Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica

Composição de Mapas Planares e Planejamento de Rotas Aplicados à Navegação de Robôs Móveis e Linhas de Transmissão

Alexandre Ramos Fonseca

Dissertação de mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Renato Cardoso Mesquita

Co-orientador: Prof. Guilherme Augusto Silva Pereira

Belo Horizonte, Março de 2006

Resumo

Encontrar o caminho de menor custo entre dois pontos em um mapa temático é um problema comum. Entretanto, esse planejamento pode se tornar complexo levando-se em conta o elevado número de variáveis e restrições do problema.

Essa dissertação propõe o uso de técnicas de sobreposição de mapas e de otimização para encontrar uma aproximação da rota ótima, considerando todas as informações topológicas e restrições necessárias. Além disso, uma técnica de pós-processamento para refinar a solução é apresentada.

O problema pode ser modelado, sem perda de generalidade, por um conjunto de mapas temáticos aos quais funções de custo são associadas a cada região, com o objetivo de estimar sua dificuldade de transposição.

A técnica de sobreposição de mapas é usada para combinar as informações dos mapas temáticos. Essa técnica produz um mapa combinado contendo as informações de cada mapa e o mapa resultante é então decomposto em regiões convexas utilizando uma triangulação.

Após a discretização, os vértices dos triângulos correspondem aos nós do grafo. O grafo é construído levando em conta o custo de transposição entre os nós e a distância topológica entre eles. Esta distância é definida como o número mínimo de arestas entre os nós. Pode-se assegurar a melhor solução somente quando o grafo completo é considerado. Entretanto, o custo computacional para este caso é proibitivo. Uma técnica de pós-processamento é proposta para encontrar boas soluções sem considerar um número elevado

de conexões. O algoritmo de Dijkstra é utilizado para calcular o caminho inicial.

Como aplicação, dois problemas de planejamento de rotas são considerados: navegação de robôs em ambientes externos e o projeto de rotas de linhas de transmissão. Em ambos os casos, exemplos práticos foram usados para a validação do método desenvolvido.

Abstract

Finding the shortest path between two points in thematic maps is a common problem. However, this planning can be slightly complex, taking into account the large number of variables and constraints of the problem.

This thesis proposes the use of map overlay and optimization techniques to find the optimal route approximation, considering all necessary topological information and constraints. Furthermore, a post-processing technique to refine the solution is shown.

The problem can be modeled by a set of thematic maps without loss of generality. Cost functions are assigned to each region in order to estimate the difficulty to transpose that region.

The map overlay technique is used to couple all the thematic map information. This technique produces a combined map which contains all the information of each map. A triangle discretization has been used to decompose the map in convex regions.

After the discretization, the vertices of the triangles are the nodes of the search graph. The graph is constructed taking into account the distance between nodes. This distance is defined as the minimal number of edges between the nodes. One can assure the best solution only considering all possible connections. However, this addresses the worst case and the computational cost is prohibitive in most cases. A post-processing technique has been proposed to find good solutions without a significative increasing of considered connections. To achieve the initial solution path, the Dijkstra

algorithm has been used.

Two route planning problems are addressed in this work, the robot motion planning in outdoor environments and the design of routes for transmission lines. In both cases, practical systems have been used to test the developed method.

Agradecimentos

Primeiramente a Deus pela minha vida e pela conclusão dessa importante etapa de minha vida.

À minha família por sempre me apoiar.

Aos meus orientadores Renato Mesquita e Guilherme Pereira pela oportunidade que me foi dada e pelo auxílio valioso.

Aos amigos do GOPAC (Grupo de Otimização e Projeto Assistido por Computador) pelo companheirismo e pela ajuda. Em especial: Adriano Lisboa, Cássia Nunes, Eduardo Carrano, Prof. Elson Silva, Felipe Terra (*Siri*), Guilherme Parreira, Juliano Correa (*Balão*), Leonardo de Queiroz, Leonardo Mozelli (*SWKid*), Luciano Pimenta (*Luizinho*) e Ricardo Adriano.

Aos meus amigos por entenderem meus longos períodos de ausência.

A todos que direta ou indiretamente contribuíram para a realização desse trabalho.

A todos vocês, muito obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

α_e	Custo da aresta e
α_e	Custo total de um segmento de reta que liga dois pontos A e B
α_f	Custo da face f
$\cap(f, f')$	Aresta compartilhada pelas faces f e f' orientada de tal forma que f está à sua direita
ϵ	Fator de precisão
$\mathbb{R}^2$	Plano real
$\mathcal{C}_{forb}$	Região proibida no mapa
$\mathcal{C}_{forb}^{\eta}$	Região proibida aumentada de uma distância η
$\mathcal{C}_{free}$	Região livre no mapa
$\mathcal{E}$	Seqüencia de arestas
$\mathcal{L}$	Curva sobre a região livre do mapa
$\mathcal{M}$	Conjunto de mapas temático
$\mathcal{P}$	Seqüência de pontos especificando um caminho linear
$\mathcal{W}$	Mapa temático composto
$\oplus$	Soma de Minkowski
θ	Ângulo de incidência
θ'	Ângulo de refração
θ_c	Ângulo crítico

Υ_f	f-ésima face do mapa
C_i	i-ésimo ciclo de um mapa planar
C_∞	Ciclo externo da face infinita
$d_{r,\mathcal{E}}(x)$	Distância ponderada de r até x sobre o caminho geodésico que passa por $\mathcal{E}$
ds	Diferencial de comprimento de arco
e	Aresta
f	Face
$f \cap f'$	Aresta compartilhada pelas faces f e f'
f_∞	Face infinita
$g(x, y)$	Função de custo para o mapa composto
I	Funcional de custo
$int(\cdot)$	Interior relativo de uma aresta ou face
M_i	i-ésimo mapa temático
$O(\cdot)$	Ordem de complexidade
P	Conjunto de pontos
p	Caminho geodésico
r	Raiz de um caminho geodésico
T	Triangulação
w_i	Peso do i-ésimo mapa temático
$z_i(x, y)$	Função de custo para o i-ésimo mapa temático

Capítulo 1

Introdução

1.1 Motivação

Considere um robô móvel que atravessa um terreno composto por diferentes regiões tais como areia, grama, pavimento e obstáculos. A Figura 1.1 ilustra um mapa temático para um exemplo como esse. Suponha que o robô se movimente com maior facilidade na região de pavimento do que na região de grama, que por sua vez é de mais fácil transposição que a região de areia. Um obstáculo intransponível é mostrado em preto. A rota que minimiza o tempo gasto pelo robô para sair do ponto de origem s e alcançar o ponto de destino t, evitando o obstáculo, é mostrada pela linha tracejada. Note que o caminho evita as regiões de custo mais elevado (areia e grama) e procura permanecer o máximo possível no pavimento onde o desempenho é maior. Entretanto, existem casos em que, mesmo perdendo eficiência, é desejável que o robô atravesse as regiões de custo mais elevado, dependendo do tamanho e disposição dessas regiões.

Pode-se tratar o problema de planejamento do traçado de linhas de transmissão de forma semelhante. Nesse caso, os mapas temáticos podem re-

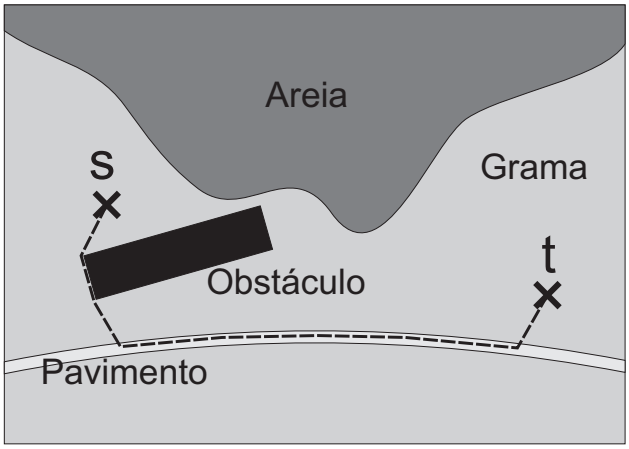


Figura 1.1: Exemplo de rota de custo mínimo

presentar restrições de clima, vegetação, relevo, áreas de risco ocupacional, travessias, restrições ambientais, custo de desapropriação, dentre outros.

1.2 Objetivos

Este trabalho tem como objetivo o desenvolvimento de uma ferramenta computacional capaz de realizar a composição de mapas temáticos que representam características específicas de um ambiente, gerando um mapa resultante. Também é objetivo que, sobre o mapa resultante, essa ferramenta seja capaz de determinar a rota de menor custo entre dois pontos quaisquer.

1.2.1 Aplicações

Ao longo desse trabalho, serão contempladas duas aplicações possíveis para essa ferramenta computacional: a determinação de rotas para a navegação de robôs móveis em ambientes externos e o auxílio ao projeto de linhas de transmissão na escolha de seu traçado (Figura 1.2).

(a) Robôs móveis (b) Linhas de Transmissão

Figura 1.2: Aplicações de interesse. Fotos retiradas dos sites a)http://www.cc.gatech.edu.ai e b)http://www.me3.org

Navegação de robôs móveis

O problema de navegação de robôs móveis consiste basicamente em determinar como conduzir o robô de um lugar a outro evitando os obstáculos do ambiente. Quando consideramos ambientes internos, exemplos de obstáculos são paredes, escadas, móveis, pessoas e outros objetos que podem, em caso de colisões, comprometer a execução da tarefa e a própria segurança das pessoas e do robô (Latombe 1991). Então, obstáculos podem ser representados por regiões proibidas para o caminho do robô. Em ambientes externos, que são amplos, esparsamente ocupados por terrenos desiguais, as restrições de locomoção do robô devido a suas características e sua interação com o ambiente também passam a ser importantes.

Projeto de linhas de transmissão

O projeto de linhas de transmissão é complexo e compreende diversas etapas: especificação das características elétricas da linha, escolha do traçado, determinação de características mecânicas das torres, levantamento topográfico e, finalmente, a locação das torres sobre o traçado escolhido. O projeto é ditado por normas técnicas rígidas que devem ser observadas pelos projetistas da linha, em todos seus aspectos (ABNT 2003). Além disto, é importante frisar que o projeto de uma linha tem um horizonte mínimo de 30 anos, o que faz com que restrições temporais de uso da linha também sejam levados em conta.

Entre as principais restrições podemos citar esforço transversal máximo, esforço vertical mínimo, altura máxima das torres, altura mínima do cabo ao solo, entre outras.

Nesse trabalho será abordado apenas o problema da escolha do traçado sem levar em consideração as restrições elétricas e mecânicas inerentes ao problema. Dessa forma, o resultado obtido pelo sistema desenvolvido é apenas uma solução inicial que pode ser utilizada como auxílio ao projetista.

1.3 Visão geral do método

1.3.1 O problema do caminho mínimo

O problema do caminho mínimo consiste em encontrar um caminho que ligue dois pontos A e B de um ambiente em que o o custo total para ir de A para B é o menor entre todos os caminhos possíveis.

Em (Mitchell 1991) utilizam-se mapas contínuos para a representação do ambiente e para calcular o menor caminho. O autor utiliza uma versão

contínua do algoritmo de Dijkstra e explora o fato de que o caminho de menor custo obedece a lei de Snell para a refração nas regiões de fronteira. Custos constantes são definidos para diferentes tipos de terreno e são usados pelo algoritmo de minimização. Em (Mata & Mitchell 1997), os autores usam uma abordagem semelhante a (Mitchell 1991), também baseada na lei de Snell. Nesse caso um grafo discreto é construído a partir do mapa contínuo e usado para determinar uma aproximação do caminho ótimo, o que reduz significativamente o custo computacional. Esses métodos são mostrados com mais detalhes no capítulo 3.

Na teoria de grafos (Bondy & Murty 1976, Wilson 1996), o problema do caminho mínimo consiste na minimização do custo de travessia do grafo entre dois nós. O planejamento de rotas para robôs móveis é uma das aplicações para o problema de caminho mínimo e vem sendo amplamente explorado na literatura tanto para problemas em ambientes internos quanto externos (Latombe 1991).

Um grafo é um conjunto de vértices (ou nós) que são conectados através de arestas (ou ramos). Dependendo da aplicação, os ramos podem ser direcionados; pode se permitir que um ramo ligue um nó a ele mesmo; nós e ramos podem ter um peso associado. Se os ramos têm uma direção associada (indicada por uma seta na representação gráfica) tem-se um grafo direcionado, ou dígrafo. Estruturas que podem ser representadas por grafos estão em toda parte e muitos problemas de interesse prático podem ser formulados utilizando os mesmos. Como será apresentado na seção 1.3.2, grafos não direcionados foram utilizados para a representação discreta de mapas temáticos. Nesse caso, o custo do caminho de ida e o de volta entre dois nós do grafo é o mesmo.

Um caminho é uma seqüência de nós tal que cada um dos nós é ligado ao seguinte por um ramo. Um caminho é chamado simples se nenhum dos nós no caminho se repete. O custo de um caminho num grafo é a soma dos custos das arestas atravessadas.

Um dos algoritmos mais utilizados para resolver o problema do caminho mínimo é o algoritmo de Dijkstra (Dijkstra 1959). Este algoritmo garante a obtenção do caminho ótimo entre um nó de referência do grafo e os demais. Uma alternativa ao uso do algoritmo de Dijkstra é o uso do algoritmo A^* (Hart, Nilsson & Raphael: 1968). Esse algoritmo é mais eficiente pois não avalia o grafo como um todo mas apenas os nós mais promissores, usando para isso uma função heuristica. Por outro lado, o algoritmo A^* só garante a obtenção do caminho ótimo se a heurística for admissível. Existe ainda o algoritmo D^* (Stentz 1995), que é uma versão dinâmica do A^*. O D^* é útil para casos onde os custos dos arcos muda, durante o curso de um robô, por exemplo, e o caminho precisa ser re-planejado em tempo real. Nesse trabalho utilizou-se o algoritmo de Dijkstra que pode ser visto em detalhes na seção 3.3.

1.3.2 Metodologia

Nesse trabalho, propõe-se uma metodologia para determinar o caminho de menor custo entre dois pontos em uma dada região. Essa região é representada por um mapa composto que reflete as propriedades de diversos mapas temáticos. Entende-se por mapa temático uma representação gráfica que representa uma característica da região como dados climáticos, econômicos, agrícolas e outros.

Na maioria dos casos, informações das diversas características de uma região estão dispostas em mapas temáticos diferentes, sendo difícil raciocinar

levando em conta todos os critérios em conjunto. A técnica de composição de mapas (*overlay*) (de Berg, van Kreveld, Overmars & Schwarzkopf 2000) é usada para gerar o mapa composto. De posse do mapa composto e do peso associado a cada mapa, calcula-se o custo composto, que, para cada região do mapa composto é igual à soma ponderada do custo das regiões dos mapas originais. A partir daí pode-se utilizar um algoritmo de otimização para encontrar o caminho de menor custo sobre o mapa composto.

Dessa forma, buscou-se desenvolver uma ferramenta computacional capaz de compor mapas temáticos e determinar uma aproximação para o caminho de menor custo. Para determinar o traçado da linha de transmissão foi utilizada a mesma técnica empregada na navegação de robôs. A maior inovação está em se adotar uma solução diferente (e mais eficiente) do que a tradicionalmente utilizada nos sistemas de geoprocessamento. Nestes, para se determinar rotas ótimas em mapas, se trabalha no nível de pixels: a cada pixel é associado um custo discreto e, a partir destes custos individuais, se computa um custo acumulado e o caminho ótimo. A determinação de uma superfície de custo discreta robusta é uma dificuldade enfrentada por estes sistemas (Berry 2005). Por outro lado, ao se atacar o problema com as técnicas da geometria computacional, trabalhando-se com grandes regiões, o problema, do ponto de vista da otimização, torna-se mais robusto e eficiente.

Nos problemas abordados nesse trabalho, os mapas representam ambientes contínuos. Entretanto, utiliza-se o método clássico de Dijkstra. Para isso, faz-se necessário a decomposição do mapa e a partir dessa decomposição gera-se um grafo de pesquisa discreto. Essa decomposição é realizada através de uma triangulação de Delaunay com restrições (CDT - *Constrained Delaunay Triangulation*)(de Berg et al. 2000). Essa técnica decompõe o mapa em triângulos, respeitando as fronteiras definidas pelas arestas originais do mapa.

Para aumentar o grau de discretização do mapa utiliza-se um refinamento da triangulação (Shewchuk 1997). Dessa forma, o caminho gerado pelo algoritmo de busca não fica limitado aos contornos das regiões, podendo passar através delas. Para gerar a triangulação e o refinamento foi usada uma ferramenta gratuita, o *Triangle* (Shewchuk 1996, Shewchuk 2005). Mais detalhes podem ser vistos na seção 3.1.

A melhor aproximação para o caminho ótimo sobre o mapa contínuo só é garantida quando o grafo gerado a partir da discretização do mapa é completo. Ou seja, no grafo, um vértice da triangulação está ligado a todos os outros. Quando consideramos o grafo completo, o número de ramos do grafo é elevado e pode tornar o processamento do algoritmo de busca demorado e muitas vezes proibitivo. Uma solução alternativa para se gerar um caminho próximo ao ótimo foi desenvolvida e baseia-se na eliminação de pontos desnecessários de um caminho previamente calculado usando-se um grafo com um número menor de ramos. Esse procedimento é chamado aqui de pós-processamento e é descrito em detalhes na seção 3.4.

1.4 Trabalhos relacionados

Em (Kobilarov & Sukhatme 2005, Guivant, Nebot, Nieto & Masson 2004, Guo, Parker, Jung & Dong 2003, Yahja, Singh & Stentz 2000, Mitchell 1991, Mata & Mitchell 1997), são apresentadas soluções para o problema de planejamento de rotas para navegação de robôs móveis em ambientes externos.

Em (Guo et al. 2003), o mapa representando o ambiente externo é decomposto em uma grade regular e o algoritmo A^* é usado para calcular o caminho de menor custo. Nesse caso o custo de um caminho é determinado por uma função que combina a rugosidade do terreno e a distância entre

as células. O problema principal com esse tipo de abordagem é que o uso de uma grade é ineficiente para representar ambientes amplos e complexos. (Kobilarov & Sukhatme 2005) também decompõe o ambiente em uma grade regular. Nesse trabalho é definida uma métrica para o cálculo do custo de cada célula baseado no tempo que o robô gasta para transpô-las.

O trabalho de (Yahja et al. 2000) propõe uma decomposição do mapa utilizando *Quadtrees*. Uma *Quadtree* é uma estrutura de dados em árvore em que cada nó interno possui até quatro filhos. *Quadtrees* são freqüentemente utilizadas para subdividir um espaço bi-dimensional recursivamente em quatro quadrantes. A vantagem dessa técnica é a representação não uniforme do mapa utilizando uma alta resolução em regiões de interesse, como obstáculos, e poucas células onde a geometria é pouco complexa. Após a decomposição do mapa em células, utiliza-se o algoritmo D^* para calcular o menor caminho no mapa.

A determinação de rotas ótimas em mapas temáticos também é um tópico de bastante explorado, não somente para o lançamento de linhas de transmissão (Vega & Sarmiento 1996, West, Dwolatzky & Meyer 1997, Boulaxis & Papadopoulos 2002) e navegação de robôs móveis (Kobilarov & Sukhatme 2005, Guivant et al. 2004, Guo et al. 2003, Yahja et al. 2000, Mitchell 1991), mas também em redes de distribuição de energia elétrica (West et al. 1997, Boulaxis & Papadopoulos 2002), na determinação da trajetória ótima de dutos (oleodutos, aquedutos, gasodutos), rodovias e ferrovias, etc. (ver, por exemplo, o capitulo 19 de (Berry 2004)).

1.5 Organização do texto

O capítulo 2 trata a representação de mapas temáticos e como utilizar a soma de Minkowski para ampliar as regiões do ambiente de forma a garantir uma margem de segurança para que o caminho permaneça nas regiões de menor custo. A seguir, é apresentada a estrutura de dados DCEL (*Doubly Connected Edge List*) utilizada para a representação de mapas planares. O capítulo trata ainda de como são atribuídos os custos para as faces e arestas do mapa planar e finalmente como obter o mapa composto.

O capítulo 3 discute o planejamento de rotas dado o mapa composto obtido como descrito pelo capítulo 2. Inicialmente são apresentadas algumas características de caminhos ótimos para a representação contínua do ambiente e a solução dada por (Mitchell 1991). A seguir uma solução simplificada, baseada na construção de grafo discreto a partir do mapa contínuo é apresentado (Mata & Mitchell 1997). Depois disso, a solução implementada é detalhada. O primeiro passo é a discretização do mapa e como, a partir dessa discretização, é construído o grafo de pesquisa. Por fim, é apresentado o algoritmo de Dijkstra utilizado para a determinação do melhor caminho sobre o grafo e a técnica de pós-processamento desenvolvida para aprimorar o caminho sobre o mapa contínuo.

O capítulo 4, apresenta algumas simulações para validar a metodologia implementada. Dois casos hipotéticos são apresentados: um para o problema de navegação de robôs móveis e um para o de linhas de transmissão. O exemplo apresentado para o caso de navegação de robôs foi apresentado em (Fonseca, Pimenta, Mesquita, Saldanha & Pereira 2005). Para o caso de linhas de transmissão foram utilizados dados reais do condado de Oneida, estado de Nova York nos Estados Unidos.

Finalizando, o capítulo 5 apresenta algumas discussões sobre o método implementado e propostas de continuidade.

Capítulo 2

Composição de mapas planares

Considere o caso de um robô operando em um ambiente externo. Esse ambiente pode ser representado por um conjunto de mapas temáticos. Cada mapa temático representa uma característica específica do ambiente como obstáculos, propriedades do solo, inclinação do terreno, intensidade de comunicação, densidade de pessoas em uma determinada região, entre outras.

No caso de linhas de transmissão, os mapas temáticos podem representar características de clima, vegetação, relevo, áreas de risco ocupacional, travessias, restrições ambientais, custo de desapropriação, etc.

Em síntese, nesse trabalho, um mapa temático é representado por um plano subdividido em regiões poligonais onde cada uma dessas regiões possui um custo α especificando o custo por unidade de distância ao se caminhar por essa região.

