recorte, basta efectuar um percurso em largura, visitando as regiões que estão contidas no recorte, mesmo que parcialmente. O recorte é obtido seleccionando o local de cada região visitada que está contido no círculo.

Mais detalhadamente, um recorte circular é implementado da seguinte forma. Primeiro é localizado o local mais próximo do centro do recorte. Como foi demonstrado atrás, esta operação é implementada eficientemente pela hierarquia de Voronoi. De seguida, é determinado se o local identificado está contido no recorte. Se não está, então o recorte não pode conter nenhum outro local (pela propriedade dos círculos vazios máximos), o que significa que o recorte é vazio. Por outro lado, se o local está contido no recorte, então a operação continua visitando as regiões vizinhas e repetindo o processo: seleccionar o local respectivo e, se aplicável, avançar para outras regiões ainda não visitadas.

Em certas condições, a estratégia descrita pode ser muito ineficiente. O percurso em largura num diagrama implica a enumeração de todas as regiões vizinhas de cada região, mesmo que seja apenas para verificar que (algumas) já foram visitadas. Por outro lado, em cada nova região, há que determinar se o local correspondente está contido no recorte, o que se resume a um simples cálculo de uma distância. Se um recorte contém um número elevado de pontos, então o custo é fortemente penalizado pelo custo do percurso em largura, uma vez que o custo da enumeração de regiões vizinhas domina sobre o custo de um cálculo de distância. Recorde-se que recortes de 30 arcmin no catálogo UCAC4 podem conter mais de 3000 estrelas, o que torna a solução descrita pouco atraente.

Uma forma de atenuar esta desvantagem é abarcando mais locais de uma só vez, agrupando vários locais em cada região, ao mesmo tempo que se reduz o número de regiões visitadas. É por este motivo que se optou por limitar a construção da hierarquia de Voronoi à camada 1, aumentando-a com as ligações entre cada região da camada 1 e os locais da base. Desta forma, o recorte é feito percorrendo a camada 1 da hierarquia, seleccionando, na visita a cada região, os locais da base que lhe estão associados. A estratégia de recorte atrás delineada também se aplica a este novo cenário. No entanto, agora pode acontecer que pontos de uma região, que não o local gerador, estejam contidos no recorte, o que invalida o anterior critério de propagação do percurso no diagrama. A solução é relativamente simples: começando na região que contém o centro, o percurso em largura no diagrama deve visitar todas as regiões intersectadas pelo círculo do recorte. Logo, basta agora avançar para uma região vizinha se

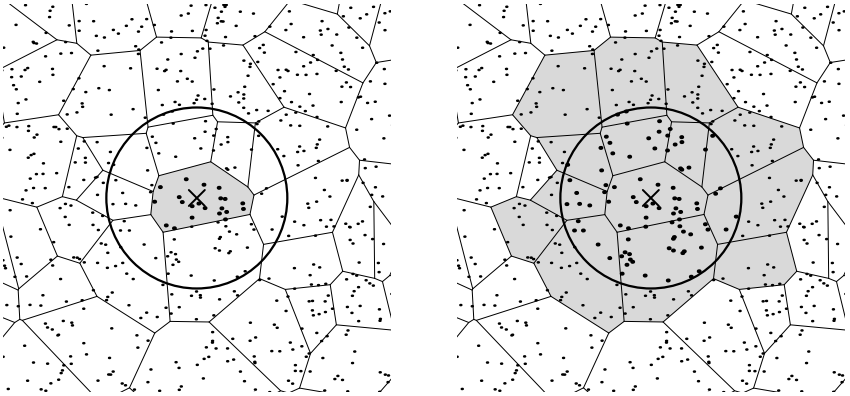


Figura 9.6: Polígonos visitados na operação de recorte (a sombreado).

Algoritmo 11 Recorte numa hierarquia H, U por um círculo (p, r).

```
 1: vector: R                                          {Guarda os índices dos locais do recorte}
 2: inteiro: n ← 0                                     {Número de locais em R}
 3: vector: E ← {0, ..., 0}                            {Identifica as regiões encontradas}
 4: fila: F ← ∅                                        {Fila FIFO com regiões a visitar}
 5: i ← F.localiza(H, p)
 6: F.insere(i)                                        {Começar pela região que contém p}
 7: E[i] ← 1                                           {Marcar região i como encontrada}
 8: enquanto F ≠ ∅ fazer
 9:    i ← F.remove()
10:    para j ∈ vizinhos(V_1, i) fazer
11:       se E[j] ≠ 1 ∧ contém_aresta(p, r, e_j) então    {e_j é uma aresta de V_1(s_i)}
12:          F.insere(j)
13:          E[j] ← 1
14:       fim
15:    fim
16:    para k ∈ U_i fazer
17:       se dist(p, s_k) ≤ r então
18:          R[n] ← k
19:          n ← n + 1
20:       fim
21:    fim
22: fim
23: devolver R
```

está contida no recorte (mesmo que parcialmente).

O Algoritmo 11 detalha o procedimento de recorte numa hierarquia de Voronoi (e ilustrado na Figura 9.6). O recorte tem início na região de Voronoi da camada 1 de H que contém p (linha 5). O recorte prossegue visitando as regiões vizinhas da região inicial. Em cada iteração são identificadas as regiões vizinhas que podem conter parte do recorte (linhas 10–15) e é identificado o (sub-)conjunto dos pontos associados à região corrente contidos no recorte (linhas 16–21). O vector E identifica as regiões já encontradas (linhas 7 e 13), garantindo

que não são repetidas no percurso (linha 11). Para determinar se uma região está contida no recorte, é suficiente determinar se contém uma das arestas (pela inclusão de um dos vértices incidentes ou, no pior dos casos, pela intersecção parcial com a aresta). O recorte é devolvido sob a forma de um vector de índices de locais (linha 23).

Capítulo 10

Resultados experimentais

Este capítulo apresenta os resultados da redução do catálogo de estrelas UCAC4. É analisado o desempenho do algoritmo de redução, desenvolvido no capítulo 8, e são estudados os catálogos reduzidos específicos para cada campo de visão.

O algoritmo de redução depende de um único parâmetro: o limiar de área. No capítulo 8, foi definida a unidade arcmin-equiv. Uma área de d arcmin-equiv é a área de um círculo da superfície da esfera unitária com d arcmin de diâmetro. Como se verá mais adiante, é agora conveniente parametrizar o limiar de área em função do inverso do diâmetro de um círculo de área equivalente. Por este motivo, o limiar de área é definido em função de um parâmetro adimensional ϕ, dado pela área do círculo esférico com $60/\phi$ minutos de arco de diâmetro. Ou seja, o limiar σ é dado por:

$$\sigma(\phi) = 2\pi \left(1 - \cos\left(\frac{60}{2\phi}\right)\right). \tag{10.1}$$

Na prática, foram usados valores de ϕ de 1.0 a 20, correspondendo a limiares de área de 60 arcmin-equiv a 3 arcmin-equiv.

Vamos agora descrever como se obteve a redução do catálogo UCAC4 e a posterior medição de desempenho. Seja d o diâmetro do campo de visão e ϕ o parâmetro de redução. O melhor catálogo para cada campo de visão foi obtido por uma procura exaustiva, medindo o desempenho de uma gama alargada de catálogos reduzidos.

O Algoritmo 12 descreve este processo. O catálogo UCAC4 foi reduzido 191 vezes, para valores de ϕ a variar de 1.0 até 20.0 (linhas 3–14). Para cada catálogo resultante da redução com o limiar $\sigma(\phi)$ (linha 4, c.f. Algoritmo 7) foi construída uma hierarquia de Voronoi correspondente (linha 5, c.f. Algoritmo 10). Para a construção de cada hierarquia de Voronoi, foi escolhido um factor de ramificação α que minimiza o custo de cada pesquisa, tal como descrito na secção 9.1 (pág. 121). A hierarquia suporta as operações de recorte (num total de 10^6 por catálogo, c.f. Algoritmo 11) necessárias para a medição do desempenho de cada catálogo reduzido (linha 7), para os dois campos de visão considerados: 15 arcmin e 30 arcmin (linhas 6–13). No final são devolvidos dois catálogos, T_{15} e T_{30} (linha 15),

Algoritmo 12 Construção e medição de desempenho de catálogos reduzidos.

catálogo: T_{15}, T_{30} {Catálogos reduzidos para 15 arcmin e 30 arcmin}
número: $\mu_{15}, \mu_{30} \leftarrow 0, 0$ {Melhor desempenho para 15 arcmin e 30 arcmin}
para $\phi = 1.0, 1.1, 1.2, \ldots, 20.0$ **fazer**
  $T \leftarrow$ redução_de_catálogo(UCAC4, $\sigma(\phi)$)
  $H \leftarrow$ construir_hierarquia(T)
  para $d = 15, 30$ **fazer**
    $\mu \leftarrow$ medir_desempenho(T, H, d)
    se $\mu > \mu_d$ **então**
      $\mu_d \leftarrow \mu$
      $\phi_d \leftarrow \phi$
      $T_d \leftarrow T$
    fim
  fim
fim
devolver $T_{15}, T_{30}, \mu_{15}, \mu_{30}$

Tabela 10.1: Detalhes da hierarquia de Voronoi do catálogo UCAC4 ($\alpha = 29$).

camada	número de pontos
4	29
3	861
2	24970
1	724137
base	21000000

que registam o melhor desempenho (μ_{15} e μ_{30}) para os campos de visão de 15 arcmin e 30 arcmin, respectivamente.

O algoritmo de redução e a hierarquia de Voronoi foram implementados na linguagem C e compilados com o compilador GNU GCC C compiler (versão 4.4.5), com optimizações. Os programas foram executados num processador Intel Xeon E5160, correndo a 3.00 GHz.

O objectivo do procedimento descrito é a validação do algoritmo de redução, averiguando a variação do desempenho com o grau de redução (para cada diâmetro de campo de visão). O resultado principal é a obtenção do catálogo reduzido que melhor desempenho apresenta para cada campo de visão considerado. Um segundo resultado, igualmente relevante, é a validação da hierarquia de Voronoi como estrutura de dados de indexação espacial de um catálogo de estrelas e o seu uso para a operação de recorte.

Comecemos primeiro por caracterizar a construção da hierarquia de Voronoi, analisando a aplicação desse processo ao catálogo UCAC4. A hierarquia de Voronoi do catálogo UCAC4 (com 21×10^6 estrelas) foi construída com um factor de ramificação de $\alpha = 29$ (valor que minimiza o custo da operação de pesquisa). Obteve-se uma hierarquia de 4 camadas, cujos detalhes estão sintetizados na Tabela 10.1.

A Figura 10.1 apresenta o histograma do número de pontos (estrelas) associados a cada

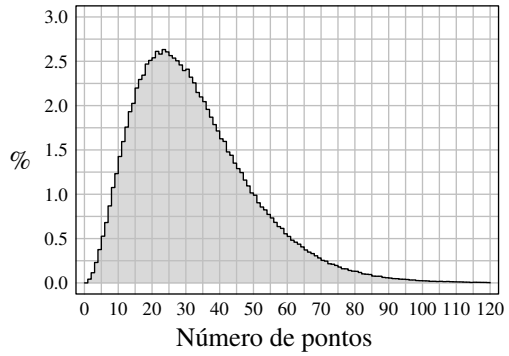


Figura 10.1: Histograma do número de pontos por região da camada 1 da hierarquia do catálogo UCAC4.

