Representação e manipulação exatas de mapas esféricos

Marcus Vinícius Alvim Andrade

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Representação e manipulação exatas de mapas esféricos

Marcus Vinícius Alvim Andrade[1]

Fevereiro de 1999

Banca Examinadora:

- Jorge Stolfi (Orientador)
- Marcelo Gattass
  Departamento de Ciência da Computação - PUC-Rio
- Paulo Feofiloff
  Instituto de Matemática e Estatística - USP
- Pedro Jussieu de Rezende
  Instituto de Computação - UNICAMP
- Cid Carvalho de Souza
  Instituto de Computação - UNICAMP
- Carlos Eduardo Ferreira (Suplente)
  Instituto de Matemática e Estatística - USP
- Ricardo Dahab (Suplente)
  Instituto de Computação - UNICAMP

[1]**Financiado em parte pelo programa PICDT/CAPES**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Andrade, Marcus Vinícius Alvim

An24r Representação e manipulação exatas de mapas esféricos / Marcus Vinícius Alvim Andrade – Campinas, [S.P. :s.n.] 1999.

Orientador: Jorge Stolfi

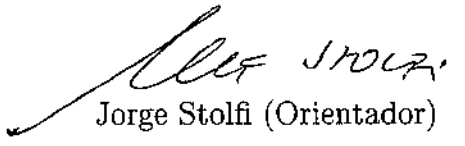
Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Algoritmos. 2. Mapas. 3. Geometria projetiva. 4. Geometria - Processamento de dados. I. Stolfi, Jorge. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Representação e manipulação exatas de mapas esféricos

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Marcus Vinícius Alvim Andrade e aprovada pela Banca Examinadora.

Campinas, 30 de Março de 1999.



Jorge Stolfi (Orientador)

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

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 15 de março de 1999, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. Marcelo Gattass
PUC-Rio

Prof. Dr. Paulo Feofiloff
IME - USP

Prof. Dr. Pedro Jussieu de Rezende
IC - UNICAMP

Prof. Dr. Cid Carvalho de Souza
IC - UNICAMP

Prof. Dr. Jorge Stolfi
IC - UNICAMP

Abstract

In this dissertation we develop a tool kit for the exact representation and manipulation of *spherical maps* (maps on the sphere composed by arcs of circles, not necessarily geodesic ones), suitable for the implementation of geographical information systems (GIS).

Firstly, we describe the data structure SMC (*spherical maps by corners*) which we developed to represent the topology of spherical maps. It allows the representation of fairly general maps, including maps that have oval edges (not incident to any vertex), isolated vertices (not incident to any edge), and faces with multiple borders. We also define a set of topological operators to build, traverse and modify this structure.

Secondly, we define the *rational circles*, a dense subset of circles on the sphere $\mathbb{S}^2$ which can be exactly represented. Based on this concept, we develop an exact representation of points, circles and circular arcs on the sphere, such that the set of exactly representable points (*sub-rational points*) includes all points of intersection of two rational circles. We also develop exact algorithms for basic geometric operations with rational circles, including intersection, relative position, circular ordering of points on rational circles, and circular ordering of circles around a point.

Thirdly, we use these tools to develop exact algorithms for point location on spherical maps, and to compute the overlay of two spherical maps. We note that most geometric operations used in GIS (region intersection, point location, feature extraction, clipping etc) can be reduced to map overlay.

Finally, we show that the proposed tool kit is economical and efficient enough to be used as the basis for the implementation of a GIS.

All algorithms described in this text were implemented (in Modula-3) as a general-purpose library for the exact representation and manipulation of spherical maps.

Resumo

Neste trabalho desenvolvemos um conjunto de ferramentas para a representação e manipulação exatas de *mapas esféricos* (mapas sobre a esfera formados por arcos de círculos, não necessariamente máximos) adequados para a implementação de sistemas de informações geográficas (SIGs).

Na primeira parte deste trabalho, descrevemos a estrutura de dados SMC (*Spherical Maps by Corners*) para representar a topologia de mapas esféricos. Esta estrutura permite a representação de mapas bastante gerais, incluindo arestas ovais (não incidentes a nenhum vértice), vértices isolados (não incidentes a nenhuma aresta), e faces com múltiplas bordas. Definimos também um conjunto de operadores topológicos para construir, percorrer e modificar esta estrutura.

Na segunda parte, definimos os *circulos racionais*, um subconjunto denso dos círculos sobre a esfera $\mathbb{S}^2$ que podem ser representados de maneira exata. Baseados neste conceito, definimos representações exatas para pontos e arcos de círculos na esfera, sendo que o conjunto dos pontos representáveis exatamente (os pontos *sub-racionais*) inclui todos os pontos de interseção entre círculos racionais. Desenvolvemos também algoritmos exatos para as operações geométricas básicas sobre círculos racionais, incluindo interseção, posição relativa, ordenação de pontos sobre um círculo e ordenação de círculos em torno de um ponto.

Na terceira parte, baseados nos resultados descritos nas duas partes anteriores, desenvolvemos algoritmos exatos para localização de pontos num mapa esférico, e para sobreposição de dois mapas esféricos. Observamos também que boa parte das operações geométricas usadas em SIGs (por exemplo, interseção de regiões, localização de pontos, extração de detalhes, recortes) podem ser reduzidas a problemas de sobreposição.

Finalmente, mostramos que este conjunto de ferramentas é econômico e eficiente o bastante para servir de base para a implementação de SIGs.

Todos os algoritmos apresentados neste trabalho foram implementados (em Modula-3) na forma de uma biblioteca genérica para a representação e manipulação exatas de mapas esféricos.

Agradecimentos

Em primeiro lugar, gostaria de agradecer imensamente ao meu orientador Jorge Stolfi pelo constante estímulo, e principalmente pela dedicação a este trabalho (especialmente na forma de inúmeras leituras e correções), sem os quais seria impossível a sua conclusão.

A toda a minha família (meu saudoso pai, minha mãe, irmãos, avós, sogros, $\cdots$), um agradecimento muito especial pela interminável paciência e compreensão pelas minhas freqüentes ausências a vários eventos familiares importantes.

A todos os meus colegas de curso, em especial, aos meus amigos Mário, Fábio, Juliano e Nalvo, um muito obrigado pelos diversos momentos agradáveis desfrutados ao longo de todos estes anos. Agradeço também às amigas Anamaria e Helena, que além de colegas de curso são também orientandas do Prof. Stolfi, pelas freqüentes palavras de incentivo e pelas trocas de experiências que foram tão importantes para nos manter incentivados pelo trabalho.

Aos grandes amigos Pedro Henrique, Paulo, Sérgio e Erik, agradeço pela amizade e pela imensa hospitalidade nas minhas diversas vindas a Campinas. Aos amigos viçosenses, um eterno muito obrigado por terem sido por inúmeras vezes a inestimável "válvula de escape".

Aos colegas (professores e funcionários) do Departamento de Informática da UFV, o meu muito obrigado pela constante ajuda em diversos sentidos, e também por que não, por sempre tentarem me manter envolvido com o trabalho através da pergunta: "quando voce vai terminar a sua tese?".

A todos os professores e funcionários do Instituto de Computação da UNICAMP, agradeço pelo convívio amistoso e saudável, o que tornou ainda mais proveitoso estes anos de estudo e trabalho.

Por último, mas mais importante, gostaria de agradecer à minha amada esposa Flávia pela sua dedicação, paciência e abdicação, pois sem sombra de dúvidas ela é a principal responsável pela conclusão deste trabalho; por isso, a ela o meu sincero muito obrigado.

Dedicada à minha esposa Flávia

Índice

Lista de Figuras

Capítulo 1

Introdução

Nos últimos anos, o desenvolvimento de *Sistemas de Informações Geográficas* (SIGs) tem atraído a atenção de pesquisadores de diversas áreas tais como geometria computacional, computação gráfica, banco de dados, engenharia de software, dentre outras. De um modo geral, um SIG é um sistema automatizado utilizado para armazenar, analisar e manipular dados geográficos, ou seja, dados que representam objetos e fenômenos em que a localização geográfica é uma característica inerente à informação e indispensável para analisá-la. Tais sistemas comportam diferentes tipos de dados e aplicações, como por exemplo, otimização de tráfego, controle cadastral, gerenciamento de serviços de utilidade pública, demografia, cartografia, administração de recursos naturais, monitoramento costeiro, etc [6, 5, 25]. Nosso objetivo neste trabalho é abordar os aspectos de geometria computacional envolvidos na representação e manipulação de mapas em SIGs.

Uma importante característica dos SIGs é que eles possibilitam a integração, numa única base de dados, de informações geográficas provenientes de fontes diversas tais como dados cartográficos, dados de censo e cadastro urbano e rural, imagens de satélites e modelos numéricos de terreno. Em outras palavras, os SIGs são utilizados para representar tanto informações globais (referentes ao planeta) como informações locais (referentes a uma região limitada, como um estado, uma cidade, etc). Em geral, devido à complexidade de se trabalhar com a forma real da Terra, mapas globais são modelados sobre uma esfera, e mapas locais sobre um plano. Entretanto, visto que é necessário integrar estes dois tipos de dados, é fundamental que eles estejam representados de maneira consistente. Portanto, é conveniente representar internamente todos os dados geográficos, mesmo que em escalas pequenas, como mapas na superfície de uma esfera; não só por esta ser a representação mais natural, mas também porque assim evita-se a necessidade da realização de transformações na integração de diferentes mapas locais, casos estes fossem expressos em relação a diferentes planos de projeção.

Desta forma, vamos considerar neste trabalho apenas mapas sobre a esfera $\mathbb{S}^2$. Além

disso, vamos também nos limitar a mapas cujas linhas e fronteiras são compostas por arcos de círculos (não necessariamente máximos), os quais denominamos *mapas esféricos*. Por exemplo, veja a figura 1.1.

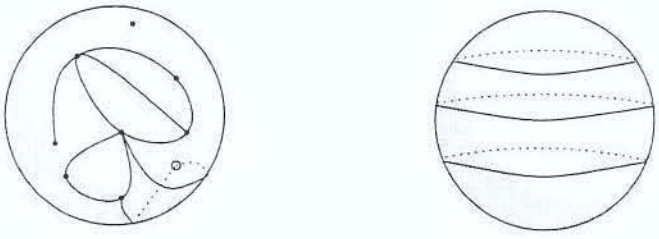


Figura 1.1: Exemplos de mapas esféricos

Vale notar que os mapas genéricos utilizados em SIGs usam uma grande variedade de modelos (não necessariamente esféricos) para a representar a superfície da Terra; mas, desde que tais mapas definam uma latitude e uma longitude nominais para cada ponto, eles podem ser internamente interpretados de maneira automática como mapas esféricos, sendo que esta representação interna não precisa ser visível para o usário.

Os mapas esféricos surgem naturalmente quando representamos grades de latitude-longitude, imagens de satélites, áreas de alcance e influência, etc (vide [25]). Em particular, a projeção estereográfica de mapas planares com arestas retilíneas ou circulares resulta sempre num mapa esférico e vice-versa.

Obviamente, existem mapas cujas linhas não são arcos de círculo; por exemplo, curvas de nível, isotermas, isobáricas, rios e fronteiras políticas. Outro exemplo são as curvas *loxodrômicas*, que descrevem a trajetória de um navegador que navega sempre numa mesma direção fixa da bússula. No entanto, curvas arbitrárias podem ser adequadamente aproximadas por uma sequência de arcos de círculos, com erro de aproximação de segunda ordem.

1.1 Sobreposição de mapas

Uma operação particularmente importante neste contexto é a operação de sobreposição de dois mapas esféricos A e B, denotada por $A \oplus B$. Informalmente, o mapa $A \oplus B$ é a partição da esfera de acordo com os dois mapas combinados. Vale notar que cada elemento (vértice, aresta ou face) e de $A \oplus B$ está contido num único elemento $\alpha(e)$ de A e num único elemento $\beta(e)$ de B. Por exemplo, veja a figura 1.2.

A importância deste conceito decorre do fato que muitas das operações envolvendo mapas podem ser reduzidas a operações de sobreposição [10]. Por exemplo, um polígono pode ser representado por um mapa onde cada elemento é rotulado como "interior" ou

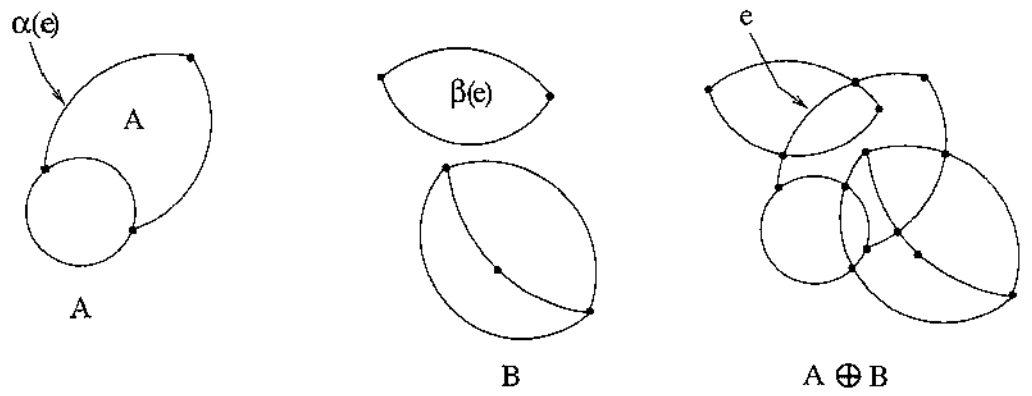


Figura 1.2: A sobreposição de dois mapas

"exterior". Neste caso, a interseção de dois polígonos A e B pode ser facilmente determinada da seguinte forma: primeiro construímos a sobreposição $C = A \oplus B$, sendo que todo o elemento e de C é rotulado de "interior" se e somente se $\alpha(e)$ e $\beta(e)$ são também rotulados "interior" nos seus respectivos mapas. Os demais elementos de C são rotulados "exterior". Feito isso, basta eliminar do mapa C todas as arestas que separam duas faces com o mesmo rótulo, e todos os vértices que não são pontas de nenhuma aresta. Veja a figura 1.3.

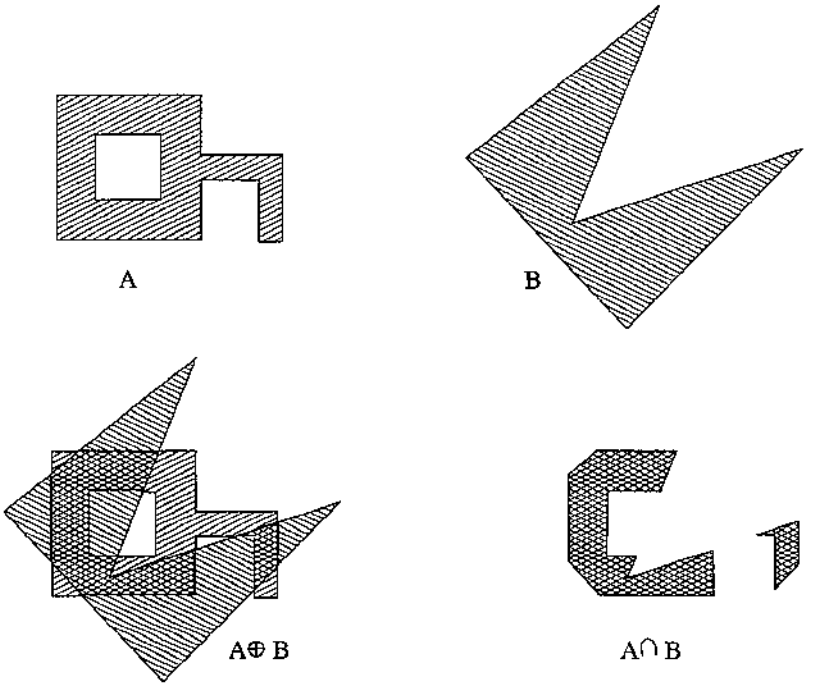


Figura 1.3: Interseção de polígonos: hachurado ("interior"); branco ("exterior").

Note que esta mesma estratégia pode ser usada para obter a união ou a subtração de dois polígonos: a etapa mais complicada — calcular o mapa $C = A \oplus B$ — é exatamente a mesma, muda apenas o critério de rotulação dos elementos de C.

Um outro exemplo que ilustra a versatilidade da operação de sobreposição é o seguinte: suponha que temos um mapa geográfico A no qual as cidades são representadas por vértices, e queremos determinar todas as cidades cuja distância de um ponto p é menor do que r. Para isto, construímos primeiro um mapa B que consiste de um disco D de centro em p e raio r (medido sobre a esfera), e computamos a sobreposição $C = A \oplus B$. Daí, as cidades desejadas são os vértices v do mapa C tais que $\beta(v)$ é o disco D.

1.2 Erros de arredondamento

Tradicionalmente, os cálculos geométricos em SIGs (assim como em diversas outras aplicações numéricas) são efetuados com uma precisão fixa — em geral, com números em ponto flutuante. Portanto, estes cálculos estão sujeitos a imprecisões numéricas provenientes de várias fontes.

Em primeiro lugar, os dados brutos são freqüentemente representados na base decimal, enquanto que nas máquinas eles são codificados em binário. Geralmente, a conversão de decimal para binário (e vice-versa) implica em erros. Por exemplo, o número decimal 0.6 equivale, em binário, à dízima periódica $0.10011001\cdots$; por outro lado, o número 2^{-50} tem 35 algarismos decimais significativos.

Além disso, o resultado exato de uma operação aritmética envolvendo dois números a e b em geral requer mais bits do que a e b (às vezes um número infinito). Por isso a representação do resultado da operação é freqüentemente arredondada. Por exemplo, com uma mantissa de 5 dígitos, o produto $0.24665 \cdot 0.63994 = 0.1578412010$ não pode ser representado exatamente e tem que ser arredondado para 0.15784, o que implica num erro de aproximadamente $1.2 \,.\, 10^{-6}$.

Finalmente, por razões de eficiência, diversas funções, como por exemplo, log, sin, cos, $\cdots$, em geral, são determinadas de maneira aproximada utilizando-se séries finitas ou aproximações racionais.

1.2.1 Consequências dos erros de arredondamento

Infelizmente, uma das áreas mais sensíveis aos problemas de erros de arredondamento é a área de geometria computacional [39, 38, 1, 21, 5]. A experiência mostra que se os erros de arredondamento não são levados em consideração e tratados de maneira adequada, a implementação de algoritmos geométricos fica freqüentemente sujeita a falhas, em geral catastróficas. Isto porque na maioria das vezes, os problemas nesta área envolvem tanto

um conjunto de dados numéricos, que descrevem a forma (a geometria) dos objetos, quanto um conjunto de dados simbólicos que descrevem o relacionamento entre os elementos que compõem o objeto (isto é, sua topologia). Nestes casos, os erros de arredondamento freqüentemente tornam inconsistentes estas duas representações de um mesmo objeto. Por exemplo, a representação topológica de uma figura pode requerer que três retas se interceptem num mesmo ponto. No entanto, se os coeficientes destas três retas são representados por números em ponto flutuante, então a computação da interseção entre cada par de retas pode resultar em três pontos distintos, ao invés de um único ponto.

Estas inconsistências têm efeitos bastante drásticos em algoritmos geométricos porque a maioria destes algoritmos se baseiam em testes de predicados topológicos e, dependendo do resultado, executam uma ou outra seqüência de operações. Portanto, em algoritmos envolvendo longas seqüências de testes desse tipo, os erros de arredondamento podem levar a um resultado final totalmente inconsistente e imprevisível; e a probabilidade de que isto ocorra pode ser inaceitavelmente alta, mesmo que a probabilidade de falha em cada teste seja pequena.

Vale notar que a causa essencial desta dificuldade é que geralmente os algoritmos são elaborados considerando-se um modelo numérico contínuo. Quando, estes algoritmos (corretos do ponto de vista teórico) são implementados utilizando números em precisão fixa, muitas das propriedades essenciais para sua corretude deixam de ser válidas.

1.2.2 Soluções propostas

Uma das propostas mais simples para tentar amenizar estes problemas é a adocão de uma *tolerância* ϵ, de tal forma que dois valores cuja diferença é menor do que ϵ são considerados iguais. Por exemplo, se a distância entre um ponto p e uma reta r é menor do que ϵ, então considera-se que p é incidente em r. Como descrito por Hoffmann [20, sec. 4.2], existem diversas situações (freqüentes na prática) onde esta abordagem ainda nos leva a inconsistências. Na verdade, o que este método consegue é, no máximo, adiar o problema, diminuindo a probabilidade de sua ocorrência.

Uma outra proposta é tentar *ajustar a geometria* para que ela fique consistente com a topologia. Por exemplo, dadas três retas r_1, r_2 e r_3 que devem passar por um mesmo ponto, podemos calcular (em geral, de maneira aproximada) o ponto de interseção p entre r_1 e r_2, e depois ajustar r_3 de modo que ela efetivamente passe por p. Desta forma, a geometria fica consistente com a topologia (as retas passam por um mesmo ponto). O problema com este método é que pode ser bastante complicado determinar o ajuste adequado; ou, até mesmo, pode ser que não exista um ajuste adequado. Por exemplo, sejam os pontos $p = r_1 \cap r_2$ e $q = r_1 \cap r_3$; os erros de arredondamento podem fazer com que p seja incidente a r_3, mas q não seja incidente a r_2. Desta forma, se ajustarmos a

reta r_2 de modo que ela passe pelo ponto q, então pode ser que o "novo" ponto $r_1 \cap r_2$ não seja mais incidente a r_3.

Uma proposta alternativa é tentar *ajustar a topologia* para que ela fique consistente com a geometria. Ou seja, a idéia neste caso é assumir os erros de arredondamento e alterar a representação topológica para que ela reflita tais erros. Por exemplo, dados três pontos colineares p_1, p_2 e p_3, seja r a reta passando por p_1 e p_2; se (por causa dos erros de arredondamento) esta reta não contém o ponto p_3, então, ajustamos a topologia de modo a dizer que p_3 não é colinear com p_1 e p_2. O problema com esta abordagem é que podem ocorrer situações em que não há nenhuma topologia consistente com a geometria. Por exemplo, no caso das três retas r_1, r_2 e r_3 visto acima, não há como ajustar a topologia para satisfazer a "geometria" resultante dos erros de arredondamento (pois $r_1 \cap r_2$ está em r_3, mas $r_2 \cap r_3$ não está em r_1).

Uma proposta que objetiva contornar o problema descrito no parágrafo anterior consiste em tentar reestruturar o algoritmo de modo a *eliminar predicados redundantes*. Desta forma, os testes geométricos efetuados sempre seriam consistentes com algum modelo topológico. Por exemplo, considerando o caso das três retas descrito acima, se um teste geométrico indica que o ponto $r_1 \cap r_2$ é incidente à reta r_3, então o algoritmo deve supor, por definição, que a interseção entre r_1 e r_3 está em r_2 (sem que este fato nunca seja explicitamente verificado). No entanto, a grande dificuldade desta abordagem é a determinação de todas as relações de dependência entre os predicados, pois em geral, esta tarefa é extremamente complexa, senão insolúvel.

Adicionalmente, há trabalhos que propõem *modificar a topologia e a geometria* simultaneamente, de modo a torná-las consistentes (incorporando os erros de arredondamento). Por exemplo, suponha novamente o caso de três retas r_1, r_2 e r_3 que deveriam ser concorrentes. Se, devido a erros de arredondamento, o ponto $p = r_1 \cap r_2$ não é incidente na reta r_3, então esta reta é "quebrada" e substituída por duas semi-retas r_3' e r_3'', ambas com origem no ponto p. Na literatura este método é conhecido com o nome de *arredondamento geométrico*, e tem sido aplicado com relativo sucesso no caso de arranjos de segmentos de retas no plano [15, 27]. Entretanto, sua extensão a objetos mais complexos ainda é um problema em aberto. Além disso, resta o problema de que muitas aplicações não toleram quaisquer pertubações topológicas, o que invabiliza a aplicação deste método.

Finalmente, há também o método de *análise de erros* que consiste em mostrar (utilizando métodos de análise numérica e álgebra linear) que a solução obtida através de uma determinada implementação de um algoritmo é "adequadamente" próxima da solução teoricamente correta. Uma dificuldade imediata desta abordagem é que os resultados (topológicos) de um algoritmo geométrico dependem de maneira descontínua dos dados de entrada. Para contornar esta dificuldade Guibas, Salesin e Stolfi [17] desenvolveram a *geometria epsilon*, que utilizando o conceito de *análise inversa de erros* [37] determina

a pertubação máxima que deve ser aplicada aos dados de entrada para que resultado obtido seja a solução exata para aquela entrada pertubada. O principal problema desta abordagem é que a análise necessária fica intratável, mesmo para problemas relativamente simples, e leva a algoritmos muito ineficientes.

Concluindo, apesar de todos os esforços investidos nos últimos 20 anos, na prática, todos os métodos descritos acima, que visam tornar robustos os algoritmos geométricos implementados com aritmética de precisão fixa, ainda estão restritos a algumas poucas aplicações específicas.

1.2.3 Computação exata

Diante destes fatos, atualmente o sentimento de grande parte da comunidade que tem trabalhado nesta área [5, 21, 39] é de que, exceto em situações muito específicas, é impraticável desenvolver algoritmos geométricos robustos dentro do paradigma de precisão fixa. Para alcançar a robustez de forma consistente, é fundamental trabalhar no *paradigma da computação exata*, que consiste em representar de maneira exata todos os valores numéricos e determinar, também de maneira exata, todos os predicados geométricos.

Vale notar que, em geral, este paradigma exige a utilização de uma representação numérica com precisão arbitrária, ou pelo menos, com uma precisão maior do que a fornecida pelas instruções da máquina. Certamente, esta é a principal desvantagem do paradigma de computação exata, pois este fato aumenta consideravelmente o tempo de processamento dos algoritmos em relação à implementação com precisão fixa. No entanto, é importante observar que não faz muito sentido comparar estes dois custos, pois no caso da precisão fixa, troca-se correção por velocidade; ou seja, para se ganhar tempo não se computa a resposta correta do problema e sim uma resposta aproximada. Em muitas aplicações, o custo de uma solução incorreta, mesmo que ocasional, é muitas vezes maior do que o custo adicional da aritmética exata.

Além disso, nos últimos anos, a velocidade das máquinas tem aumentado de tal forma que o tempo de processamento (eficiência do código) é cada vez menos importante do que o tempo de desenvolvimento dos programas. Portanto, visto que há uma certa sobre-oferta de ciclos de máquina, é em geral razoável investir parte destes ciclos para se ter a certeza de que a solução obtida para o problema é sempre uma solução correta.

Concluindo, existe hoje em dia um crescente interesse por ferramentas e algoritmos baseados no paradigma de computação exata. Este interesse se justifica principalmente porque à medida que novas ferramentas (exatas) forem colocadas à disposição dos usuários, estes poderão escolher a solução que melhor os satisfazem, optando entre rapidez e confiabilidade.

1.3 Objetivos do trabalho

Diante destas observações, vamos estabelecer como nosso objetivo principal o desenvolvimento de uma representação para mapas esféricos, e um conjunto de operadores básicos suficientes para computar a sobreposição destes mapas. Além disso, considerando todos os inconvenientes do modelo de precisão fixa, optamos por desenvolver esta representação e operações dentro do paradigma de computação exata. Resumindo, nosso objetivo é desenvolver uma representação exata de mapas esféricos, e um algoritmo também exato para a sobreposição destes mapas.

É importante frisar que, do ponto de vista de SIGs, há erros que são intrínsecos do processo de coleta de dados devido a imprecisões dos instrumentos, aproximação da superfície terrestre por uma esfera perfeita, aproximação de curvas gerais por arcos de círculos, projeções cartográficas, etc. Portanto, numa primeira etapa, vamos supor a realização de um processo de arredondamento para adequar os dados de entrada transformando-os em mapas esféricos representados exatamente de acordo com o nosso modelo. Uma vez efetuada esta transformação, os algoritmos descritos neste trabalho permitem manipular estes mapas de maneira exata.

Vale ressaltar que vamos abordar estes problemas tanto do ponto de vista teórico quanto do ponto de vista prático; ou seja, além de descrever matematicamente a representação e os algoritmos exatos, nosso objetivo inclui também a implementação destes algoritmos, na forma de uma biblioteca de rotinas genéricas e robustas para a representação e manipulação de mapas na esfera.

1.4 Estrutura da dissertação

Esta dissertação pode ser dividida em três partes: na primeira (capítulos 2 a 4), tratamos das questões envolvendo a topologia dos mapas esféricos; na segunda (capitulos 5 a 9), abordarmos os problemas relacionados à geometria destes mapas; e finalmente, na terceira (capítulos 10 e 11), tratamos dos problemas envolvendo um mapa como um todo (topologia e geometria).

Mais especificamente, no capítulo 2, definimos o conceito topológico de mapas bidimensionais, dos quais os mapas esféricos são um caso particular; no capítulo 3, definimos uma formulação combinatória para a topologia dos mapas esféricos; e no capítulo 4, definimos uma estrutura de dados para representar estes mapas e um conjunto de operadores topológicos que nos permitem manipular esta estrutura.

No que diz respeito à geometria dos mapas esféricos, apresentamos inicialmente, no capítulo 5, um breve resumo de alguns conceitos de geometria projetiva orientada (no espaço tridimensional). No capítulo 6, descrevemos uma representação analítica da geo-

metria dos mapas esféricos e as operações básicas necessárias para a manipulação da mesma. Nos capítulos 7 e 8 descrevemos uma representação exata para esta geometria, e no capítulo 9, descrevemos os algoritmos exatos para as operações geométricas definidas no capítulo 6.

Baseados nas ferramentas topológicas e geométricas definidas nos capítulos anteriores, apresentamos, no capítulo 10, um algoritmo exato para localizar um ponto num mapa, e no capítulo 11, um algoritmo também exato de sobreposição de dois mapas.

Finalmente, no capítulo 12, apresentamos algumas observações sobre aplicações práticas destas técnicas e também algumas linhas de trabalhos (futuros) que merecem uma investigação mais detalhada. Além disso, no apêndice A, apresentamos um breve resumo de alguns conceitos topológicos utilizados neste trabalho e no apêndice B, descrevemos alguns detalhes da implementação da representação e dos algoritmos que foram descritos neste trabalho.

Para manter a compatibilidade entre este texto e os artigos (escritos em inglês) e os programas dele derivados, as funções e os procedimentos descritos neste texto serão denominados utilizando palavras da língua inglesa.

Capítulo 2

Topologia dos mapas bidimensionais

Conforme vimos no capítulo anterior, nosso objetivo neste trabalho é estabelecer uma representação de mapas que permita que a geometria destes mapas seja representada exatamente (sem erros de arredondamento) e que seja fechada para a operação de sobreposição.

Para alcançarmos tal objetivo, nosso primeiro passo será definir o conceito de mapas que pretendemos considerar, e descrever um método (uma estrutura de dados) que permita a representação destes mapas. Mais especificamente, neste capítulo vamos tratar das questões relacionadas à representação da topologia dos mapas, isto é, das relações de incidência entre os seus componentes (vértices, arestas e regiões).

Vale ressaltar que as questões relacionadas à representação da geometria serão abordadas nos próximos capítulos.

No que se segue, vamos supor conhecidos certos conceitos elementares de topologia de conjuntos. Em particular, vamos usar os conceitos de *espaço topológico*, *conjunto aberto* e *fechado*, *variedade topológica*, conjunto *conexo*, *vizinhança* de um ponto, *fecho*, *interior* e *fronteira* de um conjunto. Vamos utilizar também os conceitos de *homeomorfismo* (ou *equivalência*) entre dois espaços topológicos e *orientação* de uma variedade. De modo geral, tais conceitos serão utilizados de forma bastante elementar, e os leitores que já tiveram algum contato com os mesmos não devem ter dificuldades de acompanhar as discussões a seguir. Em todo caso, no apêndice A incluímos definições sucintas desses conceitos.

2.1 Definição de mapa

Informalmente, um mapa é uma subdivisão de uma superfície num número finito de elementos: pontos (*vértices*), curvas (*arestas*) e regiões bidimensionais (*faces*), tal que a fronteira exterior de um elemento de dimensão d é a união de elementos de dimensão

menor do que d. Isto é, a fronteira exterior de uma aresta é um conjunto finito de vértices; a fronteira exterior de uma face é a união de vértices e arestas.

Geralmente, para simplificar os algoritmos de manipulação de mapas, são impostas certas restrições sobre a topologia dos elementos de um mapa. Por exemplo, algumas restrições frequentes são: não permitir vértices isolados (que não sejam incidentes a nenhuma aresta), não permitir arestas que sejam curvas fechadas e não permitir faces com buracos. Normalmente, para contornar tais restrições adota-se o conceito de *elementos fictícios* que são introduzidos artificialmente no mapa para impedir que tais restrições sejam violadas.

No entanto, este tipo de solução apresenta alguns inconvenientes, pois os algoritmos de manipulação de mapas se tornam muito mais complexos, uma vez que os elementos fictícios devem ser ignorados, o que implica que o algoritmo deve tratar diversos casos especiais; e pior ainda, não há um método determinístico de inclusão de elementos fictícios no mapa, o que torna a categorização de mapas (por exemplo, a verificação de que dois mapas são isomorfos) um processo extremamente complexo.

Para contornar estes inconvenientes, vamos a seguir apresentar uma definição de mapas bidimensionais que não impõe os tipos de restrições citadas acima, tornando desnecessária a inclusão de elementos fictícios.

Para $n > 0$, seja $\mathbb{S}^n$ a superfície da esfera n-dimensional de raio 1 centrada na origem de $\mathbb{R}^n$. Vamos considerar a esfera $\mathbb{S}^n$ como um espaço topológico, com a topologia usual (induzida pela topologia padrão do $\mathbb{R}^n$). Vamos também supor definida uma *orientação canônica* para $\mathbb{S}^n$. Em particular, definimos a orientação canônica do círculo $\mathbb{S}^1$ como sendo o sentido de percurso anti-horário; e a orientação canônica da esfera $\mathbb{S}^2$ como sendo o sentido anti-horário visto do lado de fora da esfera. Em ambos os casos, supomos que os eixos estão dispostos da maneira usual. Desta forma,

Definição 1 Uma *oval* é um espaço topológico homeomorfo a $\mathbb{S}^1$.

Em outras palavras, uma oval é uma curva simples fechada (também conhecida como curva de Jordan); por exemplo, um círculo no plano.

Definição 2 Um *arco* é um espaço topológico homeomorfo à reta real $\mathbb{R}$. Um ponto qualquer de um arco e divide esse arco em dois outros arcos disjuntos denominados *pontas* de e.

Definição 3 Um *curva* é uma oval ou um arco.

Definição 4 Um *disco aberto* é um espaço topológico homeomorfo ao interior do círculo unitário (círculo centrado na origem com raio 1) de $\mathbb{R}^2$. Um *disco fechado* é um espaço topológico homeomorfo ao círculo unitário mais o seu interior.

Note que um disco aberto é homeomorfo ao plano $\mathbb{R}^2$.

Definição 5 Um *raio* de um disco D é um arco contido em D com um extremo na fronteira de D e outro no interior de D. Um *diâmetro* de um disco D é um arco contido em D com ambos os extremos na fronteira de D.

Definição 6 Um *multi-disco* é um espaço topológico D, homeomorfo a $\mathbb{S}^2$ menos um número finito k de discos fechados $B_1, B_2, \cdots, B_k$, disjuntos dois a dois. O número k é dito o *genus* de D. Sejam $U_1, U_2, \cdots, U_k$ discos abertos de $\mathbb{S}^2$, disjuntos dois a dois, tais que U_i contém B_i, para todo $i = 1, \cdots, k$. A imagem homeomórfica inversa de cada conjunto $U_i \setminus B_i$ é denominado uma *margem* do multi-disco D.

Para ilustrar o conceito de multi-disco, veja a figura 2.1.

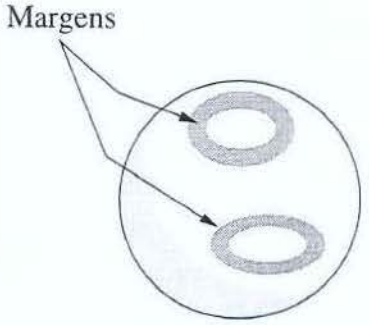


Figura 2.1: Exemplo de multi-disco e suas margens.

Definição 7 Um *mapa bidimensional* é um par $\mathcal{M} = (T_{\mathcal{M}}, \mathcal{L}_{\mathcal{M}})$, onde $T_{\mathcal{M}}$ é uma variedade bidimensional e $\mathcal{L}_{\mathcal{M}}$ é uma partição de $T_{\mathcal{M}}$ em três coleções finitas e disjuntas de *elementos*: os *vértices* $V_{\mathcal{M}}$, as *arestas* $E_{\mathcal{M}}$ e as *faces* $F_{\mathcal{M}}$, sendo que:

- um *vértice* é um subconjunto de $T_{\mathcal{M}}$ consistindo de um único ponto;
- uma *aresta* é uma curva de $T_{\mathcal{M}}$ cuja fronteira externa relativa a $T_{\mathcal{M}}$ é uma união de vértices;
- uma *face* é um multi-disco de $T_{\mathcal{M}}$ cuja fronteira externa relativa a $T_{\mathcal{M}}$ é uma união de vértices e arestas.

A união de todos os vértices e arestas de um mapa é denominada a *rede* do mapa.

A definição de arco é bastante liberal, e permite por exemplo arcos cuja fronteira externa é um conjunto de dimensão 1. No entanto, a definição de mapa restringe as

arestas de tal forma que a fronteira externa de uma aresta pode ser composta por 0, 1 ou 2 vértices. Caso esta fronteira seja vazia (isto é, possua 0 vértices), dizemos que a aresta é uma *aresta oval*; caso contrário, isto é, se a fronteira possui 1 ou 2 vértices, dizemos que a aresta é uma *aresta-arco*; além disso, se a fronteira contém apenas um vértice, dizemos que a aresta-arco é uma *aresta-laço*.

É importante ressaltar que a definição 7 generaliza o conceito de mapa normalmente utilizado na literatura. Em particular, ela permite mapas cuja rede é vazia; neste caso, o mapa é denominado *mapa trivial*, sendo composto por uma única face correspondente a toda a superfície de $T_{\mathcal{M}}$. Note que, pela definição de face, esta superfície $T_{\mathcal{M}}$ deve ser homeomorfa a $\mathbb{S}^2$. Dizemos que um mapa trivial é composto por uma única *face sem fronteira*.

Além disso, a definição também permite redes com mais de uma componente conexa; ou mapas com *vértices isolados* (vértices não incidentes a nenhuma aresta), com *arestas-laço* (arestas cujas extremidades coincidem num mesmo vértice), com arestas ovais (arestas que não possuem extremidades), etc. Veja figura 2.2.

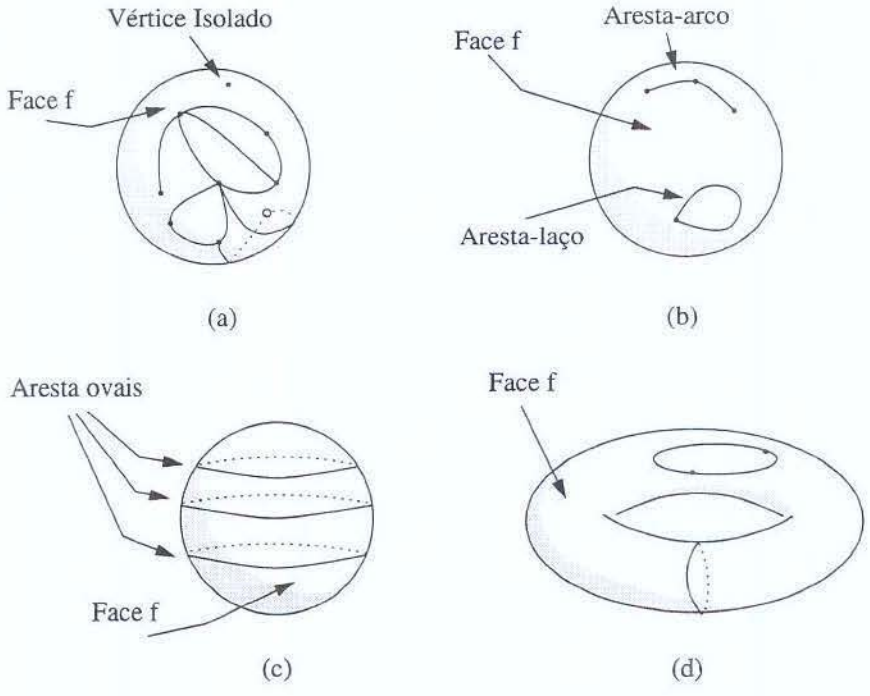


Figura 2.2: Exemplos de mapas bidimensionais.

Por outro lado, na figura 2.3, mostramos algumas subdivisões de variedades bidimensionais que não satisfazem as condições da definição 7. No caso (a) a segunda condição daquela definição é violada, pois a fronteira de e inclui pontos que não são vértices; em (b) a terceira condição é violada, pois f não é homeomorfa a um multi-disco.

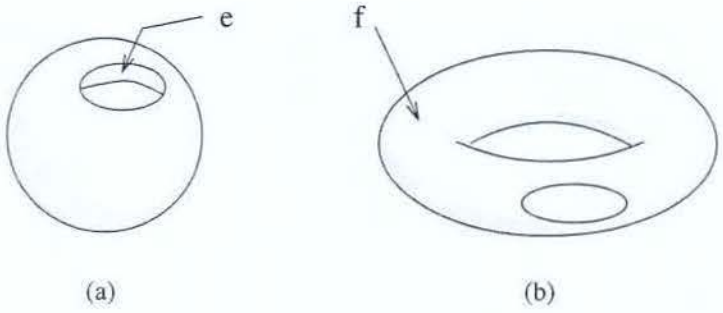


Figura 2.3: Exemplos de subdivisões que não são mapas bidimensionais.

2.2 Vizinhanças canônicas

Lema 1 *Todo vértice v de um mapa bidimensional $\mathcal{M} = (T_{\mathcal{M}}, \mathcal{L}_{\mathcal{M}})$ está contido num disco aberto D_v de $T_{\mathcal{M}}$ tal que:*

(i) D_v não contém nenhum outro vértice do mapa;

(ii) a intersecção de D_v com cada aresta do mapa consiste de $0, 1$, ou 2 arcos que são raios de D_v e pontas da aresta.

Lema 2 *Toda aresta-arco s de um mapa bidimensional $\mathcal{M} = (T_{\mathcal{M}}, \mathcal{L}_{\mathcal{M}})$ está contida num disco aberto D_s de $T_{\mathcal{M}}$ tal que:*

(i) s é um diâmetro de D_s;

(ii) D_s não intercepta nenhum vértice ou aresta do mapa exceto s.

Definição 8 Uma *faixa* de um espaço topológico T é um subespaço de T que é um cilindro ou uma fita de Möbius.

Lema 3 *Toda aresta oval c de um mapa bidimensional $\mathcal{M} = (T_{\mathcal{M}}, \mathcal{L}_{\mathcal{M}})$ está contida numa faixa H_c de $T_{\mathcal{M}}$ tal que:*

(i) c dá exatamente uma volta em torno de H_c;

(ii) H_c não intercepta nenhum vértice ou aresta do mapa exceto c.

Definição 9 Os discos abertos D_v e D_s e a faixa H_c definidos acima são denominados *vizinhanças canônicas* dos respectivos elementos: vértice v, aresta-arco s e aresta oval c.

2.3 Incidência entre os elementos

De acordo com as definições acima, a fronteira externa de uma aresta-arco de um mapa consiste de um par de pontos (não necessariamente distintos), que são denominados os *extremos* da aresta-arco.

Analogamente, a fronteira externa de uma face consiste de um número finito de componentes, cada uma das quais é uma união de vértices e arestas. Vale notar que cada uma destas componentes pode ser incidente a duas margens de duas faces distintas – figura 2.4(a) – ou a duas margens de uma mesma face – figura 2.4(b) – ou a uma margem de uma única face – veja a figura 2.4(c).

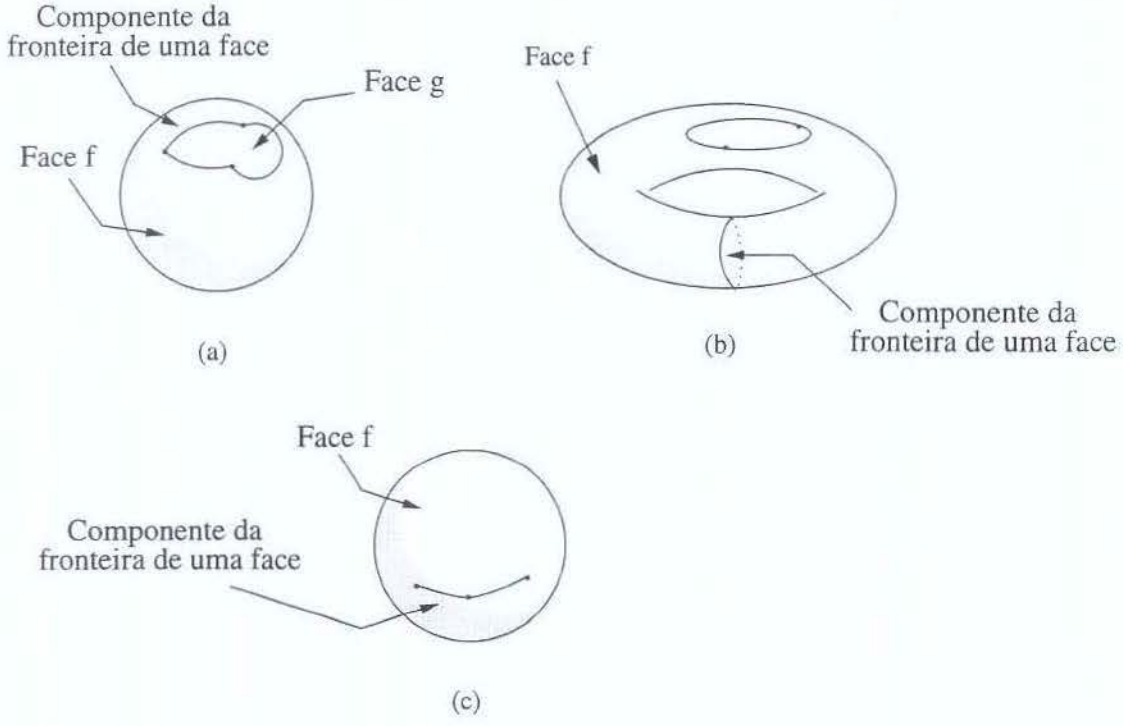


Figura 2.4: Incidência entre margens e componentes da fronteira de uma face.

2.4 Orientação

Dizemos que um mapa $\mathcal{M} = (T_{\mathcal{M}}, \mathcal{L})$ é *orientável* quando a superfície $T_{\mathcal{M}}$ é orientável. Além disso, se já temos definida alguma orientação sobre tal superfície então o mapa é dito *orientado*.

