COPPE/UFRJ

ALGORITMOS DE PLANEJAMENTO EM DIMENSÃO ARBITRÁRIA

Hélio Bomfim de Macêdo Filho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Celina Miraglia Herrera de Figueiredo
Guilherme Dias da Fonseca

Rio de Janeiro
Fevereiro de 2010

ALGORITMOS DE PLANEJAMENTO EM DIMENSÃO ARBITRÁRIA

Hélio Bomfim de Macêdo Filho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Celina Miraglia Herrera de Figueiredo, D.Sc.

Prof. Guilherme Dias da Fonseca, Ph.D.

Prof. Claudio Esperança, Ph.D.

Prof. Thomas Lewiner, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2010

Macêdo Filho, Hélio Bomfim de

Algoritmos de Planejamento em Dimensão Arbitrária/Hélio Bomfim de Macêdo Filho. – Rio de Janeiro: UFRJ/COPPE, 2010.

XV, 85 p.: il.; 29, 7cm.

Orientadores: Celina Miraglia Herrera de Figueiredo
Guilherme Dias da Fonseca

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 79 – 83.

1. Geometria Computacional. 2. Planejamento de Movimento. 3. Algoritmo. 4. Amostragem. I. Figueiredo, Celina Miraglia Herrera de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Ao meu pai Hélio Bomfim de Macêdo (*in memoriam*).*

Agradecimentos

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro. Aos meus orientadores, Celina e Guilherme, por tudo que aprendi com eles. Aprendizado que transcedeu os limites da pesquisa científica.

Aos meus genitores, especialmente ao meu pai, que dedico o título de mestre. Meu pai, especialmente, sempre foi o maior incentivador para que eu me dedicasse à carreira acadêmica. Infelizmente, ele partiu cedo demais e teve o direito tolhido de acompanhar as fases do meu mestrado, do início ao fim. À minha mãe que substituiu de forma equivalente as condições que meu pai certamente me proveria em todo esse tempo. Também agradeço ao restante de minha família. Devo ressaltar, especificamente, o meu agradecimento à duas famílias em essencial: à família da minha tia, por parte de pai, Socorro Macêdo, vulgo Totô, e à família da minha tia, por parte de mãe, Branca Stul. Guardo eterno agradecimento à ambas, por diversos motivos, especialmente por me darem segurança e terem, quaisquer que sejam as causas, o meu profundo sentimento de confiança nelas.

Aos meus amigos de colégio, que sempre estiveram presentes em minha vida. Pelo menos uma dezena de anos de amizade que, mesmo que não estejamos nas mesmas cidades ou mesmo em continentes diferentes, ainda tenho a segurança de que posso contar com eles para qualquer necessidade. Vou citar alguns, por ordem alfabética, mas que pertencem à mesma classe de equivalência da mais alta casta de amizade: Bruno Vasconcelos, Felipe Leal, Gilson Almeida, Rui Kléber e Leonardo Felipe.

Aos meus amigos de faculdade, que foram minha família por 4 anos em turnos integrais. São tantos, que possivelmente não lembrarei de todos. Portanto, para não ocasionar divergências, espero que os que se sintam encaixados nesse grupo, sejam agraciados com meu agradecimento. Aos membros do CRAb, especialmente

os professores Joaquim Bento e Creto Vidal. Aos membros do ParGO, especialmente os professores Manoel Campêlo, Cláudia Linhares e Ricardo Corrêa.

Aos meus amigos de apartamento no Rio de Janeiro, com quem compartilhei ótimos momentos e foram excelentes comigo: Heraldo Carneiro, Samuel Barbosa e Yuri Lima. Nos 2 anos que convivi com eles, particionados em conjuntos disjuntos de 1,5 ano com o Heraldo, no Jardim Guanabara, e 0,5 ano com o Samuel e o Yuri, no Leblon. Desses 2 anos, só guardei boas lembranças na convivência com eles. Agradeço a paciência do Heraldo em uma má fase da minha vida, em que me sentia muito ligado à Fortaleza, e agradeço os bons momentos que proporcionei à todos, em seguida, na melhor fase da minha vida. Devo ressaltar a convivência com Olivério Fernandes, Vinícius Pires e Fabrício Raphael que foram os primeiros amigos que fiz no Rio de Janeiro e guardo ótimas lembranças.

Aos meus amigos de COPPETEC, com quem aproveitei a essência da vida. Três pessoas merecem fortemente um agradecimento especial aqui: Clarissa Vilela, Vinícius Von Held e professor Blaschek. São diversos os motivos, mas sucintamente vou citar apenas um de cada: Clarissa pelos conselhos, Vinícius e professor Blaschek por terem acreditado em mim.

Aos meus amigos da UFRJ, com quem aprendi e estou aprendendo cada vez mais sobre a vida acadêmica e o restante que a envolve. Em especial, os que fiz amizade no início e tenho boas lembranças em específico, por ordem alfabética: André Korenchendler, Caroline Reis, Daniel Posner, Danilo Artigas, Diana Sasaki, Érika Morais, Hebert Coelho, Marcelo Lopes e Murilo Silva. Peço desculpas aos demais, mas por falha minha de não estar tão presente, compartilhei poucos momentos.

À minha namorada Marina, a quem devo atribuir como a maior "culpada" da minha felicidade nos últimos 2 meses. Ela está ocupando os poucos vazios da minha vida e preenchendo com o que há de bom. Também agradeço à sua família que tem me recebido muito bem, especialmente aos recém casados: Marcela e Felipe Delson, que conheço há anos e são bem queridos por mim. Entrei pra família certa!

Ao Leblon. Morar lá certamente foi a melhor decisão que tomei no Rio de Janeiro: crucial para me dar o fôlego que precisava para terminar o mestrado.

Finalmente, o mais esperado! Sempre presente de forma inesperada às rodas de *chopps* que participei: um brinde à mim!

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMOS DE PLANEJAMENTO EM DIMENSÃO ARBITRÁRIA

Hélio Bomfim de Macêdo Filho

Fevereiro/2010

Orientadores: Celina Miraglia Herrera de Figueiredo
Guilherme Dias da Fonseca

Programa: Engenharia de Sistemas e Computação

O movimento planejado de robôs consiste em traçar o movimento do robô de um ponto inicial q_{ini} para um ponto final q_{fim}, evitando colisão com obstáculos e retornando o movimento se existir. Há duas principais filosofias utilizadas para resolver esse problema. A primeira consiste da construção explícita do espaço em que o robô pode se movimentar sem que haja colisão. O foco é resolver problemas de planejamento de movimento em espaços bidimensionais e tridimensionais. A segunda consiste da amostragem de pontos onde o robô pode se movimentar. O foco é resolver problemas de planejamento de movimento em dimensões superiores. Entretanto, na segunda filosofia, como não há uma construção explícita, não sabemos a forma dos objetos e temos que lidar com a incerteza sobre o formato dos obstáculos. Além dessas duas abordagens, há variações como, por exemplo, modelos realistas das cenas, como o caso em que os objetos não sejam longos e estreitos. Apresentaremos uma nova forma de resolver a variação citada como exemplo.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PLANNING ALGORITHMS IN ARBITRARY DIMENSION

Hélio Bomfim de Macêdo Filho

February/2010

Advisors: Celina Miraglia Herrera de Figueiredo
Guilherme Dias da Fonseca

Department: Systems Engineering and Computer Science

Robot motion planning consists of building the path for the movement of a robot from a initial point q_{ini} to a destination point q_{end} avoiding collisions with obstacles and returning the movement, if it exists. There are two main philosophies used to solve the problem. The first is based on the explicit construction of the space where the robot moves without collisions. It is better suited to solve motion planning problems in two-dimensional and three-dimensional spaces. The second is based on sampling the points where the robot can move. It is better suited to solve motion planning problems in higher dimensions. However, in the second philosophy, as there is no explicit construction, object shapes are not known and we must deal with the uncertainty of the obstacles shapes. Besides these two approaches, there are variations as, for example, realistic models for the scenes, as when objects are not long and narrow. We present a new way to solve the variation cited.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

C	Espaço de configuração, p. 3
C_F	Espaço de configuração livre, p. 4
C_P	Espaço de configuração proibida, p. 4
E	Conjunto de arestas de um grafo, p. 50
G	Grafo, p. 48
$G(r)$	Grade ortogonal regular de resolução r, p. 44
$G(r_1, \ldots, r_d)$	Grade ortogonal com resoluções $r_1, \ldots, r_d$, p. 46
I	Hiperesfera, p. 66
L	Elipsóide, p. 66
M	Espaço métrico, p. 42
N	Conjunto de vértices de um grafo, p. 50
N_k	k-vizinhança, p. 47
N_v	Conjunto de vizinhos de um vértice v, p. 50
P	Obstáculo, p. 2
P^*	Obstáculo expandido, p. 3
R	Robô, p. 2
$R(t)$	Robô R em um posicionamento t, p. 19
$S_{m,r}$	Bola fechada de raio r centrada em um ponto m, p. 61

T	Árvore, p. 11
U_E	Conjunto sobre um objeto E que consiste de todas as hiperesferas centradas dentro de E e sem conter E inteiramente, p. 61
$V(\cdot)$	Conjunto visibilidade, p. 54
W	Espaço físico, p. 2
α	Constante positiva arbitrariamente pequena, p. 55
β	Constante positiva arbitrariamente pequena, p. 55
ϵ	Constante positiva arbitrariamente pequena, p. 42
γ	Curva, p. 21
$\mu(\cdot)$	Volume, p. 54
$\ominus$	Diferença de Minkowski, p. 14
$\oplus$	Soma de Minkowski, p. 14
$\partial\cdot$	Fronteira, p. 21
$\pm$	Mais ou menos, p. 47
ρ	Métrica, p. 42
$\sup(\cdot)$	Supremo (Menor limite inferior), p. 44
d	Constante que denota a quantidade de dimensões, p. 2
$int(\cdot)$	Interior, p. 21
k	Constante que denota a quantidade de graus de liberdade, p. 3
$n!!$	Duplo fatorial, p. 60

Capítulo 1

Introdução

Dependendo da aplicação, tanto o agente utilizado na inteligência artificial quanto o robô utilizado na robótica precisam de um plano ou uma estratégia para traçar rotas. Imagine uma linha de montagem de automóveis, em que a tarefa de um robô é realizar manutenção nas máquinas, por exemplo, para colocar óleo. Todas as informações da linha de montagem, especialmente geométricas, estão disponíveis para o robô responsável pela manutenção. Um robô autônomo precisa saber que trajetórias escolher, em meio ao ambiente em que se encontra, de modo a conseguir se locomover evitando colisão com obstáculos. Além disso, em ambientes geometricamente complexos, o robô precisa saber que operações necessita realizar para guiar seu movimento quando não há certeza sobre a forma e a disposição dos objetos na linha de montagem. Responder estas questões é o objetivo do planejamento de movimentos.

O estudo acerca de planejamento de movimento de robôs foi um assunto estudado intensivamente nas últimas duas décadas [1]. Aplicações como computação gráfica e biologia molecular também motivaram o desenvolvimento desta área. No caso da computação gráfica, o planejamento de movimentos pode reduzir bastante a interação humana nos jogos. Enquanto na biologia molecular, pode-se simular o movimento de moléculas se acoplando em uma parede celular.

Inicialmente, o problema de planejamento de movimentos foi formulado como o problema do movimento de pianos [2]: Dado um objeto R e uma região limitada por uma coleção de paredes, em que R evita colisão com elas, o problema consiste em encontrar um movimento contínuo conectando duas posições ou dizer que o

movimento não existe. As posições, assim como a orientação de R são dadas como entrada do problema.

Algumas definições são necessárias para formalizarmos o problema. A Seção 1.1 apresenta as definições preliminares. Na Seção 1.2, falamos sobre estruturas de dados necessárias para alguns algoritmos de planejamento presentes no texto. Na Seção 1.3, é feita uma breve descrição das duas principais filosofias utilizadas para a resolução do problema de planejamento. Esta introdução termina com um roteiro dos demais capítulos e como os capítulos se relacionam.

1.1 Preliminares

Iniciaremos nossas definições preliminares começando pelo robô, descrição geométrica de tamanho constante que o robô tem e o espaço físico em que o robô se move. Observe que, ao longo do texto, tratamos a dimensão como um valor arbitrário, ou seja, a dimensão tem um valor constante d.

Definição 1.1 (Descrição de complexidade constante)**.** Descrição geométrica definida por um número constante de equações e inequações polinomiais de grau máximo constante, onde o número de variáveis também é constante.

Definição 1.2 (Robô R)**.** Um poliedro R de complexidade constante d-dimensional.

Definição 1.3 (Espaço físico W)**.** O ambiente W, d-dimensional, no qual o robô R se move.

No espaço físico temos possivelmente obstáculos com os quais o robô deve evitar colidir.

Definição 1.4 (Obstáculo P)**.** Um obstáculo P é um poliedro de complexidade constante, d-dimensional.

Podemos considerar apenas a parte do espaço físico em que não haja colisão com nenhum obstáculo, ou seja, também é possível definir o espaço físico W, d-dimensional, por um conjunto de obstáculos P_i, $i = 1, \ldots, n$, tal que $W = \mathbb{R}^d \setminus \bigcup_{i=1}^{n} P_i$.

Já que temos todos os objetos em um espaço físico, agora precisamos definir o posicionamento do robô R no espaço físico.

Definição 1.5 (Configuração ou posicionamento)**.** A porção do espaço físico ocupada pelo robô em um dado instante.

Note que no restante do texto, usaremos o termo posicionamento e configuração de forma intercambiável, dependendo do contexto. As definições sobre configuração também se aplicam a posicionamento e vice-versa.

O posicionamento do robô R pode eventualmente ter interseção com algum obstáculo.

Definição 1.6 (Configuração ou posicionamento livre/proibido)**.** Um posicionamento em que o robô R é disjunto dos obstáculos é dito um posicionamento livre. Caso contrário, é dito um posicionamento proibido.

Além disso, para especificar qual o posicionamento do robô R no espaço físico, alguns parâmetros reais são necessários, por exemplo, as coordenadas da translação aplicada a R e os ângulos das rotações aplicadas a R em relação aos eixos.

Definição 1.7 (k graus de liberdade)**.** O número k de parâmetros reais que determinam o posicionamento do robô R.

Para trabalharmos no espaço físico, precisamos saber quais as posições que o robô R pode assumir de tal forma a evitar colisão com os obstáculos. Como cada posicionamento é definido por k parâmetros reais, nós trabalharemos no que chamamos de espaço de configuração.

Definição 1.8 (Espaço de configuração C)**.** Uma porção do espaço em k-dimensões é dito um espaço de configuração C, onde cada ponto desse espaço é um posicionamento distinto do robô R.

Na definição do espaço de configuração C, o uso do k é proposital, referenciando o número de graus de liberdade que representa todos os possíveis posicionamentos do robô R. Todo o conjunto de posicionamentos do robô R em que há colisão com algum obstáculo P, é chamado de obstáculo expandido P^*.

Definição 1.9 (Obstáculo expandido P^*)**.** Para um obstáculo P, o obstáculo expandido P^* é o espaço de configuração consistindo do posicionamento proibido do robô R em colisão com P.

Chamamos de espaço de configuração proibido a união dos obstáculos expandidos.

Definição 1.10 (Espaço de configuração proibida C_P)**.** O subconjunto C_P do espaço de configuração C consistindo de posicionamentos proibidos do robô R: $\bigcup_P P^*$.

Em contra-partida, o restante do espaço de configuração que não é proibido, é dito espaço de configuração livre.

Definição 1.11 (Espaço de configuração livre C_F)**.** O subconjunto C_F do espaço de configuração C consistindo de posicionamentos livres do robô R: $C_F = C \setminus \bigcup_P P^*$.