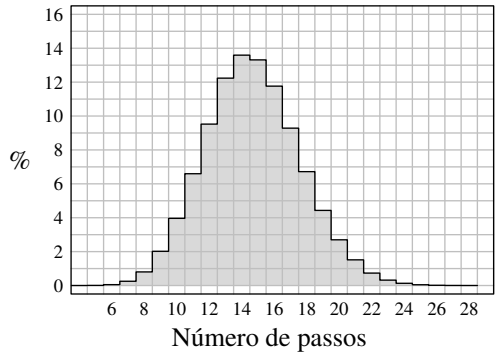


Figura 10.2: Histograma do número de passos por pesquisa para um total de 10^6 pesquisas em posições aleatórias.

região da camada 1. Idealmente, o número de pontos por região seria fixo e igual ao factor de ramificação. Mas, como mostra o histograma, o número de pontos por região apresenta uma grande variação, desde 1 ponto até mais de 120 pontos por região.

O custo de uma pesquisa na hierarquia é proporcional à soma dos comprimentos dos passeios nos diagramas de Voronoi das diferentes camadas. A Figura 10.2 apresenta este custo sob a forma do histograma do número de passos por pesquisa, de um total de 10^6 operações de pesquisa na hierarquia do catálogo UCAC4. Observa-se que as pesquisas requerem, em média, 14.96 passos, nunca ultrapassando 28 passos. Estes valores estão concordantes com o custo estimado de uma pesquisa que, para $\alpha = 29$ e $n = 21 \times 10^6$, resulta num custo esperado de $(1 + \frac{\sqrt{29}}{2}) \times 4 = 14.77$ (c.f. Eq. 9.1). As pesquisas com o menor número de passos, 4, correspondem ao valor mínimo de um só passo por camada.

Estes resultados mostram que a estratégia de selecção aleatória de nós de cada camada não produz uma hierarquia óptima; o número de pontos associados a cada nó apresenta uma variação apreciável, o que se reflecte numa variação no custo de uma operação de pesquisa. No entanto, este resultado menos óptimo não penaliza a medição do desempenho dos catálogos, uma vez que o custo de cada recorte é amortizado pelo número elevado de recortes (distribuídos uniformemente por toda a esfera), necessários para uma medição de desempenho.

A caracterização da operação de recorte está sintetizada nas Figuras 10.3 e 10.4 que apresentam, respectivamente, o tempo médio de uma operação de recorte em função do diâmetro e o tempo médio de uma operação de recorte em função do número de pontos obtidos. Na Figura 10.3 também está indicado o tempo médio de pesquisa na hierarquia de Voronoi, 5.4 µs. Ambos os gráficos apresentam a média dos valores medidos em 10^6 recortes. Para os diâmetros de recorte considerados, observa-se que o custo da operação de recorte depende, essencialmente, do percurso no diagrama de nível 1 da hierarquia, largamente superior ao custo da pesquisa na hierarquia. Este custo varia com o número de pontos visitados sendo proporcional à área recortada.

Uma redução do catálogo UCAC4 demora entre 180 s (para $\phi = 1.0$) e 130 s (para $\phi =$

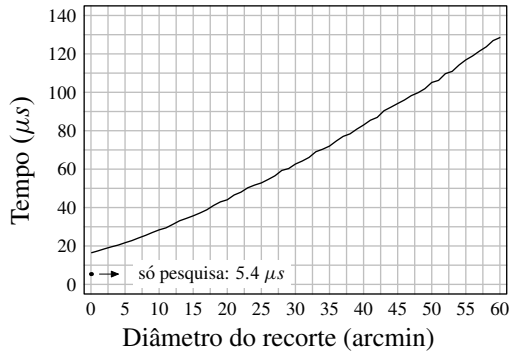


Figura 10.3: Tempo médio de execução de um recorte no catálogo UCAC4 em função do diâmetro do recorte.

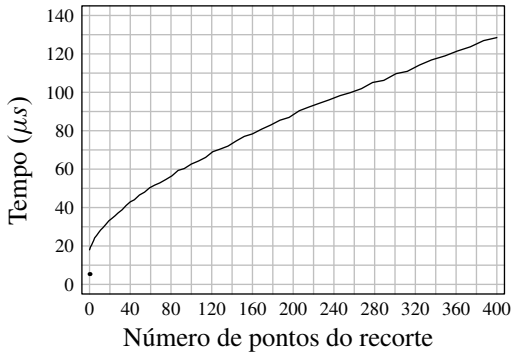


Figura 10.4: Tempo médio de execução de um recorte no catálogo UCAC4 em função do número de pontos obtidos.

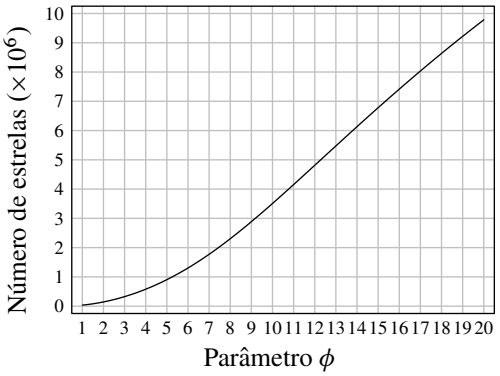
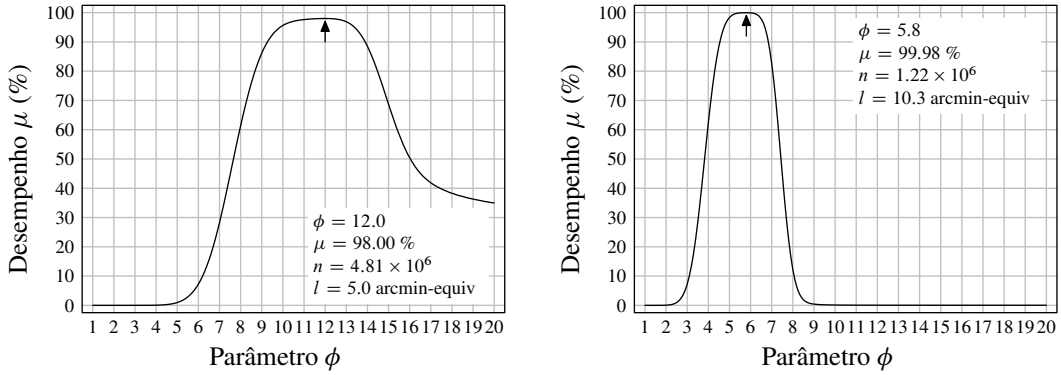


Figura 10.5: Número de estrelas nos catálogos reduzidos.

20.0), sendo a diferença justificada pelo maior número de remoções de locais no diagrama de Voronoi operadas nos catálogos *mais* reduzidos. A redução dos 191 catálogos foi executada em cerca de 10 horas.

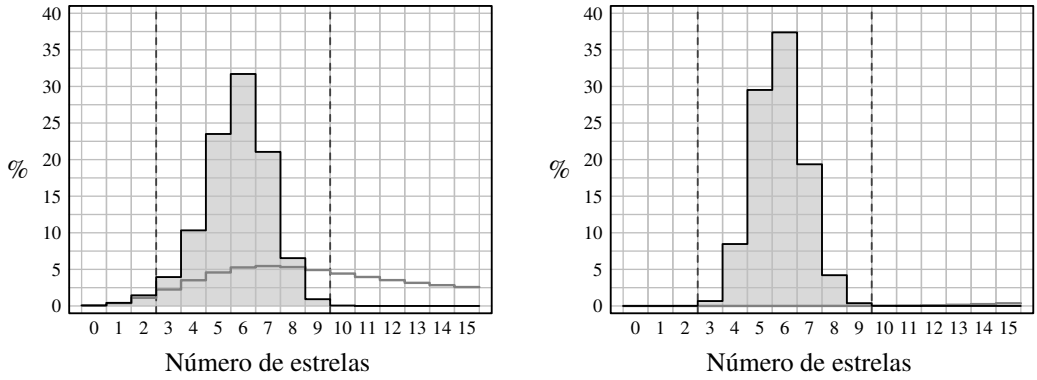
A Figura 10.5 apresenta o tamanho de cada catálogo reduzido em função do parâmetro ϕ. É este gráfico que justifica a escolha do parâmetro ϕ para a especificação do limiar de área no algoritmo de redução. O desempenho de um catálogo reduzido deve depender do número de estrelas que contém. Logo, é razoável que a procura do catálogo com melhor desempenho seja feita por uma pesquisa linear no número de estrelas. O parâmetro ϕ, ao linearizar o número de estrelas nos catálogos reduzidos, simplifica esta tarefa.

A Figura 10.6 apresenta o desempenho dos 191 catálogos reduzidos para cada um dos dois diâmetros do campo de visão alvo: 15 arcmin e 30 arcmin. O desempenho foi medido pelo método de Monte Carlo, aplicando a equação (8.3). Os dois gráficos exibem um comportamento semelhante. Para valores de ϕ baixos (redução elevada), o desempenho tende para zero. Para valores de ϕ elevados (redução baixa), o desempenho tende para o valor medido com o catálogo completo (c.f. Tabela 8.1). O máximo absoluto identifica o melhor catálogo reduzido para cada campo de visão: T_{15} para 15 arcmin e T_{30} para 30 arcmin. Para um campo de visão de 15 arcmin, não foi possível obter um desempenho superior a 98.00%. Ao se aumentar o campo de visão de 15 para 30 arcmin, obtém-se uma melhoria significativa



(a) 15 arcmin (b) 30 arcmin

Figura 10.6: Desempenho dos catálogos reduzidos em função de ϕ. Estão indicados os valores para o catálogo com melhor desempenho: o parâmetro ϕ, o desempenho μ, o número de estrelas n e o limiar de área l em arcmin-equiv.



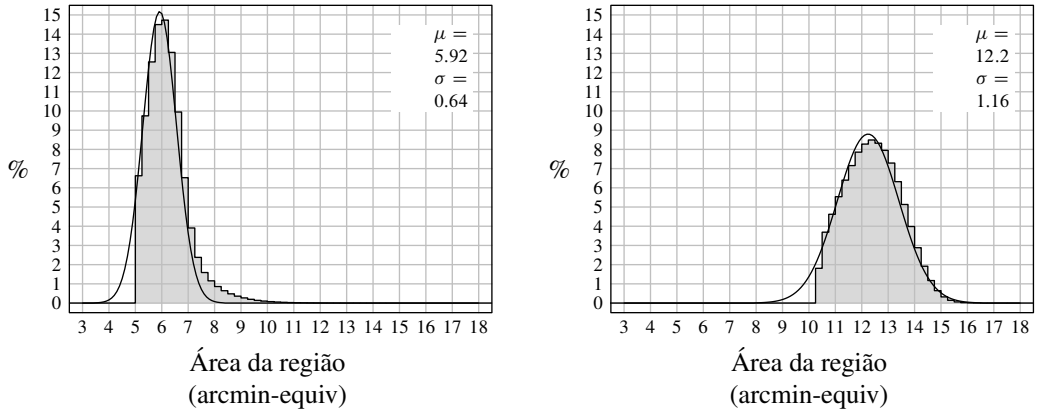
(a) Catálogo T_{15} (b) Catálogo T_{30}

Figura 10.7: Histogramas de desempenho dos catálogos seleccionados.

do desempenho, que ultrapassa os 99.9%. Claramente, a redução para um campo de visão de 15 arcmin é mais exigente que a redução para um campo de visão de 30 arcmin. Uma diminuição do campo visão aumenta a parte do catálogo em que o número de estrelas num recorte é insuficiente (< 3), o que reduz o melhor desempenho obtenível.

Vamos agora analisar em detalhe os dois catálogos atrás seleccionados, T_{15} e T_{30}, observando os respectivos histogramas de desempenho (ver Figura 10.7) e histogramas de áreas (ver Figura 10.8). Sobreposto a cada histograma de desempenho, está desenhado o histograma da Figura 8.4 (derivado da análise do catálogo UCAC4), para o campo de visão correspondente. A medida de desempenho de cada catálogo corresponde à soma dos valores das colunas entre 3 e 9, inclusive (marcadas a tracejado).