2.4.1 Orientação dos elementos

Para cada aresta de um mapa podemos definir duas *orientações internas* (ou longitudinais), que são os dois sentidos de percurso ao longo da aresta; ou as duas maneiras de distinguir a *ponta inicial* e a *ponta final* da aresta. Podemos definir também duas *orientações externas* (ou transversais), que são as duas maneiras possíveis de definir o *lado esquerdo* e o *lado direito* da aresta, numa vizinhança canônica da mesma. Estas duas orientações juntas constituem a *orientação total* da aresta.

Para indicar estas orientações, vamos utilizar um par de pequenas setas colocadas sobre um ponto qualquer da aresta: uma seta tangente à aresta, indicando a ordem de percurso da mesma, e a outra perpendicular à primeira, cruzando do lado direito para o lado esquerdo da aresta.

Uma maneira de visualizar estas orientações é imaginar um observador minúsculo andando sobre a superfície do mapa, ao longo da aresta: a orientação interna diz em que sentido ele está caminhando, e a externa diz em que lado da aresta está o seu pé esquerdo (ou seja, sobre que lado da superfície ele está andando). Veja a figura 2.5.

Figura 2.5: Orientações internas (ou longitudinais) e externas (ou transversais) de uma aresta.

Cada aresta pode ter portanto 4 orientações totais distintas. Veja figura 2.6. A orientação da superfície nos permite distinguir duas dessas orientações totais como *positivas* (aquelas nas quais o sentido de rotação da orientação interna para a externa pelo menor ângulo concorda com a orientação da superfície). As outras duas orientações são ditas *negativas*.

Note que a orientação externa de uma aresta nos permite distinguir a *face à esquerda* e a *face à direita* da aresta (que podem ser a mesma face). Adicionalmente, no caso de uma aresta-arco, a orientação interna desta aresta nos permite distinguir de maneira única o *vértice de origem* e o *vértice de destino* desta aresta (que também podem ser o mesmo vértice).

No caso de um vértice, podemos ter duas *orientações externas* que correspondem aos dois sentidos de rotação em torno do mesmo.

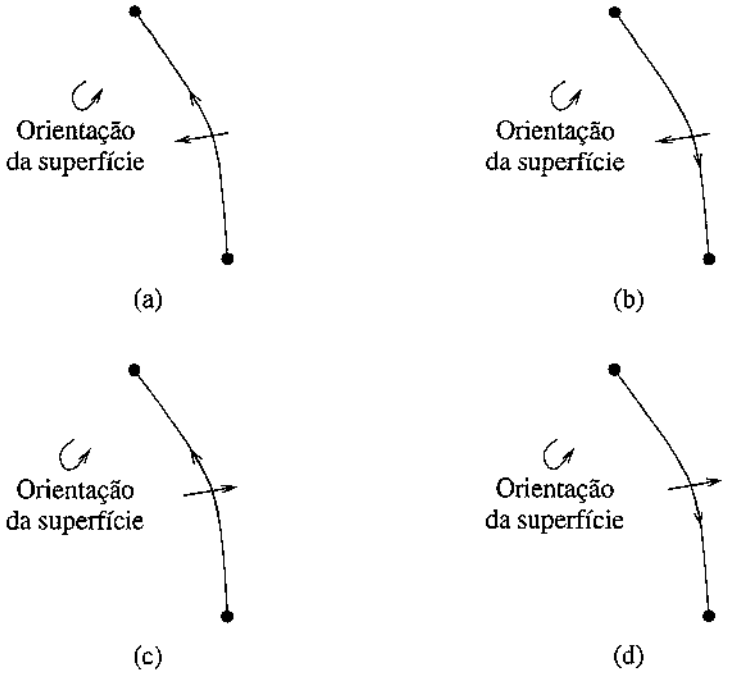


Figura 2.6: As quatro orientações de uma aresta: (a) e (d) orientações totais positivas; (b) e (c) orientações totais negativas.

No caso de uma face, podemos ter duas *orientações internas* que correspondem aos sentidos de rotação dentro da mesma.

Tanto no caso do vértice como da face, a orientação da superfície permite distinguir as orientações *positiva* e *negativa* do elemento.

De agora em diante, a menos quando dito o contrário, vamos supor que vértice, aresta e face se referem a elementos orientados, isto é, a palavra "aresta" será utilizada para indicar uma "aresta orientada"; o mesmo vale para vértice e face.

2.5 Característica de Euler

Como sabemos, os elementos de um mapa tradicional, onde não existem vértices isolados nem arestas ovais e onde as faces possuem genus 0, satisfazem a equação de Euler-Poincaré [18]:

$$n_v - n_e + n_f = \chi(T)$$

onde n_v, n_e e n_f são respectivamente os números de vértices, arestas e faces do mapa e $\chi(T)$, um número inteiro que depende apenas da topologia de T, é a *característica de Euler* da superfície T.

Para estender esta equação aos nossos mapas bidimensionais vamos utilizar o seguinte lema:

Lema 4 *Um multi-disco D de genus $k > 1$ pode ser particionado num único disco e $k-1$ arestas-arco, disjuntas duas a duas, com extremos na fronteira de D.*

Com isto podemos provar que

Teorema 5 *Se $\mathcal{M} = (T, \mathcal{L})$ é mapa bidimensional conexo e orientável então a caracterítica de Euler da superfície T é dada por*

$$\chi(T) = n_v - n_a + n_f + (n_f - n_m) \tag{2.1}$$

onde n_v, n_a, n_f e n_m são respectivamente o número de vértices, arestas-arco (excluindo arestas ovais), faces e margens no mapa.

DEMONSTRAÇÃO: Se o mapa é trivial (isto é, possui apenas uma face sem fronteira), então é fácil ver que a equação (2.1) é satisfeita, pois, por definição, a superfície T é homeomorfa a $\mathbb{S}^2$, e $n_v - n_a + n_f + (n_f - n_m) = 0 - 0 + 1 + (1 - 0) = 2$, que é a característica de Euler da esfera $\mathbb{S}^2$. Analogamente, se o mapa possui apenas um vértice isolado e zero arestas, então $n_v - n_a + n_f + (n_f - n_m) = 1 - 0 + 1 + (1 - 1) = 2$, e por outro lado, a única face do mapa é um disco; daí segue-se que a superfície T é homeomorfa a $\mathbb{S}^2$ cuja característica de Euler é 2.

Agora, suponha que o mapa possua n_o arestas ovais. Então, o número total de arestas é $n_e = n_a + n_o$. Note que se inserirmos n_o vértices no mapa, um sobre cada aresta oval, transformando-as em arestas-laço, passamos a ter um mapa com $n_v + n_o$ vértices, zero arestas ovais e $n_a + n_o$ arestas-arco. Por outro lado, o número de margens e faces permanece inalterado. Deste modo, após essa transformação, o valor do lado direito da equação (2.1) mantém-se inalterado. Logo, podemos supor que o mapa não tem arestas ovais.

Pelo lema 4, sabemos que uma face f com $k > 1$ margens pode ser decomposta em uma face com uma única margem e mais $k - 1$ arestas-arco. Neste processo de decomposição, pode ser necessária a inclusão de novos vértices nas fronteiras das margens envolvidas; porém, se inserirmos um novo vértice sobre uma aresta quebrando-a em duas novas arestas, o lado direito da equação (2.1) é preservado, pois tanto o número de vértices n_v como o número de arestas-arco n_a são acrescidos de 1. Portanto, aplicando-se este processo a todas as faces do mapa com mais de uma margem, obtemos ao final um mapa $\mathcal{M}' = (T, \mathcal{L}')$ com n_f faces que são discos, n'_e arestas que são arcos (e não são ovais) e n'_v vértices que não são isolados. Logo, neste mapa, vale a equação Euler-Poincaré [18]; isto é,

$$\chi(T) = n'_v - n'_e + n'_f$$

Visto que, nesse mapa $n'_e = n'_a$ (pois, $n'_o = 0$) e $n'_m = n'_f$ (cada face possui apenas uma margem) então

$$\chi(T) = n'_v - n'_a + n'_f + (n'_f - n'_m)$$

Isto implica que a equação (2.1) é válida no mapa $\mathcal{M}'$. Uma vez que a obtenção do mapa $\mathcal{M}'$ a partir do mapa $\mathcal{M}$ foi efetuada de modo a sempre preservar a equação (2.1) então, aplicando um processo inverso, podemos concluir que esta equação também é válida no mapa original $\mathcal{M}$. □

2.6 Mapas esféricos

Definição 10 Um mapa sobre a superfície $\mathbb{S}^2$, isto é, o mapa $\mathcal{M} = (\mathbb{S}^2, \mathcal{L}_\mathcal{M})$, é denominado um *mapa esférico*.

Corolário 6 *Um mapa bidimensional conexo e orientável é homeomorfo a um mapa esférico se e somente se*

$$n_v - n_a + n_f + (n_f - n_m) = 2$$

onde n_v, n_a, n_f e n_m são respectivamente o número de vértices, arestas-arco (excluindo arestas ovais), faces e margens do mapa.

Capítulo 3

As esquinas de um mapa esférico

De agora em diante, vamos nos restringir ao estudo dos mapas esféricos, sendo que o nosso objetivo neste capítulo é definir um método que nos permita a representação da topologia desses mapas.

Para que um mapa possa ser manipulado no computador, é importante definirmos uma codificação puramente combinatória — isto é, uma estrutura de dados — para representar a sua topologia. Na literatura há um número considerável de estruturas para esse fim. Dentre elas podemos destacar: *half-edge* [26], *quad-edge* [16], *winged-edge* [2], *vertex-edge* [36], *radial edge* [4], *grafos de incidência* [11], *3-gemas* [24], *elementos de incidência* [31], *cell-tuple structure* [3] e *n-G-maps* [22]. No entanto, devido a algumas particularidades dos mapas esféricos que nos interessam, por exemplo, a presença de vértices isolados e principalmente de ovais, não podemos utilizar diretamente nenhuma dessas estruturas. Por isso, neste capítulo vamos desenvolver uma adaptação da estrutura *half-edge* de modo a permitir a representação de tais elementos.

É fácil ver que a topologia de um mapa (em particular, de um mapa esférico) não pode ser completamente descrita considerando-se apenas as relações de incidência entre seus elementos. Por exemplo, as figuras 3.1(a) e (b) mostram dois mapas topologicamente distintos com conjuntos de elementos isomorfos e com as mesmas relações de incidência.

Por este motivo, todas as representações combinatórias de mapas devem incluir, além das relações de incidência, algumas informações adicionais. Conforme demonstrado por Edmonds [12], é suficiente incluir a descrição da ordem das arestas em torno de cada vértice e em torno de cada face.

3.1 Definição de esquinas

Visto que um mapa esférico é um mapa orientável, vamos então considerar apenas elementos com orientação (total) positiva. Portanto, arestas podem ter apenas duas orientações

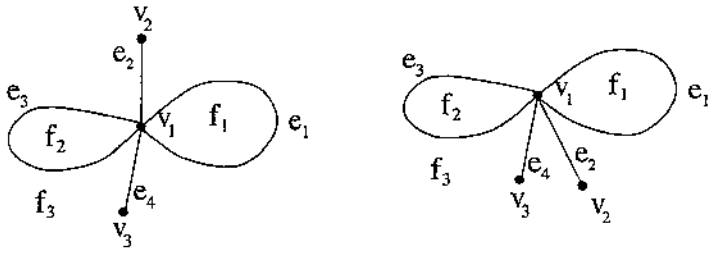


Figura 3.1: Apenas as relações de incidência não são suficientes para descrever a topologia de um mapa.

distintas, enquanto que vértices e faces podem ter apenas uma. Para indicar a aresta que consiste da mesma curva que a aresta e mas com orientação (total) invertida, vamos utilizar a notação $\neg e$.

O objetivo da estrutura *half-edge* é representar as relações de incidência e ordem entre os elementos (vértices, arestas e faces) de um mapa. Para isto, cada aresta é conceitualmente dividida em duas pontas (correspondentes às duas orientações), e para cada uma delas indicam-se o vértice de origem e a face esquerda. Além disso, para completar a representação, indica-se também a ordem das arestas em torno de cada vértice, e a ordem das arestas em torno de cada face. Para mais detalhes, o leitor pode consultar o livro de Mäntylä [26] ou a tese de doutorado de Weiler [36].

A definição abaixo é uma formalização da estrutura *half-edge* que inclui as extensões necessárias para representar vértices isolados, ovais e faces com múltiplas bordas.

Definição 11 Dado um mapa $\mathcal{M}$, seja $\varnothing$ um símbolo distinto de todos os elementos deste mapa. A tripla $c = \lfloor v, e, f \rfloor$ é denominada uma *esquina do mapa* se uma das seguintes condições é verificada:

(i) e é uma aresta-arco positivamente orientada de $\mathcal{M}$, v é a origem de e, e f é a face à esquerda de e;

(ii) e é uma aresta oval positivamente orientada de $\mathcal{M}$, f é a face à esquerda de e e $v = \varnothing$, indicando o "vértice indefinido";

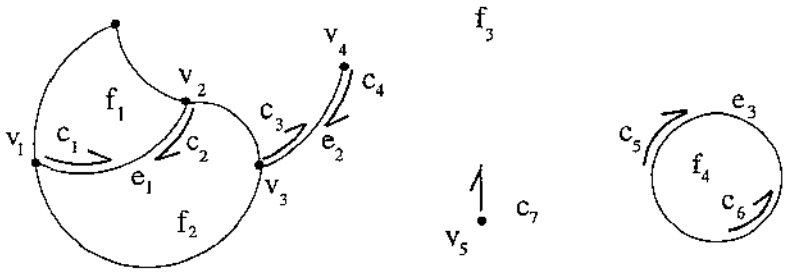
(iii) v é um vértice isolado positivamente orientado de $\mathcal{M}$, incidente na face f, e $e = \varnothing$, indicando a "aresta indefinida".

Denotaremos por $Q_{\mathcal{M}}$ o conjunto de esquinas do mapa $\mathcal{M}$. Dada uma esquina $c = \lfloor v, e, f \rfloor$ em $Q_{\mathcal{M}}$, escreveremos $Org(c) = v$, $Edge(c) = e$ e $Left(c) = f$. Note que Org é

uma função de $Q_{\mathcal{M}}$ para $V_{\mathcal{M}} \cup \{\varnothing\}$; *Edge* é uma função de $Q_{\mathcal{M}}$ para $E_{\mathcal{M}} \cup \{\varnothing\}$ e $Left$ é uma função total de $Q_{\mathcal{M}}$ para $F_{\mathcal{M}}$.

Observe que se $\mathcal{M}$ é o mapa esférico trivial (isto é, o mapa esférico com uma única face $f = \mathbb{S}^2$, zero vértices e zero arestas), o conjunto $Q_{\mathcal{M}}$ é vazio.

Nas ilustrações, indicaremos uma esquina $\lfloor v, e, f \rfloor$ pelo símbolo $\rightharpoonup$ colocado do lado esquerdo da aresta e, saindo do vértice de origem v (se houver). Veja as figuras 3.2(a) e (c). Ou seja, a seta indica a orientação interna da aresta e a meia-aspa indica a orientação externa. No caso particular de um vértice isolado a direção da haste é irrelevante e a meia-aspa indica a orientação do vértice. Por exemplo, veja a figura 3.2(b).



$c_1 = \lfloor v_1, e_1, f_1 \rfloor$ $c_2 = \lfloor v_2, \neg e_1, f_2 \rfloor$ $c_3 = \lfloor v_3, e_2, f_3 \rfloor$ $c_4 = \lfloor v_4, \neg e_2, f_3 \rfloor$
$c_5 = \lfloor \varnothing, e_3, f_3 \rfloor$ $c_6 = \lfloor \varnothing, \neg e_3, f_4 \rfloor$ $c_7 = \lfloor v_5, \varnothing, f_3 \rfloor$

Figura 3.2: Esquinas de um mapa.

3.2 Função *Sym*

Observe que sobre uma aresta (arco ou oval) sempre existem duas esquinas distintas, correspondentes às duas orientações da aresta.

Definição 12 Para toda esquina $c = \lfloor v, e, f \rfloor$ tal que e é uma aresta-arco, $Sym(c)$ é a esquina *simétrica* a c, definida sobre a aresta e tomada com orientação oposta; isto é, $Sym(c) = \lfloor u, \neg e, g \rfloor$, onde u é o vértice de destino de e e g é a face à direita de e. Além disso, para toda esquina $c = \lfloor \varnothing, e, f \rfloor$ sobre uma aresta oval, $Sym(c) = \lfloor \varnothing, \neg e, g \rfloor$, onde g é a face à direita de e. Finalmente, para toda esquina $c = \lfloor v, \varnothing, f \rfloor$ sobre um vértice isolado v, $Sym(c) = c$.

Por exemplo, na figura 3.2, $Sym(c_1) = c_2$, $Sym(c_5) = c_6$ e $Sym(c_7) = c_7$.

3.3 Propriedades das esquinas

A partir das definições 11 e 12 podemos verificar facilmente que, para toda esquina c:

P1. $Org(c)$ é um vértice isolado se e somente se $Edge(c) = \varnothing$;

P2. $Edge(c)$ é uma aresta oval se e somente se $Org(c) = \varnothing$;

P3. $Edge(Sym(c)) = \neg Edge(c)$, (convencionando-se que $\neg \varnothing = \varnothing$);

P4. $Sym(Sym(c)) = c$;

P5. $Sym(c) \neq c$ se e somente se $Edge(c) \neq \varnothing$

Observe que o conjunto de esquinas e as funções Org, $Edge$ e $Left$ descrevem completamente as relações de incidência entre os elementos que compõem um mapa.

3.4 Funções de percurso

Como vimos anteriormente, a descrição completa da topologia de um mapa exige, além das relações de incidência entre os elementos, também as relações de ordem entre os mesmos. Para estabelecer estas relações, vamos definir duas funções adicionais, denominadas *funções de percurso.*

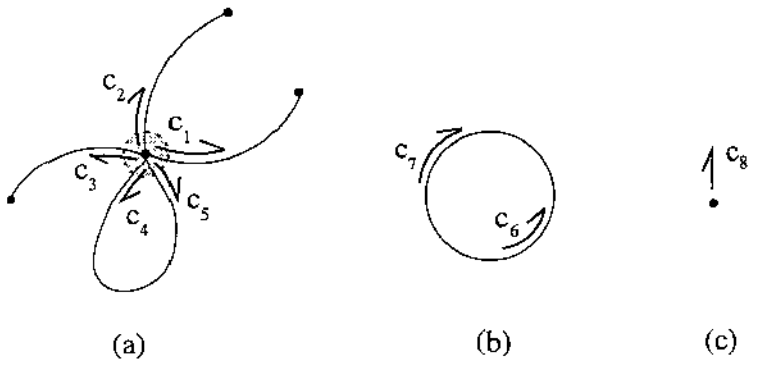
3.4.1 A função *Onext*

Pelo lema 1, as arestas-arco orientadas com origem num vértice v podem ser ordenadas ciclicamente em torno de v, com base na orientação da superfície. Esta relação de ordem será representada através da função $Onext(c)$. Ou seja, dada uma esquina $c = \lfloor v, e, f \rfloor$ definida sobre uma aresta-arco (isto é, com $v \neq \varnothing$ e $e \neq \varnothing$), então $Onext(c)$ é a próxima esquina com origem em v, no sentido positivo de rotação em torno de v. Veja a figura 3.3(a).

No caso especial da aresta $Edge(c)$ ser uma aresta oval, isto é, se $c = \lfloor \varnothing, e, f \rfloor$, definimos $Onext(c)$ como sendo a esquina $\lfloor \varnothing, \neg e, g \rfloor$, onde g é a face à direita de e; ou seja, $Onext(c) = Sym(c)$. Por exemplo, na figura 3.3(b), $Onext(c_6) = c_7$ e $Onext(c_7) = c_6$.

Finalmente, se $Org(c)$ é um vértice isolado, isto é, se $Edge(c) = \varnothing$, então $Onext(c) = c$, por definição. Por exemplo, na figura 3.3(c), $Onext(c_8) = c_8$.

Pode-se verificar que $Onext$ é uma função total e bijetora de $Q_{\mathcal{M}}$ em $Q_{\mathcal{M}}$. Denotaremos a sua inversa por $Oprev(c)$. Ou seja, se $Edge(c)$ é uma aresta-arco, então $Oprev(c)$ retorna a esquina com origem em $v = Org(c)$ que é anterior a c no sentido positivo de rotação em torno de v. Se $Edge(c)$ é uma aresta oval, então $Oprev(c) = Onext(c) = Sym(c)$; e, se $Edge(c) = \varnothing$, então $Oprev(c) = c$.



(a) (b) (c)

Esquina	Onext
c_1	c_2
c_2	c_3
c_3	c_4
c_4	c_5
c_5	c_1
c_6	c_7
c_7	c_6
c_8	c_8

Figura 3.3: Exemplos da função *Onext*.

3.4.2 A função *Lnext*

Observe que a fronteira de uma face f consiste de uma ou mais componentes conexas, sendo que estas componentes conexas são compostas pelos vértices $Org(c)$ e pelas arestas $Edge(c)$ de todas as esquinas c tais que $Left(c) = f$. Em particular, a fronteira de uma face pode ser vazia.

Como se pode verificar, a ordem em que as arestas ocorrem em cada componente conexa na fronteira de uma face é dada pela função *Lnext* definida, para toda esquina c, por

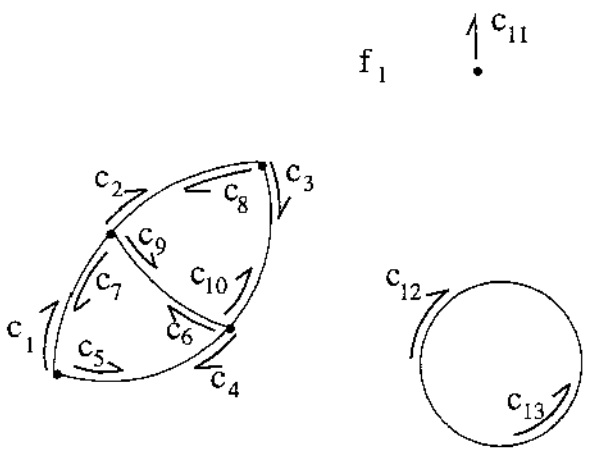
$$Lnext(c) = Oprev(Sym(c)) \tag{3.1}$$

Em particular, se $Org(c) = \varnothing$ ou $Edge(c) = \varnothing$ (isto é, se $Edge(c)$ é uma aresta oval ou se $Org(c)$ é um vértice isolado) então, pelas definições de Sym e de $Oprev$, temos que $Lnext(c) = c$. Veja a figura 3.4.

É fácil ver que *Lnext*, assim como *Onext*, é uma função total e bijetora de $Q_{\mathcal{M}}$ em $Q_{\mathcal{M}}$. Denotaremos a sua inversa por *Lprev*.

Intuitivamente, $Lnext(c)$ é a esquina seguinte a c na mesma componente conexa da face $Left(c)$, quando esta componente é percorrida no sentido positivo (visto de um ponto no interior da face $Left(c)$). Portanto, partindo-se de uma esquina c, a função *Lnext* determina uma ordenação circular das esquinas definidas sobre os elementos de uma componente conexa da face $Left(c)$. Por exemplo, na figura 3.4, partindo-se da esquina c_1 obtemos as esquinas $\langle c_1, c_2, c_3, c_4\rangle$, onde $\langle\cdots\rangle$ denota uma ordenação circular.

Mais especificamente, suponha que a sequência circular de esquinas alcançadas pela função *Lnext* a partir da esquina c_1 é $\langle c_1, c_2, \cdots, c_k\rangle$ e que $Org(c_1) \neq \varnothing$ e $Edge(c_1) \neq \varnothing$; então, a sequência circular de arestas alcançadas neste percurso é $\langle Edge(c_1), Edge(c_2), \cdots, Edge(c_k)\rangle$, sendo que o vértice de destino da aresta $Edge(c_i)$ coincide com o vértice de origem da aresta $Edge(c_{i+1})$; isto é, $Org(Sym(c_i)) = Org(c_{i+1})$.



Esquina c	$Lnext(c)$
c_1	c_2
c_2	c_3
c_3	c_4
c_4	c_1
c_5	c_6
c_6	c_7
c_7	c_5
c_8	c_9
c_9	c_{10}
c_{10}	c_8
c_{11}	c_{11}
c_{12}	c_{12}
c_{13}	c_{13}

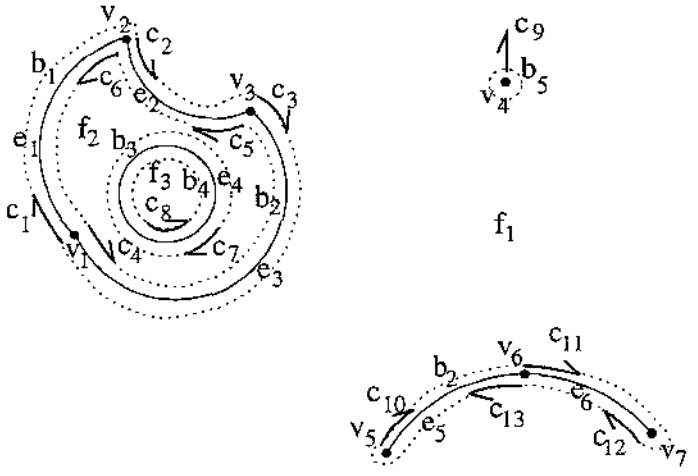
Figura 3.4: Exemplo da função *Lnext*.

Veja a figura 3.4. Note que esta observação também vale trivialmente se $Org(c) = \varnothing$ ou $Edge(c) = \varnothing$.

Assim sendo, podemos concluir que cada componente conexa da fronteira de uma face corresponde a uma órbita da função *Lnext*. Vamos denominar cada uma dessas órbitas de uma *borda* da face, e vamos denotar por $Bord(c)$ a órbita que contém a esquina c. Por exemplo, na figura 3.4, as bordas da face f_1 são $Bord(c_1) = \langle c_1, c_2, c_3, c_4 \rangle$, $Bord(c_{11}) = \langle c_{11} \rangle$ e $Bord(c_{12}) = \langle c_{12} \rangle$.

Observe que a toda borda fica associada uma orientação (um sentido de percurso) determinada pelas esquinas que compõem aquela borda.

Para melhor ilustrar a relação entre as funções de percurso *Onext*, *Lnext* e *Bord*, veja a figura 3.5.



(a)

Esquina	$Onext(c)$	$Lnext(c)$	$Bord(c)$
$c_1 = \lfloor v_1, e_1, f_1 \rfloor$	c_4	c_2	$\langle c_1, c_2, c_3 \rangle$
$c_2 = \lfloor v_2, e_2, f_1 \rfloor$	c_6	c_3	
$c_3 = \lfloor v_3, e_3, f_1 \rfloor$	c_5	c_1	
$c_4 = \lfloor v_1, \neg e_3, f_2 \rfloor$	c_1	c_5	$\langle c_4, c_5, c_6 \rangle$
$c_5 = \lfloor v_3, \neg e_2, f_2 \rfloor$	c_3	c_6	
$c_6 = \lfloor v_2, \neg e_1, f_2 \rfloor$	c_2	c_4	
$c_7 = \lfloor \varnothing, e_4, f_2 \rfloor$	c_8	c_7	$\langle c_7 \rangle$
$c_8 = \lfloor \varnothing, \neg e_4, f_3 \rfloor$	c_7	c_8	$\langle c_8 \rangle$
$c_9 = \lfloor v_4, \varnothing, f_1 \rfloor$	c_9	c_9	$\langle c_9 \rangle$
$c_{10} = \lfloor v_5, e_5, f_1 \rfloor$	c_{10}	c_{11}	$\langle c_{10}, c_{11}, c_{12}, c_{13} \rangle$
$c_{11} = \lfloor v_6, e_6, f_1 \rfloor$	c_{13}	c_{12}	
$c_{12} = \lfloor v_7, \neg e_6, f_1 \rfloor$	c_{12}	c_{13}	
$c_{13} = \lfloor v_6, \neg e_5, f_1 \rfloor$	c_{11}	c_{10}	

(b)

Figura 3.5: Exemplo das funções *Onext*, *Lnext* e *Bord*.

Capítulo 4

Representação da topologia de mapas esféricos

Baseado na descrição apresentada no capítulo anterior, vamos agora definir uma estrutura de dados denominada SMC (*Spherical Maps by Corners*) que nos permite representar a topologia dos mapas esféricos. Esta estrutura consiste basicamente da estrutura *half-edge* [26] com extensões motivadas pela definição 11.

Além disso, vamos também definir um conjunto de *operadores topológicos* que nos permitem manipular esta estrutura, sendo que vamos considerar três categorias de operadores: de criação de mapas e elementos (seção 4.3), de transferência de elementos entre mapas (seção 4.4), e de conexão e desconexão de elementos num mapa (seção 4.5).

4.1 A estrutura de dados SMC

Conforme vimos anteriormente, a topologia de um mapa esférico pode ser completamente descrita utilizando-se as funções Org, $Edge$, $Left$, Sym, $Onext$ e $Lnext$. Na verdade, basta especificar quaisquer duas dentre as três funções $Onext$, Sym e $Lnext$, pois o valor da terceira função pode ser obtido utilizando-se uma das seguintes relações:

$$\begin{aligned} Lnext(c) &= Onext^{-1}(Sym(c)) \\ Sym(c) &= Onext(Lnext(c)) \\ Onext(c) &= Sym(Lnext^{-1}(c)) \end{aligned}$$

Note que se escolhermos representar as funções $Onext$ e Sym, então para determinarmos o valor de $Lnext(c)$ precisamos computar o valor de $Onext^{-1}(c')$, onde $c' = Sym(c)$, e para isto é necessário percorrer todo o ciclo de esquinas em torno do vértice $Org(c')$, o que não pode ser realizado a um custo constante. Uma situação análoga ocorre se

escolhermos representar as funções Sym e $Lnext$. Por outro lado, se representarmos as funções $Onext$ e $Lnext$, então o valor de Sym pode ser determinado diretamente a partir delas a um custo constante.

Por este motivo, na estrutura SMC, cada esquina c será representada por um objeto, de tipo Corner, contendo apontadores para os objetos que representam as esquinas $Onext(c)$ e $Lnext(c)$.

Além desses apontadores — que ligam os nós do tipo Corner entre si — cada Corner c contém três outros apontadores, denominados $c.org$, $c.edge$, $c.bord$, que apontam para objetos que representam o vértice $Org(c)$, a aresta $Edge(c)$ e a borda $Bord(c)$, de tipos Vertex, Edge e Border, respectivamente. Os casos especiais $Org(c) = \varnothing$ e $Edge(c) = \varnothing$ são representados por $c.org$ = Nil e $c.edge$ = Nil, respectivamente. Vale notar que para cada aresta não orientada do mapa existem 2 registros do tipo Edge, um para cada orientação da aresta, com S-círculos opostos.

A razão para armazenarmos um apontador para a borda $Bord(c)$ que contém a esquina c, e não para a face $Left(c)$, é que a fronteira de uma face pode ser composta por um conjunto com várias bordas que, em geral, precisam ser diferenciadas nos algoritmos de manipulação de mapas.

Num objeto b do tipo Border, armazenamos um apontador $b.face$ para a face cuja fronteira contém aquela borda; e, além disso, em $b.desc$ armazenamos uma esquina qualquer da borda, a qual denominamos *a descritora* da borda. Note que as outras esquinas que compõem a borda podem ser obtidas a partir de $b.desc$ utilizando-se a função $Lnext$.

Para representar uma face f, utilizamos um objeto do tipo Face com um campo $f.frontier$, onde armazenamos o conjunto de bordas que constituem a fronteira da face.

Desta forma, para nos referirmos a um mapa, basta especificarmos uma esquina ou uma face deste mapa. Lembramos que o mapa trivial não tem esquinas.

A figura 4.1 mostra a definição em Modula-3 [28] da estrutura SMC, e a figura 4.2, ilustra o encadeamento entre os elementos desta estrutura. É importante notar que, no caso dos objetos Vertex e Edge não há apontadores de retorno para as esquinas definidas sobre estes elementos.

O desenvolvimento de um algoritmo para verificar a consistência/validade da estrutura SMC está fora do escopo deste trabalho. No entanto, uma maneira simples de se efetuar esta validação é converter a estrutura SMC para uma *quad-edge* [16] (para superfícies orientadas) e validar a *quad-edge* obtida desta conversão.

4.2 Informações não topológicas

Os nós e apontadores descritos acima representam apenas as informações topológicas do mapa. No entanto, existem outras informações dependentes da aplicação, associadas aos

```
Corner = object
    org : Vertex;
    edge : Edge;
    bord : Border;
    onext : Corner;
    lnext : Corner;
end;

Vertex = object
    props : VertProp;          {geometria do ponto e outras propriedades}
end;

Edge = object
    props : EdgeProp;          {geometria da aresta e outras propriedades}
end;

Border = object
    face : Face;
    desc : Corner;
end

Face = object
    props : FaceProp;          {propriedades dependentes da aplicação}
    frontier : set of Border;  {conjunto (lista) das bordas na fronteira da face}
end;
```

Figura 4.1: Estrutura de dados SMC.

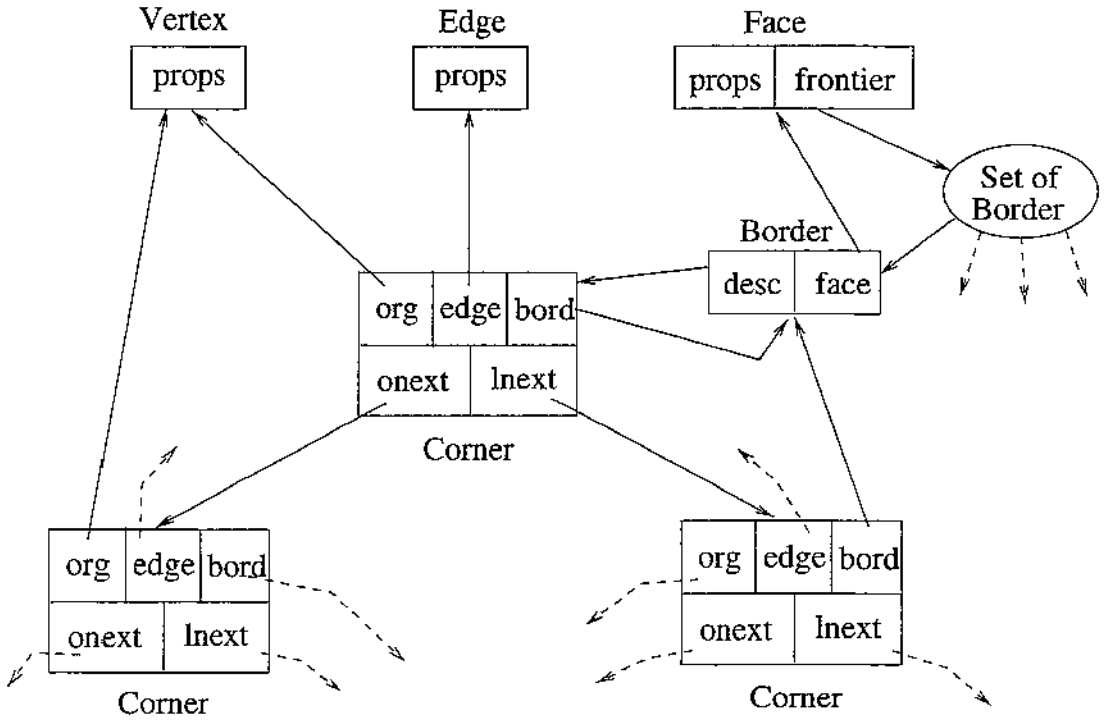


Figura 4.2: Ilustração da estrutura SMC.

elementos que compõem o mapa, como por exemplo: a geometria, a cor, a textura, etc, que também devem ser armazenadas na estrutura. Por isso, vamos supor a existência de objetos dos tipos VertProp, EdgeProp e FaceProp onde são armazenadas tais informações, sendo que estes objetos são apontados pelos campos *props* nos nós Vertex, Edge e Face, respectivamente.

As aplicações que fazem uso da estrutura SMC devem definir estes objetos conforme necessário. Em particular, nos objetos VertProp e EdgeProp deve ser incluída a representação da geometria do vértice e da aresta.

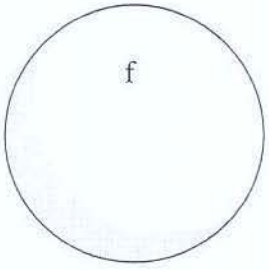
É importante ressaltar que, nos operadores topológicos que descreveremos a seguir, quando são criados objetos do tipo Vertex, Edge e Face, os respectivos campos *props* são deixados indefinidos, isto é, com valores nulos. Fica a cargo do cliente da estrutura SMC a devida atualização destes valores.

4.3 Operadores de criação

Conforme veremos posteriormente, os operadores de conexão, desconexão e transferência nos permitem criar qualquer mapa esférico a partir dos seguintes *mapas elementares*: o mapa trivial (com uma face, sem arestas e sem vértices); o mapa com um único vértice

isolado numa face; o mapa com uma aresta-arco com dois vértices distintos numa face; e o mapa com uma aresta oval delimitando duas faces. É fácil perceber que a criação de cada um destes mapas exige a criação de elementos diferentes, isto é, objetos de tipos diferentes na estrutura SMC. Por este motivo, é conveniente definir um operador topológico distinto para criar cada um destes mapas elementares: MakeFace, MakeVertex, MakeArcEdge e MakeOvalEdge.

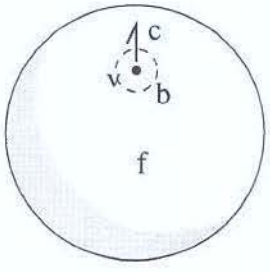
O operador MakeFace cria um mapa trivial, isto é, um mapa esférico com uma única face com zero bordas. Veja figura 4.3. Do ponto de vista da estrutura SMC, este operador consiste em criar um objeto f do tipo Face cujo campo $f.frontier$ é o conjunto vazio; este operador retorna o objeto f.




```
procedure MakeFace() : Face;
    f ← new(Face);
    f.frontier ← {};
    return f;
```

Figura 4.3: Operador MakeFace

O operador MakeVertex cria um novo mapa com um único vértice v, zero arestas, e uma única face f com uma única borda b consistindo do vértice v. Portanto, este operador cria um mapa com uma única esquina $c = \lfloor v, \varnothing, f \rfloor$, que é o valor retornado pelo operador. Veja figura 4.4.

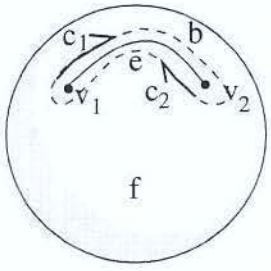



```
procedure MakeVertex() : Corner;
    f ← new(Face);
    b ← new(Border);
    v ← new(Vertex);
    c ← new(Corner);
    f.frontier ← {b};
    b.face ← f;
    b.desc ← c;
    c.org ← v;
    c.edge ← ∅;
    c.bord ← b;
    c.onext ← c;
    c.lnext ← c;
    return c;
```

Figura 4.4: Operador MakeVertex

O operador MakeArcEdge cria um mapa com dois vértices v_1 e v_2, uma aresta-arco e, e uma face f. A fronteira de f é composta por uma única borda b que contém as duas esquinas $c_1 = \lfloor v_1, e, f \rfloor$ e $c_2 = \lfloor v_2, \neg e, f \rfloor$. Por definição, este operador retorna uma das esquinas. Veja figura 4.5.

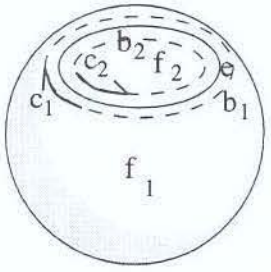
Para criar uma aresta-laço basta utilizar este operador seguido pelo operador JoinVerticesSplitBorder descrito na seção 4.5.4.




```
procedure MakeArcEdge() : Corner;
    f ← new(Face);
    b ← new(Border);
    e ← new(Edge);
    v1 ← new(Vertex);    v2 ← new(Vertex);
    c1 ← new(Corner);    c2 ← new(Corner);
    f.frontier ← {b};
    b.face ← f;
    b.desc ← c1;
    c1.org ← v1;         c2.org ← v2;
    c1.edge ← e;         c2.edge ← ¬e;
    c1.bord ← b;         c2.bord ← b;
    c1.onext ← c1;       c2.onext ← c2;
    c1.lnext ← c2;       c2.lnext ← c1;
    return c1;
```

Figura 4.5: Operador MakeArcEdge

O operador MakeOvalEdge cria um mapa com zero vértices, uma única aresta oval e, e com duas faces f_1 e f_2. A fronteira de cada uma dessas faces contém apenas uma borda, cada uma com uma única esquina. Por definição, o resultado retornado por este operador é uma das esquinas sobre a aresta oval. Veja figura 4.6.




```
procedure MakeOvalEdge() : Corner;
    f1 ← new(Face);          f2 ← new(Face);
    b1 ← new(Border);        b2 ← new(Border);
    e ← new(Edge);
    c1 ← new(Corner);        c2 ← new(Corner);
    f1.frontier ← {b1};      f2.frontier ← {b2};
    b1.face ← f1;            b2.face ← f2;
    b1.desc ← c1;            b2.desc ← c2;
    c1.org ← ∅;              c2.org ← ∅;
    c1.edge ← e;             c2.edge ← ¬e;
    c1.bord ← b1;            c2.bord ← b2;
    c1.onext ← c2;           c2.onext ← c1;
    c1.lnext ← c1;           c2.lnext ← c2;
    return c1;
```

Figura 4.6: Operador MakeOvalEdge

Como é fácil perceber, todos esses operadores têm custo constante.

4.4 Operador de transferência

Visto que os operadores de criação apresentados na seção anterior sempre criam um novo mapa para conter o novo elemento criado, é necessário definir operadores que nos permitam transferir elementos de um mapa para outro. Isto é, vamos definir o operador

Transfer(b, g) que transfere uma borda b da fronteira da sua atual face para a fronteira de uma face qualquer g.

Mais precisamente, seja f a face situada à esquerda da borda b. Intuitivamente, o operador Transfer(b, g) corresponde a recortar a borda b da fronteira da face f, retirando-a daquela face. O buraco resultante na face f é implicitamente preenchido com um disco que é incorporado à mesma. Veja as figuras 4.7(a), (b) e (c). Adicionalmente, o operador recorta um disco da face g e cola, sobre o buraco resultante, a borda b. Veja as figura 4.7(d), (e) e (f). Note que todos os elementos situados na região à direita da borda b são transferidos junto com a mesma.

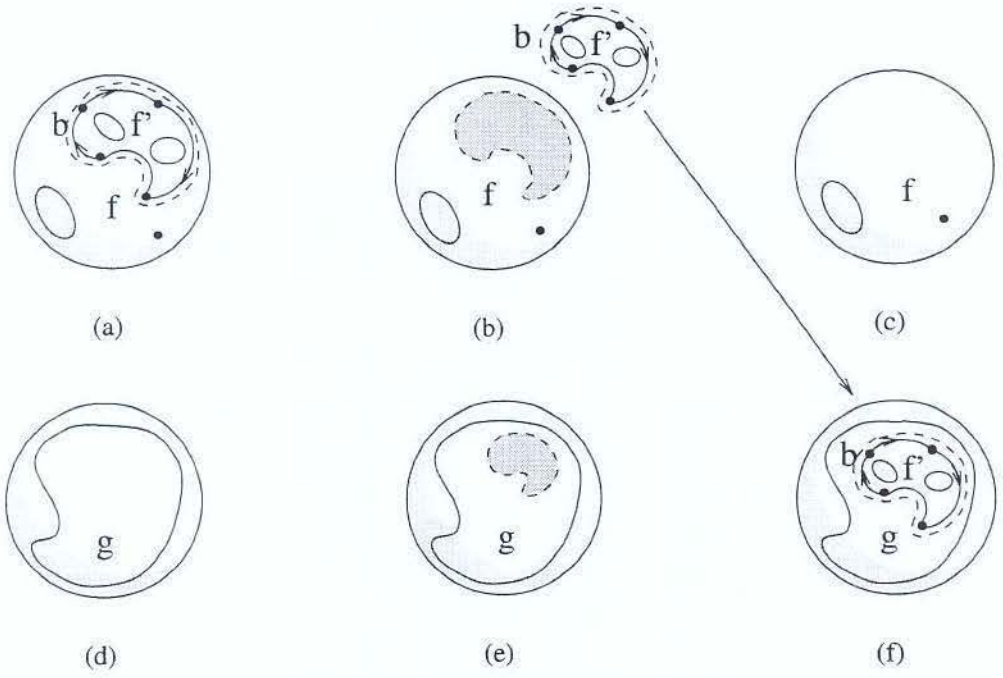


Figura 4.7: Operador topológico de transferência: Transfer(b, g)

Em particular, se a borda b é um vértice isolado ou uma aresta solta, a operação Transfer(b, g) corresponde a simplesmente remover este elemento da face f e inserí-lo na face g.

Na estrutura de dados SMC, a implementação da operação Transfer(b, g) consiste apenas em remover a borda b do conjunto *f.frontier* e incluí-la no conjunto *g.frontier*; e,

além disso, alterar o apontador $b.face$ para indicar a face g. Mais precisamente,

---

procedure Transfer(b : Border; g : Face);

$f \leftarrow b.face$;

$f.frontier \leftarrow f.frontier \setminus \{b\}$;

$g.frontier \leftarrow g.frontier \cup \{b\}$;

$b.face \leftarrow g$;

É importante notar que a borda b (a ser transferida) e a face g (que vai receber a borda b) podem ou não pertencer a um mesmo mapa. No caso em que a borda b e a face g pertencem a mapas distintos, a operação de transferência procede exatamente como mostrado na figura 4.7, e neste caso, podemos garantir que os mapas resultantes continuam sendo mapas esféricos. Por outro lado, se a borda b e a face g pertencem a um mesmo mapa esférico, a execução de Transfer(b, g) pode resultar num mapa cuja superfície não é homeomorfa à esfera. Isto ocorre quando a face g está na parte da esfera à direita da borda b. Por exemplo, veja a figura 4.8: após a operação Transfer(b, g), o mapa original fica dividido em dois mapas, sendo que um deles tem a topologia de um toro.

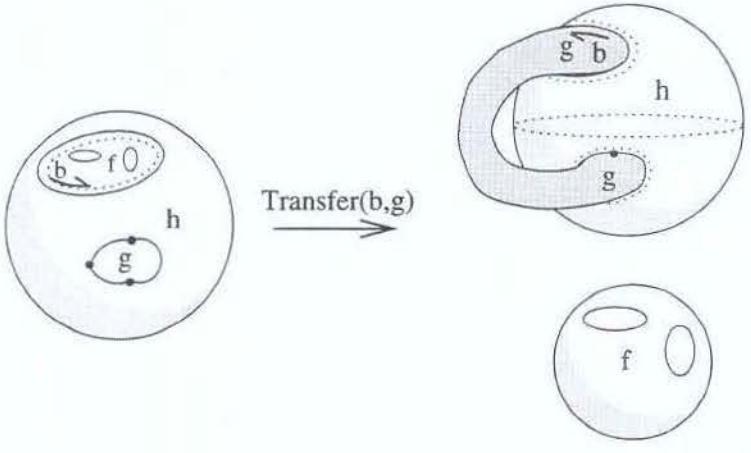


Figura 4.8: Resultado de Transfer(b, g) quando b e g são elementos de um mesmo mapa e a face g está na parte da esfera à direita da borda b.