Nesse capítulo será mostrada a estrutura de dados utilizada para representar e armazenar mapas temáticos e como um conjunto de mapas pode ser combinado para se obter um único mapa composto contendo o custo global associado a todas as restrições consideradas.

2.1 Mapas temáticos

Um ambiente pode ser representado por um conjunto de k mapas temáticos:

$$\mathcal{M} = \{M_1, M_2, ..., M_k\}. \tag{2.1}$$

Cada mapa é definido como sendo

$$M_i = \{(x, y, z_i(x, y))\}, \tag{2.2}$$

onde $0 \leqslant z_i(x, y) \leqslant +\infty$ é uma função que representa um característica específica do ambiente na posição (x, y). Na prática, $z_i(x, y)$ retorna o custo por unidade de distância para se transpor a região no ponto (x, y). Valores altos de $z_i(x, y)$ indicam regiões de alto custo de transposição e, no limite extremo, $z_i(x, y) = +\infty$, representa uma impossibilidade de transposição, um obstáculo que deve ser contornado.

Para analisar o ambiente como um todo, ou um conjunto de características principais de interesse, propõe-se a combinação dos mapas temáticos em um único mapa temático global $\mathcal{W}$ dado por:

$$\mathcal{W} = \{(x, y, g(x, y)|g(x, y) = \sum_{i=1}^{k} w_i z_i(x, y))\}, \tag{2.3}$$

onde w_i são fatores de peso usados para estabelecer prioridade entre os mapas.

Definem-se as regiões proibidas de um mapa, $\mathcal{C}_{forb}$, como sendo as regiões onde $z(x, y) = +\infty$ e, as regiões livres como sendo

$$\mathcal{C}_{free} = \mathcal{W}/\mathcal{C}_{forb}^{\eta}, \tag{2.4}$$

onde $\mathcal{C}_{forb}^{\eta}$ é a região $\mathcal{C}_{forb}$ aumentada de uma distância η.

Para o caso de navegação de robôs, por exemplo, essa constante η especifica uma margem de segurança para que o robô não colida com o obstáculo. Essa margem de segurança pode encapsular o tamanho do robô e/ou erros relativos à sua localização estimada. Um robô pode ser tratado como um único ponto, basta que os obstáculos sejam aumentados de tal forma que, quando o ponto que representa o robô alcança o obstáculo aumentado, algum ponto extremo do robô toca o obstáculo real. Para tratar o robô como um único ponto, deve-se processar a soma de Minkowski entre os obstáculos e o robô rotacionado de 180^o (Pimenta 2005, de Berg et al. 2000).

2.1.1 Soma de Minkowski

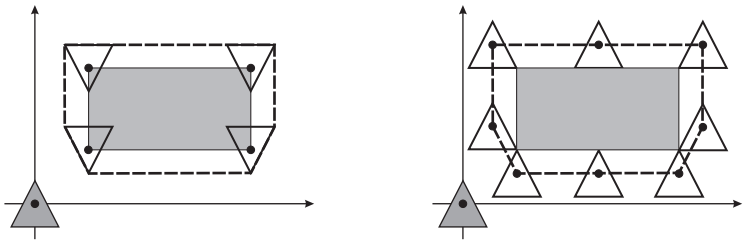
Dado dois conjuntos S_1 e S_2 em $\mathbb{R}^2$, a soma de Minkowski é definida como

$$S_1 \oplus S_2 := \{p + q : p \in S_1, q \in S_2\}, \tag{2.5}$$

onde $p + q$ é a soma vetorial dos vetores p e q, isto é, se $p = (x_p, y_p)$ e $q = (x_q, y_q)$, então

$$p + q := (x_p + x_q, y_p + y_q). \tag{2.6}$$

Dependendo do formato do robô, a soma de Minkowski faz com que os obstáculos cresçam de forma desigual o que torna o valor de η variável, como exemplificado na Figura 2.1. Na Figura 2.1(a), a linha pontilhada é o resultado da soma de Minkowski para o obstáculo e o robô triangular com ponto de referência na origem. A Figura 2.1(b) mostra que quando o ponto de referência do robô toca a linha pontilhada, algum ponto do robô toca o obstáculo real.

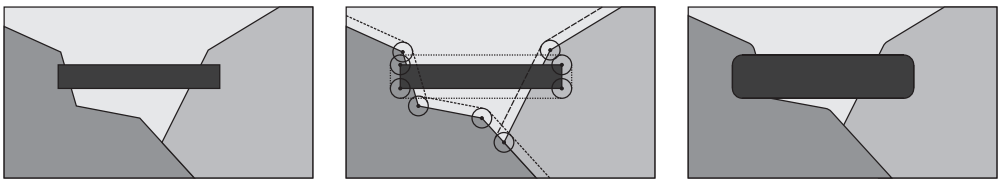


(a) (b)

Figura 2.1: Soma de Minkowski entre obstáculo e robô triangular

Além disso, se o robô é não circular, o resultado da soma de Minkowski depende da orientação do robô. Por simplicidade, podemos considerar o robô como sendo o menor disco que engloba todos os seus pontos e o centro desse disco passa a ser o ponto de referência. Assim, a soma de Minkowski entre os obstáculos e o disco é única e não depende da orientação do robô.

Para mapas temáticos, onde as regiões fazem fronteira umas com as outras, faz-se a soma de Minkowski para cada região de forma que as regiões de maior dificuldade de transposição sobrepõem as regiões de menor dificuldade. Dessa forma, quando consideramos o robô como um único ponto e fazemos com que ele caminhe sobre a fronteira de duas regiões, garantimos que ele se encontra inteiramente dentro da região de menor dificuldade de transposição. A Figura 2.2 mostra um mapa temático antes e depois de se fazer a soma de Minkowski com um robô circular. A dificuldade de transposição das regiões é indicada pela cor da região. Quanto mais escura a região maior a dificuldade de transposição.



(a) mapa original (b) soma de Minkowski (c) mapa final

Figura 2.2: Soma de Minkowski para um mapa temático e um robô circular

2.2 Representação de mapas planares

Mapas temáticos podem ser representados por mapas planares. Um mapa planar, como ilustrado na Figura 2.3, subdivide o plano em vértices, arestas e faces de tal forma que cada aresta é delimitada por seus pontos extremos e cada face delimitada por um conjunto de arestas. Duas faces podem ter uma aresta comum, ter um único vértice em comum ou podem não possuir interseção. Duas arestas nunca se cruzam, exceto em seus pontos extremos.

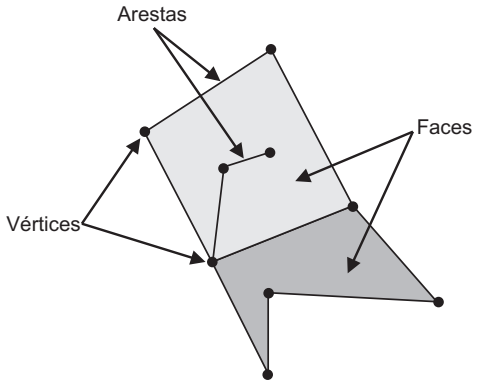


Figura 2.3: Representação de mapas planares

Como mapas temáticos são representados como um conjunto de regiões poligonais, regiões circulares ou curvas são aproximadas por um conjunto de segmentos como ilustrado na Figura 2.4. Quanto maior o número de segmentos melhor a aproximação.

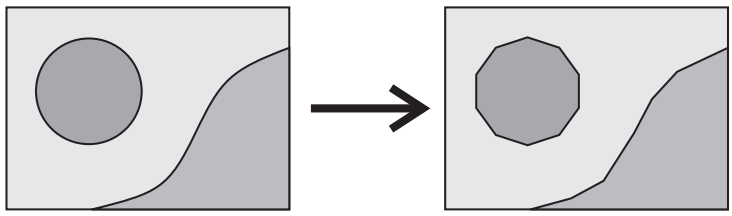


Figura 2.4: Representação de mapas planares

2.2.1 Lista de Arestas Duplamente Conectadas

Uma estrutura de dados apropriada para representar e armazenar mapas planares é a *Doubly-Connected Edge List* (DCEL) (Gangnet, Hervé, Pudet & van Thong 1989, Guibas, Ramshaw & Stolfi 1983, de Berg et al. 2000). Em uma DCEL, cada aresta é representada por duas semi-arestas com orientações opostas. Cada semi-aresta é definida por seus vértices de origem e de destino.

Cada semi-aresta possui um ponteiro para a semi-aresta oposta, chamada semi-aresta gêmea. Se uma semi-aresta e pertence à face f, a semi-aresta gêmea de e pertence a uma face adjacente a f.

Uma face f possui um contorno externo e contornos internos caso exista um ou mais buracos (faces internas a f). As semi-arestas são orientadas de tal forma a circular o contorno externo das faces no sentido anti-horário e os contornos internos no sentido horário. Existe uma face simbólica denominada face infinita (f_∞) que engloba todo o plano. Se o mapa planar está vazio, então a DCEL armazena apenas a face infinita. Se inserimos uma face em uma DCEL vazia, então teremos a face inserida, a face infinita e um buraco sobre a face infinita.

Uma DCEL armazena a informação de vértices, faces e semi-arestas em registros. Dessa forma, a DCEL possui 3 listas com registros, uma para os vértices, uma para as faces e uma para as semi-arestas.

O registro de um vértice v armazena as suas coordenadas e um ponteiro para uma das semi-arestas que possuem v como vértice de origem.

O registro de uma face armazena um ponteiro para uma das semi-arestas que está em seu contorno externo. Para a face infinita esse ponteiro é nulo. Além disso, cada face armazena uma lista de ponteiros para semi-arestas, um para cada buraco existente no seu interior.

O registro de uma semi-aresta e possui ponteiros para seu vértice de

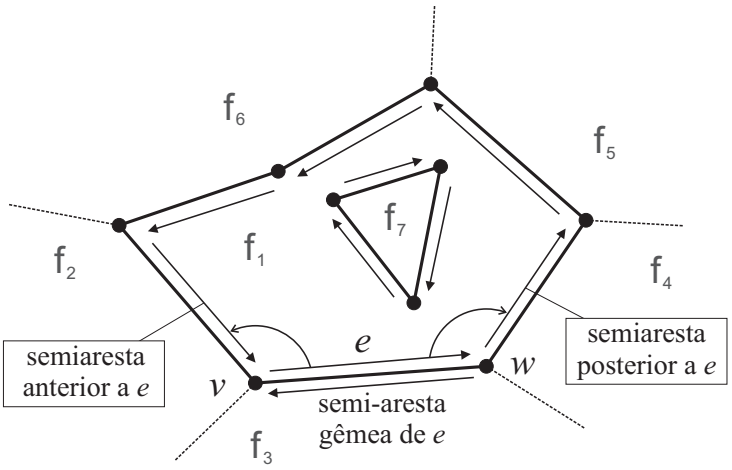


Figura 2.5: Estrutura da DCEL

origem, para as semi-arestas anterior e posterior na mesma face de e, para a sua semi-aresta gêmea e para a face que está à sua esquerda. Não é necessário um ponteiro para seu vértice de destino já que esse é o ponto de origem da aresta gêmea de e. A Figura 2.5 mostra a representação gráfica de uma DCEL. Nessa figura, v e w são os vértices de origem e destino da semi-aresta e.

Além das informações geométricas e topológicas, cada registro deve armazenar ainda um atributo com informações adicionais sobre a região do mapa descrito por ele. No problema considerado, esse atributo é um registro com o custo de transposição e o nome da região.

Nesse trabalho utilizou-se a *Computational Geometry Algorithms Library* (CGAL) (CGAL 2005), uma biblioteca gratuita de geometria computacional, que oferece várias estruturas de dados para a representação de estruturas planares e tridimensionais.

A CGAL provê uma estrutura de dados, a Planar_map_2, que encapsula uma DCEL. Além de podermos percorrer o mapa planar a partir de uma semi-aresta, a estrutura de dados oferecida pela CGAL nos fornece outras

facilidades como a possibilidade de percorrer a lista de faces, arestas e vértices através de iteradores. Maiores detalhes a CGAL e as as estruturas de dados utilizadas podem ser vistas no Apêndice C.

2.3 Custo das faces e arestas

A maneira como se representa a dificuldade para se transpor uma dada região em um mapa temático é similar à utilizada por (Mitchell 1991), onde faces e arestas possuem custos uniformes. A f-ésima face do mapa, $\Upsilon_f \subset \mathbb{R}^2$, representa uma propriedade específica do mapa temático e tem associado um custo por unidade de distância $\alpha_f \in [0, +\infty]$ para transpô-la. Dessa forma, a função custo $z_i(x, y)$, para um mapa temático i, apresentada anteriormente na Equação (2.2) pode ser definida como:

$$z_i(x, y) = \alpha_f \qquad \forall (x, y) \in \Upsilon_f, f = 1, 2, ..., n, \tag{2.7}$$

onde n é o número de faces para o mapa.

Para cada aresta e também é associado um custo $\alpha_e \in [0, +\infty]$. Normalmente, no caso de duas faces adjacentes f e f' o custo associado à aresta que as separa é igual ao menor custo entre as duas faces, $\alpha_e = min\{\alpha_f, \alpha_{f'}\}$, como ilustrado na Figura 2.6. Nesse caso, caminhar sobre a fronteira é o mesmo que caminhar por dentro da região de menor custo.

Entretanto, existem casos onde o custo da aresta é menor que o custo das faces vizinhas, $\alpha_e < min\{\alpha_f, \alpha_{f'}\}$. Nesse caso, é mais barato andar sobre a aresta do que no interior das duas regiões. Pode-se usar essa técnica para modelar uma estrada em um terreno para a navegação de robôs, por exemplo.

É possível também atribuir um custo para a aresta maior que o das faces vizinhas, $\alpha_e > max\{\alpha_f, \alpha_{f'}\}$. Se a aresta modela um obstáculo intransponível, por exemplo, pode-se atribuir a ela um custo infinito.

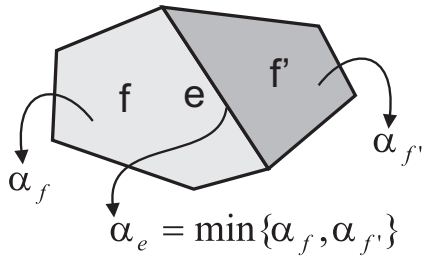
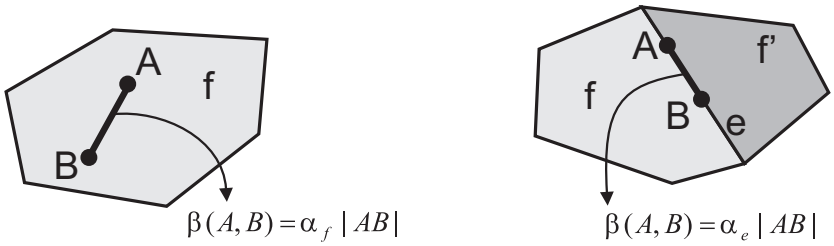


Figura 2.6: Determinando custo das arestas



(a) custo sobre a face (b) custo sobre a aresta

Figura 2.7: Cálculo do custo sobre faces e arestas

O custo total $\beta(A, B)$ de um segmento de reta que liga dois pontos A e B sobre uma face convexa é dado por

$$\beta(A, B) = \alpha_f |AB|, \tag{2.8}$$

onde $|AB|$ é a distância euclidiana entre A e B (Figura 2.7(a)). No Capítulo 3 será visto que o mapa é previamente triangulado antes de se processar o caminho de custo mínimo. Dessa forma, todas as faces são convexas, dispensando a análise de faces não convexas.

A Equação (2.8) também pode ser usada para determinar o custo total para o caso em que A e B se encontram sobre a mesma aresta e. Basta usar o custo α_e da aresta no lugar de α_f, como ilustrado na Figura 2.7(b).

Para calcular o custo de um segmento de reta que liga dois pontos quaisquer sobre a subdivisão planar, faz-se o somatório das sub-partes (faces e

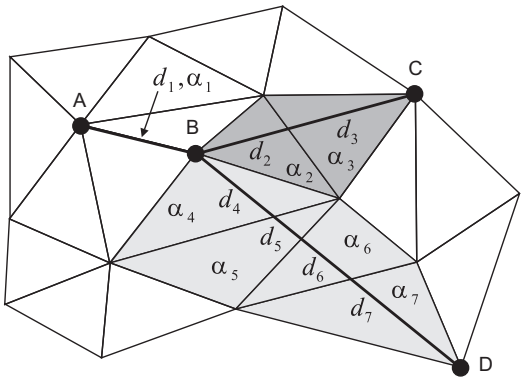


Figura 2.8: Cálculo do custo total de segmentos sobre uma subdivisão planar. $\beta(A,B) = \alpha_1 d_1$, $\beta(B,C) = \alpha_2 d_2 + \alpha_3 d_3$, $\beta(B,D) = \alpha_4 d_4 + \alpha_5 d_5 + \alpha_6 d_6 + \alpha_7 d_7$

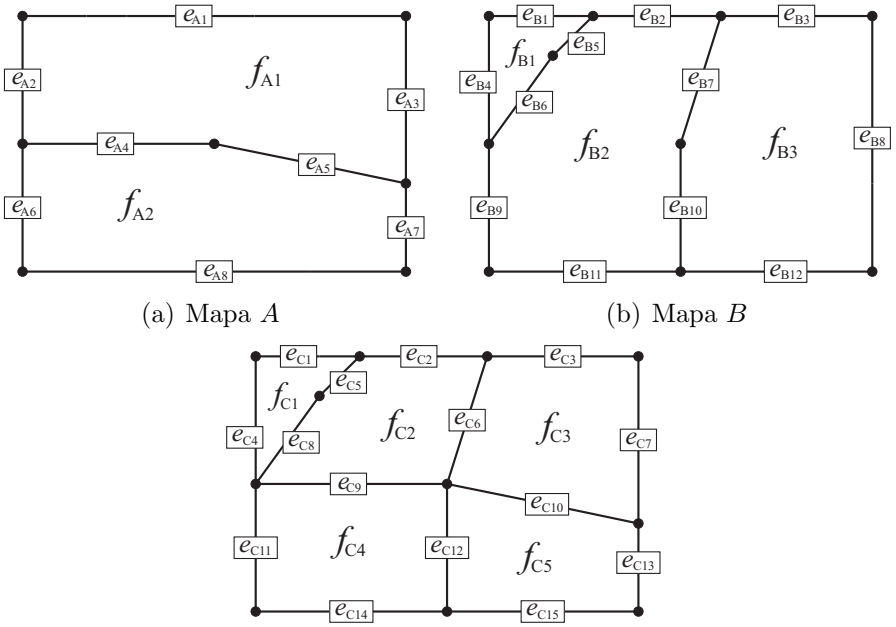
arestas). A Figura 2.8 mostra o cálculo do custo dos segmentos de A para B, de B para C e de C para D, onde A, B, C e D são vértices da subdivisão planar.Esse procedimento é necessário para a construção do grafo de pesquisa (seção 3.2) e para o pós-processamento (seção 3.4).

2.4 Composição dos mapas

A composição de mapas, ou *overlay*, é uma técnica utilizada para combinar as informações de diferentes mapas temáticos. A Figura 2.9 mostra dois mapas: A, Figura 2.9(a), e B, Figura 2.9(b), além do mapa resultante da combinação dos dois, Figura 2.9(c). Para calcular os custos combinados do mapa faz-se a soma ponderada dos custos originais de acordo com a Equação (2.3). Analogamente, o custo de uma aresta do mapa resultante é uma combinação linear dos custos de duas arestas ou de uma face e uma aresta dos mapas originais.

Para o mapa combinado da Figura 2.9(c) tem-se $\alpha_{f_{C1}} = w_A \alpha_{f_{A1}} + w_B \alpha_{f_{B1}}$ e $\alpha_{f_{C2}} = w_A \alpha_{f_{A1}} + w_B \alpha_{f_{B2}}$ como exemplo de custos resultantes para as faces $\alpha_{f_{C1}}$ e $\alpha_{f_{C2}}$, onde w_A e w_B são os custos atribuídos aos mapas A e

B, respectivamente. Da mesma forma, tem-se para as arestas os custos $\alpha_{e_{C1}} = w_A\alpha_{e_{A1}} + w_B\alpha_{e_{B1}}$, $\alpha_{e_{C2}} = w_A\alpha_{e_{A1}} + w_B\alpha_{e_{B2}}$, $\alpha_{e_{C3}} = w_A\alpha_{e_{A1}} + w_B\alpha_{e_{B3}}$ e assim por diante.



(a) Mapa A (b) Mapa B

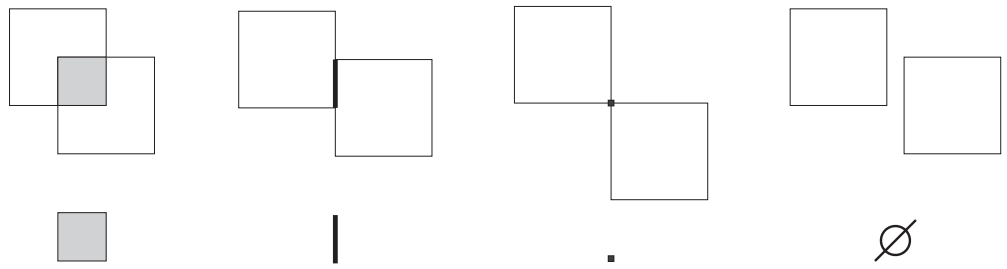
(c) Mapa Combinado

Figura 2.9: Exemplo de composição de mapas.

2.4.1 Composição dos mapas por operações booleanas de interseção entre as faces

O método implementado para realizar a composição dos mapas baseia-se na operação booleana de interseção das faces do primeiro mapa com as faces do segundo.

Como ilustrado na Figura 2.10, a operação de interseção entre duas faces convexas pode resultar em uma nova face, (Figura 2.10(a)), um segmento (Figura 2.10(b)), um único ponto (Figura 2.10(c)) ou ainda um conjunto vazio. (Figura 2.10(d)).



(a) Face (b) Segmento (c) Ponto (d) Sem interseção

Figura 2.10: Possíveis interseções entre duas faces convexas.

Para cada face do primeiro mapa verifica-se a existência de interseção com uma ou mais faces do segundo mapa. As interseções que não resultam em novas faces são consideradas como interseções inválidas e, portanto, são descartadas. As interseções que resultam em faces são combinadas para gerar o mapa final. A Figura 2.11 exemplifica esse processo para os mapas A e B da Figura 2.9.