Agora, o problema consiste em encontrar um movimento contínuo livre de colisão conectando duas posições ou dizer que o movimento não existe, sendo nosso objetivo encontrar um movimento livre de colisão.

Definição 1.12 (Movimento livre de colisão)**.** Um caminho contido em um espaço de configuração livre C_F em que quaisquer dois posicionamentos do robô R sejam posicionamentos livres, que possam ser alcançados de um para o outro (e vice-versa) através de um caminho livre de colisão e que estejam na mesma componente conexa de C_F.

Com a terminologia básica, agora estamos aptos a apresentar a formulação mais básica do problema de movimento de pianos:

Formulação 1 (Problema do movimento de pianos)**.** *Seguem os ingredientes do problema do movimento de pianos:*

- *Um espaço físico W em que $d = 2$ ou $d = 3$.*
- *Um robô R em que $d = 2$ ou $d = 3$.*
- *Obstáculos P_i, $i = 1, \ldots, n$.*
- *Um espaço de configuração C.*
- *Uma configuração $q_{ini} \in C_F$, dita configuração inicial.*
- *Uma configuração $q_{fim} \in C_F$, dita configuração final.*

- *Um algoritmo que retorna em tempo finito um movimento contínuo livre de colisão $\tau : [0,1] \rightarrow C_F$, tal que $\tau(0) = q_{ini}$ e $\tau(1) = q_{fim}$ caso haja solução. Caso não haja solução, deve retornar que um caminho não existe.*

Vários exemplos de definições e componentes do problema do movimento de pianos podem ser encontrados na Figura 1.1.

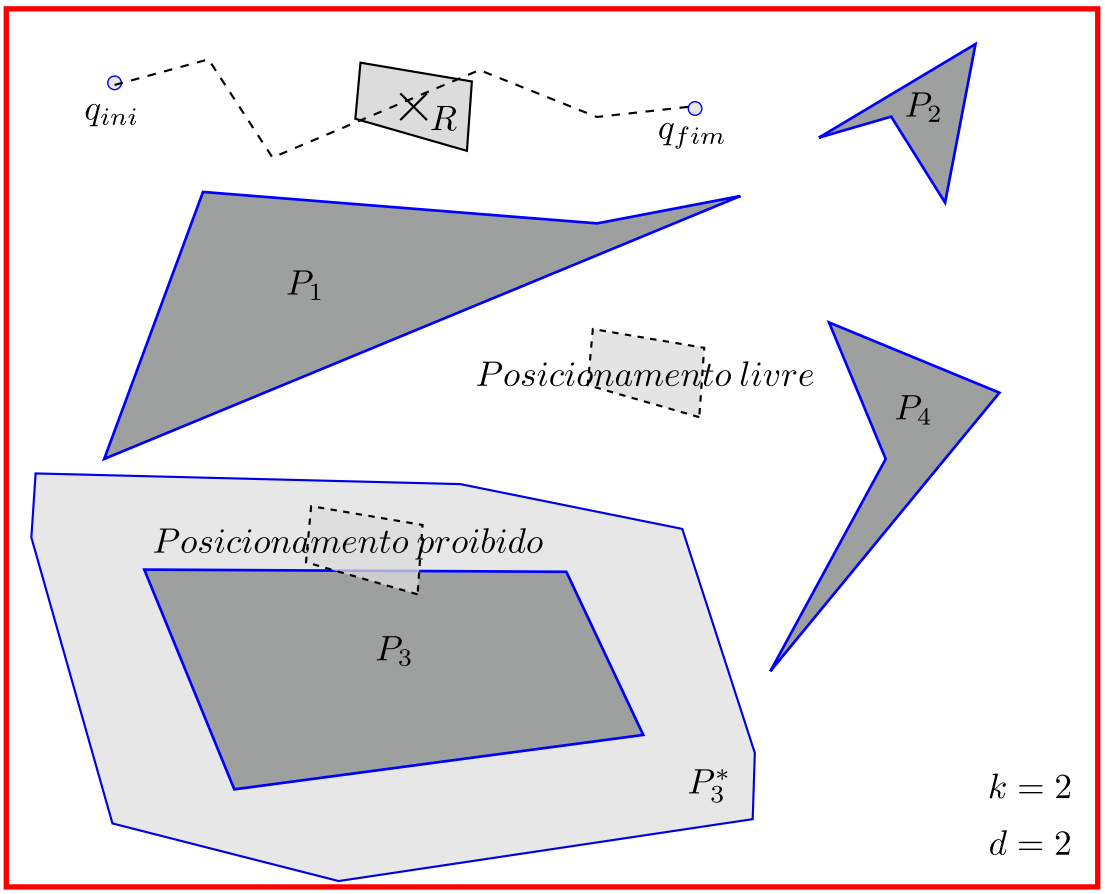


Figura 1.1: Exemplo de espaço físico W

Algumas definições que envolvem o espaço de configuração C podem ser encontradas na Figura 1.2, que foi obtido a partir do espaço físico da Figura 1.1. Estão ilustrados o espaço de configuração proibida C_P e o espaço de configuração livre C_F, no caso em que há apenas a translação.

Na robótica, o movimento planejado é fundamental e já se encontra formulado de diversas formas, inclusive utilizando vários robôs, que se comunicam por juntas, não se comunicam ou têm uma comunicação sem a necessidade de juntas.

Modificando o terceiro item da Formulação 1, obtemos uma das possíveis variações mencionadas:

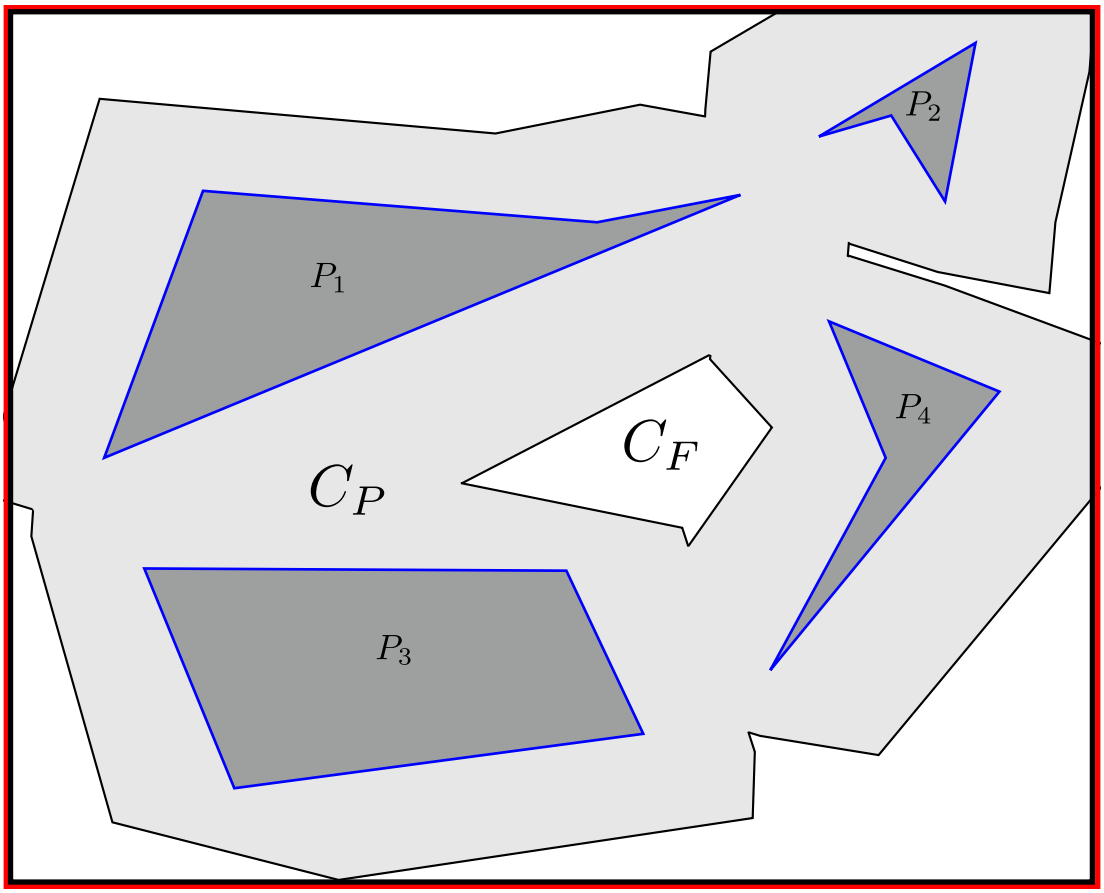


Figura 1.2: Exemplo de espaço de configuração C

Formulação 2 (Variação do problema do movimento de pianos)**.** *Seguem os ingredientes de uma variação do problema do movimento de pianos:*

- *Um espaço físico W em que $d = 2$ ou $d = 3$.*
- *Um robô R em que $d = 2$ ou $d = 3$ e formado pela ligação de m juntas, $A_1, A_2, \ldots, A_m$, onde cada parâmetro que define a posição de uma junta, aumenta em 1 o grau de liberdade.*
- *Obstáculos P_i, $i = 1, \ldots, n$.*
- *Um espaço de configuração C.*
- *Uma configuração $q_{ini} \in C_F$, dita configuração inicial.*
- *Uma configuração $q_{fim} \in C_F$, dita configuração final.*
- *Um algoritmo que retorna em tempo finito um movimento contínuo livre de colisão $\tau : [0, 1] \to C_F$, tal que $\tau(0) = q_{ini}$ e $\tau(1) = q_{fim}$ caso haja solução.*

Caso não haja solução, deve retornar que um caminho não existe.

1.2 Estruturas de Dados

Esta seção está focada nas estruturas de dados que serão necessárias ao longo do texto, pois precisamos de estruturas que preprocessam o cenário de forma a obter informações de forma eficiente e que representem o cenário.

1.2.1 Mapa Trapezoidal

Suponha que queiramos preprocessar n obstáculos bidimensionais para consultas de localização de pontos. Para tanto, vamos dividir o espaço de configuração em células, utilizando um mapa trapezoidal.

A construção do mapa trapezoidal é bem simples. Para cada vértice que pertence ao espaço de configuração proibido C_P, trace raios para cima e para baixo através do espaço de configuração livre C_F até que haja colisão com algum elemento do espaço de configuração proibido C_P.

O mapa trapezoidal [3] tem este nome porque particiona o espaço (no nosso caso, o espaço de configuração livre C_F) em um conjunto de células d-dimensionais, com $d = \{1, 2\}$, onde uma célula 2-dimensional é um trapézio ou um triângulo, sendo o triângulo denotado por trapézio degenerado, como se fosse um trapézio onde uma das arestas verticais tem tamanho infinitesimal.

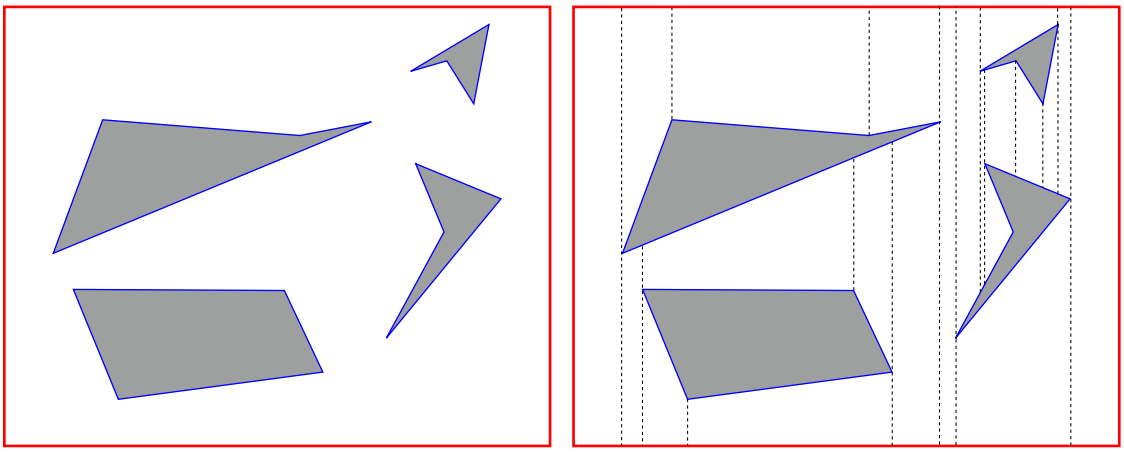
Um exemplo de um mapa trapezoidal construído sobre um espaço de configuração pode ser visualizado na Figura 1.3(b), obtido a partir da Figura 1.3(a).

Observe que na decomposição trapezoidal, obtemos 4 diferentes possibilidades de se traçar os prolongamentos:

1. Um vértice só pode ter prolongamentos para cima. Veja um exemplo na Figura 1.4(a).

2. Um vértice só pode ter prolongamentos para baixo. Veja um exemplo na Figura 1.4(b).

3. Um vértice tem ambos os prolongamentos. Veja um exemplo na Figura 1.4(c).



(a) Espaço de configuração C

(b) Mapa trapezoidal do espaço de configuração C

Figura 1.3: Mapa trapezoidal de um espaço de configuração

4. Um vértice não tem nenhum prolongamento. Veja um exemplo na Figura 1.4(d).

1.2.2 Lista de arestas duplamente conexas (DCEL)

Os espaços de configuração bidimensionais são subdivisões induzidas por grafos, onde um grafo G é uma estrutura contida no espaço de configuração C. Vamos definir os conceitos de vértice, aresta e face.

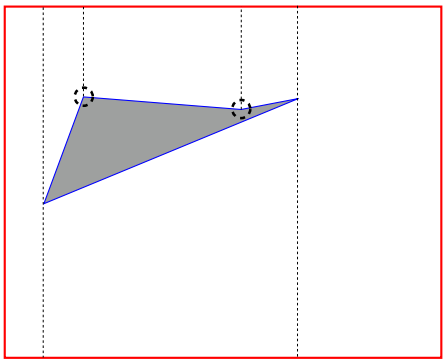
Definição 1.13 (Vértice)**.** Um nó do grafo contido no espaço de configuração é dito um vértice.

Definição 1.14 (Aresta)**.** Um arco do grafo contido no espaço de configuração é dito uma aresta.

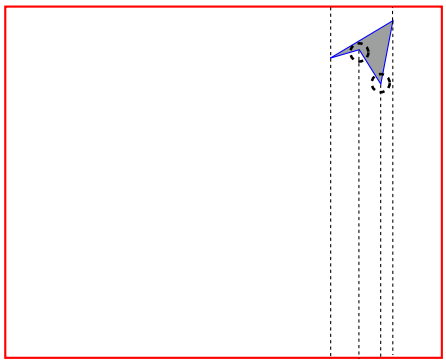
Definição 1.15 (Face)**.** Um subconjunto conexo maximal de um plano que não contém um ponto em uma aresta ou em um vértice é dito uma face da subdivisão.

Essas definições foram apresentadas porque uma *DCEL* contém vértice, aresta e face como entidades. Além disso, para cada vértice, aresta e face do espaço de configuração C em que o grafo está contido, a *DCEL* tem um registro.

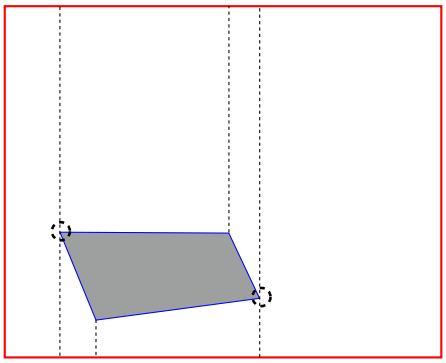
Para manter as informações topológicas e geométricas, cada registro mantém informações adicionais, de forma a garantir que estas informações sejam preservadas.