Neste trabalho, a operação Transfer(b, g) sempre será aplicada a elementos (uma borda b e uma face g) pertencentes a mapas esféricos distintos, o que garante que os mapas resultantes são mapas esféricos.

Além disso, é importante ressaltar que Transfer é uma operação puramente topológica (combinatória) que não leva em consideração quaisquer propriedades geométricas dos elementos envolvidos. Portanto, cabe ao cliente desta operação garantir que a geometria da borda b é consistente com a sua nova posição na face g.

É fácil ver que esta operação pode ser realizada em tempo constante, desde que o conjunto de bordas de uma face seja representado através de uma estrutura de dados adequada à realização de operações de remoção, por exemplo, uma lista duplamente encadeada.

4.5 Operadores de conexão e desconexão

Vamos agora definir os operadores que nos permitem conectar e desconectar esquinas de um mapa $\mathcal{M}$, alterando os valores das funções *Onext* e *Lnext*. Isto é, dadas duas esquinas c_1 e c_2, os operadores de conexão e desconexão alteram as órbitas destas esquinas sob a função *Onext* — isto é, alteram os vértices $Org(c_1)$ e $Org(c_2)$ — e sob a função *Lnext* — isto é, alteram as bordas $Bord(c_1)$ e $Bord(c_2)$. Em cada caso, se as órbitas são distintas, elas são agrupadas dando origem a uma única (nova) órbita; caso contrário, se as esquinas estão numa mesma órbita, esta órbita é dividida em duas novas órbitas.

Considerando todas as possíveis combinações, temos quatro casos que serão tratados por operadores distintos:

(i) $Org(c_1) \neq Org(c_2)$ e $Bord(c_1) \neq Bord(c_2)$: JoinVerticesJoinBorders — (seção 4.5.2);

(ii) $Org(c_1) = Org(c_2)$ e $Bord(c_1) = Bord(c_2)$: SplitVertexSplitBorder — (seção 4.5.3);

(iii) $Org(c_1) \neq Org(c_2)$ e $Bord(c_1) = Bord(c_2)$: JoinVerticesSplitBorder — (seção 4.5.4);

(iv) $Org(c_1) = Org(c_2)$ e $Bord(c_1) \neq Bord(c_2)$: SplitVertexJoinBorders — (seção 4.5.5).

4.5.1 Junção e divisão de vértices e bordas

Antes de apresentarmos os operadores propriamente ditos, vamos inicialmente especificar a maneira como os ciclos de esquinas em torno dos vértices e em torno das bordas são agrupados ou quebrados.

Por definição, ao agrupar dois vértices distintos v_1 e v_2, dados respectivamente pelas esquinas c_1 e c_2 (isto é, $v_1 = Org(c_1)$ e $v_2 = Org(c_2)$), a órbita da esquina c_2 sob *Onext* é inserida imediatamente *após* a esquina c_1, e, por sua vez, a órbita da esquina c_1 sob *Onext* é inserida imediatamente *após* a esquina c_2. Veja por exemplo a figura 4.9. Reciprocamente, se $Org(c_1) = Org(c_2)$, a órbita de *Onext* em torno deste vértice é quebrada imediatamente *após* a esquina c_1 e imediatamente *após* a esquina c_2.

Analogamente, ao agrupar duas bordas distintas b_1 e b_2, dadas respectivamente pelas esquinas c_1 e c_2 (isto é, $b_1 = Bord(c_1)$ e $b_2 = Bord(c_2)$), as esquinas de $Bord(c_1)$ são inseridas imediatamente *antes* da esquina c_2, e as esquinas de $Bord(c_2)$ são inseridas imediatamente *antes* da esquina c_1. Isto é, a esquina que precedia c_1 (antes da operação) passa

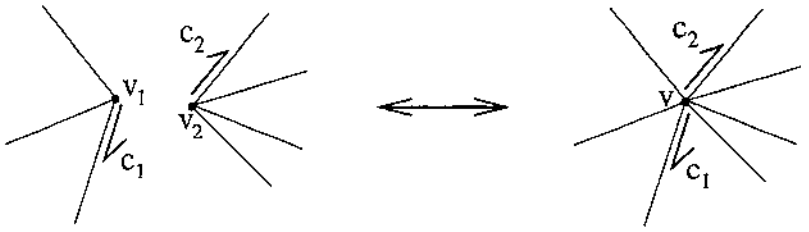


Figura 4.9: Junção e divisão de vértices

a preceder a esquina c_2 (depois da operação), e a esquina que precedia c_2 passa a preceder a esquina c_1. Veja por exemplo a figura 4.10. Reciprocamente, se $Bord(c_1) = Bord(c_2)$, esta borda é quebrada imediatamente *antes* das esquinas c_1 e c_2; mais precisamente, a esquina que precedia c_1 passa a preceder a esquina c_2 e vice-versa.

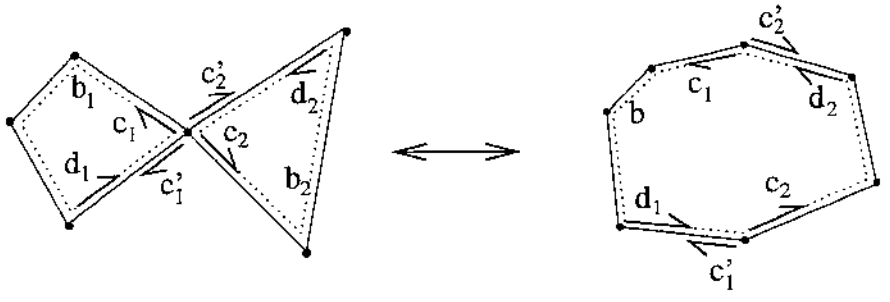


Figura 4.10: Junção e divisão de bordas

É fácil perceber que tanto a operação de junção como a operação de quebra das órbitas de *Onext* correspondem à uma mesma seqüência de atualizações nos campos *onext* dos objetos Corner envolvidos na operação. O mesmo vale para a junção e quebra das órbitas de *Lnext*. Mais precisamente, dadas duas esquinas c_1 e c_2, tanto no caso da junção como no caso da quebra dos ciclos de esquinas, os campos *onext* e *lnext* são atualizados conforme o seguinte procedimento:

---

procedure JoinOrSplitOrbits(c_1, c_2 : Corner)

$c'_1 \leftarrow c_1.onext$; $c'_2 \leftarrow c_2.onext$;

$d_1 \leftarrow c'_1.lnext.onext$; $d_2 \leftarrow c'_2.lnext.onext$;

$c_1.onext \leftarrow c'_2$; $c_2.onext \leftarrow c'_1$;

$d_1.lnext \leftarrow c_2$; $d_2.lnext \leftarrow c_1$;

4.5.2 O operador JoinVerticesJoinBorders

O operador JoinVerticesJoinBorders é utilizado quando $Org(c_1) \neq Org(c_2)$ e $Bord(c_1) \neq Bord(c_2)$. Neste caso, por simplicidade, vamos nos restringir a esquinas tais que $Left(c_1) = Left(c_2)$. Este operador junta os dois vértices e as duas bordas (isto é, junta as órbitas de *Onext* e de *Lnext*) dando origem a um novo vértice e a uma nova borda na fronteira da face $Left(c_1)$. Veja a figura 4.11. Com a restrição adotada, o operador JoinVerticesJoinBorders(c_1, c_2) não cria novas faces no mapa.

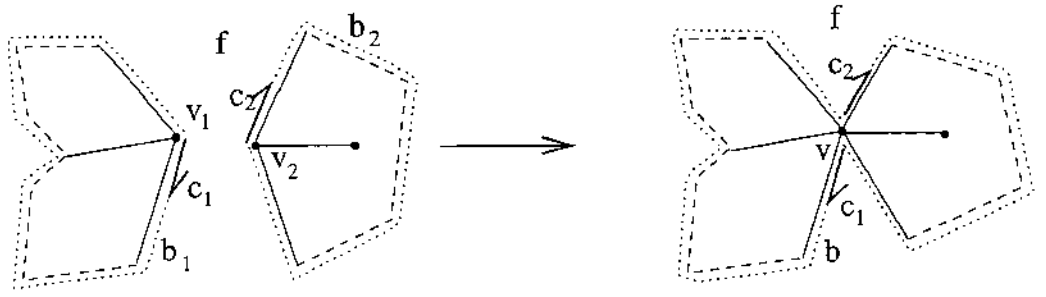


Figura 4.11: Operador JoinVerticesJoinBorders(c_1, c_2).

---

procedure JoinVerticesJoinBorders$(c_1, c_2$: Corner);
$f \leftarrow Left(c_1)$; $b_1 \leftarrow c_1.bord$; $b_2 \leftarrow c_2.bord$;
JoinOrSplitOrbits(c_1, c_2);
$f.frontier \leftarrow f.frontier \setminus \{b_1, b_2\}$;
$v \leftarrow$ **new**(Vertex);
$b \leftarrow$ **new**(Border);
$b.face \leftarrow f$; $b.desc \leftarrow c_1$;
$f.frontier \leftarrow f.frontier \cup \{b\}$;
$d \leftarrow c_1$; **repeat** $d.org \leftarrow v$; $d \leftarrow d.onext$ **until** $d = c_1$;
$d \leftarrow c_1$; **repeat** $d.bord \leftarrow b$; $d \leftarrow d.lnext$ **until** $d = c_1$;

4.5.3 O operador SplitVertexSplitBorder

O operador SplitVertexSplitBorder é utilizado quando $Org(c_1) = Org(c_2)$ e $Bord(c_1) = Bord(c_2)$ (o que implica que $Left(c_1) = Left(c_2)$). Este operador divide o vértice e a borda, dando origem a dois novos vértices e a duas novas bordas. Veja figura 4.12. Assim como no caso anterior, este operador também não cria novas faces no mapa.

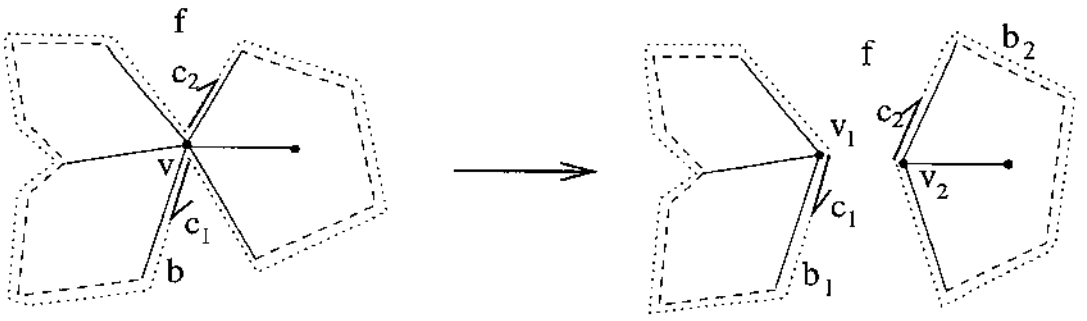


Figura 4.12: Operadores SplitVertexSplitBorder(c_1, c_2).

---

procedure SplitVertexSplitBorder(c_1, c_2 : Corner)

$f \leftarrow Left(c_1)$; $b \leftarrow c_1.bord$;

JoinOrSplitOrbits(c_1, c_2);

$f.frontier \leftarrow f.frontier \setminus \{b\}$;

$b_1 \leftarrow$ **new**(Border); $b_2 \leftarrow$ **new**(Border);

$b_1.face \leftarrow f$; $b_1.desc \leftarrow c_1$;

$b_2.face \leftarrow f$; $b_2.desc \leftarrow c_2$;

$f.frontier \leftarrow f.frontier \cup \{b_1, b_2\}$;

$v_1 \leftarrow$ **new**(Vertex);

$v_2 \leftarrow$ **new**(Vertex);

$d \leftarrow c_1$; **repeat** $d.org \leftarrow v_1$; $d \leftarrow d.onext$ **until** $d = c_1$;

$d \leftarrow c_2$; **repeat** $d.org \leftarrow v_2$; $d \leftarrow d.onext$ **until** $d = c_2$;

$d \leftarrow c_1$; **repeat** $d.bord \leftarrow b_1$; $d \leftarrow d.lnext$ **until** $d = c_1$;

$d \leftarrow c_2$; **repeat** $d.bord \leftarrow b_2$; $d \leftarrow d.lnext$ **until** $d = c_2$;

4.5.4 O operador JoinVerticesSplitBorder

O operador JoinVerticesSplitBorder é usado quando $Org(c_1) \neq Org(c_2)$ e $Bord(c_1) = Bord(c_2)$ (o que implica que $Left(c_1) = Left(c_2)$). Este operador junta os dois vértices $v_1 = Org(c_1)$ e $v_2 = Org(c_2)$ e divide a borda $b = Bord(c_1) = Bord(c_2)$, dando origem a um novo vértice v e a duas novas bordas b_1 e b_2. Por definição, as bordas b_1 e b_2 são inseridas nas fronteiras de duas faces f_1 e f_2 fornecidas como parâmetros. Além disso, a borda original b é retirada da face $f = Left(c_1) = Left(c_2)$, e esta face, que agora faz parte de um novo mapa $\mathcal{M}'$ disjunto do mapa $\mathcal{M}$, é retornada como um sub-produto da operação. Veja a figura 4.13.

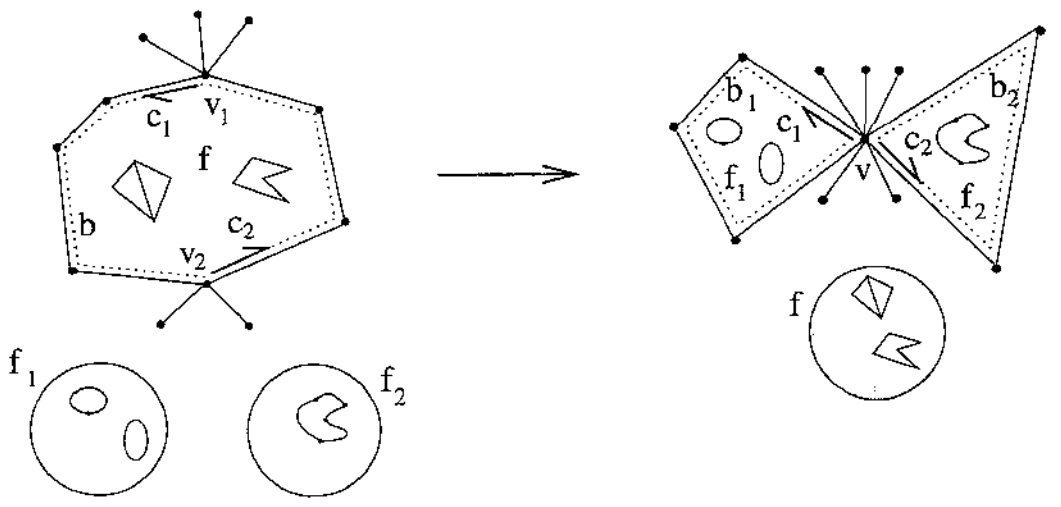


Figura 4.13: Operador JoinVerticesSplitBorder(c_1, c_2, f_1, f_2).

---

procedure JoinVerticesSplitBorder(c_1, c_2 : Corner; f_1, f_2 : Face) : Face;

$f \leftarrow Left(c_1)$; $b \leftarrow c_1.bord$;

JoinOrSplitOrbits(c_1, c_2);

$f.frontier \leftarrow f.frontier \setminus \{b\}$;

$b_1 \leftarrow$ **new**(Border); $b_2 \leftarrow$ **new**(Border);

$b_1.face \leftarrow f_1$; $b_1.desc \leftarrow c_1$; $f_1.frontier \leftarrow f_1.frontier \cup \{b_1\}$;

$b_2.face \leftarrow f_2$; $b_2.desc \leftarrow c_2$; $f_2.frontier \leftarrow f_2.frontier \cup \{b_2\}$;

$v \leftarrow$ **new**(Vertex);

$d \leftarrow c_1$; **repeat** $d.org \leftarrow v$; $d \leftarrow d.onext$ **until** $d = c_1$;

$d \leftarrow c_1$; **repeat** $d.bord \leftarrow b_1$; $d \leftarrow d.lnext$ **until** $d = c_1$;

$d \leftarrow c_2$; **repeat** $d.bord \leftarrow b_2$; $d \leftarrow d.lnext$ **until** $d = c_2$;

return f;

É importante observar que quaisquer outras bordas originalmente presentes nas faces f, f_1 e f_2 permanecem nestas faces após a operação. Além disso, para que o mapa $\mathcal{M}$, resultante desta operação, ainda seja um mapa esférico, as faces f_1 e f_2 (fornecidas como parâmetros) devem ser faces de mapas $\mathcal{M}_1$ e $\mathcal{M}_2$ disjuntos entre si, e ambos devem ser disjuntos do mapa $\mathcal{M}$.

4.5.5 O operador SplitVertexJoinBorders

O operador SplitVertexJoinBorders é utilizado quando $Org(c_1) = Org(c_2)$ e $Bord(c_1) \neq Bord(c_2)$. Em mapas esféricos, estas condições implicam que $Left(c_1) \neq Left(c_2)$. Este operador divide o vértice $v = Org(c_1) = Org(c_2)$ em dois novos vértices v_1 e v_2 e junta as duas bordas $b_1 = Bord(c_1)$ e $b_2 = Bord(c_2)$, dando origem a uma nova borda b. Por definição, as bordas b_1 e b_2 são retiradas das suas respectivas faces $f_1 = Left(c_1)$ e $f_2 = Left(c_2)$, e a nova borda b é inserida na fronteira de uma face f fornecida como parâmetro. Além disso, as faces f_1 e f_2 são retornadas como sub-produtos da operação, sendo que estas duas faces passam a constituir dois novos mapas $\mathcal{M}_1$ e $\mathcal{M}_2$, disjuntos entre si e disjuntos do mapa original $\mathcal{M}$. Veja figura 4.14.

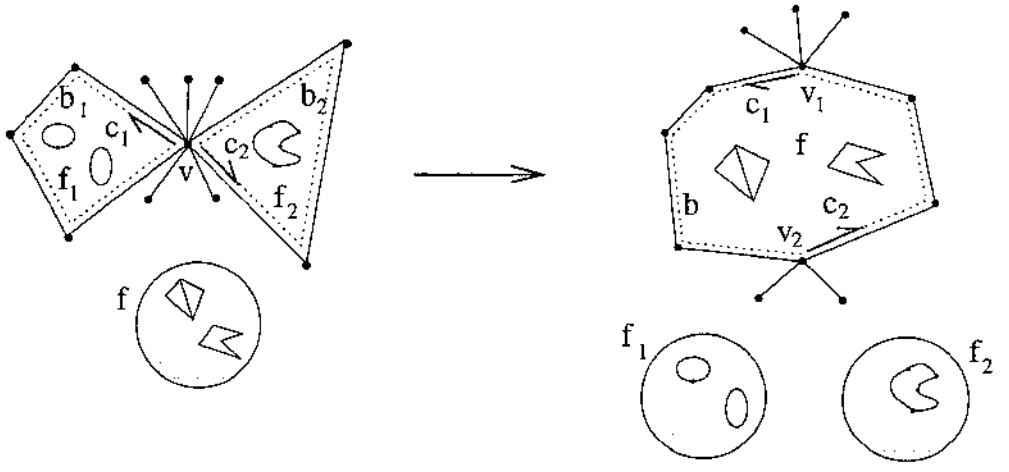


Figura 4.14: Operador SplitVertexJoinBorders(c_1, c_2, f).

---

procedure SplitVertexJoinBorders(c_1, c_2 : Corner; f : Face) : (Face,Face);

$f_1 \leftarrow Left(c_1)$; $f_2 \leftarrow Left(c_2)$;

$b_1 \leftarrow c_1.bord$; $b_2 \leftarrow c_2.bord$;

$f_1.frontier \leftarrow f_1.frontier \setminus \{b_1\}$; $f_2.frontier \leftarrow f_2.frontier \setminus \{b_2\}$;

JoinOrSplitOrbits(c_1, c_2);

$b \leftarrow$ **new**(Border); $b.face \leftarrow f$; $b.desc \leftarrow c_1$;

$f.frontier \leftarrow f.frontier \cup \{b\}$;

$v_1 \leftarrow$ **new**(Vertex); $v_2 \leftarrow$ **new**(Vertex);

$d \leftarrow c_1$; **repeat** $d.org \leftarrow v_1$; $d \leftarrow d.onext$ **until** $d = c_1$;

$d \leftarrow c_2$; **repeat** $d.org \leftarrow v_2$; $d \leftarrow d.onext$ **until** $d = c_2$;

$d \leftarrow c_1$; **repeat** $d.bord \leftarrow b$; $d \leftarrow d.lnext$ **until** $d = c_1$;

return (f_1, f_2);

De modo análogo ao operador JoinVerticesSplitBorder, quaisquer outras bordas originalmente presentes nas faces f, f_1 e f_2 permanecem nas suas respectivas faces após a operação. Além disso, para que o mapa $\mathcal{M}$ resultante desta operação, ainda seja um mapa esférico, a face f (fornecida como parâmetro) deve ser uma face de um mapa $\mathcal{M}'$ disjunto do mapa $\mathcal{M}$.

4.5.6 Complexidade dos operadores de conexão e desconexão

Conforme vimos nas seções anteriores, todos os quatro operadores de conexão e desconexão envolvem a criação de novos vértices e/ou de novas bordas. Por isso, em cada operador devemos percorrer todas as esquinas d das órbitas afetadas, atualizando os apontadores $d.org$ e $d.bord$. Portanto, o custo de cada operador é proporcional ao número total de esquinas nas órbitas $Onext$ e $Lnext$ das esquinas c_1 e c_2 envolvidas na operação.

4.5.7 Invertibilidade dos operadores de conexão e desconexão

Como podemos verificar facilmente, exceto pela criação de vértices e bordas, os quatro operadores de conexão e desconexão são todos inversíveis. Mais especificamente, os operadores JoinVerticesJoinBorders e SplitVertexSplitBorder são inversos entre si; e o mesmo ocorre com os operadores JoinVerticesSplitBorder e SplitVertexJoinBorders. Ou seja, dadas duas esquinas c_1 e c_2 num mapa $\mathcal{M}$ tais que $Org(c_1) \neq Org(c_2)$ e $Bord(c_1) \neq Bord(c_2)$, então a sequência de operações

$$\begin{aligned} &\mathsf{JoinVerticesJoinBorders}(c_1, c_2);\\ &\mathsf{SplitVertexSplitBorder}(c_1, c_2); \end{aligned}$$

não altera a topologia do mapa $\mathcal{M}$.

Analogamente, dadas duas esquinas c_1 e c_2 num mapa $\mathcal{M}$ tais que $Org(c_1) \neq Org(c_2)$ e $Bord(c_1) = Bord(c_2)$, então a sequência de operações

$$\begin{aligned} f_1 &\leftarrow \mathsf{MakeFace}();\\ f_2 &\leftarrow \mathsf{MakeFace}();\\ f &\leftarrow \mathsf{JoinVerticesSplitBorder}(c_1, c_2, f_1, f_2);\\ (f_1, f_2) &\leftarrow \mathsf{SplitVertexJoinBorders}(c_1, c_2, f); \end{aligned}$$

não altera a topologia do mapa $\mathcal{M}$.

4.6 Suficiência dos operadores topológicos

Observe que qualquer mapa esférico pode ser "desmontado" (isto é, reduzido a um conjunto de mapas elementares) por meio dos operadores de desconexão e transferência. Mais precisamente, o "desmonte" de um mapa pode ser realizado da seguinte forma: enquanto houver duas esquinas distintas com uma mesma origem sobre arestas-arco, isto é, enquanto houver esquinas c_1 e c_2 tais que $c_1 \neq c_2$ e $Org(c_1) = Org(c_2) \neq \varnothing$, utilizamos um dos operadores SplitVertexSplitBorder ou SplitVertexJoinBorders para quebrar o vértice em comum destas duas esquinas, separando-as. Ao fim deste processo, teremos um ou mais mapas nos quais toda borda tem exatamente um elemento: um vértice isolado ou uma aresta (arco ou oval). Utilizando o operador Transfer, podemos transferir cada uma destas bordas para um mapa trivial criado com o operador MakeFace, obtendo assim um conjunto de mapas elementares.

Não é difícil perceber que este processo de "desmonte" de um mapa pode ser invertido. Para isto, basta utilizar os operadores de criação (MakeFace, MakeVertex, MakeArcEdge, MakeOvalEdge) para obter os mapas elementares resultantes do "desmonte"; e, para cada operação de desconexão e transferência realizada, aplicar a respectiva operação inversa, na ordem reversa.

Isto nos permite concluir que o conjunto de operadores topológicos definidos nas seções anteriores é suficiente para construir qualquer mapa esférico.

4.7 Operadores auxiliares

Embora o conjunto de operadores topológicos definido anteriormente seja suficiente para manipular a topologia dos mapas esféricos, por questões de eficiência e simplicidade vamos definir dois operadores auxiliares denominados BreakEdge e Connect.

4.7.1 O operador BreakEdge

Dada uma esquina c tal que $Edge(c) \neq \varnothing$, o objetivo deste operador é quebrar a aresta $Edge(c)$, inserindo sobre ela um novo vértice. A figura 4.15 ilustra o resultado da aplicação deste operador.

É fácil ver que esta operação pode ser realizada utilizando-se os operadores de criação, transferência e conexão definidos anteriormente. Entretanto, isto exigiria a criação de diversos elementos (objetos na estrutura SMC) que seriam posteriormente descartados. Além disso, diversas operações de transferências teriam que ser realizadas para recolocar as demais bordas das faces adjacentes à aresta em questão de volta nas devidas faces. Portanto, para evitar estas operações desnecessárias, convém implementar o operador

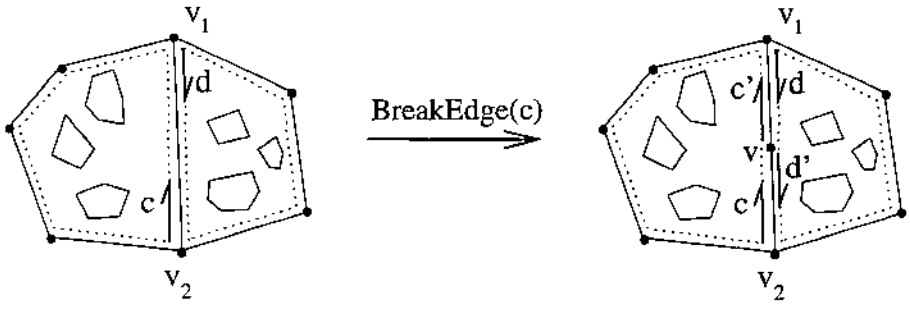


Figura 4.15: OperadorBreakEdge

BreakEdge(c) diretamente.

Este operador cria um novo objeto Vertex e (em geral), dois novos objetos do tipo Corner, e atualiza os apontadores afetados pela operação.

Um caso particular ocorre quando a aresta $Edge(c)$ é uma aresta oval; neste caso, não precisamos criar novos objetos do tipo Corner, basta atualizar o campo que armazena o vértice das esquinas c e $Sym(c)$. Mais precisamente,

---

procedure BreakEdge(c : Corner) : Corner;

$d \leftarrow Sym(c)$;

$v \leftarrow$ **new**(Vertex);

if $Org(c) = \varnothing$ **then** {$Edge(c)$ é uma aresta oval}

$c.org \leftarrow v$;	$d.org \leftarrow v$;

else

$c' \leftarrow$ **new**(Corner);	$d' \leftarrow$ **new**(Corner);
$c'.org \leftarrow v$;	$d'.org \leftarrow v$;
$c'.edge \leftarrow$ **new**(Edge);	$d'.edge \leftarrow$ **new**(Edge);
$c'.edge.props \leftarrow c.edge.props$;	$d'.edge.props \leftarrow d.edge.props$;
$c'.bord \leftarrow c.bord$;	$d'.bord \leftarrow d.bord$;
$c'.onext \leftarrow d'$	$d'.onext \leftarrow c'$;
$c'.lnext \leftarrow c.lnext$;	$d'.lnext \leftarrow d.lnext$;
$c.lnext \leftarrow c'$;	$d.lnext \leftarrow d'$;

return $c.lnext$;

É fácil perceber que esta operação pode ser executada em tempo constante.

4.7.2 O operador Connect

O operador Connect acrescenta uma nova aresta-arco num mapa, ligando dois vértices de uma mesma face. Mais precisamente, dadas duas esquinas c_1 e c_2 tais que $Left(c_1) = Left(c_2)$, $Org(c_1) \neq Org(c_2)$, $Org(c_1) \neq \varnothing$ e $Org(c_2) \neq \varnothing$, a operação Connect$(c_1, c_2)$ cria uma nova aresta ligando os vértices $Org(c_1)$ e $Org(c_2)$ através da face $f = Left(c_1) = Left(c_2)$. Veja figura 4.16.

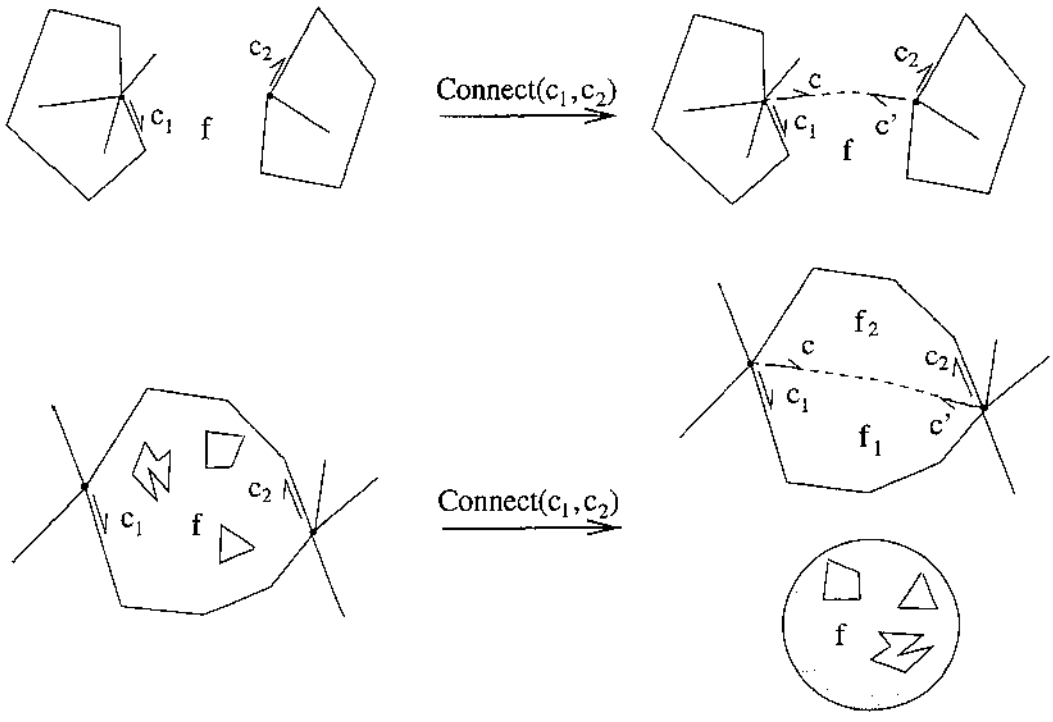


Figura 4.16: Operador Connect

As esquinas c_1 e c_2 podem pertencer a bordas distintas ou a uma mesma borda. No primeiro caso, este operador junta as bordas, que permanecem na fronteira da mesma face f original. No segundo caso, a face f é retirada do mapa (formando um novo mapa $\mathcal{M}'$) e é substituída por duas novas faces f_1 e f_2, separadas pela nova aresta. Na verdade, estas operações são efetuadas pelos operadores JoinVerticesJoinBorders e JoinVerticesSplitBorder

conforme o caso.

```
procedure Connect(c1, c2 : Corner)
    c ← MakeArcEdge();
    c' ← Sym(c);
    Transfer(Bord(c), f);
    JoinVerticesJoinBorders(c1, c);
    if Bord(c') ≠ Bord(c2) then
        JoinVerticesJoinBorders(c2, c');
    else
        f1 ← MakeFace();
        f2 ← MakeFace();
        f ← JoinVerticesSplitBorder(c2, c', f1, f2);
```

Capítulo 5

Geometria projetiva orientada

A partir de agora, vamos tratar das questões relacionadas à representação da geometria dos elementos de um mapa esférico. Para isso, vamos apresentar um breve resumo da geometria projetiva orientada, sendo que por questões de espaço, neste resumo vamos frequentemente recorrer a definições informais. As definições formais incluindo as demonstrações das propriedades aqui anunciadas podem ser encontradas no livro de Stolfi [33].

Como se sabe, a geometria de $\mathbb{R}^3$ fica bastante simplificada se tratarmos $\mathbb{R}^3$ como um subespaço de um *espaço projetivo* $\mathbb{P}^3$ [7, 33] o qual é composto pelos pontos ordinários de $\mathbb{R}^3$ mais os *pontos no infinito* onde, por definição, as retas e planos paralelos se interceptam. Esta extensão elimina muitos casos especiais que devem ser tratados isoladamente nos algoritmos geométricos e além disso, permite a unificação de muitos algoritmos que, no modelo cartesiano, parecem não ter a menor relação entre si.

Apesar destas vantagens, o modelo projetivo apresenta um inconveniente: muitos algoritmos da geometria cartesiana são baseados em operações que testam a posição relativa entre dois elementos, por exemplo, a posição de um ponto em relação a um plano, que consiste em verificar se um ponto pertence a um dos semi-espaços determinado por um plano. Entretanto, em geometria projetiva, este tipo teste não faz sentido, pois os dois semi-espaços determinados por um plano são conectados pelos pontos no infinito.

Para contornar este problema, a solução é considerar o *espaço projetivo orientado* $\mathbb{T}^3$, que consiste de duas cópias separadas de $\mathbb{R}^3$ mais os pontos no infinito. A geometria de $\mathbb{T}^3$, assim como a de $\mathbb{P}^3$, é definida exclusivamente pelas relações entre seus *elementos*: pontos, retas e planos. A vantagem deste modelo é que ele incorpora todas as conveniências de $\mathbb{P}^3$ sem perder as características de orientabilidade e separabilidade que são importantes para a geometria cartesiana.

5.1 Pontos

Um *ponto* p de $\mathbb{T}^3$ é, por definição, uma quádrupla não-nula de números $[p_0, p_1, p_2, p_3]$, *as coordenadas homogêneas* do ponto, sendo que $[p_0, p_1, p_2, p_3]$ e $[\lambda p_0, \lambda p_1, \lambda p_2, \lambda p_3]$, para todo $\lambda > 0$, representam o mesmo ponto. É importante ressaltar que $p = [p_0, p_1, p_2, p_3]$ e $q = [-p_0, -p_1, -p_2, -p_3]$ representam pontos distintos; dizemos que estes pontos são *antípodas* entre si, o que denotamos por $p = \neg q$.

Por definição, *as coordenadas cartesianas* de um ponto $p = [p_0, p_1, p_2, p_3]$ de $\mathbb{T}^3$, com $p_0 \neq 0$, são $(p_1/p_0, p_2/p_0, p_3/p_0)$. Neste caso, dizemos que p é um *ponto finito*; se $p_0 > 0$ dizemos que p é um ponto no *aquém* de $\mathbb{T}^3$ e caso contrário, se $p_0 < 0$ dizemos que p é um ponto no *além*. Note que p e $\neg p$ têm as mesmas coordenadas cartesianas, mas são pontos distintos de $\mathbb{T}^3$. Reciprocamente, para cada ponto com coordenadas cartesianas (x, y, z), há dois pontos distintos de $\mathbb{T}^3$, a saber $[1, x, y, z]$ e $[-1, -x, -y, -z]$. Desta forma, o além e o aquém de $\mathbb{T}^3$ podem ser vistos como duas cópias de $\mathbb{R}^3$ contidas em $\mathbb{T}^3$.

Note que a origem do aquém é $O = [1, 0, 0, 0]$, e a origem do além é o ponto $\neg O = [-1, 0, 0, 0]$. Observe que em geral o ponto $\neg p$ é diferente do ponto $\overline{p} = [p_0, -p_1, -p_2, -p_3]$, o ponto simétrico a p em relação à origem O. Na verdade, as únicas exceções são os pontos no infinito.

A topologia de $\mathbb{T}^3$ é definida de tal forma que uma função $f(t) = [p_0(t), p_1(t), p_2(t), p_3(t)]$ de $\mathbb{R}$ para $\mathbb{T}^3$ é contínua se $p_0(t)$, $p_1(t)$, $p_2(t)$ e $p_3(t)$ são funções reais contínuas que não são simultaneamente nulas para nenhum valor de t.

Podemos com isso dizer que um ponto de $\mathbb{T}^3$ da forma $[0, x, y, z]$ é um *ponto no infinito*, na direção do vetor cartesiano (x, y, z) quando visto de qualquer outro ponto no aquém. Por outro lado, visto de um ponto no além, dizemos que o mesmo ponto está no infinito na direção $(-x, -y, -z)$.

5.2 Planos

Um *plano* α em $\mathbb{T}^3$ é, por definição, uma quádrupla não-nula de números $\langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$, os *coeficientes homogêneos* do plano, sendo que $\langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ e $\langle \lambda\alpha_0, \lambda\alpha_1, \lambda\alpha_2, \lambda\alpha_3 \rangle$, para todo $\lambda > 0$, representam o mesmo plano. É importante ressaltar que $\langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ e $\alpha' = \langle -\alpha_0, -\alpha_1, -\alpha_2, -\alpha_3 \rangle$ são planos distintos; dizemos que estes planos são *opostos* entre si, o que denotamos por $\alpha = \neg\alpha'$.

Por definição, o plano α é *incidente* a todo ponto $[p_0, p_1, p_2, p_3]$ de $\mathbb{T}^3$ tal que

$$\alpha_0 p_0 + \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 = 0$$

Dizemos que dois planos são *coincidentes* se eles incidem sobre o mesmo conjunto de pontos. Portanto, dois planos são coincidentes se eles são iguais ou opostos entre si.

Em geral, o conjunto de pontos que são incidentes a um plano consiste de duas cópias de um mesmo plano euclidiano: uma cópia no aquém e outra no além de $\mathbb{T}^3$, mais um conjunto de pontos no infinito nas direções paralelas ao plano. As exceções são os planos $\Omega_2 = (1,0,0,0)$ e o seu oposto $\neg\Omega_2$, denominados *planos no infinito*, que são incidentes a todos os pontos no infinito e a apenas estes pontos.

Todo plano α divide $\mathbb{T}^3$ em dois semi-espaços: o *lado positivo* e o *lado negativo* de α. Por definição, o lado positivo consiste de todos os pontos $[p_0, p_1, p_2, p_3]$ de $\mathbb{T}^3$ tais que $\alpha_0 p_0 + \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 > 0$. A *posição* de um ponto p em relação a α, denotada por $p \diamond \alpha$, é dada por

$$p \diamond \alpha = sign(\alpha_0 p_0 + \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3)$$

Note que $p \diamond \alpha = +1$ se e somente se $(\neg p) \diamond \alpha = -1$. Portanto, o lado positivo de α consiste de um semi-espaço cartesiano no aquém e do semi-espaço complementar no além (mais um subconjunto dos pontos no infinito).

A *orientação externa* de um plano, definida por '$\diamond$', pode ser visualizada como uma seta direcionada do semi-espaço negativo para o positivo. Um plano α também tem uma *orientação interna*, a qual pode ser visualizada através de uma pequena seta circular desenhada sobre o plano. Veja figura figure 5.1. Deslocando esta seta ao longo do plano

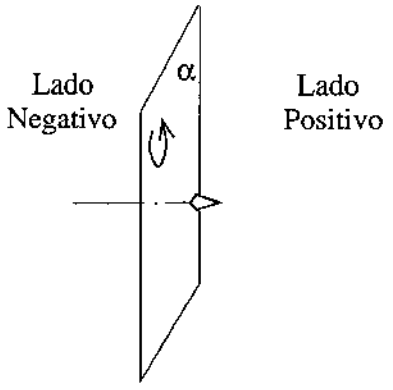


Figura 5.1: Orientações (externa e interna) de um plano.

podemos determinar se o sentido de rotação em torno de um ponto do plano é *positivo* (concordando com o arco) ou *negativo* (em sentido contrário ao arco).

As orientações interna e externa de um plano em $\mathbb{T}^3$ não são independentes. Por definição, o lado positivo do plano é aquele de onde, no aquém, "vemos" o arco circular orientado em sentido anti-horário, supondo os eixos cartesianos nas posições convencionais.

É importante observar que as orientações externas de um mesmo plano, em dois pontos antípodas, têm sentidos opostos; e o mesmo se aplica às orientações internas.

5.3 Retas

Informalmente, uma *reta finita* de $\mathbb{T}^3$ consiste de duas cópias de uma mesma reta cartesiana de $\mathbb{R}^3$: uma cópia no aquém e outra no além, mais os dois pontos no infinito nas direções paralelas a estas retas. Uma *reta no infinito* consiste de todos os pontos no infinito nas direções paralelas a um mesmo plano.

Por definição, em $\mathbb{T}^3$ uma reta tem uma orientação *externa* e uma *interna*. A orientação *interna* define o sentido positivo de percurso ao longo da reta, e a orientação *externa* define o sentido positivo de rotação em torno da reta. Estas orientações podem ser respectivamente visualizadas como uma seta colocada sobre a reta e como um arco circular contornando a reta. Veja figura 5.2.

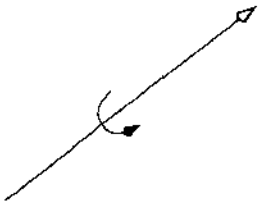


Figura 5.2: Orientações (externa e interna) de uma reta.

Como no caso dos planos, as orientações interna e externa de uma reta em $\mathbb{T}^3$ não são independentes. Por definição, elas satisfazem a regra da mão direita, supondo os eixos cartesianos nas posições convencionais.

Dizemos que duas retas l e m são *opostas* entre si, se elas são a mesma reta euclidiana com orientações interna e externa opostas; o que denotamos por $l = \neg m$. Além disso, dizemos que duas retas são *coincidentes* se elas incidem sobre o mesmo conjunto de pontos, isto é, se elas são iguais ou opostas entre si.

5.4 Operações *junção* e *encontro*

As operações básicas da geometria projetiva orientada são *junção* e *encontro*, denotadas respectivamente por $\vee$ e $\wedge$. Em $\mathbb{T}^3$, a operação *junção* devolve a reta $p \vee q$ conectando os dois pontos p e q, ou o plano $p \vee l$ que contém o ponto p e a reta l. Por outro lado, a operação *encontro* retorna a reta $\alpha \wedge \beta$ comum aos dois planos α e β, ou o ponto $l \wedge \alpha$ onde a reta l intercepta o plano α. Em todos estes casos, as orientações dos objetos resultantes destas operações ficam precisamente estabelececidas a partir das orientações dos operandos. Por exemplo, na figura 5.3 mostramos a parte no aquém dos elementos resultantes das operações: $p \vee q$, $p \vee l$, $\alpha \wedge \beta$ e $l \wedge \alpha$.

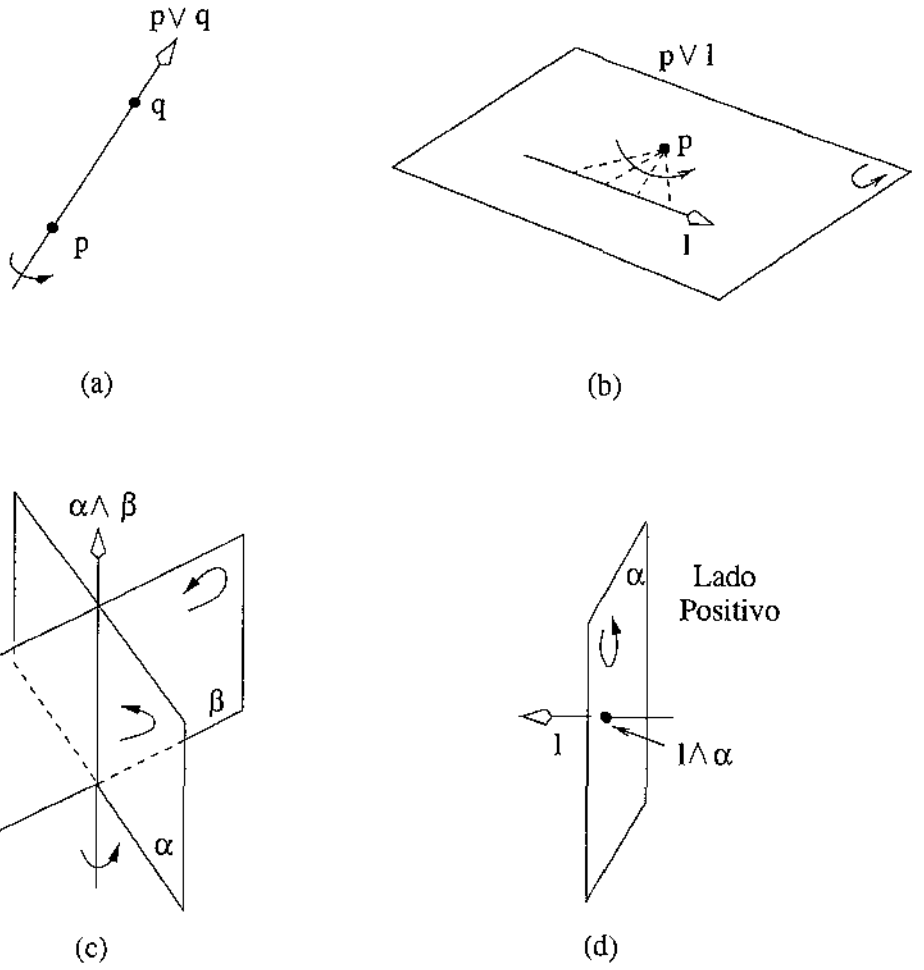


Figura 5.3: Operações *junção* e *encontro* (parte no aquém): (a) $p \vee q$; (b) $p \vee l$; (c) $\alpha \wedge \beta$; (d) $l \wedge \alpha$.

A orientação da reta $l = p \vee q$ é definida como sendo a direção de p para q que segue pelo menor segmento que une estes dois pontos. Veja figura 5.3(a). No caso do plano $p \vee l$, a orientação deste plano é definida de tal forma que l gira em torno de p no sentido positivo; mais precisamente, movendo um ponto q sobre l no sentido positivo, o sentido de rotação da reta $p \vee q$ em torno do ponto p define a orientação do plano $p \vee l$. Veja figura 5.3(b).

Para visualizar a orientação da reta $l = \alpha \wedge \beta$, podemos imaginar o plano α rodando em torno da reta euclidiana $\alpha \cap \beta$, em direção ao plano β, pelo menor ângulo que faz a posição e a orientação destes planos coincidirem. O sentido desta rotação define a orientação externa da reta l; e a orientação interna fica então determinada pela regra da mão direita. Veja figura 5.3(c). Note que $\alpha \wedge \beta = \neg(\beta \wedge \alpha)$; isto é, em $\mathbb{T}^3$ a interseção de dois planos é anti-comutativa.