O algoritmo de redução, ao eliminar estrelas de um catálogo, modifica o histograma de desempenho "deslocando-o", como um todo, no sentido das contagens menores. Neste processo de "deslocamento de histograma", há uma parte que se acumula entre as colunas 3 e 9, e que contribui positivamente para o desempenho, enquanto que o resto é repartido por dois interva-



(a) Catálogo T_{15} (b) Catálogo T_{30}

Figura 10.8: Histogramas das áreas dos catálogos seleccionados. A curva sólida representa o ajuste por uma Gaussiana (de parâmetros μ, σ).

los: abaixo de 3 estrelas e acima de 9 estrelas. Para o campo de visão de 15 arcmin, observa-se um ligeiro incremento nas contagens com duas estrelas, enquanto que para o campo de visão de 30 arcmin esse incremento é imperceptível. No extremo oposto, também é muito reduzido o número de contagens para 10 estrelas (sendo nulo para mais de 10 estrelas). Em resumo, este resultado mostra como o algoritmo de redução é eficaz em produzir um catálogo com o número de estrelas no campo de visão dentro dos limites impostos.

Uma outra forma de analisar os desempenhos obtidos é por intermédio dos histogramas das áreas das regiões de Voronoi (Figura 10.8). Os histogramas das áreas são indicadores da regularidade da distribuição espacial. Se o número de estrelas no campo de visão tende para um valor constante, então será de esperar que as estrelas do catálogo estejam aproximadamente equidistantes. Consequentemente, as áreas das regiões de Voronoi também devem tender para um valor constante. E é o que se observa em ambos os histogramas de áreas, cujas curvas seguem a forma de uma Gaussiana (representada por uma linha sólida), com excepção de dois pormenores. Do lado das áreas menores, ambos os histogramas estão truncados pelo limiar de área usado na construção de cada catálogo (c.f. Figura 10.6). E no histograma do catálogo T_{15}, o excesso no histograma (acima da curva Gaussiana) do lado das áreas maiores revela a existência de uma parte do catálogo não conforme com a média de áreas das regiões. Corresponde à parte do catálogo UCAC4 em que a densidade de estrelas é inferior à necessária para satisfazer o requisito mínimo de três estrelas no campo de visão. O catálogo T_{30} não apresenta esse sintoma.

A análise dos histogramas de áreas revela a dificuldade em um único catálogo reduzido satisfazer os dois limites do número de estrelas no campo de visão: por um lado quer-se três ou mais estrelas, por outro lado quer-se nove estrelas ou menos. Claramente, o primeiro requisito é o mais importante: menos de três estrelas impedem a determinação da atitude. Comparativamente, o segundo critério é menos crítico; mais de nove estrelas é apenas considerado desperdício de memória. Vamos analisar estes dois critérios em separado, por intermédio de mapas de falhas. Um mapa de falhas assinala posições da esfera celeste onde se observa que o

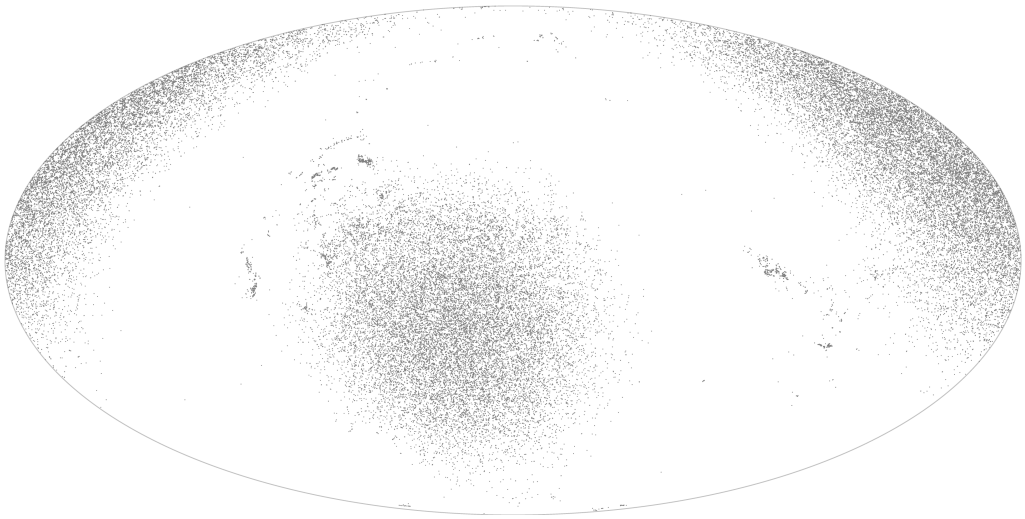


Figura 10.9: Mapa de falhas por defeito para o catálogo T_{15} (falha em 2.06 %).

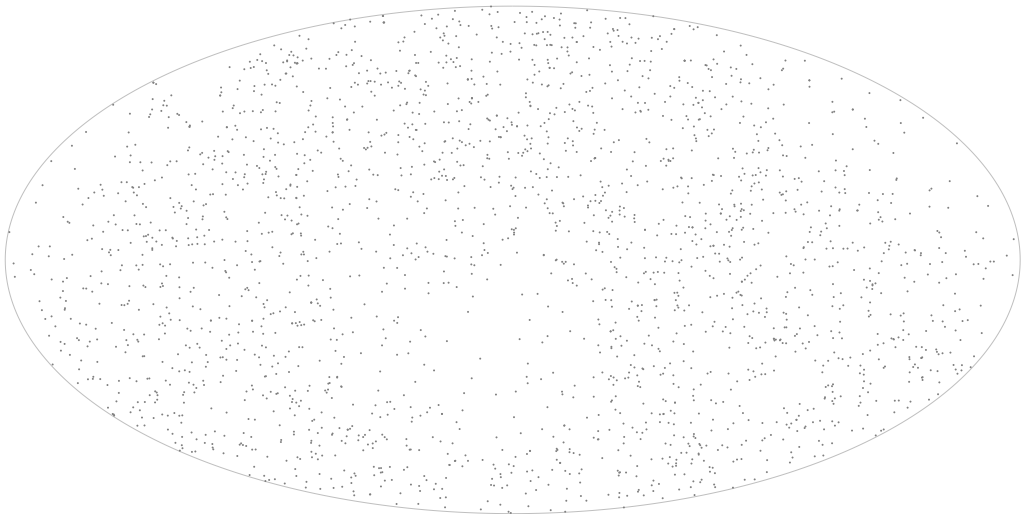


Figura 10.10: Mapa de falhas por excesso para o catálogo T_{15} (falha em 0.06 %).

número de estrelas (num recorte) não cumpre os requisitos. As posições são escolhidas aleatoriamente, segundo uma distribuição uniforme, seguindo a mesma estratégia usada no cálculo do desempenho dos catálogos. Desta forma, um mapa de falhas representa uma amostra das posições em que um catálogo falha os requisitos e não todas as possíveis localizações em que um catálogo falha os requisitos. Para garantir a representatividade da amostragem, os mapas que se seguem foram construídos a partir de 4×10^6 posições aleatórias. Os mapas de falhas são de dois tipos: os mapas de falhas por defeito, que assinalam posições com menos de três estrelas, e os mapas por excesso, que assinalam posições com mais de nove estrelas.

As Figuras 10.9 e 10.10 apresentam, respectivamente, os mapas de falhas por defeito e por excesso para o catálogo T_{15}. Registaram-se falhas por defeito da ordem de 2.06 % e falhas por excesso da ordem de 0.06 %. Em conjunto, estes mapas revelam a razão da dificuldade em se conseguir um catálogo reduzido com desempenho elevado para o campo

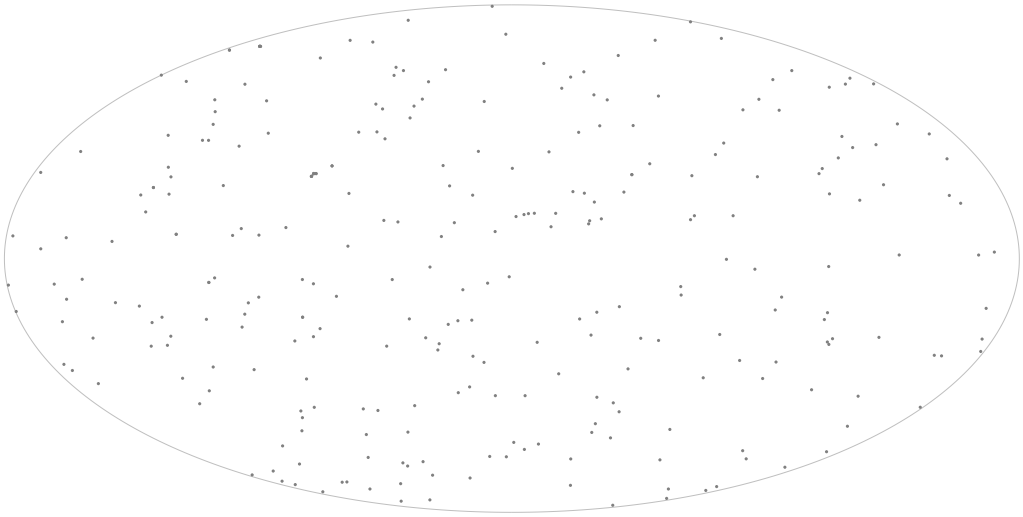


Figura 10.11: Mapa de falhas por defeito para o catálogo T_{30} (falha em 0.007 %).

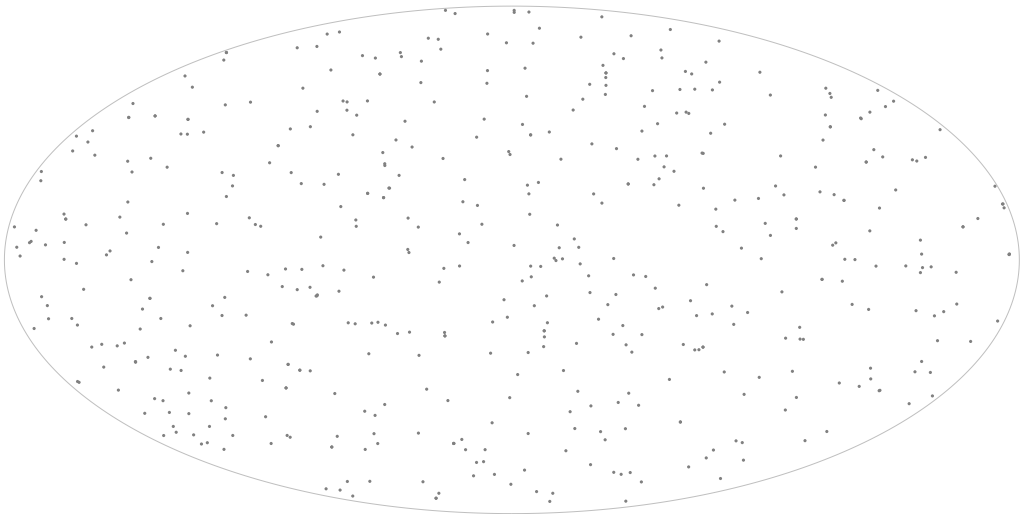


Figura 10.12: Mapa de falhas por excesso para o catálogo T_{30} (falha em 0.013 %).

de visão de 15 arcmin. Nas regiões em volta dos pólos da galáxia (região central e "ombros" da elipse da Figura 10.9) a densidade do catálogo UCAC4 é, em média, insuficiente para garantir um número mínimo de três estrelas no campo de visão, sendo responsável pela maior falta de desempenho. Por outro lado, apesar da densidade média do catálogo UCAC4 ao longo do plano da galáxia (c.f. região mais densa na Figura 8.2) superar o máximo requerido, observam-se falhas por excesso por toda esta região (embora em muito menor grau que as falhas nas regiões polares).