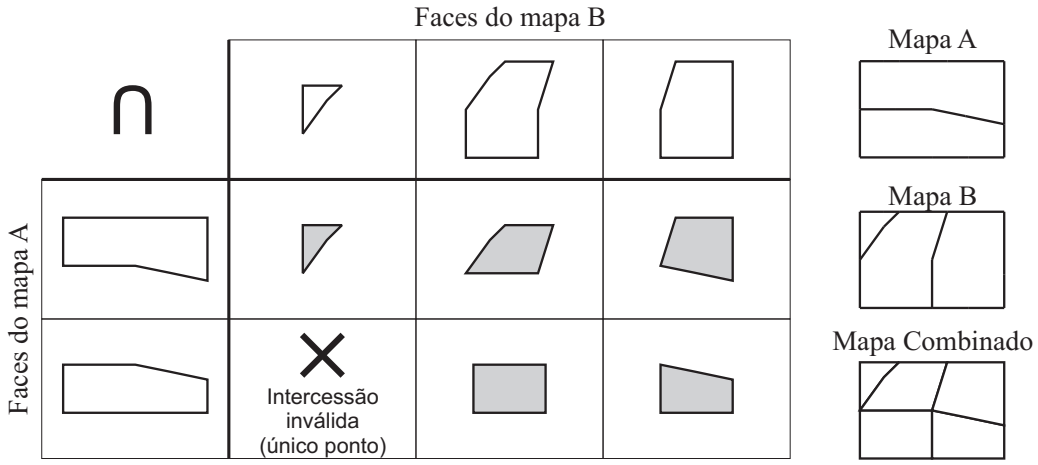


Figura 2.11: Composição de mapas a partir das interseções das faces

A CGAL oferece uma classe para representar poliedros planares (*Planar Nef Polyhedra*). Na verdade, um *Nef polyhedron* representa qualquer subdivisão do plano obtida a partir da combinação de um conjunto finito de semiplanos por meio de operações de interseção e complemento.

É possível construir um *Nef polyhedron* que representa um polígono simples como o da Figura 2.12 a partir de seus vértices ordenados no sentido anti-horário. A partir de dois *Nef's* é possível gerar um terceiro como resultado de operações booleanas como interseção, complemento, união, diferença ou diferença simétrica.

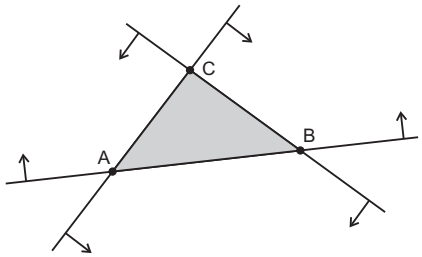


Figura 2.12: *Nef polyhedron* representando um polígono convexo

Se um mapa A possui n faces e um mapa B possui m faces, a ordem de complexidade do algoritmo implementado é $O(mn)$, pois verifica-se a interseção de cada face de A com cada face de B.

Apesar de ser um algoritmo de complexidade quadrática, optou-se por esse método por que a CGAL garante robustez para operação de interseção entre *Nef's* tratando de maneira adequada casos degenerados, como linhas sobrepostas. Além disso, a estrutura simplifica o polígono resultante eliminando vértices co-lineares. O exemplo da Figura 2.13(a) mostra dois polígonos A e B que possuem duas arestas sobrepostas. A Figura 2.13(b) mostra o resultado da interseção entre A e B sem simplificação, enquanto a Figura 2.13(c) mostra o resultado eliminando os pontos co-lineares.

Outra facilidade reside no fato de que sabe-se, ao longo do processo, quais faces deram origem às faces do mapa combinado. Portanto a determinação dos custos das faces do mapa combinado é simples.

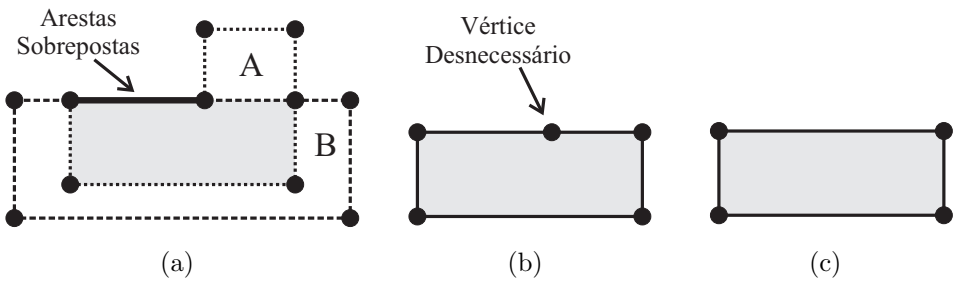


Figura 2.13: Interseção entre polígonos.

Outros métodos

Uma maneira mais eficiente de se fazer a composição de dois mapas planares é usar um algoritmo de varredura (Chazelle & Edelsbrunner 1992, Balaban 1995, de Berg et al. 2000). Inicialmente calculam-se as interseções entre todos os segmentos dos dois mapas. O próximo passo consiste em classificar as regiões onde existe interseção e combinar os custos por unidade de distância dos mapas originais (U. Finke 1995). Maiores detalhes sobre esse método são apresentados no Apêndice A.

Existem ainda outros métodos para calcular os pontos de interseção para um conjunto de segmentos baseados na partição do espaço. Um exemplo é dividir o plano em uma grade uniforme e processar um método força bruta para calcular as intercessões entre os segmentos de cada célula da grade.

Esses métodos não garantem uma boa ordem de complexidade para o pior caso. Entretanto, para mapas temáticos, onde os segmentos são relativamente curtos, espaçados e possuem poucas interseções por segmento, esses métodos podem apresentar uma eficiência satisfatória quando comparados com o método de varredura. Uma comparação entre os vários métodos para calculo de interseção de segmentos pode ser vista em (Andrews, Snoeyink, Boritz, Chan, Denham, Harrison & Zhu 1994, Andrews & Soenyink 1995).

Uma questão importante para esses métodos é determinar o espaçamento ideal para a grade. Existem técnicas estatísticas baseadas no tamanho médio dos segmentos, como descrito em (Franklin, Chandrasekhar, Kankanhalli, Seshan & Akman 1988).

Capítulo 3

Planejamento de rotas

Com o mapa temático combinado pelo método descrito no capítulo anterior, deseja-se determinar um caminho de menor custo sobre esse mapa que interliga dois pontos determinados.

Em (Mitchell 1991) é proposta uma solução teórica exata, a menos de um fator de precisão ϵ, baseada no algoritmo de Dijkstra aplicado para o mapa contínuo. Como essa solução possui complexidade computacional alta, uma aproximação do método é apresentada em (Mata & Mitchell 1997). O Apêndice B mostra com detalhes algumas propriedades do caminho mínimo em mapas contínuos e os métodos citados acima.

Uma técnica mais simples foi desenvolvida neste trabalho e é apresentada nesse capítulo: o mapa é discretizado em triângulos e, baseado nessa discretização, é construído um grafo de pesquisa que, após ser processado por um algoritmo de busca (Dijkstra tradicional), nos retorna um caminho que minimiza o custo sobre a discretização. Ou seja, o caminho encontrado é uma aproximação do caminho contínuo. Por fim, um pós-processamento é aplicado sobre o caminho resultante para diminuir o seu custo total e torná-lo mais suave.

Neste trabalho, desenvolveu-se uma nova metodologia, resumida na Figura 3.1: discretiza-se o mapa resultante da composição dos diversos mapas temáticos de interesse. Para isso, escolheu-se a Triangulação de Delaunay Restrita (TDR) (Shewchuk 1997), onde os pontos de origem e destino são forçados a serem vértices da triangulação, além dos contornos do mapa serem respeitados.

O mapa triangulado é então mapeado em um grafo de pesquisa que é usado por um algoritmo de busca como o A^*, D^* ou Dijkstra para se determinar o caminho de menor custo sobre o grafo. No software desenvolvido, preferiu-se usar o Algoritmo de Dijkstra descrito na seção 3.3 pois, apesar de possuir complexidade computacional maior, garante encontrar o caminho de custo mínimo para um dado grafo.

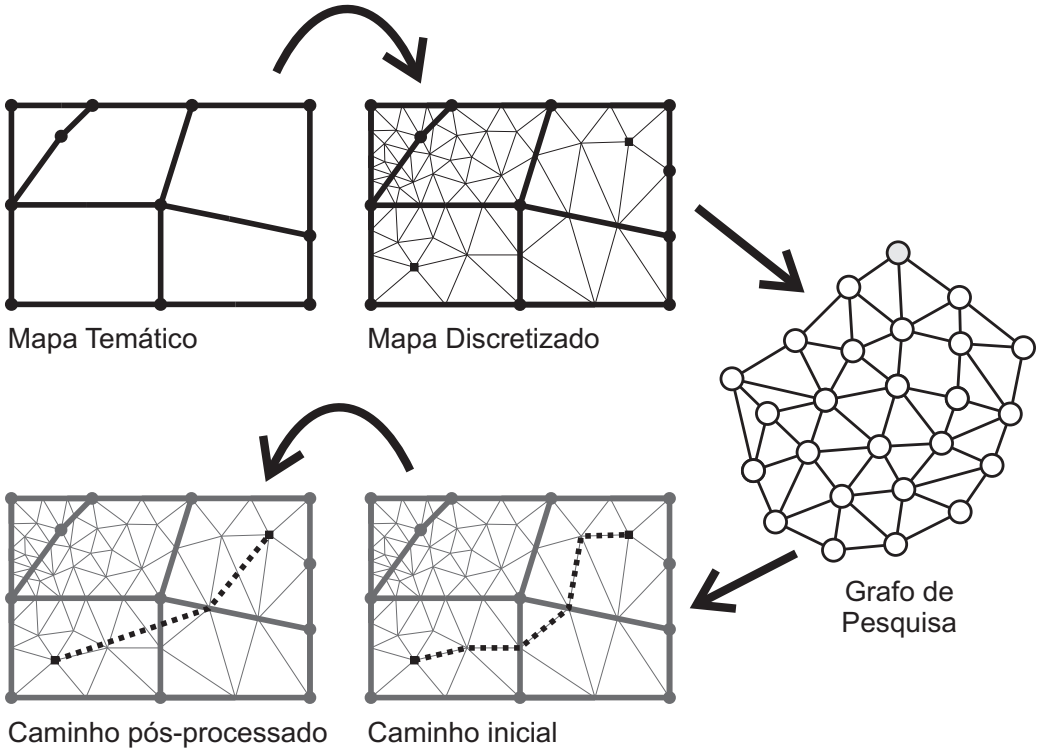


Figura 3.1: Visão geral do método implementado

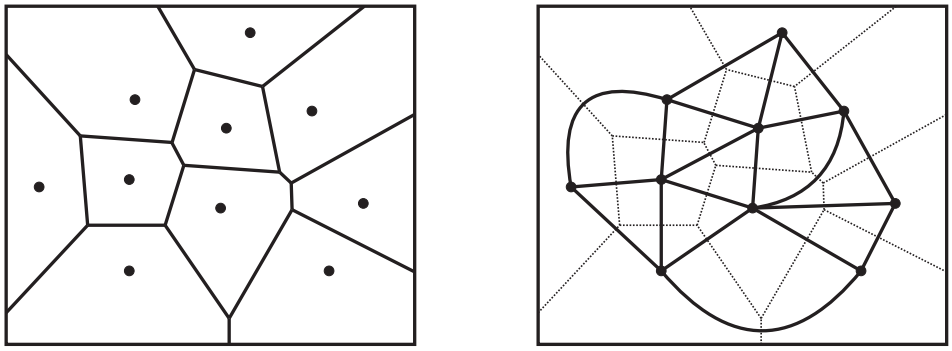
O caminho de menor custo sobre o grafo não é necessariamente o caminho de menor custo do mapa planar, já que o grafo representa uma discretização do espaço contínuo. Por isso, é possível melhorar o caminho encontrado aplicando a ele um pós-processamento como descrito na seção 3.4.

3.1 Discretização do mapa

Dado um conjunto P de n pontos, ou sítios, no plano queremos determinar para cada ponto p de P qual é a região dos pontos do plano que estão mais próximos de p do que de qualquer outro ponto em P. A região determinada por um sitio p é chamada de célula de Voronoi e determina o conjunto de pontos que estão mais próximos de p do que qualquer outro sítio. O conjunto das n células de Voronoi geradas pelos sítios formam uma subdivisão do plano chamada de Diagrama de Voronoi. A Figura 3.2(a) mostra um exemplo do diagrama de Voronoi.

O grafo dual do Diagrama de Voronoi, ilustrado no exemplo da Figura 3.2(b), é o Grafo de Delaunay. O Grafo de Delaunay é o grafo cujos vértices são os sítios de Voronoi. Estes são ligados por arcos se as células de Voronoi correspondentes forem vizinhas. Cada sítio do diagrama de Voronoi corresponde a um nó do grafo de Delaunay. Dessa forma, para cada sítio e aresta do diagrama de Voronoi existe um nó e um arco no Grafo de Delaunay.

É possível demonstrar a partir das propriedades do Diagrama de Voronoi que um Grafo de Delaunay de um conjunto de pontos é um grafo planar (de Berg et al. 2000). Isso nos garante que se os arcos do grafo de Delaunay são segmentos de reta e não existem dois arcos que se cruzam. Dessa forma, podemos usar a DCEL descrita na seção 2.2 para representar a subdivisão planar descrita pelo grafo de Delaunay.



(a) Diagrama de Voronoi (b) Grafo de Delaunay

Figura 3.2: Exemplo de Diagrama de Voronoi e Grafo de Delaunay para um conjunto de pontos

Se um dado vértice v, no Diagrama de Voronoi, é um vértice para k células de, então a face f correspondente a v no grafo de Delaunay terá k vértices, que estarão sobre uma mesma circunferência em torno de v, como ilustrado na Figura 3.3

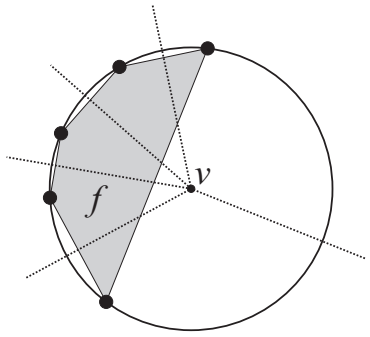
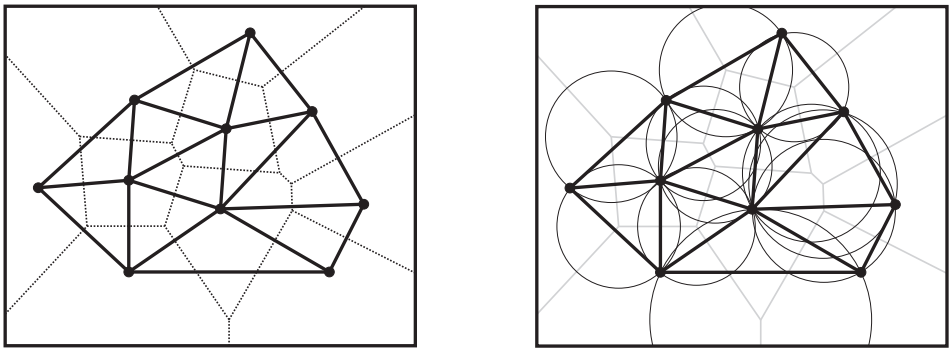


Figura 3.3: Sítios do diagrama de Voronoi sobre uma mesma círcunferência

Um conjunto de pontos está em posição geral se este conjunto não contém um subconjunto qualquer de quatro pontos sobre uma mesma circunferência. Sob a hipótese de que os sítios do Diagrama de Voronoi estão em posição geral teremos que todos os vértices do Diagrama de Voronoi terão grau três e, conseqüentemente, todas as faces limitadas do Diagrama de Delaunay serão triângulos.



(a) Triangulação de Delaunay

(b) Propriedade do circuncírculo vazio

Figura 3.4: Exemplo de Triangulação de Delaunay

Definimos a Triangulação de Delaunay de um conjunto de pontos P como qualquer triangulação obtida da inserção de arestas ao grafo de Delaunay de P. Como todas as faces do grafo de Delaunay são convexas, é fácil obter essa triangulação. A Triangulação de Delaunay será única se e somente se o conjunto de pontos, para qual desejamos obter a triangulação, estiver em posição geral. A Figura 3.4(a) mostra a triangulação de Delaunay para os pontos da Figura 3.2(a). A seguir são apresentadas algumas propriedades importantes para a Triangulação de Delaunay. As provas dessas e de outras propriedades estão em (de Berg et al. 2000).

Seja P um conjunto de pontos no plano. Três pontos p_i, p_j e p_k, pertencentes a P, são vértices de uma mesma face do grafo de Delaunay de P se e somente se a circunferência que passa por p_i, p_j e p_k não contiver nenhum ponto de P em seu interior.

Dois pontos p_i e p_j, pertencentes a P, formam uma aresta do grafo de Delaunay de P se e somente se existir um disco fechado C contendo p_i e p_j em seu contorno e não contiver nenhum outro ponto de P.

Seja T uma triangulação de P. Então T é uma triangulação de Delaunay de P se e somente se o circuncírculo de qualquer triângulo de T não contiver

um ponto de P em seu interior. A Figura 3.4(b) mostra a propriedade do circuncírculo vazio. Note que o centro dos circuncírculos são os vértices do diagrama de Voronoi.

Em uma triangulação T de P, uma aresta é ilegal se é possível aumentar localmente o menor ângulo dos triângulos, invertendo a aresta. T é uma triangulação legal se todas as arestas são legais. Isso ocorrerá se e somente se T é uma triangulação de Delaunay de P. Quando os pontos de P estão em posição geral, haverá apenas uma triangulação ótima em relação a maximização do ângulo mínimo que será a Triangulação de Delaunay. Caso os pontos de P não estejam em posição geral, então qualquer triangulação obtida do grafo de Delaunay será legal.

Seja P um conjunto de pontos no plano. Qualquer triangulação de ângulo ótimo (que maximize o ângulo mínimo) de P será uma Triangulação de Delaunay de P. E ainda, qualquer Triangulação de Delaunay de P maximiza o ângulo mínimo entre todas as triangulações de P.

Quando faz-se a triangulação de Delaunay para um mapa planar, as arestas que representam as fronteiras das regiões não são levadas em consideração. Dessa forma, as arestas originais do mapa podem não aparecer no mapa triangulado. Para resolver esse problema faz-se uso da triangulação de Delaunay restrita (Shewchuk 1997).

A triangulação de Delaunay restrita (TDR) é similar à triangulação de Delaunay mas possui a possibilidade de determinarmos segmentos que devem aparecer na triangulação. Esses segmentos são chamados de restrições.

Outro problema está relacionado com a densidade de triângulos utilizada para se discretizar o mapa quando se deseja encontrar um caminho sobre o mapa. Quanto maior a densidade de triângulos utilizada na discretização, maior é a precisão do caminho encontrado. No caso hipotético de se dividir o

plano em um número infinito de triângulos, ter-se-ia uma representação exata do espaço contínuo. Entretanto, o custo computacional para o tratamento do mapa se eleva a medida que a densidade de triângulos aumenta.

Através da TDR, obtém-se uma triangulação com tamanhos de triângulos variados. Em regiões onde a geometria é mais complicada, utiliza-se uma densidade maior de triângulos. Por outro lado, em regiões onde a geometria é mais simples, usam-se menos triângulos. Assim, tem-se uma boa representação do espaço contínuo com um número reduzido de triângulos, reduzindo o custo computacional.

O algoritmo implementado faz uso do refinamento de Delaunay descrito em (Shewchuk 1997). Essa técnica insere novos vértices na triangulação original gerando uma nova triangulação refinada. A localização desses novos vértices é escolhida tal que:

- o contorno original do mapa seja respeitado e forçado a existir na triangulação final;
- as restrições impostas pelos pontos de origem e destino sejam respeitadas;
- a qualidade da triangulação seja melhorada, evitando a existência de triângulos com um ou dois ângulos muito pequenos.

A Figura 3.5 mostra uma possível Triangulação de Delaunay Restrita com Refinamento para o mapa mostrado na Figura 2.9(c).

Uma vantagem da TDR sobre outros tipos de triangulação é que, já que se trata de uma triangulação muito utilizada, existem várias ferramentas computacionais gratuitas já desenvolvidas. Entretanto, outros tipos de triangulação poderiam ser utilizados produzindo resultados semelhantes. Nesse

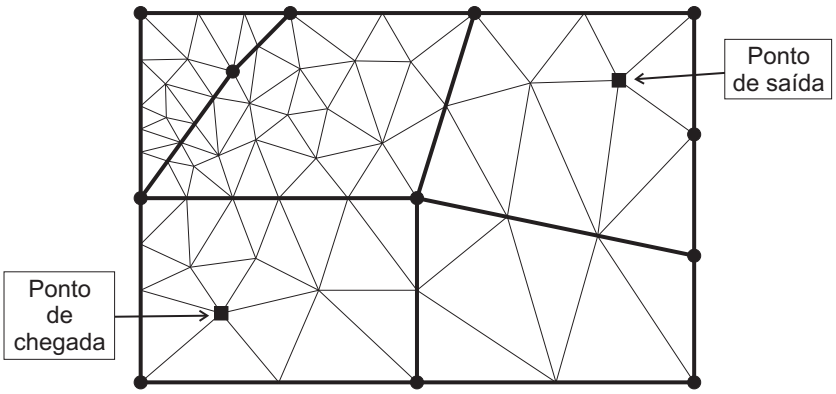


Figura 3.5: Triangulação de Delaunay Restrita com Refinamento para o mapa da Figura 2.9(c)

trabalho utilizou-se o programa *Triangle* disponível em (Shewchuk 2005). Maiores informações podem ser encontradas em (Shewchuk 1996).

3.2 Grafo de pesquisa

Em um grafo, pode-se classificar quaisquer dois vértices quanto ao *nível de vizinhança* entre eles. Esse *nível de vizinhança* é definido como o número mínimo de arestas que os separam. Caminhando-se apenas sobre as arestas, pode existir mais de um caminho ligando um vértice A até um vértice B. Entre todos os caminhos possíveis, usa-se o que possui o menor número de arestas para se fazer essa classificação.