Finalmente, por definição, $l \wedge \alpha$ é o ponto onde l sai do lado positivo de α. Veja figura 5.3(d). Por consistência, é conveniente definir $l \wedge \alpha = \alpha \wedge l$; isto é, a interseção entre uma reta e um plano é comutativa.

Como podemos verificar, para quaisquer a e b em $\mathbb{T}^3$ temos que

$$\begin{aligned}(\neg a) \vee b &= a \vee (\neg b) = \neg(a \vee b)\\ (\neg a) \wedge b &= a \wedge (\neg b) = \neg(a \wedge b)\end{aligned}$$

Com estas convenções, $\vee$ e $\wedge$ são operações associativas. Portanto, o plano determinado por três pontos não colineares p, q e r é dado por

$$p \vee q \vee r = (p \vee q) \vee r = p \vee (q \vee r)$$

Analogamente, o ponto de interseção de três planos α, β e γ, que não têm uma reta em comum, é dado por

$$\alpha \wedge \beta \wedge \gamma = (\alpha \wedge \beta) \wedge \gamma = \alpha \wedge (\beta \wedge \gamma)$$

5.5 Fórmulas algébricas

Dados três pontos não colineares $p = [p_0, p_1, p_2, p_3]$, $q = [q_0, q_1, q_2, q_3]$ e $r = [r_0, r_1, r_2, r_3]$, os coeficientes do plano determinado por estes três pontos são

$$p \vee q \vee r = \left\langle -\begin{vmatrix} p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \\ r_1 & r_2 & r_3 \end{vmatrix}, \begin{vmatrix} p_0 & p_2 & p_3 \\ q_0 & q_2 & q_3 \\ r_0 & r_2 & r_3 \end{vmatrix}, -\begin{vmatrix} p_0 & p_1 & p_3 \\ q_0 & q_1 & q_3 \\ r_0 & r_1 & r_3 \end{vmatrix}, \begin{vmatrix} p_0 & p_1 & p_2 \\ q_0 & q_1 & q_2 \\ r_0 & r_1 & r_2 \end{vmatrix} \right\rangle \tag{5.1}$$

As coordenadas homogêneas do ponto de interseção entre três planos $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$, $\beta = \langle \beta_0, \beta_1, \beta_2, \beta_3 \rangle$ e $\gamma = \langle \gamma_0, \gamma_1, \gamma_2, \gamma_3 \rangle$, que não têm uma reta em comum, são

$$\alpha \wedge \beta \wedge \gamma = \left[\begin{vmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_2 & \beta_3 \\ \gamma_1 & \gamma_2 & \gamma_3 \end{vmatrix}, -\begin{vmatrix} \alpha_0 & \alpha_2 & \alpha_3 \\ \beta_0 & \beta_2 & \beta_3 \\ \gamma_0 & \gamma_2 & \gamma_3 \end{vmatrix}, \begin{vmatrix} \alpha_0 & \alpha_1 & \alpha_3 \\ \beta_0 & \beta_1 & \beta_3 \\ \gamma_0 & \gamma_1 & \gamma_3 \end{vmatrix}, -\begin{vmatrix} \alpha_0 & \alpha_1 & \alpha_2 \\ \beta_0 & \beta_1 & \beta_2 \\ \gamma_0 & \gamma_1 & \gamma_2 \end{vmatrix} \right] \tag{5.2}$$

5.6 Coeficientes de Plücker de uma reta

Para estabelecer as fórmulas algébricas das operações $\vee$ e $\wedge$ envolvendo retas, precisamos primeiro estabelecer como uma reta de $\mathbb{T}^3$ pode ser algebricamente representada.

Dados dois planos $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ e $\beta = \langle \beta_0, \beta_1, \beta_2, \beta_3 \rangle$, a reta $l = \alpha \wedge \beta$ pode ser precisamente representada pelos seis coeficientes $\langle l_{01}, l_{02}, l_{12}, l_{03}, l_{13}, l_{23} \rangle$ onde

$$l_{ij} = \det \begin{vmatrix} \alpha_i & \alpha_j \\ \beta_i & \beta_j \end{vmatrix} = \alpha_i \beta_j - \alpha_j \beta_i$$

Estes seis números são denominados os *coeficientes de Plücker* [33, 9] da reta l. Por simplicidade, vamos escrever $l = \langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle$, com os determinantes implicitamente ordenados conforme definido acima.

Os coeficientes de Plücker de uma reta não são independentes: uma sêxtupla $\langle l_0, .. \, l_5 \rangle$ representa uma reta de $\mathbb{R}^3$ se e somente se

$$l_0 l_5 - l_1 l_4 + l_2 l_3 = 0 \tag{5.3}$$

Além disso, estes coeficientes são homogêneos, isto é, para todo $\lambda > 0$, $\langle l_0, .. \, l_5 \rangle$ e $\langle \lambda l_0, .. \, \lambda l_5 \rangle$ representam a mesma reta. Observe que $\langle -l_0, .. \, -l_5 \rangle$ é a reta oposta a $\langle l_0, .. \, l_5 \rangle$.

Dados dois pontos $p = [p_0, .. \, p_3]$ e $q = [q_0, .. \, q_3]$, dois planos $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ e $\beta = \langle \beta_0, \beta_1, \beta_2, \beta_3 \rangle$ e uma reta $l = \langle l_0, .. \, l_5 \rangle$, pode-se demonstrar que

$$\begin{aligned}
\alpha \wedge \beta &= \left\langle \begin{vmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{vmatrix}, \begin{vmatrix} \alpha_0 & \alpha_2 \\ \beta_0 & \beta_2 \end{vmatrix}, \begin{vmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{vmatrix}, \begin{vmatrix} \alpha_0 & \alpha_3 \\ \beta_0 & \beta_3 \end{vmatrix}, \begin{vmatrix} \alpha_1 & \alpha_3 \\ \beta_1 & \beta_3 \end{vmatrix}, \begin{vmatrix} \alpha_2 & \alpha_3 \\ \beta_2 & \beta_3 \end{vmatrix} \right\rangle \\
&= \langle \alpha_0\beta_1 - \alpha_1\beta_0, \alpha_0\beta_2 - \alpha_2\beta_0, \alpha_1\beta_2 - \alpha_2\beta_1, \alpha_0\beta_3 - \alpha_3\beta_0, \alpha_1\beta_3 - \alpha_3\beta_1, \alpha_2\beta_3 - \alpha_3\beta_2 \rangle \\
l \wedge \alpha &= [-l_2\alpha_3 + l_4\alpha_2 - l_5\alpha_1, l_1\alpha_3 - l_3\alpha_2 + l_5\alpha_0, -l_0\alpha_3 + l_3\alpha_1 - l_4\alpha_0, l_0\alpha_2 - l_1\alpha_1 + l_2\alpha_0] \\
p \vee l &= \langle l_0 p_1 + l_1 p_2 + l_3 p_3, -l_0 p_0 + l_2 p_2 + l_4 p_3, -l_1 p_0 - l_2 p_1 + l_5 p_3, -l_3 p_0 - l_4 p_1 - l_5 p_2 \rangle \\
p \vee q &= \left\langle \begin{vmatrix} p_2 & p_3 \\ q_2 & q_3 \end{vmatrix}, -\begin{vmatrix} p_1 & p_3 \\ q_1 & q_3 \end{vmatrix}, \begin{vmatrix} p_0 & p_3 \\ q_0 & q_3 \end{vmatrix}, \begin{vmatrix} p_1 & p_2 \\ q_1 & q_2 \end{vmatrix}, -\begin{vmatrix} p_0 & p_2 \\ q_0 & q_2 \end{vmatrix}, \begin{vmatrix} p_0 & p_1 \\ q_0 & q_1 \end{vmatrix} \right\rangle \\
&= \langle p_2 q_3 - p_3 q_2, p_3 q_1 - p_1 q_3, p_0 q_3 - p_3 q_0, p_1 q_2 - p_2 q_1, p_2 q_0 - p_0 q_2, p_0 q_1 - p_1 q_0 \rangle
\end{aligned} \tag{5.4}$$

É importante ressaltar que as fórmulas (5.1), (5.2) e (5.4) retornam a n-túpla nula $\langle 0, .. \, 0 \rangle = \langle \mathbf{0} \rangle$ quando a operação não está definida. Por exemplo, se α e β são coincidentes então $\alpha \wedge \beta = \langle 0, 0, 0, 0, 0, 0 \rangle = \langle \mathbf{0} \rangle$. Analogamente, se l é uma reta no plano α então $l \wedge \alpha = [0, 0, 0, 0] = [\mathbf{0}]$.

A partir destas fórmulas, é fácil mostrar que a distância euclidiana entre o plano α e a origem O, denotada por $dist(\alpha, O)$, e a distância euclidiana entre a reta l e a origem, denotada por $dist(l, O)$, são dadas por

$$\begin{aligned}
dist(\alpha, O) &= \sqrt{\frac{\alpha_0^2}{\alpha_1^2 + \alpha_2^2 + \alpha_3^2}} \\
dist(l, O) &= \sqrt{\frac{l_0^2 + l_1^2 + l_3^2}{l_2^2 + l_4^2 + l_5^2}}
\end{aligned} \tag{5.5}$$

Daí, podemos concluir que o plano α passa pela origem O se e somente se $\alpha_0 = 0$, e a reta l passa por O se e somente se $l_0 = l_1 = l_3 = 0$.

A direção de uma reta finita l é o ponto no infinito onde a reta passa do aquém para o além, isto é, o ponto $l \wedge \Omega_2$ que é dado por

$$dir(l) = [0, l_5, -l_4, l_2] \tag{5.6}$$

O *plano bissetor* da reta l, denotado por $bisect(l)$, é o plano perpendicular a l que passa pela origem e está orientado de tal forma que $dir(l)$ fica do seu lado positivo; mais precisamente,

$$bisect(l) = \langle 0, l_5, -l_4, l_2 \rangle \tag{5.7}$$

Utilizando as fórmulas (5.4), podemos demonstrar que duas retas $l = \langle l_0, .. l_5 \rangle$ e $m = \langle m_0, .. m_5 \rangle$ se interceptam se e somente se

$$l_0 m_5 - l_1 m_4 + l_2 m_3 + l_3 m_2 - l_4 m_1 + l_5 m_0 = 0 \tag{5.8}$$

Neste caso, se as retas não são paralelas — isto é, se l e m se interceptam num ponto finito — então as coordenadas deste ponto de interseção podem ser determinadas pela fórmula

$$\begin{aligned} l \cap m = [\ & \kappa_{24}^2 + \kappa_{25}^2 + \kappa_{45}^2, \quad (l_3 m_2 - l_1 m_4 + l_5 m_0)\kappa_{24} - \kappa_{15}\kappa_{25} - \kappa_{35}\kappa_{45}, \\ & (l_0 m_5 + l_3 m_2 - l_4 m_1)\kappa_{25} + \kappa_{04}\kappa_{24} + \kappa_{34}\kappa_{45}, \\ & (l_0 m_5 - l_1 m_4 + l_2 m_3)\kappa_{45} - \kappa_{02}\kappa_{24} - \kappa_{12}\kappa_{25}\ ] \end{aligned} \tag{5.9}$$

onde

$$\kappa_{ij} = \begin{vmatrix} l_i & l_j \\ m_i & m_j \end{vmatrix}$$

Visto que a interseção entre duas retas nem sempre é definida, não utilizaremos a notação $l \wedge m$ para esta operação.

5.7 Orientações relativas

5.7.1 Posição relativa de dois pontos em relação a uma reta

Dados dois pontos $p = [p_0, .. p_3]$ e $q = [q_0, .. q_3]$ numa reta $l = \langle l_0, .. l_5 \rangle$, dizemos que o par p, q está *positivamente orientado em relação à reta* l se $p \vee q = l$; por outro lado, se $p \vee q = \neg l$, dizemos que o par p, q está *negativamente orientado em relação à reta* l. Em particular, se $p \vee q = \langle 0 \rangle$ (isto é, se $p = q$ ou $p = \neg q$), a orientação deste par de pontos em relação a uma reta é indefinida.

Este conceito é formalizado pelo predicado

$$\otimes_l(p,q) = \begin{cases} +1 & \text{se } p \vee q = l \\ 0 & \text{se } p \vee q = [\mathbf{0}] \\ -1 & \text{se } p \vee q = \neg l \end{cases}$$

5.7.2 Posição relativa de dois planos em relação a uma reta

A relação dual à posição relativa de dois pontos numa reta é a relação que descreve a posição relativa de dois planos $\alpha = \langle \alpha_0, .. \alpha_3 \rangle$ e $\beta = \langle \beta_0, .. \beta_3 \rangle$ passando por uma mesma reta $l = \langle l_0, .. l_5 \rangle$. Dizemos que o par α, β está *positivamente orientado em relação à reta* l se $\alpha \wedge \beta = l$; por outro lado, se $\alpha \wedge \beta = \neg l$, dizemos que o par α, β está *negativamente orientado em relação à reta* l. Em particular, se $\alpha \wedge \beta = \langle \mathbf{0} \rangle$ (isto é, se α e β são coincidentes), a orientação deste par de planos em relação a uma reta é indefinida.

Este conceito é formalizado pelo predicado

$$\otimes_l(\alpha,\beta) = \begin{cases} +1 & \text{se } \alpha \wedge \beta = l \\ 0 & \text{se } \alpha \wedge \beta = \langle \mathbf{0} \rangle \\ -1 & \text{se } \alpha \wedge \beta = \neg l \end{cases}$$

5.7.3 Posição relativa de três pontos num plano

Dados três pontos não colineares $p = [p_0, .. p_3]$, $q = [q_0, .. q_3]$ e $r = [r_0, .. r_3]$ sobre um plano $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$, dizemos que o triângulo pqr está *positivamente orientado em relação ao plano* α (denotado por $\Delta(p,q,r,\alpha) = +1$) se o plano $p \vee q \vee r$ coincide com α; caso contrário, se $p \vee q \vee r$ coincide com $\neg\alpha$, dizemos que o triângulo está *negativamente orientado em relação a* α (denotado por $\Delta(p,q,r,\alpha) = -1$). Caso o triângulo seja degenerado (isto é, se p, q e r são colineares), escrevemos $\Delta(p,q,r,\alpha) = 0$.

Como podemos verificar,

$$\Delta(p,q,r,\alpha) = sign \begin{vmatrix} p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 \end{vmatrix}$$

5.7.4 Posição relativa de duas retas

Dadas duas retas l e m, sejam p e q dois pontos sobre l e sejam r e s dois pontos sobre m tais que $p \vee q = l$ e $r \vee s = m$. A *orientação relativa* das retas l e m é definida por

$$l \diamond m = sign \begin{vmatrix} p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ s_0 & s_1 & s_2 & s_3 \end{vmatrix} \tag{5.10}$$

sendo que, se $l \diamond m = +1$ (resp. $l \diamond m = -1$), dizemos que l e m são positivamente (resp. negativamente) orientadas.

Verifica-se que a fórmula (5.10) não depende da escolha dos pontos p, q, r e s, mas apenas das orientações das retas.

Intuitivamente, $l \diamond m = +1$ se e somente se l "contorna" m (e m "contorna" l) de acordo com a regra da mão direita. Veja figura 5.4.

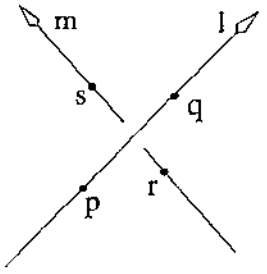


Figura 5.4: Retas positivamente orientadas.

5.8 Ordenação cíclica

Baseados nos predicados $\otimes_l(p, q)$ e $\otimes_l(\alpha, \beta)$ que definem respectivamente a posição de dois pontos em relação a uma reta e a posição de dois planos em relação a uma reta, vamos agora definir dois predicados que definem a ordenação cíclica de três pontos sobre uma reta e de três planos em torno de uma reta.

5.8.1 Ordenação cíclica de três pontos em relação a uma reta

Dados três pontos p, q e r sobre uma mesma reta l, vamos definir o predicado $\otimes_l(p, q, r)$ que retorna $+1$ se, considerando a orientação interna da reta l, estes pontos ocorrem nesta ordem (cíclica) ao longo de l; retorna -1 se eles ocorrem em ordem oposta; e retorna 0 se pelo menos dois desses pontos são iguais. Veja figura 5.5.

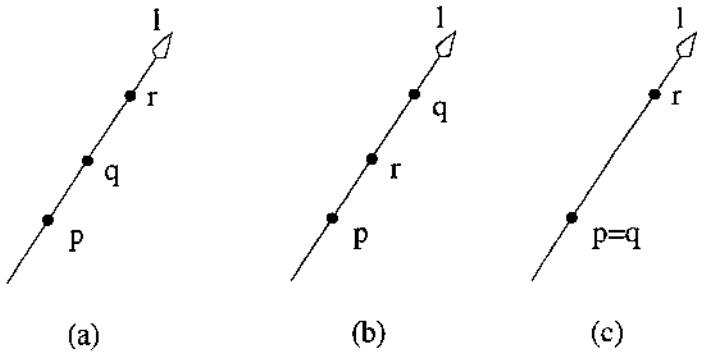


Figura 5.5: Ordem cíclica de três pontos ao longo de uma reta: (a) $\circledast_l(p,q,r) = +1$; (b) $\circledast_l(p,q,r) = -1$; (c) $\circledast_l(p,q,r) = 0$.

Como podemos verificar,

$$\circledast_l(p,q,r) = sign(\otimes_l(p,q) + \otimes_l(q,r) + \otimes_l(r,p))$$

ou seja, $\circledast_l(p,q,r)$ é determinado pela "maioria" dos sinais de $\otimes_l(p,q)$, $\otimes_l(q,r)$ e $\otimes_l(r,p)$.

5.8.2 Ordenação cíclica de três planos em torno de uma reta

A operação dual à ordenação de três pontos ao longo de uma reta é a ordenação cíclica de três planos em torno de uma reta comum a estes três planos. Mais precisamente, dados três planos α, β e γ e uma reta l comum a estes planos, vamos definir o predicado $\circledast_l(\alpha,\beta,\gamma)$ que retorna $+1$ se, considerando a orientação externa da reta l, estes planos ocorrem nesta ordem (cíclica) em torno de l; retorna -1 se eles ocorrem em ordem oposta; e retorna 0 se pelo menos dois desses planos são coincidentes. Veja figura 5.6.

De modo análogo ao predicado anterior, podemos verificar que

$$\circledast_l(\alpha,\beta,\gamma) = sign(\otimes_l(\alpha,\beta) + \otimes_l(\beta,\gamma) + \otimes_l(\gamma,\alpha))$$

5.9 A esfera $\mathbb{S}^2$

Em $\mathbb{T}^3$, a *esfera unitária* $\mathbb{S}^2$ é o conjunto de pontos do aquém à distância 1 da origem O, isto é,

$$\mathbb{S}^2 = \{[p_0,p_1,p_2,p_3] \mid p_1^2 + p_2^2 + p_3^2 - p_0^2 = 0 \text{ e } p_0 > 0\}$$

A posição de um ponto $p = [p_0,p_1,p_2,p_3]$ de $\mathbb{T}^3$ em relação a $\mathbb{S}^2$ é definida por

$$p \circ \mathbb{S}^2 = sign(p_1^2 + p_2^2 + p_3^2 - p_0^2)$$

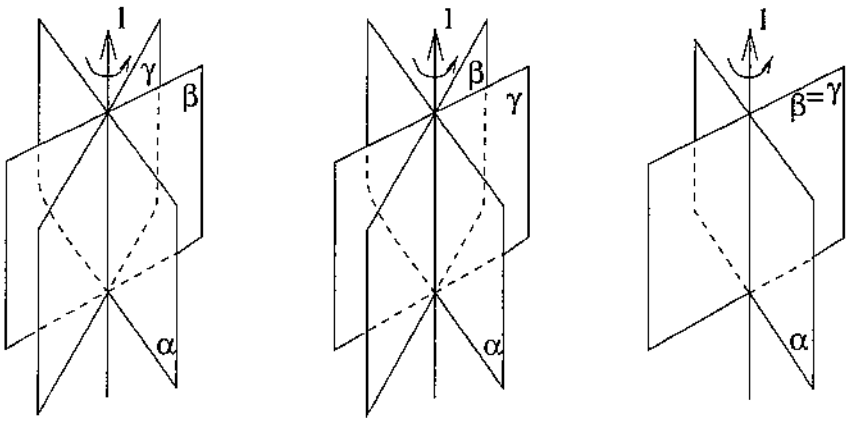


Figura 5.6: Ordem cíclica de três planos em torno de uma reta: (a) $\circledast_l(\alpha, \beta, \gamma) = +1$; (b) $\circledast_l(\alpha, \beta, \gamma) = -1$; (c) $\circledast_l(\alpha, \beta, \gamma) = 0$.

Note que o lado negativo de $\mathbb{S}^2$, isto é, o conjunto de pontos $p \in \mathbb{T}^3$ tais que $p \circ \mathbb{S}^2 = -1$, consiste de duas bolas unitárias separadas: uma no além e outra no aquém. Note também que $p \circ \mathbb{S}^2 = 0$ indica que o ponto p está sobre $\mathbb{S}^2$ ou sobre $\neg\mathbb{S}^2$, *a cópia antípoda* de $\mathbb{S}^2$.

Dados dois pontos p e q em $\mathbb{S}^2$, definimos a *distância entre p e q sobre* $\mathbb{S}^2$, denotada por $Sdist(p, q)$, como sendo o comprimento do menor arco de círculo máximo que passa por estes dois pontos. Mais precisamente, seja d a distância euclidiana entre p e q; então,

$$Sdist(p, q) = 2 \arccos \sqrt{1 - \frac{d^2}{4}}$$

5.10 Polaridade na esfera

Considere duas quádruplas de números a_0, a_1, a_2, a_3 e b_0, b_1, b_2, b_3 tais que $a_0b_0 + a_1b_1 + a_2b_2 + a_3b_3 = 0$. Pelo exposto nas seções anteriores, note que podemos interpretar esta equação de duas formas distintas: podemos dizer que o ponto com coordenadas $[a_0, a_1, a_2, a_3]$ pertence ao plano com coeficientes $\langle b_0, b_1, b_2, b_3 \rangle$ ou então, que o ponto com coordenadas $[b_0, b_1, b_2, b_3]$ pertence ao plano com coeficientes $\langle a_0, a_1, a_2, a_3 \rangle$. Portanto, a partir desta equação podemos perceber que o papel entre pontos e planos é perfeitamente intercambiável. Na verdade, este conceito denominado *dualidade*, não é uma exclusividade desta equação; ele é um dos conceitos mais importantes da geometria projetiva (convencional ou orientada).

Na geometria projetiva orientada, o princípio de dualidade é baseado num duomorfismo, isto é, uma bijeção entre dois espaços projetivos que troca pontos com planos, preservando suas posições relativas. Em consequência, um duomorfismo também troca

as operações $\vee$ e $\wedge$. Portanto, um duomorfismo define para cada conceito e teorema da geometria projetiva orientada, um conceito ou teorema *dual*. Por exemplo, o dual da sentença "o *encontro* de dois planos é uma reta" é "a *junção* de dois pontos é uma reta".

É importante ressaltar que há vários duomorfismos distintos que mapeiam planos em pontos e retas em retas. Cada um deles estabelece um tipo especial de dualidade que tem suas vantagens e desvantagens dependendo do contexto; veja [33, 29, 8]. Para os nossos propósitos, vamos considerar o seguinte mapeamento:

Definição 13 *O* complemento polar *relativo à esfera* $\mathbb{S}^2$, *denotado por* $*$, *é dado por:*

$$\begin{aligned}
[p_0, p_1, p_2, p_3]^* &= \langle -p_0, p_1, p_2, p_3 \rangle \\
\langle a_0, a_1, a_2, a_3 \rangle^* &= [-a_0, a_1, a_2, a_3] \\
\langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle^* &= \langle -l_5, l_4, l_3, -l_2, -l_1, l_0 \rangle
\end{aligned}$$

Vamos agora mostrar que $*$ é um duomorfismo de $\mathbb{T}^3$.

Primeiramente, considere o mapa projetivo [33] definido pela matriz

$$M = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note que este mapa projetivo leva os pontos do *aquém* de $\mathbb{T}^3$ em pontos do *além* e vice-versa. Além disso, ele mantém fixo os pontos no infinito, isto é, ele mapeia Ω_2 no próprio Ω_2. Este mapa projetivo é denominado *reflexão através* de Ω_2 e é fácil mostrar que

$$\begin{aligned}
\mathbb{T}^3 M &= \neg \mathbb{T}^3 \\
\Omega_2 M &= \Omega_2 \\
(a \vee b) M &= (aM) \vee (bM) \\
(a \wedge b) M &= \neg((aM) \wedge (bM))
\end{aligned}$$

Além disso, dado um ponto $p = [p_0, p_1, p_2, p_3]$, um plano $\alpha = \langle a_0, a_1, a_2, a_3 \rangle$ e uma reta $l = \langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle$ temos que

$$\begin{aligned}
[p_0, p_1, p_2, p_3]\, M &= [-p_0, p_1, p_2, p_3] \\
\langle a_0, a_1, a_2, a_3 \rangle\, M &= \langle a_0, -a_1, -a_2, -a_3 \rangle \\
\langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle\, M &= \langle l_0, l_1, -l_2, l_3, -l_4, -l_5 \rangle
\end{aligned}$$

Agora, seja $\vdash$ o *complemento polar à direita* [33, pág. 85] definido por

$$\begin{aligned}
[p_0, p_1, p_2, p_3]^{\vdash} &= \langle p_0, p_1, p_2, p_3 \rangle \\
\langle a_0, a_1, a_2, a_3 \rangle^{\vdash} &= [-a_0, -a_1, -a_2, -a_3] \\
\langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle^{\vdash} &= \langle l_5, -l_4, l_3, l_2, -l_1, l_0 \rangle
\end{aligned}$$

Daí, temos que

$$\begin{aligned}
([p_0, p_1, p_2, p_3] \cdot M)^{\vdash} &= [-p_0, p_1, p_2, p_3]^{\vdash} \\
&= \langle -p_0, p_1, p_2, p_3 \rangle \\
&= [p_0, p_1, p_2, p_3]^{*}
\end{aligned}$$

$$\begin{aligned}
(\langle a_0, a_1, a_2, a_3 \rangle \cdot M)^{\vdash} &= \langle a_0, -a_1, -a_2, -a_3 \rangle^{\vdash} \\
&= [-a_0, a_1, a_2, a_3] \\
&= \langle a_0, a_1, a_2, a_3 \rangle^{*}
\end{aligned}$$

$$\begin{aligned}
(\langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle \cdot M)^{\vdash} &= \langle l_0, l_1, -l_2, l_3, -l_4, -l_5 \rangle^{\vdash} \\
&= [-l_5, l_4, l_3, -l_2, -l_1, l_0] \\
&= \langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle^{*}
\end{aligned}$$

o que mostra que $*$ corresponde à composição $M \vdash$.

Visto que $\vdash$ é um duomorfismo, então, usando o teorema 7 de Stolfi [33, pág. 93], podemos concluir que $* = (M \vdash)$ também é um duomorfismo, denominado *polaridade esférica*. Para todo ponto, reta ou plano a, dizemos que a^* é o *polar esférico* de a.

Lema 7 *Sejam a e b dois elementos de $\mathbb{T}^3$. Então*

$$\begin{aligned}
(a \vee b)^* &= a^* \wedge b^* \\
(a \wedge b)^* &= \neg(a^* \vee b^*)
\end{aligned}$$

DEMONSTRAÇÃO: Como M é um mapa projetivo de $\mathbb{T}^3$ em $\neg\mathbb{T}^3$, verifica-se, para todo a e b em $\mathbb{T}^3$, que

$$\begin{aligned}
(a \vee b)M &= (aM) \vee (bM) \\
(a \wedge b)M &= \neg((aM) \wedge (bM))
\end{aligned}$$

Portanto,

$$(a \vee b)^* = (a \vee b)(M \vdash) = ((aM) \vee (bM))^{\vdash} = (aM)^{\vdash} \wedge (bM)^{\vdash} = a^* \wedge b^*$$

e

$$(a \wedge b)^* = (a \wedge b)(M \vdash) = (\neg((aM) \wedge (bM)))^{\vdash} = \neg((aM)^{\vdash} \vee (bM)^{\vdash}) = \neg(a^* \vee b^*)$$

□

Lema 8 *Seja a um elemento de* $\mathbb{T}^3$ *e seja d a sua dimensão. Então*

$$(a^*)^* = \neg^d a$$

DEMONSTRAÇÃO: Da definição de $\vdash$, segue que $(a^{\vdash})^{\vdash} = \neg^{d+1} a$. Por outro lado, pode-se provar que

$$\begin{aligned} aMM &= a \\ aM\vdash &= \neg(a \vdash M) \end{aligned}$$

Desta forma,

$$(a^*)^* = (aM\vdash)(M\vdash) = \neg(aM\vdash)(\vdash M) = \neg\neg^{d+1}(aMM) = \neg^d a$$

□

Do ponto de vista geométrico, a polaridade definida por $*$ pode ser descrita da seguinte forma: dado um ponto $p = [p_0, p_1, p_2, p_3]$ do lado positivo ("externo") da esfera $\mathbb{S}^2$, seja c o círculo correspondente à interseção entre $\mathbb{S}^2$ e o cone tangente a $\mathbb{S}^2$ com vértice em p. Veja figura 5.7 (a). Como podemos provar, p^* é o plano de suporte do círculo c. Além disso, se uma reta $l = \langle l_0, .. l_5 \rangle$ intercepta $\mathbb{S}^2$ então $l^* = \varphi_1 \wedge \varphi_2$, onde φ_1 e φ_2 são os planos tangentes aos pontos de interseção entre a reta l e a esfera $\mathbb{S}^2$. Veja figura 5.7 (b).

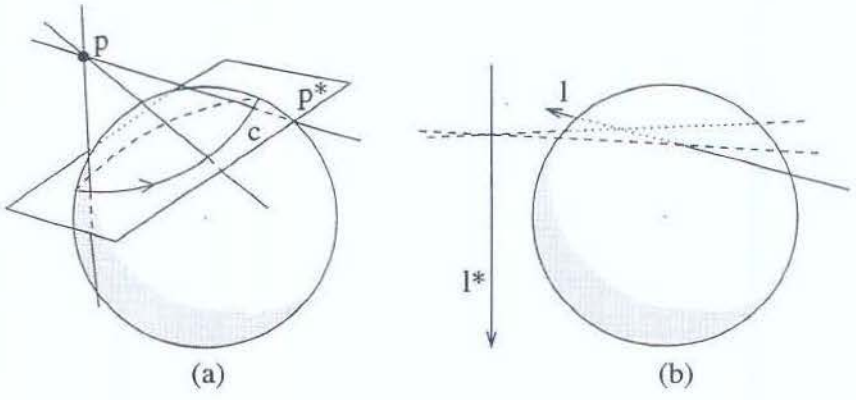


Figura 5.7: Polaridade esférica: (a) Ponto-Plano; (b) Reta-reta.

Para concluir esta seção, vale a pena mencionar que a dualidade é um conceito extremamente útil, pois ela permite reduzir consideravelmente o número de teoremas que precisam ser demonstrados, visto que a prova de um teorema automaticamente estabelece a veracidade do teorema dual. Além disso, pode-se escolher dentre os dois teoremas aquele que é mais simples de se provar.

Por outro lado, a dualidade também é útil do ponto de vista computacional pois ela permite que todo algoritmo geométrico faça o trabalho de dois. Por exemplo, o mesmo algoritmo que computa $a \vee b$ pode ser usado para computar $a \wedge b$, uma vez que $a \wedge b = \neg^{d_a+d_b}(a^* \vee b^*)^*$, onde d_a e d_b são respectivamente as dimensões de a e de b.

Capítulo 6

Geometria na esfera

Neste capítulo vamos descrever um método de representação da geometria dos mapas esféricos, ignorando por enquanto questões de erros de arredondamento. Além disso, vamos estabelecer algumas funções básicas para a manipulação desta geometria.

6.1 Mapas esféricos

Do ponto de vista geométrico, um *mapa esférico* é uma partição da esfera $\mathbb{S}^2$ em três tipos de *elementos*: os *vértices* (pontos), *arestas* (círculos ou arcos de círculos, não necessariamente máximos) e *faces* (regiões abertas). É importante notar que vamos permitir arestas que são um círculo completo, ou seja, neste caso a aresta não possui extremidade.

A representação da geometria dos vértices de um mapa esférico (ignorando questões de erros de arredondamento) não requer nenhuma consideração adicional — uma vez que um vértice é um ponto de $\mathbb{S}^2$, e como tal pode ser representado pelas suas coordenadas homogêneas, isto é, uma quádrupla de números reais $[w, x, y, z]$.

Uma aresta pode ser um círculo completo ou um arco de círculo, sendo que este círculo pode não ser máximo. Portanto, para representarmos uma aresta, é necessário definirmos uma representação para círculos arbitrários sobre $\mathbb{S}^2$.

6.2 Círculos sobre $\mathbb{S}^2$

Definimos um *círculo orientado na esfera*, ou *S-círculo*, como sendo um círculo sobre $\mathbb{S}^2$ com uma orientação interna (um sentido de percurso) especificado.

É fácil ver que todo S-círculo corresponde à interseção entre $\mathbb{S}^2$ e um plano orientado. Mais precisamente, dado um S-círculo c, há um único plano orientado α tal que $c = \alpha \cap \mathbb{S}^2$ e tal que o sentido de percurso de c concorda com a orientação interna do plano α e vice-

versa. Dizemos que α é o *plano de suporte* de c e escrevemos $\alpha = spln(c)$. Reciprocamente, dizemos que c é o S-círculo definido por α e escrevemos $c = scrc(\alpha)$. Veja figura 6.1.

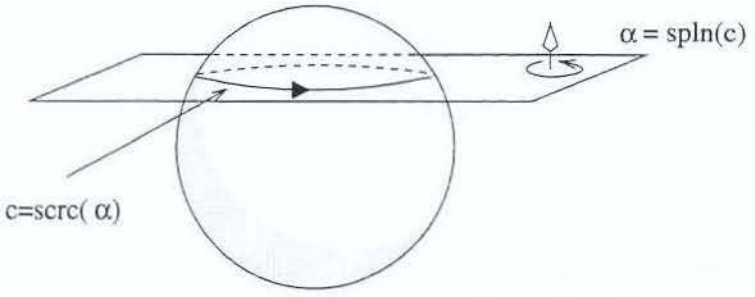


Figura 6.1: Um S-círculo

Desta forma, um S-círculo $c = scrc(\alpha)$ pode ser representado de maneira não ambígüa pelos coeficientes homogêneos $\langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ do plano orientado que o define. Neste caso, vamos indicar o S-círculo definido pelo plano α por $((\alpha_0, \alpha_1, \alpha_2, \alpha_3))$. É importante observar que esta notação só faz sentido quando o plano α define um S-círculo, o que é estabelecido pelo seguinte lema:

Lema 9 *Um plano $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ define um S-círculo se e somente se*

$$\alpha_0^2 < \alpha_1^2 + \alpha_2^2 + \alpha_3^2$$

DEMONSTRAÇÃO: Por definição, o plano α define um S-círculo se e somente se ele intercepta $\mathbb{S}^2$ em mais de um ponto; e isto ocorre se e somente se a distância entre α e a origem O é menor do que 1.

Pela fórmula (5.5), temos que a distância Euclidiana entre o plano α e a origem O é dada por

$$dist(\alpha, O) = \sqrt{\frac{\alpha_0^2}{\alpha_1^2 + \alpha_2^2 + \alpha_3^2}}$$

o que implica que $dist(\alpha, O) < 1$ se e somente se

$$\alpha_0^2 < \alpha_1^2 + \alpha_2^2 + \alpha_3^2.$$

□

É importante ressaltar que um plano $\langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ tangente a $\mathbb{S}^2$ (isto é, com $\alpha_0^2 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2$) não define um S-círculo.

Dizemos que dois S-círculos são *coincidentes* se eles incidem sobre um mesmo conjunto de pontos. Como é fácil perceber, os S-círculos $c = ((c_0, c_1, c_2, c_3))$ e $c' = ((-c_0, -c_1, -c_2, -c_3))$ são coincidentes, mas têm orientações opostas; neste caso, dizemos que eles são *opostos* entre si, o que denotamos por $c' = \neg c$.

6.3 Elementos de um S-círculo

A direção ortogonal ao plano $spln(c)$ que aponta para o lado positivo deste plano no aquém é denominada a *normal* do S-círculo c. Se $c = (\!(c_0, c_1, c_2, c_3)\!)$, a direção desta normal é dada pelo vetor (c_1, c_2, c_3) de $\mathbb{R}^3$. Em $\mathbb{T}^3$, é conveniente definir esta direção como sendo o ponto no infinito $snrm(c) = [0, c_1, c_2, c_3]$. O *eixo* de c é a reta $axis(c) = O \vee snrm(c) = \langle 0, 0, c_3, 0, -c_2, c_1 \rangle$. Veja figura 6.2.

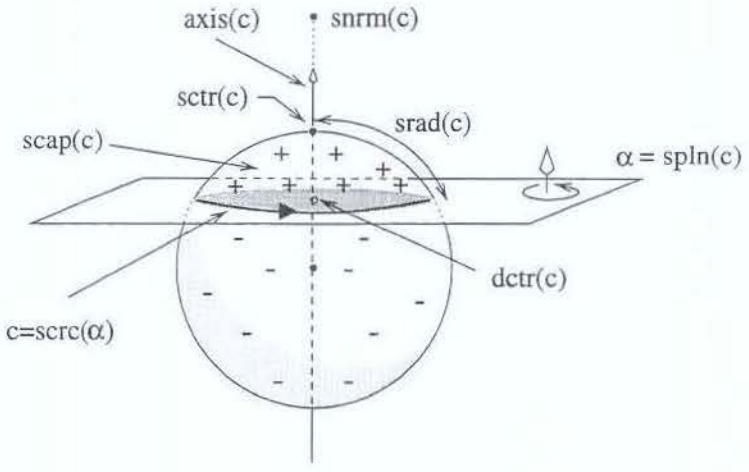


Figura 6.2: Elementos de um S-círculo

Um S-círculo c divide $\mathbb{S}^2$ em duas regiões abertas e não vazias denominadas *calotas*. Por definição, a *calota positiva* de c, denotada por $scap(c)$, é a parte de $\mathbb{S}^2$ localizada do lado positivo do plano $spln(c)$.

Podemos definir dois centros para um S-círculo: o *centro esférico*, ou *S-centro*, é o ponto $sctr(c)$ onde a reta $axis(c)$ cruza a calota positiva de c; o *centro planar*, ou *D-centro*, é o ponto no aquém $dctr(c)$, onde a reta $axis(c)$ cruza o plano $spln(c)$. Mais precisamente, pode-se demonstrar que

$$\begin{aligned} sctr(c) &= \left[\sqrt{c_1^2 + c_2^2 + c_3^2},\ c_1,\ c_2,\ c_3\right] \\ dctr(c) &= \neg(axis(c) \wedge spln(c)) = [c_1^2 + c_2^2 + c_3^2, -c_0c_1, -c_0c_2, -c_0c_3] \end{aligned} \tag{6.1}$$

O *raio esférico* ou *S-raio* de c é o comprimento $srad(c)$ do arco de círculo máximo que vai de $sctr(c)$ até um ponto qualquer de c; isto é,

$$srad(c) = \arccos\left(\frac{-c_0}{\sqrt{c_1^2 + c_2^2 + c_3^2}}\right) \in (0 \_\_ \pi). \tag{6.2}$$

Em outras palavras, dado um ponto p qualquer sobre o S-círculo c, $srad(c) = Sdist(p, sctr(c))$.

Seja $p = [p_0, p_1, p_2, p_3]$ um ponto em $\mathbb{S}^2$ e seja θ um número real no intervalo $(0 \_\_ \pi)$. Então, como podemos demonstrar, o S-círculo c cujo S-centro é p e cujo S-raio é θ é dado por

$$c = ((-p_0 \cos\theta, p_1, p_2, p_3)) \tag{6.3}$$

6.4 Arcos de S-círculos

Dois pontos distintos p e q sobre um S-círculo $c = scrc(\alpha)$ dividem este S-círculo em duas partes conexas denominadas *arcos esféricos* ou *S-arcos*. Veja figura 6.3. Por definição, *o S-arco aberto de c que vai de p até q*, denotado por $(p, \widehat{c}, q)$, é a seqüência (contínua) de pontos obtidos no percurso de p para q ao longo de c no sentido determinado pela orientação de c, excluindo os extremos p e q. Dizemos que p e q são respectivamente a *origem* e o *destino* de A, o que denotamos por $p = A.org$ e $q = A.dst$. Além disso, dizemos que c é o *S-círculo de suporte* de A, o que denotamos por $c = A.scirc$.

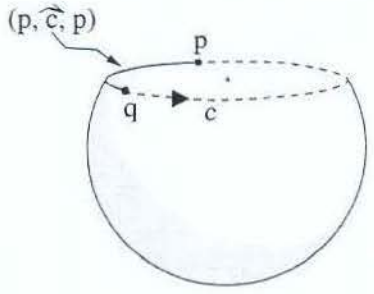


Figura 6.3: O S-arco $(p, \widehat{c}, q)$.

Como $p \neq q$, o S-arco $(p, \widehat{c}, q)$ é a parte de c contida no lado positivo do plano $\beta = p \vee q \vee snrm(c)$. Portanto, um ponto r de c está em $(p, \widehat{c}, q)$ se e somente se $r \diamond \beta = +1$.

Por conveniência, no caso particular em que $p = q$, definimos o S-arco $A = (p, \widehat{c}, q)$ como sendo todos os pontos de c exceto o ponto $p = q$. Neste caso, $A.org = p = q = A.dst$; um ponto r de c é um ponto de A se e somente se $r \neq p$.

Note que todo S-arco tem uma orientação própria que coincide com a orientação do seu S-círculo de suporte. O S-arco $A = (p, \widehat{c}, q)$, tomado em sentido inverso, é o S-arco $\neg A = (q, \widehat{\neg c}, p)$.

Dizemos que uma *S-curva* é um S-círculo ou um S-arco. Vamos representar uma S-curva genérica pela mesma notação $(p, \widehat{s}, q)$ usada para S-arcos, adotando a convenção que, no caso de um S-círculo, $A.org = \varnothing = A.dst$.

6.5 Operações envolvendo S-círculos

Vamos descrever a seguir algumas operações geométricas importantes para o desenvolvimento dos algoritmos de manipulação de mapas esféricos. Como veremos, estas operações podem ser resolvidas com relativa facilidade usando o fato de que um S-círculo equivale à interseção entre um plano e a esfera $\mathbb{S}^2$.

6.5.1 Posição de um ponto de $\mathbb{S}^2$ em relação a um S-círculo

Para localizar um ponto $p = [p_0, p_1, p_2, p_3] \in \mathbb{S}^2$ em relação a um S-círculo $c = ((c_0, c_1, c_2, c_3))$ vamos definir a operação $p \diamond c$ que retorna $+1$ caso p esteja na calota positiva de c; 0 caso p esteja sobre c; e -1 caso p esteja na calota negativa de c. Note que esta operação, na verdade, consiste em localizar o ponto p em relação ao plano de suporte do S-círculo c, ou seja,

$$p \diamond c = p \diamond spln(c) = sign(p_0c_0 + p_1c_1 + p_2c_2 + p_3c_3)$$

6.5.2 Interseção canônica entre dois S-círculos

A interseção entre dois S-círculos, quando existe, consiste em geral de dois pontos — ou de um único ponto, se os S-círculos são tangentes. Veja figura 6.4. Mais precisamente, a interseção entre dois S-círculos $a = ((a_0, a_1, a_2, a_3))$ e $b = ((b_0, b_1, b_2, b_3))$ é o conjunto dos pontos $p = [w, x, y, z]$ que satisfazem às equações:

$$\begin{cases} a_0w + a_1x + a_2y + a_3z &= 0 \\ b_0w + b_1x + b_2y + b_3z &= 0 \\ -w^2 + x^2 + y^2 + z^2 &= 0 \end{cases}$$

Resolvendo-se este sistema temos

$$\begin{aligned} p_1 &= \left[\mu,\ -l_1l_2 - l_3l_4 + l_5\sqrt{\delta},\ l_0l_2 - l_3l_5 - l_4\sqrt{\delta},\ l_0l_4 + l_1l_5 + l_2\sqrt{\delta}\ \right] \\ p_2 &= \left[\mu,\ -l_1l_2 - l_3l_4 - l_5\sqrt{\delta},\ l_0l_2 - l_3l_5 + l_4\sqrt{\delta},\ l_0l_4 + l_1l_5 - l_2\sqrt{\delta}\ \right] \end{aligned} \tag{6.4}$$

onde

$$\begin{aligned} l_0 &= a_0b_1 - a_1b_0, \quad l_1 = a_0b_2 - a_2b_0, \quad l_2 = a_0b_3 - a_3b_0 \\ l_3 &= a_1b_2 - a_2b_1, \quad l_4 = a_1b_3 - a_3b_1, \quad l_5 = a_2b_3 - a_3b_2 \end{aligned}$$

$$\mu = l_2^2 + l_4^2 + l_5^2 \quad \text{e} \quad \delta = \mu - (l_0^2 + l_1^2 + l_3^2)$$

É importante notar que o sinal de δ na equação (6.4) discrimina a maneira como os dois S-círculos se interceptam. Definimos portanto o *discriminante* dos dois S-círculos como sendo a função

$$\Diamond(a,b) = sign(\delta) = sign(l_2^2 + l_4^2 + l_5^2 - l_0^2 - l_1^2 - l_3^2)$$

Verifica-se facilmente que, para quaisquer S-círculos a e b,

$$\Diamond(a,b) = \begin{cases} -1 & \text{se } a \text{ e } b \text{ não se interceptam (em pontos com coordenadas reais)} \\ 0 & \text{se } a \text{ e } b \text{ são coincidentes ou tangentes entre si} \\ 1 & \text{se } a \text{ e } b \text{ se interceptam em dois pontos distintos} \end{cases}$$

Além disso, verifica-se também que, se $\Diamond(a,b) > 0$, então o ponto p_1 da fórmula (6.4) é o ponto onde o S-círculo a alcança o S-círculo b vindo pelo lado positivo de b. Veja a figura 6.4(a).

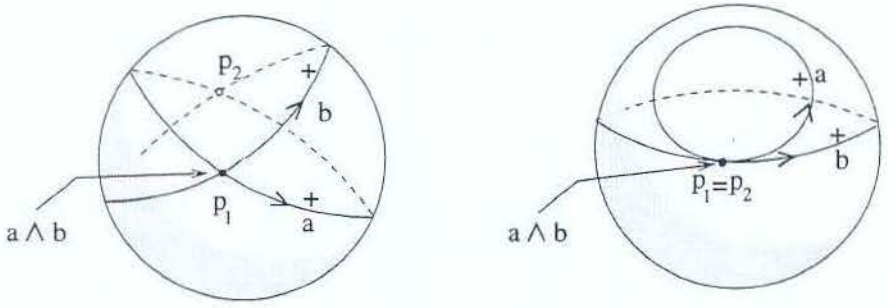


Figura 6.4: Interseção canônica entre dois S-círculos.