As Figuras 10.11 e 10.12 apresentam, respectivamente, os mapas de falhas por defeito e por excesso para o catálogo T_{30}. Comparativamente, agora o número de falhas é muito inferior, registando-se falhas por defeito da ordem de 0.007 % e falhas por excesso da ordem de 0.013 %. No entanto, e o que é mais relevante, agora não se observa nenhuma aglomeração de falhas (em nenhum dos casos), que são, simultaneamente, em número insignificante. Ou seja,

as falhas registadas devem-se a características locais da densidade do catálogo UCAC4, não penalizando o desempenho global do catálogo. Portanto, para o campo de visão de 30 arcmin, foi possível construir um catálogo reduzido que cumpre os requisitos na globalidade da esfera celeste.

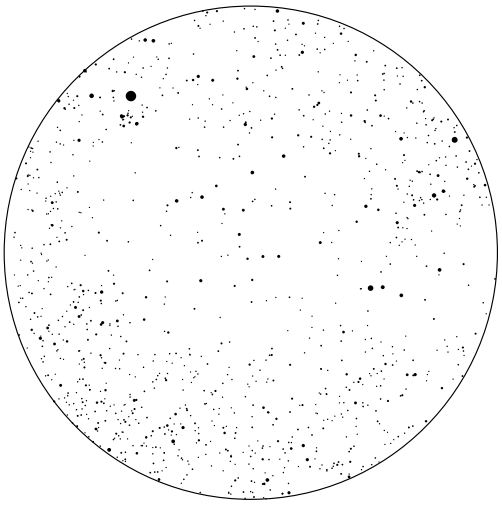
Para finalizar, vamos observar o resultado do algoritmo de redução comparando recortes nos catálogos UCAC4, T_{30} e T_{15}. Todos os recortes têm um diâmetro de 180 arcmin. O centro de cada recorte é indicado em coordenadas equatoriais pelo par ascensão recta e declinação, (α, δ). Cada estrela é indicada por um pequeno círculo, cujo raio é directamente proporcional ao brilho.

As Figuras 10.13 e 10.14 apresentam recortes centrados, respectivamente, na zona de menor densidade e na zona de maior densidade do catálogo UCAC4. Estes dois exemplos ilustram a hipótese formulada na construção do algoritmo de redução: a remoção das estrelas é proporcional à densidade local. Ou seja, a densidade do catálogo reduzido é aproximadamente constante, independentemente da densidade (local) do catálogo inicial. Regra geral, comparando o catálogo T_{15} com o catálogo T_{30}, também se observa que o algoritmo de redução retém, preferencialmente, as estrelas mais brilhantes.

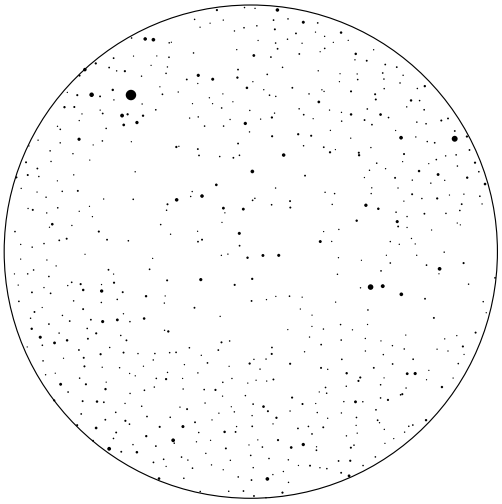
Os círculos vazios máximos centrados nos vértices do diagrama de Voronoi constituem uma forma expedita de avaliar o "espaço livre" disponível. Em particular, permitem conhecer o maior buraco circular vazio de um catálogo. A Figura 10.15 apresenta os recortes centrados no maior círculo vazio máximo, igual nos três catálogos, com um diâmetro de 30.45 arcmin (indicado pelo círculo central). Esta figura ilustra uma propriedade expectável do algoritmo de redução: a redução não aumenta a "fronteira" das regiões vazias. Isto é, se uma estrela faz fronteira com uma região vazia (à escala do campo de visão), então essa estrela não é removida. Portanto, verifica-se que os efeitos da redução se manifestam maioritariamente nas zonas mais densas não afectando as zonas vazias.

Pela análise anterior conclui-se que o algoritmo de redução satisfaz plenamente os requisitos quantitativos definidos no capítulo 8. Resta apenas analisar o requisito qualitativo de reter preferencialmente as estrelas mais brilhantes. No algoritmo, este requisito foi contemplado analisando as estrelas por ordem crescente de brilho. Desta forma, a remoção de uma estrela de menor brilho aumenta a área das regiões das estrelas ainda não analisadas (que são mais brilhantes) promovendo a sua retenção. Porém, também pode ocorrer o oposto. Uma estrela mais brilhante que as suas vizinhas mas cuja região tem uma área inferior ao limiar só escapa à remoção se algumas (ou todas) as estrelas vizinhas forem removidas. Caso contrário, pode observar-se a eliminação de estrelas muito mais brilhantes que as que a circundam, contrariando o requisito do problema. Os círculos nas Figuras 10.15 e 10.16 (esta ilustrando um recorte em torno da Estrela Polar) assinalam este comportamento, em que uma estrela brilhante pode estar contida no catálogo T_{15} mas não no catálogo T_{30}, e vice-versa.

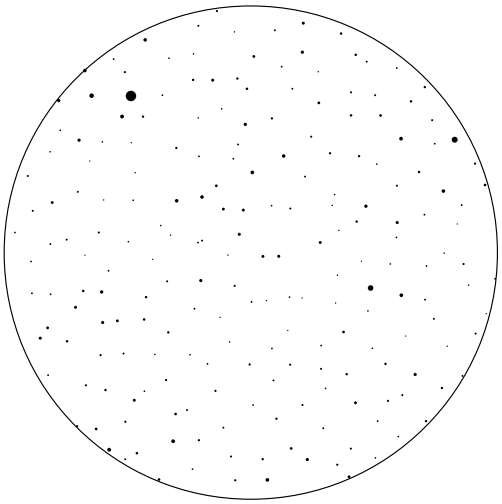
A justificação para este comportamento deve-se ao conflito inerente aos dois tipos de requisitos. No desenho do algoritmo de redução foi dada prioridade aos requisitos quantitativos (três a nove estrelas no campo de visão), o que pode implicar a remoção indiscriminada de estrelas, mesmo que relevantes do ponto de vista qualitativo. Esta desvantagem poderia ser



UCAC4

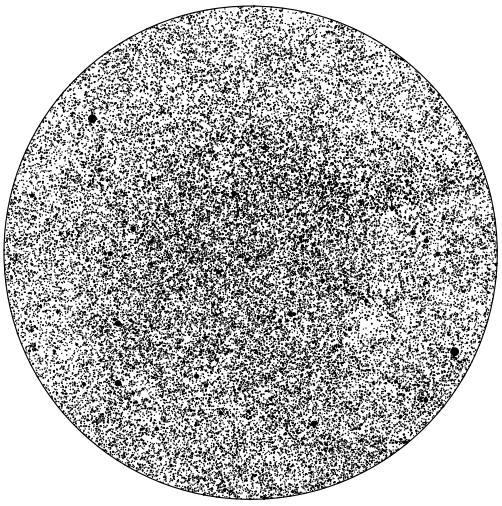


T_{15}

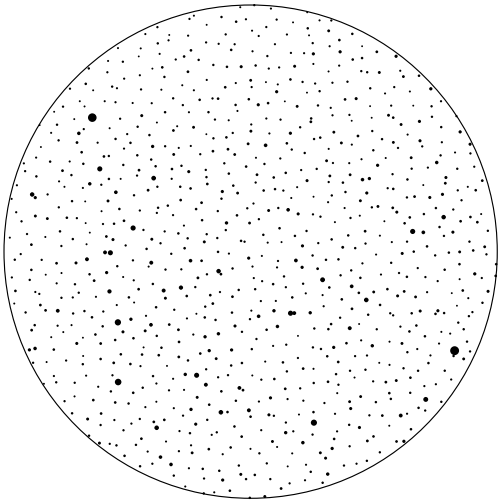


T_{30}

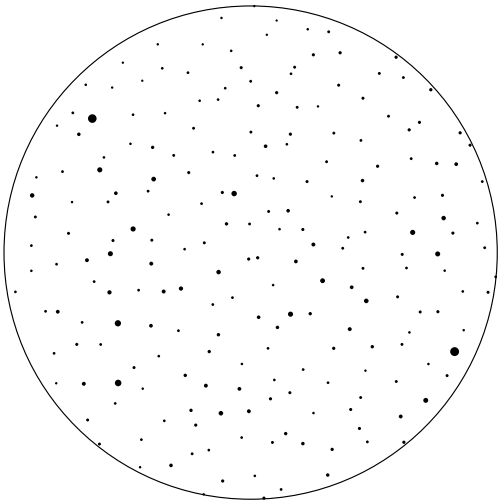
Figura 10.13: Resultado da redução em redor de uma zona de densidade reduzida ($\alpha = 03^h\ 40^m\ 52^s$, $\delta = 31^\circ\ 20'\ 19''$).



UCAC4

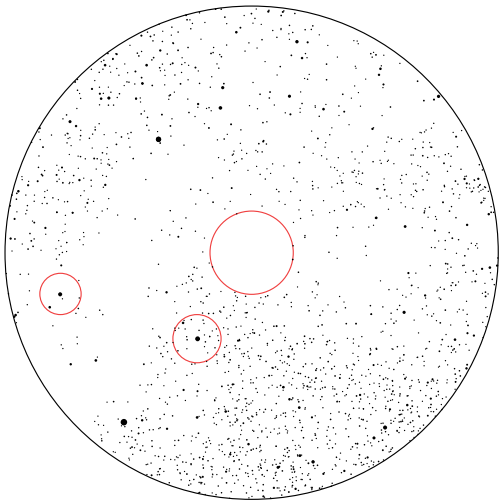


T_{15}

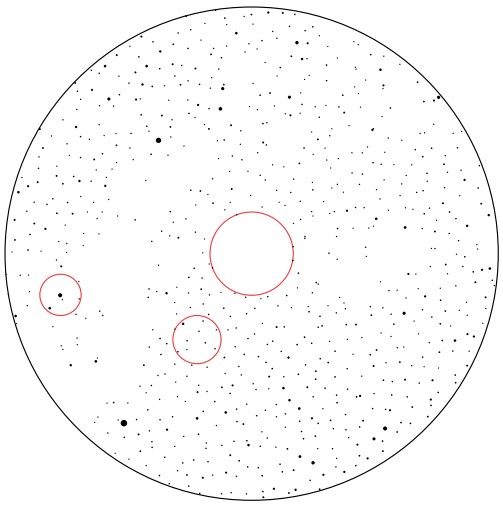


T_{30}

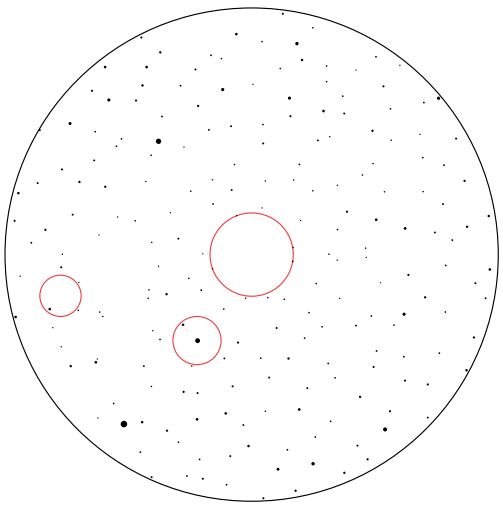
Figura 10.14: Resultado da redução em redor de uma zona de densidade elevada ($\alpha = 18^h\ 13^m\ 44^s$, $\delta = -27^\circ\ 51'\ 43''$).