Por exemplo, se o caminho de menor número de arestas entre A e B possui 4 arestas, então A é um vizinho de 4^o nível de B. As Figuras 3.6(a), 3.6(b) e 3.6(c) mostram os vizinhos de 1^o, 2^o e 3^o níveis do nó de interesse destacado na Figura 3.6(a).

Então, baseado na triangulação gerada sobre o mapa resultante da combinação dos mapas temáticos de interesse, constrói-se um grafo no qual os vértices da triangulação são os nós e os ramos do grafo são determinados de acordo com o nível de vizinhança desejado.

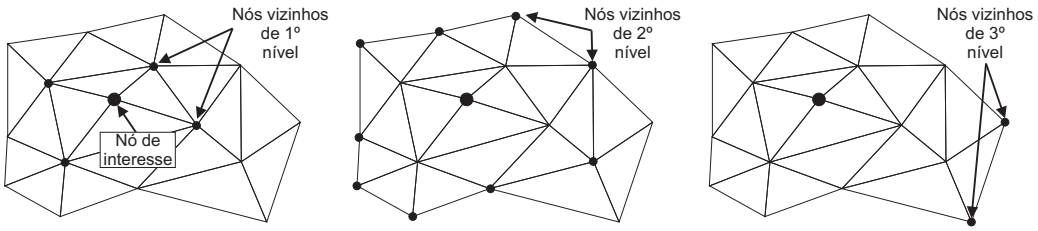


Figura 3.6: Níveis de vizinhança entre nós

Se apenas vizinhos de 1^o nível são usados, o número de ramos do grafo será igual ao número de arestas do mapa planar. Entretanto, usando o i-ésimo nível de vizinhança, ramos adicionais serão incorporadas ao grafo de tal forma que seja possível sair de um nó diretamente para um nó vizinho de 1^o, 2^o, ..., $(i-1)^o$ ou i^o nível.

As Figuras 3.7(a), 3.7(b) e 3.7(c) mostram os grafos de 1^o, 2^o e 3^o níveis de vizinhança, respectivamente, para um mesmo mapa. Note que, à medida que se aumenta o nível de vizinhança, a complexidade do grafo cresce aumentando o seu tempo de construção e o tempo de pesquisa sobre ele. Além disso, existe um limite para o nível de vizinhança sendo este no máximo $n/2$ onde n é o número de vértices do mapa planar triangulado.

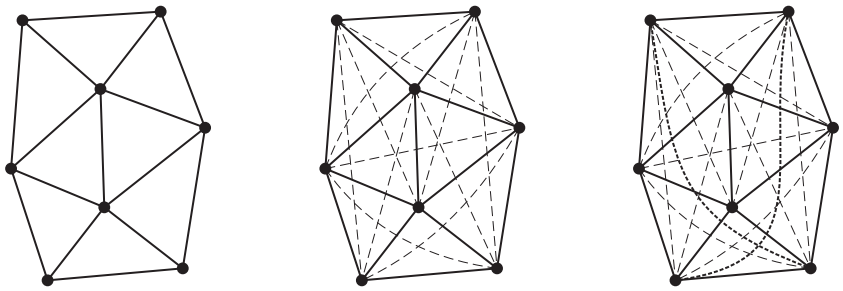


Figura 3.7: Possíveis grafos variando o nível de vizinhança dos nós

Para uma mesma triangulação, a vantagem de se construir um grafo com níveis de vizinhança mais elevados é que o caminho encontrado pode ser mais próximo do caminho ótimo. Entretanto, grafos com níveis de vizinhança elevados aumentam a complexidade computacional.

Uma solução mais simples e mais eficiente para encontrar um caminho mais suave é fazer o pós-processamento desse caminho como será visto na seção 3.4.

3.3 Algoritmo de Dijkstra

Dado um grafo onde os nós estão interligados por caminhos com um custo associado, objetiva-se determinar o caminho de menor custo entre os nós e o nó destino. Para o exemplo da Figura 3.8, o nó A em destaque é o nó destino. É importante notar também que um determinado nó não possui necessariamente caminhos diretos para todos os outros nós.

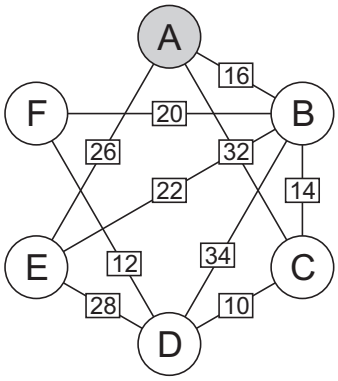


Figura 3.8: Exemplo de grafo de busca

Para determinar o caminho de menor custo pode-se utilizar o algoritmo de Dijkstra descrito no Algoritmo 3.1 (Dijkstra 1959). O algoritmo apresentado possui ordem de complexidade $O(n^2)$ onde n é o número de nós do grafo. É possível otimizar o algoritmo utilizando filas de prioridade, diminuindo a

algoritmo 3.1 Calcula o menor caminho entre os nós de um grafo ao nó destino.

Entradas: Um grafo onde cada nó possui um valor de custo associado, um apontador para o próximo nó e um estado (permanente, tentativa ou nunca visitado). Os nós são ligados por ramos e cada ramo possui um custo associado. Um dos nós é marcado como sendo o nó destino.

Saídas: O próprio grafo com os ramos orientadas de tal forma que indiquem o caminho de custo mínimo entre qualquer nó e o nó destino.

1: Inicialmente cada nó recebe o estado nunca visitado, um apontador para NULL e um custo infinito.

2: Na primeira iteração, o nó destino é marcado como permanente, o seu custo associado é 0 e o apontador é mantido como NULL.

3: O nó permanente da iteração corrente visita cada nó que pode alcançar diretamente. Se o nó visitado estiver marcado como permanente não há alteração. Caso contrário, o nó visitado passa para o estado tentativa. Se o custo da aresta que liga o nó permanente ao nó tentativa somado ao custo associado ao nó permanente for menor que o custo do nó visitado, este tem seu custo atualizado para o valor da soma e passa a apontar para o nó permanente.

4: Quando todos os nós diretamente ligados ao nó permanente da iteração corrente forem visitados então escolhe-se o nó permanente para a próxima iteração. Esse nó é o nó marcado como tentativa que possui menor custo associado. Repete-se então o passo anterior.

5: Quando todos os nós são marcados como permanente, o custo total associado a cada nó é o custo para se atingir o destino a partir desse nó. Para se encontrar a rota de menor custo, basta seguir os apontadores de nó em nó a partir de um nó inicial até o nó destino.

ordem de complexidade para $O(n \log(n))$ (Sedgewick 2001), entretanto esse procedimento não foi implementado.

A seguir é mostrada a execução do algoritmo de Dijkstra passo a passo para o problema exemplo da Figura 3.8. Na Figura 3.9(a) o nó inicial é marcado como permanente e o seu custo associado é nulo. Nessa e nas próximas Figuras (3.9(b), 3.9(c), 3.9(d), 3.9(e), 3.9(f), 3.9(g), 3.9(h) e 3.10), o custo associado de um nó é mostrado em um retângulo ao lado do nó e o nó permanente da iteração corrente é destacado com um pequeno círculo.

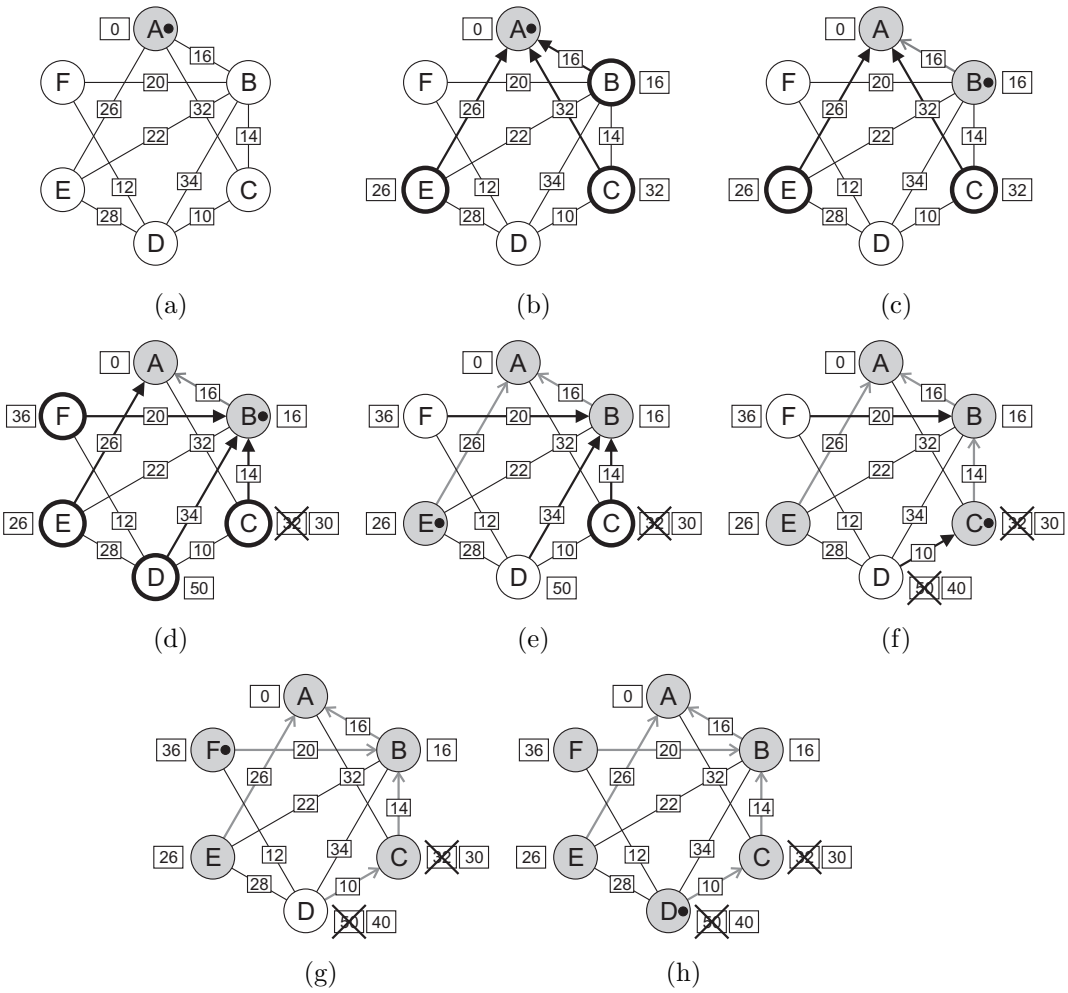


Figura 3.9: Exemplo passo a passo do algoritmo de Dijkstra

Na Figura 3.9(b), A, nó permanente da iteração corrente, altera o estado dos nós alcançáveis (B, C e E) para tentativa. Como esses nós estavam marcados anteriormente como nunca visitados, eles têm seus custos atualizados e passam a apontar para o nó A. Como o nó B é o nó tentativa de menor custo associado, ele é marcado como o nó permanente da próxima iteração (Figura 3.9(c)).

Na Figura 3.9(d) os nós não marcados como permanente alcançáveis pelo nó B são marcados como tentativa (nós C, D, E e F). Os nós recém marcados

(D e F) passam a apontar para nó B e o custo associado a eles é o custo associado ao nó B somado ao valor das arestas que os ligam a ele. O nó C possui custo menor passando pelo nó B. Por isso, o nó C tem seu custo associado e seu ponteiro atualizados.

Na Figura 3.9(e), o nó E passa a ser o novo nó permanente por ser o nó tentativa de menor custo. O nó D é o único nó alcançável não marcado como permanente. Seu custo atual (50) é menor que o custo passando pelo nó E ($28 + 26 = 54$). Nada é feito.

O nó C é então marcado como permanente. Dessa vez o custo do nó D é menor passando pelo nó C ($10 + 30 = 40 < 50$). O nó D é então atualizado passando a apontar para o nó C (Figura 3.9(f)).

Na Figura 3.9(g), o nó F passa a ser o nó permanente. O nó D é o único nó alcançável ainda não marcado como permanente mas seu caminho passando pelo nó F tem custo maior que o caminho atual. Nada é feito.

Na Figura 3.9(h), o nó D é marcado como permanente. Nesse ponto, finaliza-se a execução do algoritmo já que todos os nós estão no estado permanente.

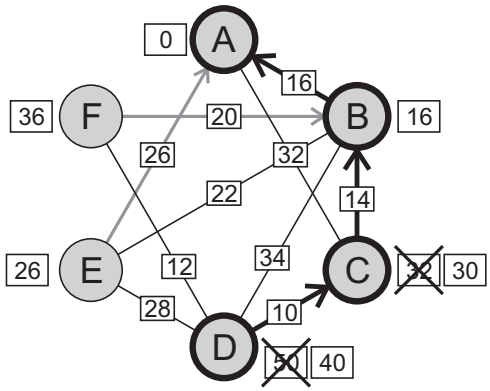


Figura 3.10: Resultado do algoritmo de Dijkstra

Depois de completo o algoritmo, o grafo está pronto para ser usado para determinar a rota de custo mínimo entre qualquer nó e o nó destino (A). Basta que, partindo de um nó, caminhemos de nó em nó de acordo com os apontadores. A Figura 3.10 mostra o caminho de menor custo do nó D ao nó A.

3.4 Pós-processamento

Como a discretização do mapa não representa com exatidão o espaço contínuo, o caminho encontrado pelo algoritmo de busca sobre o grafo geralmente não é ótimo. O objetivo de se fazer o pós-processamento do caminho gerado é tentar suavizar esse caminho e torná-lo ainda mais barato.

O mecanismo desenvolvido para esse fim é apresentado no Algoritmo 3.2. Um exemplo desse procedimento é mostrado na Figura 3.11.

É fácil ver que, exceto os vértices inicial e final, no Algoritmo 3.2 cada vértice do caminho original é testado sendo este mantido ou não na nova seqüencia. Dessa forma, o passo 3 do Algoritmo 3.2 é executado $k-2$ vezes para um caminho com uma seqüência de k vértices mostrando que esse algoritmo possui ordem de complexidade linear, no tamanho do caminho original, $O(k)$.

É possível, para mapas complicados, que o caminho pós-processado possa ser melhorado ainda mais sendo pós-processado novamente. Podemos então repetir o processo do Algoritmo 3.2 sobre sua saída sucessivas vezes até que o número de vértices da seqüência de saída seja igual ao da seqüência de entrada.

Para o pior caso, onde todos os vértices intermediários são eliminados, um a cada iteração, o algoritmo seria executado $k-2$ vezes. Para a primeira

algoritmo 3.2 Melhoria da qualidade de um caminho inicial através de eliminação de pontos.

Entradas: Um mapa planar triangulado e uma seqüência de vértices indicando o caminho inicial sobre o mapa. Deve ser possível calcular o custo de se caminhar em linha reta entre quaisquer dois vértices.

Saídas: Uma seqüência de vértices indicando o novo caminho sobre o mapa que será um subconjunto da seqüência original.

1: Inicia-se a nova seqüência com o primeiro vértice da seqüência original (vértice origem).
2: Inicializa-se 3 apontadores v_1, v_2 e v_3 com os 3 primeiros vértices da seqüência original.
3: Se o custo de se caminhar diretamente de v_1 a v_3 é menor ou igual à soma dos custos de v_1 a v_2 e de v_2 a v_3, ou seja, $\beta(v_1, v_3) \leqslant \beta(v_1, v_2) + \beta(v_2, v_3)$, então faz-se $v_2 = v_3$. Caso contrário, insere-se o vértice v_2 na nova seqüência e faz-se $v_1 = v_2$ e $v_2 = v_3$.
4: Se v_3 apontar para o último vértice da seqüência original, interrompe-se o loop. Caso contrário, faz-se v_3 apontar para o próximo vértice da seqüência original e repete-se o passo 3.
5: Finaliza-se a nova seqüência com o último vértice da seqüência original (vértice destino).

iteração,o passo 3 do Algoritmo 3.2 é executado $k-2$ vezes. Na segunda iteração, o caminho possui um ponto a menos e o passo 3 do Algoritmo 3.2 é executado $k-3$ vezes. Dessa forma, o passo 3 do Algoritmo 3.2 é executado um total de

$$\begin{aligned} & (k-2) + (k-3) + ... + [k-(k-2)] + [k-(k-1)] \\ = \; & (k-2)k - \frac{[2+(k-1)](k-2)}{2} \\ = \; & \frac{k^2 - 3k + 2}{2} \end{aligned}$$

vezes. Tem-se então uma ordem de complexidade global de $O(k^2)$. Na prática, muitos pontos são eliminados de uma só vez, sendo necessário poucas iterações do algoritmo.

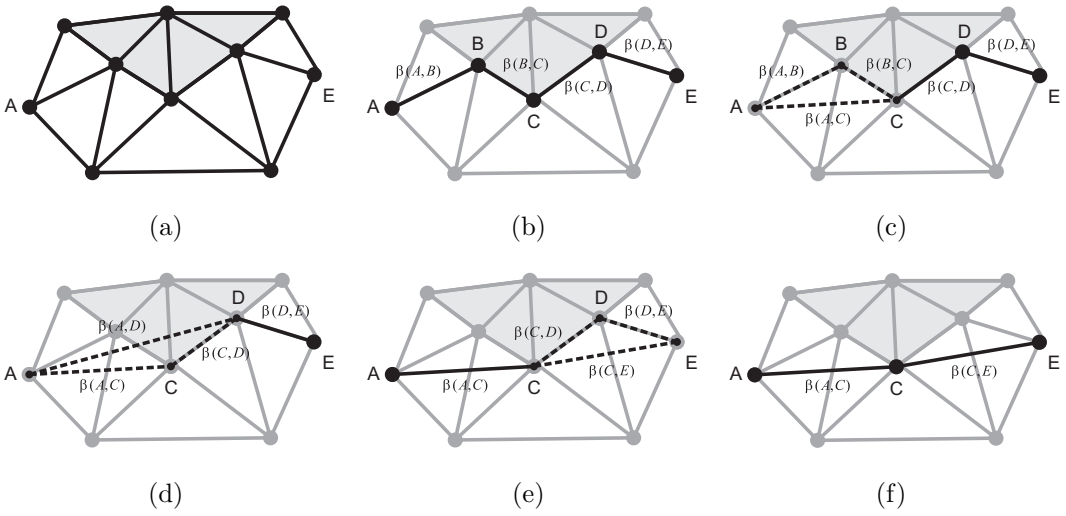


Figura 3.11: Pós-processamento

Como exemplo, seja o mapa da Figura 3.11(a), onde deseja-se encontrar o caminho de menor custo entre os pontos A e E. Nesse mapa, o custo de se caminhar sobre a região cinza α_c é três vezes maior que o da região branca α_b. O custo das arestas que separam as duas regiões é igual ao custo da menor região $\alpha_e = min\{\alpha_c, \alpha_b\} = \alpha_b$.

Usando-se um grafo de 1^o nível de vizinhança, o algoritmo de Dijkstra nos retorna o caminho dado pela seqüência de pontos A, B, C, D e E como indicado na Figura 3.11(b).

Para os três primeiros vértices do caminho (A, B e C) verifica-se que o caminho direto de A para C é menor que o caminho passando pelo vértice B (Figura 3.11(c)). Por isso, elimina-se o vértice B. Com os vértices remanescentes e o próximo (A, C e D) verifica-se novamente qual o caminho mais curto. Dessa vez, o caminho direto entre A e D passa pela região mais cara e por isso, o vértice C é mantido (Figura 3.11(d)).

Como o vértice C não foi eliminado, ele entra na próxima comparação com o próximo vértice (C, D e E). Assim, como ilustrado na Figura 3.11(e),

verifica-se que o caminho direto entre C e E é mais curto. Por fim, na Figura 3.11(f) tem-se o caminho final pós-processado.

Capítulo 4

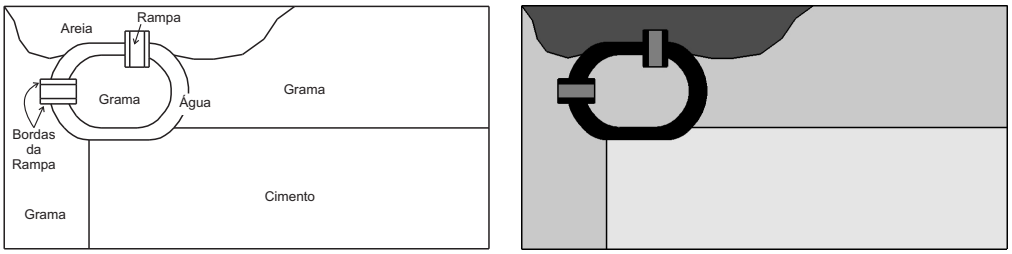
Resultados

4.1 Navegação de robôs

As simulações para o problema de navegação de robôs a serem apresentados nessa seção foram publicados em (Fonseca et al. 2005). Elas correspondem à aplicação da metodologia discutida nos capítulos 2 e 3. Para o problema proposto, foi criado um ambiente hipotético representado por três mapas temáticos mostrados nas Figuras 4.1(a), 4.4(a) e 4.5(a). Nesses mapas são descritas as propriedades do terreno, obstáculos e densidade de pessoas e intensidade de sinal para comunicação com antenas, respectivamente.

O mapa da Figura 4.1(a) mostra as propriedades do terreno. Para cada tipo de terreno foi definido um valor, também hipotético, para a dificuldade de transposição do robô. A região de água, que deve ser evitada pelo robô, recebeu um custo mais elevado seguida pelas regiões de areia, grama e cimento. Para atravessar a região de água foram colocadas duas rampas com custo intermediário entre areia e grama. As bordas das rampas são consideradas obstáculos pois podem provocar a queda do robô e, por isso, possuem custo elevado. Os valores de custo atribuídos a cada região podem ser vistos na Tabela 4.1.

A Figura 4.1(b) representa o mapa da Figura 4.1(a) com uma escala de tons de cinza. Quanto mais escura a região maior é o seu custo. Todos os mapas em tom de cinza neste capítulo representam custos dessa forma.