Por outro lado, se $\Diamond(a,b) = 0$, então os S-círculos são coincidentes ou são tangentes entre si; neste caso, o S-círculo a (exceto o ponto comum) está todo no lado positivo do S-círculo b se e somente se o centro planar de a está do lado positivo do plano de suporte de b, isto é, $dctr(a) \diamond spln(b) = +1$. Veja a figura 6.4(b).

Isto motiva a seguinte definição:

Definição 14 Sejam a e b dois S-círculos. Se $\Diamond(a,b) > 0$, ou $\Diamond(a,b) = 0$ e $dctr(a) \diamond spln(b) = +1$, então a *interseção canônica* entre a e b, denotada por $a \wedge b$, é o ponto onde o S-círculo a alcança o S-círculo b vindo pelo lado positivo de b. Caso contrário, $a \wedge b = [0,0,0,0] = [\mathbf{0}]$.

Note que se $a \wedge b \neq [\mathbf{0}]$, então este ponto é o ponto p_1 da fórmula (6.4). Note também que, se $\Diamond(a,b) > 0$, então

$$(\neg a) \wedge b = a \wedge (\neg b)$$

é o outro ponto de interseção entre a e b, ou seja, é o ponto p_2 da fórmula (6.4), onde a cruza b passando do lado negativo para o lado positivo de b.

6.5.3 Ponto diametralmente oposto num S-círculo

Uma operação particularmente importante na elaboração dos algoritmos de ordenação cíclica, é a determinação do ponto *diametralmente oposto* a um dado ponto p sobre um S-círculo c que passa por p. Vamos denotar esta operação por $\oslash_c(p)$. Veja a figura 6.5.

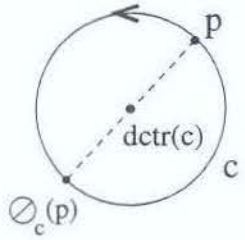


Figura 6.5: Ponto diametralmente oposto a p no S-círculo c.

Como podemos verificar através da figura, $\oslash_c(p)$ é o ponto simétrico a p em relação ao ponto $dctr(c)$. Mais precisamente, se $p = [p_0, p_1, p_2, p_3]$ e $c = ((c_0, c_1, c_2, c_3))$, pode-se verificar que

$$\oslash_c(p) = [p_0\kappa, -2c_0c_1p_0 - p_1\kappa, -2c_0c_2p_0 - p_2\kappa, -2c_0c_3p_0 - p_3\kappa]$$

onde $\kappa = c_1^2 + c_2^2 + c_3^3$.

6.5.4 Posição relativa de dois pontos sobre um S-círculo

Definimos a *posição relativa* de dois pontos p e q sobre um S-círculo c, como sendo o predicado $\otimes_c(p, q)$ que, considerando a orientação de c, retorna $+1$ se q ocorre depois de p e antes do ponto $\oslash_c(p)$; retorna -1 se q ocorre após $\oslash_c(p)$ e antes de p e retorna 0 caso $q = p$ ou $q = \oslash_c(p)$. Veja as figuras 6.6.

Em outras palavras, o predicado $\otimes_c(p, q)$ descreve o sentido, em relação à orientação de c, do caminho mais curto que vai de p até q. É fácil perceber que

$$\otimes_c(p, q) = \otimes_{\neg c}(q, p) = -\otimes_c(q, p)$$

Note que este predicado corresponde a determinar se a reta $p \vee q$ é orientada positiva ou negativamente em relação à reta $axis(c) = O \vee snrm(c)$. Portanto, de acordo com a

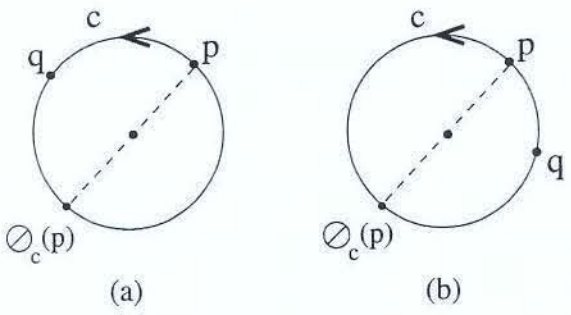


Figura 6.6: Definição de $\otimes_c(p,q)$: (a) $\otimes_c(p,q) = +1$; (b) $\otimes_c(p,q) = -1$.

fórmula (5.10), temos que

$$\otimes_c(p,q) = sign \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 \\ p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \end{vmatrix} = sign \begin{vmatrix} c_1 & c_2 & c_3 \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix} \tag{6.5}$$

6.5.5 Ordenação cíclica de três pontos sobre um S-círculo

Definimos a *ordem cíclica* de três pontos $p = [p_0, p_1, p_2, p_3]$, $q = [q_0, q_1, q_2, q_3]$ e $r = [r_0, r_1, r_2, r_3]$ sobre um S-círculo $c = ((c_0, c_1, c_2, c_3))$, como sendo o predicado $\circledast_c(p,q,r)$ que retorna $+1$ se os três pontos p, q e r ocorrem nesta ordem (cíclica) ao longo de c, quando este é percorrido no sentido positivo; retorna -1 se estes pontos ocorrem na ordem oposta; e retorna 0 se quaisquer dois dos pontos são iguais. Veja figura 6.7.

Como podemos notar, esta operação corresponde a determinar a orientação do triângulo determinado pelos 3 pontos em relação ao plano $spln(c)$. Como vimos na seção 5.7.3,

$$\circledast_c(p,q,r) = \Delta(p,q,r,\alpha) = sign \begin{vmatrix} p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ c_0 & c_1 & c_2 & c_3 \end{vmatrix}$$

Um resultado particularmente interessante que relaciona os predicados $\circledast$ e $\otimes$ é

Teorema 10 *Para quaisquer três pontos p, q e r sobre um S-círculo c, temos que*

$$\circledast_c(p,q,r) = sign(\otimes_c(p,q) + \otimes_c(q,r) + \otimes_c(r,p))$$

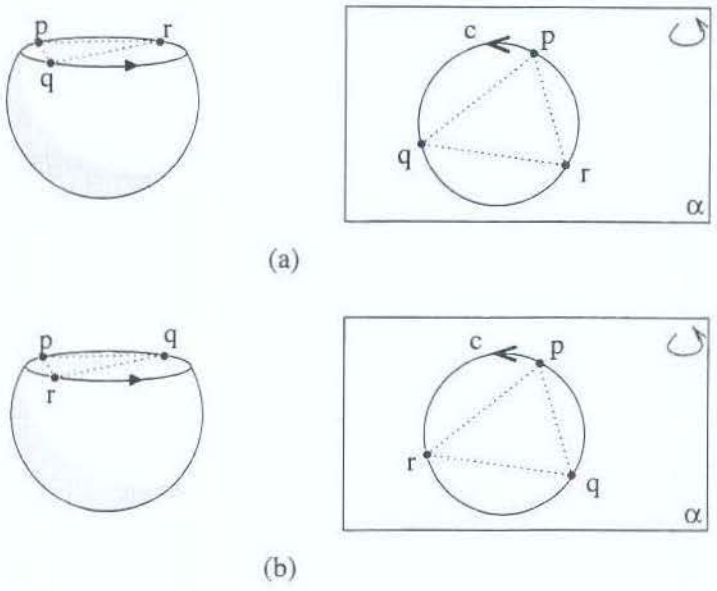


Figura 6.7: Ordenação cíclica de três pontos sobre um S-círculo: (a) $\circledast_c(p,q,r) = +1$; (b) $\circledast_c(p,q,r) = -1$.

DEMONSTRAÇÃO: Seja $\alpha = spln(c)$. Como definido por Stolfi [33], a *assinatura* de um ponto u de α em relação ao triângulo pqr e ao plano α é a sequência de sinais $\sigma_0\sigma_1\sigma_2$, onde

$$\begin{aligned} \sigma_0 &= \Delta(u,q,r,\alpha) \\ \sigma_1 &= \Delta(p,u,r,\alpha) \\ \sigma_2 &= \Delta(p,q,u,\alpha) \end{aligned}$$

Ou seja, σ_i é a orientação relativa a α do triângulo pqr, com seu i-ésimo vértice substituido pelo ponto u. Veja figura 6.8. Em particular, se u está dentro do triângulo pqr então

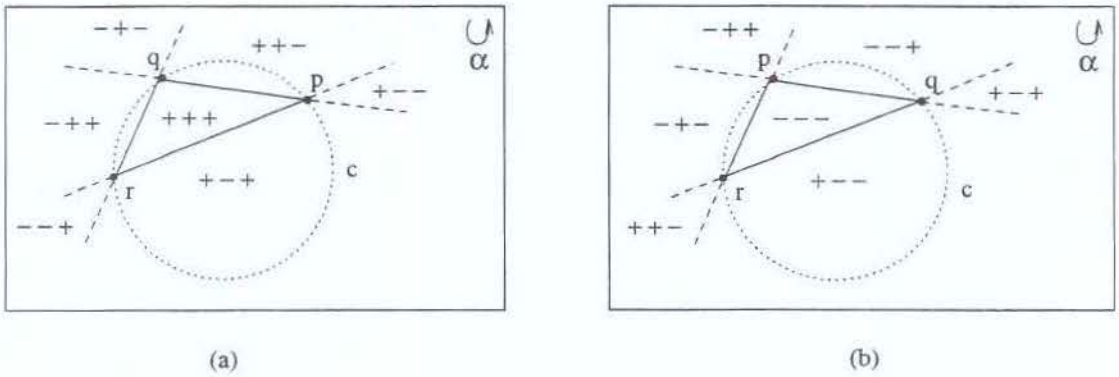


Figura 6.8: Assinatura de um ponto relativa ao triângulo pqr e ao plano α.

a assinatura de u é ou '$+++$' ou '$---$', dependendo da orientação do triângulo em relação ao plano α. Mais especificamente, se u está no interior do triângulo então, para todo $i = 0, 1, 2$, $\sigma_i = \Delta(p, q, r, \alpha)$.

Pela figura 6.8, podemos verificar que a assinatura de qualquer ponto u no disco delimitado por c é sempre dominada pela orientação do triângulo pqr em relação a α. Isto é, se u está no interior do círculo c, então

$$\Delta(p, q, r, \alpha) = sign(\sigma_0 + \sigma_1 + \sigma_1)$$

Em particular, tomando-se $u = dctr(c)$, temos que

$$\sigma_0 = sign \begin{vmatrix} \sqrt{c_1^2 + c_2^2 + c_3^2} & c_1 & c_2 & c_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ c_0 & c_1 & c_2 & c_3 \end{vmatrix} = sign \begin{vmatrix} \sqrt{c_1^2 + c_2^2 + c_3^2} & c_1 & c_2 & c_3 \\ c_0 & c_1 & c_2 & c_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \end{vmatrix} \tag{6.6}$$

Conforme vimos na seção 5.7, esta fórmula define a orientação relativa das retas $dctr(c) \vee [c_0, c_1, c_2, c_3]$ e $q \vee r$.

Como $c_0^2 < c_1^2 + c_2^2 + c_3^2$ (pois c é um S-círculo), então o ponto $c' = [c_0, c_1, c_2, c_3]$ está do lado positivo ("exterior") de $\mathbb{S}^2$. Além disso, temos que $c' \diamond spln(c) = +1 = snrm(c) \diamond spln(c)$, ou seja, os pontos c' e $snrm(c)$ estão do mesmo lado do plano $spln(c)$. Portanto, como $q \vee r$ está no plano α temos que

$$(dctr(c) \vee c') \diamond (q \vee r) = (dctr(c) \vee snrm(c)) \diamond (q \vee r) = (O \vee snrm(c)) \diamond (q \vee r)$$

Isto implica que, no último determinante de (6.6), podemos substituir $dctr(c)$ por O e c' por $snrm(c)$. Daí,

$$\sigma_0 = sign \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & c_1 & c_2 & c_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \end{vmatrix}$$

Agora, comparando esta expressão com a fórmula (6.5), podemos concluir que $\sigma_0 = \otimes_c(q, r)$.

Utilizando um raciocínio análogo, podemos concluir que $\sigma_1 = \otimes_c(r, p)$ e $\sigma_2 = \otimes_c(p, q)$.

Logo,

$$\circledast_c(p, q, r) = sign(\otimes_c(p, q) + \otimes_c(q, r) + \otimes_c(r, p))$$

□

6.6 Ordenação cíclica de três S-círculos em torno de um ponto

Outra operação importante para o desenvolvimento de algoritmos de manipulação de mapas esféricos é a determinação da ordem (cíclica) em que três S-círculos saem de um ponto em comum. Para melhor definir esta operação, vamos precisar do seguinte conceito:

Definição 15 Um *S-vetor* é um par (p, c), onde p é um ponto de $\mathbb{S}^2$ e c é um S-círculo passando por p.

Intuitivamente, um S-vetor (p, c) consiste dos pontos do S-círculo c logo adiante de p, no sentido positivo de c, ou seja, é o trecho inicial do S-círculo saindo do ponto p.

Desta forma, dados três S-círculos a, b e c, todos passando pelo ponto p, vamos definir o predicado $\otimes_p(a, b, c)$, como sendo a ordem cíclica dos três S-vetores $s_1 = (p, a)$, $s_2 = (p, b)$ e $s_2 = (p, c)$ em torno do ponto p. Mais precisamente, $\otimes_p(a, b, c)$ é igual a $+1$ se esta ordem (cíclica) coincide com a orientação canônica de $\mathbb{S}^2$; é -1 se esta ordem é oposta; e, é 0 se pelo menos dois dos S-vetores são iguais. Veja figura 6.9.

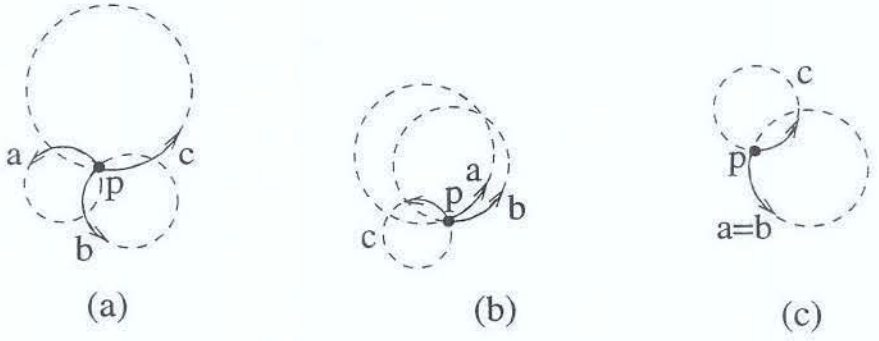


Figura 6.9: Ordem cíclica de três S-círculos em torno de um ponto em comum: (a) $\otimes_p(a, b, c) = +1$; (b) $\otimes_p(a, b, c) = -1$; (c) $\otimes_p(a, b, c) = 0$.

Note que, se os planos de suporte dos três S-círculos têm uma única reta em comum, então o valor de $\otimes_p(a, b, c)$ pode ser facilmente obtido pelo predicado $\otimes_l(spln(a), spln(b), spln(c))$ — visto na seção 5.8.2 — onde, l é a reta comum aos três planos. Por outro lado, se os planos não têm uma única reta em comum, então, neste caso, para obter o valor do predicado $\otimes_p(a, b, c)$ é necessário levar em consideração a orientação e curvatura dos S-círculos. Na seção 9.5, vamos apresentar um algoritmo (exato) para determinar o valor deste predicado.

6.7 P-círculos

Definimos um *P-círculo* como sendo um círculo orientado no plano $\mathbb{R}^2$. Da geometria euclidiana, sabemos que um P-círculo é o conjunto dos pontos $(x, y) \in \mathbb{R}^2$ que satisfazem à equação

$$a_0(x^2 + y^2) + a_1x + a_2y + a_3 = 0 \tag{6.7}$$

Esta equação define um P-círculo se e somente se

$$a_1^2 + a_2^2 - 4a_0a_3 > 0$$

Visto que os números a_0, a_1, a_2 e a_3 — os *coeficiente* do P-círculo — definem univocamente o P-círculo, então, vamos denotar estes coeficientes por $[\![a_0, a_1, a_2, a_3]\!]$.

Por definição, todo P-círculo é orientado. O *lado positivo* de $a = [\![a_0, a_1, a_2, a_3]\!]$ consiste dos pontos (x, y) de $\mathbb{R}^2$ tais que

$$a_0(x^2 + y^2) + a_1x + a_2y + a_3 > 0$$

Além disso, o *sentido positivo* de percurso do P-círculo a é o sentido anti-horário, visto de um ponto no seu lado positivo.

Destas definições, é fácil perceber que os coeficientes de um P-círculo são homogêneos, isto é, para todo $\lambda > 0$, os coeficientes $[\![a_0, a_1, a_2, a_3]\!]$ e $[\![\lambda a_0, \lambda a_1, \lambda a_2, \lambda a_3]\!]$, definem o mesmo P-círculo.

Novamente, da geometria euclidiana, temos que um círculo em $\mathbb{R}^2$ com centro no ponto (x_c, y_c) e com raio $r > 0$ consiste do conjunto de pontos $(x, y) \in \mathbb{R}^2$ tais que

$$(x^2 + y^2) - 2x_cx - 2y_cy + (x_c^2 + y_c^2 - r^2) = 0 \tag{6.8}$$

ou seja, os coeficientes deste P-círculo são $[\![1, -2x_c, -2y_c, x_c^2 + y_c^2 - r^2]\!]$. Comparando-se esta equação com a equação (6.7), podemos concluir que, dado um P-círculo $a = [\![a_0, a_1, a_2, a_3]\!]$, o centro de a é o ponto

$$pctr(a) = \left(\frac{-a_1}{2a_0}, \frac{-a_2}{2a_0}\right)$$

e o raio de a é dado por

$$prad(a) = \frac{\sqrt{a_1^2 + a_2^2 - 4a_0a_3}}{2a_0}$$

Note que quando $a_0 = 0$, o P-círculo $[\![a_0, a_1, a_2, a_3]\!]$ se degenera numa reta, isto é, num círculo com raio infinito. Neste caso, podemos dizer que o centro deste P-círculo é um ponto no infinito na direção $(-a_1, -a_2)$.

6.8 Projeção estereográfica

Para estabelecer a relação entre S-círculos e P-círculos, vamos utilizar o conceito de *projeção estereográfica* que define uma correspondencia biunívoca entre todos os pontos de $\mathbb{S}^2$, exceto um – o *centro de projeção* – e os pontos de um *plano de projeção*. Para os nossos propósitos definimos a *projeção estereográfica canônica*, denotada por ψ, na qual o polo norte $\mathcal{N} = [1,0,0,1]$ é o centro de projeção, e o plano equatorial $\langle 0,0,0,1\rangle$ — identificado com o plano $\mathbb{R}^2$ — é o plano de projeção. Veja figura 6.10.

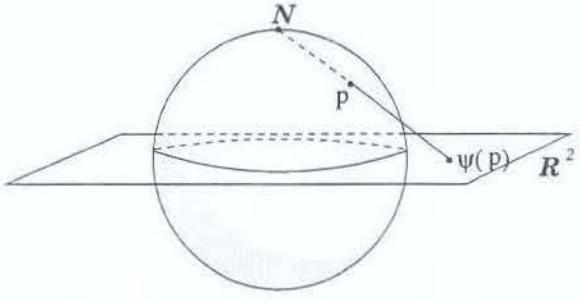


Figura 6.10: Projeção estereográfica canônica.

Mais precisamente, dado um ponto $[w,x,y,z]$ em $(\mathbb{S}^2 \setminus \mathcal{N})$ definimos

$$\psi([w,x,y,z]) = \left(\frac{x}{w-z}, \frac{y}{w-z}\right)$$

Reciprocamente, a *projeção estereográfica (canônica) inversa* que mapeia os pontos (x,y) de $\mathbb{R}^2$ nos pontos de $\mathbb{S}^2 \setminus \mathcal{N}$ é dada por

$$\psi^{-1}(x,y) = \left[x^2+y^2+1, 2x, 2y, x^2+y^2-1\right]$$

É importante notar que a projeção estereográfica ψ estabelece uma bijeção entre S-círculos e P-círculos. Mais precisamente,

Teorema 11 *Para todo S-círculo* $((s_0,s_1,s_2,s_3))$ *e todo P-círculo* $[\![a_0,a_1,a_2,a_3]\!]$, *temos que*

$$\begin{aligned}\psi(((s_0,s_1,s_2,s_3)) \setminus \mathcal{N}) &= [\![s_0+s_3, 2s_1, 2s_2, s_0-s_3]\!] \\ \psi^{-1}([\![a_0,a_1,a_2,a_3]\!]) &= ((a_0+a_3, a_1, a_2, a_0-a_3)) \setminus \mathcal{N}\end{aligned}$$

Note que esta projeção leva S-círculos que não passam pelo polo norte em P-círculos de raio finito e vice-versa; e S-círculos que passam pelo polo norte para retas de $\mathbb{R}^2$, isto é, P-círculos com raio infinito, e vice-versa. Por exemplo, o S-círculo equatorial $((0,0,0,1))$

corresponde ao P-círculo $[\![1, 0, 0, -1]\!]$, cujo centro é o ponto $(0, 0)$ e cujo raio é 1; e o S-círculo $((0, 1, 0, 0))$ corresponde a $[\![0, 2, 0, 0]\!]$ que é o eixo Y de $\mathbb{R}^2$.

É importante notar que podemos definir uma infinidade de projeções estereográficas "não canônicas" escolhendo diferentes centros de projeção e/ou diferentes planos de projeção. Cada uma destas projeções estabelece uma correspondência diferente entre S-círculos e círculos no plano de projeção.

Capítulo 7

Geometria exata na esfera

A implementação convencional da maioria das operações descritas no capítulo 6 utiliza números em ponto flutuante para definir as coordenadas e coeficientes dos pontos, retas e planos. Portanto, esta forma de implementação está sujeita a erros de arredontamento inerentes deste esquema de representação numérica. Como observamos no capítulo 1, quaisquer erros de arredondamento são fatais para algoritmos geométricos. Para evitar estes problemas, nosso objetivo neste capítulo é adequar a representação da geometria dos mapas esféricos, descrita nos capítulos anteriores, de modo que ela possa ser manipulada de maneira exata (isto é, livre de erros de arredondamentos).

De um modo geral, para evitar os erros de arredondamento inerentes às operações em ponto flutuante é necessário nos restringirmos aos números racionais e às operações racionais tais como soma, subtração, multiplicação e divisão. Como veremos a seguir, a utilização de coordenadas homogêneas nos permite reduzir de maneira natural todos os cálculos a operações com números inteiros.

7.1 Pontos racionais

Um ponto de $\mathbb{R}^3$ é dito *racional* se todas as suas coordenadas cartesianas (X, Y, Z) são números racionais. Visto que as coordenadas homogêneas de um ponto $(X, Y, Z) \in \mathbb{R}^3$ são $[\lambda, \lambda X, \lambda Y, \lambda Z]$ para todo $\lambda > 0$, então qualquer ponto racional de coordenadas cartesianas $\left(\frac{p_x}{q_x}, \frac{p_y}{q_y}, \frac{p_z}{q_z}\right)$ pode ser representado pela quádrupla de números inteiros $\left[\lambda, \frac{\lambda}{q_x}p_x, \frac{\lambda}{q_y}p_y, \frac{\lambda}{q_z}p_z\right]$ com $\lambda = mmc(q_x, q_y, q_z)$. Portanto, podemos dizer que um ponto de $\mathbb{R}^3$ é um ponto racional se e somente se ele possui coordenadas homogêneas inteiras.

Analogamente, uma *direção racional* — a direção de um vetor $\left(\frac{p_x}{q_x}, \frac{p_y}{q_y}, \frac{p_z}{q_z}\right)$ com coordenadas racionais — pode ser representada pelo ponto no infinito $\left[0, \frac{\lambda}{q_x}p_x, \frac{\lambda}{q_y}p_y, \frac{\lambda}{q_z}p_z\right]$ com $\lambda = mmc(q_x, q_y, q_z)$.

7.2 Pontos de $\mathbb{S}^2$ representáveis exatamente

Há dois conjuntos de pontos de $\mathbb{S}^2$ que podem ser representados exatamente de modo bastante simples. Um deles, que denotamos por $\mathcal{A}$, é o conjunto de pontos racionais sobre a esfera $\mathbb{S}^2$, isto é,

$$\mathcal{A} = \left\{[a_0, .. a_3] \mid a_1^2 + a_2^2 + a_3^2 = a_0^2,\ a_0 > 0 \text{ e } (a_0, .. a_3) \in \mathbb{Z}^4\right\}$$

O outro, denotado por $\mathcal{B}$, é o conjunto dos pontos $p = [w, x, y, z]$ de $\mathbb{S}^2$ cujas três últimas coordenadas homogêneas x, y e z são números racionais. Mais precisamente

$$\mathcal{B} = \left\{[b_0, .. b_3] \mid b_1^2 + b_2^2 + b_3^2 = b_0^2,\ b_0 > 0 \text{ e } (b_1, b_2, b_3) \in \mathbb{Z}^3\right\}$$

Em outras palavras, $\mathcal{B}$ é o conjunto de pontos de $\mathbb{S}^2$ cujas direções em $\mathbb{R}^3$, relativas à origem, são racionais; equivalentemente, $\mathcal{B}$ consiste das projeções radiais sobre $\mathbb{S}^2$ de todos os pontos racionais de $\mathbb{R}^3$, exceto $(0, 0, 0)$.

Visto que os pontos de $\mathcal{B}$ são pontos em $\mathbb{S}^2$, então a coordenada w pode ser determinada pela fórmula $w = \sqrt{x^2 + y^2 + z^2}$ e portanto pode ser deixada implícita. Ou seja, um ponto p em $\mathcal{B}$ pode ser representado exatamente pela tripla $[x, y, z]$ correspondente à direção do vetor que sai da origem de $\mathbb{S}^2$ e passa pelo ponto p.

Note que $\mathcal{A} \subset \mathcal{B}$.

7.2.1 Aproximação de pontos genéricos por pontos de $\mathcal{B}$

Nos resultados a seguir usaremos a notação $\lfloor\!\lfloor x \rceil\!\rceil$ para indicar a parte inteira de x, isto é

$$\lfloor\!\lfloor x \rceil\!\rceil = \begin{cases} \lfloor x \rfloor & \text{se } x \geq 0 \\ \lceil x \rceil & \text{se } x < 0 \end{cases}$$

Vamos agora mostrar que o conjunto $\mathcal{B}$ é denso sobre $\mathbb{S}^2$, e nesta demonstração, vamos definir um método de aproximação de pontos genéricos de $\mathbb{S}^2$ por pontos de $\mathcal{B}$.

Lema 12 *Para todo ponto $p = (x, y, z)$ em $\mathbb{S}^2$ e todo real $\epsilon > 0$, seja η um real tal que $\eta \geq \sqrt{3} \cdot \max\left\{1, \frac{\pi}{2\epsilon}\right\}$ e sejam $q_1 = \lfloor\!\lfloor \eta x \rceil\!\rceil$, $q_2 = \lfloor\!\lfloor \eta y \rceil\!\rceil$, $q_3 = \lfloor\!\lfloor \eta z \rceil\!\rceil$ e $q_0 = \sqrt{q_1^2 + q_2^2 + q_3^2}$. Então, o ponto $q = [q_0, .. q_3]$ está em $\mathcal{B}$, e $Sdist(p, q) < \epsilon$.*

DEMONSTRAÇÃO: Sejam δ_1, δ_2 e δ_3 os três números reais tais que $\lfloor\!\lfloor \eta x \rceil\!\rceil = \eta x - \delta_1$, $\lfloor\!\lfloor \eta y \rceil\!\rceil = \eta y - \delta_2$ e $\lfloor\!\lfloor \eta z \rceil\!\rceil = \eta z - \delta_3$. Portanto, $\vec{v} = \eta\vec{u} - \vec{\delta}$, onde $\vec{v} = (q_1, q_2, q_3)$, $\vec{u} = p = (x, y, z)$ e $\vec{\delta} = (\delta_1, \delta_2, \delta_3)$.

Note que $|\delta_i| < 1$, para todo i; portanto, $\|\vec{\delta}\| < \sqrt{3}$. Logo,

$$\|\vec{v}\| = \|\eta\vec{u} - \vec{\delta}\| \geq \|\eta\vec{u}\| - \|\vec{\delta}\| = \eta - \|\vec{\delta}\| > \sqrt{3} - \sqrt{3} = 0$$

ou seja, $\|\vec{v}\| > 0$, o que implica que pelo menos uma das coordenadas q_1, q_2 ou q_3 é não nula; portanto, $[q_0, q_1, q_2, q_3]$ é um ponto de $\mathcal{B}$.

Seja θ o ângulo entre $\vec{u}$ e $\vec{v}$. Então, $\theta \in [0 \_\_ \pi]$ e $Sdist(p, q) = \theta$. Além disso, seja d a distância euclidiana entre o ponto $\eta\vec{u}$ e a reta Oq — veja a figura 7.1 — daí,

$$\sin\theta = \frac{d}{\|\eta\vec{u}\|} = \frac{d}{\eta} \leq \frac{\|\vec{\delta}\|}{\eta} < \frac{\sqrt{3}}{\eta}$$

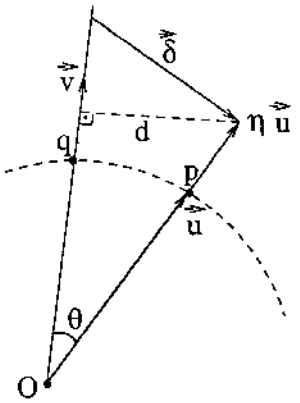


Figura 7.1: Distância d entre o ponto $\eta\vec{u}$ e a reta Oq.

Daí, $\theta < \arcsin\left(\frac{\sqrt{3}}{\eta}\right)$. Visto que $\eta \geq \frac{\pi\sqrt{3}}{2\epsilon}$ e visto que, para todo $\gamma \in [0 \_\_ 1]$, $\arcsin(\gamma) \leq \frac{\pi}{2}\gamma$, então

$$Sdist(p, q) = \theta < \arcsin\left(\frac{\sqrt{3}}{\eta}\right) < \frac{\pi}{2}\frac{\sqrt{3}}{\eta} \leq \frac{\pi}{2}\sqrt{3}\frac{2\epsilon}{\pi\sqrt{3}} = \epsilon$$

□

O lema 12 dá origem a um algoritmo bastante simples para aproximar um ponto genérico da esfera $\mathbb{S}^2$ por um ponto de $\mathcal{B}$. Vamos denominar este algoritmo de BAprox.

Corolário 13 *Para todo ponto p de $\mathbb{S}^2$ e todo real ϵ com $0 < \epsilon < \pi/2$, existe um ponto $q = [q_0, .. q_3]$ em $\mathcal{B}$ tal que $Sdist(p, q) < \epsilon$, e cujas coordenadas inteiras q_1, q_2 e q_3 têm no máximo $\lfloor \log_2 \frac{1}{\epsilon} \rfloor + 3$ bits.*

Corolário 14 *O conjunto $\mathcal{B}$ é denso sobre $\mathbb{S}^2$.*

7.2.2 Aproximação de pontos genéricos por pontos de $\mathcal{A}$

Considerando a fórmula da projeção estereográfica canônica ψ (seção 6.8), podemos verificar facilmente os seguintes resultados:

Lema 15 *A projeção estereográfica canônica ψ é uma bijeção contínua entre os pontos de $\mathcal{A} \setminus \mathcal{N}$ e os pontos racionais $\mathbb{Q}^2$ de $\mathbb{T}^2$.*

Uma vez que os pontos racionais são um subconjunto denso de $\mathbb{R}^2$, o lema 15 nos permite concluir que

Corolário 16 *O conjunto $\mathcal{A}$ é denso sobre $\mathbb{S}^2$.*

Este resultado sugere um algoritmo, denominado AAprox, para aproximar um ponto genérico $p = [p_0, p_1, p_2, p_3]$ de $\mathbb{S}^2$ por um ponto de $\mathcal{A}$ com uma precisão arbitrária $\epsilon > 0$ dada.

Sem perda de generalidade, podemos supor que p está no hemisfério sul da esfera $\mathbb{S}^2$, pois caso contrário, basta determinar uma aproximação $q = [q_0, q_1, q_2, q_3]$ para o ponto $\overline{p} = [p_0, -p_1, -p_2, -p_3]$ e tomar o ponto $\overline{q} = [q_0, -q_1, -q_2, -q_3]$. Pode-se verificar que, para quaisquer dois pontos p e q no hemisfério sul da esfera $\mathbb{S}^2$

$$Sdist(p, q) \leq 2\, dist(\psi(p) - \psi(q)) \tag{7.1}$$

Portanto, o algoritmo AAprox consiste em: (1) calcular inicialmente o ponto $p' = \psi(p)$ de $\mathbb{R}^2$ com erro $\delta < \frac{\epsilon}{2}$; (2) aproximar o ponto p' por um ponto racional q' de $\mathbb{R}^2$ com erro $\eta \leq \frac{\epsilon}{2} - \delta$; e (3) computar $q = \psi^{-1}(q')$, com aritmética exata.

Pelo lema 15, temos que q é um ponto de $\mathcal{A}$; portanto, pela fórmula (7.1), podemos concluir que $Sdist(p, q) \leq \epsilon$.

Visto que $\mathcal{A} \subset \mathcal{B}$, o corolário 16 implica que $\mathcal{B}$ é um subconjunto denso de $\mathbb{S}^2$ (corolário 14). No entanto, a aproximação por pontos de $\mathcal{A}$ fornecida pelo algoritmo AAprox requer o dobro de bits que a aproximação por pontos de $\mathcal{B}$ fornecida pelo algoritmo BAprox (lema 12).

Corolário 17 *Para todo ponto p de $\mathbb{S}^2$ e todo real $\epsilon > 0$, existe um ponto $q = [q_0, .. q_3]$ em $\mathcal{A}$ tal que $Sdist(p, q) < \epsilon$, e cujas coordenadas inteiras q_0, q_1, q_2 e q_3 têm no máximo $2 \left\lfloor \log_2 \frac{1}{\epsilon} \right\rfloor + 3$ bits.*

DEMONSTRAÇÃO: É fácil ver que para todo ponto p do plano $\mathbb{R}^2$ e todo $\epsilon > 0$ existe um ponto racional $q' = \left(\frac{x}{w}, \frac{y}{w}\right)$ tal que $dist(p, q') \leq \epsilon$ e x, y, e w são inteiros com $\left\lfloor \log_2 \frac{1}{\epsilon} \right\rfloor + 1$ bits. Visto que ψ^{-1} é uma função quadrática, podemos concluir que as coordenadas inteiras do ponto $q = \psi^{-1}(q')$, eliminando os fatores comuns, têm $2 \left\lfloor \log_2 \frac{1}{\epsilon} \right\rfloor + 3$ bits. □

7.3 Planos e S-círculos racionais

Dizemos que um plano $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ é *racional* se todos os coeficientes α_i são números racionais. Por sua vez, um S-círculo é dito *racional* se o seu plano de suporte é um plano racional.

Visto que um plano α pode ser representado por qualquer quádrupla $\langle \lambda\alpha_0, \lambda\alpha_1, \lambda\alpha_2, \lambda\alpha_3 \rangle$ com $\lambda > 0$, então todo plano racional pode ser representado por um quádrupla de coeficientes inteiros. Portanto, podemos dizer que um plano é racional se ele possui coeficientes homogêneos inteiros. O mesmo se aplica, obviamente, aos S-círculos racionais.

7.3.1 Construção de um S-círculo racional com centro em $\mathcal{B}$

Vamos agora mostrar que dado um ponto $p = [p_0, .. p_3]$ em $\mathcal{B}$ e um raio $\theta \in (0 \_\_ \pi)$, podemos sempre obter um S-círculo racional com S-centro no ponto dado e cujo S-raio é suficientemente próximo de θ. Isto é,

Lema 18 *Para todo ponto $p = [p_0, .. p_3]$ de $\mathcal{B}$ com coordenadas p_1, p_2 e p_3 inteiras, todo $\theta \in (0 \_\_ \pi)$ e todo $\epsilon > 0$, seja $\sigma = \left\lceil \frac{18}{\epsilon^2 p_0} \right\rceil$. Então, $b = (\!(-\lfloor \sigma p_0 \cos\theta \rfloor, \sigma p_1, \sigma p_2, \sigma p_3)\!)$ é um Scirculo racional tal que $sctr(b) = p$ e $|srad(b) - \theta| < \epsilon$.*

DEMONSTRAÇÃO: Sem perda de generalidade podemos supor que $\theta \leq \frac{\pi}{2}$, pois caso contrário basta provar o lema para o ponto $\overline{p} = [p_0, -p_1, -p_2, -p_3]$ e raio $\overline{\theta} = \pi - \theta$, obtendo assim um S-círculo racional $a = (\!(a_0, .. a_3)\!)$ tal que $sctr(a) = \overline{p}$ e $\left|srad(a) - \overline{\theta}\right| < \epsilon$; daí, basta observar que $sctr(\neg a) = p$ e $|srad(\neg a) - \theta| = \left|srad(a) - \overline{\theta}\right|$.

Note que σ é inteiro e $\sigma \geq 1$. Visto que $p \in \mathcal{B}$, então $p_0 = \sqrt{p_1^2 + p_2^2 + p_3^2}$; como p_1, p_2 e p_3 são inteiros e não todos nulos, então $p_0 \geq 1$. Obviamente, os coeficientes de b são inteiros. Além disso, com a suposição de que $\theta \leq \frac{\pi}{2}$, temos que $\sigma p_0 \cos\theta \geq 0$. Portanto, $b_0 = -\lfloor \sigma p_0 \cos\theta \rfloor = -\lfloor \sigma p_0 \cos\theta \rfloor$; daí

$$b_0^2 = \lfloor \sigma p_0 \cos\theta \rfloor^2 \leq \sigma^2 p_0^2 \cos^2\theta \leq \sigma^2 p_0^2 = \sigma^2 (p_1^2 + p_2^2 + p_3^2)$$

o que significa que b é um S-círculo, obviamente racional, com $sctr(b) = p$.
Agora, seja $\varphi = srad(b)$; então

$$\varphi = \arccos\left(\frac{\lfloor \sigma p_0 \cos\theta \rfloor}{\sigma p_0}\right) = \arccos\left(\frac{\lfloor \sigma p_0 \cos\theta \rfloor}{\sigma p_0}\right)$$

Visto que $\lfloor \sigma p_0 \cos\theta \rfloor > \sigma p_0 \cos\theta - 1$, então

$$\cos\varphi = \frac{\lfloor \sigma p_0 \cos\theta \rfloor}{\sigma p_0} > \frac{(\sigma p_0 \cos\theta - 1)}{\sigma p_0} = \cos\theta - \frac{1}{\sigma p_0}$$

Por outro lado, visto que $\lfloor \sigma p_0 \cos\theta \rfloor \leq \sigma p_0 \cos\theta$, então

$$\cos\varphi = \frac{\lfloor \sigma p_0 \cos\theta \rfloor}{\sigma p_0} \leq \frac{\sigma p_0 \cos\theta}{\sigma p_0} = \cos\theta$$

Logo,

$$0 \leq \cos\theta - \cos\varphi < \frac{1}{\sigma p_0}$$

Visto que $\sigma \geq 1$ e $p_0 \geq 1$, então $\frac{1}{\sigma p_0} \leq 1$, o que implica que

$$(\cos\theta - \cos\varphi)^2 < \left(\frac{1}{\sigma p_0}\right)^2 \leq \frac{1}{\sigma p_0} \tag{7.2}$$

Como φ é um S-raio, então tanto θ como φ pertencem ao intervalo $(0 \_\_ \pi)$, e portanto $\sin\theta \geq 0$ e $\sin\varphi \geq 0$. Daí,

$$\begin{aligned}
(\sin\theta - \sin\varphi)^2 &= |\sin\theta - \sin\varphi|\,|\sin\theta - \sin\varphi| \leq |\sin\theta + \sin\varphi|\,|\sin\theta - \sin\varphi| \\
&= \left|\sin^2\theta - \sin^2\varphi\right| = |1 - \cos^2\theta - (1 - \cos^2\varphi)| \\
&= |\cos^2\varphi - \cos^2\theta| = |\cos\varphi + \cos\theta|\,|\cos\varphi - \cos\theta| \\
&= \leq 2\,|\cos\varphi - \cos\theta| < 2\tfrac{1}{\sigma p_0}
\end{aligned} \tag{7.3}$$

De (7.2) e (7.3) temos que

$$\begin{aligned}
\cos^2\theta + \cos^2\varphi - 2\cos\theta\cos\varphi &= (\cos\theta - \cos\varphi)^2 < \tfrac{1}{\sigma p_0} \\
\sin^2\theta + \sin^2\varphi - 2\sin\theta\sin\varphi &= (\sin\theta - \sin\varphi)^2 < \tfrac{2}{\sigma p_0}
\end{aligned}$$

Somando estas duas desigualdades obtemos

$$2 - 2(\cos\theta\cos\varphi + \sin\theta\sin\varphi) < \frac{3}{\sigma p_0}$$

o que implica que

$$\cos(\theta - \varphi) > 1 - \frac{3}{2\sigma p_0}$$

Agora, usando a expansão de Taylor da função $\cos(x)$, temos que

$$\cos(\theta - \varphi) \leq 1 - \frac{(\theta - \varphi)^2}{2} + \frac{(\theta - \varphi)^4}{24}$$

Portanto

$$\frac{(\theta - \varphi)^2}{2} - \frac{(\theta - \varphi)^4}{24} < \frac{3}{2\sigma p_0}$$

Por outro lado, podemos mostrar que para todo x no intervalo $[-\sqrt{10} \_\_ \sqrt{10}\,]$

$$\frac{x^2}{12} \leq \frac{x^2}{2} - \frac{x^4}{24}$$

Novamente, visto que θ e φ pertencem ao intervalo $(0 \_\_ \pi)$, então $(\theta - \varphi) \in (-\pi \_\_ \pi) \subset [-\sqrt{10} \_\_ \sqrt{10}\,]$. Portanto,

$$\frac{(\theta-\varphi)^2}{12} \leq \frac{(\theta-\varphi)^2}{2} \_\_ \frac{(\theta-\varphi)^4}{24} < \frac{3}{2\sigma p_0}$$

Logo,

$$|srad(a) - srad(b)| = |\theta - \varphi| < \sqrt{\frac{18}{\sigma p_0}} \leq \epsilon$$

□

A partir do lema 18, podemos concluir que o número de bits necessário para construir o S-círculo desejado é proporcional à precisão utilizada. Ou seja,

Corolário 19 *Para todo ponto* $p = [p_0, .. p_3]$ *de* $\mathcal{B}$ *com coordenadas* p_1, p_2 *e* p_3 *inteiras de* m *bits, todo* $\theta \in (0 \_\_ \pi)$ *e todo* $\epsilon > 0$, *existe um S-círculo racional* b *tal que* $sctr(b) = p$, $|srad(b) - \theta| < \epsilon$ *e cujos coeficientes são inteiros com* $\max\{m, 2\left\lfloor \log \frac{1}{\epsilon} \right\rfloor\} + 6$ *bits.*

Além disso, o mesmo lema 18 nos permite concluir que todo S-círculo com S-centro num ponto de $\mathcal{B}$ pode ser adequadamente aproximado por um S-círculo racional; mais precisamente,

Corolário 20 *Para todo S-círculo* $a = ((a_0, a_1, a_2, a_3))$ *tal que* a_1, a_2 *e* a_3 *são inteiros e todo real* $\epsilon > 0$, *sejam* $\kappa = \sqrt{a_1^2 + a_2^2 + a_3^2}$ *e* $\sigma = \left\lceil \frac{18}{\epsilon^2 \kappa} \right\rceil$ *e seja* $b = ((\lfloor \sigma a_0 \rfloor, \sigma a_1, \sigma a_2, \sigma a_3))$. *Então* b *é um S-círculo racional tal que* $sctr(b) = sctr(a)$ *e* $|srad(a) - srad(b)| < \epsilon$.

7.4 Aproximação racional de S-círculos genéricos

Vamos agora mostrar que um S-círculo genérico (com S-centro e S-raio quaisquer) pode ser aproximado por um S-círculo racional com uma precisão arbitrária.

Para comparar os S-círculos usaremos o seguinte conceito:

Definição 16 Para quaisquer dois conjuntos A e B de $\mathbb{R}^n$, a *distância de Hausdorff* entre A e B, denotada por $Hdist(A, B)$, é dada por

$$Hdist(A,B) = \max\left\{\sup_{x\in A}\{\inf_{y\in B} dist(x,y)\},\ \sup_{y\in B}\{\inf_{x\in A} dist(x,y)\}\right\}$$

Mostra-se [19] que $Hdist$ é uma distância sobre o conjunto dos subconjuntos compactos de $\mathbb{R}^n$.

Para obter a aproximação dos S-círculos genéricos, utilizaremos o seguinte lema:

Lema 21 *Dados dois S-círculos a e b, a distância de Hausdorff entre estes dois S-círculos é tal que*

$$Hdist(a,b) \leq Sdist(sctr(a), sctr(b)) + |srad(a) - srad(b)|$$

DEMONSTRAÇÃO: Seja c um S-círculo com o mesmo S-centro que b e com o mesmo S-raio que a. Pela desigualdade triangular temos que

$$Hdist(a,b) \leq Hdist(a,c) + Hdist(c,b)$$

Visto que a e c têm o mesmo S-raio, é fácil verificar que $Hdist(a,c) \leq Sdist(sctr(a), sctr(c))$. Analogamente, visto que b e c têm o mesmo S-centro, temos que $Hdist(c,b) = |srad(c) - srad(b)| = |srad(a) - srad(b)|$. Logo,

$$Hdist(a,b) \leq Sdist(sctr(a), sctr(b)) + |srad(a) - srad(b)|$$

□

O lema 21 nos permite combinar os lemas 12 e 18 para obter o seguinte resultado:

Corolário 22 *Para todo S-círculo a e todo $\epsilon > 0$, sejam $p = (x,y,z) = sctr(a)$, $\theta = srad(a)$, $\eta \geq \sqrt{3} \cdot \max\{1, \frac{\pi}{\epsilon}\}$, $\kappa = \sqrt{\{\eta x\}^2 + \{\eta y\}^2 + \{\eta z\}^2}$ e $\sigma = \lceil \frac{72}{\epsilon^2 \kappa} \rceil$. Então,*

$$b = ((-\{\sigma\kappa\cos\theta\}, \sigma\{\eta x\}, \sigma\{\eta y\}, \sigma\{\eta z\}))$$

é um S-círculo racional tal que $Hdist(a,b) < \epsilon$.