UCAC4

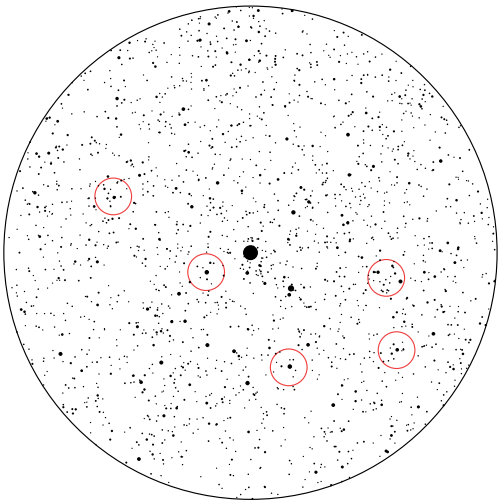


T_{15}

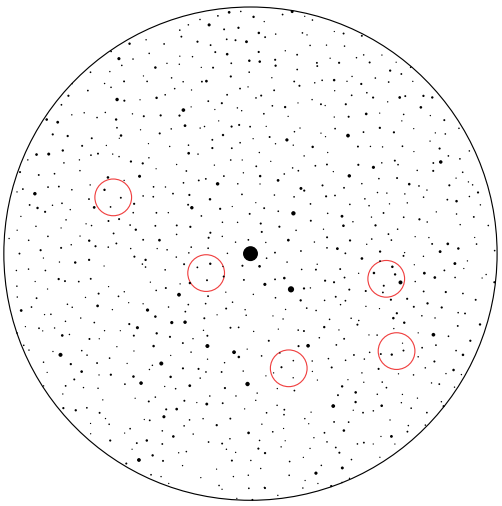


T_{30}

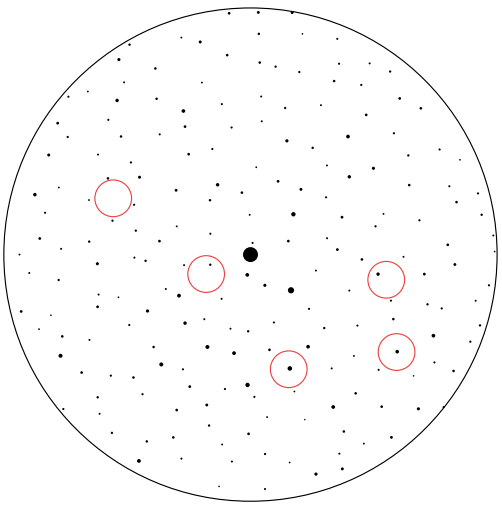
Figura 10.15: Resultado da redução em redor do maior "buraco" (círculo vazio máx.) ($\alpha = 04^h\ 35^m\ 21^s$, $\delta = 26°\ 15'\ 06''$).



UCAC4



T_{15}



T_{30}

Figura 10.16: Resultado da redução em redor da Estrela Polar (α UMi) ($\alpha = 02^h\ 31^m\ 52^s$, $\delta = 89°\ 15'\ 51''$).

compensada re-analisando as estrelas removidas uma segunda vez, agora por ordem decrescente de brilho, reinserindo estrelas cuja brilho supera o brilho das estrelas circundantes e que, por isso, seria vantajoso que pertencessem ao catálogo reduzido. No entanto, a prioridade relativa a atribuir a cada requisito depende, fundamentalmente, das propriedades do algoritmo de identificação de estrelas em imagens. Como é óbvio, os dois algoritmos — redução de catálogo e identificação de estrelas — devem estar em sintonia. Para se maximizar a robustez na determinação da atitude de uma nave, as estrelas que compõem um catálogo reduzido devem corresponder às mais prováveis de identificar em imagens. Independentemente destas considerações, o algoritmo de redução de catálogos de estrelas revelou-se uma solução robusta, constituindo uma base sólida para posteriores refinamentos.

Em conclusão, mostrou-se experimentalmente que o algoritmo de redução produziu resultados satisfatórios, tendo sido capaz de produzir catálogos com uma efectiva redução do número de estrelas e com um desempenho elevado. É de realçar como um resultado exigente foi obtido por um critério relativamente simples: a eliminação de estrelas com base na comparação da área da respectiva região de Voronoi. Sem dúvida, é a capacidade de síntese de vizinhanças espaciais do diagrama de Voronoi que possibilita tal simplicidade de estratégia.

Capítulo 11

Conclusões

O tema central desta dissertação são os diagramas de Voronoi na esfera. Na primeira parte da tese foi apresentado um novo algoritmo para o cálculo do diagrama de Voronoi de locais pontuais na esfera, que é uma adaptação da técnica de varrimento do plano ao domínio esférico. O varrimento do plano é uma técnica comum que está na base de muitos algoritmos geométricos, aplicada inicialmente por Shamos e Hoey [55] num algoritmo de detecção de intersecções de segmentos de recta.

O que se demonstra inovador neste trabalho é a adaptação da técnica de varrimento do plano à superfície da esférica, aqui aplicada a um algoritmo de construção do diagrama de Voronoi esférico, embora através de uma variante menos usual. Com efeito, o novo algoritmo não imita o varrimento do plano do algoritmo de Fortune [34], mas sim o varrimento circular do algoritmo de Dehne e Klein [21], que se mostrou capaz de se adaptar à natureza circular da esfera.

O varrimento da superfície da esfera é feito por um círculo de raio crescente, centrado num ponto arbitrário. Se centrado no pólo Norte, a superfície da esfera é varrida percorrendo paralelos de latitude até que, quando o raio do círculo atinge π, o círculo de varrimento concentra-se num único ponto, o pólo Sul. Para a construção do diagrama de Voronoi, é necessário prosseguir o varrimento, aumentando o raio do círculo de varrimento para valores superiores a π, o que corresponde a fazer um segundo varrimento da esfera, agora de Sul para Norte. Foi demonstrado que o duplo varrimento não introduz qualquer descontinuidade no processo de varrimento, nem mesmo quando um local está posicionado no pólo Sul.

O novo algoritmo apresenta o mesmo custo computacional que os algoritmos equivalentes no plano: calcula o diagrama de Voronoi esférico de n locais em tempo $O(n \log n)$ e espaço $\Theta(n)$, que é óptimo no pior caso. Além do mais, os resultados experimentais confirmaram que o algoritmo é tão prático, eficiente e de fácil implementação como o algoritmo de Fortune, sendo a escolha preferencial para o cálculo do diagrama de Voronoi na esfera.

O técnica de varrimento circular tem a vantagem de permitir a construção parcial de um diagrama de Voronoi. Por exemplo, possibilita a construção eficiente da região de Voronoi de um único local, centrando o varrimento no local e parando o varrimento assim que a frente de

onda termina o varrimento da região.

Mas a versatilidade do varrimento circular não se restringe à construção parcial de um diagrama. Neste trabalho foram desenvolvidos dois algoritmos de edição de diagramas de Voronoi — um para a remoção de um local e outro para a inserção de um novo local — ambos baseados na técnica de varrimento circular. A inserção de um local num diagrama pode ser decomposta em duas partes: construção da região do novo local seguida da sua "colagem" ao diagrama. O que se descobriu é que estas duas operações podem ser implementadas por um único varrimento (circular), com o varrimento "apoiado" no diagrama em edição. Mais exactamente, a inserção faz-se com um varrimento circular construído de forma a que as intersecções de arcos da frente de onda percorram as arestas e vértices que devem ser eliminados do diagrama, abrindo espaço para a região de Voronoi do novo local. A operação de remoção de um local faz-se de um modo muito semelhante. Conceptualmente, equivale a efectuar o mesmo varrimento que o da inserção, mas em sentido inverso. Neste caso, o varrimento descobre arestas e vértices que devem ser adicionados ao diagrama para implementar a remoção. Como ambos os algoritmos seguem a estratégia de varrimento circular, aplicam-se igualmente à edição de diagramas no plano e na esfera.

O custo computacional dos algoritmos de edição depende do número de arestas m da região inserida ou removida e, numa inserção, do número de arestas k da região que contém o local a inserir. A inserção é feita em tempo $O(m+k)$ e espaço $O(n)$, enquanto que a remoção é feita em tempo $O(m \log m)$ e espaço $\Theta(m)$.

A aplicação da técnica de varrimento à esfera abre novas possibilidades a algoritmos anteriores, baseados na técnica de varrimento. É de esperar que estes possam ser adaptados ao domínio esférico, desde que seja possível convertê-los num varrimento circular. Um exemplo é o algoritmo de pesquisa do vizinho mais próximo de Dinis e Mamede [28]. Dados dois conjuntos de pontos, uns vermelhos e outros azuis, este algoritmo usa a frente de onda do algoritmo de Fortune para encontrar um ponto azul mais próximo de cada ponto vermelho. Este algoritmo é um corolário do algoritmo de Fortune. Durante o varrimento que constrói o diagrama de Voronoi dos pontos vermelhos, o arco da frente de onda que cruza cada ponto azul identifica o ponto vermelho que lhe está mais próximo. Embora desenvolvido no domínio planar, é trivial a sua adaptação ao domínio esférico.

Na segunda parte da tese exemplificou-se a aplicação dos algoritmos de construção e edição de diagramas de Voronoi esféricos. Foi atacado o problema de redução de um catálogo de estrelas, que não é mais que uma remoção selectiva de estrelas de um catálogo, segundo um critério pré-definido. Com a redução, pretende-se determinar um sub-conjunto de estrelas de um catálogo cuja distribuição espacial seja aproximadamente uniforme, formado por estrelas aproximadamente equi-espaçadas entre si. A solução desenvolvida fez uso dos diagramas de Voronoi esféricos em diferentes etapas.

Em primeiro lugar, os diagramas de Voronoi foram usados como ferramenta de análise da vizinhança espacial de uma estrela. Uma região de Voronoi com uma área relativamente pequena indica que a estrela correspondente está muito próxima de estrelas vizinhas e, por

isso, pode ser removida do catálogo.

Em segundo lugar, os diagramas de Voronoi foram usados como peça fundamental da hierarquia de Voronoi [23, 36], que é uma estrutura de dados desenhada para responder eficientemente a pesquisas de localização. Conceptualmente, uma hierarquia é composta por vários diagramas de Voronoi, dispostos em camadas e interligados entre si, em que os locais de cada diagrama são obtidos por uma selecção aleatória e parcial dos locais da camada inferior. A eficiência de uma pesquisa advém da forma como se efectua um percurso numa hierarquia, em que os diagramas de Voronoi são percorridos do topo para a base. Mais concretamente, passeios nas camadas mais altas permitem uma aproximação mais rápida do ponto de pesquisa, enquanto que os passeios nas camadas mais baixas permitem uma aproximação mais precisa. O local pretendido é encontrado pelo passeio na camada de base.

Em terceiro lugar, os diagramas de Voronoi foram utilizados para a operação de recorte circular, onde se selecciona o sub-conjunto dos pontos contidos num círculo. O recorte é suportado pelo diagrama de Voronoi de um conjunto de pontos. Começando no local mais próximo do centro de recorte, os restantes pontos são obtidos por um percurso em largura no diagrama de Voronoi. Para este fim, o diagrama de Voronoi é utilizado como estrutura de indexação espacial.

O algoritmo de redução foi aplicado ao catálogo de estrelas UCAC4 [64], tendo sido produzidos dois catálogos reduzidos, específicos para dois diâmetros do campo de visão (um de 15 arcmin e outro de 30 arcmin). Foi também medido e analisado o desempenho de cada um destes catálogos. Globalmente, observou-se um desempenho muito positivo, confirmando a validade do algoritmo de redução de catálogos desenvolvido.

O resultado do trabalho desenvolvido permite a construção de diagramas de Voronoi de pontos na superfície da esfera. A seguir são discutias algumas linhas de desenvolvimento futuro.