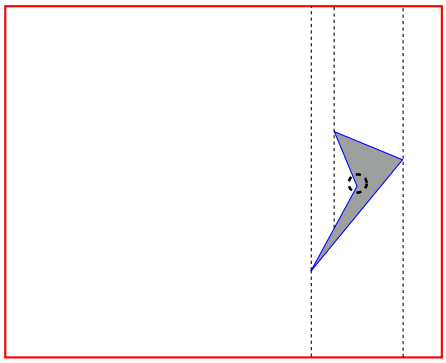
(a) Caso I: Vértice com prolongamento apenas para cima



(b) Caso II: Vértice com prolongamento apenas para baixo



(c) Caso III: Vértice com prolongamento para baixo e para cima



(d) Caso IV: Vértice sem prolongamento

Figura 1.4: Tipos possíveis de prolongamentos no mapa trapezoidal

A lista de arestas duplamente conexas [4] tem este nome porque uma aresta geralmente limita duas faces. Assim, vemos os diferentes lados de uma aresta, como uma semi-aresta cada e, além disso, para toda semi-aresta, temos um ponteiro para a próxima semi-aresta na circulação da face e outro ponteiro para a semi-aresta anterior. Antes de apresentarmos as informações armazenadas em todas as entidade, não somente as semi-arestas, uma terminologia a mais nos será útil:

Definição 1.16 (Incidência)**.** Quando um vértice é um ponto final de uma aresta, dizemos que o vértice e a aresta são incidentes. Da mesma forma que uma face e uma aresta ou vértice da fronteira da face são incidentes.

Agora seguem as entidades:

- Vértice, digamos v.

  1. Coordenadas de v;

2. Ponteiro para uma aresta que tem v como origem.

- Face, digamos f.

  1. Ponteiro para uma aresta que seja incidente. Ponteiro *nulo* caso f seja ilimitada.
  2. Lista com um ponteiro para uma semi-aresta de cada buraco de f.

- Semi-aresta, digamos e.

  1. Ponteiro para o vértice que seja sua origem.
  2. Ponteiro para a outra semi-aresta que, junto com e, constituem a aresta.
  3. Ponteiro para a face à qual e seja incidente.
  4. Ponteiro para a próxima semi-aresta que seja incidente à mesma face de e.
  5. Ponteiro para a anterior semi-aresta que seja incidente à mesma face de e.

Um exemplo de *DCEL* pode ser encontrado na Figura 1.5, onde os ponteiros das semi-arestas para a face f_2 foram omitidos para evitar a sobrecarga de setas na figura.

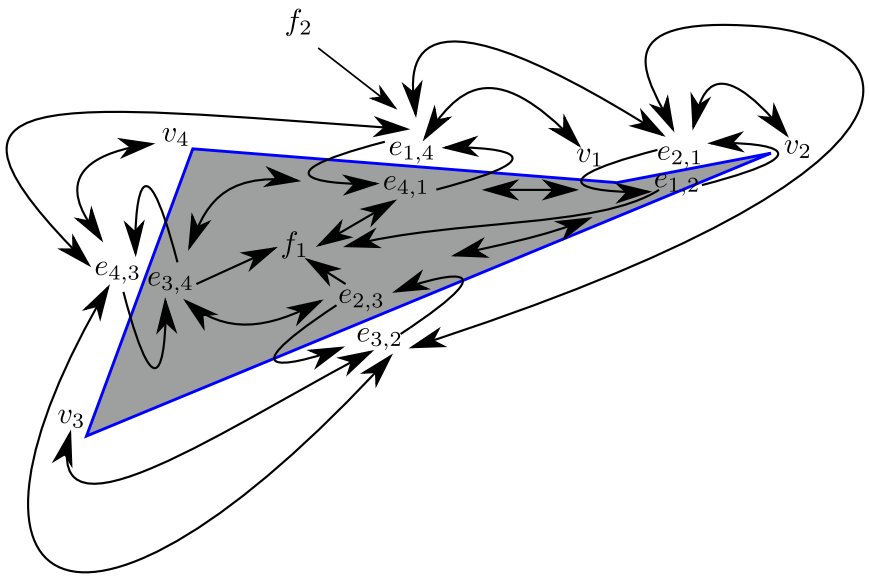


Figura 1.5: Exemplo de *DCEL*, com ponteiros das semi-arestas para face f_2 omitidos

1.2.3 Quadtree

A quadtree [5] é uma estrutura de dados muito simples e uma das mais poderosas para utilizar em problemas geométricos. Apresentaremos nesta seção uma variação de quadtree que utilizaremos no restante do texto.

Suponha que queremos preprocessar n obstáculos d-dimensionais para consultas de localização de pontos. Para tanto, construa uma árvore T, onde a raiz corresponde ao espaço físico limitado por uma caixa envolvente que contenha todos os obstáculos. Além disso, cada nó interno da árvore T é um hipercubo d-dimensional que tem 2^d filhos, onde os 2^d filhos correspondem aos 2^d hipercubos de mesmo tamanho obtidos por um corte perpendicular a cada um dos d eixos.

A construção da quadtree é recursiva e iniciada a partir da raiz, onde a cada nó v da árvore é verificado se há uma quantidade maior que uma constante c de poliedros que intersectam. Caso haja, criamos os filhos de v e, para cada filho do nó v, será atribuída uma lista de poliedros que intersectam o respectivo filho. Realizamos o procedimento recursivamente nos filhos até que não haja mais do que c interseções entre o nó e sua lista de poliedros. Esta lista, chamada de lista de conflito, é armazenada apenas em cada nó folha.

Um exemplo de quadtree bidimensional pode ser visualizado na Figura 1.6(a).

O leitor interessado pode encontrar bastante informação e aplicações de quadtrees em SAMET [6].

1.2.4 Quadtree compactada

Note na Figura 1.6(a) que os vértices b e c da representação hierárquica tem apenas um nó filho com poliedros intersectando. O único nó filho que esses vértices têm vão ter os mesmos poliedros intersectando que os pais tinham. Logo, esses vértices podem ser eliminados. Essa é a idéia da quadtree compactada [7].

Se tivermos uma sequência de arestas na representação hierárquica que propague essa propriedade de ter apenas um filho com poliedros intersectando, podemos trocá-las por apenas uma única aresta. Para não perder a informação da localização do hipercubo que um nó, digamos v, representa, vamos armazenar as coordenadas do hipercubo e o nível em que se encontra na representação hierárquica.

O processo é recursivo e aplica-se a substituição de cada sequências de arestas

que mantêm a propriedade de ter apenas um filho com poliedros intersectando, por arestas ligada por 2 vértices, sendo o pai o primeiro vértice da sequência e o filho sendo o último vértice da sequência. A árvore resultante é a quadtree compactada.

Um exemplo de quadtree compactada bidimensional pode ser visualizado na Figura 1.6(b), obtida a partir da quadtree bidimensional da Figura 1.6(a).

1.2.5 Hierarquia de Separadores

A hierarquia de separadores é uma árvore contendo separadores, onde um separador é um nó v obtido de uma árvore T com n nós tal que a remoção v de T produz uma floresta F, onde toda árvore em F tem no máximo $\left\lceil \frac{n}{2} \right\rceil$ nós.

Utilizaremos a hierarquia de separadores obtido apartir de sucessivamente extrairmos separadores, recursivamente, da quadtree até que não tenhamos mais vértices na quadtree. Como a quadtree é uma árvore e toda árvore tem um separador e pode ser computado em tempo $O(n)$ [8], temos que um separador pode ser obtido da quadtree em tamanho $O(n)$.

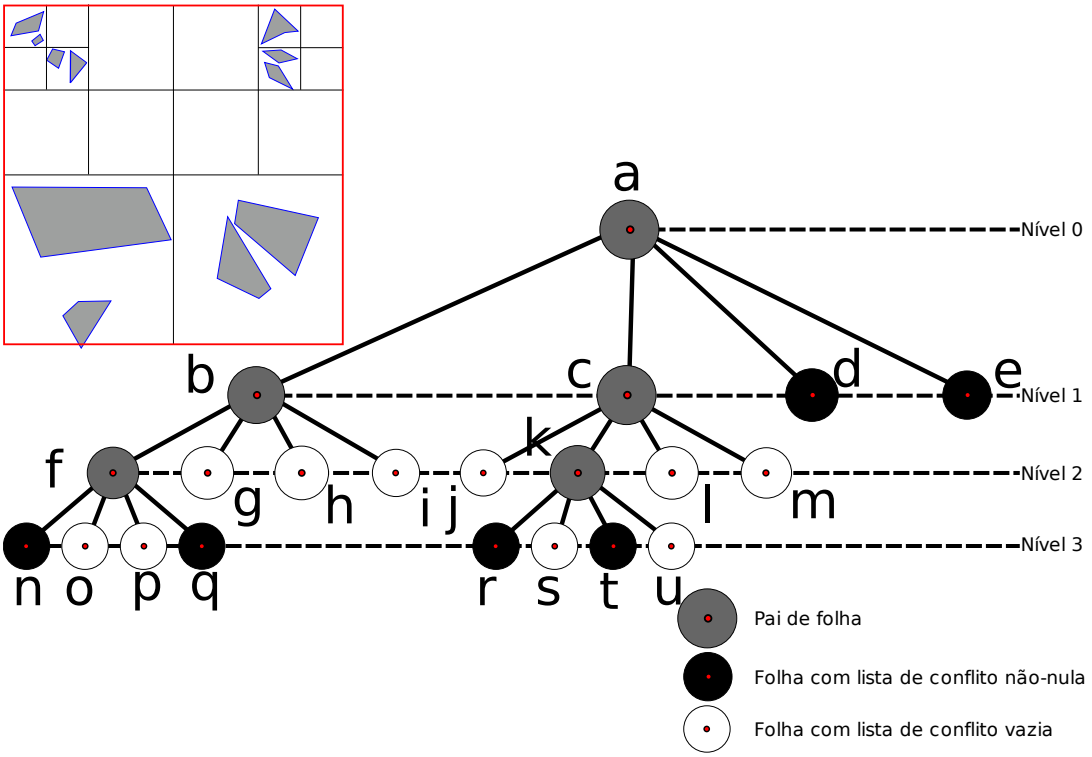
O processo inicia na raiz e repetidamente segue para a maior sub-árvore, até que o separador seja alcançado. A hierarquia de separadores T' é construída atribuindo à raíz o separador v de uma quadtree, digamos T. Os filhos de v serão as raízes da hierarquia de separadores das árvores obtidas por remover v de T. A hierarquia de separadores T' tem tamanho $O(n)$, altura $O(\log n)$ e pode ser construída em tempo $O(n \log n)$ [8].

Um exemplo de hierarquia de separadores pode ser visualizado na Figura 1.7.

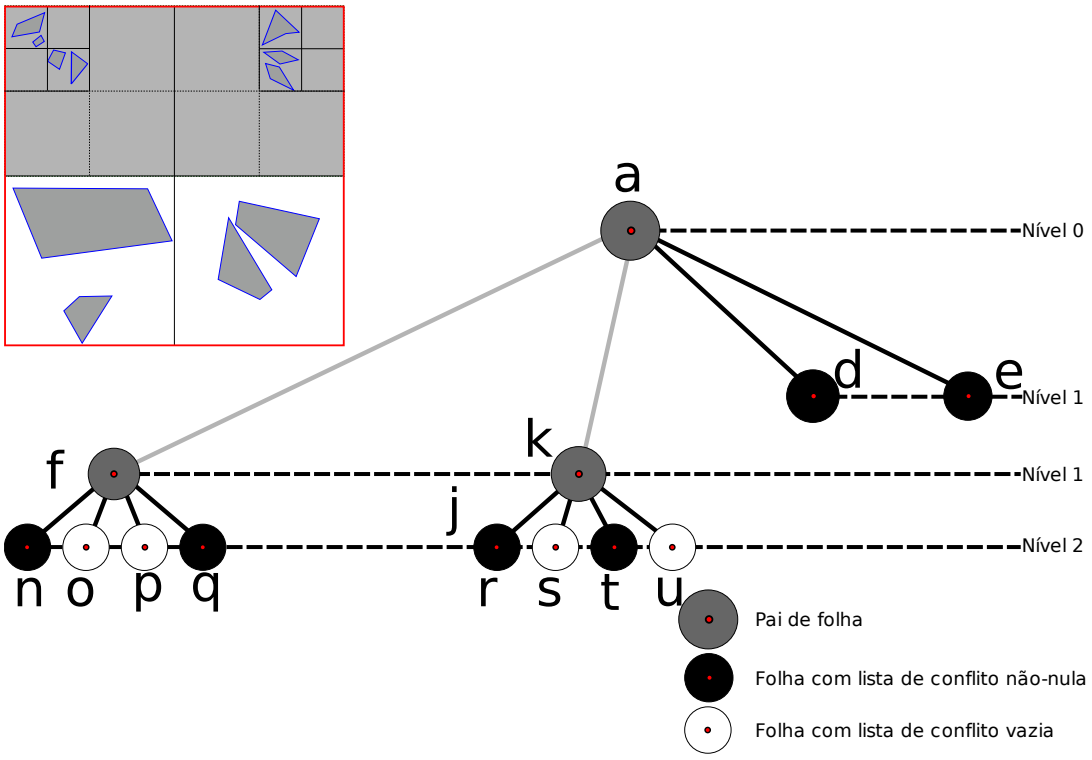
1.3 Duas principais filosofias

1.3.1 Construção explícita

Conceitualmente, o problema é construir de forma eficiente e/ou simples uma fronteira explícita ou uma representação sólida do espaço de configuração livre C_F e/ou espaço de configuração proibida C_P. Principalmente em dimensões inferiores, são bastante utilizadas e, na maioria dos casos, os algoritmos que utilizam essa filosofia retornam a solução correta de forma exata.



(a) Quadtree e sua representação hierárquica



(b) Quadtree compactada e sua representação hierárquica

Figura 1.6: Quadtree e sua representação hierárquica antes e depois da compactação

Várias soluções utilizando construção explícita são simples e elegantes. Entre-

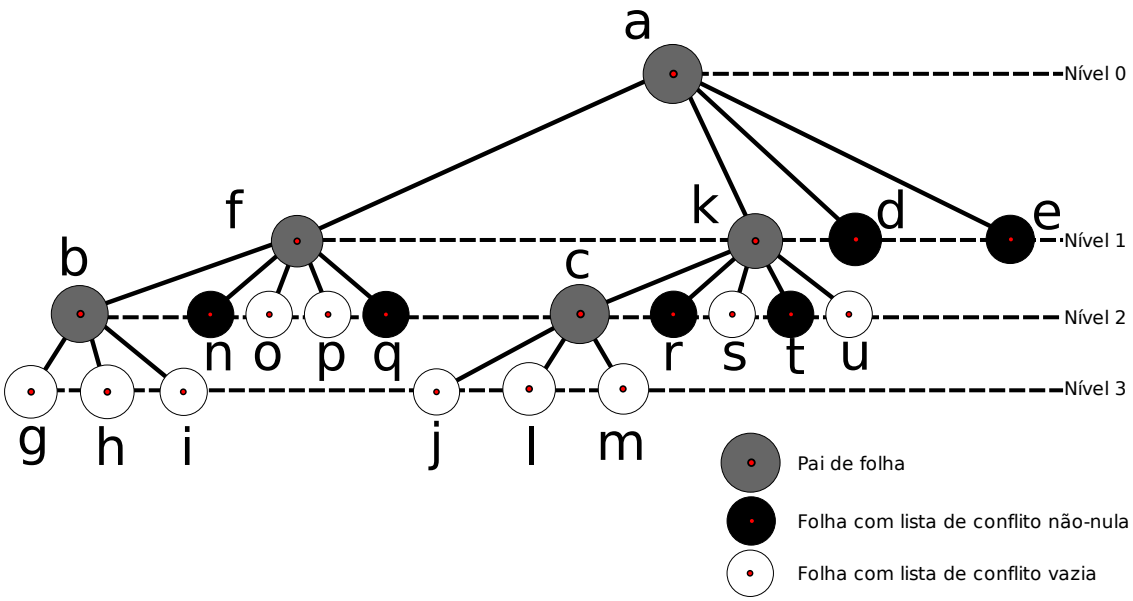


Figura 1.7: Exemplo de hierarquia de separadores obtida a partir da quadtree da Figura 1.6(a)

tanto, dezenas de anos foram necessários para que elas fossem descobertas. Quando ainda formulado como o problema de movimento de pianos, as soluções eram complexas e extremamente não-triviais, além de menos gerais, resolvendo casos bem específicos de planejamento de movimento [9].