(a) descrição (b) representação em tons de cinza

Figura 4.1: Propriedades do terreno

Tabela 4.1: Custos para o terreno

	Custo por unidade de distância da Fig.				
Propriedade	4.1(b)/4.2	4.3(a)	4.3(b)	4.3(c)	4.3(d)
Água	∞	∞	∞	∞	∞
Borda da Rampa	∞	∞	∞	∞	∞
Areia	51	2	51	51	51
Rampa	35	2	35	35	35
Grama	11	2	2	2,25	20
Cimento	2	2	2	2	2

A Figura 4.2(a) mostra o mapa com as propriedades do terreno após a triangulação. Nesta figura são destacados os vértices de origem e destino para a trajetória do robô. Para as simulações apresentadas pelas Figuras 4.2(b), 4.2(c), 4.2(d) e 4.2(e) foram utilizados grafos com níveis de vizinhança 1, 2, 3 e 7, respectivamente (níveis de vizinhança são definidos na seção 3.2). Nota-se uma melhora gradativa do caminho quando se aumenta o nível de vizinhança do grafo. Mas essa melhora fica pouco sensível quando o nível de vizinhança é muito alto. Além disso, o tempo de processamento também aumenta, como pode ser observado nos dados da Tabela 4.2. O computador

utilizado nessas e nas outras simulações apresentadas neste capítulo foi um Pentium 4, 1,70Ghz, com 512Mb de RAM, rodando o Windows XP. A Figura 4.2(f) mostra o caminho encontrado aplicando o pós-processamento proposto na seção 3.4. O resultado encontrado é equivalente ao da Figura 4.2(e), porém com um tempo de processamento menor. Nas próximas simulações serão usados apenas grafos com nível de vizinhança 1 e pós-processamento. A Tabela 4.2 também mostra o valor total do custo do caminho para as simulações. Esses valores foram obtidos de acordo com o método mostrado mostrado na seção 2.3 levando-se em conta que os mapas possuem dimensão de $100m$ por $50m$.

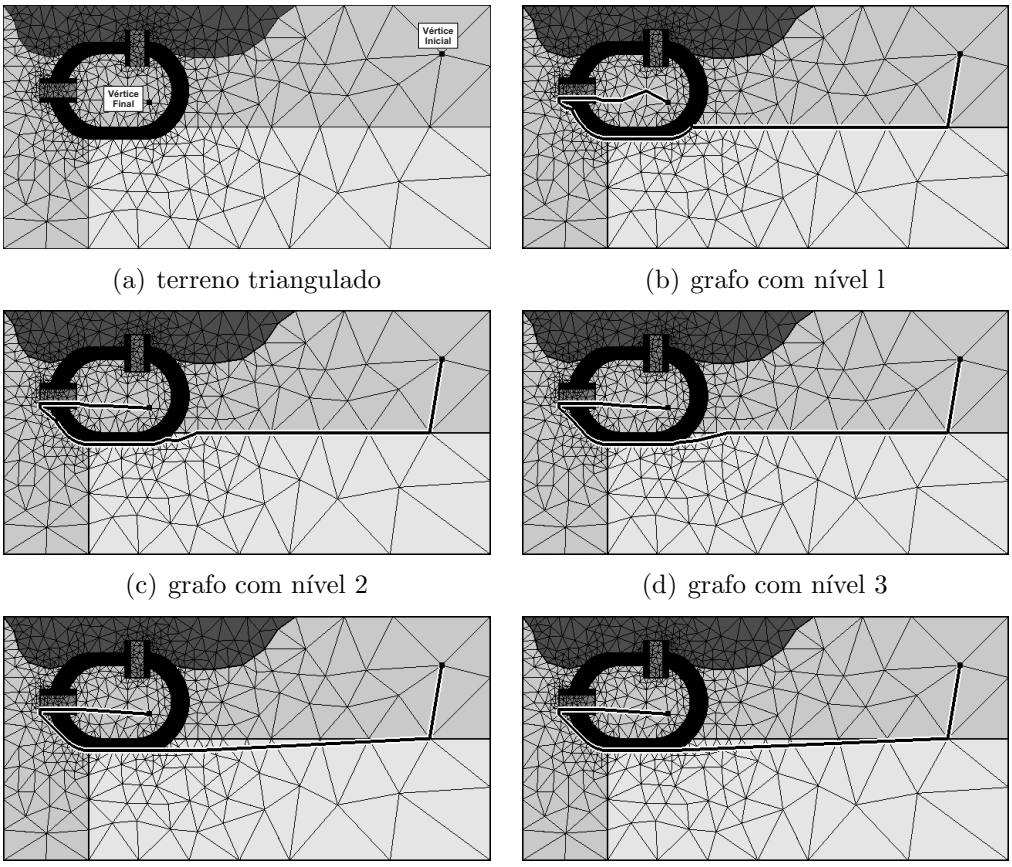
Tabela 4.2: Tempo de simulação para o terreno

Figura da simulação	4.2(b)	4.2(c)	4.2(d)	4.2(e)	4.2(f)
Tempo de busca no grafo (s)	6,43	6,85	7,76	22.47	6,43
Tempo de pós-processamento (s)	-	-	-	-	0,05
Custo do caminho	903,4	889,2	888,1	887,5	887,4

Nota-se nessa primeira simulação que os caminhos encontrados evitam a região de areia que possui um custo mais elevado. Partindo do ponto inicial, o caminho sai da região de grama para a região de cimento e permanece nessa região o máximo possível. O melhor caminho foi obtido com o pós-processamento. Nesse caso, o caminho é mais retilíneo, fazendo menos curvas dentro de uma mesma região.

Para ilustrar como a escolha dos custos é determinante, a Figura 4.3 mostra algumas simulações variando o valor dos custos das regiões. A Tabela 4.1 mostra os valores utilizados para essas simulações.

A primeira simulação (Figura 4.3(a)) utiliza pesos iguais para todas as regiões. Dessa forma, o caminho possui o menor comprimento possível. Quando aumentamos o peso da região de areia (Figura 4.3(b)), o caminho



(a) terreno triangulado (b) grafo com nível l

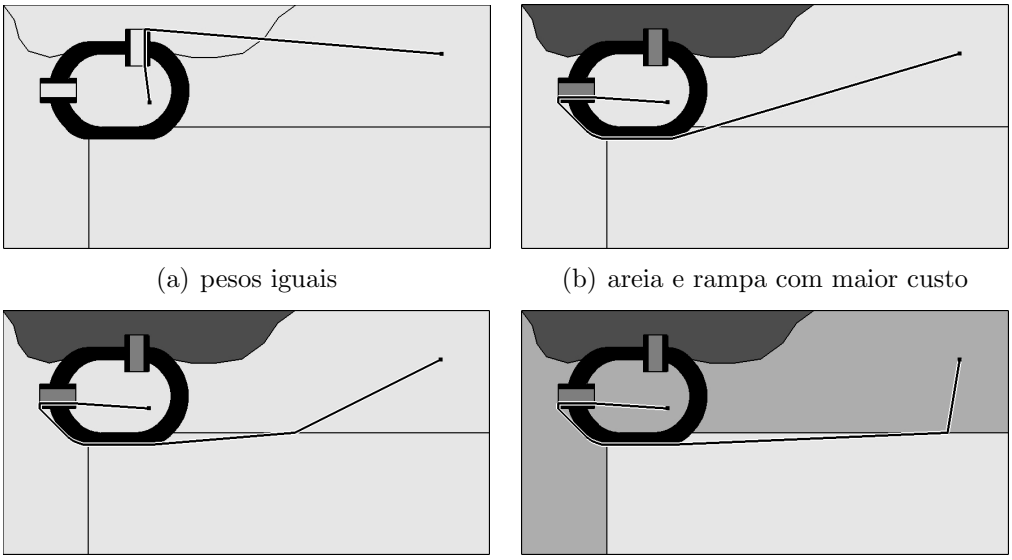
(c) grafo com nível 2 (d) grafo com nível 3

(e) grafo com nível 7 (f) grafo com nível l + pós-processamento

Figura 4.2: Simulações para as propriedades do terreno variando o nível de vizinhança

passa a evitar essa região passando pela região de cimento e grama. Como as regiões de cimento e grama ainda possui mesmo custo, o caminho não muda de direção na fronteira dessas regiões. À medida que aumentamos o custo da região de grama e, a diferença entre o custo das regiões de grama e cimento aumenta, o caminho passa para a região de cimento mais rapidamente, evitando permanecer na região de grama. Isso pode ser visto nas Figuras 4.3(c) e 4.3(d).

O mapa da Figura 4.4(a) representa um prédio (obstáculo) e a probabilidade de se encontrar pessoas pelo ambiente. Nota-se que a probabilidade



(a) pesos iguais (b) areia e rampa com maior custo

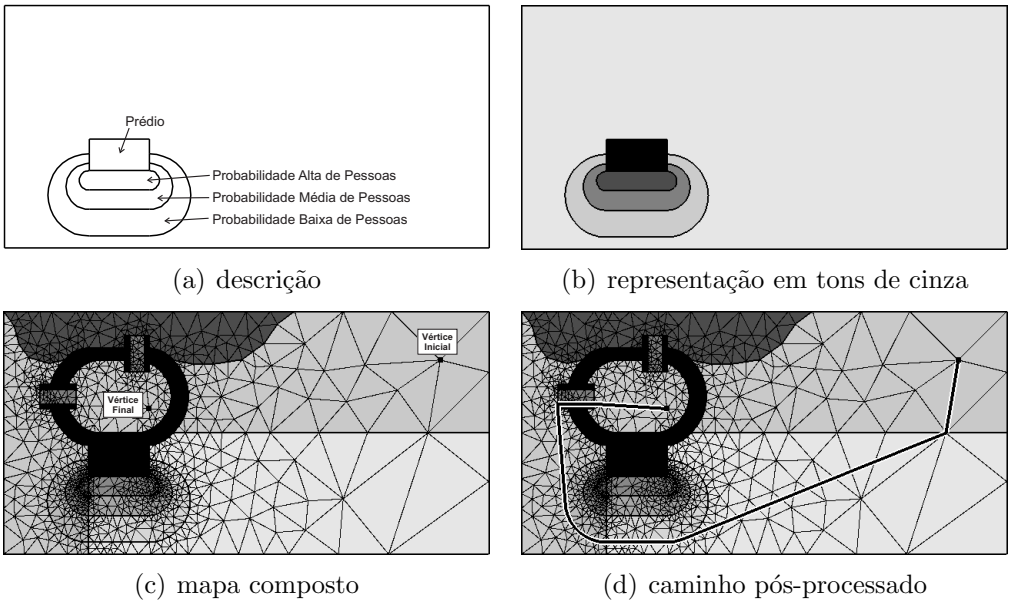
(c) pequena diferença entre grama e cimento (d) grande diferença entre grama e cimento

Figura 4.3: Simulações para as propriedades do terreno variando os pesos

de se encontrar pessoas é grande em frente ao prédio e diminui à medida que se afasta do prédio. Quanto maior é a probabilidade de existirem pessoas transitando em uma dada região, maior é a probabilidade de haver colisões entre o robô e essas pessoas, o que poderia colocar em risco a segurança das pessoas, provocar danos ao robô ou comprometer a missão do robô. Dessa forma, para regiões de maior probabilidade de se encontrar pessoas, maior é o custo de transposição atribuído.

Os custos atribuídos à dificuldade de transposição das regiões da Figura 4.4(a) são mostrados em tons de cinza pela Figura 4.4(b) e podem ser vistos na Tabela 4.3. A composição dos mapas com as propriedades do terreno e probabilidade de pessoas é mostrada já triangulada na Figura 4.4(c) e o caminho encontrado, já pós-processado, é mostrado na Figura 4.4(d).

Observando os mapas que descrevem o custo do terreno (Figura 4.1(b)) e o custo para a densidade de pessoas (Figura 4.4(b)) e comparando com o mapa combinado da Figura 4.4(c), nota-se a mudança nos tons de cinza



(a) descrição (b) representação em tons de cinza

(c) mapa composto (d) caminho pós-processado

Figura 4.4: Densidade de pessoas

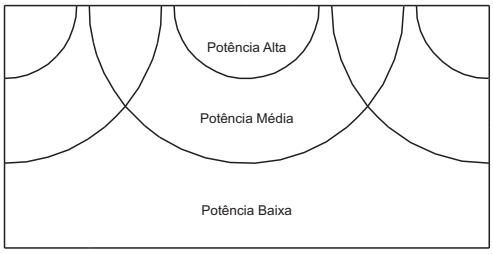
Tabela 4.3: Custos para a probabilidade de pessoas

Propriedade	Custo por unidade de distância
Prédio	∞
Probabilidade alta	30
Probabilidade média	20
Probabilidade baixa	5
Probabilidade mínima	0

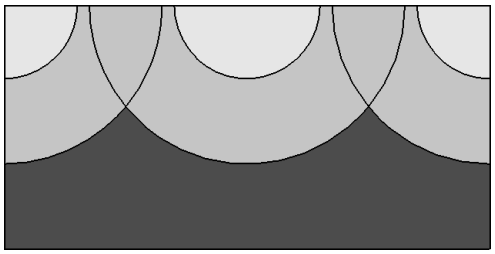
em algumas regiões indicando o resultado da combinação dos custos. Nessa composição e nas outras que serão apresentadas, o custo atribuído a cada mapa é unitário. Dessa forma, o custo combinado de uma região no mapa combinado é a soma algébrica dos custos dos mapas originais.

O mapa da Figura 4.5(a) representa a intensidade do sinal de comunicação entre o robô e três antenas fixas. Supondo uma missão onde seja desejável que o robô não perca comunicação com as antenas, então tem-se um custo elevado para as regiões distantes das antenas (potência baixa). A Figura 4.5(b) mostra o mapa com os custos em tons de cinza de acordo com a Tabela 4.4.

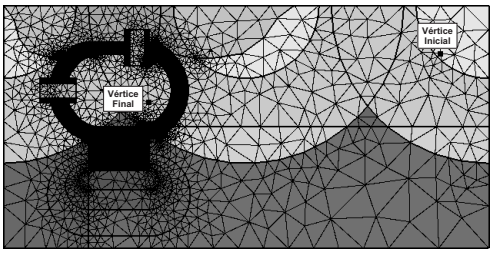
A Figura 4.5(c) mostra a composição dos mapas de terreno, probabilidade de pessoas e intensidade de sinal já triangulado. Figura 4.5(d) mostra o caminho pós-processado obtido para o mapa composto.



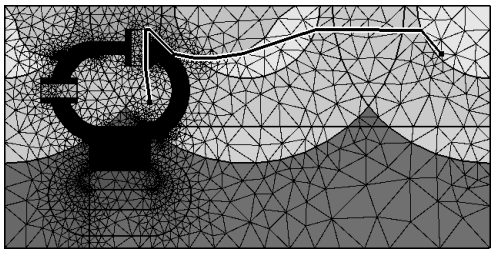
(a) descrição



(b) representação em tons de cinza



(c) mapa composto



(d) caminho pós-processado

Figura 4.5: Intensidade de sinal das antenas

Tabela 4.4: Custos para a intensidade de sinal

Propriedade	Custo por unidade de distância
Potência alta	10
Potência média	40
Potência baixa	150

Nessa última simulação, o caminho encontrado passa pela região de areia pois a região de cimento fica longe das antenas e seu custo combinado fica elevado. Nota-se que o caminho permanece o máximo possível na região de grama percorrendo a uma distância pequena sobre a areia.

A Tabela 4.5 resume alguns parâmetros de desempenho para os mapas das Figuras 4.2(f), 4.4(d) e 4.5(d). Os tempos relacionados à construção das estruturas de dados da CGAL estão embutidos nos tempos de composição

dos mapas, onde se utilizaram *Nefs*, e de busca no grafo, onde se utilizaram *Planar Maps*, como descrito nas seções 2.4.1 e 2.2.1, respectivamente. O mesmo vale para os tempos apresentados na Tabela 4.9 para o problema de linhas de transmissão.

Tabela 4.5: Comparação entre as simulações para navegação de robô

Figura da simulação	4.2(f)	4.4(d)	4.5(d)
Nº de vértices do mapa composto	112	226	318
Nº de vértices do mapa triangulado	595	1093	2848
Tempo para combinar os mapas (s)	-	26,19	69,21
Tempo de busca no grafo (s)	6,43	20,61	59,05
Tempo de pós-processamento (s)	0,05	0,06	0,04

Observa-se na Tabela 4.5 que o tempo de pós-processamento se altera muito pouco, independentemente do número de vértices do mapa. Além disso, o tempo gasto pelo pós-processamento é o que menos influencia o tempo total gasto pelo processo. Isso é uma boa indicação de que o uso do pós-processamento é válido para se obter um caminho melhor sem comprometer o tempo de execução do método.

4.2 Linhas de transmissão

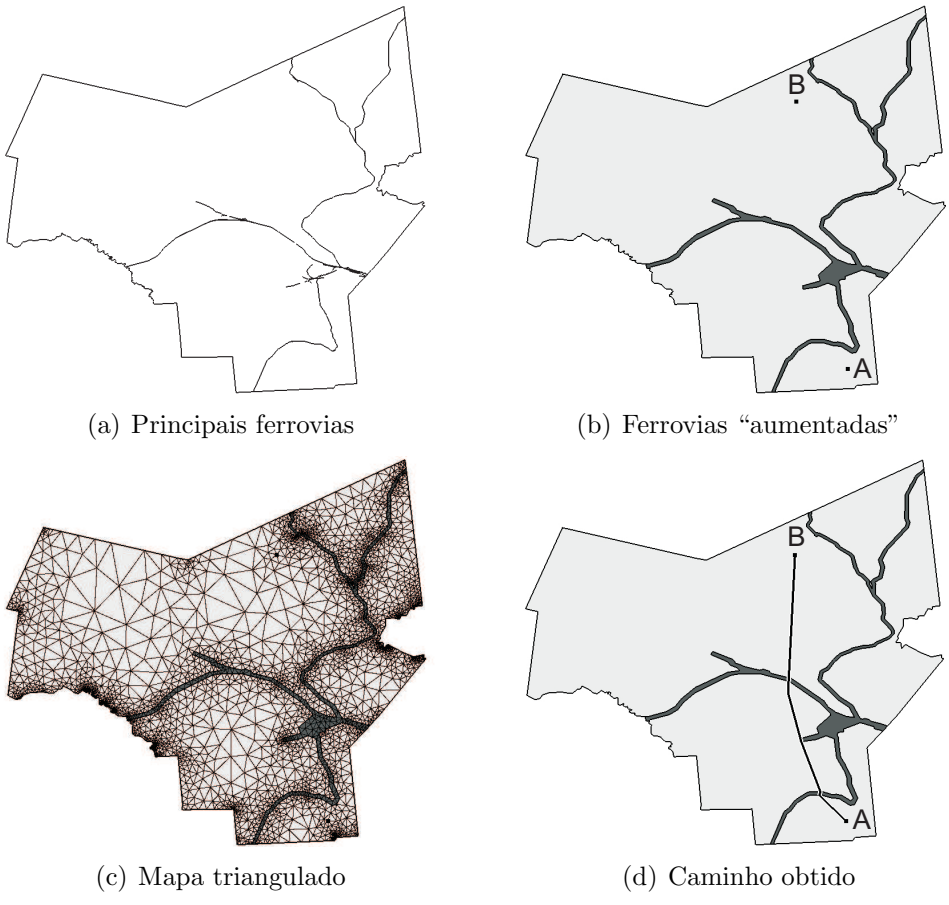
Para o caso de linhas de transmissão foi também simulado um problema hipotético onde deseja-se conectar um dado ponto A a um ponto B com base em mapas reais do estado de Nova York nos Estados Unidos. Os dados se referem mais especificamente ao condado de Oneida como ilustrado pela Figura 4.6. A Figura 4.6 e os dados dos mapas foram encontrados no site da CUGIR (*Cornell University Geospatial Data Information Repository*) (CUGIR 2006).

As coordenadas dos vértices dos mapas utilizados estão expressas em longitude e latitude.

Figura 4.6: Estado de NY

Foram utilizados três mapas temáticos com dados sobre as principais rodovias, as áreas de maior concentração urbana e os principais rios das regiões (Figuras 4.7, 4.8 e 4.10).

A Figura 4.7(a) mostra as principais ferrovias da região. Para garantir que o caminho da linha de transmissão evite ficar próximo às ferrovias, mas podendo atravessá-las, usou-se a representação da Figura 4.7(b), onde foi atribuído um custo elevado nas proximidades das ferrovias, de acordo com a Tabela 4.6. A Figura 4.7(c) mostra a triangulação utilizada e a Figura 4.7(d) mostra o caminho obtido levando em conta apenas essa restrição conectando o ponto A ao ponto B.



(a) Principais ferrovias

(b) Ferrovias "aumentadas"

(c) Mapa triangulado

(d) Caminho obtido

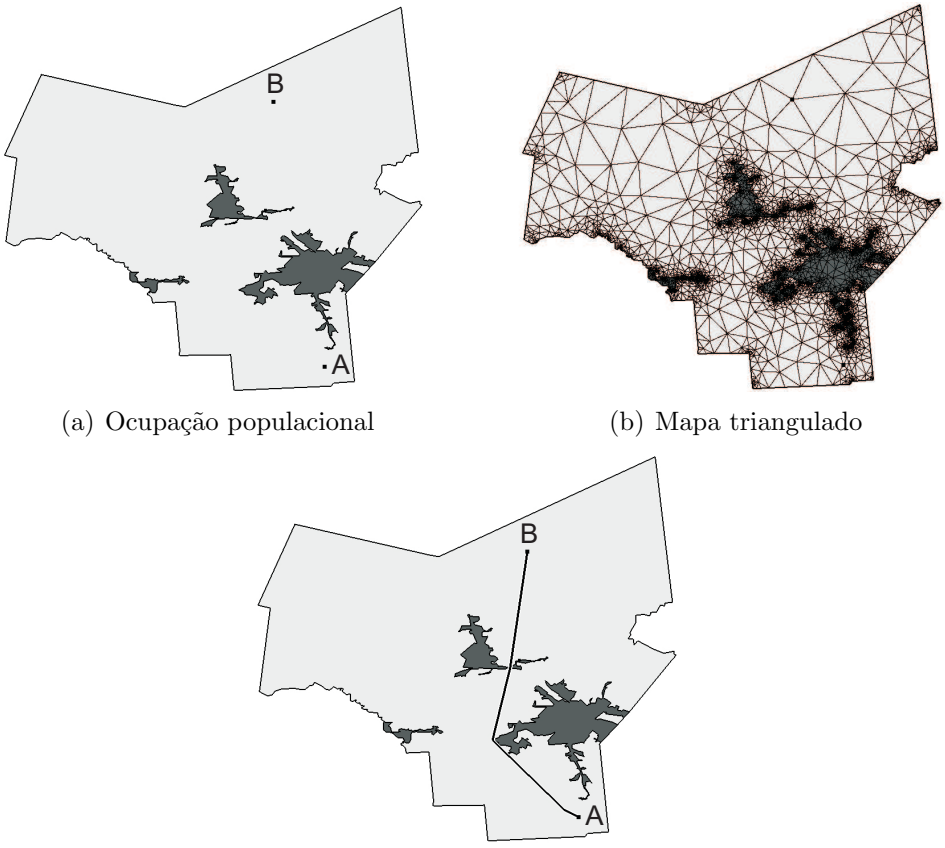
Figura 4.7: Principais ferrovias

Como esperado, o caminho encontrado evita permanecer nas regiões próximas às ferrovias e tenta atravessá-las em pontos onde a permanência do caminho sobre essas regiões é pequena.