Com o processamento do catálogo UCAC4, restringido às 21×10^6 estrelas mais brilhantes, foi obtido um diagrama de Voronoi que ocupa cerca de 2 GB. A partir do catálogo completo, com cerca de 114×10^6 estrelas, obter-se-ia um diagrama que ocuparia mais de 11 GB. A tendência actual aponta para uma recolha cada vez mais massiva de dados, tanto no domínio das ciências da Terra como no domínio da astronomia. Um exemplo disso é a missão GAIA [45], da agência espacial europeia (ESA), que inclui o lançamento de um satélite com o objectivo de medir a posição absoluta de cerca de 10^9 estrelas. Com este volume de dados, torna-se evidente que não é exequível implementar o diagrama de Voronoi em memória principal.

A solução mais óbvia parece ser a adopção de uma estrutura semelhante à de uma árvore-B^+ [19], que segmenta o volume de dados em partes manejáveis, implementadas em memória secundária. De facto, e como já foi notado anteriormente [36], a hierarquia de Voronoi, apresentada no capítulo 9, parece adaptar-se naturalmente a essa solução.

Mas, para uma construção eficiente de um diagrama de Voronoi de um conjunto de dados massivo, como o conjunto de 10^9 estrelas atrás referido, seria interessante investigar a pos-

sibilidade da construção paralela do diagrama, varrendo o domínio com vários círculos em simultâneo. No caso geral, a região de Voronoi de um local tem um alcance limitado, não formando arestas com locais muito afastados. Logo, será de esperar que varrimentos circulares em paralelo se possam fazer, na maioria dos casos, de forma independente, necessitando de se sincronizar apenas quando os varrimentos se cruzam.

As duas ideias apresentadas – implementação em memória secundária e construção paralela – parecem complementar-se. Pode imaginar-se que a repartição de um conjunto (massivo) de dados, necessária para a construção de uma hierarquia de Voronoi, seja apropriada para a construção do diagrama em paralelo. Para completar, seriam também necessários algoritmos de actualização da hierarquia de Voronoi, quando implementada em memória externa.

A segunda proposta de trabalho está relacionada com a forma dos locais. Neste trabalho apenas foi discutida a construção de diagramas de Voronoi de locais com forma pontual. Subindo na escala de complexidade geométrica, encontram-se os locais com a forma de segmentos de recta. Os segmentos de recta são os elementos geométricos naturais para representar certos tipos de objectos em aplicações geográficas, tais como percursos (como estradas, rios, etc.) ou fronteiras de áreas (como lagos, territórios, etc.). Sendo os sistemas de informação geográfica um domínio natural de aplicação dos diagramas de Voronoi esféricos, seria interessante adaptar os novos algoritmos de construção e de edição a locais representados por pontos e segmentos de círculo máximo. No plano, existem diversas soluções para o problema [44, 42, 43, 8]. Em particular, é possível construir este diagrama pelo algoritmo de Fortune [34], embora a implementação não seja trivial. A complicação deriva da forma geométrica da mediatriz entre dois segmentos de recta (ou entre um ponto e um segmento de recta) que agora consiste numa sequência de linhas rectas e arcos de parábola. Esta forma mais complexa origina um número alargado de eventos-círculo, complicando a determinação de eventos e o cálculo das prioridades respectivas. O desafio é conceber um algoritmo de construção de diagramas de segmentos de recta baseado num varrimento circular e, em particular, adaptá-lo ao domínio esférico.

A última proposta tem origem na forma geométrica da Terra: pode o algoritmo de varrimento ser adaptado à construção do diagrama de Voronoi num elipsóide? Em primeira aproximação, a Terra tem a forma de uma esfera. No entanto, o objecto geométrico que mais se aproxima da forma real da Terra é o elipsóide oblato [59], sendo este o modelo mais rigoroso usado em sistemas de informação geográfica. Daí o interesse em construir um diagrama de Voronoi num elipsóide.

A adaptação directa do algoritmo de varrimento ao elipsóide parece ser exequível, uma vez que este possui a mesma simetria circular que a esfera. Pode imaginar-se o mesmo processo de varrimento, com um círculo de raio crescente, centrado no pólo Norte, percorrendo paralelos de latitude de Norte para Sul. Aparentemente, o varrimento do elipsóide possui as mesmas características que o varrimento esférico. Isto é, o círculo de varrimento e um local definem uma elipse na superfície do elipsóide, o envelope inferior de todas as elipses define uma frente de onda formada por arcos de elipses e as intersecções de arcos percorrem as ares-

tas do diagrama de Voronoi. Estas propriedades são facilmente deduzidas por analogia com o varrimento esférico.

Portanto, será de esperar que a construção do diagrama de Voronoi elipsoidal se possa fazer de modo igual ao do varrimento esférico, recorrendo às mesmas estruturas de dados. As diferenças deverão concentrar-se no cálculo das prioridades de cada evento, no cálculo da relação de ordem usada na frente de onda e, em particular, no cálculo do círculo circunscrito a três pontos. Mas, ao contrário dos círculos na superfície da esfera, um círculo no elipsóide não admite uma expressão analítica, não sendo o seu cálculo trivial.

Uma forma expedita de construir uma aproximação do diagrama de Voronoi num elipsóide é por intermédio de uma projecção numa *esfera auxiliar*. É um método habitual em cartografia [59], como passo intermédio na planificação de um elipsóide. Para isso, as coordenadas geodéticas de latitude e longitude (no elipsóide) são convertidas em coordenadas geocêntricas de latitude e longitude (na esfera). Uma vez obtido o diagrama na esfera auxiliar, o diagrama no elipsóide é construído projectando as posições dos vértices no sentido inverso, da esfera auxiliar para o elipsóide.

Dado o diminuto achatamento do elipsóide terrestre, pode argumentar-se que a aproximação dada pelo diagrama numa esfera auxiliar é suficiente para uma grande maioria de aplicações. No entanto, seria interessante aferir a magnitude do erro cometido nesta aproximação e identificar as condições que maximizam esse erro.

Bibliografia

[1] AGGARWAL, A., GUIBAS, L., SAXE, J., AND SHOR, P. A linear time algorithm for computing the Voronoi diagram of a convex polygon. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC'87)* (New York, NY, USA, 1987), ACM, pp. 39–45.

[2] AURENHAMMER, F. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys 23*, 3 (1991), 345–405.

[3] AURENHAMMER, F., AND KLEIN, R. Voronoi diagrams. In *Handbook of Computational Geometry*, J. Sack and G. Urrutia, Eds. Elsevier Science Publishing, 2000, pp. 201–290.

[4] BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software 22*, 4 (1996), 469–483.

[5] BARBER, C. B., AND HUHDANPAA, H. Qhull (ver. 2011.1), 2011. `http://www.qhull.org`.

[6] BENTLEY, J., AND OTTMANN, T. Algorithms for reporting and counting geometric intersections. *Computers, IEEE Transactions on C-28*, 9 (Sept. 1979), 643 –647.

[7] BOISSONNAT, J.-D., DEVILLERS, O., PION, S., TEILLAUD, M., AND YVINEC, M. Triangulations in CGAL. *Computational Geometry: Theory and Applications 22*, 1-3 (2002), 5–19.

[8] BOISSONNAT, J.-D., DEVILLERS, O., SCHOTT, R., TEILLAUD, M., AND YVINEC, M. Applications of random sampling to on-line algorithms in computational geometry. *Discrete & Computational Geometry 8* (1992), 51–71.

[9] BROWN, K. Q. *Geometric transforms for fast geometric algorithms*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1979.

[10] CAROLI, M., M. M. DE CASTRO, P., LORIOT, S., ROUILLER, O., TEILLAUD, M., AND WORMSER, C. Robust and efficient Delaunay triangulations of points on or close to a sphere. In *Proceeding of the Symposyum on Experimental Algorithms (SEA'10)* (2010), pp. 462–473.

[11] CGAL: Computational Geometry Algorithms Library (release 3.9), 2011. http://www.cgal.org.

[12] CHAZELLE, B. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry 10* (1993), 377–409.

[13] CHEE YAP, T. D. The exact computation paradigm. In *Computing in Euclidean Geometry*, D. Z. Du and F. K. Hwang, Eds., second ed. World Scientific Press, Singapore, 1995, pp. 452–486.

[14] CHEW, L. P. Building Voronoi Diagrams for Convex Polygons in Linear Expected Time. Tech. Rep. PCS-TR90-147, Dartmouth College, Computer Science, Hanover, NH, Jan. 1990.

[15] CLARKSON, K. L. Safe and effective determinant evaluation. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS'92)* (1992), IEEE Computer Society, pp. 387–395.

[16] CLARKSON, K. L. Hull (ver. 1.0), 1995. http://netlib.org/voronoi/hull.html.

[17] CLARKSON, K. L., MEHLHORN, K., AND SEIDEL, R. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications 3*, 4 (1993), 185–212.

[18] CLARKSON, K. L., AND SHOR, P. W. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry 4*, 1 (1989), 387–421.

[19] CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. *Introduction to Algorithms*, second ed. MIT Press, Cambridge, MA, 2001.

[20] DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. *Computational Geometry: Algorithms and Applications*, second ed. Springer-Verlag, Berlin, 2000.

[21] DEHNE, F., AND KLEIN, R. A sweepcircle algorithm for Voronoi diagrams. In *Proceedings of the 13th Graph-Theoretic Concepts in Computer Science* (Kloster Banz/Staffelstein, Germany, 1987), pp. 59–83.

[22] DEHNE, F. K. H. A., AND KLEIN, R. The big sweep : On the power of the wavefront approach to Voronoi diagrams. *Algorithmica 17*, 1 (1997), 19–32.

[23] DEVILLERS, O. The Delaunay hierarchy. *International Journal of Foundations of Computer Science 13* (2002), 163–180.

[24] DEVILLERS, O. On deletion in Delaunay triangulation. *International Journal of Computational Geometry and Applications 12* (2002), 193–205.

[25] DEVROYE, L., MÜCKE, E. P., AND ZHU, B. A note on point location in Delaunay triangulations of random points. *Algorithmica 22* (1998), 477–482.

[26] DINIS, J. Reconhecimento de padrões de conjuntos de pontos. Tese de mestrado, Faculdade de Ciências e Tecnologia da UNL, Jan. 2003.

[27] DINIS, J. Star identification. Tech. Rep. 2002-TN-1400.5, INETI, Lisboa, Mar. 2004. Parte do projecto AUTONAV.

[28] DINIS, J., AND MAMEDE, M. A sweepline algorithm for nearest neigbour queries. In *Proceedings of the 14th Canadian Conference on Computational Geometry* (Lethbridge, Canada, 2002), pp. 123–127.

[29] DINIS, J., AND MAMEDE, M. Sweeping the sphere. In *Proceedings of the 2010 International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'10)* (June 2010), IEEE, pp. 151–160.

[30] DINIS, J., AND MAMEDE, M. Updates on Voronoi diagrams. In *Proceedings of the 2011 International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'11)* (June 2011), IEEE, pp. 192–199.

[31] DUBÉ, T., AND YAP, C.-K. A basis for implementing exact geometric algorithms. (artigo retirado de `http://cs.nyu.edu/cs/faculty/yap`), Oct. 1993.

[32] EDELSBRUNNER, H., AND SEIDEL, R. Voronoi Diagrams and Arrangements. In *Proceedings of the First Annual Symposium on Computational Geometry (SCG'85)* (New York, NY, USA, 1985), ACM, pp. 251–262.

[33] EDELSBRUNNER, H., AND SEIDEL, R. Voronoi Diagrams and Arrangements. *Discrete & Computational Geometry 1* (1986), 25–44.

[34] FORTUNE, S. A sweepline algorithm for Voronoi diagrams. *Algorithmica 2* (1987), 153–174.