Apresentaremos um conceito chamado de soma de Minkowski que é bastante útil na construção explícita do espaço de configuração C.

Diferença e soma de Minkowski

A soma de Minkowski é uma operação realizada em dois conjuntos, obtendo um conjunto. O resultado é a soma de cada elemento de um conjunto com todos os outros elementos do outro conjunto. Antes de introduzirmos a definição formal da soma de Minkowski, introduziremos a diferença de Minkowski, por ser definida de formas diferentes na literatura [10]. Posteriormente, mostraremos como reescrever a diferença de Minkowski utilizando a soma de Minkowski, para que não haja necessidade de utilizar um termo que pode se tornar ambíguo ao longo do texto, por suas definições diferentes na literatura. Basicamente, a diferença de Minkowski é uma operação realizada em dois conjuntos, obtendo um conjunto. O resultado é a diferença de cada elemento do primeiro conjunto com todos os outros elementos do outro conjunto.

Definição 1.17 (Diferença de Minkowski)**.** Seja $S_1 \subset \mathbb{R}^d$ e $S_2 \subset \mathbb{R}^d$. A diferença de Minkowski de S_1 por S_2, denotado por $S_1 \ominus S_2$ é:

$$S_1 \ominus S_2 = \{p - q : p \in S_1, q \in S_2\},$$

onde $p - q$ é a diferença de vetores $p \in \mathbb{R}^d$ e $q \in \mathbb{R}^d$.

Agora, definiremos formalmente a soma de Minkowski.

Definição 1.18 (Soma de Minkowski)**.** Seja $S_1 \subset \mathbb{R}^d$ e $S_2 \subset \mathbb{R}^d$. A soma de Minkowski de S_1 por S_2, denotado por $S_1 \oplus S_2$ é:

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

onde $p + q$ é a soma de vetores $p \in \mathbb{R}^d$ e $q \in \mathbb{R}^d$.

Note que podemos fazer a diferença de Minkowski a partir da soma de Minkowski, aplicando a reflexão de S_2 sobre a origem, ou seja, $S_1 \ominus S_2 = S_1 \oplus (-S_2)$, onde $-S_2 = \{-q : q \in S_2\}$.

A Figura 1.8 mostra um exemplo da soma de Minkowski, onde S_1 é um triângulo, S_2 é um quadrilátero e $S_1 \oplus S_2$ é um hexágono.

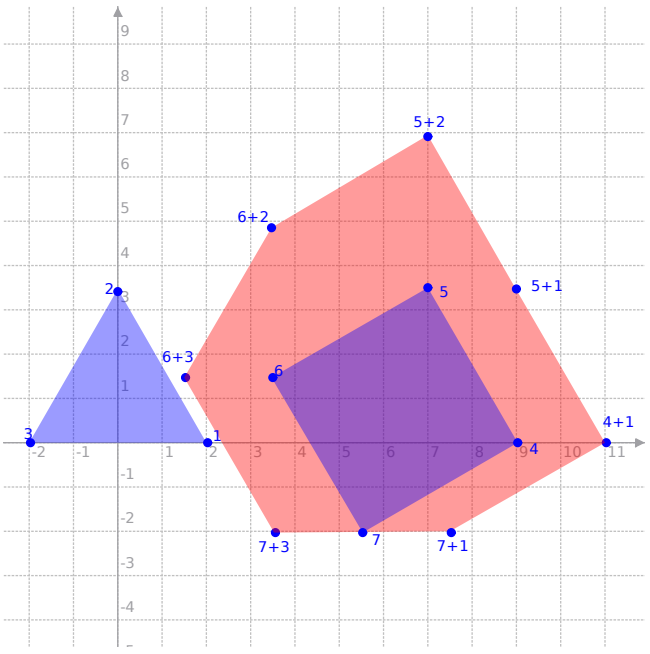


Figura 1.8: Soma de Minkowski de um triângulo e um quadrilátero

A soma de Minkowski envolve um número possivelmente infinito de somas, mas no nosso caso em que utilizaremos em polígonos, esta soma tem um limite superior

quadrático de somas em função da quantidade de vértices de cada polígono, seja convexo ou não-convexo.

1.3.2 Amostragem

A filosofia da amostragem, pode ser dividida em determinística e probabilística. Alguns métodos utilizam uma amostragem determinística do espaço de configuração C, geralmente utilizando grades. Outros métodos utilizam uma amostragem aleatória. As amostragens aleatórias têm algumas vantagens sobre a determinística, principalmente quanto à quantidade de pontos utilizados para resolver o problema de planejamento.

O espaço de configuração C para o planejamento de movimento tem uma quantidade incontável infinita de pontos. Desta forma, um algoritmo de planejamento baseado em amostragem pode considerar no máximo um número contável de amostras, para que o algoritmo retorne uma solução em tempo finito. Se o algoritmo não termina em tempo finito, a amostragem pode ser infinita contável e não há um caminho possível. Mas o que esperamos é terminar em tempo finito, considerando apenas um número finito de amostras. Ao contrário da filosofia de construção explícita, lidamos principalmente com dimensões superiores.

Entretanto, como o espaço de configuração C não é construído de forma explícita, não há como saber a forma de alguns objetos e temos que lidar com a incerteza do que pode acontecer. Esta filosofia é bem utilizada na prática e tem funcionado bem. Alguns estudos recentes tentam fazer uma análise mais profunda deste método para entender por que a técnica de amostragem tem funcionado [11]. Apresentaremos alguns destes estudos que, apesar de presentes na literatura, atualmente ainda não se encontram nos livros especializados no assunto, inclusive nos mais recentes.

1.4 Roteiro

O trabalho é organizado de acordo com o roteiro abaixo.

No Capítulo 2, apresentamos abordagens determinísticas encontradas na literatura para a construção explícita do espaço de configuração C. O foco é resolver problemas de planejamento de movimento em dimensões inferiores.

No Capítulo 3, apresentamos diversas abordagens baseadas em amostragens encontradas na literatura, onde não há uma construção explícita do espaço de configuração C ou sua construção se torna proibitiva pela alta complexidade geométrica da cena. O foco desse capítulo é a solução dos problemas de planejamento de movimento em dimensões superiores. Os algoritmos lidam com a incerteza do ambiente e sobre as formas dos objetos, pois não têm todas as informações do espaço de configuração C.

Desta forma, com os Capítulos 2 e 3, estamos aptos a resolver o problema de planejamento de movimento em dimensão arbitrária, o que sugere o título desta dissertação, podendo escolher dentre as diversas técnicas a que melhor convier de acordo com a dimensão.

No Capítulo 4, discutimos variações do problema de planejamento de movimento utilizadas na prática, em especial o caso de que para todo ponto p pertencente a um obstáculo, a vizinhança de p requer uma "alta densidade" do objeto a que ele pertence, evitando que muitos outros objetos de um certo tamanho mínimo estejam na vizinhança de p, ou seja, os obstáculos não podem estar tão aglutinados e cada objeto não pode ser longo e estreito. Propomos uma nova forma de resolver esse problema que é muito mais simples que o resultado mais atual e de melhor complexidade, utilizando técnicas do Capítulo 3.

No Capítulo 5, fazemos uma conclusão do trabalho realizado e trabalhos futuros que estudaremos e atacaremos para aprofundar em outras filosofias e variações do problema de planejamento do movimento.

Segue uma diagramação dos capítulos na Figura 1.9, indicando os pré-requisitos dos capítulos. Cada capítulo que tem uma ligação com outro capítulo disposto mais acima significa que este é pré-requisito daquele.

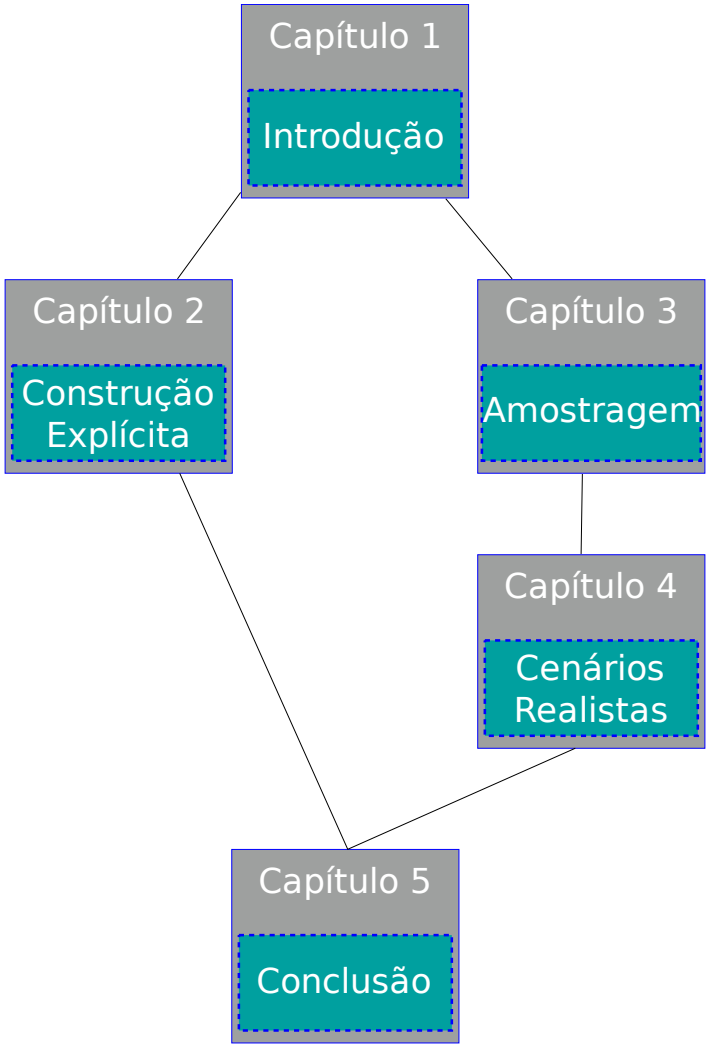


Figura 1.9: Disposição dos capítulos

Capítulo 2

Construção explícita

Para resolver o problema de planejamento de robôs, precisamos de algoritmos que percorram o espaço de configuração C para computar uma sequência de configurações que possa ser traduzida em um caminho no nosso espaço físico W, que seja livre de colisão. Precisamos remover de C as configurações proibidas para obtermos o espaço de configuração livre C_F e, assim, um algoritmo de planejamento que encontre um caminho contínuo livre de colisão de uma configuração inicial q_{ini} a uma configuração final q_{fim}. Todos esses ingredientes, aliados a um algoritmo que resolva este problema de uma forma exata é o objetivo deste capítulo, onde construímos explicitamente o espaço de configuração livre C_F para que o algoritmo tenha todas as informações necessárias.

Uma dúvida que podemos ter é como identificar as configurações proibidas para que as removamos do espaço de configuração C:

Questionamento 1. Dados um robô R, um obstáculo P e um posicionamento p qualquer, como podemos descobrir se R está colidindo com P?

Para facilitar a notação, apresentamos a seguinte definição:

Definição 2.1 (Robô $R(t)$)**.** Um Robô R em um posicionamento $t = (t_1, t_2, \ldots, t_k)$ é denotado por $R(t)$. Note que um Robô $R(0)$ não sofreu nenhuma transformação e, por isso, denotaremos apenas por R.

Agora, precisamos saber como responder o Questionamento 1. Suponha que $R(t)$ colide com P. Seja $p \in R(t) \bigcap P$ um ponto da interseção. Como $p - t \in R(0) = R$ e

$p \in P$, temos:

$$\begin{aligned}(t_1, t_2, \ldots, t_k) &= t \\ &= t + p - p \\ &\in \{t + p - p\} \\ &\subseteq \{p - \underbrace{(p - t)}_{r} : p \in P, r \in R\}.\end{aligned}$$

Em contra-partida, seja $(t_1, t_2, \ldots, t_k) \in \{p - r : p \in P, r \in R\}$. Logo, existe $p' = (p'_1, p'_2, \ldots, p'_k) \in P$ e $r' = (r'_1, r'_2, \ldots, r'_k) \in R$, tal que $(t_1, t_2, \ldots, t_k) = (p'_1 - r'_1, p'_2 - r'_2, \ldots, p'_k - r'_k)$, ou seja, $p'_1 = r'_1 + t_1$, $p'_2 = r'_2 + t_2$, $\ldots$, $p'_k = r'_k + t_k$, o que implica $p' \in R(t)$.

Note que $\{p - r : p \in P, r \in R\}$ é a diferença de Minkowski $P \ominus R$, introduzida na Definição 1.17, Página 15. Desta forma, concluímos que:

Teorema 2.1. *$R(t)$ intersecta $P \Leftrightarrow (t_1, t_2, \ldots, t_k) \in P \ominus R$*

Podemos redefinir o espaço de configuração proibida C_P, onde $P = \bigcup_i P_i$, da seguinte forma:

$$\begin{aligned}C_P &= \left\{q \in C \mid \exists i, P_i \bigcap R(q) \neq \emptyset\right\} \\ &= \{q \in C \mid \exists i, q \in P_i \ominus R\} \\ &= \{q \in C \mid \exists i, q \in P_i \oplus (-R)\}\end{aligned}$$

Na Figura 2.1 temos um exemplo unidimensional do cálculo de C_P usando diferença de Minkowski.

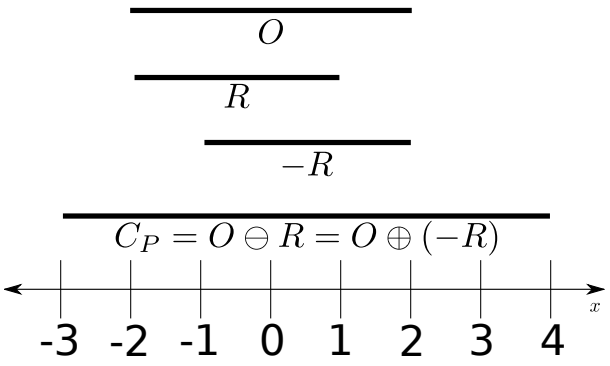


Figura 2.1: Exemplo de computação do espaço de configuração proibida C_P

Neste capítulo, precisamos de algumas terminologias que se farão necessárias no decorrer do texto, principalmente quanto à forma dos objetos, já que estamos tratando aqui da construção explícita do espaço de configuração C. Logo, se torna importante a definição do que está dentro ou que está na fronteira de um objeto. Todo elemento que não estiver dentro ou na fronteira do objeto está fora.

Precisamos saber quando considerar se um ponto está dentro do objeto.

Definição 2.2 (Ponto interior)**.** Sejam M um espaço métrico com métrica ρ, X um subconjunto de M e x um ponto de X. Dizemos que x é um ponto interior de X se $\exists r > 0, \forall y$ com distância $\rho(x, y) < r$, onde $y \in X$.

Após saber identificar quando um ponto pertence ao interior de um objeto, ao reunir todos os pontos que atendem a essa propriedade, temos o interior deste objeto.