DEMONSTRAÇÃO: Dados $\eta \geq \sqrt{3} \cdot \max\{1, \frac{\pi}{\epsilon}\}$ e $\kappa = \sqrt{\{\eta x\}^2 + \{\eta y\}^2 + \{\eta z\}^2}$, pelo lema 12, temos que $q = [\kappa, \{\eta x\}, \{\eta y\}, \{\eta z\}]$ é um ponto de $\mathcal{B}$ tal que $Sdist(p,q) = Sdist(sctr(a), q) \leq \frac{\epsilon}{2}$. Agora, pelo lema 18, dado $\sigma = \lceil \frac{72}{\epsilon^2 \kappa} \rceil$, temos que $b = ((-\{\sigma\kappa\cos\theta\}, \sigma\{\eta x\}, \sigma\{\eta y\}, \sigma\{\eta z\}))$ é um S-círculo racional tal que $sctr(b) = q$ e $|srad(b) - \theta| < \frac{\epsilon}{2}$.

Portanto, pelo lema 21,

$$Hdist(a,b) \leq Sdist(sctr(a), sctr(b)) + |srad(a) - srad(b)| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon$$

□

A partir deste resultado podemos concluir que

Corolário 23 *Para todo S-círculo a e todo real $\epsilon > 0$, existe um S-círculo racional $b = ((b_0, .. b_3))$ tal que $Hdist(a,b) \leq \epsilon$ e cujos coeficientes são inteiros com no máximo $2\lfloor \log_2 \frac{1}{\epsilon} \rfloor + 9$ bits.*

Note que, assim como nos casos anteriores, o número de bits necessário para aproximar um S-círculo genérico por um S-círculo racional é proporcional à precisão desejada.

Finalmente, temos que

Corolário 24 *O conjunto dos S-círculos racionais é denso no conjunto de todos os S-círculos.*

7.5 P-círculos racionais

Um P-círculo — isto é, um círculo no plano $\mathbb{R}^2$ — $a = [\![a_0, a_1, a_2, a_3]\!]$ é dito *racional* se os seus coeficientes $a_0, \cdots, a_3$ são racionais; equivalentemente, inteiros. Há uma estreita correspondência entre os círculos racionais de $\mathbb{R}^2$ e de $\mathbb{S}^2$:

Lema 25 *A projeção estereográfica canônica ψ é uma bijeção entre os S-círculos racionais e os P-círculos racionais.*

Capítulo 8

Interseção exata de S-círculos

Como o leitor deve se lembrar, o nosso objetivo é elaborar uma representação da geometria dos mapas esféricos que além de ser exata também seja fechada para a operação de sobreposição.

A representação descrita no capítulo anterior nos permite aproximar adequadamente os elementos (vértices e arestas) de um mapa esférico por elementos cuja geometria pode ser representada exatamente. Mas, para que a operação de sobreposição possa ser realizada de maneira exata é fundamental que o ponto de interseção $a \wedge b$ entre dois S-círculos racionais a e b também possa ser representado exatamente.

Infelizmente, como podemos verificar através do exemplo a seguir, mesmo que a e b sejam S-círculos racionais, o ponto $a \wedge b$ pode não estar contido nos conjuntos $\mathcal{A}$ e $\mathcal{B}$ que definimos no capítulo anterior: por exemplo, se $a = (\!(1, 2, 2, 2)\!)$ e $b = (\!(1, 2, -2, 2)\!)$, então

$$a \wedge b = \left[4, -1 + \sqrt{7}, 0, -1 - \sqrt{7}\right]$$

que não é um ponto do conjunto $\mathcal{B}$, pois $(-1 + \sqrt{7})/(-1 - \sqrt{7})$ não é um número racional. Para contornar este problema, vamos estender a representação exata de pontos de $\mathbb{S}^2$, definindo um conjunto (maior) de pontos de $\mathbb{S}^2$ com representação exata que inclua também os pontos de interseção entre dois S-círculos racionais.

8.1 Interseção entre S-círculos pela reta em comum aos planos de suporte

Sejam a e b dois S-círculos e sejam α e β os seus respectivos planos de suporte, isto é, $\alpha = spln(a)$ e $\beta = spln(b)$. Além disso, seja l a reta de interseção entre estes planos, isto é, $l = \alpha \wedge \beta$. É fácil perceber que os pontos de interseção entre os S-círculos a e b corresponde aos pontos de interseção entre a reta l e a esfera $\mathbb{S}^2$. Veja figura 8.1.

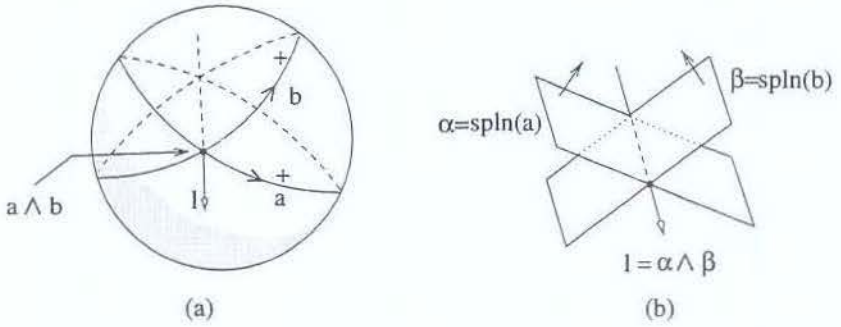


Figura 8.1: Interseção entre dois S-círculos e entre os seus planos de suporte.

Portanto, os pontos de interseção entre dois S-círculos podem ser identificados pela reta de interseção entre seus planos de suporte, sendo que a orientação desta reta nos permite distinguir de maneira não ambígua o ponto de interseção canônica $a \wedge b$ do outro ponto de interseção $a \wedge (\neg b)$ — caso eles sejam distintos.

Vamos denotar por $ent(l)$ o ponto do aquém onde a reta l "entra" em $\mathbb{S}^2$, isto é, deixa o lado positivo ("exterior") de $\mathbb{S}^2$; e por $ext(l)$, o ponto do aquém onde a reta l "sai" de $\mathbb{S}^2$, isto é, entra no lado positivo de $\mathbb{S}^2$. Além disso, é conveniente denotar por $mid(l)$ o ponto médio entre estes dois pontos, isto é, $mid(l) = l \wedge bisect(l)$, o que corresponde ao ponto de l mais próximo da origem O. Veja a figura 8.2.

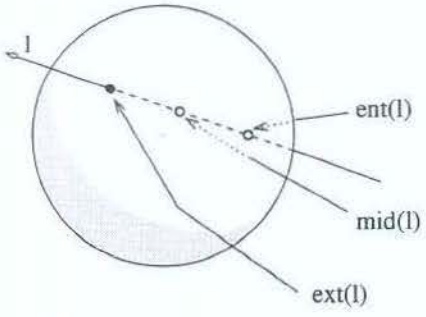


Figura 8.2: Os pontos $ent(l)$, $mid(l)$ e $ext(l)$.

Como podemos verificar, para toda reta $l = \langle l_0, .. l_5 \rangle$ que intercepta $\mathbb{S}^2$

$$\begin{aligned} ent(l) &= [\,\mu,\ -l_1l_2 - l_3l_4 - l_5\sqrt{\delta},\ l_0l_2 - l_3l_5 + l_4\sqrt{\delta},\ l_0l_4 + l_1l_5 - l_2\sqrt{\delta}\,] \\ mid(l) &= [\,\mu,\ -l_1l_2 - l_3l_4,\ l_0l_2 - l_3l_5,\ l_0l_4 + l_1l_5\,] \\ ext(l) &= [\,\mu,\ -l_1l_2 - l_3l_4 + l_5\sqrt{\delta},\ l_0l_2 - l_3l_5 - l_4\sqrt{\delta},\ l_0l_4 + l_1l_5 + l_2\sqrt{\delta}\,] \end{aligned} \tag{8.1}$$

onde $\mu = l_2^2 + l_4^2 + l_5^2$ e $\delta = \mu - (l_0^2 + l_1^2 + l_3^2)$.

Pela fórmula (5.5), uma reta l é tangente à $\mathbb{S}^2$ se e somente se $l_2^2 + l_4^2 + l_5^2 = l_0^2 + l_1^2 + l_3^2$. Note pela fórmula (8.1), que isto implica que $ent(l) = mid(l) = ext(l)$ — o que está de

acordo com a definição destes pontos.

Portanto, considerando a definição de interseção canônica entre dois S-círculos (seção 6.5.2), podemos concluir que

Teorema 26 *Sejam a e b dois S-círculos, seja $l = spln(a) \wedge spln(b)$ e seja $\delta = (l_2^2 + l_4^2 + l_5^2) - (l_0^2 + l_1^2 + l_3^2)$. Então,*

$$a \wedge b = ext(l)$$

se e somente se $\delta > 0$, ou $\delta = 0$ e $dctr(a) \diamond spln(b) = +1$. Caso contrário, $a \wedge b = [\mathbf{0}]$.

8.2 Pontos sub-racionais de $\mathbb{S}^2$

De acordo com as equações (5.4), a interseção entre dois planos racionais é uma reta com coeficientes de Plücker racionais; isto é, uma *reta racional*. Por outro lado, pelo teorema 26, a interseção canônica entre dois S-círculos pode ser representada pelos coeficientes de Plücker da reta $spln(a) \wedge spln(b)$. Portanto, a interseção canônica entre dois S-círculos racionais pertence ao conjunto

$$\mathcal{C} = \{\, ext(l) \mid l \text{ é uma reta racional} \,\}$$

que é denominado o conjunto dos pontos *sub-racionais* de $\mathbb{S}^2$.

Assim, todo ponto $p \in \mathcal{C}$ pode ser representado exatamente pelos seis coeficientes racionais — equivalentemente inteiros — de alguma reta racional l tal que $ext(l) = p$. Neste caso, se $l = \langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle$, vamos denotar

$$ext(l) = \langle\!\langle l_0, l_1, l_2, l_3, l_4, l_5 \rangle\!\rangle$$

É fácil ver que $\mathcal{A} \subset \mathcal{B} \subset \mathcal{C}$. O relacionamento entre os conjuntos $\mathcal{A}$ e $\mathcal{C}$ é estabelecido mais precisamente pelos seguintes teoremas:

Teorema 27 *Sejam $\langle l_0, .. l_5 \rangle$ os coeficientes inteiros de uma reta racional l. Então $ext(l)$ pertence a $\mathcal{A}$ se e somente se $(l_2^2 + l_4^2 + l_5^2) - (l_0^2 + l_1^2 + l_3^2)$ é um quadrado perfeito.*

DEMONSTRAÇÃO: Se $\delta = (l_2^2 + l_4^2 + l_5^2) - (l_0^2 + l_1^2 + l_3^2)$ é um quadrado perfeito, então as coordenadas de $ext(l)$ na fórmula (8.1) são números inteiros. Portanto, $ext(l) \in \mathcal{A}$.

Reciprocamente, suponha que δ não é um quadrado perfeito. Visto que l intercepta $\mathbb{S}^2$, então pela fórmula (5.5), podemos concluir que $(l_2, l_4, l_5) \neq (0, 0, 0)$. Daí, pela fórmula (8.1), pelo menos uma das coordenadas homogêneas de $ext(l)$ é da forma $\alpha + l_i\sqrt{\delta}$, com α racional e $l_i \neq 0$. Como $\mu = l_2^2 + l_4^2 + l_5^2$ é um número racional, então a razão $(\alpha + l_i\sqrt{\delta})/\mu$ não é um número racional. Logo, $ext(l)$ não é um ponto racional. □

Corolário 28 *Para toda reta racional l, $ext(l) \in \mathcal{A}$ se e somente se $ent(l) \in \mathcal{A}$.*

Teorema 29 *Um ponto p pertence a $\mathcal{A}$ se e somente se existem pelo menos duas retas racionais distintas l e m tais que $ext(l) = p = ext(m)$.*

DEMONSTRAÇÃO: Inicialmente, suponha que $p \in \mathcal{A}$. Sejam q_1 e q_2 dois pontos distintos de $\mathcal{A}$ tais que $q_1 \neq p \neq q_2$. Então, as duas retas $l = q_1 \vee p$ e $m = q_2 \vee p$ são racionais, são distintas (pois não há três pontos colineares em $\mathbb{S}^2$), e $ext(l) = p = ext(m)$.

Reciprocamente, sejam l e m duas retas racionais distintas tais que $ext(l) = p = ext(m)$. Se $l \neq \neg m$ então, pela fórmula (5.9), podemos concluir que $p = l \cap m$ é um ponto racional. Por outro lado, se $l = \neg m$ então o fato de que $ext(l) = ext(m)$ implica que l e m são tangentes a $\mathbb{S}^2$. Neste caso, $p = mid(l)$ e portanto, pela fórmula (8.1), p é um ponto racional. □

Corolário 30 *Um ponto p pertence a $\mathcal{C} \setminus \mathcal{A}$ se e somente se existe uma única reta racional l tal que $p = ext(l)$.*

Corolário 31 *Um ponto p pertence a $\mathcal{B} \setminus \mathcal{A}$ se e somente se $l = O \vee p$ é a única reta racional tal que $p = ext(l)$.*

DEMONSTRAÇÃO: Para todo $p = [p_0, p_1, p_2, p_3] \in \mathbb{S}^2$, seja $l = O \vee p = \langle 0, 0, p_3, 0, -p_2, p_1 \rangle$. É fácil ver que $p = ext(l)$. Se $p \in \mathcal{B} \setminus \mathcal{A}$, então p_1, p_2 e p_3 são números inteiros, o que implica que l é racional; além disso, pelo corolário 30, a reta l é única. Reciprocamente, se l é uma reta racional, então $dir(l) = [0, p_1, p_2, p_3]$ é uma direção racional e portanto $p \in \mathcal{B}$. Além disso, se l é única, pelo corolário 30, $p \notin \mathcal{A}$. □

Teorema 32 *Um ponto p de $\mathbb{R}^3$ pertence a $\mathcal{C}$ se e somente se*

$$p = \left[a_0, a_1 + b_1\sqrt{c}, a_2 + b_2\sqrt{c}, a_3 + b_3\sqrt{c}\,\right] \tag{8.2}$$

onde a_0, a_1, a_2, a_3, b_1, b_2, b_3 e c são números inteiros que satisfazem às seguintes condições:

(i) $a_0 = b_1^2 + b_2^2 + b_3^2 \neq 0$

(ii) $a_1 b_1 + a_2 b_2 + a_3 b_3 = 0$

(iii) $a_0(a_0 - c) = a_1^2 + a_2^2 + a_3^2$

Neste caso, $p = \langle\!\langle a_2 b_3 - a_3 b_2,\ a_3 b_1 - a_1 b_3,\ a_0 b_3,\ a_1 b_2 - a_2 b_1,\ -a_0 b_2,\ a_0 b_1 \rangle\!\rangle$

DEMONSTRAÇÃO: Inicialmente, vamos mostrar que todo ponto $p \in \mathcal{C}$ é da forma (8.2), sendo que os inteiros a_0, a_1, a_2, a_3, b_1, b_2, b_3 e c satisfazem as condições (i) — (iii).

Seja $\langle l_0, .. \, l_5 \rangle$ os coeficientes de Plücker inteiros de uma reta racional l tal que $ext(l) = p$. Tomando-se

$$\begin{array}{rclcrcl} a_0 &=& l_2^2 + l_4^2 + l_5^2 & & b_1 &=& l_5 \\ a_1 &=& -l_1 l_2 - l_3 l_4 & & b_2 &=& -l_4 \\ a_2 &=& l_0 l_2 - l_3 l_5 & & b_3 &=& l_2 \\ a_3 &=& l_0 l_4 + l_1 l_5 & & c &=& a_0 - (l_0^2 + l_1^2 + l_3^2) \end{array} \tag{8.3}$$

temos que

(i) $a_0 = l_5^2 + l_4^2 + l_2^2 = b_1^2 + b_2^2 + b_3^2$; e $a_0 \neq 0$, pois de acordo com a fórmula (5.5), se $l_2 = l_4 = l_5 = 0$ então l seria uma reta no infinito.

(ii) $a_1 b_1 + a_2 b_2 + a_3 b_3 = l_5(-l_1 l_2 - l_3 l_4) - l_4(l_0 l_2 - l_3 l_5) + l_2(l_0 l_4 + l_1 l_5) = 0$.

(iii) $$\begin{array}{rclcl} a_1^2 &=& (-l_1 l_2 - l_3 l_4)^2 &=& l_1^2 l_2^2 + l_3^2 l_4^2 + l_1 l_2 l_3 l_4 + l_1 l_2 l_3 l_4 \\ a_2^2 &=& (l_0 l_2 - l_3 l_5)^2 &=& l_0^2 l_2^2 + l_3^2 l_5^2 - l_0 l_2 l_3 l_5 - l_0 l_2 l_3 l_5 \\ a_3^2 &=& (l_0 l_4 + l_1 l_5)^2 &=& l_0^2 l_4^2 + l_1^2 l_5^2 + l_0 l_1 l_4 l_5 + l_0 l_1 l_4 l_5 \end{array}$$

Daí,

$$\begin{array}{rcl} a_1^2 + a_2^2 + a_3^2 &=& l_0^2(l_2^2 + l_4^2) + l_1^2(l_2^2 + l_5^2) + l_3^2(l_4^2 + l_5^2) + \\ & & + l_2 l_3(l_1 l_4 - l_0 l_5) + l_0 l_5(l_1 l_4 - l_2 l_3) + l_1 l_4(l_2 l_3 + l_0 l_5) \end{array}$$

Aplicando a condição de Plücker (5.3) em cada um dos três últimos termos, obtemos

$$\begin{array}{rcl} a_1^2 + a_2^2 + a_3^2 &=& l_0^2(l_2^2 + l_4^2) + l_1^2(l_2^2 + l_5^2) + l_3^2(l_4^2 + l_5^2) + \\ & & + l_2 l_3(l_2 l_3) + l_0 l_5(l_0 l_5) + l_1 l_4(l_1 l_4) \\ &=& l_0^2(l_2^2 + l_4^2 + l_5^2) + l_1^2(l_2^2 + l_4^2 + l_5^2) + l_3^2(l_2^2 + l_4^2 + l_5^2) \\ &=& (l_0^2 + l_1^2 + l_3^2)(l_2^2 + l_4^2 + l_5^2) \\ &=& a_0(a_0 - c) \end{array}$$

o que mostra que os inteiros a_0, a_1, a_2, a_3, b_1, b_2, b_3 e c satisfazem as condições (i) — (iii). Além disso, da fórmula (8.1), sabemos que

$$ext(l) = \left[\mu, -l_1 l_2 - l_3 l_4 + l_5\sqrt{\delta}, l_0 l_2 - l_3 l_5 - l_4\sqrt{\delta}, l_0 l_4 + l_1 l_5 + l_2\sqrt{\delta}\right] \tag{8.4}$$

onde $\mu = l_2^2 + l_4^2 + l_5^2 = a_0^2 b_3^2 + a_0^2 b_2^2 + a_0^2 b_1^2 = a_0^3$ e $\delta = \mu - (l_0^2 + l_1^2 + l_3^2)$.
Portanto, aplicando as respectivas equações dadas em (8.3) obtemos que

$$p = ext(l) = \left[a_0, a_1 + b_1\sqrt{c}, a_2 + b_2\sqrt{c}, a_3 + b_3\sqrt{c}\,\right]$$

o que mostra que p é da forma (8.2).

Reciprocamente, sejam a_0, a_1, a_2, a_3, b_1, b_2, b_3 e c números inteiros que satisfazem (i) — (iii) e seja

$$l = \langle\, a_2b_3 - a_3b_2,\ a_3b_1 - a_1b_3,\ a_0b_3,\ a_1b_2 - a_2b_1,\ -a_0b_2,\ a_0b_1 \,\rangle$$

Pela condição (i), $a_0 \neq 0$ e existe $i \in \{1,2,3\}$ tal que $b_i \neq 0$. Desta forma, $a_0b_i \neq 0$, o que implica que pelo menos um coeficiente de l é diferente de zero. Visto que

$$l_0l_5 - l_1l_4 + l_2l_3 = a_0b_1(a_2b_3 - a_3b_2) - (-a_0b_2)(a_3b_1 - a_1b_3) + a_0b_3(a_1b_2 - a_2b_1) = 0$$

então a condição de Plücker (5.3) é satisfeita, o que mostra que a sêxtupla l define uma reta, obviamente racional.

Daí, pela fórmula (8.4),

$$\begin{aligned} l_0^2 + l_1^2 + l_3^2 &= (a_2b_3 - a_3b_2)^2 + (a_3b_1 - a_1b_3)^2 + (a_1b_2 - a_2b_1)^2 \\ &= a_2^2b_3^2 + a_3^2b_2^2 - 2a_2a_3b_2b_3 + a_1^2b_3^2 + a_3^2b_1^2 - 2a_1a_3b_1b_3 + \\ &\quad + a_1^2b_2^2 + a_2^2b_1^2 - 2a_1a_2b_1b_2 \\ &= a_1^2(b_2^2 + b_3^2) + a_2^2(b_1^2 + b_3^2) + a_3^2(b_1^2 + b_2^2) - \\ &\quad - 2a_1a_2b_1b_2 - 2a_1a_3b_1b_3 - 2a_2a_3b_1b_3 \end{aligned}$$

Usando a condição (ii) temos $(a_1b_1 + a_2b_2 + a_3b_3)^2 = 0$, o que implica que $a_1^2b_1^2 + a_2^2b_2^2 + a_3^2b_3^2 = -2a_1a_2b_1b_2 - 2a_1a_3b_1b_3 - 2a_2a_3b_2b_3$. Portanto,

$$\begin{aligned} l_0^2 + l_1^2 + l_3^2 &= a_1^2(b_2^2 + b_3^2) + a_2^2(b_1^2 + b_3^2) + a_3^2(b_1^2 + b_2^2) + a_1^2b_1^2 + a_2^2b_2^2 + a_3^2b_3^2 \\ &= a_1^2(b_1^2 + b_2^2 + b_3^2) + a_2^2(b_1^2 + b_2^2 + b_3^2) + a_3^2(b_1^2 + b_2^2 + b_3^2) \\ &= (a_1^2 + a_2^2 + a_3^2)(b_1^2 + b_2^2 + b_3^2) \\ &= a_0(a_1^2 + a_2^2 + a_3^2) \end{aligned}$$

De onde obtemos que

$$\delta = \mu - (l_0^2 + l_1^2 + l_3^2) = a_0^3 - a_0(a_1^2 + a_2^2 + a_3^2) = a_0(a_0^2 - a_1^2 - a_2^2 - a_3^2)$$

Além disso,

$$\begin{aligned} -l_1l_2 - l_3l_4 &= -(a_3b_1 - a_1b_3)a_0b_3 - (a_1b_2 - a_2b_1)(-a_0b_2) \\ &= -a_0a_3b_1b_3 + a_0a_1b_3^2 + a_0a_1b_2^2 - a_0a_2b_1b_2 \\ &= -a_0b_1(a_3b_3 + a_2b_2) + a_0a_1(b_2^2 + b_3^2) \\ &= a_0a_1b_1^2 + a_0a_1(b_2^2 + b_3^2) \\ &= a_0a_1(b_1^2 + b_2^2 + b_3^2) = a_0^2a_1 \end{aligned}$$

Desta forma, temos que

$$\begin{aligned} -l_1l_2 - l_3l_4 - l_5\sqrt{\delta} &= a_0^2a_1 + a_0b_1\sqrt{a_0(a_0^2 - a_1^2 - a_2^2 - a_3^2)} \\ &= a_0^2\left(a_1 + b_1\sqrt{\frac{a_0(a_0^2 - a_1^2 - a_2^2 - a_3^2)}{a_0^2}}\right) \\ &= a_0^2(a_1 + b_1\sqrt{c}) \end{aligned}$$

Analogamente, a terceira e a quarta coordenadas homogêneas do ponto $ext(l)$ na fórmula (8.4) são respectivamente

$$\begin{aligned} l_0l_2 - l_3l_5 - l_4\sqrt{\delta} &= a_0^2(a_2 + b_2\sqrt{c}) \\ l_0l_4 + l_1l_5 + l_2\sqrt{\delta} &= a_0^2(a_3 + b_3\sqrt{c}) \end{aligned}$$

Daí,

$$ext(l) = [a_0^3, a_0^2(a_1 + b_1\sqrt{c}), a_0^2(a_2 + b_2\sqrt{c}), a_0^2(a_3 + b_3\sqrt{c})]$$

Finalmente, visto que $a_0 \neq 0$, pela condição (i), então

$$ext(l) = \left[a_0,\ a_1 + b_1\sqrt{c},\ a_2 + b_2\sqrt{c},\ a_3 + b_3\sqrt{c}\right] = p$$

Portanto, existe uma reta racional l tal que $p = ext(l)$; logo, $p \in \mathcal{C}$.

Note que a segunda parte da demonstração estabelece que se os inteiros a_0, a_1, a_2, a_3, b_1, b_2, b_3 e c satisfazem as condições (i) — (iii) então

$$p = \langle\langle a_2b_3 - a_3b_2,\ a_3b_1 - a_1b_3,\ a_0b_3,\ a_1b_2 - a_2b_1,\ -a_0b_2,\ a_0b_1\rangle\rangle$$

□

8.3 Representação computacional

Antes de iniciarmos as descrições dos algoritmos de manipulação de mapas esféricos, precisamos definir uma representação computacional para a geometria dos elementos do mapa.

8.3.1 Representação canônica dos pontos de $\mathcal{C}$

Pelo corolário 30, se um ponto p pertence a $(\mathcal{C} \setminus \mathcal{A})$, então existe uma única reta racional l tal que $ext(l) = p$. Neste caso, vamos denotar esta reta por $stab(p)$. Por outro lado, se $p \in \mathcal{A}$ então há infinitas retas racionais l tais que $ext(l) = p$. Em particular, para qualquer

ponto racional q no lado negativo ("interior") de $\mathbb{S}^2$, a reta $q \vee p$ tem esta propriedade. Neste caso, definimos $stab(p)$ como sendo a reta $O \vee p = \langle 0, 0, p_3, 0, -p_2, p_1 \rangle$.

Para tornar os algoritmos geométricos mais simples, vamos supor que todo ponto $p \in \mathcal{C}$ é representado por uma sêxtupla de inteiros $\langle\langle p_0, .. p_5 \rangle\rangle$ onde $\langle p_0, .. p_5 \rangle$ são os coeficientes de Plücker da reta canônica $stab(p)$. Por convenção, estes coeficientes estarão sempre na sua forma *reduzida*; isto é, se $stab(p) = \langle p_0, .. p_5 \rangle$ então $mdc(p_0, \cdots, p_5) = 1$.

Em outras palavras, o ponto $p = \langle\langle p_0, .. p_5 \rangle\rangle$ de $\mathcal{C}$ está representado na sua forma canônica se e somente se $stab(p) = \langle p_0, .. p_5 \rangle$.

Vamos então supor que os pontos, retas, planos, S-círculos e S-arcos são representados pelos respectivos objetos definidos (em Modula-3) na figura 8.3.

8.3.2 Algoritmos para obtenção da representação canônica

Dado um ponto p de $\mathcal{C}$ e uma reta racional l tal que $ext(l) = p$, então a representação canônica do ponto p pode ser determinada pelos procedimentos a seguir. O procedimento AExt recebe como parâmetro uma reta l e verifica se o ponto $ext(l)$ pertence a $\mathcal{A}$; em caso afirmativo, retorna as coordenadas homogêneas de $ext(l)$; caso contrário, retorna $[\mathbf{0}] = [0, 0, 0, 0]$.

---

procedure AExt(l : Line) : APoint;
  $\mu \leftarrow l_2^2 + l_4^2 + l_5^2$;
  $\delta \leftarrow \mu - (l_0^2 + l_1^2 + l_3^2)$;
  if δ *é um quadrado perfeito* **then**
    return $[\mu, -l_1 l_2 - l_3 l_4 + l_5\sqrt{\delta}, l_0 l_2 - l_3 l_5 - l_4\sqrt{\delta},\ l_0 l_4 + l_1 l_5 + l_2\sqrt{\delta}\ ]$
  else return $[\mathbf{0}]$

O procedimento CExt recebe como parâmetro uma reta l e retorna a representação canônica do ponto $ext(l) \in \mathcal{C}$. Ele usa o procedimento auxiliar Reduce(l) que elimina os fatores comuns nos coeficientes de Plücker da reta l.

---

procedure CExt(l : Line) : CPoint;
  $p \leftarrow$ AExt(l);
  if $p = [\mathbf{0}]$ **then** $m \leftarrow$ Reduce(l)
  else $m \leftarrow$ Reduce($O \vee p$)
  return $\langle\langle m_0, .. m_5 \rangle\rangle$

Como se pode perceber, a representação canônica nos permite verificar facilmente se dois pontos p e q de $\mathcal{C}$ são iguais; para isto, basta verificar a igualdade dos coeficientes de Plücker das retas $stab(p)$ e $stab(q)$, que estão na forma reduzida.

```
APoint = object
    coord : array[0..3] of integer;
end;

BPoint = object
    coord : array[1..3] of integer;      {coordenada 0 é implícita}
end;

CPoint = object
    coeff : array[0..5] of integer;      {coeficientes da reta canônica}
end;

Line = object
    coeff : array[0..5] of integer;
end;

Plane = object
    coeff : array[0..3] of integer;
end;

SCircle = object
    coeff : array[0..3] of integer;
end;

SCurve = object
    org : CPoint
    scirc : SCircle
    dst : CPoint
end;

SArc = SCurve;      {Um S-arco é uma S-curva com org ≠ ∅ e dst ≠ ∅}
```

Figura 8.3: Definição em Modula-3 da representação geométrica.

8.3.3 Representação da geometria na estrutura SMC

Vamos agora estender a estrutura de dados SMC (descrita no capítulo 4), para incluir dados sobre a representação geométrica dos elementos do mapa.

A representação exata dos pontos sub-racionais (pontos de $\mathbb{C}$) descrita neste capítulo, nos permite representar exatamente o ponto de interseção entre dois S-círculos racionais. Portanto, se nos restringirmos apenas a mapas esféricos cujos vértices são pontos de $\mathbb{C}$ e cujas arestas são arcos de S-círculos racionais (ou o próprio S-círculo no caso de arestas ovais), então o mapa resultante da sobreposição de dois desses mapas também terá vértices que são pontos de $\mathbb{C}$ e arestas cujo S-círculo de suporte é racional; conseqüentemente, esta sobreposição também pode ser representada de maneira exata. Ou seja, adotando esta restrição, a representação exata para mapas passa a ser fechada para a operação de sobreposição. Note que esta restrição é perfeitamente aceitável, pois, como vimos no capítulo 7, todo ponto de $\mathbb{S}^2$ pode ser adequadamente aproximado por um ponto de $\mathbb{C}$ e todo S-círculo também pode ser adequadamente aproximado por um S-círculo raional.

A representação geométrica dos elementos do mapa será incorporada à estrutura SMC através dos campos *props* dos objetos Vertex e Edge, sendo que, como vimos na seção 4.1, os tipos destes campos são respectivamente VertProp e EdgeProp cuja definição é a seguinte:

```
VertProp = object
    geom : CPoint;
    {Outras características específicas da aplicação}
end;

EdgeProp = object
    orient : {−1, +1};     {orientação da aresta em relação ao S-círculo}
    geom : SCurve;
    {Outras características específicas da aplicação}
end;
```

Associadas a estas definições, vamos supor a existência das funções: $point(v)$ que retorna a geometria do vértice v, isto é, retorna $v.props.geom$; $sarc(e)$ que retorna a geometria da aresta e, isto é, tomando $p = e.props.geom.org$, $c = e.props.geom.scirc$ e $q = e.props.geom.dst$, se $e.props.geom.orient = +1$, $sarc(e)$ retorna $(p, \overset{\frown}{c}, q)$; caso contrário, se $e.props.geom.orient = -1$, retorna $(q, \overset{\frown}{\neg c}, p)$; e a função $scircle(e)$ que retorna a geometria do S-círculo de suporte da aresta e, isto é, retorna $(e.props.geom.orient) \times (e.props.geom.scirc)$.

Por simplicidade, dada uma esquina c da estrutura SMC, vamos abreviar $point(Org(c))$ por $point(c)$, $sarc(Edge(c))$ por $sarc(c)$ e $scircle(Edge(c))$ por $scircle(c)$.

Capítulo 9

Operações exatas com pontos de $\mathcal{C}$

Considerando a representação geométrica descrita nos capítulos anteriores, vamos agora desenvolver os operadores geométricos necessários para a elaboração dos algoritmos exatos de manipulação de mapas esféricos.

9.1 Posição de um ponto de $\mathcal{C}$ em relação a um plano racional

Inicialmente, vamos desenvolver o algoritmo $\mathsf{SideOf}(p, \alpha)$, que dado um ponto $p = [p_0, .. p_3] \in \mathcal{C}$ e um plano racional $\alpha = \langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle$ determina, de maneira exata, a posição de p em relação a α. Como vimos na seção 5.2, este predicado é definido pela fórmula

$$p \diamond \alpha = sign(p_0\alpha_0 + p_1\alpha_1 + p_2\alpha_2 + p_3\alpha_3) \tag{9.1}$$

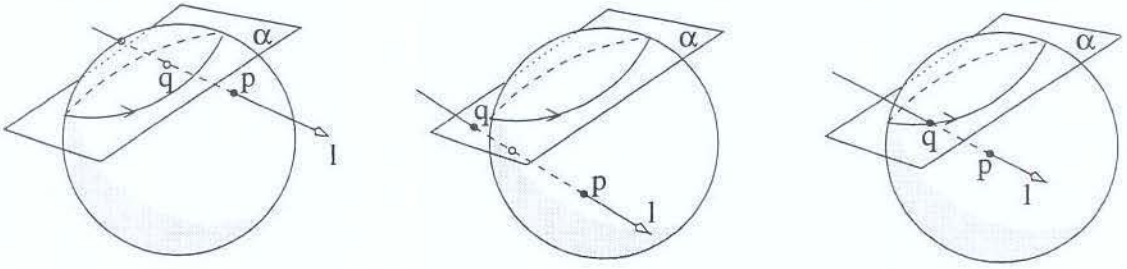
Entretanto, esta fórmula só pode ser usada diretamente se p é um ponto de $\mathcal{A}$. Caso contrário, se p é um ponto de $\mathcal{C} \setminus \mathcal{A}$, então seja $l = stab(p)$, isto é, $p = ext(l)$, e seja $q = l \wedge \alpha$. Vamos considerar quatro situações:

(i) se $q = [\mathbf{0}]$, então l está no plano α, e portanto $p \diamond \alpha = 0$;

(ii) se q está no lado negativo (no "interior") de $\mathbb{S}^2$, então podemos verificar que os pontos $ext(l)$ e $dir(l)$ estão sempre do mesmo lado de α. Veja 9.1(a). Como $dir(l)$ é um ponto racional, então basta computar $dir(l) \diamond \alpha$.

(iii) se q está no lado positivo (no "exterior") de $\mathbb{S}^2$, então podemos verificar que tanto $ext(l)$ como $ent(l)$ estão do mesmo lado de α, o que implica que $mid(l)$ também está deste mesmo lado. Veja 9.1(b). Visto que $mid(l)$ é um ponto racional então basta computar $mid(l) \diamond \alpha$.

(iv) se q é um ponto de $\mathbb{S}^2$, isto implica que $q = ext(l)$ ou $q = ent(l)$. Como podemos verificar, $q = ent(l)$ se e somente se os pontos $ext(l)$ e $mid(l)$ estão de um mesmo lado do plano α. Veja 9.1(c).



(a) (b) (c)

Figura 9.1: Posição de um ponto de $\mathcal{C}$ em relação a um plano racional.

Desta forma, o predicado $p \diamond \alpha$ pode ser computado de maneira exata através do seguinte procedimento:

---

procedure SideOf(p : CPoint; α : Plane) : Sign;
  $ap \leftarrow$ AExt(p);
  if $ap \neq [0]$ **then**
    return $ap \diamond \alpha$
  else
    $l \leftarrow stab(p)$; $\quad q \leftarrow l \wedge \alpha$;
    $\sigma \leftarrow sign(q_1^2 + q_2^2 + q_3^2 - q_0^2)$;
    $\mu \leftarrow mid(l) \diamond \alpha$;
    $\delta \leftarrow dir(l) \diamond \alpha$;
    if $\sigma = -1$ **then return** δ
    elsif $\sigma = 1$ **or** $\mu = \delta$ **then return** μ
    else return 0

Note que no algoritmo acima tratamos os casos (i) e (iv) em conjunto, pois, no caso (i), temos que $q = [0]$, o que implica que $\sigma = 0$, $\mu = 0$ e $\delta = 0$, e portanto podemos retornar o valor de μ.

Vale ressaltar que este mesmo procedimento nos permite determinar, de maneira exata, a posição de um ponto $p \in \mathcal{C}$ em relação a um S-círculo racional c; para isto, basta aplicar SideOf(p, $spln(c)$).

9.2 Ponto diametralmente oposto num S-círculo

Como vimos na seção 6.5.3, o ponto diametralmente oposto a um ponto $p = [p_0, .. p_3]$ sobre um S-círculo $c = ((c_0, .. c_3))$ é dado por

$$\oslash_c(p) = [p_0\kappa, -2c_0c_1p_0 - p_1\kappa, -2c_0c_2p_0 - p_2\kappa, -2c_0c_3p_0 - p_3\kappa]$$

onde $\kappa = c_1^2 + c_2^2 + c_3^3$.

É fácil ver que se $p \in \mathcal{A}$, então as coordenadas de $\oslash_c(p)$ podem ser computadas diretamente pela fórmula acima, de maneira exata. Por outro lado, se $p \in \mathcal{C} \setminus \mathcal{A}$, então a reta racional $l = stab(p)$ está sobre o plano $\alpha = spln(c)$ (pois, do contrário, $p = l \wedge \alpha$ seria um ponto racional). Daí, seja $\beta = \langle \beta_0, \beta_1, \beta_2, \beta_3 \rangle$ o plano determinado por $snrm(c) \vee l$. Obviamente, β é um plano ortogonal a α e $l = \alpha \wedge \beta$. Agora, seja γ o plano β rodado de 180° em torno da reta $axis(c)$. Pode-se verificar que $\gamma = \langle \beta_0, -\beta_1, -\beta_2, -\beta_3 \rangle$ e que $\oslash_c(p) = ext(\alpha \wedge \gamma)$. Veja figura 9.2.

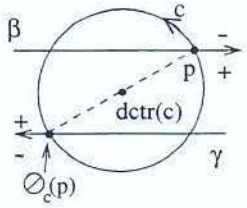


Figura 9.2: Um corte de $\mathbb{S}^2$ paralelo ao plano $spln(c)$.

Portanto, o ponto $\oslash_c(p)$ pode ser determinado pelo seguinte procedimento:

```
procedure DiamOpp(p : CPoint; c : SCircle) : CPoint;
    q ← AExt(p);
    if q ≠ [0] then
        return CExt(q ∨ dctr(c));
    else
        β ← stab(p) ∨ snrm(c);
        γ ← ⟨β0, −β1, −β2, −β3⟩;
        return CExt(spln(c) ∧ γ);
```

9.3 Posição relativa de dois pontos sobre um S-círculo

Como vimos na seção 6.5.4, a posição relativa de dois pontos $p = [p_0, .. p_3]$ e $q = [q_0, .. q_3]$ sobre um S-círculo $c = ((c_0, .. c_3))$ é dada por

$$\otimes_c(p,q) = sign \begin{vmatrix} c_1 & c_2 & c_3 \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix}$$

Entretanto, o cálculo deste determinante, em geral, não pode ser realizado de maneira exata, pois os pontos p e q podem não ser racionais. Assim sendo, vamos descrever um método alternativo, semelhante ao procedimento DiamOpp, que nos permita determinar o valor de $\otimes_c(p,q)$ de maneira exata.

Sejam $l = stab(p)$, $\alpha = spln(c)$ e $d = dctr(c)$; e seja também $\beta = \langle \beta_0, \beta_1, \beta_2, \beta_3 \rangle$ o plano $l \vee snrm(c)$. Como já vimos β é um plano perpendicular a α que corta o S-círculo c no ponto p. Além disso, β passa pela origem O de $\mathbb{T}^3$ se e somente se β passa pelo ponto d. Na verdade, temos que $d \diamond \beta = O \diamond \beta$.

Daí, podemos verificar — veja figura 9.3(a) — que, se β passa pela origem O então $\otimes_c(p,q) = +1$ se e somente se o ponto q está do lado negativo do plano β; mais especificamente, $\otimes_c(p,q) = -\mathsf{SideOf}(q,\beta)$.

Caso contrário, se o plano β não passa pela origem O, seja γ o plano β rodado de 180° em torno da reta $axis(c)$; isto é, $\gamma = \langle \beta_0, -\beta_1, -\beta_2, -\beta_3 \rangle$. Como vimos no algoritmo anterior, $ext(\alpha \wedge \gamma) = \oslash_c(p)$.

Agora, supondo inicialmente que a origem O (equivalentemente d) está do lado positivo do plano β, então $\otimes_c(p,q)$ é $+1$ se e somente se uma das seguintes situações ocorre:

- q está do lado negativo de β — veja figura 9.3(b); ou
- q está sobre β e $p \neq q$ — veja figura 9.3(c); ou
- q está do lado positivo de β e ao mesmo tempo do lado positivo de γ e do lado negativo do plano $bisect(l)$ — veja figura 9.3(d).

Por outro lado, se a origem O (equivalentemente d) está no lado negativo de β — como mostra a figura 9.3(e) — o predicado pode ser reduzido ao caso anterior invertendo-se a orientação do S-círculo c e usando a relação $\otimes_c(p,q) = -\otimes_{\neg c}(p,q)$. É importante notar que a inversão da orientação de c implica que as orientações dos planos β e γ também se invertem.

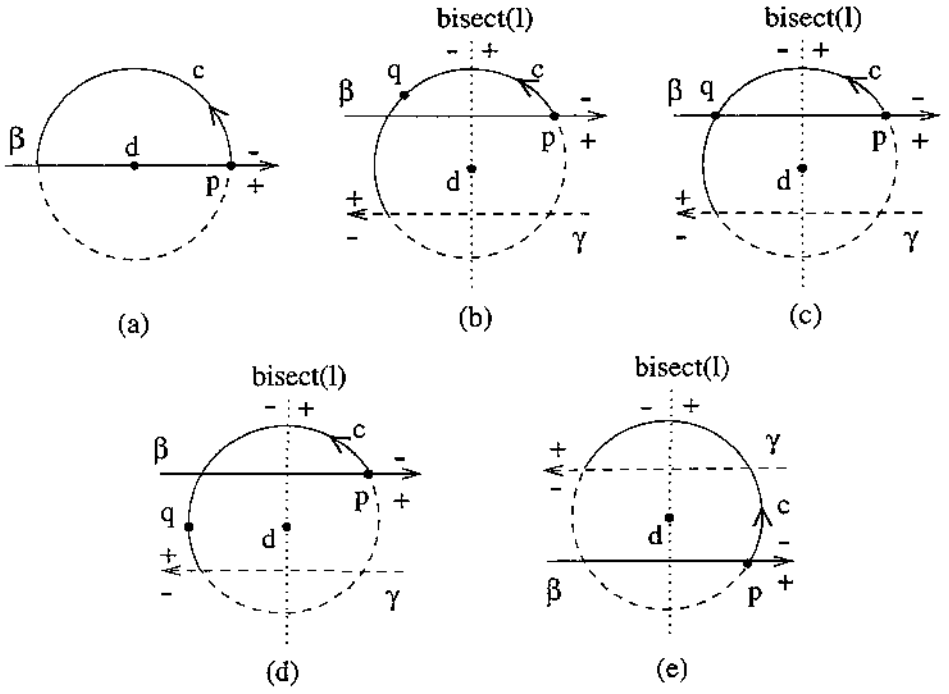


Figura 9.3: Cálculo do predicado $\otimes_c(p, q)$.

Logo, o predicado $\otimes_c(p, q)$ pode ser determinado através do seguinte algoritmo:

---

procedure Ahead(p, q : CPoint; c : SCircle) : Sign;
  $\beta \leftarrow stab(p) \vee snrm(c)$;
  $d_c \leftarrow$ SideOf(O, β);
  if $d_c = 0$ **then return** $-$SideOf(q, β);
  case $d_c \cdot$ SideOf(q, β) **of**
  -1 : $d_m \leftarrow +1$;
  0 : **if** $p = q$ **then** $d_m \leftarrow 0$
    else $d_m \leftarrow +1$;
  $+1$: **if** SideOf$(q, bisect(l)) \geq 0$ **then** $d_m \leftarrow -1$
    else
      $\gamma \leftarrow \langle \beta_0, -\beta_1, -\beta_2, -\beta_3 \rangle$;
      $d_m \leftarrow d_c \cdot$ SideOf(q, γ)
  return $d_c \cdot d_m$

9.4 Ordenação cíclica de três pontos de $\mathcal{C}$ sobre um S-círculo

Conforme vimos na seção 6.5.5, o predicado $\circledast_c(p,q,r)$, que determina a ordem cíclica de três pontos $p = [p_0,..p_3]$, $q = [q_0,..q_3]$ e $r = [r_0,..r_3]$ sobre um S-círculo $c = (\!(c_0,..c_3)\!)$ é dado por

$$\circledast_c(p,q,r) = sign \begin{vmatrix} p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & q_3 \\ r_0 & r_1 & r_2 & r_3 \\ c_0 & c_1 & c_2 & c_3 \end{vmatrix}$$

Mas, de modo análogo ao caso anterior, em geral o valor deste determinante não pode ser computado exatamente, pois os elementos envolvidos no cálculo podem não ser todos inteiros.

Visto que o predicado $\otimes$ pode ser computado exatamente — através do procedimento Ahead — então podemos contornar este problema, usando o teorema 10. Isto é,

```
procedure COrder(p, q, r : CPoint; c : SCircle) : Sign;
    return sign(Ahead(p, q, c) + Ahead(q, r, c) + Ahead(r, p, c))
```

9.5 Ordenação cíclica de três S-círculos em torno de um ponto

Nosso objetivo agora é desenvolver um algorimo exato para determinar a ordem cíclica em que três S-círculos $a = (\!(a_0,..a_3)\!)$, $b = (\!(b_0,..b_3)\!)$ e $c = (\!(c_0,..c_3)\!)$ saem de um ponto em comum $p = [p_0,..p_3]$, isto é, um algoritmo exato para determinar o valor do predicado $\circledast_p(a,b,c)$, definido na seção 6.6.