Tabela 4.6: Custos para as ferrovias

Propriedade	Custo por unidade de distância
Ferrovia	200
Fora da ferrovia	10

A Figura 4.8(a) mostra as regiões urbanas, onde existe maior concentração populacional. Supondo que o custo de desapropriação da área urbana é elevado, atribuiu-se a essas regiões custos elevados, enquanto à região de baixa densidade populacional (rural) foi atribuído um custo baixo, como visto na Tabela 4.7. A Figura 4.8(b) mostra a triangulação utilizada e a Figura 4.8(c) mostra o caminho obtido utilizando apenas essa restrição.



(a) Ocupação populacional

(b) Mapa triangulado

(c) Caminho obtido

Figura 4.8: Regiões com ocupação populacional

Tabela 4.7: Custos para as áreas de ocupação populacional

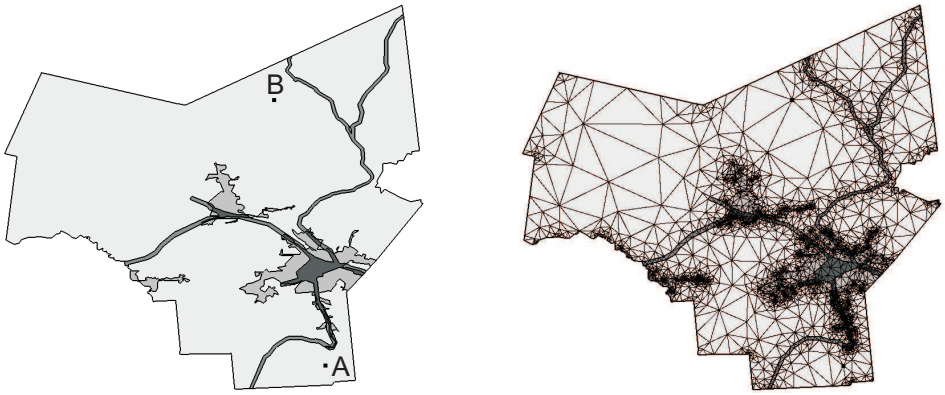
Propriedade	Custo por unidade de distância
Área urbana	100
Área rural	10

Novamente o caminho encontrado evita as regiões de custo elevado tentando contorná-las. Entretanto, o caminho atravessa um pequeno trecho da região urbana no centro do mapa. Isso indica que, com os valores de custo atribuídos às regiões, o caminho sobre uma pequena porção da região de maior custo é mais barato que contorná-lo. Poder-se-ia atribuir um valor mais elevado à essa região de tal forma que o caminho encontrado contornaria toda a região.

Combinando as restrições das ferrovias (Figura 4.7(b)) e de densidade populacional (Figura 4.8(a)), obtém-se o mapa combinado mostrado na Figura 4.9(a). A Figura 4.9(b) mostra a triangulação utilizada para a obtenção do caminho da Figura 4.9(c).

Nesse caso, nota-se a mudança da escala de tons de cinza provocada pela combinação dos custos dos mapas originais. Onde a região urbana é sobreposta por uma ferrovia encontramos os tons mais escuros indicando um custo mais elevado. O caminho encontrado evita as regiões de custo elevado tentando se desviar delas e as atravessando quando não há como contorná-las ou quando a região a ser atravessada é pequena a ponto de ser mais barato passar sobre a região de alto custo do que contorná-la.

A última restrição considerada é dada pelos principais rios da região mostrados na Figura 4.10(a). De forma semelhante ao caso das ferrovias, gerou-se um mapa (Figura 4.10(b)) com custo elevado para a região próxima aos rios e baixo custo nas demais regiões para que a linha de transmissão evite as regiões próximas aos rios. Os valores dos custos são mostrados na Tabela 4.8.



(a) Mapa composto

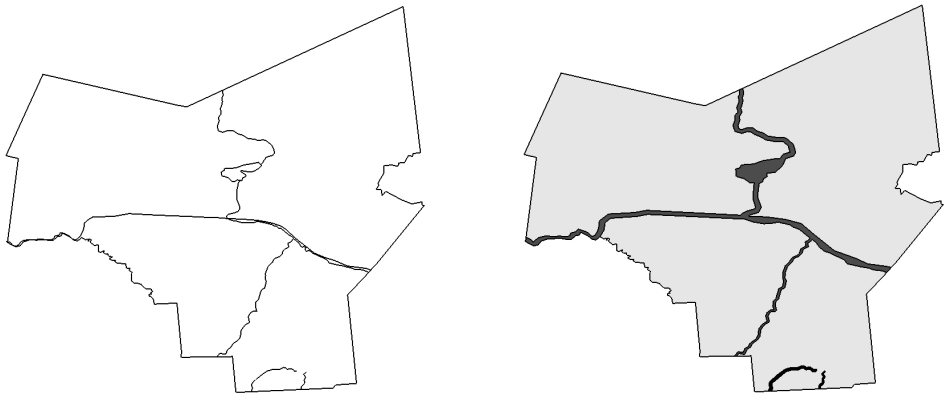
(b) Mapa triangulado

(c) Caminho obtido

Figura 4.9: Combinação dos mapas de ocupação populacional e ferrovias

A combinação das três restrições consideradas, ferrovias, regiões urbanas e rios, resulta no mapa da Figura 4.11(a). A triangulação utilizada é mostrada na Figura 4.11(b) e o caminho encontrado pela Figura 4.11(c).

A Tabela 4.9 resume alguns parâmetros de desempenho para os mapas das Figuras 4.7(d), 4.8(c), 4.9(c) e 4.11(c). Como ocorreu no exemplo de navegação de robôs (Tabela 4.5), os tempos para o pós-processamento da Tabela 4.9 são praticamente constantes. Além disso, o pós-processamento é a etapa mais barata e, mesmo que o caminho pós-processado apresente pouca ou nenhuma redução no seu custo total, seu tempo de execução não influencia significativamente o tempo total de busca pelo caminho.



(a) Principais rios (b) Rios "aumentados"

Figura 4.10: Principais rios

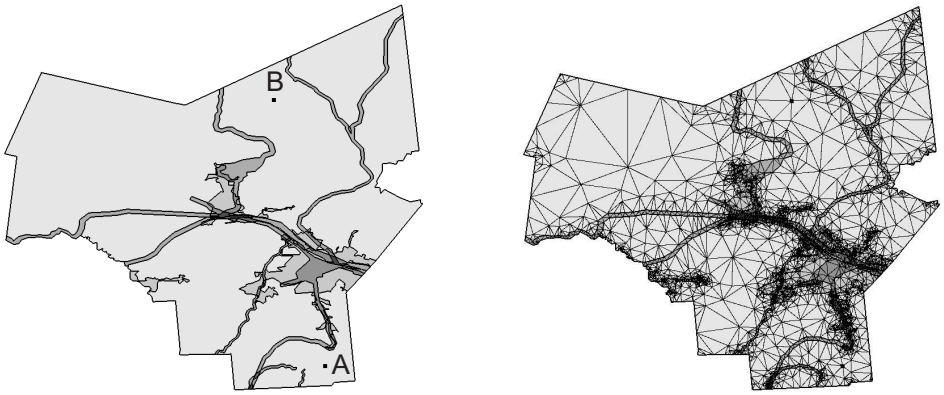
Tabela 4.8: Custos para os rios

Propriedade	Custo por unidade de distância
Rio	200
Fora do rio	10

Em todas as simulações apresentadas, tanto para o caso de navegação de robôs quanto para o caso de linhas de transmissão, usou-se o *Triangle* (Shewchuk 2005) para realizar as triangulações como descrito na seção 3.1. Nota-se que em todos os mapas utilizou-se uma densidade de triângulos maior nas regiões onde a geometria dos mapas é mais complexa e uma densidade menor nas regiões de baixa complexidade. Dessa forma, foi possível diminuir o número total de vértices dos grafos de pesquisa comparado com o uso de densidades uniformes.

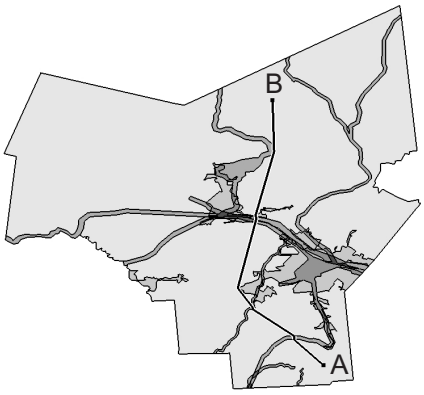
Tabela 4.9: Comparação entre as simulações para linha de transmissão

Figura da simulação	4.7(d)	4.8(c)	4.9(c)	4.11(c)
No de vértices do mapa composto	368	874	1138	1431
No de vértices do mapa triangulado	5312	6680	5446	1539
Tempo para combinar os mapas (s)	-	-	92,01	390,01
Tempo de busca no grafo (s)	152,65	197,83	148,97	219,70
Tempo de pós-processamento (s)	0,03	0,03	0,03	0,05



(a) Mapa composto

(b) Mapa triangulado



(c) Caminho obtido

Figura 4.11: Combinação dos mapas de ocupação populacional, ferrovias e rios

Capítulo 5

Conclusões

Foi desenvolvido um sistema computacional capaz de efetuar a composição de mapas temáticos e encontrar uma aproximação do caminho de menor custo sobre o mapa combinado. A principal vantagem da técnica implementada sobre outros métodos apresentados é a capacidade de encontrar bons resultados com custo computacional mais baixo que o de técnicas mais simples baseadas em grades e sem a complexidade de implementação dos métodos mais precisos.

A técnica de composição de mapas permitiu a análise de vários tipos de restrições simultaneamente, tanto para o caso de planejamentos de rotas de robôs móveis (propriedades do solo, obstáculos, densidade de obstáculos, intensidade de sinal para comunicação com antenas) quanto para o traçado de linhas de transmissão (densidade populacional, rios, ferrovias).

A triangulação utilizada mostrou ser uma ferramenta eficiente na representação de mapas complexos. Como são utilizados elementos não uniformes, áreas simples de menor complexidade são representadas com uma densidade baixa de triângulos, diminuindo a complexidade do grafo de pesquisa.

A técnica de pós-processamento, baseada na eliminação de vértices desnecessários, mostrou-se capaz de melhorar significativamente o resultado encontrado pelo algoritmo de busca sobre grafos mais simples. Dessa forma, foi possível encontrar resultados satisfatórios sem a necessidade da utilização de um grafo completo.

O grafo utilizado para a representação do mapa discretizado é não orientado e, além disso, as regiões têm custo constante independente da direção em que se caminha sobre elas. Isso implica em uma limitação do método proposto, uma vez que não é possível definir uma dificuldade maior para subir uma rampa do que para descê-la, por exemplo. Outra dificuldade é que mapas de regiões extensas normalmente possuem muitos detalhes e podem exigir uma representação complexa com um número elevado de vértices e segmentos. Nesse caso, o custo computacional se torna elevado exigindo talvez o uso de técnicas para simplificação dos mapas (Goodman & O'Rourke 2004) utilizando um número reduzido de segmentos em curvas, por exemplo.

5.1 Trabalhos futuros

O custo computacional do sistema pode ser melhorado, substituindo o método de composição de mapas implementado pelo apresentado no apêndice A. Entretanto, apesar de ser mais eficiente que o método utilizado, a implementação da composição através da interseção de segmentos é complicada e exige o tratamento de casos degenerados, o que torna difícil obter uma implementação robusta. A equipe de desenvolvimento da CGAL está implementando o algoritmo de composição de mapas planares para uma versão futura da biblioteca.

Utilizou-se o algoritmo de Dijkstra para busca sobre o grafo para encontrar o caminho de custo mínimo sobre o grafo de pesquisa gerado a partir do mapa triangulado. É possível melhorar a versão implementada, utilizando fila de prioridades reduzindo a ordem de complexidade de $O(n^2)$ para $O(n \log n)$ (Sedgewick 2001). Pode-se ainda utilizar o algoritmo A^* para ambientes muito grandes e complexos, porém não garantindo para todos os casos a obtenção do caminho ótimo sobre o grafo.

Não foi desenvolvida uma técnica para se determinar o valor dos custos das regiões dos mapas temáticos. Nos exemplos vistos os valores dos custos são empíricos. No caso de navegação de robôs, uma proposta de trabalho futuro é o desenvolvimento de uma metodologia baseada no consumo de bateria, por exemplo, ou em algum outro critério que se deseja minimizar. Se o interesse é minimizar o tempo total de percurso, pode-se atribuir um custo proporcional ao tempo de transposição por unidade de distância em um determinado terreno. No caso de linhas de transmissão é necessário um estudo mais detalhado sobre impacto ambiental, valor de desapropriação, densidade populacional, entre outros. Esse tipo de estudo exigiria uma parceria com uma concessionária de energia que tenha interesse em levar esse projeto adiante. Além disso, não foi considerado no cálculo do custo a distancia máxima entre torres, sua altura e a topografia do terreno, entre outras várias restrições elétricas e mecânicas (de Figueiredo & Gonzaga 2003).

Produção bibliográfica durante o período do mestrado

A seguir estão listadas as publicações relativas à esse trabalho e a outros trabalhos realizados durante o período de mestrado do aluno:

- Fonseca, A. R.; Pimenta, L. C. A.; Mesquita, R. C.; Saldanha, R. R.; Pereira, G. A. S.. Path Planning for Mobile Robots Operating in Outdoor Environments using Map Overlay and Triangular Decomposition. In: 18th International Congress of Mechanical Engineering, 2005, Ouro Preto, Brazil. *Proceedings of the 18th International Congress of Mechanical Engineering*, 2005.

- Pimenta, L. C. A.; Fonseca, A. R.; Pereira, G. A. S.; Mesquita, R. C.; Silva, E. J.; Caminhas, W. M.; Campos, M. F. M.. On Computing Complex Navigation Functions. In: IEEE International Conference on Robotics and Automation, 2005, Barcelona, Spain. *Proceedings of IEEE International Conference on Robotics and Automation*, 2005. p. 3463-3468.

- Pimenta, L. C. A.; Fonseca, A. R.; Pereira, G. A. S.; Mesquita, R. C.; Silva, E. J.; Caminhas, W. M.; Campos, M. F. M.. Robot navigation based on electrostatic field computation. *IEEE Transactions on Magnetics*, (accepted), v. 42, n. 4, 2006.

- Parreira, G. F.; Silva, E. J.; Fonseca, A. R.; Mesquita, R. C.. The Element-free Galerkin Method in 3-Dimensiona Electromagnetic Problems. *IEEE Transactions on Magnetics*, (accepted), v. 42, n. 4, 2006.

- Parreira, G. F.; Fonseca, A. R.; Lisboa, A. C.; Silva, E. J.; Mesquita, R. C.. Efficient Algorithms and Data Structures for Element-free Galerkin Method. *IEEE Transactions on Magnetics*, (accepted), v. 42, n. 4, 2006.

Referências Bibliográficas

ABNT (2003). Norma NBR 5422 - Projeto de linhas aéreas de transmissão de energia elétrica, *Technical report*, Associação Brasileira de Normas Técnicas. Revisão (Draft 04).

Aleksandrov, L., Maheshwari, A. & Sack, J.-R. (2005). Determining approximate shortest paths on weighted polyhedral surfaces, *Journal of the ACM (JACM)* **52**(1): 25–53.

Andrews, D. S., Snoeyink, J., Boritz, J., Chan, T., Denham, G., Harrison, J. & Zhu, C. (1994). Further comparison of algorithms for geometric intersection problems, *Proceedings of 6th International Symposium on Spatial Data Handling*, Edinburgh, UK, pp. 709–724.

Andrews, D. S. & Soenyink, J. (1995). Geometry in GIS is not combinatorial: segment intersection for polygon overlay, *Proceedings of the eleventh annual symposium on Computational geometry*, Vancouver, British Columbia, Canada, pp. 424–425.

Balaban, I. J. (1995). An optimal algorithm for finding segments intersections, *Proceedings of the eleventh annual symposium on Computational geometry*, Vancouver, British Columbia, Canada, pp. 211–219.

Berry, J. K. (2004). Optimal path analysis and corridor routing: Infusing stakeholder perspective in calibration and weighting of model criteria, *GeoTech Conference on Geographic Information Systems*, Toronto, Ontario, Canada, pp. 28–31.

Berry, J. K. (2005). Procedures and applications in gis modeling, online book, in preparation, `http://www.innovativegis.com/basis/MapAnalysis/Default.html`. Acesso em janeiro, 2005.

Bondy, J. A. & Murty, U. S. R. (1976). *Graph Theory with Applications*, MacMillan.

Boulaxis, N. & Papadopoulos, M. (2002). Optimal feeder routing in distribution system planning using dynamic programming technique and GIS facilities, *IEEE Transactions on Power Delivery* **17**(1): 242–247.

CGAL (2005). The CGAL home page, `http://www.cgal.org`. Acesso em fevereiro, 2005.

Chazelle, B. & Edelsbrunner, H. (1992). An optimal algorithm for intersecting line segments in the plane, *Journal of the ACM (JACM)* **39**(1): 1–54.

CUGIR (2006). Cornell University Geospatial Data Information Repository home page, `http://cugir.mannlib.cornell.edu/`. Acesso em janeiro, 2006.

de Berg, M., van Kreveld, M., Overmars, M. & Schwarzkopf, O. (2000). *Computational Geometry Algorithms and Applications*, second edn, Springer-Verlag.

de Figueiredo, J. N. & Gonzaga, C. C. (2003). Aplicação de métodos de busca em grafos com nós parcialmente ordenados à locação de torres de transmissão, *Pesquisa Operacional* **23**(1): 209–220.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs, *Numerische Mathematik* **1**: 269–271.

Fonseca, A. R., Pimenta, L. C. A., Mesquita, R. C., Saldanha, R. R. & Pereira, G. A. S. (2005). Path planning for mobile robots operating in outdoor environments using map overlay and triangular decomposition, *Proceedings of the 18th International Congress of Mechanical Engineering*, Ouro Preto, Brazil.

Franklin, W. R., Chandrasekhar, N., Kankanhalli, M., Seshan, M. & Akman, V. (1988). Efficiency of uniform grids for intersection detection on serial and parallel machines, *Proceedings of Computer Graphics International*, Geneva, Switzerland, pp. 51–62.

Gangnet, M., Hervé, J.-C., Pudet, T. & van Thong, J.-M. (1989). Incremental computation of planar maps, *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, pp. 345–354.

Goodman, J. E. & O'Rourke, J. (2004). *Handbook of Discrete and Computational Geometry*, 2nd edn, Chapman & Hall/CRC.

Guibas, L. J., Ramshaw, L. & Stolfi, J. (1983). A kinetic framework for computational geometry, *Proceedings of 24th IEEE Symposium on the Foundations of Computer Science*, Los Alamitos, California, pp. 100–111.

Guivant, J., Nebot, E., Nieto, J. & Masson, F. (2004). Navigation and mapping in large unstructured environments, *The International Journal of Robotics Research* **23**(4-5): 449–472.

Guo, Y., Parker, L. E., Jung, D. & Dong, Z. (2003). Performance-based rough terrain navigation for nonholonomic mobile robots, *Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society*, Roanoke, Virginia, USA, pp. 2811–2816.

Hart, P. E., Nilsson, N. J. & Raphael:, B. (1968). A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics* **4**: 100–107.

Kobilarov, M. B. & Sukhatme, G. S. (2005). Near time-optimal constrained trajectory planning on outdoor terrain, *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 1833–1840.

Latombe, J.-C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA.

Mata, C. S. & Mitchell, J. S. B. (1997). A new algorithm for computing shortest paths in weighted planar subdivisions, *Proceedings of the thirteenth annual symposium on Computational geometry*, ACM Press, Nice, France, pp. 264–273.

Mitchell, J. S. B. (1991). The weighted region problem: finding shortest paths through a weighted planar subdivision, *Journal of the Association for Computing Machinery* **38**(1): 18–73.

Pimenta, L. C. A. (2005). *Navegação de robôs móveis baseada na equação de Laplace: Uma nova abordagem utilizando elementos finitos*, Master's thesis, Universidade Federal de Minas Gerais.

Sedgewick, R. (2001). *Algorithms in C++ Part 5: Graph Algorithms*, 3rd edn, Addison-Wesley Professional.

Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, *in* M. C. Lin & D. Manocha (eds), *Applied Computational Geometry: Towards Geometric Engineering*, Vol. 1148 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

Shewchuk, J. R. (1997). *Delaunay Refinement Mesh Generation*, PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. Available as Technical Report CMU-CS-97-137.

Shewchuk, J. R. (2005). The Triangle - A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator home page, `http://www.cs.cmu.edu/~quake/triangle.html`. visited in August, 2005.

Stentz, A. (1995). The focussed D* algorithm for real-time replanning, *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 605 – 611.

U. Finke, K. H. H. (1995). Overlaying simply connected planar subdivisions in linear time, *Proceedings of the eleventh annual symposium on Computational geometry*, Vancouver, British Columbia, Canada, pp. 119–126.

Vega, M. & Sarmiento, H. G. (1996). Image processing application maps optimal transmission routes, *IEEE Computer Applications in Power* **9**(2): 47–51.