Definição 2.3 (Interior de S)**.** O interior de um conjunto S, denotado por $int(S)$, é o conjunto de todos os pontos interiores de S.

O que não faz parte do interior do objeto, mas continua fazendo parte do objeto, chamaremos de fronteira.

Definição 2.4 (Fronteira de S)**.** A fronteira de um conjunto S, denotado por ∂S, é o conjunto de todos os pontos de S que não sejam pontos interiores, ou seja:

$$\partial S = \{q \mid q \in S \wedge q \notin int(S)\}.$$

Agora, iniciaremos outra série de definições importantes que culmina em um objeto de nosso interesse particular. Primeiro, precisamos definir um mapeamento de um intervalo em um espaço, que denotamos por curva, formalmente definida abaixo.

Definição 2.5 (Curva γ)**.** Seja $I = [a, b]$, onde $a, b \in \mathbb{R}$. Uma curva γ é um mapeamento contínuo $\gamma : I \to X$, onde X é um espaço métrico.

Em particular, temos interesse quando a curva é definida utilizando um domínio em que as extremidades do intervalo são iguais na imagem do mapeamento.

Definição 2.6 (Curva fechada)**.** Uma curva γ em um domínio $I = [a, b]$ é fechada se $\gamma(a) = \gamma(b)$.

O objeto de nosso particular interesse é o polígono, uma curva definida por linhas fechadas em um plano.

Definição 2.7 (Polígono)**.** Um polígono é uma figura geométrica plana limitado, cuja fronteira é. uma linha poligonal fechada.

O restante do capítulo é organizado da seguinte forma. Na Seção 2.1, falaremos sobre um algoritmo para resolver o problema de translação em um espaço físico bidimensional, cujo material é baseado em DE BERG *et al.* [10], Capítulo 13. Na Seção 2.2, será feita uma breve indicação para o leitor mais interessado em resolver o problema de dimensão arbitrária utilizando construção explícita do espaço de configuração C. Sendo uma das partes mais importantes deste texto, a construção explícita do espaço de configuração C não é trivial. Se em dimensões inferiores a construção já é complicada, em dimensão arbitrária precisaríamos de muitos elementos, o que fugiria um pouco do objetivo da dissertação, pois daria uma ênfase muito maior à construção explícita e não ao movimento planejado de robôs como um todo e suas principais técnicas.

2.1 Movimento planejado $2D$

Nesta seção supomos que nosso Robô R apenas translada e os obstáculos são disjuntos. Agora, com a soma de Minkowski, sabemos responder o Questionamento 1, porém precisamos aprender a computar eficientemente a soma de Minkowski. Um estudo sobre soma de Minkowski pode ser encontrado em JAVGAL [12].

2.1.1 Soma de Minkowski

Surge uma dúvida interessante:

Questionamento 2. A soma de Minkowski é comutativa?

Lema 2.2. *A soma de Minkowski é comutativa:*

$$P \oplus Q = Q \oplus P$$

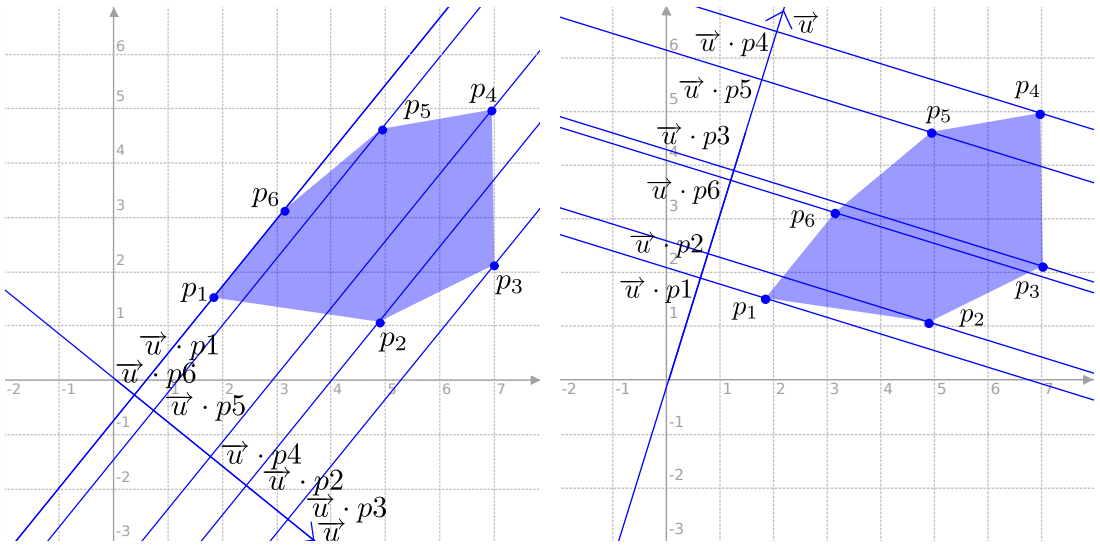
Prova. Trivial, pois a soma de vetores é comutativa ($p + q = q + p$). □

Naturalmente, surgem outras dúvidas, tal como calcular esta soma computacionalmente, afinal estamos tratando de uma definição que envolve uma operação da soma de todos os pontos de um conjunto possivelmente infinito com todos os pontos do outro conjunto também possivelmente infinito, ou seja, uma soma possivelmente infinita de pontos. Por outro lado, estamos lidando com polígonos que podemos definir a partir de um número finito de vértices. É a partir disto que podemos alcançar alguma forma finita de calcular a soma de Minkowski. Começaremos pela introdução da seguinte notação:

Definição 2.8 (Ponto extremo)**.** Seja uma direção dada por um vetor $\overrightarrow{u}$ e $p_1, \ldots, p_n$ os vértices do polígono P. Um vértice $p_i, 1 \leq i \leq n$, é extremo na direção de $\overrightarrow{u}$ se para to vértice p_j, $1 \leq j \leq n$,

$$\overrightarrow{u} \cdot p_j \leq \overrightarrow{u} \cdot p_i,$$

onde $\cdot$ é o produto escalar.



(a) Ponto extremo p_3 em uma direção

(b) Ponto extremo p_4 em outra direção

Figura 2.2: Pontos extremos de um mesmo polígono

Para ilustrar bem a Definição 2.8, veja a Figura 2.2. O polígono está sendo projetado na direção da reta. O que será que podemos aproveitar desse conceito de ponto extremo?

Lema 2.3. *Sejam P e Q dois polígonos e $PQ = P \oplus Q$. Um ponto extremo na direção $\overrightarrow{d}$ em PQ é a soma de pontos extremos na direção $\overrightarrow{d}$ de P e Q.*

Prova. Suponha, por absurdo, que o ponto extremo na direção $\overrightarrow{d}$ em PQ não necessariamente é a soma de pontos extremos na direção $\overrightarrow{d}$ em P e Q. Seja o ponto pq, soma de $p \in P$ e $q \in Q$, extremo na direção $\overrightarrow{d}$ em PQ. Sem perda de generalidade (pois a soma de Minkowski é comutativa, pelo Lema 2.2), suponha que o ponto p não é extremo na direção $\overrightarrow{d}$, afinal pelo menos um dos dois pontos não é extremo na direção $\overrightarrow{d}$. Seja q um ponto de Q e p' um ponto de P que seja extremo na direção $\overrightarrow{d}$. Sabemos que $p' + q \in PQ$, pela Definição 1.18 *(soma de Minkowski)*, e que $\overrightarrow{d} \cdot (p' + q) > \overrightarrow{d} \cdot (p + q)$, pois $\overrightarrow{d} \cdot (p') > \overrightarrow{d} \cdot p$, o que contradiz o fato de que $p + q$ é ponto extremo de PQ na direção $\overrightarrow{d}$. □

A soma de Minkowski de dois polígonos convexos resulta sempre em um polígono convexo? Se for o caso, isto aliado à propriedade anterior, garantiria que precisamos apenas computar os vértices que são únicos em relação a ser um ponto extremo em uma dada direção, pois qualquer outro ponto do conjunto é obtido através da combinação linear desses vértices. Em suma, bastaria passear pelo par de vértices em que cada vértice do par é o ponto extremo de cada polígono em uma mesma direção. Este passeio percorreria os polígonos no sentido anti-horário. Segue o resultado que almejamos:

Teorema 2.4. *Sejam P e Q dois polígonos convexos com n e m arestas, respectivamente, $PQ = P \oplus Q$ e k o número de pares distintos de arestas paralelas de P e Q na mesma direção. Então a soma de Minkowski PQ é um polígono convexo com $|E| = n + m - k$, onde E é o conjunto de arestas de PQ.*

Prova. Para provar que a soma de Minkowski de dois polígonos convexos P, Q é um polígono convexo, precisamos mostrar que um segmento de reta ligando dois pontos de PQ está inteiramente contido em PQ. Sejam esses dois pontos $pq = p + q$ e $p'q' = p' + q'$. A equação paramétrica do segmento de reta com extremidades nos pontos pq e $p'q'$ é dada por:

$$z(t) \;=\; pq + t(p'q' - pq),$$

com $0 < t < 1$. Reescrevendo, temos:

$$\begin{aligned} z(t) &= p + q + tp' + q't - tp - tq \\ &= \underbrace{p + t(p' - p)}_{p(t)} + \underbrace{q + t(q' - q)}_{q(t)} \end{aligned}$$

Note que p_t é a equação paramétrica ligando p e p' e $q(t)$ é a equação paramétrica ligando q e q'. Sabemos que P e Q são convexos e, portanto, qualquer segmento de reta definido por dois pontos de um mesmo conjunto está inteiramente contido no mesmo conjunto, logo $p(t) \in P$ e $q(t) \in Q$. Isso implica que $z(t) \in P \oplus Q$, pela Definição 1.18 *(soma de Minkowski)*. Portanto, todo o segmento de reta que liga os pontos pq e $p'q'$ também pertencem a PQ.

Considere uma aresta extrema na direção de uma normal externa em PQ. Pelo Lema 2.3, sabemos que P e/ou Q têm uma aresta extrema nesta direção. Se ambos têm aresta extrema nesta direção, elas gerarão uma aresta apenas (veja a Figura 2.3 em que a normal $\overrightarrow{n_1}$ da aresta e_1 e a normal $\overrightarrow{n_2}$ da aresta e_4) coincidem em PQ. Logo,

$$|E| = m + n - k.$$

□

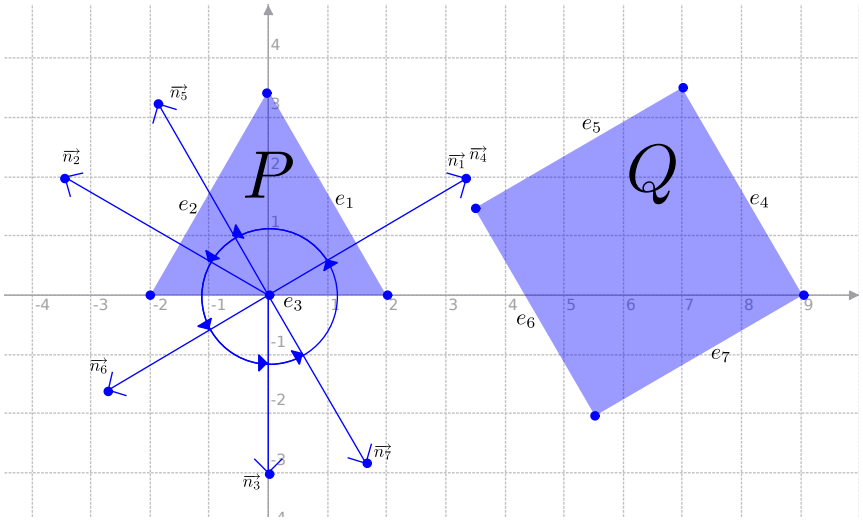


Figura 2.3: Cada normal distinta das arestas de P e Q define uma aresta em $P \oplus Q$.

Corolário. *Sejam P e Q dois polígonos convexos com n e m arestas, respectivamente, $PQ = P \oplus Q$ e E é o conjunto de arestas de PQ. PQ é um polígono convexo com os seguintes limites sobre a cardinalidade de suas arestas:*

$$\begin{cases} n \leq |E| \leq 2n, \text{ caso } n = m \\ max(n, m) + 1 \leq |E| \leq n + m, \text{ caso contrário} \end{cases}$$

Prova. Com $n = m$, podemos ter de 0 a todas as arestas paralelas. Caso contrário, sem perda de generalidade suponha $n < m$. Desta forma, não podemos ter n arestas

paralelas, pois a soma dos angulos internos daria 180^o, mas podemos ter até $n-1$ arestas paralelas. Basta aplicar o Teorema 2.4 que o resultado segue. □

Com o resultado em mãos, delineamos o algoritmo:

Algoritmo 1: Soma de Minkowski de polígonos convexos

Entrada: P: Polígono convexo com vértices $p_1, \ldots, p_n$ ordenados em sentido anti-horário

Q: Polígono convexo com vértices $q_1, \ldots, q_m$ ordenados em sentido anti-horário

p_1 e q_1 são vértices com menor ordenada (e menor abscissa em caso de empate)

Saída: $P \oplus Q$: Soma de Minkowski de P e Q

```
1  início
2      i ⟵ 1;
3      j ⟵ 1;
4      p_{n+1} ⟵ p_1;
5      p_{n+2} ⟵ p_2;
6      q_{m+1} ⟵ q_1;
7      q_{m+2} ⟵ q_2;
8      repita
9          Adicionar p_i + q_j como vértice de P ⊕ Q;
10         se angulo(p_i p_{i+1}) < angulo(q_j q_{j+1}) então
11             i ⟵ i + 1;
12         senão
13             se angulo(p_i p_{i+1}) > angulo(q_j q_{j+1}) então
14                 j ⟵ j + 1;
15             senão
16                 i ⟵ i + 1;
17                 j ⟵ j + 1;
18     até i = n + 1 e j = m + 1 ;
19     retorna P ⊕ Q;
20 fim
```

Já que calculamos a soma de Minkowski para polígonos convexos, agora generalizaremos para qualquer tipo de polígono.

Seja apenas um dos polígonos não convexo. Mais precisamente, seja R um polígono convexo e PQ um polígono não-convexo, com n e m vértices respectivamente. Podemos construir um exemplo que sua complexidade é $O(nm)$: R é um polígono de n vértices, obtido pela metade de cima de um polígono $(2n-2)$-regular e PQ é uma coroa de m vértices com $\left\lfloor \frac{m}{2} \right\rfloor$ pontas. A soma de Minkowski tem $(n+1)\left\lfloor \frac{m}{2} \right\rfloor + 1 = O(nm)$ vértices, como pode ser verificado na Figura 2.4.

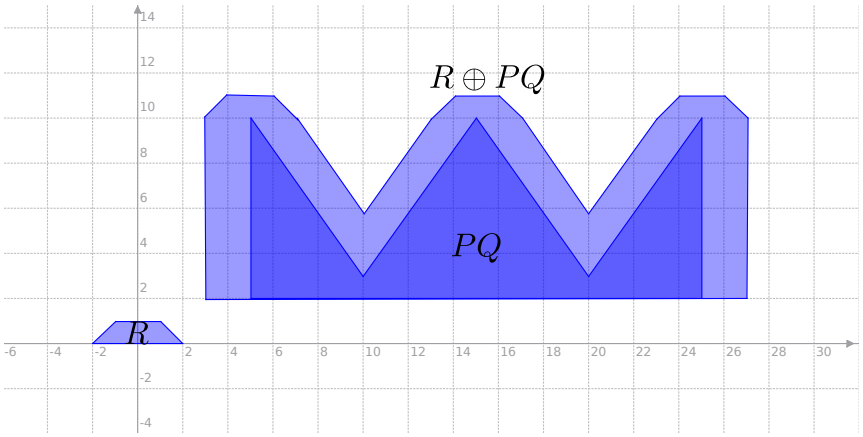


Figura 2.4: Limite inferior da soma de Minkowski de um polígono convexo com um polígono não-convexo

A idéia seria quebrar o polígono não-convexo em partes convexas, calcular a soma de Minkowski para cada pedaço e, ao final, utilizarmos a união dos resultados colando-os. Exemplificando melhor o exemplo, veja a Figura 2.5 e observe que um polígono não-convexo, digamos PQ, de quatro lados pode ser decomposto em dois triângulos, digamos P e Q, que são obviamente convexos.