Sejam $\alpha = spln(a)$, $\beta = spln(b)$ e $\gamma = spln(c)$ e seja $l = stab(p)$. Note que se p é um ponto de $\mathcal{C} \setminus \mathcal{A}$, então, pelo corolário 30, a reta racional l é única e não é tangente a $\mathbb{S}^2$. Portanto, as retas $\alpha \wedge \beta$, $\alpha \wedge \gamma$ e $\beta \wedge \gamma$ devem ser coincidentes com a reta l, ou seja, a reta l é comum aos três planos α, β e γ. Neste caso, a ordem dos S-círculos a, b e c em torno do ponto p é igual à ordem dos planos α, β e γ em torno desta reta l; isto é,

$$\circledast_p(a,b,c) = \circledast_l(\alpha,\beta,\gamma)$$

Por outro lado, se p é um ponto de $\mathcal{A}$ então os três planos α, β e γ podem não ter uma reta comum. Neste caso, seja $\sigma = p^* = \langle -p_0, p_1, p_2, p_3 \rangle$. Note que o plano σ é um plano racional, tangente a $\mathbb{S}^2$ no ponto p e orientado de tal forma que o ponto $dir(O \vee p)$ está do lado positivo de σ. Além disso, note também que a direção do S-círculo a no ponto p,

que vamos indicar por d_a, é dada por $d_a = dir(\alpha \wedge \sigma)$. Analogamente, $d_b = dir(\beta \wedge \sigma)$ e $d_c = dir(\gamma \wedge \sigma)$. Se estas três direções são distintas entre si, então $\circledast_p(a, b, c)$ é a ordem cíclica destas direções relativas à reta no infinito $\sigma \wedge \Omega_2$ do plano σ; isto é,

$$\circledast_p(a, b, c) = \circledast_{\sigma \wedge \Omega_2}(d_a, d_b, d_c)$$

Finalmente, se duas ou mais destas direções são iguais, então isto significa que os respectivos S-círculos são coincidentes ou são tangentes entre si no ponto p. Neste caso, devemos comparar as curvaturas e orientações destes S-círculos. Como podemos perceber, isto equivale a verificar a posição dos planos de suporte desses S-círculos em torno da reta comum aos mesmos. Mais precisamente,

---

procedure ScircOrder(a, b, c : SCircle; p : CPoint) : Sign;
  $\alpha \leftarrow spln(a)$; $\beta \leftarrow spln(b)$; $\gamma \leftarrow spln(c)$;
  $l \leftarrow stab(p)$; $u \leftarrow$ AExt(l);
  if $u = [\mathbf{0}]$ **then return** $\circledast_l(\alpha, \beta, \gamma)$
  else
    $\sigma \leftarrow \langle -u_0, u_1, u_2, u_3 \rangle$;
    $d_a \leftarrow dir(\alpha \wedge \sigma)$; $d_b \leftarrow dir(\beta \wedge \sigma)$; $d_c \leftarrow dir(\gamma \wedge \sigma)$
    if $d_a \neq d_b$ **and** $d_b \neq d_c$ **and** $d_c \neq d_a$ **then return** $\circledast_{\sigma \wedge \Omega_2}(d_a, d_b, d_c)$
    elsif $d_a = d_b$ **and** $d_b = d_c$ **then return** $\circledast_{\alpha \wedge \sigma}(\alpha, \beta, \gamma)$
    else
      while $d_a \neq d_b$ **do**
        $(\alpha, \beta, \gamma) \leftarrow (\beta, \gamma, \alpha)$;
        $(d_a, d_b, d_c) \leftarrow (d_b, d_c, d_a)$
      return $\otimes_{\alpha \wedge \sigma}(\alpha, \beta)$

Note que este algoritmo nos permite também determinar a ordem cíclica de três S-arcos em torno de uma mesma origem, pois esta ordem coincide com a ordem dos respectivos S-círculos de suporte em torno do ponto de origem.

9.6 Algoritmo para verificar se um ponto pertence a uma S-curva

Vamos agora descrever o algoritmo InCurve que determina se um ponto p de $\mathcal{C}$ pertence a uma S-curva $A = (q_1, \overset{\frown}{c}, q_2)$. Mais precisamente, vamos elaborar um algoritmo que retorna true se o ponto p pertence ao S-arco A e retorna false, caso contrário.

Para efetuar este teste, o primeiro passo é verificar se o ponto p pertence ao S-círculo c de suporte da S-curva, ou seja, devemos verificar se $p \diamond spln(c) = 0$. Em caso afirmativo, se A é um S-círculo (isto é, se $q_1 = \varnothing = q_2$), então, nada mais há a fazer.

Caso contrário, se A é um S-arco, temos que verificar se o ponto p ocorre entre os pontos q_1 e q_2, considerando o sentido determinado pelo círculo c. Para isto, primeiro verificamos se $q_1 = q_2$; em caso afirmativo, por definição (seção 6.4), o S-arco é todo o círculo menos o ponto $q_1 = q_2$; portanto, neste caso, basta verificar se p é diferente de q_1. Por outro lado, se $q_1 \neq q_2$, entao basta verificar se $\otimes_c(q_1, p, q_2) = +1$. Ou seja,

```
procedure InCurve(p : CPoint; A : SCurve) :Boolean;
    c ← A.scirc;
    if SideOf(p, c) = 0 then
        if A.org = A.dst then return p ≠ A.org
        else return ⊗c(A.org, p, A.dst) = +1;
    else return false
```

Capítulo 10

Localização de um ponto num mapa

Neste capítulo vamos tratar do problema fundamental da *localização* de um ponto num mapa que, resumidamente, consiste em determinar qual elemento de um mapa $\mathcal{M}$ contém um dado ponto p.

De agora em diante, vamos supor que um mapa é representado pela estrutura SMC e referenciado através de um apontador para um objeto do tipo Face desta estrutura. Vamos denotar esta face por $RefFace(\mathcal{M})$. Além disso, vamos também supor que o ponto p é um ponto de $\mathbb{C}$ dado pela sua representação canônica.

10.1 O conceito de endereço

Em princípio, o resultado da localização de um ponto poderia ser um vértice, uma aresta ou uma face, isto é, um objeto do tipo Vertex, Edge ou Face da estrutura SMC. Entretanto, visto que a manipulação de um mapa é realizada através de suas esquinas, e visto que a nossa estrutura SMC não permite obter as esquinas a partir de objetos dos tipos Vertex ou Edge, então vamos supor que a operação de localização retorna não apenas o elemento que contém o ponto, mas também uma esquina definida sobre este elemento. Formalmente,

Definição 17 Um *endereço* de um ponto p num mapa não trivial $\mathcal{M}$ é um par $l = (l.h, l.c) \in L_{\mathcal{M}} \times Q_{\mathcal{M}}$, onde $l.h$ é o elemento do mapa $\mathcal{M}$ que contém o ponto p e $l.c$ é uma esquina tal que, se $l.h$ é um vértice então $l.h = Org(l.c)$; se $l.h$ é uma aresta, $l.h = Edge(l.c)$; e, se $l.h$ é uma face, $l.h = Left(l.c)$. Por outro lado, o único endereço de um ponto p num mapa trivial $\mathcal{M}$ é o par $(f, \varnothing)$, onde $f = RefFace(\mathcal{M})$ é a única face do mapa.

Note que um ponto p pode ter mais de um endereço num mesmo mapa. Por exemplo, se p está sobre um vértice v que é a origem de várias arestas, então todo par (v, c) com

$Org(c) = v$ é um endereço de p. Analogamente, se p está sobre uma aresta $e = Edge(c)$, então tanto (e, c) quanto $(\neg e, Sym(c))$ são endereços de p.

10.2 Localização absoluta

Assim, o problema de localização de um ponto p num mapa $\mathcal{M}$ pode ser resolvido por um algoritmo, que vamos denominar de $\mathsf{Locate}(p, \mathcal{M})$, que dado um ponto p de $\mathbb{S}^2$ e um mapa esférico $\mathcal{M}$, determina um endereço para p em $\mathcal{M}$.

A idéia básica deste algoritmo consiste em: se $\mathcal{M}$ é o mapa trivial, então o algoritmo simplesmente retorna o endereço $(RefFace(\mathcal{M}), \varnothing)$; caso contrário, o algoritmo simula o movimento de um ponto q ao longo de um *caminho de referência*, isto é, uma seqüência arbitrária de S-arcos que começa num *ponto de referência* r cujo endereço no mapa é conhecido, e termina no ponto p dado. Ao longo deste percurso, o endereço do ponto q é atualizado examinando-se os vértices e arestas encontrados pelo caminho.

10.2.1 Obtenção de um ponto de referência

Desta forma, o primeiro passo do algoritmo é obter um ponto de referência r e um endereço l_r deste ponto no mapa. Para isto, vamos supor a existência do procedimento $\mathsf{GenRefPoint}(\mathcal{M})$ que retorna o par (r, l_r) e consiste simplesmente em: seja $f = RefFace(\mathcal{M})$, seja b um objeto do tipo Border desta face, e seja $c = b.desc$. Se $Org(c) \neq \varnothing$, então basta tomar $r = point(c)$ e $l_r = (Org(c), c)$; por outro lado, se $Org(c) = \varnothing$, isto é, se $Edge(c)$ é uma aresta oval, então basta gerar um ponto qualquer r sobre esta aresta e tomar $l_r = (Edge(c), c)$.

10.2.2 Obtenção de um caminho de referência

O passo seguinte é determinar um caminho de referência que começa no ponto r e termina no ponto p.

Visto que nosso objetivo é elaborar um algoritmo de localização exato, então é fundamental que o caminho de referência seja formado por S-arcos cujos S-círculos de suporte sejam racionais. O ideal seria que este caminho fosse composto por um único S-arco. No entanto, o ponto p dado e o ponto r gerado acima são, em geral, pontos de $\mathcal{C} \setminus \mathcal{A}$, e nem sempre existe um S-círculo racional passando por estes dois pontos. Por exemplo, suponha que $p = \langle\langle -7, 0, 0, -12, 14, 0\rangle\rangle$ e $r = \langle\langle 0, -1, 2, 0, 0, -2\rangle\rangle$. Pela fórmula (8.1), as coordenadas homogêneas destes pontos são $p = [14, 12, -\sqrt{3}, -7]$ e $r = [4, 1 + \sqrt{7}, 0, 1 - \sqrt{7}\,]$, que não podem ser reduzidas a coordenadas homogêneas racionais. Portanto, visto que p e r são pontos de $\mathcal{C} \setminus \mathcal{A}$, então, pelo corolário 30, $l_p = stab(p) = \langle -7, 0, 0, -12, 14, 0\rangle$

e $l_r = stab(r) = \langle 0, -1, 2, 0, 0, -2\rangle$ são as únicas retas racionais tais que $ext(l_p) = p$ e $ext(l_r) = r$. Suponha agora que existe um S-círculo racional passando por p e por r. O plano $\alpha = spln(c)$ deve conter a reta l_p, pois caso contrário, o ponto $l_p \wedge \alpha = p$ seria racional. Pelo mesmo motivo, α deve conter a reta l_r. Mas, pela fórmula (5.8), $l_p \diamond l_m \neq 0$, ou seja, as retas l_p e l_r não são coplanares.

No entanto, mesmo neste caso, podemos sempre tomar um caminho de referência formado por dois S-arcos $A_1 = (r, \widehat{s_1}, q)$ e $A_2 = (q, \widehat{s_2}, p)$, onde q é um ponto qualquer de $\mathcal{A}$, e s_1 e s_2 são os S-círculos $scrc(stab(r) \vee q)$ e $scrc(stab(p) \vee q)$. Note que, p e r não são pontos de $\mathcal{A}$ e portanto, s_1 e s_2 estão bem definidos e são racionais.

Assim sendo, vamos supor a existência do procedimento BuildRefPath(r, p) que recebe como parâmetros os pontos r e p de $\mathcal{C}$ e retorna uma lista $A_1, A_2, \cdots A_m$ de S-arcos cujos S-círculos de suporte são racionais tais que $A_1.org = r$, $A_m.dst = p$, e $A_i.dst = A_{i+1}.org$, para todo $i = 1, \cdots m-1$. De acordo com o resultado descrito no parágrafo anterior, podemos sempre supor que $m \leq 2$.

10.2.3 Percorrendo o caminho de referência

Finalmente, o terceiro passo do algoritmo Locate consiste em aplicar o algoritmo auxiliar LocateRel, que vamos descrever na seção 10.3, a cada arco do caminho de referência, sendo que o algoritmo LocateRel recebe como parâmetros um S-arco A_i e um endereço do ponto $A_i.org$ e retorna o endereço do ponto $A_i.dst$. Desta forma, o algoritmo de localização é

```
procedure Locate(p : CPoint; M : Map) : Address;
    if M é o mapa trivial then
        return (RefFace(M), ∅)
    else
        (r, l_r) ← GenRefPoint(M);
        (A_1, ···, A_m) ← BuildRefPath(r, p);
        for i = 1, ··· m do
            (r, l_r) ← LocateRel(A_i, l_r);
        return l_r;
```

10.3 Localização relativa

10.3.1 Endereço de um S-vetor

Na elaboração do algoritmo LocateRel, vamos utilizar o conceito adicional de endereço de um S-vetor (p, s), isto é, o endereço dos pontos de s logo adiante de p. Vamos usar o símbolo $p + \epsilon s$ para indicar o ponto p deslocado um infinitésimo ϵ (no sentido positivo) ao longo do S-círculo s.

Definição 18 Dado um S-vetor (p, s), dizemos que o par $m = (m.h, m.c) \in L_{\mathcal{M}} \times Q_{\mathcal{M}}$ é um *endereço do S-vetor* (p, s) se os pontos $p + \epsilon s$ estão contidos no elemento $m.h$ e se uma das condições abaixo é satisfeita:

(1) $p \in Left(m.c)$ (e portanto, $m.h = Left(m.c)$);

(2) $p \in Edge(m.c)$, e

  (2.1) $scircle(m.c) = s$ (e portanto, $m.h = Edge(m, c)$); ou

  (2.2) os pontos $p + \epsilon s$ estão no lado positivo de $scircle(m.c)$ (e portanto, $m.h = Left(m.c)$); ou

(3) $p \in Org(m.c)$, e

  (3.1) $Edge(m.c) = \varnothing$ (e portanto, $m.h = Left(m.c)$); ou

  (3.2) $scircle(m.c) = s$ (e portanto, $m.h = Edge(m.c)$); ou

  (3.3) $m.c = Onext(m.c)$ (e portanto, $m.h = Left(m.c)$); ou

  (3.4) $\otimes_p(r, s, t) = +1$, onde $r = scircle(m.c)$, e $t = scircle(Onext(m.c))$ (e portanto, $m.h = Left(m.c)$).

Para ilustrar o conceito de endereço de um S-vetor, veja a figura 10.1.

Intuitivamente, se $m = (m.h, m.c)$ é um endereço para o S-vetor (p, s), então não há nenhuma outra esquina c' tal que $p \in Org(c') \cup Edge(c')$, e tal que $scircle(c')$ está entre $scircle(m.c)$ e s, no sentido positivo de rotação em torno de p.

10.3.2 Algoritmo LocateRel

Vamos agora descrever o algoritmo LocateRel(A, l) que, dado um S-arco A e um endereço l para $A.org$, retorna um endereço para $A.dst$. Este algoritmo utiliza três procedimentos auxiliares: WhereTo, ClosestEdgeExit e ClosestFaceExit que são respectivamente apresentados nas seções 10.3.3, 10.3.6 e 10.3.7. Intuitivamente, WhereTo simula o avanço infinitésimo

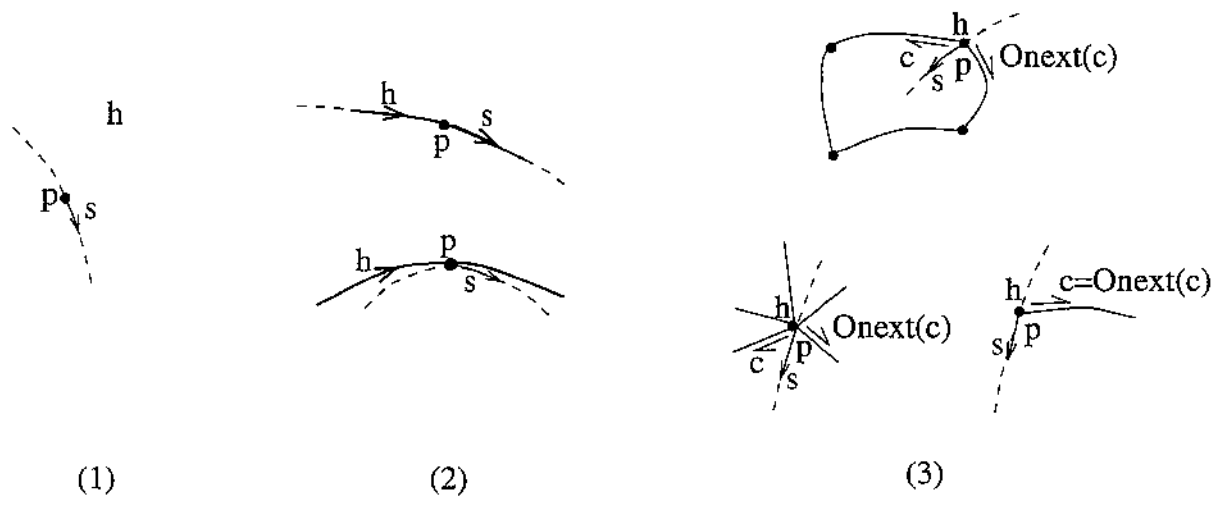


Figura 10.1: Endereços (h, c) do S-vetor (p, s): (1) h é uma face; (2) h é uma aresta; (3) h é um vértice.

do ponto q saindo de uma aresta ou vértice para dentro de uma face ou para uma aresta colinear com o S-arco A; ClosestEdgeExit simula o avanço do ponto q sobre uma aresta contida em $A.scirc$ até o fim da mesma ou até o fim do S-arco A; e ClosestFaceExit simula o percurso de q dentro de uma face do mapa, até encontrar um vértice ou uma aresta na fronteira da face ou até encontrar o fim do S-arco A.

```
procedure LocateRel(A : SArc; l : Address) : Address;
  q ← A.org;    m ← l;    B ← A;
  while q ≠ A.dst do
    m ← WhereTo(q, m, s);
    if m.h ∈ E_M then {sarc(m.h) ⊆ s}
      (q, m) ← ClosestEdgeExit(B, m.c);
    else {m.h ∈ F_M}
      (q, m) ← ClosestFaceExit(B, m);
    B ← (q, ŝ, A.dst);
  return m
```

É importante ressaltar que a implementação que apresentaremos a seguir não é mais eficiente, como veremos no final deste capítulo. No entanto, a estrutura deste algoritmo e vários dos seus procedimentos auxiliares serão utilizados no desenvolvimento do algoritmo de sobreposição.

10.3.3 O procedimento WhereTo

Este procedimento recebe como parâmetros um ponto q, um endereço l do ponto q num mapa $\mathcal{M}$, e um S-círculo s passando por q; e retorna um endereço m para o S-vetor (q, s). Note que, no endereço m, o elemento $m.h$ é necessariamente a face $Left(m.c)$, ou a aresta $Edge(m.c)$ cujo S-círculo de suporte é s. O procedimento deve considerar três casos:

1. se $l.h$ é uma face, isto é, se $q \in Left(l.c)$, um deslocamento infinitesimal em qualquer direção nos mantém dentro dessa mesma face. Portanto, neste caso, o procedimento retorna o endereço $m = (l.h, l.c)$;

2. se $l.h$ é uma aresta, isto é, se $q \in Edge(l.c)$, o procedimento determina o elemento h dentre $Edge(l.c)$, $Edge(Sym(l.c))$, $Left(l.c)$ e $Left(Sym(l.c))$ que contém o ponto $q + \epsilon s$. Nos dois primeiros casos, a aresta h escolhida é aquela cujo S-círculo de suporte é igual a s. O procedimento retorna então o endereço $m = (h, l.c)$ ou $m = (h, Sym(l.c))$ conforme o caso. Esta tarefa é realizada utilizando-se o procedimento auxiliar TestEdgeCorner, descrito na seção 10.3.4.

3. se $l.h$ é um vértice, isto é, se $q \in Org(l.c)$, o procedimento determina o elemento h (face ou aresta) que contém o ponto de $q + \epsilon s$ e uma esquina c sobre este elemento que satisfaz a condição (3) da definição 18. O procedimento retorna então o endereço $m = (h, c)$. Este caso é tratado pelo procedimento auxiliar TestVertexCorner, descrito na seção 10.3.5.

```
procedure WhereTo(q : CPoint; l : Address; s : SCircle) : Address;
  if l.h ∈ F_M then {q ∈ Left(l.c)}
    return (l.h, l.c)
  elsif l.h ∈ E_M then {q ∈ Edge(l.c)}
    c ← l.c;
    while true do
      h ← TestEdgeCorner(q, c, s);
      if h ≠ ∅ then return (h, c)
      c ← Sym(c);
  else {q ∈ Org(l.c)}
    c ← l.c;
    while true do
      h ← TestVertexCorner(c, s);
      if h ≠ ∅ then return (h, c);
      c ← Onext(c);
```

10.3.4 O procedimento TestEdgeCorner

Dados um ponto q, uma esquina c tal que $Edge(c) \neq \varnothing$ e tal que $q \in Edge(c)$ e um S-círculo s passando por q, este procedimento verifica se um dos elementos $Edge(c)$ ou $Left(c)$ contém o ponto $q + \epsilon s$. Além disso, se $Edge(c)$ satisfaz esta primeira condição, então o procedimento verifica ainda se $scircle(c) = s$. Caso tais condições sejam satisfeitas, o procedimento retorna o respectivo elemento; caso contrário retorna $\varnothing$.

Note que este procedimento deve considerar três casos: seja $t = scircle(c)$, então

(i) se s e t são coincidentes, então se $s = t$ retorna $Edge(c)$; se $s = \neg t$, retorna $\varnothing$; — veja a figura 10.2(a);

(ii) s é tangente a t, então todos os pontos de s (exceto q) estão de um mesmo lado do S-círculo t; portanto, basta verificar se o ponto $dctr(s)$ está do lado positivo do plano $spln(t)$; em caso afirmativo, o procedimento retorna $Left(c)$, senão, retorna $\varnothing$ — veja a figura 10.2(b).

(iii) se s cruza t, então, basta verificar se, no ponto q, s está entrando no lado positivo de t, equivalentemente, se $q = t \wedge s$. Em caso afirmativo, o procedimento retorna $Left(c)$, senão, retorna $\varnothing$ — veja a figura 10.2(c).

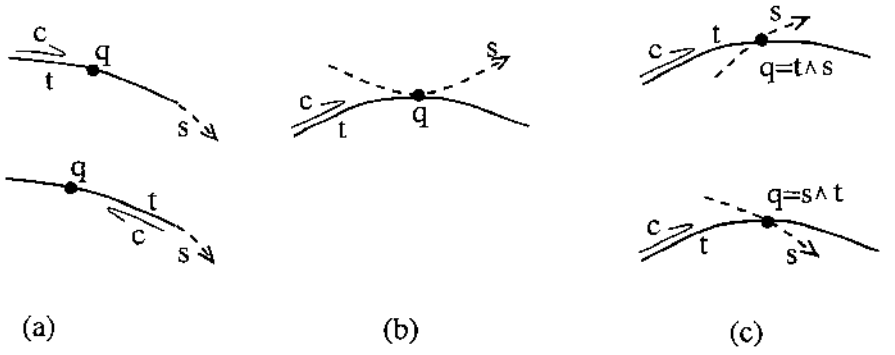


Figura 10.2: Casos a serem tratados pelo procedimento TestEdgeCorner.

Note que os casos (i) e (ii) podem ser tratados em conjunto, pois de acordo com a definição do predicado $\daleth(s,t)$, tanto em (i) como em (ii), temos que $\daleth(s,t) = 0$. Além disso, note que pelas condições de entrada do procedimento, $\daleth(s,t) \geq 0$, pois $q \in Edge(c)$

e s passa por q, isto é, os S-círculos s e t têm pelo menos o ponto q em comum.

```
procedure TestEdgeCorner(q : CPoint; c : Corner; s : SCircle) : Element;
    t ← scircle(c);
    if ⨀(s,t) = 0 then {s e t são coincidentes ou tangentes entre si}
        if s = t then return Edge(c);
        elsif dctr(s) ◇ spln(t) = +1 then return Left(c)
        else return ∅;
    else {⨀(s,t) > 0}
        p ← t ∧ s;
        if p = q then return Left(c)
        else return ∅;
```

10.3.5 O procedimento TestVertexCorner

Dados uma esquina c tal que $Org(c) \neq \varnothing$ e um S-círculo s passando por $Org(c)$, este procedimento verifica se um dos elementos $Edge(c)$ ou $Left(c)$ contém o ponto $q+\epsilon s$. No primeiro caso, isto é, se $Edge(c)$ satisfaz esta primeira condição, o procedimento verifica ainda se $scircle(c) = s$. No segundo caso, o procedimento verifica além disso, se não há nenhuma outra aresta com origem em $Org(c)$ entre a aresta $Edge(c)$ e o S-círculo s. Caso todas estas condições sejam satisfeitas, o procedimento retorna $Edge(c)$ ou $Left(c)$ conforme o caso; senão retorna $\varnothing$.

```
procedure TestVertexCorner(c : Corner; s : SCircle) : Element;
    if Edge(c) = ∅ then return Left(c);
    else
        if s = scircle(c) then return Edge(c)
        elsif Onext(c) = c then return Left(c);
        else
            q ← point(c);
            if ⊗q(scircle(c), s, scircle(Onext(c))) = +1 then
                return Left(c)
            else return ∅
```

10.3.6 O procedimento ClosestEdgeExit

Este procedimento é chamado pelo algoritmo LocateRel quando o ponto q está se deslocando ao longo de uma aresta do mapa. Ele recebe como parâmetros um S-arco $A = (p_1, \widehat{s}, p_2)$ e uma esquina c tal que $scircle(c) = s$ e tal que $p_1 + \epsilon s \in Edge(c)$. A partir destes dados, o procedimento determina qual das S-curvas A e $B = sarc(c)$ termina primeiro, quando o S-círculo s é percorrido a partir de p_1. Visto que p_1 está dentro do S-arco B, então a resposta deve necessariamente ser $A.dst$ ou $B.dst$. O procedimento retorna um par (q, l_q), onde q é o destino da S-curva que termina primeiro e l_q é um endereço do ponto q. Por definição, este endereço é sempre da forma $(h, Sym(c))$, onde h ou é o vértice de destino de $Edge(c)$, se a aresta $Edge(c)$ termina primeiro, ou a própria aresta $Edge(Sym(c))$, se o S-arco A termina primeiro. Veja figura 10.3.

Vale observar que a S-curva B pode ser um S-arco ou um S-círculo completo; neste segundo caso, o S-arco A sempre termina primeiro. Veja figura 10.3(i).

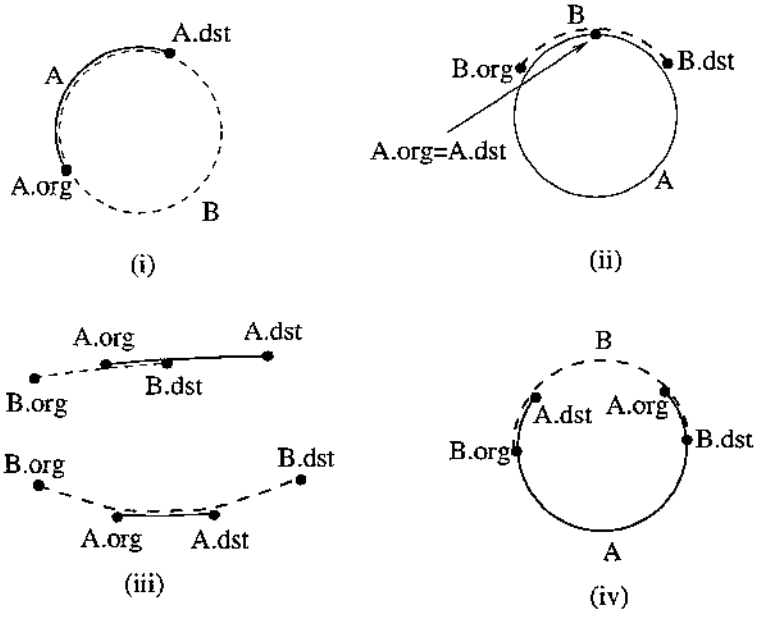


Figura 10.3: Alguns casos a serem tratados pelo procedimento ClosestEdgeExit: (i) B é um S-círculo completo; (ii) A é um S-círculo menos um ponto; (iii) e (iv) A e B são arcos ordinários.

```
procedure ClosestEdgeExit(A : SArc; c : Corner) : (CPoint,Address);
  if Org(c) = ∅ then {Edge(c) é uma aresta oval}
    α ← true;
  else
    B ← sarc(c);
    if A.dst = B.dst or A.org = A.dst then α ← false;
    elsif B.dst ∈ A then α ← false;
    else α ← true;
  c' ← Sym(c);
  if α then {A termina primeiro}
    q ← A.dst;    h ← Edge(c');
  else {A não termina primeiro}
    q ← B.dst;    h ← Org(c');
  l_q ← (h, c');
  return (q, l_q);
```

10.3.7 O procedimento ClosestFaceExit

Este procedimento é chamado pelo algoritmo LocateRel quando o ponto q está se deslocando dentro de uma face f (ao longo de um trecho do caminho de referência). O procedimento recebe como parâmetros um S-arco A e um endereço l do S-vetor $(A.org, A.scirc)$, e retorna um ponto q que, partindo de $A.org$, é o primeiro ponto no S-círculo $A.scirc$ que está fora do S-arco A ou fora da face f. Além disso, o procedimento também retorna um endereço m do S-vetor $(q, \neg A.scirc)$.

De acordo com esta especificação, se o S-arco A e o seu destino $A.dst$ estão contidos na face f, então o procedimento retorna o ponto $A.dst$ e o próprio endereço l. Por outro lado, se A está inteiramente contido na face f, mas $A.dst$ está sobre algum elemento h da fronteira de f, então o procedimento retorna o ponto $A.dst$ e o endereço $m = (h, c)$, onde c é a única esquina tal que (f, c) é um endereço do S-vetor $(A.dst, \neg A.scirc)$.

Note que se o S-arco A encontra algum elemento na fronteira da face f, então considera-se que o S-arco sai da face naquele ponto, mesmo que volte a entrar na face imediatamente em seguida. (Isto inclui o caso em que A é tangente a alguma aresta da fronteira de f.) Veja por exemplo a figura 10.4: em (a) o S-arco A sai da face f nos pontos q_1, q_2, q_3, q_4 e q_5; neste caso, o resultado é o par (q_1, l), onde $l = (Edge(c_1), c_1)$; em (b), o resultado é (q, l), onde $l = (Edge(c_2), c_2)$; e em (c), o resultado é (q, l) onde $l = (Org(c_1), c_1)$.

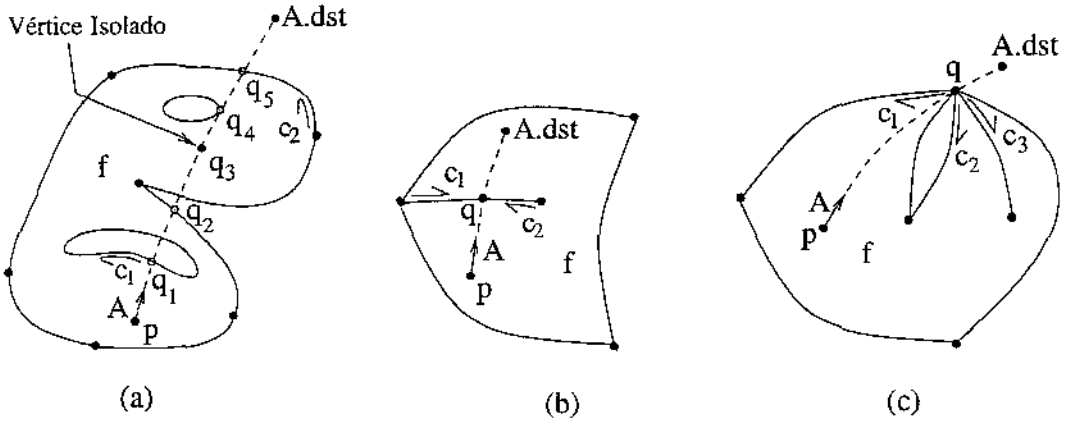


Figura 10.4: Resultado do procedimento ClosestFaceExit.

A implementação deste procedimento usa o fato que um S-arco A, cujo início está dentro de uma face f, sai desta face se e somente existe uma aresta e na fronteira de f que é interceptada por A ou que contém $A.dst$; ou existe um vértice v na fronteira de f tal que $point(v) \in A$ ou $point(v) = A.dst$. Portanto, o procedimento pode obter o ponto desejado da seguinte forma: inicialmente, toma $q = A.dst$ e, para cada elemento h (aresta ou vértice) na fronteira da face f, verifica se o S-arco $B = (A.org, \widehat{s}, q)$ encontra h ou se o ponto $B.dst$ está contido em h. Toda vez que isso ocorre, o procedimento encurta o S-arco B trazendo o ponto q para este ponto de encontro. Ao final deste processo, o ponto mais próximo de $A.org$ onde o S-arco A sai da face f é obtido.

Esta lógica determina o ponto de saída q, mas não necessariamente o endereço correto, pois podem existir duas ou mais esquinas na fronteira de f cuja origem ou cuja aresta contém q; por exemplo, na figura 10.4(b), temos que $Edge(c_1)$ e $Edge(c_2)$ contêm o ponto q, e na figura 10.4(c), $Org(c_1)$, $Org(c_2)$ e $Org(c_3)$ contêm o ponto q. Portanto, se o ponto q está sobre uma aresta $e = Edge(c)$ mas q não é a interseção canônica entre s e $scircle(c)$, a esquina c pode ser ignorada. Em particular, se a face f é incidente aos dois lados da aresta e, eventualmente a esquina simétrica será alcançada, e neste caso, o ponto q não será ignorado; por exemplo, veja a figura 10.4(b). Analogamente, se q está sobre um vértice $v = Org(c)$, basta verificar se a esquina c fornece um endereço para o S-vetor $(q, \neg s)$, o que pode ser feito através do procedimento TestVertexCorner. Se a esquina c não satisfaz as condições desejadas, ela pode ser ignorada, pois uma outra esquina (com mesma origem v) satisfazendo tais condições será eventualmente encontrada. Veja figura 10.4(c).

Com isto, o procedimento ClosestFaceExit pode ser implementado da seguinte forma:

```
procedure ClosestFaceExit(A : SArc; l : Address) : (CPoint,Address);
  p ← A.org;   q ← A.dst;   s ← A.scirc;   f ← Left(l.c);
  m ← l;   B ← A;
  for each b ∈ f.frontier do
    d ← b.desc;
    repeat
      if Edge(d) ≠ ∅ then
        C ← sarc(d);   u ← s ∧ C.scirc;
        if u ≠ ⟪0⟫ and u ∈ C and (u = B.dst or u ∈ B) then
          q ← u;   m ← (Edge(d), d);   B ← (p, ŝ, q);
      v ← Org(d);
      if v ≠ ∅ and (point(v) ∈ B or point(v) = B.dst) then
        h ← TestVertexCorner(d, ¬s);
        if h ≠ ∅ then
          q ← point(v);   m ← (v, d);   B ← (p, ŝ, q);
      d ← Lnext(d);
    until d = b.desc;
  return (q, m);
```

10.4 Complexidade do algoritmo Locate

Para analisar a complexidade dos algoritmos vistos nas seções anteriores, vamos definir a complexidade de um elemento do mapa como sendo o número de esquinas associadas àquele elemento. Por exemplo, a complexidade de um vértice v é o número de esquinas c tais que $Org(c) = v$. Analogamente, a complexidade de uma face f é o número de esquinas c tais que $Left(c) = f$. Note que a complexidade de uma aresta é 2, pois há sempre duas esquinas definidas sobre uma aresta. Definimos a complexidade total do mapa como sendo a soma das complexidades dos seus elementos.

Se q é um ponto numa face ou numa aresta, pode-se verificar facilmente que o custo do procedimento WhereTo é $O(1)$; por outro lado, se q é um ponto sobre um vértice v, o custo, no pior caso, é $\Theta(g)$, onde g é a complexidade do vértice v.

Além disso, podemos verificar também que o custo do procedimento ClosestEdgeExit(A, c) é $O(1)$, e que o custo, no pior caso, de ClosestFaceExit(A, l) é $\Theta(g)$, onde g é a complexidade da face $Left(c)$. Veja figura 10.5.

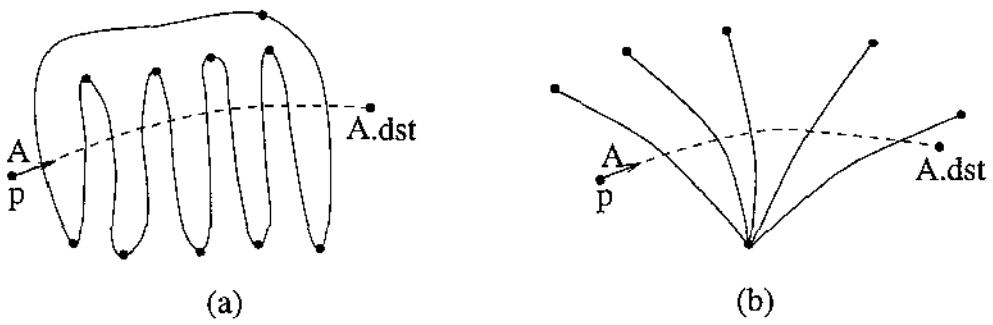


Figura 10.5: Exemplos do pior caso do procedimento ClosestFaceExit.

Portanto, o custo do algoritmo LocateRel(A, l) é $\Theta(\sum_i g_i)$, onde os g_i's são as complexidades de todos os vértices e faces encontrados pelo S-arco A. Note que cada vez que uma face é encontrada pelo S-arco A, a complexidade desta face é considerada uma vez na somatória. Visto que, no pior caso, uma face com complexidade g pode ser encontrada $\Theta(g)$ vezes por A, então podemos concluir que, no pior caso, o custo deste algoritmo, para um mapa de complexidade total n, é $\Theta(n^2)$.

O pior caso ocorre, por exemplo, se o mapa tem uma face com n arestas e o S-arco A intercepta todas as arestas deste mapa. Veja as figuras 10.5(a) e (b).

10.5 O algoritmo FarthestFaceExit

Para evitar o custo quadrático no pior caso do algoritmo Locate, precisamos evitar percorrer a fronteira de uma mesma face mais de uma vez; mesmo que esta fronteira seja cruzada várias vezes pelo S-arco A.

Felizmente, não precisamos determinar todos os cruzamentos entre o S-arco A e a fronteira da face f; basta obter o *último* destes pontos. Desta forma, cada face f é considerada uma única vez.

Esta observação nos leva a elaborar o algoritmo FarthestFaceExit, que de modo análogo ao algoritmo ClosestFaceExit, recebe como parâmetros um S-arco A e um endereço l do S-vetor $(A.org, A.scirc)$, e retorna um par (q, m). Porém, neste caso q é o ponto *mais distante* de $A.org$ onde o S-arco A sai da face f, e m é um endereço de q tal que $(f, m.c)$ é um endereço do S-vetor $(q, \neg A.scirc)$. Por exemplo, na figura 10.4(a), o resultado de FarthestFaceExit é o ponto q_5 e o endereço $(Edge(c_2), c_2)$. Em particular, se o S-arco A está inteiramente contido na face f, por definição, q é o ponto $A.dst$.

O princípio básico deste algoritmo é idêntico ao algoritmo ClosestFaceExit. A única diferença é que, a cada ponto de encontro q obtido, ao invés de encurtarmos o S-arco B trazendo o seu destino para q, levamos a sua origem para q; ou seja, em cada passo tomamos o S-arco $B = (q, \widehat{s}, A.dst)$. Como é fácil perceber, com esta simples modificação,

ao final do algoritmo obtemos o ponto mais distante de $A.org$ onde o S-arco A sai da face $Left(l.c)$.

Portanto, se no algoritmo LocateRel substituirmos o procedimento ClosestFaceExit por FarthestFaceExit, a complexidade daquele algoritmo, no pior caso, passa a ser $O(n)$.

Capítulo 11

Sobreposição de mapas

Neste capítulo vamos tratar do problema de sobreposição de mapas esféricos. Para isto, vamos primeiramente definir o problema e depois, seguindo uma estratégia semelhante àquela utilizada no capítulo anterior, vamos elaborar passo a passo um algoritmo que computa a sobreposição de dois mapas esféricos.

Definição 19 Um mapa $\mathcal{M}' = (T, \mathcal{L}')$ é um *refinamento* do mapa $\mathcal{M} = (T, \mathcal{L})$ se todo elemento de $\mathcal{L}'$ está inteiramente contido em algum elemento de $\mathcal{L}$. Indicamos esta relação por $\mathcal{M}' \succcurlyeq \mathcal{M}$.

É fácil verificar que $\succcurlyeq$ é uma relação de ordem parcial sobre os mapas. Além disso, $\mathcal{M}'$ é um refinamento de $\mathcal{M}$ se todo elemento de $\mathcal{M}$ é a união de um ou mais elementos de $\mathcal{M}'$.

Definição 20 A *sobreposição* $\mathcal{M}_1 \oplus \mathcal{M}_2$ de dois mapas $\mathcal{M}_1$ e $\mathcal{M}_2$ é o mapa menos refinado que é um refinamento de $\mathcal{M}_1$ e de $\mathcal{M}_2$.

Informalmente, dados os mapas $\mathcal{M}_1 = (T, \mathcal{L}_1)$ e $\mathcal{M}_2 = (T, \mathcal{L}_2)$, então $\mathcal{M}_1 \oplus \mathcal{M}_2$ corresponde a repartir a superfície T segundo esses dois mapas ao mesmo tempo. Por exemplo, veja figura 11.1.

Note que para cada elemento m do mapa $\mathcal{M}_1 \oplus \mathcal{M}_2$ existe um único elemento de $\mathcal{M}_1$ e um único elemento de $\mathcal{M}_2$ que contêm m. Na verdade, os elementos de $\mathcal{M}_1 \oplus \mathcal{M}_2$ são todas as componentes conexas de todos os conjuntos da forma $m_1 \cap m_2$, onde m_1 é um elemento de $\mathcal{M}_1$ e m_2 é um elemento de $\mathcal{M}_2$. Note que a interseção $m_1 \cap m_2$ pode consistir de zero, uma ou mais componentes conexas.

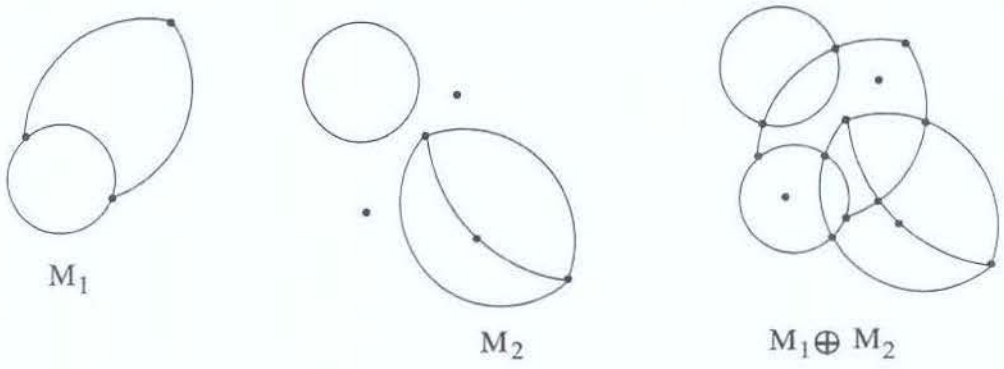


Figura 11.1: Sobreposição de dois mapas.

11.1 Mapas algébricos

É importante ressaltar que, no caso de mapas genéricos, o conceito de sobreposição pode não estar bem definido pois a interseção entre duas curvas (uma de cada mapa) pode ter um número arbitrário (talvez infinito) de componentes conexas. Portanto, a topologia do mapa $\mathcal{M}_1 \oplus \mathcal{M}_2$ pode ser arbitrariamente mais complicada do que as topologias de $\mathcal{M}_1$ e de $\mathcal{M}_2$. Por exemplo, a interseção das curvas

$$\begin{aligned} a &= \{[1, t, t\sin(1/t)] \mid 0 < t < 1\} \\ b &= \{[1, t, 0] \mid 0 < t < 1\} \end{aligned}$$

é um conjunto infinito de pontos.

Entretanto, conforme descrito por Rezende e Stolfi [10], este problema não ocorre no caso de *mapas algébricos* (mapas cujos elementos são conjuntos semi-algébricos), pois a sobreposição de dois mapas algébricos é um mapa algébrico, e todo conjunto semi-algébrico tem um número finito de componentes conexas.

Portanto, a sobreposição de mapas esféricos está bem definida, pois todo mapa esférico é um mapa algébrico. Na verdade, note que a interseção entre duas arestas de um mapa esférico consiste de no máximo duas componentes conexas (arcos ou pontos).

11.2 Um algoritmo incremental de sobreposição

Observe que $\oplus$ é uma operação associativa e comutativa, cujo elemento neutro é o mapa trivial $\mathcal{M} = (T, \{T\})$; isto é, o mapa com uma única face (sem fronteira). Além disso, pode-se verificar facilmente que todo mapa esférico $\mathcal{M}$ pode ser descrito como a sobreposição dos mapas elementares $\{\mathcal{M}_v : v \in V_\mathcal{M}\}$ e $\{\mathcal{M}_e : e \in E_\mathcal{M}\}$, onde $\mathcal{M}_v$ é um *mapa-vértice* contendo apenas o vértice v e uma face; e $\mathcal{M}_e$ é um *mapa-aresta* contendo

apenas a aresta e, seus vértices extremos (se houver) e uma ou duas faces, dependendo da aresta ser um arco, um laço ou uma oval. Por exemplo, veja a figura 11.2.

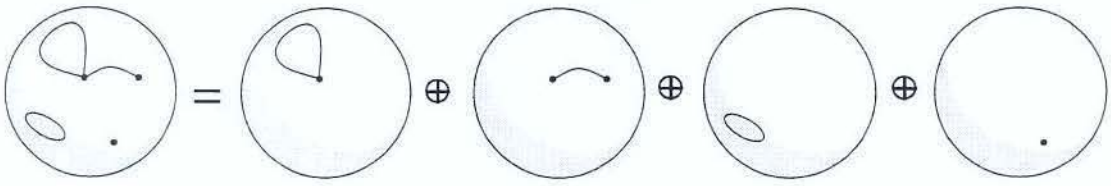


Figura 11.2: Decomposição de um mapa em mapas elementares.

Portanto, podemos concluir que o mapa $\mathcal{M}_1 \oplus \mathcal{M}_2$ pode ser obtido sobrepondo-se ao mapa $\mathcal{M}_1$ cada um dos vértices (pontos) e cada uma das arestas (arcos ou ovais) do mapa $\mathcal{M}_2$. Logo, para resolvermos o problema da sobreposição de mapas esféricos, basta elaborarmos algoritmos para tratar cada uma destas operações elementares.