West, N. A., Dwolatzky, B. & Meyer, A. S. (1997). Terrain based routing of distribution cables, *IEEE Computer Applications in Power* **10**(1): 42–46.

Wilson, R. J. (1996). *Introduction to Graph Theory*, 4th edn, Prentice Hall.

Yahja, A., Singh, S. & Stentz, A. (2000). An efficient on-line path planner for outdoor mobile robots, *Robotics and Autonomous Systems* **32**: 129–143.

Apêndice A

Composição dos mapas pelo cálculo das interseções entre segmentos

A.1 Cálculo das interseções

Para calcular as interseções entre os segmentos pode-se pensar inicialmente em um método força bruta, no qual se testaria se um segmento possui interseção com todos os outros segmentos. Claramente, esse procedimento possui uma ordem de complexidade $O(n^2)$ onde n é o número total de segmentos. Como na prática o número de interseções entre os segmentos de dois mapas temáticos é muito menor que a combinação dos segmentos dois a dois, esse algoritmo se torna ineficiente pois realiza um número de testes muito maior que o número efetivo de intercessões.

A idéia do algoritmo de varredura consiste em deslocar uma linha l horizontal pelo conjunto de segmentos, buscando identificar os segmentos que são cortados em um dado momento pela linha, verificando se existem interseções

entre eles. Para diminuir mais ainda o número de cálculos de interseções, os segmentos são ordenados segundo a coordenada dos pontos em que cortam a linha. O algoritmo de varredura procura, assim, identificar as interseções de novos segmentos na linha, saída de segmentos e inversões na ordem em que esta linha encontra dois segmentos quaisquer. Dessa forma, testes desnecessários são evitados e o tempo de execução passa a ser sensível não só ao número total de segmentos considerados mas também ao número total de interseções.

Para implementar esta idéia, é necessário definir uma relação de comparação entre os segmentos da seguinte forma: Considerem-se dois segmentos s_1 e s_2 no plano, sendo que s_1 não intercepta s_2. Diz-se que s_1 é comparável a s_2 se, para alguma ordenada y, existe uma linha horizontal que intercepta tanto s_1 quanto s_2. Seja então $S = \{s_1, s_2, ..., s_n\}$ um conjunto de segmentos para os quais deseja-se calcular todas as intercessões. Se as projeções de dois segmentos no eixo y não se sobrepõem, esses segmentos não se interceptam e, dessa forma, não se faz necessário testar se existe interseção entre eles (Figura A.1).

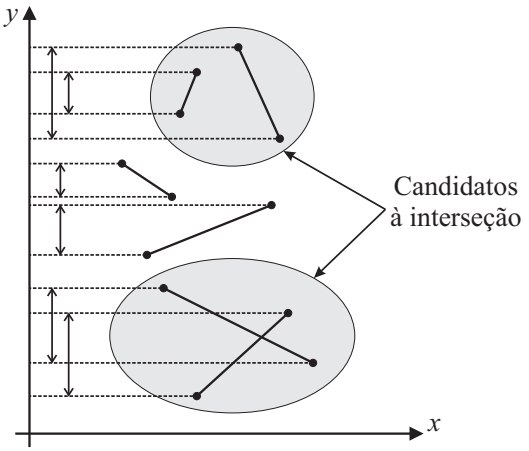


Figura A.1: Segmentos candidatos ao teste de interseção

O ***estado da linha de varredura*** é o conjunto de segmentos que a interceptam. O estado da linha de varredura muda à medida que ela se desloca para baixo e encontra um ponto extremo de um segmento ou um ponto de interseção entre segmentos. Estes pontos são chamados de ***eventos de ponto***.

No momento que a linha de varredura alcança um evento de interseção, atualiza-se o seu estado e realizam-se alguns testes de interseção como ilustrado na Figura A.2.

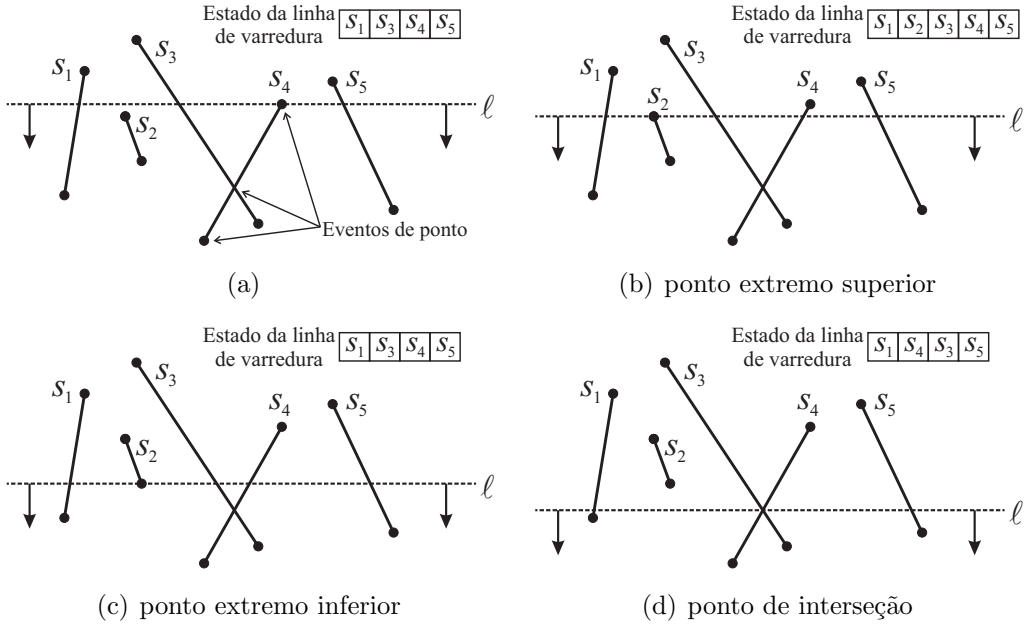


Figura A.2: Mudanças de estado para a linha de varredura.

Se um ponto extremo superior de um segmento é encontrado significa que esse segmento passa a ser interceptado pela linha de varredura e passa a fazer parte do estado da linha de varredura (Figura A.2(b)). Nesse caso, deve-se testar se existe interseção entre esse novo segmento e alguns segmentos que já estão listados no estado. Se, no entanto, a linha de varredura encontra um evento de ponto que é o ponto extremo inferior de um segmento, isso

significa que o segmento deve deixar o estado e não mais será testado se existe interseção entre esse segmento com algum outro (Figura A.2(c)).

Quando um novo segmento é incluído no estado da linha de varredura não é necessário testar se esse segmento possui interseção com todos os outros segmentos listados no estado. Para isso, é necessário que os segmentos do estado da linha de varredura estejam ordenados da esquerda para a direita de acordo com a ordem em que cruzam a linha de varredura. Dessa forma, o segmento é incluído já ordenado e precisa ser testado apenas contra os segmentos adjacentes a ele.

Quando existe interseção entre dois segmentos, o ponto de interseção é identificado como um evento de interseção e os segmentos que se interceptaram trocam de ordem no estado da linha de varredura como ocorre na Figura A.2(d).

Portanto, é necessário operar duas estruturas de dados no processo de varredura. A primeira (estado da linha de varredura) é a responsável por manter a ordenação das interseções dos segmentos com a linha de varredura e é usualmente implementada como uma árvore binária balanceada. Essa árvore deve suportar as operações de inserção e exclusão com complexidade $O(log(n))$, onde n é o número de elementos armazenados na árvore, e duas funções para determinar qual segmento está imediatamente à direita e imediatamente à esquerda de um segmento com ordem de complexidade $O(1)$.

A segunda estrutura de dados é responsável por manter a seqüência dos eventos de ponto que serão analisados pela linha de varredura, e é implementada como uma fila de prioridades.

Essa fila de prioridades deve ser dinâmica pois novos eventos são computados à medida que a linha de varredura caminha e encontra interseções. Dessa forma, devem ser suportadas operações de inserção e exclusão do ele-

mento de mais alta prioridade e uma função que verifica se um determinado elemento está ou não na estrutura. Todas essas funções devem possuir ordem de complexidade $O(\log(n))$, onde n é o numero de eventos armazenados, para garantir a complexidade global de $O(n \log(n))$ (de Berg et al. 2000).

Os eventos de ponto devem ficar ordenados na fila de prioridade de tal forma que o ponto com maior ordenada y seja sempre o ponto de maior prioridade. Pontos com mesma ordenada y são tratados da esquerda para a direita, o de menor coordenada x tem maior prioridade. É interessante também poder tratar dois pontos extremos coincidentes como um único evento já que isso é comum em mapas planares.

Usando essas estruturas é possível implementar um algoritmo que calcula as interseções de um conjunto de n segmentos com ordem de complexidade $O(n \log n + I \log n)$, onde I é o número de pontos de interseção. O algoritmo em pseudo linguagem e maiores detalhes sobre a o cálculo da ordem de complexidade podem ser encontrados em (de Berg et al. 2000).

Além de ser um algoritmo mais complexo e difícil de ser implementado, casos degenerados como dois segmentos sobrepostos não são considerados na formulação e devem ser tratados a parte.

A.2 Composição de duas subdivisões

Para fazer a composição de dois mapas planares D_1 e D_2 representados por duas DCEL's usa-se o algoritmo de varredura descrito anteriormente.

Como ilustrado na Figura A.3, inicialmente constrói-se um terceiro mapa D_3 com uma cópia dos mapas originais. Note que esse DCEL é inválida já que existirão segmentos se cruzando. À medida que o algoritmo de varredura é processado sobre os segmentos do novo mapa as semi-arestas devem ser

Figura A.3: Composição de duas subdivisões planares pelo método da linha de varredura

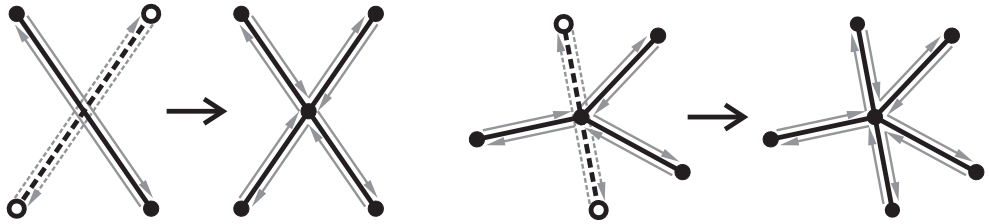
rearranjadas a fim de tornar a DCEL válida. As informações das faces são processadas posteriormente.

Se o evento de ponto é um vértice extremo de segmentos de apenas um dos dois mapas, esse vértice pode ser reaproveitado. Se o evento envolve segmentos de ambos os mapas, modificações locais são necessárias.

A Figura A.4 mostra duas das possíveis situações onde ocorre interseção entre segmentos dos dois mapas. A Figura A.4(a) mostra o caso em que os segmentos se cruzam em um ponto não extremo para ambos os segmentos. Nesse caso, um novo vértice deve ser adicionado no ponto de cruzamento. Além disso, cada aresta deve ser quebrada em duas gerando quatro novas semi-arestas. Os ponteiros das semi-arestas para o vértice de origem, vértice de destino, próxima semi-aresta e semi-aresta anterior devem ser atualizados para todas as semi-arestas envolvidas.

A Figura A.4(b) mostra o caso em que um segmento de um dos mapas passa sobre um vértice do outro mapa. Nesse caso, somente o segmento do primeiro mapa deve ser quebrado em dois. Além dos ponteiros das semiarestas do primeiro mapa, é necessário atualizar apenas os ponteiros das semi-arestas adjacentes à aresta do primeiro mapa.

Existem ainda outros casos possíveis. Tanto os casos apresentados quanto a maior partes dos outros casos possíveis podem ser processados com $O(1)$ (de Berg et al. 2000).



(a) (b)

Figura A.4: Quebra de segmentos com interseção

A.3 Classificação das regiões

Depois de processar o algoritmo de varredura teremos uma DCEL com os registros dos vértices e arestas corretos. Falta agora adequar os registros das faces. Cada face deve apontar para uma das semi-arestas do seu contorno externo. Deve-se criar uma lista dos contornos internos que definem os buracos da face e todas as semi-arestas dos contornos externo e internos devem ser atualizados para apontar para a face. Por fim cada face deve ter sua propriedade atualizada a partir das propriedades das faces das subdivisões originais.

Toda face possui um único contorno externo, com exceção da face infinita. Dessa forma, o número de registros de faces a serem criados é igual ao número de contornos externos mais um. Para saber se um contorno é externo ou interno, basta analisar o vértice mais à esquerda do ciclo (em caso de empate escolhe-se o mais abaixo), como ilustrado na Figura A.5. Se o ângulo formado pelas semi-arestas que incidem sobre o vértice é menor que 180^o significa que esse é um ciclo externo. Caso contrário trata-se de do contorno de um buraco.

Para saber quais contornos pertencem a uma mesma face constroi-se um grafo como o ilustrado na Figura A.6. Cada ciclo, externo ou interno, é representado por um nó no grafo. A face infinita possui um ciclo externo imaginário que também é representado por um nó no grafo. Existirá um

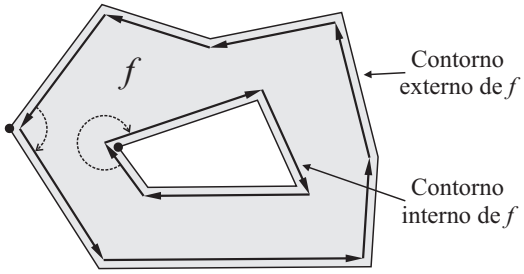


Figura A.5: Contornos das faces

arco entre dois nós do grafo se e somente se um dos ciclos é o contorno de um buraco e o outro ciclo possuir uma semi-aresta imediatamente à esquerda do vértice mais à esquerda do primeiro ciclo. Se não existe nenhuma semiaresta à esquerda do vértice mais à esquerda, então o nó representando o contorno do buraco é ligado ao nó do contorno imaginário da face infinita.

No exemplo da Figura A.6, as linhas pontilhadas representam as ligações dos contornos dos buracos com outros contornos. Cada linha pontilhada corresponde a um arco no grafo. No grafo, os nós que representam o contorno externo das faces estão representados pelos círculos em negrito. Os demais nós representam os contornos de buracos. Os nós C_1, C_8 e C_{10} estão ligados ao nó C_∞. Isso significa que os contornos C_1, C_8 e C_{10} definem buracos na face infinita. Da mesma forma que C_3 e C_6 ligados ao nó C_2 indica que os contornos C_3 e C_6 definem buracos na face definida pelo contorno externo C_2. A construção do grafo pode ser feita durante o processo de varredura do algoritmo de interseção de segmentos.

O último passo diz respeito à atualização das propriedades de cada face. Cada face f deve receber os atributos das faces originais. Para encontrar as faces originais considera-se um vértice arbitrário v de f. Se v é um ponto de interseção entre uma aresta e_1 da primeira subdivisão com uma aresta e_2 da segunda subdivisão, então basta verificar os ponteiros das semi-arestas

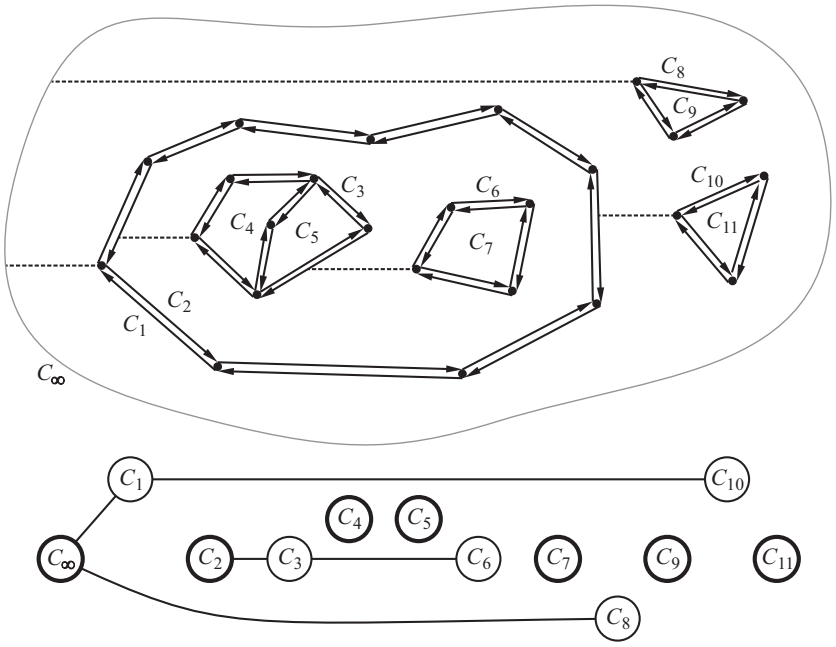


Figura A.6: Classificação das faces

adequadas de e_1 e e_2. Se v não é um ponto de interseção mas um vértice da primeira subdivisão, então só é possivel determinar a face original pertencente à primeira subdivisão. Para determinar a face original da segunda subdivisão é necessário determinar qual face contem o vértice v. Esse procedimento também pode ser realizado durante o processo de varredura do algoritmo de interseção. Mais detalhes dos procedimentos podem ser encontrados em (de Berg et al. 2000).

Apêndice B

Obtenção do caminho mínimo utilizando o método de Dijkstra contínuo

Dado um mapa planar, representado por uma subdivisão poligonal com custos α_f para as faces como descrito no Capítulo 2. Tem-se como objetivo determinar o caminho pelo plano que minimiza o custo total baseado na métrica Euclidiana ponderada.

Nesse capítulo usam-se as seguintes notações: uma aresta e compartilhada pelas faces f e f' orientada de tal forma que f está à sua direita, é expressa como: $e = \cap(f, f')$. Se não estamos preocupados com a orientação da aresta, basta escrever $e = f \cap f'$. Usa-se $int(\cdot)$ para referir-se ao interior relativo de uma aresta ou face. Um ponto está no interior de uma aresta ou face se esse ponto está sobre a aresta ou sobre a face.

Pode-se especificar um caminho linear dando a seqüência de pontos $\mathcal{P} = \{q_1, q_2, .., q_k\}$ de tal forma que esse caminho seja composto pela seqüência de segmentos de reta $\overline{q_i q_{i+1}}$, $1 \leqslant i \leqslant k-1$.

B.1 Caminhos Geodésicos

Um caminho geodésico é um caminho localmente ótimo e que não pode ser encurtado ao se aplicar perturbações leves. Um caminho ótimo é um caminho geodésico que é globalmente ótimo.

O objetivo é encontrar um caminho dado por uma curva $\mathcal{L} \in \mathcal{F}$, onde $\mathcal{F}$ é a região livre do mapa (ver Equação (2.4)), que conecta um ponto inicial $q_0 = (x_0, y_0)$ a um ponto de destino $q_d = (x_d, y_d)$ e que minimiza o funcional de custo dado por

$$I = \int_{\mathcal{L}} g(x, y) ds, \tag{B.1}$$

onde ds é o diferencial de comprimento de arco e $g(x, y)$ é a função custo do mapa apresentada na Equação (2.3).

É assumido que cada região possui um custo **constante**. Dessa forma, cada caminho geodésico será sempre linear por partes exceto para regiões de peso zero (onde os sub-caminhos podem ser arbitrários).

Caminhos geodésicos são necessariamente caminhos simples (não passam por um mesmo ponto mais de uma vez). Assim, caminhos ótimos devem ser simples, exceto para faces onde o peso é zero. No interior dessas faces o caminho é qualquer (navegação livre). Mas todo caminho complexo no interior de faces com peso zero pode ser substituído por um caminho simples linear por partes.

Assumindo, por enquanto, que $\alpha_e = min\{\alpha_f, \alpha_{f'}\}$ para a aresta $e = f \bigcap f'$. Então, um caminho geodésico que atravessa e obedece a Lei de Snell para a refração da luz. A Lei de Snell afirma que caminho de um raio de luz que passa pela fronteira entre duas regiões f e f' de índices de refração α_f e $\alpha_{f'}$ obedece a relação $\alpha_f \operatorname{sen}(\theta) = \alpha_{f'} \operatorname{sen}(\theta')$ onde θ e θ' são os ângulos de incidência e de refração, respectivamente (Figura B.1).

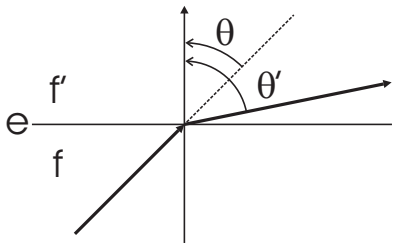


Figura B.1: Ângulos de incidência e de refração

Assumindo-se que $\alpha_e = min\{\alpha_f, \alpha_{f'}\}$ e que $\alpha_f, \alpha_{f'} < \infty$, se p é um caminho geodésico que passa pelo interior de e, então p obedece a lei de Snell na aresta e.

No problema abordado, um caminho geodésico nunca incidirá em uma aresta com um ângulo maior (em módulo) que o ângulo crítico. Não se faz necessário o tratamento de reflexão total. A única forma de reflexão possível para um caminho geodésico ocorre da seguinte forma: o caminho incide sobre a aresta e a partir da face f com o ângulo crítico $\theta = \theta_c$ no ponto y, então esse caminho permanece sobre a aresta por uma certa distância e então deixa a aresta retornando para a face f a partir de um ponto y também com o ângulo crítico $\theta' = \theta_c$, como ilustrado na Figura B.2.

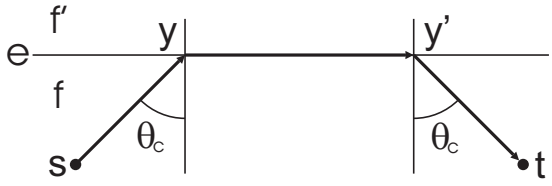


Figura B.2: Caminho criticamente refletido

Se $\alpha_e < min\{\alpha_f, \alpha_{f'}\}$, é possível que o caminho permaneça sobre a aresta e antes de atravessá-la. O caminho alcança a aresta com o ângulo crítico $\theta_c(f, e) = \arcsin(\alpha_e/\alpha_f)$ no ponto $p \in int(e)$, permanece sobre a aresta por alguma distância e então deixa a aresta no ponto $q \in int(e)$ com o ângulo crítico $\theta_c(f', e) = \arcsin(\alpha_e/\alpha_{f'})$, conforme a Figura B.3. Dizemos então que

o caminho *usou criticamente* a aresta e ao longo do segmento crítico $\overline{pq'}$. Isso também vale quando o caminho é criticamente refletido.