[35] GEODAS, Marine Trackline Geophysics Data. Available from the National Geophysical, Data Center, National Oceanic and Atmospheric Administration, U.S., Department of Commerces, `http://www.ngdc.noaa.gov`.

[36] GOLD, C. M., AND ANGEL, P. Voronoi hierarchies. In *GIScience* (2006), vol. 4197 of *Lecture Notes in Computer Science*, Springer, pp. 99–111.

[37] GOODRICH, M. T., AND TAMASSIA, R. *Data Structures and Algorithms in JAVA*, fourth ed. Jonh Wiley & Sons, Inc., Hoboken, NJ, 2006.

[38] GREEN, P. J., AND SIBSON, R. Computing Dirichlet tessellation in the plane. *The Computer Journal 21*, 2 (May 1978), 168–173.

[39] GUIBAS, L., AND STOLFI, J. Primitives for the manipulation of general subdivisions and the computation of Voronoi. *ACM Trans. Graph. 4*, 2 (Apr. 1985), 74–123.

[40] GUIBAS, L. J., KNUTH, D. E., AND SHARIR, M. Randomized incremental construction of delaunay and Voronoi diagrams. *Algorithmica 7*, 4 (1992), 381–413.

[41] GUIBAS, L. J., AND STOLFI, J. Ruler, compass, and computer: The design and analysis of geometric algorithms. Tech. Rep. SRC-RR-37, DEC/HP Labs, Systems Research Center, Palo Alto, CA, Feb. 1989.

[42] HELD, M. VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Computational Geometry 18*, 2 (2001), 95 – 123.

[43] IMAI, T. A topology oriented algorithm for the Voronoi diagram of polygons. In *Proceedings of the Eighth Canadian Conference on Computational Geometry* (Carleton University, Ottawa, Canada, Aug. 1996), F. Fiala, E. Kranakis, and J.-R. Sack, Eds., Carleton University Press, pp. 107–112.

[44] KARAVELAS, M. I. A robust and efficient implementation for the segment Voronoi diagram. In *Proceedings of the International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'04)* (Hongo, Tokyo, Japan, Sept. 2004), pp. 51–62.

[45] LINDEGREN, L. Gaia: Astrometric performance and current status of the project. *Proceedings of the International Astronomical Union 5* (Mar. 2009), 296–305.

[46] NA, H.-S., LEE, C.-N., AND CHEONG, O. Voronoi diagrams on the sphere. *Computational Geommetry Theory Applications 23*, 2 (2002), 183–194.

[47] OKABE, A., BOOTS, B., SUGIHARA, K., AND CHIU, S. N. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*, second ed. John Wiley, Chichester, 2000.

[48] O'ROURKE, J. *Computational geometry in C*, second ed. Cambridge University Press, New York, NY, USA, 1998.

[49] PREPARATA, F. P., AND HONG, S. J. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM 20*, 2 (1977), 87–93.

[50] PREPARATA, F. P., AND SHAMOS, M. I. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[51] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[52] RENKA, R. J. Algorithm 772: Stripack: Delaunay triangulation and Voronoi diagram on the surface of a sphere. *ACM Transactions on Mathematical Software 23*, 3 (Sept. 1997), 416–434.
http://doi.acm.org/10.1145/275323.275329.

[53] SEIDEL, R. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proceedings of the Eighteenth Annual ACM symposium on Theory of Computing (STOC'86)* (New York, NY, USA, 1986), ACM, pp. 404–413.

[54] SHAMOS, M. I., AND HOEY, D. Closest-point problems. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 1975), IEEE Computer Society, pp. 151–162.

[55] SHAMOS, M. I., AND HOEY, D. Geometric intersection problems. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, Oct. 1976), IEEE Computer Society, pp. 208 –215.

[56] SHARMAN, G., METZGER, D., CAMPAGNOLI, J., BUTLER, T., BERGGREN, T., DIVINS, D., AND STEELE, M. Geodas: A hydro/bathy data management system. *Surveying and Land Information Systems 58*, 3 (1998), 141–146.

[57] SHUSTER, M. D., AND OH, S. D. Three-axis attitude determination from vector observations. *Journal of Guidance and Control 4* (Feb. 1981), 70–77.

[58] SIBSON, R. A brief description of Natural Neighbour interpolation. In *Interpreting multivariate data*, V. Barnett, Ed., vol. 21. John Wiley & Sons, 1981, pp. 21–36.

[59] SNYDER, J. P. *Map Projections - A Working Manual*. U. S. Government Printing Office, 1987. U. S. Geological Survey Professional Paper 1935.

[60] SUGIHARA, K., AND IRI, M. Construction of the Voronoi diagram for "one million" generators in single-precision arithmetic. *Proceedings of the IEEE 80*, 9 (Sept. 1992), 1471–1484.

[61] SUGIHARA, K., IRI, M., INAGAKI, H., AND IMAI, T. Topology-Oriented Implementation–An Approach to Robust Geometric Algorithms. *Algorithmica 27* (2000), 5–20.

[62] TOUSSAINT, G. T. A simple linear algorithm for intersecting convex polygons. *The Visual Computer 1* (1985), 118–123.

[63] YAP, C.-K. Towards exact geometric computation. *Computational Geometry 7*, 1–2 (1997), 3–23.

[64] ZACHARIAS, N., FINCH, C. T., GIRARD, T. M., HENDEN, A., BARTLETT, J. L., MONET, D. G., AND ZACHARIAS, M. I. The Fourth US Naval Observatory CCD Astrograph Catalog (UCAC4). *The Astronomical Journal 145*, 2 (2013), 44.

[65] ZWILLINGER, D. *CRC Standard Mathematical Tables and Formulae*, 30 ed. CRC Press, 1996.

Apêndices

Apêndice A

Área de um polígono esférico

O algoritmo de redução de catálogos de estrelas, apresentado na secção 8.1, depende do cálculo da área de um polígono esférico, que é a região de Voronoi de uma estrela. Aqui é apresentada uma forma de calcular essa área a partir das coordenadas dos vértices do polígono esférico.

A área de um polígono esférico é dada por um corolário do teorema de Girard [65], que exprime a área de um triângulo esférico na esfera unitária:

$$A_{\text{triângulo}} = \alpha + \beta + \gamma - \pi \ , \tag{A.1}$$

em que α, β e γ são os ângulos internos do triângulo, em radianos. Um polígono convexo esférico com $n \geq 3$ lados é facilmente subdividido em triângulos, ligando um vértice arbitrário a todos os $n-3$ vértices não-adjacentes, gerando $n-2$ triângulos. Logo, a área do polígono é dada pela soma das áreas dos triângulos, que se traduz por:

$$A_{\text{polígono}} = \sum_{i=1}^{n} \alpha_i - (n-2)\,\pi \ , \tag{A.2}$$

sendo α_i os ângulos internos em cada vértice do polígono, em radianos.

Os ângulos internos α_i são calculados da seguinte forma. Sejam v_i, com $i = 0, \ldots, n-1$, os vértices do polígono esférico. O ângulo interno α_i no vértice i é dado pelo ângulo entre os planos definidos por (v_{i-1}, v_i) e (v_i, v_{i+1}) e pela origem. Ou seja:

$$\alpha_i = \arccos\left(\vec{u}_a \cdot \vec{u}_b\right) \ \text{ onde}$$
$$\begin{cases} \vec{a} = \vec{v}_{i-1} \times \vec{v}_i, \\ \vec{b} = \vec{v}_{i+1} \times \vec{v}_i, \\ \vec{u}_a = \vec{a}/\|\vec{a}\|, \\ \vec{u}_b = \vec{b}/\|\vec{b}\|, \end{cases} \tag{A.3}$$

com $i-1$ e $i+1$ calculados módulo n.

Apêndice B

Topologia da árvore vermelha-preta

Na secção 4.2.3, foi indicado que numa implementação da frente de onda por uma árvore binária de pesquisa vermelha-preta [19] os nós folha representam os arcos enquanto que os nós internos representam as intersecções de arcos. Aqui é demonstrada essa propriedade.

Uma árvore vermelha-preta é uma árvore binária em que a cada nó é atribuída uma cor (vermelho ou preto), para além dos habituais ponteiros para o nó filho esquerdo, nó filho direito e nó pai. Se um dos nós filhos ou nó pai não existe, o ponteiro aponta para um nó auxiliar, denominado por nó *nulo*. As árvores vermelha-preta satisfazem as seguintes propriedades:

P1) cada nó ou é vermelho ou é preto;

P2) a raiz da árvore é preta;

P3) os nós nulos são pretos;

P4) se um nó é vermelho, os nós filhos (se existirem) são pretos;

P5) todos os caminhos de um nó a um nó folha descendente contêm o mesmo número de nós pretos.

O que se quer demonstrar é que a implementação da frente de onda por uma árvore vermelha-preta também satisfaz as seguintes propriedades:

Q1) os nós folha representam um arco;

Q2) os nós internos têm sempre dois filhos e representam uma intersecção de arcos.

Assumindo que uma árvore binária satisfaz as propriedades Q (Q1 e Q2), observa-se que, considerando todas as configurações possíveis, um nó da frente de onda satisfaz as seguintes propriedades:

F1) se um nó folha é vermelho, então o nó irmão é necessariamente nó folha e igualmente vermelho;

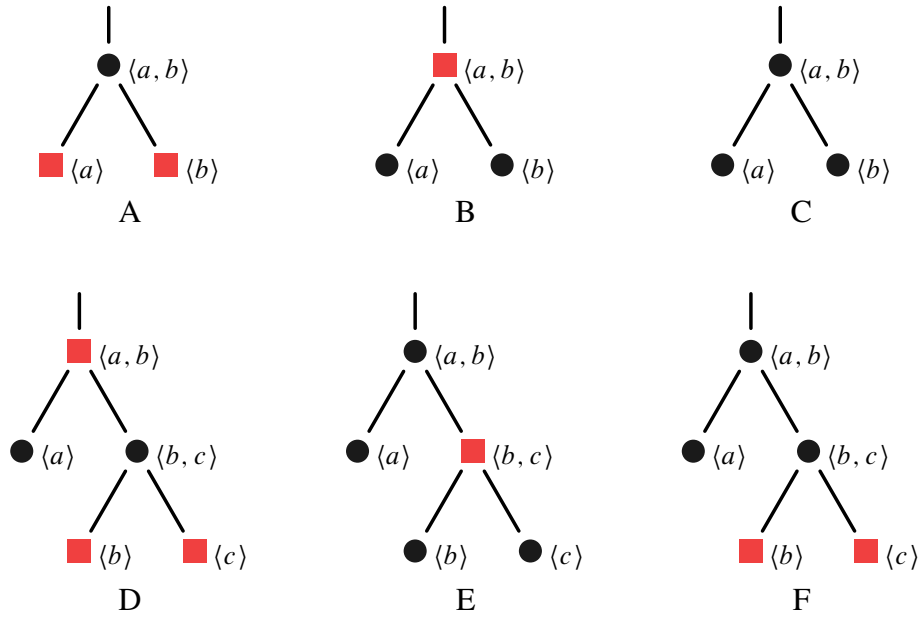


Figura B.1: Configurações possíveis da árvore vermelha-preta na representação da frente de onda (a menos de uma simetria). Os quadrados representam os nós vermelhos.

F2) se um nó folha é preto, então:

(a) ou o nó irmão é folha e preto;

(b) ou o nó irmão é preto e tem dois filhos vermelhos;

(c) ou o nó irmão é vermelho e tem dois filhos pretos.

Estas propriedades permitem, a menos de uma simetria, gerar um total de seis configurações distintas entre um nó folha, o nó pai e o nó irmão (e, eventualmente, nós sobrinhos), ilustradas na Figura B.1.