11.3 Inclusão de um ponto no mapa

Conceitualmente, a operação de incluir um ponto p num mapa $\mathcal{M}$ corresponde a computar a sobreposição do mapa $\mathcal{M}$ com um mapa-vértice $\mathcal{M}_v$, onde $p = point(v)$. Ao final, este algoritmo retorna uma esquina c do mapa resultante tal que $Org(c)$ é o vértice que contém o ponto p. Basicamente, o algoritmo consiste em localizar o ponto p no mapa $\mathcal{M}$ e chamar o procedimento LocalOverlayPoint que efetua a inserção propriamente dita.

```
procedure OverlayPoint(p : CPoint; M : Map) : Corner;
    l ← Locate(p, M);
    return LocalOverlayPoint(p, l);
```

O procedimento LocalOverlayPoint recebe como parâmetros o ponto p a ser incluído no mapa e um endereço l deste ponto no mapa, e insere p no elemento $l.h$ procedendo da seguinte forma: se p está numa face f, então basta criar um novo vértice isolado com geometria p na fronteira desta face; se p está numa aresta e, então basta quebrar esta aresta inserindo sobre ela um vértice com geometria p; e, finalmente, se p está num vértice então não há nada a fazer. Ao final, o procedimento retorna a esquina c tal que

$point(Org(c)) = p$.

```
procedure LocalOverlayPoint(p : CPoint; l : Address) : Corner;
  if l.h ∈ V_M then return l.c;
  else
    if l.h ∈ E_M then
      c ← BreakEdge(l.c);
    else {l.h ∈ F_M}
      c ← MakeVertex();
      Transfer(Bord(c), l.h);
    Org(c).props.geom ← p;
    return c
```

11.4 Inclusão de uma S-curva num mapa

De maneira análoga à inclusão de um ponto no mapa, a operação de incluir uma S-curva A num mapa $\mathcal{M}$ corresponde a computar a sobreposição do mapa $\mathcal{M}$ com um mapa-aresta $\mathcal{M}_e$, onde $sarc(e) = A$. Neste caso, dependendo de A ser um arco ou uma oval, esta operação é realizada por um dos respectivos procedimentos OverlayArc ou OverlayOval.

Visto que a S-curva a ser incluída no mapa pode cruzar vários elementos desse mapa, então esta operação pode resultar na criação de diversos sub-arcos no mapa e conseqüentemente, na criação de diversas novas esquinas. Por isso, os procedimentos OverlayArc e OverlayOval retornam a lista de todas as esquinas c no mapa $\mathcal{M} \oplus \mathcal{M}_e$ tais que a aresta $Edge(c)$ é parte da S-curva inserida e tal que ambas têm a mesma orientação.

11.4.1 O procedimento OverlayArc

O procedimento OverlayArc recebe como parâmetros um S-arco $A = (p, \widehat{s}, q)$ e um mapa $\mathcal{M}$, e efetua a sobreposição de $\mathcal{M}$ com o mapa-aresta $\mathcal{M}_e$, onde e é uma aresta-arco com geometria $sarc(e) = (p, \widehat{s}, q)$.

A idéia básica deste procedimento é a seguinte: primeiramente, inserimos o ponto $A.org = p$ no mapa $\mathcal{M}$ utilizando o procedimento OverlayPoint. Esta operação nos fornece como resultado uma esquina c tal que $point(Org(c)) = p$. Daí, temos que $l = (Org(c), c)$ é um endereço do ponto p no mapa $\mathcal{M}$, e podemos utilizar este endereço para iniciar o processo de inclusão do S-arco propriamente dito, que é realizado pelo procedimento

LocalOverlayArc. Isto é,

```
procedure OverlayArc(A : SArc; M : Map) : List of Corner;
    c ← OverlayPoint(A.org, M);
    l ← (Org(c), c);
    return LocalOverlayArc(A, l);
```

11.4.2 O procedimento OverlayOval

O procedimento OverlayOval recebe como parâmetros um S-círculo s e um mapa $\mathcal{M}$, e efetua a sobreposição de $\mathcal{M}$ com o mapa-aresta $\mathcal{M}_e$, onde e é uma aresta oval com geometria s, isto é, $sarc(e) = (\varnothing, \overset{\frown}{s}, \varnothing)$. A implementação deste procedimento é semelhante à do procedimento OverlayArc. Porém, neste caso não temos um ponto de origem para iniciar o processo de inserção. Para contornar este problema, geramos um ponto p qualquer sobre o S-círculo s e localizamos este ponto p no mapa $\mathcal{M}$, obtendo assim um endereço l_1. Daí,

- se $l_1.h$ é uma face f, então avançamos o ponto p sobre s até obter o primeiro ponto q onde s sai da face f. Se não existir tal ponto, isto é, se o S-círculo está inteiramente contido na face, então, utilizamos o procedimento InsertOval, descrito na seção 11.4.5, para inserirmos uma aresta oval na fronteira da face f;
- se $l_1.h$ é uma aresta e, avançamos o ponto p sobre s até obter o primeiro ponto q onde s sai da aresta e. Se não existir tal ponto, isto é, se e é uma aresta oval, então não há nada a fazer;
- finalmente, se $l_1.h$ é um vértice v, então, tomamos $q = point(v)$;

Nestes três casos, uma vez determinado o ponto q, a inserção do S-círculo s no mapa se reduz à inserção do S-arco $(p, \overset{\frown}{s}, p)$.

---

procedure OverlayOval(s : SCircle; $\mathcal{M}$: Map) : **List of** Corner;
  $p \leftarrow$ um ponto qualquer de s;
  $A \leftarrow (p, \overset{\frown}{s}, p)$;
  $l \leftarrow$ Locate$(p, \mathcal{M})$;
  if $l.h \in F_{\mathcal{M}}$ **then** {p está na face $l.h$}
    $(q, l') \leftarrow$ ClosestFaceExit(A, l);
    if $l'.h = l.h$ **then** {s está inteiramente contido na face $l.h$}
      $c \leftarrow$ InsertOval$(s, l.h, \mathcal{M})$;
      return $\langle c \rangle$;
  elsif $l.h \in E_{\mathcal{M}}$ **then** {p está sobre a aresta $l.h$}
    $m \leftarrow$ WhereTo(p, l, s);
    if $m.h \in E_{\mathcal{M}}$ **then** {os S-círculos s e $scircle(l.h)$ são coincidentes}
      $(q, l') \leftarrow$ ClosestEdgeExit$(A, m.c)$;
      if $q = p$ **then** {o S-arco A termina antes da aresta $m.h$}
        return $\langle m.c \rangle$;
    else {$m.h \in F_{\mathcal{M}}$}
      $q \leftarrow p$; $l' \leftarrow (m.h, m.c)$;
  else {p está sobre o vértice $l.h$}
    $q \leftarrow point(l.h)$; $l' \leftarrow l$;
  {nesta altura, q pertence a um vértice de $\mathcal{M} \oplus \mathcal{M}_e$ e l' é um endereço de q}
  $c \leftarrow$ LocalOverlayPoint(q, l');
  $B \leftarrow (q, \overset{\frown}{s}, q)$;
  return LocalOverlayArc(B, l');

Note que do ponto de vista prático, este procedimento pode ser ligeiramente melhorado observando-se que, se o ponto p que foi gerado sobre o S-círculo s se localiza numa face do mapa, isto é, se $l.h \in F_{\mathcal{M}}$, então não precisamos obter o ponto mais próximo de p onde s sai da face $l.h$; basta obter *qualquer* ponto onde s intercepta a fronteira da face e qualquer endereço para este ponto. Em outras palavras, ao invés de chamarmos o procedimento ClosestFaceExit, basta percorrermos a fronteira da face até obtermos uma esquina c tal que o S-círculo s intercepta o vértice $Org(c)$ ou a aresta $Edge(c)$.

11.4.3 O procedimento LocalOverlayArc

Dados um S-arco A e um endereço l do ponto $A.org$ num mapa $\mathcal{M}$, o procedimento LocalOverlayArc sobrepõe o S-arco A ao mapa, retornando, por definição, a lista de esquinas c do novo mapa tais que $sarc(c) \subset A$ e $scircle(c) = A.scirc$, na ordem em que estes arcos ocorrem sobre A. O procedimento supõe que o ponto $A.org$ pertence a um vértice do mapa, ou seja, supõe que a origem do arco já tenha sido inserida no mapa.

De maneira semelhante ao algoritmo LocateRel, este procedimento simula o movimento de um ponto p sobre o S-arco A. A cada passo, o ponto p ou atravessa uma face ou percorre um trecho de aresta de $\mathcal{M}$. Toda vez que o ponto p cruza uma face, o procedimento acrescenta uma nova aresta ao mapa, conectando os pontos de entrada e de saída da face.

No início de cada passo, o procedimento determina (com WhereTo) um endereço m_1 para o S-vetor (p, s). Se o elemento $m_1.h$ deste endereço é uma aresta e, o procedimento verifica quem termina primeiro: o S-arco A ou a aresta e. No primeiro caso, o procedimento insere um vértice com geometria $A.dst$ sobre a aresta e e avança p para $A.dst$.

Por outro lado, se $m_1.h$ é uma face f, o procedimento determina o ponto q mais próximo de p onde A sai da face f. Caso A termine dentro da face f, então toma-se $q = A.dst$. Além disso, o procedimento determina também um endereço m_2 para o S-vetor $(q, \neg s)$. Se q está sobre a aresta, o procedimento insere na mesma um novo vértice com geometria q. Em todos estes casos, o procedimento acrescenta uma aresta ligando os vértices $Org(m_1.c)$ e $Org(m_2.c)$, e avança o ponto p para o ponto q.

Este processo é repetido até que p alcance o ponto $A.dst$.

```
procedure LocalOverlayArc(A : SArc; l : Address) : List of Corner;
  p ← A.org    s ← A.scirc;    m₁ ← l;
  L ← ⟨⟩; {Lista vazia}
  while true do
    B ← (p, s⌢, A.dst);
    m₁ ← WhereTo(p, m₁, s);
    if m₁.h ∈ E_M then {s = scircle(m₁.h)}
      (q, m₂) ← ClosestEdgeExit(B, m₁.c);
      if m₂.h ∉ V_M then {o S-arco B termina antes da aresta m₁.h}
        c ← LocalOverlayPoint(q, m₂);
        L ← L · ⟨Sym(c)⟩;    {insere a esquina Sym(c) na lista L}
        return L;
      else {o S-arco B não termina antes da aresta m₁.h}
        L ← L · ⟨m₁.c⟩;    {insere a esquina m₁.c na lista L}
    else {m₁.h ∈ F_M}
      (q, m₂) ← ClosestFaceExit(B, m₁);
      c ← LocalOverlayPoint(q, m₂);
      m₂ ← (Org(c), c);
      c ← InsertArc(m₁.c, m₂.c, s);
      L ← L · ⟨c⟩;
      if q = A.dst then return L;
    p ← q;
    m₁ ← m₂
```

11.4.4 O procedimento InsertArc

O procedimento InsertArc recebe como parâmetros duas esquinas c_1 e c_2 de um mapa $\mathcal{M}$ e um S-círculo s, e insere uma nova aresta no mapa conectando os dois vértices $v_1 = Org(c_1)$ e $v_2 = Org(c_2)$ ao longo do S-arco $A = (point(c_1), \overset{\frown}{s}, point(c_2))$. O procedimento supõe que o S-arco A está inteiramente contido na face $f = Left(c_1) = Left(c_2)$. Por definição, as esquinas c_1 e c_2, fornecidas ao procedimento, devem ser tais que $Left(c_1) = Left(c_2)$, $Org(c_1) \neq \varnothing$, $Org(c_2) \neq \varnothing$. Além disso, $m_1 = (f, c_1)$ e $m_2 = (f, c_2)$ devem ser respectivamente os endereços dos S-vetores $(point(c_1), s)$ e $(point(c_2), \neg s)$. Em outras

palavras, as esquinas c_1 e c_2 devem ser tais que o S-círculo s passa pelos vértices v_1 e v_2, e, além disso, no vértice v_1, s entra na face f entre as arestas $Edge(c_1)$ e $Edge(Onext(c_1))$ e, no vértice v_2, s sai da face f entre as arestas $Edge(c_2)$ e $Edge(Onext(c_2))$. Veja a figura 11.3.

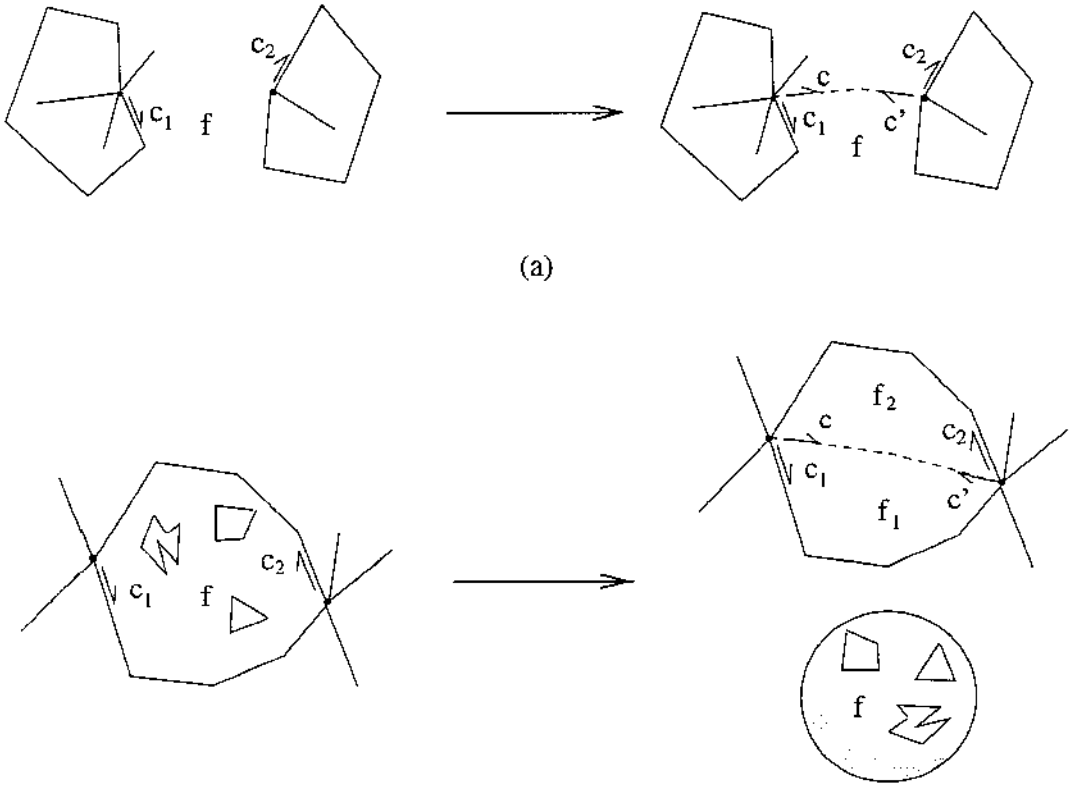


Figura 11.3: Resultado do procedimento InsertArc.

O procedimento InsertArc consiste basicamente na aplicação do operador topológico Connect, descrito na seção 4.7.2. Conforme vimos na descrição daquele operador, se $Bord(c_1) = Bord(c_2)$ — figura 11.3(b) — então, a face f é dividida e substituída por duas novas faces f_1 e f_2. Portanto, neste caso, as bordas adicionais na face f devem ser transferidas para as novas faces f_1 e f_2. Para realizar esta distribuição, o procedimento InsertArc utiliza o procedimento auxiliar DistributeBorders.

Note que o procedimento InsertArc (mais precisamente, o operador Connect) cria duas novas esquinas c e c' sobre a nova aresta; por definição, ele retorna a esquina c, tal que

$scircle(c) = s$.

```
procedure InsertArc(c1, c2 : Corner; s : SCircle) : Corner;
    f ← Left(c1);
    c ← Connect(c1, c2);
    c' ← Sym(c);
    Org(c).props.geom ← point(Org(c1));
    Org(c').props.geom ← point(Org(c2));
    Edge(c).props.geom ← s;
    Edge(c').props.geom ← ¬s;
    if Left(c) ≠ f then
        DistributeBorders(f, Left(c), Left(c'));
```

11.4.5 O procedimento InsertOval

O procedimento InsertOval recebe como parâmetros um S-círculo s e uma face f tal que s está inteiramente contido em f. Daí, ele cria uma aresta oval na fronteira da face f com a geometria s. De modo análogo ao procedimento InsertArc, este procedimento também cria duas esquinas c e c' e, por definição, retorna a esquina c tal que $scircle(c) = s$.

Vale notar que neste procedimento, a face f será substituída por duas novas faces f_1 e f_2, e portanto, as bordas da face f devem ser redistribuídas entre f_1 e f_2, como no procedimento anterior.

```
procedure InsertOval(s : SCircle; f : Face) : Corner;
    c ← MakeEdgeOval();
    Edge(c).props.geom ← s;      Edge(Sym(c)).props.geom ← ¬s;
    DistributeBorders(f, Left(c), Left(Sym(c)));
```

11.4.6 O procedimento DistributeBorders

O procedimento DistributeBorders(f, f_1, f_2) é utilizado quando a inclusão de uma nova aresta no mapa particiona uma face f em duas novas faces f_1 e f_2. Sua função é redistribuir entre as faces f_1 e f_2 as bordas da face f que não foram afetadas pela inserção da nova aresta. O procedimento pressupõe que nenhuma dessas bordas é interceptada pela nova aresta, e portanto, para cada borda b' na face f, basta decidir se um ponto p qualquer sobre b' está em f_1 ou em f_2 e daí, aplicar o operador Transfer correspondente.

Como mostra a figura 11.4, é importante notar que não basta determinar a posição

do ponto p em relação ao S-círculo de suporte da nova aresta. Portanto, a determinação de qual das faces contém o ponto p deve ser realizada utilizando-se o algoritmo Locate. Por questões de eficiência, convém utilizar uma versão especializada do mesmo, que toma como ponto de referência um ponto qualquer da face f_1 (ou f_2).

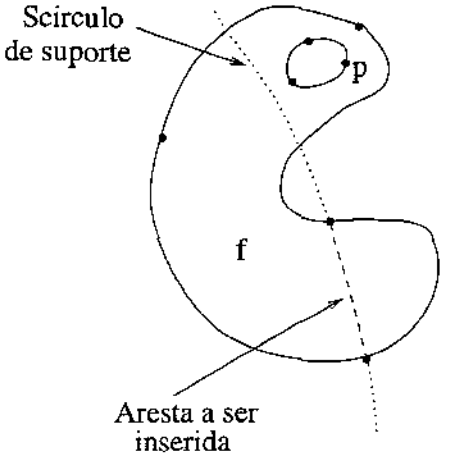


Figura 11.4: Na distribuição das bordas não basta localizar o ponto p em relação ao S-círculo de suporte da nova aresta.

11.5 Complexidade do algoritmo de sobreposição

Vamos agora determinar a complexidade de pior caso do algoritmo de sobreposição de dois mapas cada um deles com $O(n)$ elementos.

É fácil perceber que a complexidade do algoritmo OverlayPoint é $\Theta(n)$, pois a complexidade do algoritmo LocalOverlayPoint é $\Theta(1)$ e a complexidade do algoritmo Locate (versão mais eficiente, seção 10.5) é $\Theta(n)$.

Como vimos no capítulo anterior, a complexidade de ClosestFaceExit é $\Theta(n)$. Por sua vez, a complexidade de InsertArc é determinada pela complexidade do algoritmo DistributeBorders que é $\Theta(n^2)$, pois, para cada borda na face f a ser distribuída, aplicamos o algoritmo Locate. Visto que, no pior caso, o número de bordas a serem distribuídas é $\Theta(n)$, então a complexidade do algoritmo DistributeBorders é $\Theta(n^2)$.

É fácil verificar que a complexidade do algoritmo LocalOverlayArc é $\Theta(n)$ vezes o máximo entre as complexidades dos algoritmos ClosestFaceExit e InsertArc. Por exemplo, quando a S-curva a ser inserida intercepta todas as $\Theta(n)$ arestas do mapa. Portanto, podemos concluir que a complexidade do algoritmo LocalOverlayArc, no pior caso, é $\Theta(n^3)$.

Visto que a complexidade da inserção de uma S-curva num mapa (isto é, a complexidade dos algoritmos OverlayArc e OverlayOval), é determinada pelo máximo entre as

complexidades dos algoritmos OverlayPoint e LocalOverlayArc, então concluímos que esta operação também tem complexidade $\Theta(n^3)$, no pior caso.

Conseqüentemente, a complexidade do algoritmo de sobreposição é $\Theta(n^4)$.

Para ilustrar o pior caso do algoritmo de sobreposição, considere a figura 11.5. Nesta figura, suponha que as arestas tracejadas $1, \cdots, n$ sejam inseridas, nesta ordem, no mapa formado pelas n arestas contínuas e pelos n vértices isolados. Note que a inserção de cada aresta implica a realização de $\Theta(n)$ operações Connect, e portanto $\Theta(n)$ operações DistributeBorders, sendo que cada uma destas operações de distribuição deve localizar os n vértices isolados numa face de complexidade $\Theta(n)$.

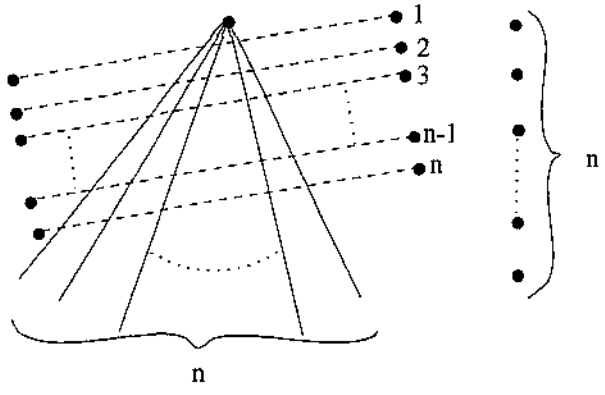


Figura 11.5: Pior caso do algoritmo de sobreposição.

11.6 Uma versão mais eficiente de LocalOverlayArc

Podemos obter uma versão mais eficiente do algoritmo LocalOverlayArc procedendo da seguinte forma: inicialmente, determinamos a lista $q_1, \cdots, q_k$ de todos os pontos onde o S-arco A intercepta todos os elementos (vértices e arestas) do mapa; além disso, para cada um dos pontos q_i determinamos também o seu endereço l_i no mapa. Daí, ordenamos sobre o S-arco A os pontos q_i obtidos, e criamos uma aresta nova conectando cada um dos pares de pontos q_i e q_{i+1}.

É fácil perceber que a complexidade, no pior caso, desta etapa do algoritmo é $\Theta(n \log n)$.

Resta-nos agora, distribuir de maneira mais eficiente, as bordas adicionais das faces que porventura foram particionadas. Note que, como todas as interseções já foram computadas, então podemos adiar estas operações de distribuição realizando-as após todas as conexões dos pontos $q_1, \cdots, q_k$ terem sido efetuadas. Para isto, podemos proceder da seguinte forma: seja Q o conjunto de todas as bordas removidas das faces particionadas; para cada borda b em Q, geramos um ponto p sobre esta borda e, utilizando o algoritmo

Locate, determinamos qual face do mapa $\mathcal{M}$ (com as conexões já efetuadas), contém o ponto p; daí, transferimos a borda b para aquela face. Desta forma, a complexidade, no pior caso, desta operação de distribuição é $\Theta(n^2)$, pois, pode ser necessária a redistribuição de $\Theta(n)$ bordas e o custo da localização de cada borda é $\Theta(n)$.

Portanto, adotando-se esta estratégia, a complexidade do algoritmo LocalOverlayArc passa a ser $\Theta(n^2)$, no pior caso; e a complexidade da operação de sobreposição de mapas passa a ser $\Theta(n^3)$, também no pior caso.

É importante ressaltar que, embora esta otimização melhore o custo de pior caso da operação de sobreposição, na prática, a complexidade média do algoritmo original é $\Theta(1)$, pois em geral, as arestas dos mapas a serem sobrepostos são "curtas", isto é, as arestas de um mapa cruzam $O(1)$ elementos do outro mapa e a fronteira das faces têm $O(1)$ elementos. Portanto, na prática podemos esperar que situações como as da figura 11.5, em que todas as arestas de um mapa cruzam (quase) todas as arestas (e vértices) do outro mapa não são muito freqüentes.

Capítulo 12

Conclusões e trabalhos futuros

A principal conclusão que podemos extrair deste trabalho é que é perfeitamente possível, do ponto de vista teórico, resolver os problemas geométricos de SIGs de maneira exata. Além disso, tudo nos leva a crer que a solução proposta neste trabalho também vale do ponto de vista prático.

12.1 Precisão *versus* números de bits

Os resultados apresentados no capítulo 7, mostram que a representação exata de mapas terrestres não exige um número muito grande de bits. Pelo corolário 23, podemos concluir que para representar um círculo arbitrário sobre a superfície terrestre (cujo raio aproximado é 6375000m) por um S-círculo racional, com uma precisão em torno de 1m, é suficiente representar os coeficientes inteiros com no máximo

$$2 \cdot \left\lfloor \log_2 \left(\frac{6375000m}{1m} \right) \right\rfloor + 9 = 53 \text{ bits por coeficiente}$$

Neste caso, para se representar os pontos de interseção de dois desses S-círculos seriam necessários $2 \cdot 53 + 1 = 107$ bits. Aqueles mesmos resultados também nos permitem concluir que, utilizando números de 64 bits (o tamanho padrão de uma palavra nos computadores modernos), é possível representar exatamente um círculo sobre a superfície terrestre com erro menor do que 25cm. Vale notar que para os padrões de SIGs, esta é uma precisão muito boa.

Por exemplo, suponha que se deseja representar exatamente, com precisão de 1m, um círculo de raio 10m com centro no Edifício Arthur Bernardes (prédio principal) da UFV (Universidade Federal de Viçosa - MG), cujas coordenadas são latitude 20° 45′ 14″ S e longitude 42° 52′ 53″ W. Este ponto corresponde ao ponto da esfera unitária com coordenadas cartesianas $(0.6852160987, -0.636327441, -0.354354508)$. Daí, pelo corolário 22,

tomando $\epsilon = 1/6375000 = 1.5686274 \cdot 10^{-7}$, temos que $\eta = 34688912.84$, $\kappa = 34688926.58$ e $\sigma = 84353288$, e portanto

$$((-2926124965288688, 2005027128140488, -1861972893141016, -1036885124861536))$$

é o S-círculo racional desejado.

Uma observação importante do ponto de vista prático é que os elementos do mapa resultante da sobreposição de dois mapas esféricos podem ser representados exatamente utilizando-se a mesma quantidade de bits dos mapas originais. Portanto, se num determinado SIG a precisão máxima necessária é conhecida, então o número de bits de cada coeficiente dos elementos geométricos (pontos e arcos de círculos) pode ser fixado a priori. Neste caso, torna-se desnecessária a utilização de um pacote de aritmética com precisão arbitrária; e além disso todas as variáveis temporárias podem ser alocadas estaticamente. Isto torna os programas bem mais eficientes.

Com relação à complexidade das operações geométricas básicas, pode-se verificar que todas estas operações têm um grau limitado, sendo que no pior caso elas envolvem testes de ordem 6. Portanto, o número de bits necessários para efetuar tais operações é no máximo 6 vezes o número de bits dos operandos.

12.2 Eficiência dos algoritmos

Obviamente, a utilização de aritmética exata, ao invés de aritmética com precisão fixa, torna os programas menos eficientes. Porém, não faz muito sentido comparar os tempos destas duas soluções, pois, como discutimos no capítulo 1, devido aos erros de arredondamento — que têm efeitos desastrosos para os algoritmos geométricos — a solução obtida utilizando precisão fixa pode ser arbitrariamente diferente da solução correta.

Além disso, vimos que, no pior caso, a complexidade do algoritmo de localização (capítulo 10) é linear no número de elementos do mapa. Porém, na prática é quase sempre possível usar a localização incremental a partir de um vértice "próximo" ao ponto pesquisado. Nestas condições, o custo do algoritmo de localização é, em média, $O(1)$.

Por sua vez, a complexidade do algoritmo de sobreposição (capítulo 11) é, no pior caso, cúbica no número de elementos do mapa. Esta complexidade é dominada basicamente pelo custo da redistribuição das bordas de faces que são particionadas. Felizmente, na prática, em geral um arco a ser inserido no mapa cruza poucas faces e estas têm poucas bordas. Portanto, na prática o custo da sobreposição de dois mapas esféricos tem custo $O(1)$ por aresta, ou seja, $O(n)$ para mapas com n arestas.

12.3 Trabalhos futuros

Certamente, este trabalho levanta diversas questões interessantes, tanto teóricas quanto práticas, que merecem uma melhor investigação.

A primeira delas se refere a tentar tornar o algoritmo de localização de pontos em mapas esféricos mais eficiente. Uma possibilidade é utilizar uma BSP [35] com planos passando pela origem para construir um índice aproximado do mapa. No entanto, esta estratégia não nos permite estabelecer garantias de performance e nem sobre o tamanho deste índice. Nesta linha de raciocínio, uma abordagem alternativa que chegamos a investigar (embora apenas superficialmente) é considerar o arranjo espacial dos planos de suporte dos S-círculos organizando-o em forma de uma partição binária hierárquica do espaço, semelhante à estrutura BSP. Em princípio, isto nos permitiria localizar um ponto arbitrário em tempo $O(\log n)$. No entanto, existem algumas complicações que precisam ser tratadas antes que esta estrutura possa ser aplicada a mapas esféricos, pois a interseção de uma célula da BSP com a esfera não é necessariamente conexa, e muito menos convexa.

Uma outra abordagem que pode simplificar as operações de localização e sobreposição é a *trapezoidalização* do mapa, que consiste em decompor cada face do mesmo em trapezóides definidos por 2 arcos de meridianos e 2 arcos de círculos quaisquer. Note que para obter esta trapezoidalização, seriam incluídas arestas fictícias no mapa. Desta forma, toda face seria um polígono simples de complexidade $O(1)$. Em particular, na operação de sobreposição não seria mais necessário redistribuir as bordas ao particionar uma face.

Do ponto de vista geométrico, um problema em aberto de grande importância prática é a operação de *arredondamento geométrico* de mapas, que consiste em construir uma aproximação racional de um mapa esférico, com um número preestabelecido de bits por coordenada, preservando a consistência topológica. O objetivo aqui seria tentar adaptar os trabalhos de Greene e Yao [14] e de Guibas e Marimont [15] para o caso de mapas esféricos.

Outro problema geométrico importante é a geração de sistemas de referência locais, sobre planos tangentes à esfera com origem e eixos racionais, utilizando poucos bits, para permitir uma integração eficiente de dados locais. Além disso, também seria interessante elaborar um algoritmo para a operação de *engordamento* de um mapa esférico, que informalmente consiste em aumentar a "espessura" dos vértices e arestas do mapa.

Finalmente, duas questões de implementação que também podem resultar em trabalhos interessantes são o desenvolvimento de uma interface para visualizar os mapas esféricos, e a implementação dos algoritmos geométricos exatos, descritos neste trabalho, seguindo os padrões definidos pela biblioteca geométrica CGal [30].

Apêndice A

Conceitos básicos de topologia

Este apêndice apresenta de maneira bastante resumida alguns conceitos topológicos utilizados neste trabalho. Para maiores detalhes, recomendamos consultar um livro texto da área, como por exemplo [18, 23].

A.1 Espaço topológico; abertos e fechados

Um *espaço topológico* é um par $(X, \mathcal{A})$, onde X é um conjunto qualquer (os *pontos*), e $\mathcal{A}$ é uma coleção de subconjuntos de X (os *abertos*), satisfazendo os seguintes axiomas:

1. o conjunto X e o conjunto vazio são abertos;
2. a união de qualquer coleção de subconjuntos abertos é um aberto;
3. a intersecção de uma coleção *finita* de subconjuntos abertos é um aberto.

Nessas condições, dizemos que a coleção $\mathcal{A}$ *determina uma estrutura topológica* sobre o conjunto X, ou que *é uma topologia para* X.

No resto deste apêndice, vamos supor que $(X, \mathcal{A})$ é um espaço topológico.

A.2 Subconjuntos fechados

Por definição, um subconjunto Z de X é *fechado* se seu complemento $X \setminus Z$ é aberto. Portanto, X e $\{\}$ são fechados; a intersecção de qualquer família de subconjuntos fechados é fechada; e a união *finita* de subconjuntos fechados é fechada.

Em geral, num espaço $(X, \mathcal{A})$ existem subconjuntos de pontos que não são nem abertos nem fechados.

A.3 Sub-espaços

Dizemos que um espaço $(Z, \mathcal{B})$ é um *sub-espaço* de $(X, \mathcal{A})$ se e somente se $Z \subseteq X$, e

$$\mathcal{B} = \{A \cap Z | A \in \mathcal{A}\}$$

Neste caso, dizemos que $\mathcal{B}$ é a *topologia de Z induzida por $\mathcal{A}$.*

A.4 Produto cartesiano

O *produto cartesiano* de dois espaços topológicos $(X, \mathcal{A})$ e $(Y, \mathcal{B})$ é o espaço cujos pontos são os pares $X \times Y$, e cujos abertos são todas as uniões (finitas e infinitas) de produtos $A \times B$ com $a \in \mathcal{A}$ e $B \in \mathcal{B}$.

A.5 Topologia natural de $\mathbb{R}^n$

Um *intervalo aberto* de números reais é um conjunto da forma

$$\{x | a < x < b\}$$

onde a e b são reais com $a < b$.

Por definição, na *topologia natural* da reta $\mathbb{R}$, um subconjunto é aberto se e somente se ele é a união (possivelmente infinita) de intervalos abertos.

Para o espaço $\mathbb{R}^n$, com $n > 1$, a *topologia natural* é a definida pela regra do produto cartesiano acima. Ou seja, um subconjunto de $\mathbb{R}^n$ é aberto se e somente se ele é a união (possivelmente infinita) de produtos de n intervalos abertos de $\mathbb{R}$.

Se Z é um subconjunto de $\mathbb{R}^n$, a *topologia natural* de Z é a induzida sobre Z pela topologia natural de $\mathbb{R}^n$.

Em particular, a topologia natural da esfera $\mathbb{S}^2$ é a induzida na mesma pela topologia natural de $\mathbb{R}^3$. Identificando-se os pontos de $\mathbb{S}^2$ com os do plano projetivo orientado $\mathbb{T}^2$, esta mesma regra define a *topologia natural* para $\mathbb{T}^2$.

Sempre que mencionarmos um subconjunto de pontos de $\mathbb{R}^n$, $\mathbb{S}^n$, ou $\mathbb{T}^n$ como um espaço topológico, sem especificar uma topologia, estaremos supondo a topologia natural.

A.6 Vizinhança

Uma *vizinhança* de um ponto $x \in X$ é qualquer subconjunto aberto de X que inclui x. Por extensão, uma vizinhança de um conjunto $Z \subseteq X$ é qualquer subconjunto aberto de X que contém Z.

Na topologia natural de $\mathbb{R}^n$, toda bola com centro x é uma vizinhança de x.

A.7 Ponto de acumulação

Dizemos que um elemento $p \in X$ é um *ponto de acumulação* de um conjunto $Z \subseteq X$ se e somente se qualquer vizinhança de p contém um número infinito de pontos de Z.

Dizemos que uma seqüência infinita de pontos $p_1, p_2, \ldots$ de um espaço X *converge* para um conjunto de pontos $Z \subseteq X$ se qualquer vizinhança de Z contém todos os pontos da seqüência, exceto por um subconjunto finito. Se uma seqüência converge para um conjunto com um único ponto, dizemos que esse ponto é *o limite* da seqüência.

A.8 Fecho, interior, e fronteira

Se $Z \subseteq X$, definimos o *interior de Z (em X)* como sendo a união de todos os subconjuntos abertos de X contidos em Z.

Definimos também o *fecho de Z (em X)* como sendo a intersecção de todos os subconjuntos fechados de X que contêm Z.

A *fronteira de Z (em X)* é a diferença entre o fecho de Z e o interior de Z.

A.9 Espaço quociente

Se $\equiv$ é uma relação de equivalência sobre os pontos de um espaço $(X, \mathcal{A})$, definimos o *espaço quociente* $(X, \mathcal{A})/\equiv$ como sendo $(Y, \mathcal{B})$, onde Y é o conjunto $X/\equiv$ das classes de equivalência de X por $\equiv$; e $\mathcal{B}$ é a família de todos os conjuntos $A/\equiv$, para todo $A \in \mathcal{A}$ que é união de classes de $X/\equiv$.

Espaços quocientes aparecem naturalmente quando uma superfície é descrita por meio da colagem de faces. Nesse caso, o espaço $(X, \mathcal{A})$ original é a união de polígonos regulares isolados, um para cada face. A relação de equivalência define a maneira como as beiradas dessas faces devem ser coladas. Isto é, se o ponto x numa aresta de um polígono deve ser colado com o ponto y em outra aresta (de outro polígono, ou do mesmo), então por definição $x \equiv y$; exceto por este caso, $x \equiv y$ se e somente se $x = y$.

Outro exemplo clássico de espaço quociente é o plano projetivo não orientado, que pode ser definido como o quociente da esfera $\mathbb{S}^2$ (com a topologia natural) pela relação de equivalência que identifica cada ponto da mesma com seu antípoda.

A.10 Separabilidade

Dois pontos de um espaço topológico são *separáveis* se existe um conjunto aberto que contém um mas não o outro; caso contrário eles são *inseparáveis*.

Do ponto de vista topológico, dois pontos inseparáveis são essencialmente indistinguíveis

Um espaço é *separável* se todos os pares de pontos são separáveis. Não se perde muita coisa interessante se nos restringimos a espaços separáveis; pois o quociente de qualquer espaço por sua relação de inseparabilidade é um espaço separável, cujos abertos correspondem aos abertos do espaço original de maneira biunívoca e compatível com $\subseteq$.

A.11 Continuidade

Definimos uma *continuidade* de um espaço topológico $(X, \mathcal{A})$ para outro espaço topológico $(Y, \mathcal{B})$ como sendo uma função de X para Y tal que a imagem inversa de todo subconjunto fechado de Y é um subconjunto fechado de X.

A.12 Equivalência topológica

Um dos conceitos fundamentais da topologia é o de *homeomorfismo*: uma bijeção entre dois espaços é um homeomorfismo se e somente se ela leva conjuntos abertos para conjuntos abertos. Dizemos que dois espaços topológicos são *homeomorfos*, ou *equivalentes*, se e somente se existe um homeomorfismo entre eles.

Decorre da definição que uma bijeção entre dois espaços é um homeomorfismo se e somente se ela é contínua, e sua inversa também é contínua.

Uma *propriedade topológica* é uma propriedade de pontos e subconjuntos de um espaço X que pode ser definida apenas em termos de conjuntos abertos de X (e das operações da teoria de conjuntos). Um exemplo é o fato de um conjunto de pontos ser fechado: que significa, por definição, que o seu complemento é aberto. Outro exemplo é o fato de um espaço X ser *conexo*, que significa que não existe nenhum subconjunto próprio e não-vazio de X que é ao mesmo tempo fechado e aberto. Ainda outro exemplo é fato de um conjunto de pontos Z num espaço X *separar* dois pontos $u, v \in X \setminus Z$, significando que não existe nenhum sub-espaço de $X \setminus Z$ que seja conexo e contenha u e v.

Portanto, um homeomorfismo f preserva todas as propriedades topológicas de pontos e subconjuntos de pontos. Em particular, se um espaço é conexo, qualquer espaço homeomorfo a ele também será conexo; se Z separa u e v, então $f(Z)$ separa $f(u)$ de $f(v)$. E assim por diante.

Apêndice B

Implementação

A estrutura de dados SMC, juntamente com os seus operadores topológicos e todos os algoritmos para a manipulação dos mapas esféricos, descritos nos capítulos anteriores, foram implementados em Modula-3 [28], produzindo ao final uma biblioteca denominada SMap que nos permite representar e manipular exatamente os mapas esféricos.

A arquitetura do sistema por nós implementado é bastante simples, sendo composta basicamente por três módulos: SMTop, SMGeo e SMap, além de uma biblioteca de geometria projetiva com precisão arbitrária LibOPG; veja a figura B.1.

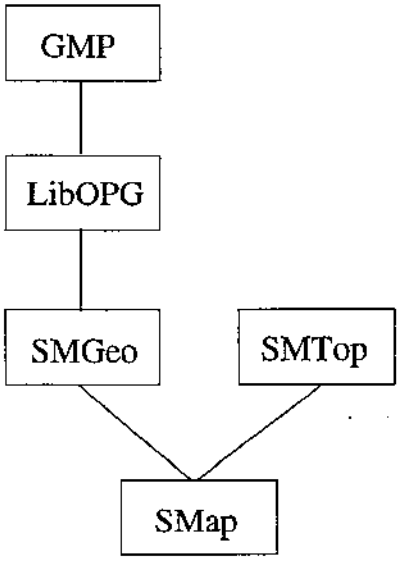


Figura B.1: Arquitetura do sistema para representação e manipulação exatas de mapas esféricos.

B.1 Módulo SMTop

No módulo SMTop — *spherical maps topology* — implementamos a estrutura de dados SMC juntamente com os operadores topológicos definidos nos capítulos 4. Mais especificamente, os tipos Corner, Vertex, Edge, Border e Face e os operadores JoinOrSplitOrbits, JoinVerticesJoinBorders, SplitVertexSplitBorder, JoinVerticesSplitBorder e SplitVertexJoinBorders.

B.2 Módulo SMGeo

No módulo SMGeo — *spherical maps geometry* — implementamos a representação geométrica dos mapas esféricos. Mais especificamente, definimos a representação dos pontos dos conjuntos $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ (capítulos 7 e 8) e também dos pontos de $\mathbb{S}^2$ (em certas situações, utilizados como dados de entrada e também, para a visualização dos mapas). Além disso, definimos também a representação de retas e planos.

Nas definições destas representações geométricas, utilizamos uma variação dos módulos geométricos criados por J. Stolfi [34] para implementar a geometria projetiva orientada. A modificação que fizemos nestes módulos foi substituir a representação numérica de precisão fixa (ponto flutuante) para precisão arbitrária. Para isto, implementamos inicialmente uma primeira versão utilizando o pacote de precisão arbitrária BigNum [32], do laboratório da DEC de Paris. No entanto, este pacote se revelou bastante restrito (em particular, ele não permite verificar exatamente se um número é um quadrado perfeito); portanto, implementamos uma nova versão baseada no pacote de aritmética exata GMP (*The GNU Multiple Precision Arithmetic Library*) [13] da Free Software Foundation.

Além de definir estas representações, todas os procedimentos geométricos exatos descritos no capítulo 9 também são implementados neste módulo.

B.3 Módulo SMap

Finalmente, no módulo SMap — *spherical maps* — integramos os dois módulos SMTop e SMGeo de modo a desenvolver as rotinas de manipulação de mapas esféricos como um todo. Neste módulo implementamos todos os procedimentos auxiliares necessários à implementação dos algoritmos de localização e sobreposição descritos nos capítulos 10 e 11, respectivamente.

Bibliografia

[1] A. Aho, D. S. Jonhson, R. M. Karp, S. R. Kosaraju, C. C. McGeoch, C. H. Papadimitriou, and P. Pevzner. Theory of computing: Goals and directions. Manuscript, 1996.

[2] B. Baumgart. A polyedron representation for computer vision. In *Proc. National Computer Conference*, pages 589–596, 1975.

[3] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete Comput. Geom.*, 9:387–426, 1993.

[4] P. R. Cavalcanti, P. C. P. Carvalho, and L. F. Martha. Nonmanifold modeling: an approach based on spatial subdivisions. Manuscript, 1995.

[5] CG Impact Task Force. Applications challenges to computational geometry. Technical Report TR-521-96, Princeton University, 1996.

[6] G. Câmara, M. A. Casanova, A. S. Hemerly, G. C. Magalhães, and C. M. B. Medeiros. *Anatomia de Sistemas de Informações Geográficas.* X Escola de Computação - Campinas - Brazil, 1996.

[7] H. S. M. Coxeter. *The Real Projective Plane.* McGraw-Hill, 1949.

[8] H. S. M. Coxeter and S. L. Gretizer. Geometry revisited. *Math. Assoc. Amer.*, 1967.

[9] J. Little D. Cox and D. O'Shea. *Ideals, Varieties, and Algorithms.* Springer-Verlag, 1992.

[10] P. J. de Rezende and J. Stolfi. *Fundamentos de Geometria Computacional.* IX Escola de Computação - Recife - Brazil, 1994.

[11] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Heidelberg, West Germany, 1987.

[12] J. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices Amer. Math. Soc.*, 7:646, 1960.

[13] T. Granlund. The GNU multiple precision arithmetic library. Technical report, Free software foundation, 1996.

[14] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Ann. IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[15] L. Guibas and D. Marimont. Rounding arrangements dynamically. In *Proc. 11th Ann. ACM Sympos. Comput. Geom.*, pages 190–199, 1995.

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

[17] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. In *Proc. 5th Ann. ACM Sympos. Comput. Geom.*, pages 208–217, 1989.

[18] M. Henle. *A Combinatorial Introduction to Topology.* Dover, 1979.

[19] J. G. Hocking and G. S. Young. *Topology.* Dover, 1988.

[20] C. M. Hoffmann. *Geometric and Solid Modeling: an introduction.* Morgan Kaufmann, 1989.

[21] C. M. Hoffmann. The problems of accuracy and robustness in geometric computation. *IEEE Computer*, (22):31–42, 1989.

[22] P. Leinhardt. Subdivision of n-dimensional spaces and n-dimensional generalized maps. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 228–236, 1989.

[23] E. L. Lima. *Elementos de Topologia Geral.* Livros técnicos e científicos, 1976.

[24] S. Lins and A. Mandel. Graph-encoded 3-manifolds. *Discrete Mathematics*, 3(57):261–284, 1985.

[25] D. J. Maguire, M. F. Goodchild, and D. Rhind. *Geographical Information Systems - principles and applications.* John Wiley & Sons, 1991.

[26] M. Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, 1988.

[27] V. J. Milenkovic. Practical methods for set operations on polygons using exact arithmetic. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 55–60, 1995.

[28] Greg Nelson, editor. *Systyms programming with Modula-3.* Prentice Hall, 1991.

[29] J. O'Rourke. *Computational Geometry in C.* Cambridge, 1993.

[30] Mark H. Overmars. Designing the Computational Geometry Algorithms Library CGAL. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry (Proc. WACG '96)*, volume 1148 of *Lecture Notes Comput. Sci.*, pages 53–58. Springer-Verlag, 1996.

[31] R. M. Persiano. Representação computacional de subdivisões de variedades bidimensionais. Manuscript, 1995.

[32] B. Serpette, J.Vuillemin, and J. C. Hervé. BigNum: a portable and efficient package for arbitrary-precision arithmetic. Technical Report 2, Digital Paris Research Laboratory, 1989.

[33] J. Stolfi. *Oriented Projetive Geometry - A framework for geometric computations.* Academic Press, 1991.

[34] *Stolfi's home page.* http://www.dcc.unicamp.br/~stolfi.

[35] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. *Comput. Graph.*, 21(4):153–162, 1987. Proc. SIGGRAPH '87.

[36] Kevin J. Weiler. *Topological structures for geometric modeling.* PhD thesis, Rensselaer Polytechnic Institute, 1986.

[37] J. H. Wilkinson. Error analysis of floating-point computation. *Journal Numerische Mathematik*, 2:319–340, 1960.

[38] C. K. Yap. Towards exact geometric computation. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 405–419, 1993.

[39] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific Press, Singapore, 2nd edition, 1995.

Índice Remissivo