Agora, suponha que $R \oplus PQ = (R \oplus P) \cup (R \oplus Q)$. Dado isto, pode-se utilizar o algoritmo de soma de Minkowski para polígonos convexos como procedimento do algoritmo para calcular $R \oplus P$ e $R \oplus Q$ e, com cada resultado, utilizar a união para nos dar a solução. Para mostrar que isso é possível, provaremos que a suposição é verdadeira

Proposição 2.5. *Sejam R, P e Q conjuntos.*

$$R \oplus (P \cup Q) = (R \oplus P) \cup (R \oplus Q)$$

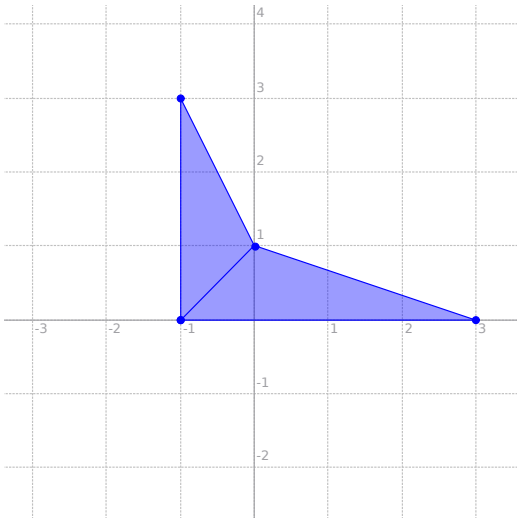


Figura 2.5: Decomposição de um polígono não convexo em dois polígonos convexos

Demonstração. ($\Rightarrow$) Seja $x \in R \oplus (P \cup Q)$. Pela Definição 1.18 *(soma de Minkowski)*, Página 15, temos: $x = a + b$, onde $a \in R$ e $b \in P \cup Q$. Temos 2 casos não-necessariamente disjuntos:

1. $b \in P$. Logo, $x \in (R \oplus Q) \subseteq (R \oplus P) \cup (R \oplus Q)$.

2. $b \in Q$. Análogo ao caso acima.

($\Leftarrow$) Seja $y \in (R \oplus P) \cup (R \oplus Q)$. Temos 2 casos não-necessariamente disjuntos:

1. $y \in (R \oplus P)$. Resultado segue da definição da soma de Minkowski que nos garante, $R \oplus P \subseteq R \oplus (P \cup Q)$.

2. $y \in (R \oplus Q)$. Análogo ao caso acima.

□

Agora, vamos generalizar esta idéia. Sabe-se que um polígono P não-convexo com n vértices pode ser triangulado em $n-2$ triângulos: $t_1, \ldots, t_{n-2}$. Tal resultado pode ser visto em DE BERG *et al.* [10], Capítulo 3. Disto, pode-se utilizar a idéia de calcular as $n-2$ somas de Minkowski de polígonos convexos e, com o resultado destas somas, computar a união entre eles. Resumindo, seja R o polígono convexo, P o polígono não convexo e $t_1, \ldots, t_{n-2}$ os triângulos obtidos através da triangulação de P. Logo,

$$R \oplus P = R \oplus \bigcup_{i=1}^{n-2} t_i \tag{2.1}$$

Com o exemplo da Figura 2.4, sabemos que o limite inferior da soma de Minkowski com apenas um polígono sendo não-convexo tem complexidade igual a $O(nm)$. Se a união tiver complexidade linear em relação ao tamanho da soma de arestas dos polígonos a serem unidos, obteremos o limite superior com complexidade a $O(nm)$ e, portanto, a complexidade da soma de Minkowski em questão tem $O(nm)$.

Como computamos a soma de Minkowski $n-2$ vezes e cada vez tem complexidade limitado por $\underbrace{3}_{lados\ do\ triângulo} + \underbrace{m}_{lados\ de\ R}$, então a complexidade desta etapa é $O(nm)$. Para que a complexidade da soma de Minkowski seja igual a $O(nm)$ precisamos mostrar que a união de polígonos convexos com nm vértices ao total tem complexidade igual a $O(nm)$. Para tanto, necessita-se de mais resultados, logo a seguir.

Pseudodiscos

Necessitamos de alguma notação que nos auxiliará a apresentar a complexidade da união de uma coleção de somas de Minkowski. Seguindo as definições do início do capítulo, temos interesse em um caso de curvas, onde não temos dois elementos diferentes do domínio cujo mapeamento leve em um único elemento, ou seja, podemos desenhar no espaço uma curva que não cruza ela mesma em nenhum ponto, podendo apenas terminar aonde começou.

Definição 2.9 (Curva simples)**.** Uma curva γ em um domínio $I = [a, b]$ é simples se é injetiva, ou seja: $\gamma(x) = \gamma(y) \rightarrow x = y, \forall x, y \in I$.

Começamos por definir o que são pseudodiscos, coleções de pseudodiscos e avanço de fronteira entre pares de pseudodiscos.

Definição 2.10 (Pseudodiscos)**.** Considere dois objetos planares p_1 e p_2, cada um limitado por uma curva fechada simples. O par p_1, p_2 é um par de pseudodiscos se suas fronteiras (∂p_1 e ∂p_2, respectivamente) se intersectam em, no máximo, dois pontos.

Quando temos uma coleção de n objetos, em que qualquer combinação desses n objetos, dois a dois, sempre produz um par de pseudodiscos, temos uma coleção de pseudodiscos.

Definição 2.11 (Coleção de pseudodiscos)**.** Uma coleção de pseudodiscos é uma coleção de objetos, cada um limitado por uma curva fechada simples, se todo par de objetos na coleção é um par de pseudodiscos.

Uma terminologia que será útil quando estivermos trabalhando com pseudodiscos é o momento em que um par de pseudodiscos se interceptam em um ponto, chamado de avanço de fronteira.

Definição 2.12 (Avanço de fronteira)**.** Sejam os polígonos P e Q. Um ponto de interseção $p \in \partial P \cap \partial Q$ é um avanço de fronteira se ∂P avança do interior de Q para o exterior de Q em p.

Para as nossas necessidades, temos polígonos disjuntos. Esta informação adicional será muito importante mais à frente. Neste momento, precisamos de um pouco mais de terminologia. Precisamos olhar para pontos extremos em várias direções. Quando um ponto extremo em uma determinada direção pertence a um polígono P, o polígono P é extremo naquela direção.

Definição 2.13 (Polígono extremo)**.** O polígono é mais extremo numa direção $\overrightarrow{n}$ do que os demais polígonos, se seu ponto extremo é mais extremo naquela direção do que os outros pontos extremos dos demais polígonos.

Podemos reunir as informações de todos os pontos extremos em todas as direções, modelando com um círculo unitário que possa ser divido em fatias, onde cada fatia significa que todos os pontos extremos naquelas direções pertencem a um mesmo polígono.

Definição 2.14 (Direções extremas)**.** Modelando o conjunto de todas as direções pelo círculo unitário, centrado na origem, dizemos que as direções extremas de uma direção $\overrightarrow{v_1}$ a uma direção $\overrightarrow{v_2}$, correspondem a todas as direções em sentido anti-horário cujos pontos extremos são de um mesmo polígono.

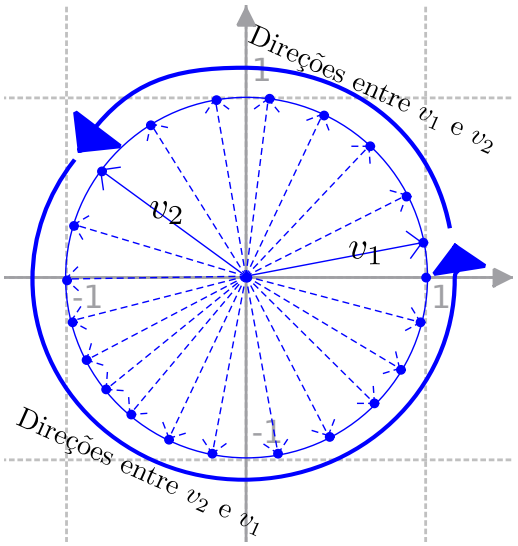


Figura 2.6: Direções extremas

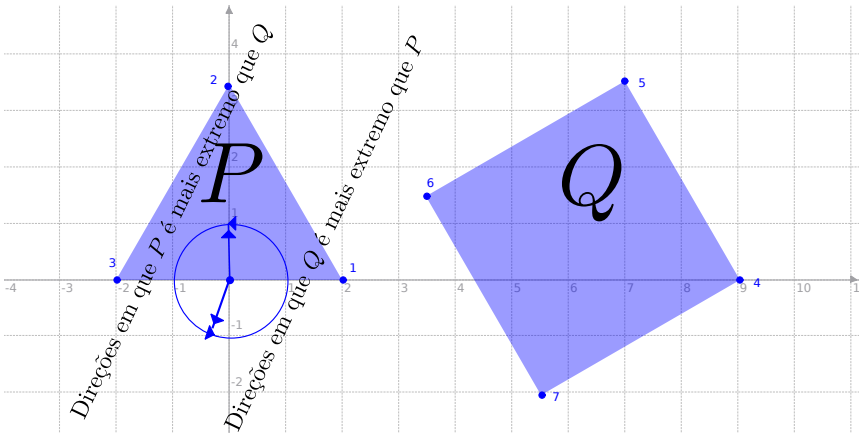


Figura 2.7: Polígonos e suas direções extremas

Pela Figura 2.6, temos uma idéia do que seria a modelagem de direções extremas . Já pela Figura 2.7, temos um exemplo prático onde traçamos a modelagem de direções extremas para polígonos disjuntos.

O resultado abaixo é a ligação entre soma de Minkowski e a propriedade de pseudodiscos, servindo para provar que a soma de Minkowski de dois polígonos convexos disjuntos são pseudodiscos.

Teorema 2.6. *Sejam P e Q polígonos convexos com interiores disjuntos. Sejam $\overrightarrow{n_1}$ e $\overrightarrow{n_2}$ direções no qual P é mais extremo que Q. Então P é mais extremo do que Q em todas as direções de $\overrightarrow{n_1}$ a $\overrightarrow{n_2}$ ou é mais extremo em todas as direções de $\overrightarrow{n_2}$ a $\overrightarrow{n_1}$.*

Demonstração. Suponha, por absurdo, que P não é mais extremo do que Q em

todas as direções de $\overrightarrow{n_1}$ a $\overrightarrow{n_2}$, nem de $\overrightarrow{n_2}$ a $\overrightarrow{n_1}$. Logo, temos pelo menos uma direção, digamos $\overrightarrow{n_3}$, de $\overrightarrow{n_1}$ a $\overrightarrow{n_2}$ em que o ponto mais extremo seja o de Q e pelo menos uma direção, digamos $\overrightarrow{n_4}$, de $\overrightarrow{n_2}$ a $\overrightarrow{n_1}$ em que o ponto mais extremo seja o de Q. Como P é convexo, todo ponto obtido pela combinação linear, digamos l_{12}, dos pontos extremos de P nas direções $\overrightarrow{n_1}$ e $\overrightarrow{n_2}$ também pertencem a P. Sem perda de generalidade, suponha que a combinação linear esteja totalmente contida no espaço definido entre as direções de $\overrightarrow{n_1}$ a $\overrightarrow{n_2}$. Por outro lado, há um ponto de Q mais extremo que P, na direção $\overrightarrow{n_4}$, entre as direções $\overrightarrow{n_2}$ e $\overrightarrow{n_1}$. Como Q é convexo, a combinação linear, digamos l_{34}, do ponto mais extremo da direção $\overrightarrow{n_4}$, com o ponto mais extremo na direção $\overrightarrow{n_3}$, também pertence a Q. Note que l_{34} intersecta l_{12}, pois um ponto extremo, digamos p, de l_{34} pertence ao espaço definido entre as direções $\overrightarrow{n_2}$ e $\overrightarrow{n_1}$ e o outro ponto extremo de l_{34} pertence ao espaço definido entre as direções $\overrightarrow{n_1}$ e $\overrightarrow{n_2}$, onde p é o ponto mais extremo na direção $\overrightarrow{n_3}$, sendo mais extremo que um ponto de l_{12}, o que implica que o ponto da intersecção pertence a P e a Q (absurdo, pois P e Q são disjuntos). Veja a Figura 2.8 para visualizar os ingredientes da prova. □

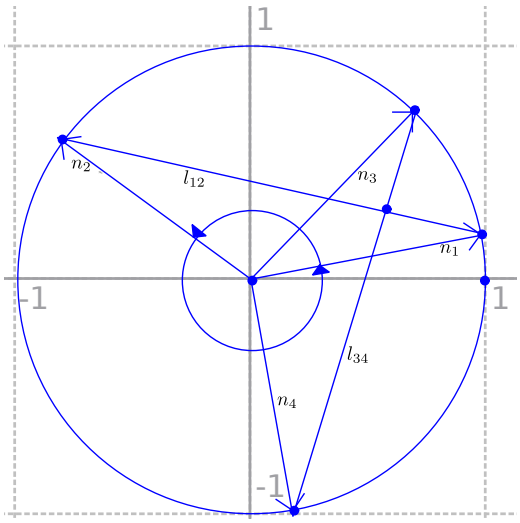


Figura 2.8: Ingredientes da demonstração do Teorema 2.6

Teorema 2.7. *Sejam R um polígono convexo e o par P, Q dois polígonos convexos com interiores disjuntos. Então as duas somas de Minkowski $R \oplus P$ e $R \oplus Q$ são pseudodiscos.*

Demonstração. Para provar que $R \oplus P$ e $R \oplus Q$ são pseudodiscos, temos dois casos:

1. $\partial(R \oplus P) \cap int(R \oplus Q)$ é conexo: suponha, por absurdo, que não é conexo. Então, percorrendo $\partial(R \oplus P)$ temos pelo menos 4 pontos, digamos a, b, c, d, que alternam (em sentido anti-horário) entre pertencer e não pertencer a $int(R \oplus Q)$. Sem perda de generalidade, suponha $a, c \in int(R \oplus Q)$ e $b, d \notin int(R \oplus Q)$. Sejam então as normais para fora do polígono $R \oplus P$, digamos $\overrightarrow{n_a}$, $\overrightarrow{n_b}$, $\overrightarrow{n_c}$, $\overrightarrow{n_d}$, nos pontos a, b, c, d respectivamente. Tem-se então que o ponto mais extremo de $\overrightarrow{n_a}, \overrightarrow{n_c}$ pertence a $R \oplus Q$ e $\overrightarrow{n_b}, \overrightarrow{n_d}$ pertence a $R \oplus P$. Mas, pelo Lema 2.3, temos que P é mais extremo que Q nas direções $\overrightarrow{n_b}, \overrightarrow{n_d}$ e Q é mais extremo que P nas direções $\overrightarrow{n_a}, \overrightarrow{n_c}$. Mas isto contradiz o Teorema 2.6, que diz que P é mais extremo que Q nas direções de $\overrightarrow{n_a}, \overrightarrow{n_c}$ ou nas direções de $\overrightarrow{n_c}, \overrightarrow{n_a}$.