Qualquer um dos algoritmos de varrimento (linear, circular ou na esfera) começa por processar dois eventos-local. O primeiro evento-local inicializa a frente de onda com um arco. O segundo evento-local aumenta a frente de onda para um total de três arcos e duas intersecções de arco, implementada por uma árvore binária vermelha-preta com cinco nós. Dadas as propriedades das árvores vermelhas-pretas, uma árvore com cinco nós só pode assumir uma de duas configurações, que correspondem aos casos E e F da Figura B.1 (a menos de uma simetria e assumindo que o nó $\langle a,b\rangle$ é a raiz), e verificam as propriedades Q. Obviamente, o processamento dos restantes eventos modifica a frente de onda, adicionando e removendo nós. No entanto, observa-se que, qualquer que seja o tipo de evento, as actualizações na árvore vermelha-preta são feitas aos pares: ou são adicionados dois nós ou são removidos dois nós.

Num evento-local, um arco é substituído por três arcos e duas intersecções de arcos. Na prática, esta operação equivale a substituir, duas vezes, um arco $\langle a\rangle$ pela sequência $\{\langle a\rangle, \langle a,b\rangle, \langle b\rangle\}$. Esta operação é feita redefinindo o nó $\langle a\rangle$ pela intersecção $\langle a,b\rangle$, inserindo

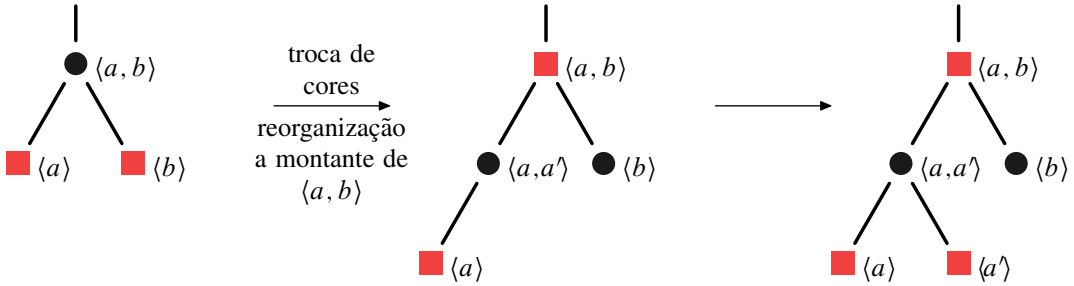


Figura B.2: Substituição do arco $\langle a \rangle$ pela sequência $\{\langle a \rangle, \langle a,a' \rangle, \langle a' \rangle\}$ na configuração A.

depois os nós $\langle a \rangle$ e $\langle b \rangle$ como filho esquerdo e filho direito, respectivamente. Esta operação resume-se a inserir um par de nós filhos num nó folha.

Num evento-círculo, é eliminado um arco e uma das intersecções adjacentes (e redefinida a outra intersecção adjacente). Embora seja indiferente qual das intersecções adjacentes é removida, a operação é simplificada se escolher a intersecção representada pelo nó pai do arco a remover. Desta forma, esta operação resume-se a remover um par nó folha e nó pai respectivo.

Num evento-rotação, um arco e uma intersecção de arcos é removido de um extremo da frente de onda e depois adicionado no extremo oposto. Por exemplo, é removida a sequência $\{\langle a \rangle, \langle a,b \rangle\}$, do lado do mínimo da árvore, e substitui-se o arco $\langle a \rangle$, do lado do máximo da árvore, pela sequência $\{\langle a \rangle, \langle a,b \rangle, \langle b \rangle\}$. Esta operação equivale a remover dois, tal como num evento-círculo, seguida de uma redefinição de um nó e adição de dois nós filhos, tal como num evento-local.

Em resumo, qualquer um dos eventos opera sobre a árvore binária com uma combinação de duas operações: remoção de um par de nós ou inserção de um par de nós. Prova-se a seguir que qualquer uma destas operações não invalidam as propriedades Q, quando aplicadas a uma árvore vermelha-preta com cinco ou mais nós (e que verifica as propriedades Q, como atrás indicado).

Vamos ver primeiro a operação e adição de um par de nós que implementa a substituição de arco $\langle a \rangle$ pela sequência $\{\langle a \rangle, \langle a,b \rangle, \langle c \rangle\}$. Seja o nó do arco $\langle a \rangle$ denominado por nó âncora. Aplicando esta operação aos nós âncora da Figura B.1, e por aplicação das regras de organização das árvores vermelha-preta (c.f. [19]), observa-se o seguinte. A redefinição do nó âncora por $\langle a,b \rangle$ não modifica a árvore topologicamente. Depois são-lhe adicionados dois nós filhos, inicialmente com cor vermelha. Se o nó âncora é preto, então a operação termina aqui; a adição de nós vermelhos não obriga a nenhuma operação de reorganização da árvore. Se o nó âncora é vermelho (apenas na configuração A, ver Figura B.2), então a adição do primeiro nó filho obriga a uma troca de cores: o nó âncora e irmão passam a preto e o pai (de ambos) passa a vermelho. Eventualmente, a passagem do nó pai a vermelho pode despoletar um processo de reorganização acima de $\langle a,b \rangle$, que não altera a sub-árvore abaixo deste. A adição do segundo nó filho não causa mais nenhuma alteração, uma vez que o nó âncora é agora preto.

Vamos agora analisar a operação de remoção de um par de nós. Considere-se a remo-

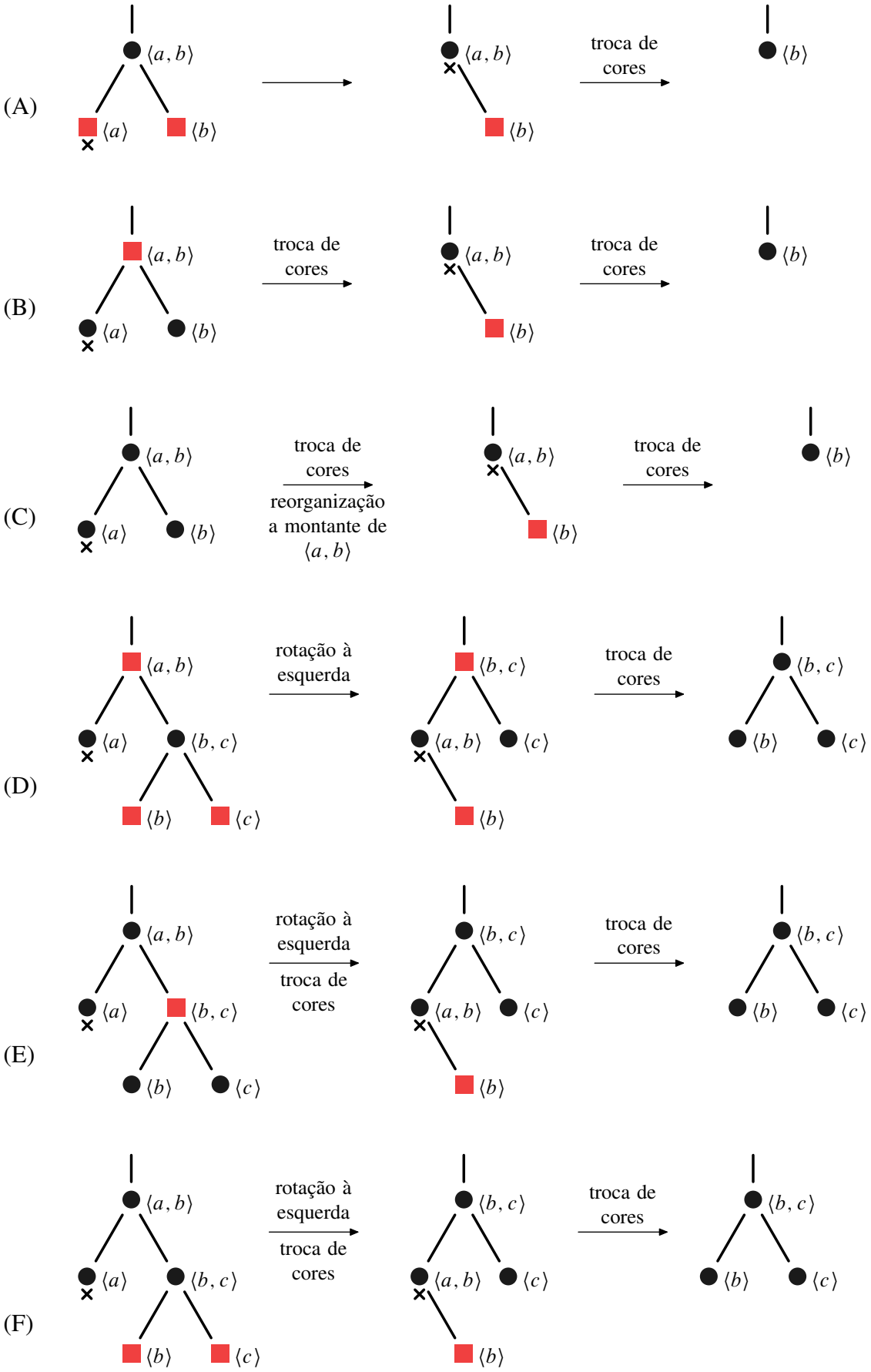


Figura B.3: Remoção da sequência $\{\langle a\rangle, \langle a,b\rangle\}$ nas várias configurações.

ção dos nós $\langle a \rangle$ e $\langle a,b \rangle$, tal como indicado na Figura B.3. Na maioria das configurações (A,B,D,E), a presença de nós vermelhos é suficiente para garantir que a remoção do par de nós se faz, no máximo, com uma rotação. Nos restantes casos (C,F), a remoção do par de nós implica a reorganização da árvore acima do subconjunto ilustrado (ver Figura B.3). Analisemos cada caso em detalhe. Por aplicação das regras de organização das árvores vermelha-preta (c.f. [19]), observa-se que:

(A) o nó $\langle a \rangle$ é removido, o nó $\langle a,b \rangle$ (agora só com um filho) é removido, o nó $\langle b \rangle$ muda de cor para preto;

(B) o nó $\langle a \rangle$ é removido, os nós $\langle a,b \rangle$ e $\langle b \rangle$ trocam de cor, o nó $\langle a,b \rangle$ é removido, o nó $\langle b \rangle$ muda de novo para preto;

(C) o nó $\langle a \rangle$ é removido, o nó $\langle b \rangle$ troca de cor (para vermelho) criando um défice de nós pretos, que é resolvido por uma reorganização da árvore acima do nó $\langle a,b \rangle$; por fim, o nó $\langle a,b \rangle$ é removido e o nó $\langle b \rangle$ muda de cor para preto;

(D) o nó $\langle a \rangle$ é removido, os nós $\langle a,b \rangle$ e $\langle b,c \rangle$ trocam de cor e o nó $\langle c \rangle$ passa a preto, é feita uma rotação à esquerda e, por fim, é removido o nó $\langle a,b \rangle$ (que só tem um filho) e o nó $\langle b \rangle$ passa a preto;

(E) o nó $\langle a \rangle$ é removido, os nós $\langle b \rangle$ e $\langle b,c \rangle$ trocam de cor e é feita uma rotação à esquerda; por fim, o nó $\langle a,b \rangle$ é removido e o nó $\langle b \rangle$ passa a preto;

(F) o nó $\langle a \rangle$ é removido, o nó $\langle c \rangle$ passa a preto e é feita uma rotação à esquerda; o nó $\langle a,b \rangle$ é removido e o nó $\langle b \rangle$ passa a preto.

Em qualquer das configurações, a primeira operação de edição (inserção de $\langle a \rangle$ ou remoção de $\langle a \rangle$) apenas invalida as propriedades Q temporariamente, sendo estas repostas logo após a segunda operação de edição (inserção de $\langle a' \rangle$ ou remoção de $\langle a,b \rangle$).