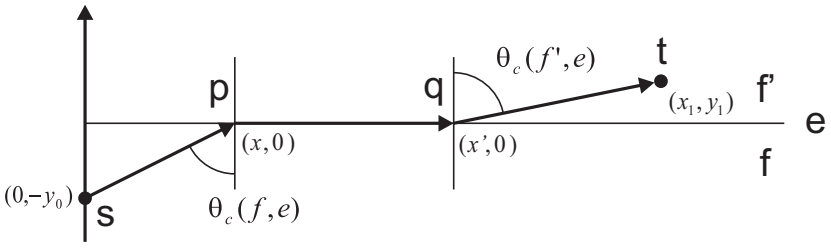


Figura B.3: Caminho sobre aresta de baixo custo

Se p é um caminho geodésico que passa por regiões com $\alpha_i > 0$ começando de um ponto s para um ponto x que conecta uma seqüencia de arestas $\mathcal{E} = (e_1, ..., e_k)$ com $e_i \neq e_{i+1}$ e não existindo segmentos compartilhados, então p é único e obedece a lei de Snell para cada ponto de cruzamento em cada aresta.

Pontos críticos: para um dado caminho geodésico p, um ponto $y \in p \bigcap int(f \bigcap f')$ é chamado de *ponto crítico de entrada do caminho p a partir da face f* se y é o mais próximo do ponto fonte (s) entre os dois pontos extremos do segmento compartilhado $\overline{yy'}$, e p alcança o ponto y a partir da face f. Ou seja, y é interior a aresta $e = f \bigcap f'$ e o ângulo que p faz no ponto y é crítico: $\angle xyx' = \theta_c + \frac{\pi}{2}$. Analogamente, y' é chamado de *ponto crítico de saída do caminho p para a face f'* (Figura B.4).

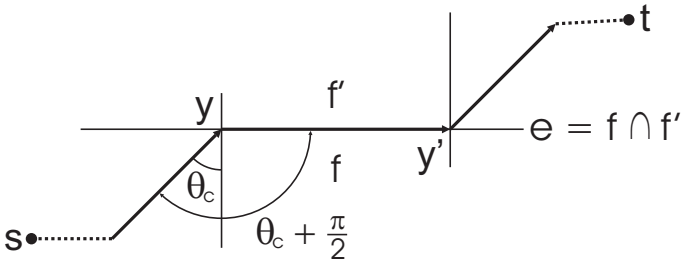


Figura B.4: Pontos críticos

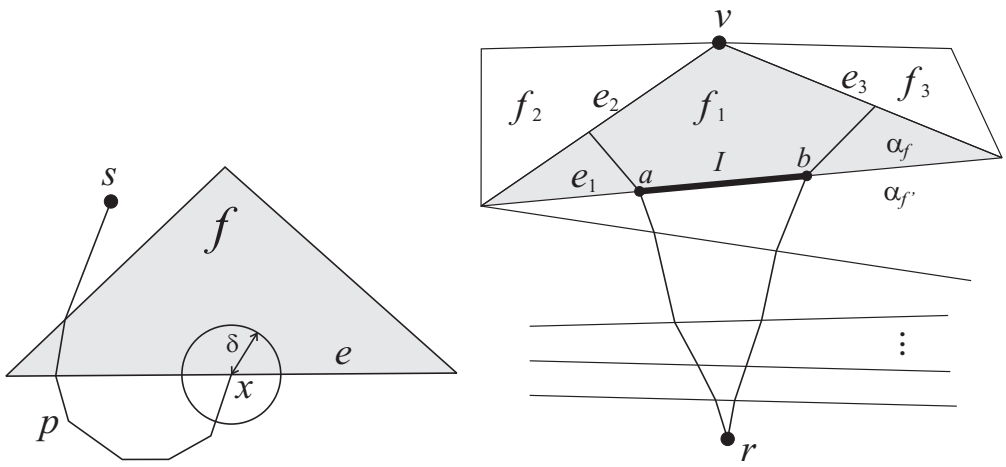
Retrocedendo a partir do ponto x sobre um caminho p e seja r o primeiro vértice ou ponto crítico de entrada que encontramos. Então r é a *raiz* do caminho p. Denota-se a seqüencia de arestas que p atravessa entre r e x de *última seqüencia de arestas* do caminho p. Se p é um caminho geodésico para o ponto x que possui última seqüencia de arestas $\mathcal{E}$ e raiz r, então define-se $d_{r,\mathcal{E}}(x)$ como sendo a distância ponderada de r até x sobre p. Essa distância é unicamente determinada por r, x e $\mathcal{E}$.

B.2 O método de Dijkstra contínuo

A idéia básica do método descrito em (Mitchell 1991) está em simular o efeito de uma frente de onda que se propaga a partir do vértice de saída s. A medida que a frente de onda se afasta do ponto de origem, ela colide com outros vértices e arestas da subdivisão. Essas colisões são chamadas de eventos. Uma frente de onda que dista d de s é o conjunto de todos os pontos P em que o caminho geodésico até s possui custo d. Essa frente de onda consiste de um conjunto de arcos de circunferência denominados *wavelets*. Cada wavelet é um arco de circunferência centrado em um vértice já alcançado pela frente de onda. Como será visto, o método assemelha-se em muito ao método de Dijkstra discreto, descrito com detalhes na seção 3.3, proposto para encontrar o menor caminho em um grafo visitando os nós à medida que a distância ao nó fonte aumenta.

O método exige como entrada uma triangulação qualquer da subdivisão planar que representa o mapa temático. Além disso, é necessário que as coordenadas dos vértices e os pesos das regiões tenham valores inteiros. Na verdade o peso das regiões deve ser $\alpha \in \{0, 1, ..., W, +\infty\}$, onde W é maior valor de peso finito.

São necessárias algumas estruturas de dados para implementar o método. Uma delas é uma lista com intervalos candidatos a intervalos de otimalidade. Um intervalo de otimalidade descreve um subconjunto de uma aresta no qual os caminhos f-free para a aresta possuem a mesma estrutura (partindo de um mesmo ponto raiz, atravessam a mesma seqüência de arestas). Dada uma face f e e, uma das 3 arestas de f, um caminho localmente f-free para $x \bigcap e$ é definido como sendo um caminho geodésico p a partir de s (ponto de origem) até x de tal forma que existe um $\delta > 0$ em que o caminho não passa através da interseção do interior de f com a bola de raio δ com centro em x (Figura B.5(a)). Intuitivamente, um caminho localmente f-free até $x \bigcap e$ "alcança x pelo exterior de f" e é localmente ótimo. No exemplo da Figura B.5(b), $I = [a, b]$ é um intervalo de otimalidade de r até e_1. O caminho ótimo entre r e qualquer ponto no interior desse intervalo possui a mesma seqüência de arestas.



(a) Caminho f-free (b) Intervalo de otimalidade

Figura B.5: Propagação de Intervalos

Cada par face-aresta possui associado a ele uma lista ordenada de intervalos candidatos de acordo com a ordem em que os intervalos aparecem sobre a aresta.

Cada vértice v tem um ponteiro para cada intervalo candidato para os quais v é raiz. Além dos ponteiros é associado ao vértice um valor $d(v)$ que indica o custo do melhor caminho até v. Inicialmente $d(v) = \infty$ para todo vértice v. À medida que o algoritmo é processado cada vértice v (e eventuais pontos críticos de entrada) é marcado como permanente, significando que $d(v)$ é o valor do caminho ótimo até v partindo do ponto de saída s. Antes de ser marcado como permanente, v é marcado como temporário em concordância com o método de Dijkstra discreto.

O método mantém também uma fila de prioridades que implementa uma fila de eventos. As entradas dessa fila são pontos de algum intervalo candidato. O ponto é um extremo do intervalo ou um ponto da fronteira que se sabe ser o ponto que possui menor custo para se chegar ao vértice de saída.

O algoritmo é processado selecionando-se o próximo ponto da fila de eventos. Propaga-se a frente de onda através do correspondente intervalo projetando-o sobre a face apropriada. Depois disso, atualiza-se a lista de intervalos candidatos. O evento escolhido é sempre o que possui menor custo em relação ao vértice de saída. Depois de processado, ele é marcado como permanente. Faz-se isso enquanto existirem eventos na fila.

O método apresentado com detalhes em (Mitchell 1991) garante encontrar um caminho com um fator de $(1+\epsilon)$ do caminho ótimo, onde $epsilon > 0$ é o grau de precisão desejado. Apesar disso, o método é de difícil implementação e computacionalmente caro. O tempo de execução possui uma ordem de complexidade de $O(E \cdot S)$, onde E é o número de eventos e S é a complexidade de se resolver o sub-problema de se encontrar o caminho ótimo com uma precisão de $(1 + \epsilon)$ entre dois pontos s e t da subdivisão, dada a seqüência de arestas que o caminho atravessa de um ponto ao outro. É mostrado que $E = O(n^4)$ no limite superior, onde n é o número de vértices da subdivisão.

Para encontrar o caminho entre s e t pode-se aplicar uma busca binária que pode ter uma complexidade de até $S = O(k^2 \log(nNW/\epsilon))$ para o pior caso onde k é o número de arestas, N é a maior coordenada inteira de qualquer vértice da triangulação. É possível provar ainda que $k = O(n^2)$ o que gera uma complexidade final de $O(n^8 \log(nNW/\epsilon))$ para o método descrito.

A próxima seção apresenta uma aproximação do método de Dijkstra contínuo na qual a ordem de complexidade é menor, $O(n^3)$ para o pior caso.

B.3 Aproximação do método de Dijkstra contínuo

O princípio de funcionamento desse método, (Mata & Mitchell 1997), baseia-se em construir um grafo de pesquisa a partir da subdivisão planar capaz de aproximar o caminho ótimo entre dois vértices da subdivisão. Para cada vértice v da subdivisão planar traçam-se k raios igualmente espaçados de forma a aproximar toda a faixa de $[0, 2\pi]$, como ilustrado na Figura B.6. Dois raios consecutivos definem um cone de propagação.

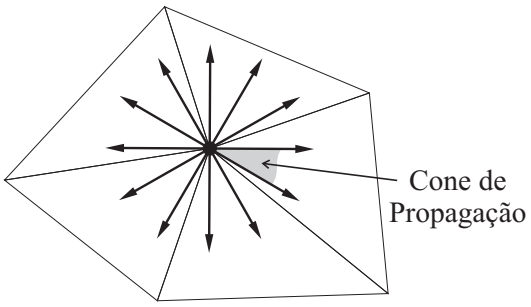


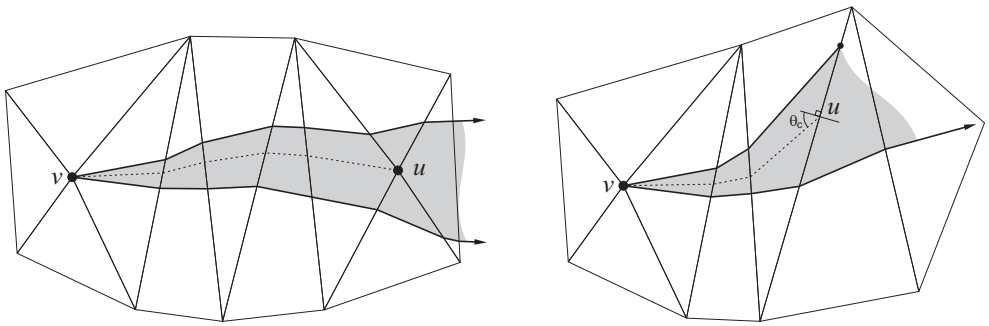
Figura B.6: Cones de propagação a partir de um vértice

Cada um desses cones varre o plano a procura de um vértice u ou de um ponto crítico de incidência, a fim de criar uma ligação entre v e u no

grafo. Cada cone de varredura determina no máximo uma ligação sendo que essa pode não existir. A possível ligação é delimitada pelos raios limites do cone de propagação traçados, a partir de v, obedecendo a Lei de Snell para a refração em cada ponto de cruzamento entre os raios e as arestas.

Os raios que limitam um cone cruzam a mesma seqüência de arestas até que uma ***bifurcação topológica*** seja encontrada. O primeiro tipo de bifurcação é quando os raios que limitam o cone alcançam arestas diferentes, como ilustrado na Figura B.7(a). Nesse caso, existe pelo menos um vértice u da subdivisão alcançado pelo cone. Então faz-se uma ligação no grafo entre v e u e atribui-se a essa ligação o valor do caminho de refração de v até u. O caminho localmente ótimo entre v e u é único e respeita a lei de Snell em todo o cruzamento de aresta. Pode-se encontrar um caminho arbitrariamente próximo ao caminho localmente ótimo através de uma busca binária, por exemplo.

O segundo tipo de bifurcação acontece quando os raios encontram uma mesma aresta e mas pelo menos um deles incide com um ângulo com módulo maior que o ângulo crítico θ_c, como ilustrado na Figura B.7(b). Existem no máximo dois caminhos localmente ótimos partindo do vértice v até a aresta e, correspondendo aos ângulos críticos θ_c e $-\theta_c$.



(a) vértice (b) ponto crítico

Figura B.7: Bifurcações topológicas

O *caminho crítico de refração* é o caminho que une o vértice v à aresta e e incide criticamente em e. Nesse caso, determina-se o caminho crítico de refração de v até e e o ponto u em que o caminho toca e é adicionado como um vértice da subdivisão. Esse caminho passa a ser uma ligação no grafo.

É possível ainda que o cone de propagação alcance o contorno externo da subdivisão sem encontrar nenhuma bifurcação topológica. Nesse caso simplesmente nenhuma ligação é acrescida ao grafo.

Uma vez que o grafo tenha sido construído, usa-se o algoritmo de Dijkstra tradicional descrito na seção 3.3 para encontrar o caminho entre dois vértices sobre o grafo.

O grafo gerado pelo método possui $O(kn)$ nós, onde k é o número de cones e n é o número de vértices da subdivisão.

Em (Aleksandrov, Maheshwari & Sack 2005), além do método apresentado pelo autor, é apresentada uma comparação entre a ordem de complexidade de diversos métodos para se encontrar o menor caminho em mapas poligonais, inclusive os descritos anteriormente (Mitchell 1991, Mata & Mitchell 1997).

Apêndice C

CGAL - Computational Geometry Algorithms Library

A CGAL, Computational Geometry Algorithms Library, é uma biblioteca de algoritmos para geometria computacional e foi utilizada para implementar grande parte desse trabalho. Nesse apêndice será dada uma visão geral da estrutura da CGAL e das duas principais classes utilizadas.Mais informações podem ser encontradas no site da biblioteca (CGAL 2005).

A CGAL foi inicialmente desenvolvida por uma parceria entre as universidades de ETH Zürich (Suíça), Freie Universität Berlin (Alemanha), INRIA Sophia-Antipolis (França), Martin-Luther-Universität Halle-Wittenberg (Alemanha), Max-Planck Institut für Informatik, Saarbrücken (Alemanha), RISC Linz (Áustria) Tel-Aviv University (Israel) e Utrecht University (Holanda). A biblioteca se tornou um projeto aberto em novembro de 2003.

A CGAL é escrita em C++ e consiste de três partes principais.

A primeira parte é o *kernel*, que consiste de um conjunto fixo, não modificável, de objetos geométricos primitivos e operações sobre esses objetos. Os objetos podem ser representados por classes parametrizadas independentes ou podem ser membros de outras classes do *kernel*.

A segunda parte é uma coleção de estruturas de dados geométricos básicas e algoritmos parametrizados por uma classe *traits* que define a interface entre a estrutura de dados ou algoritmo e as primitivas em uso. Em muitos casos, classes fornecidas pelo *kernel* podem ser usadas como classes *traits* para essas estruturas de dados e algoritmos. A coleção de algoritmos geométricos básicos e estruturas de dados inclui polígonos, estruturas de dados baseadas em semi-arestas, superfícies poliédricas, mapas topológicos, mapas planares, arranjos de curvas, triangulações, fechos convexos, algoritmos de otimização, conjuntos dinâmicos de pontos para pesquisas geométricas e árvores de busca multi-dimensionais.

A terceira parte da biblioteca consiste em um suporte a algumas facilidades não geométricas como suporte a tipos numéricos, extensões da STL (*Standard Template Library*) para a CGAL, *handles*, *circulators*, proteção para o acesso a representações internas, temporizadores, operadores de entrada e saída, ferramentas de visualização, entre outros.

A seguir serão brevemente descritas as duas principais estruturas de dados utilizadas nesse trabalho.

C.1 Planar_map_2

Como visto na seção 2.2.1, um mapa planar subdivide o plano em vértices, arestas e faces. A estrutura de dados Planar_map_2 fornecida pela CGAL é capaz de encapsular um mapa planar e foi utilizada para esse fim nesse trabalho.

A classe Planar_map_2<Dcel,Traits> é parametrizada por dois objetos. O objeto Dcel mantém uma lista de arestas duplamente conectadas. O objeto Traits provê as funcionalidades geométricas e é capaz de tratar uma família

específica de curvas e encapsula o tipo numérico para a representação das coordenadas.

O trecho de código a seguir é um exemplo simples de como se utiliza um Planar_map_2. Nesse exemplo um mapa planar é criado e três segmentos são adicionados a ele formando um triângulo.

```
//includes da CGAL, necessários para definir os tipos.
#include <CGAL/Cartesian.h>
#include <CGAL/MP_Float.h>
#include <CGAL/Quotient.h>
#include <CGAL/Pm_segment_traits_2.h>
#include <CGAL/Pm_default_dcel.h>
#include <CGAL/Planar_map_2.h>

//Definição de tipos a serem usados
typedef CGAL::Quotient<CGAL::MP_Float>    Number_type;
typedef CGAL::Cartesian<Number_type>      Kernel;
typedef CGAL::Pm_segment_traits_2<Kernel> Traits;
typedef Traits::Point_2                   Point_2;
typedef Traits::X_monotone_curve_2        X_monotone_curve_2;
typedef CGAL::Pm_default_dcel<Traits>     Dcel;
typedef CGAL::Planar_map_2<Dcel,Traits>   Planar_map;

int main()
{
  Planar_map pm;                      //Instancia o mapa planar
  X_monotone_curve_2 cv[3];           //cria um vetor de segmentos
  Point_2 a1(0,0), a2(0,4), a3(4,0);  //cria os pontos extremos
                                      //dos segmentos

  cv[0] = X_monotone_curve_2(a1,a2);  //segmento de a1 a a2
  cv[1] = X_monotone_curve_2(a2,a3);  //segmento de a2 a a3
  cv[2] = X_monotone_curve_2(a3,a1);  //segmento de a3 a a1
  pm.insert(&cv[0], &cv[3]);          //insere segmentos no
                                      //mapa

  return 0;
}
```

A classe Planar_map_2<Dcel,Traits> inclui um conjunto de funções de interface que permitem construir, modificar, fazer pesquisas, salvar e restaurar um mapa planar. Alguns de seus construtores permitem a escolha entre várias estratégias de localização de pontos.

Uma vez que o mapa planar foi construído, pode-se inserir uma curva ou um conjunto de curvas, remover uma curva ou dividir uma curva em duas, localizar pontos, etc.

C.2 Nef_polyhedron_2

Um *Nef polyhedron* planar é qualquer conjunto que pode ser obtido a partir de um conjunto finito de semiplanos abertos através de um conjunto de operações de complemento e de interseção. Devido ao fato de que todas as outras operações binárias como união, diferença e diferença simétrica podem ser reduzidas a cálculos de interseção e complemento, todas essas operações são suportadas pela classe Nef_polyhedron_2. Além dessas operações, a classe também permite algumas operações unárias. Dado um Nef_polyhedron_2, é possível determinar o seu interior, seu contorno e operações de regularização.

A seguir é mostrado um exemplo simples de como construir objetos da classe Nef_polyhedron_2.

```
//includes da CGAL, necessários para definir os tipos.
#include <CGAL/Gmpz.h>
#include <CGAL/Filtered_extended_homogeneous.h>
#include <CGAL/Nef_polyhedron_2.h>

//Definição de tipos a serem usados
typedef CGAL::Gmpz RT;
typedef CGAL::Filtered_extended_homogeneous<RT> Extended_kernel;
typedef CGAL::Nef_polyhedron_2<Extended_kernel> Nef_polyhedron;
typedef Nef_polyhedron::Point Point;
typedef Nef_polyhedron::Line  Line;

int main()
{
  Nef_polyhedron N1(Nef_polyhedron::COMPLETE);   //todo o plano

  Line l(2,4,2); // l : 2x + 4y + 2 = 0          //defindo reta l
  Nef_polyhedron N2(l,Nef_polyhedron::INCLUDED); //plano à esquerda de l
                                                 //incluindo a reta.
  Nef_polyhedron N3 = N2.complement();           //complemento de N2
  CGAL_assertion(N1 == N2.join(N3));             //uniao de N2 e N3

  Point p1(0,0), p2(10,10), p3(-20,15);     //pontos extremos do triângulo
  Point triangle[3] = { p1, p2, p3 };       //cria vetor de pontos
  Nef_polyhedron N4(triangle, triangle+3);  //gera triângulo a partir do
                                            //vetor de pontos
  Nef_polyhedron N5 = N2.intersection(N4);  //interseção entre N2 e N4
  CGAL_assertion(N5 <= N2 && N5 <= N4);     //verifica se o resultado da
                                            //interseção é menor ou igual
                                            //aos Nef's originais

  return 0;
}
```

Semiplanos são modelados como linhas orientadas, localizando-se à esquerda da mesma. No exemplo anterior, $N1$ é um *Nef polyhedron* representando todo o plano, $N2$ é o semiplano fechado à esquerda da reta orientada que possui equação $2x + 4y + 2 = 0$ incluindo a reta, $N3$ é o complemento de $N2$ e, dessa forma, tem-se $N2 \cup N3 = N1$. Adicionalmente, pode-se construir um *Nef polyhedron* a partir de um conjunto de pontos que definem o contorno de um polígono. No exemplo, $N4$ é o triângulo definido pelos vértices $(0, 0)$, $(10, 10)$ e $(-20, 15)$.