2. $\partial(R \oplus Q) \cap int(R \oplus P)$ é conexo: análogo ao item acima.

□

Agora, sabemos que a soma de Minkowski com polígonos convexos e disjuntos entre si são pseudodiscos. Logo, uma coleção de somas de Minkowski é uma coleção de pseudodiscos. Tudo que precisamos saber é se a união de pseudodiscos com n arestas é linear em n. Assim, temos o que queríamos: para calcular a Equação 2.1, precisamos de uma complexidade linear na soma total de arestas para computar a união, provando nossa complexidade de $O(nm)$ para resolver a soma de Minkowski de um polígono convexo com um não-convexo.

Uma propriedade importante sobre pseudodiscos e avanço de fronteira, segue:

Lema 2.8. *Um par de pseudodiscos poligonais P, Q definem no máximo 2 avanços de fronteiras.*

Demonstração. Suponha que existem mais de 2 avanços de fronteiras. Sejam 2 avanços de fronteira consecutivos, digamos p e p', em que, sem perda de generalidade, definem uma curva que pertence a $int(P) \cap \partial Q$ ($int(p)$ é o interior do objeto P). O terceiro avanço de fronteira, digamos p'', vai começar a definir outra curva disjunta. Absurdo, pois P e Q são um par de pseudodiscos e, por isso, $int(P) \cap \partial Q$ é conexo.

□

Teorema 2.9. *Seja S uma coleção de pseudodiscos poligonais com n arestas no total. Então a complexidade da união é $O(n)$.*

Demonstração. Existem dois tipos de vértices na união das fronteiras:

- Vértices do pseudodisco: estes vértices são simplesmente os mesmos vértices que estarão presentes na fronteira da união.

- Pontos de interseção entre interiores de duas arestas de pseudodiscos: seja um vértice imaginário, digamos v, que é a interseção de uma aresta e de um pseudodisco $P \in S$ e uma aresta e' de um pseudodisco $P' \in S$. Então $e \cap e'$ é um avanço de fronteira. Pelo Lema 2.8, temos no máximo mais um avanço de fronteira. Este avanço de fronteira está em e ou e' ou em nenhum dos dois (é importante notar que as mesmas arestas não contêm outro avanço de fronteira, o que simplifica bastante, pois nossos pseudodiscos são poligonais). Suponha, sem perda de generalidade, que e não tem um avanço de fronteira com $\partial P'$. Começando em e, siga e em direção ao interior de P' e alcance a extremidade de e, que certamente não alcança o exterior de P' (pois supomos que e não possui outro avanço de fronteira). Atribuímos o vértice v a esta extremidade de e.

Precisamos provar que, desta forma em que atribuímos os vértices da união das fronteiras, atribuímos, no máximo, cada vértice de qualquer pseudodisco duas vezes. Obviamente, o primeiro tipo de vértice, em que está presente apenas na fronteira de um pseudodisco, é atribuído apenas uma vez, por ele mesmo, pois, obviamente, faz parte da união das fronteiras. O segundo tipo de vértice, pode ser visto em dois casos:

- Vértice de um pseudodisco que está no interior de outro pseudodisco: este vértice v estará no interior da união de pseudodiscos. Siga as arestas de P incidentes a v até a que a fronteira da união seja alcançada em um avanço de alguma aresta. Estes avanços (possivelmente até 2) são os únicos atribuídos a v.

- Vértice de um pseudodisco que está na fronteira de outro pseudodisco: este vértice v pode ser atribuído por ele mesmo, por achar que está na união da fronteira ou então pelos avanços (possivelmente até 2) ao longo de suas duas arestas incidentes. Pode ser atribuído pelos avanços, apenas se as arestas

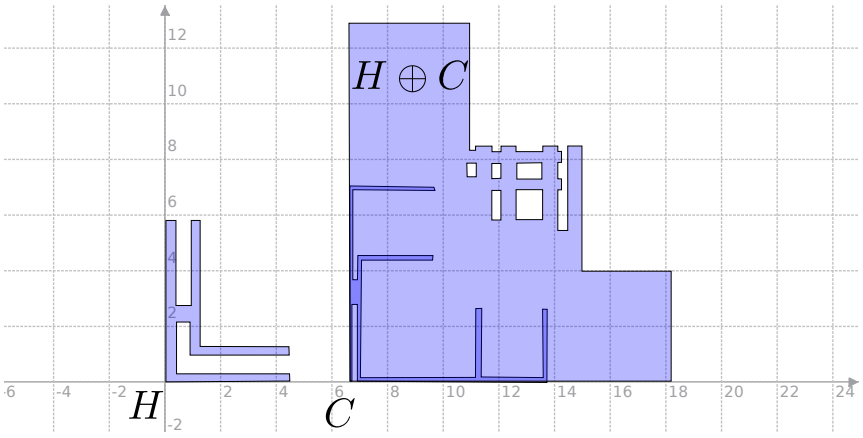


Figura 2.9: Limite inferior da soma de Minkowski de 2 polígonos não-convexos

incidentes forem de v para dentro da união dos pseudodiscos. Neste caso, ele não é atribuído por ele mesmo.

□

Para o leitor mais interessado, extensões do resultado do Teorema 2.9 podem ser encontrados em PACH *et al.* [13].

Concluímos que a complexidade da soma de Minkowski de um polígono não-convexo com um polígono convexo é $O(nm)$, onde n é a quantidade de vértices do primeiro polígono e m a quantidade de vértices do segundo polígono. Para calcular a complexidade da soma de Minkowski de um polígono não-convexo com um não-convexo, a idéia é basicamente a mesma. Triangular ambos os polígonos e utilizar a soma de minkowski em $O(n^2m^2)$ pares de triângulos. Assim temos o limite superior da complexidade. Para obter o limite inferior da complexidade, basta construir um exemplo com que a soma de Minkowski de dois polígonos não-convexos seja $O(n^2m^2)$. Veja a Figura 2.9 para um exemplo.

As complexidades da soma de Minkowski já conhecidas podem ser visualizadas na Tabela 2.1, retirada de JAVGAL [12].

Seja o conjunto dos obstáculos P_i, $1 \leq i \leq n$, denotado por P. Para computar cada obstáculo expandido P_i^*, basta utilizarmos a soma de Minkowski do robô R com o obstáculo P_i. Triangular todos os obstáculos resulta em $O(n)$ obstáculos triangulares com interiores disjuntos, onde n é o número de arestas. Note que os novos obstáculos são convexos, pelo fato de serem triangulares. Como o robô R tem descrição de complexidade constante, os obstáculos expandidos P_i^*, $1 \leq i \leq n$, também

Tabela 2.1: Limites da complexidade da soma de Minkowski

	A: Polígono/Poliedro de complexidade n			
	B: Polígono/Poliedro de complexidade m			
$2D$	B	A	Complexidade Combinatorial de $A \oplus B$	Complexidade de tempo do algoritmo
	Convexo	Convexo	$O(m+n)$	$O(m+n)$
	Convexo	Monótono	$O(mn)$	$O(mn)$
	Convexo	Simples	$O(mn)$	$O(mn\log(mn))$
	Disco	Simples	$O(n)$	$O(n)$
	Formato de estrela	Formato de estrela	$O(mn\min(m,n))$	$O(mn\log(mn))$
	Monótono	Simples	$O(mn^2)$	$O((mn+k)\log(mn)), k = O(mn^2)$
	Simples	Simples	$O(m^2n^2)$	$O(m\log m+n\log n+s+(s+k)\log s+k(m+n)\log(m+n)), k = O(m^2n^2), s = O(mn)O(m^2n^2)$

têm complexidade constante e, pelo Teorema 2.7, formam um conjunto de pseudo-discos. Finalmente, a união desse conjunto, pelo Teorema 2.9, tem complexidade linear, provando o Teorema 2.10.

Teorema 2.10. *Seja um robô R convexo de descrição de complexidade constante que translada entre um conjunto S de obstáculos poligonais com um total de n arestas. A complexidade do espaço de configuração livre C_F é $O(n)$.*

Sejam os triângulos $t_1, t_2, \ldots, t_n$ resultantes da triangulação do conjunto de obstáculos P. O espaço de configuração proibida C_P é:

$$C_P = \bigcup_{i=1}^{n} t_i \oplus (-R).$$

Note que precisamos fazer a união de regiões planares. Utilizando a estrutura de dados *DCEL*, introduzida na Página 8, podemos fazer essa união utilizando um algoritmo presente em DE BERG *et al.* [10], Capítulo 2.

Com uma abordagem de divisão-e-conquista, delineamos o algoritmo para realizar o processamento do espaço de configuração proibida C_P.

Algoritmo 2: Computação do espaço de configuração proibida C_P

Entrada: P_i, $1 \leq i \leq n$: Uma coleção de obstáculos

R: Robô

Saída: C_P: Espaço de configuração proibida

```
1 início
2     se n = 1 então
3         retorna R ⊕ P_1;
4     senão
5         C_P^1 ⟵ Algoritmo 2(P_1, ..., P_⌈n/2⌉) ;
6         C_P^2 ⟵ Algoritmo 2(P_⌈n/2⌉ + 1, ..., P_n) ;
7         retorna C_P^1 ⋃ C_P^2;
8 fim
```

Para analisarmos a complexidade do Algoritmo 2, seja $|P_i|$ a complexidade do obstáculo P_i, $1 \leq i \leq n$. Sabemos que a triangulação é linear [14], portanto triangular todos os obstáculos utiliza tempo linear.

Pela Linha 3 do Algoritmo 2, temos que a soma de Minkowski é linear em cada triângulo. Pela Linha 7 do Algoritmo 2, temos uma complexidade de

$$O((|C_P^1| + |C_P^2| + |C_P^1 \bigcup C_P^2|) \log(|C_P^1| + |C_P^2|)),$$

onde $|C_P^1|$, $|C_P^2|$ e $|C_P^1 \bigcup C_P^2|$ denotam a complexidade de C_P^1, C_P^2 e $C_P^1 \bigcup C_P^2$, respectivamente. Sabemos que o espaço livre é linear, pelo Teorema 2.10, logo $|C_P^1|$, $|C_P^2|$ e $|C_P^1 \bigcup C_P^2|$ são $O(n)$. Como resultado, para cada nível de recursão temos

$$O((|C_P^1| + |C_P^2| + |C_P^1 \bigcup C_P^2|) \log(|C_P^1| + |C_P^2|)) = O(n \log n),$$

obtendo a seguinte relação de recorrência sobre o tempo de execução do Algoritmo 2:

$$\begin{aligned} T(n) &= 2 * T(\left\lceil \frac{n}{2} \right\rceil) + O(n \log n) \\ &= O(n \log^2 n). \end{aligned}$$

Com todo o espaço de configuração livre C_F construído explicitamente, precisamos percorrê-lo. Para esse fim, utilizaremos o mapa trapezoidal, introduzido na Página 7. O mapa trapezoidal é construído em tempo esperado de $O(n \log n)$ utilizando um algoritmo randomizado presente em DE BERG *et al.* [10], Capítulo 6.

Após a construção do mapa trapezoidal, criaremos um guia em que cada trapézio e cada prolongamento terá um vértice em seus centros. Ligaremos com arestas os pares de vértices em que um está no centro de um trapézio, digamos $trap$, e o outro está em um prolongamento pertencente à fronteira de $trap$. Utilizando a estrutura de dados *DCEL*, introduzida na Página 8, o custo da construção do guia é linear, pois precisamos apenas percorrer as listas de arestas.

Dadas a configuração inicial q_{ini} e a configuração final q_{fim} do robô R, encontraremos um caminho da seguinte forma: escolhemos os trapézios em que as configurações pertencem, sendo cada busca por localização de ponto no trapézio executada em $O(\log n)$. Se ambas as configurações estiverem no mesmo trapézio, basta traçar apenas uma reta entre as configurações. Caso contrário, a configuração inicial q_{ini} vai ter um linha reta para o vértice do trapézio que q_{ini} representa e um algoritmo de busca em grafo será efetuado - busca em profundidade, por exemplo, cujo tempo é linear - para descobrir o caminho para o trapézio, digamos $trap'$, que a configuração final q_{fim} pertence. Depois, será traçado uma linha reta do vértice que representa o trapézio $trap'$ para a configuração final q_{fim}. Caso o algoritmo de busca no grafo não ache um resultado, isto significa que não há um caminho contínuo livre de colisão entre a configuração inicial q_{ini} e a configuração final q_{fim} do robô R. Com isto, concluímos:

Teorema 2.11. *Seja um robô R convexo de descrição de complexidade constante que translada entre um conjunto P de obstáculos poligonais com um total de n arestas. Preprocessamos o conjunto de obstáculos P em tempo esperado de $O(n \log^2 n)$, tal que para qualquer par de configurações $\langle q_{ini}, q_{fim} \rangle$ respondemos, em tempo linear, se há um caminho contínuo livre de colisão para R ligando essas duas configurações. Caso haja, esse caminho será retornado.*

Um exemplo sobre toda essa estrutura montada, da construção do espaço de configuração livre C_F ao guia, pode ser encontrado na Figura 2.10.

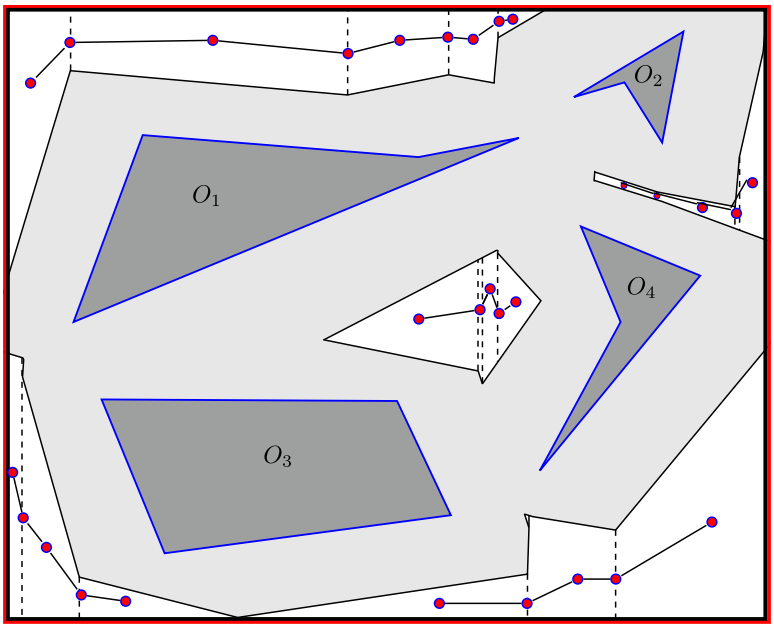


Figura 2.10: Guia em espaço de configuração livre C_F

2.2 Movimento planejado em dimensão arbitrária

Existem algoritmos de construção explícita que resolvem o problema de planejamento de movimento de robôs em dimensão árbitrária. As soluções que mostramos já foram produtos de décadas de pesquisa para a obtenção de algoritmos eficientes e práticos, mas que não são tão simples. Os algoritmos que resolvem em dimensão arbitrária infelizmente são extremamente desafiadores para entender e implementar, fora a complexidade, que deixa a desejar. A solução que utiliza decomposição algébrica cilíndrica - uma variação do mapa trapezoidal, onde os prolongamentos não param quando interceptam os outros obstáculos, podendo ser visualizado um exemplo na Figura 2.11 - tem tempo de execução duplamente exponencial na dimensão e depende de outros fatores [15].

Outro algoritmo que não depende da decomposição cilíndrica e que tem uma complexidade melhor (somente exponencial, com um expoente igual ao número de graus de liberdade do robô) é chamado de guia de Canny [16]. Sendo mais preciso sobre sua complexidade, o algoritmo executa em tempo de $O(n^k \log n)$. Variações e melhorias desse algoritmo podem ser encontradas em BAUSU *et al.* [15].

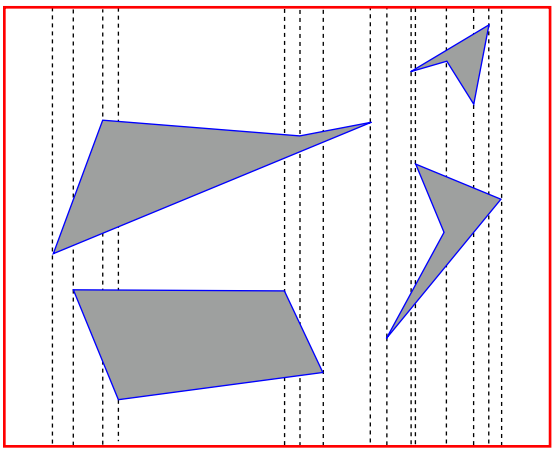


Figura 2.11: Decomposição cilíndrica de um espaço de configuração C

Capítulo 3

Amostragem

Imagine uma cena com centenas de milhares de primitivas geométricas. Computar a representação exata do espaço de configuração livre C_F levaria um tempo que tornaria essa abordagem proibitiva em termos de aplicação prática do movimento planejado de robôs. Evitando a construção explícita do espaço e utilizando um esquema baseado em amostragem, algoritmos para testar se existe um determinado caminho que seja livre de colisão podem ser mais eficientes, mesmo manuseando modelos geométricos complexos [11].

Com a amostragem, utiliza-se algoritmos de detecção de colisão, permitindo que os algoritmos de planejamento possam ser independentes de modelos geométricos particulares, não importando se são politopos, poliedros não convexos, conjuntos semi-algébricos ou primitivas geométricas, além de outras estruturas geométricas.

Uma vez que as amostras foram coletadas, o próximo passo é verificar se há colisão dessas amostras com o espaço de configuração C. Entretanto, a detecção de colisão foge do escopo deste capítulo, apesar de ser uma componente crítica do planejamento baseado em amostragem, pois geralmente o maior tempo gasto em aplicações de planejamento é na detecção de colisão. O leitor mais interessado deve consultar GOODMAN e O'ROURKE [17], Capítulo 35.

Mesmo que a detecção de colisão verifique que amostras pertencem ao espaço de configuração livre C_F, algoritmos de planejamento requerem que todo o caminho seja mapeado no espaço de configuração livre C_F. Logo, no problema de planejamento de robôs utilizando amostragem, identificamos um subproblema que é o problema do planejamento de caminho:

Definição 3.1 (Planejamento de caminho)**.** Temos como hipótese que a construção explícita do espaço de configuração C é proibitiva e um algoritmo de colisão está disponível como uma caixa preta, que responde de forma eficiente se uma determinada configuração pertence ou não ao espaço de configuração livre C_F. Sejam o espaço de configuração $C \subset \mathbb{R}^d$ e o par de configurações $\langle q_{ini}, q_{fim} \rangle$ sendo uma consulta de planejamento de caminho. O problema de planejamento de caminho consiste em encontrar um caminho contínuo livre de colisão $\tau : [0, 1] \longrightarrow C_F$, tal que $\tau(0) = q_{ini}$ e $\tau(1) = q_{fim}$.

Há de se notar que a amostragem de todo o espaço de configuração C é infinita, mesmo que seja apenas verificar se um caminho pertence ao espaço de configuração livre C_F, pois um caminho tem uma quantidade infinita de pontos. Em termos práticos, isto é impossível de ser tratado por um computador. Entretanto, será que poderíamos ter amostras finitas que podem resolver nosso problema?

Com um pouco de terminologia apresentada abaixo, matematicamente podemos aproximar um conjunto por um subconjunto seu, o que torna possível aproximar um conjunto infinito de pontos por um conjunto finito de pontos, que seria o nosso espaço de configuração C. Começamos pela definição de aderência, que permite aproximar um ponto a um subconjunto:

Definição 3.2 (Aderente)**.** Um ponto p é dito aderente a um subconjunto X de um espaço métrico M quando, para cada $\epsilon > 0$, $\rho(p, X) < \epsilon$, onde ρ denota a métrica de M.

Intuitivamente, um ponto p é aderente a um subconjunto X, que existem pontos de X arbitrariamente próximos de p, ou seja, para cada $\epsilon > 0$, podemos encontrar um $x \in X$, tal que $\rho(a, x) < \epsilon$. Com isso, podemos unir todos os pontos que são aderentes a X e teremos o que denotamos por fecho:

Definição 3.3 (Fecho)**.** O fecho de um conjunto X num espaço métrico M, denotado por $fecho(X)$ é o conjunto dos pontos de M que são aderentes a X.

Para entender o próximo passo, precisamos definir um termo bem usual em topologia: a bola aberta de um certo elemento, que corresponde a um conjunto de pontos que estão a uma distância menor que um certo valor.

Definição 3.4 (Bola aberta)**.** Sejam M um espaço métrico com métrica ρ, X um subconjunto de M e x um ponto de X. O conjunto de pontos de X que estão a uma distância de x estritamente inferior a δ é chamado de bola aberta centrada em x de raio δ, ou seja:

$$B(x, \delta) = \{y \in X : \rho(x, y) < \delta\},$$

onde ρ denota a métrica de M.

Quando toda bola aberta em M contém algum ponto de X, podemos aproximar M utilizando os pontos de X. Isso é possível quando o conjunto é denso:

Definição 3.5 (Denso)**.** Um subconjunto $X \subset M$ é dito denso em M quando $fecho(X) = M$.

Observe que as definições são bem gerais, mas no nosso contexto, o espaço métrico M que utilizamos é o espaço Euclidiano. A definição de denso também é uma definição bem geral. Por exemplo, o conjunto dos números racionais ($\mathbb{Q}$) é denso no conjunto dos números reais ($\mathbb{R}$).

A notação de denso foi introduzida porque qualquer espaço de configuração C pode ser bem aproximado por uma amostragem em que o número de iterações tende para o infinito [18]. Isto é um problema, pois o interessante é ter a garantia de que o problema vai ser resolvido em tempo finito. Por isto, estamos interessados nos algoritmos ditos completos:

Definição 3.6 (Algoritmo completo)**.** Um algoritmo é dito completo se, para qualquer entrada, o algoritmo reporta em tempo finito se há uma solução. Além disso, se essa solução existe, também será reportada em tempo finito.

Observe que no caso de não haver solução, o algoritmo pode executar indefinidamente.

A amostragem pode ser feita usando as abordagens determinística e probabilística. Utilizando uma abordagem determinística, as amostras podem ser dispostas em formato de grade, onde são fixados k' pontos por eixo, sendo necessários k'^d pontos para formar a grade. Outra forma de construir a grade é definir o tamanho do intervalo entre os pontos do mesmo eixo, ou seja, a resolução, como está sendo definido abaixo.

Definição 3.7 (Grade ortogonal regular $G(r)$)**.** Uma grade ortogonal regular $G(r)$ com resolução r é definida por:

$$G(r) = \{(z_1 r, \ldots, z_d r) \mid z_1, \ldots, z_d \in \mathbb{Z}\}.$$

Ilustramos um exemplo de grade ortogonal regular $G(r)$ na Figura 3.1.

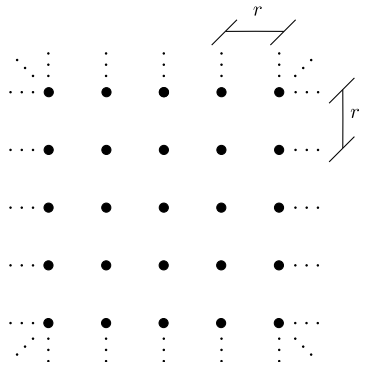


Figura 3.1: Exemplo bidimensional de $G(r)$

Uma generalização da idéia de resolução é um critério chamado de dispersão [19]. À medida que a resolução é aumentada a dispers ao diminui, por isso temos uma generalização.

Definição 3.8 (Dispersão)**.** Sejam $a, b \in \mathbb{R}$, $X = [a, b]^d \subset \mathbb{R}^d$ o espaço em que os pontos serão amostrados e $P = \{p_1, p_2, \ldots, p_n\}$ um conjunto finito de n pontos em X. A dispersão dessa amostragem é:

$$\delta(P, \rho) = \sup_{x \in X} \rho(x, P),$$

onde ρ denota qualquer métrica e sup denota o menor limite superior.

Intuitivamente, a dispersão sobre $\mathbb{R}^n$, utilizando como métrica de distância a métrica Euclidiana, ou seja, $\rho(x, p) = \sqrt{\sum_{i=1}^{n} |x_i - p_i|^2}$, pode ser interpretada como o raio da maior bola vazia que pode ser colocada no espaço depois que os pontos são amostrados. Podemos visualizar um exemplo na Figura 3.2, onde algumas bolas vazias foram colocadas, para entender melhor o conceito.

A definição de dispersão foi desenvolvida para limitar o erro de problemas de otimização [20]. O uso de dispersão no problema de movimento planejado de robôs é importante para evitar que ocorram espaços grandes em que não existam pontos

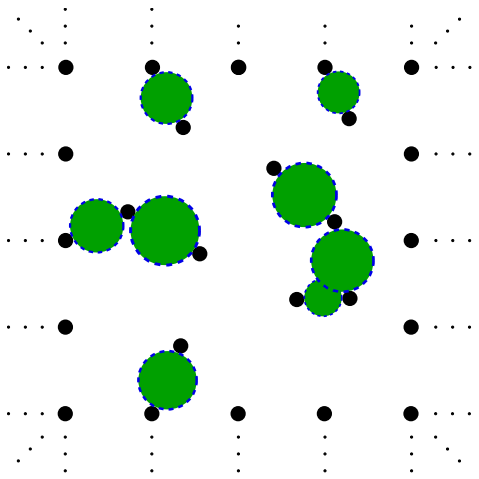


Figura 3.2: Exemplo bidimensional de dispersão utilizando métrica Euclidiana.

amostrados, de forma que ao melhorar a dispersão, os pontos se distribuem mais uniformemente no espaço de configuração.

A respeito da definição de algoritmos completos, densidade e segundo as abordagens determinística e probabilística, introduzimos as notações a seguir.

Definição 3.9 (Resolução completa)**.** Um algoritmo utilizando a abordagem determinística de amostragem densa é dito de resolução completa.

Em outras palavras, um algoritmo de resolução completa achará a solução em tempo finito se ela existir. Porém, poderá executar infinitamente caso a solução não exista. Já utilizando a abordagem probabilística basta que, com uma quantidade de pontos suficiente, a probabilidade que encontre uma solução convirja para 1.

Definição 3.10 (Probabilisticamente completo)**.** Um algoritmo utilizando a abordagem probabilística de amostragem em que, com uma quantidade de pontos suficiente, a probabilidade que seja denso convirja para 1 é dito probabilisticamente completo.

Observe que nem todo algoritmo de resolução completa é algoritmo completo e que nem todo algoritmo probabilisticamente completo é algoritmo completo.

A amostragem, como foi dito anteriormente, pode ser feita utilizando as abordagens determinística e probabilística. Uma abordagem determinística bastante comum é a amostragem baseada em grade. Na Seção 3.1, apresentaremos um algoritmo que, *a priori*, não é resolução completa, nem mesmo um algoritmo completo, porém

probabilisticamente completo. Entretanto, na Seção 3.1.1, discutimos brevemente formas de resolver o problema de resolução completa do algoritmo apresentado na Seção 3.1. Uma abordagem probabilística bastante comum é o guia probabilístico, discutido na Seção 3.2. Na Seção 3.3, traçaremos uma análise do porquê do guia probabilístico funcionar tão bem.

3.1 Amostragem baseada em grade

Definir uma grade sobre o espaço de configuração e utilizar algoritmos de buscas é considerado por muitos como a forma mais simples de planejamento de caminho [20].

Podemos generalizar ainda mais a definição de grade apresentada na Definição 3.7 para o caso em que cada dimensão tem uma resolução possivelmente diferente.

Definição 3.11 (Grade ortogonal $G(r_1, \ldots, r_d)$)**.** Uma grade ortogonal $G(r_1, \ldots, r_d)$ com resoluções $r_1, \ldots, r_d$ é definida por:

$$G(r_1, \ldots, r_d) = \{(z_1 r_1, \ldots, z_d r_d) \mid z_1, \ldots, z_d \in \mathbb{Z}\}.$$

Ilustramos um exemplo de grade ortogonal $G(r_1, r_2)$ na Figura 3.3.

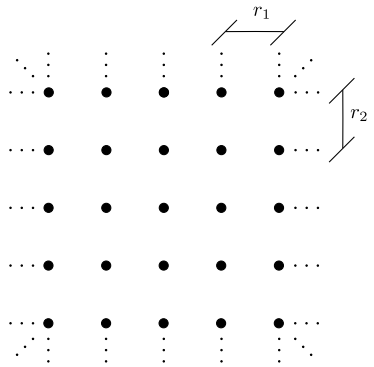


Figura 3.3: Exemplo bidimensional de $G(r_1, r_2)$

Depois de definida a grade, precisamos explicitar a representação de um ponto da grade.

Definição 3.12 (Ponto da grade)**.** Sejam $a, b \in \mathbb{R}$, $X = [a, b]^d \subset \mathbb{R}^d$ em que a grade $G(r_1, \ldots, r_d)$ será amostrada. Um ponto $q = (z_1 r_1, \ldots, z_d r_d)$ é dito um ponto da grade, onde $z_1, \ldots, z_d \in \mathbb{Z}$ e $z_1 r_1, \ldots, z_d r_d \in [a, b]^d$.

A partir de um ponto da grade, seria interessante buscar os vizinhos que estejam distantes até k passos dados em dimensões diferentes, onde cada dimensão fornece apenas dois passos a serem dados, do tamanho da distância entre pontos adjacentes da grade na mesma direção, mas em ambos sentidos. O conjunto obtido com esta definição é formalmente apresentado abaixo.

Definição 3.13 (k-vizinhança)**.** Uma k vizinhança de um ponto q da grade d-dimensional, que não seja um ponto de fronteira da grade, é definida recursivamente como:

$$\begin{aligned} N_k(q) &= \{q \pm \Delta r_{a_1} \pm \Delta r_{a_2} \pm \cdots \pm \Delta r_{a_k} | \\ &\quad 1 \leq a_1, a_2, \ldots, a_k \leq d, a_i \neq a_j, i \neq j, 1 \leq i, j \leq k\} \\ &\quad \bigcup N_{k-1}(q) \bigcup \cdots \bigcup N_1(q), \end{aligned}$$

onde Δr_i é o ponto d-dimensional em que a i-ésima componente é a i-ésima resolução e o resto das componentes são nulas.

Tendo em mãos a definição de grade, ponto da grade e vizinhança, um algoritmo de planejamento de caminho é facilmente obtido. Primeiro, se os pontos q_{ini} e q_{fim} não têm seus equivalentes em pontos da grade, eles devem ser ligados a pontos da grade mais pertos, conectando q_{ini} e q_{fim} à grade.

Note que se todos os pontos da grade e as arestas forem verificados por colisões, ao removermos os que colidem, nós obteremos um guia que poderá servir para múltiplas consultas, pois conterá, *a priori*, a informação de arestas livres de colisão para responder a qualquer par de configuração inicial e final do problema de planejamento de caminho. Entretanto, não se fazem necessárias todas as verificações *a priori*, podendo ser feitas a medida que as consultas são realizadas. Os vértices e arestas, à medida que o algoritmo está sendo executado, que se tornam candidatos à verificação por colisão ao utilizarmos algoritmos de buscas para resolver o problema de planejamento de caminho, são verificados se colidem com algum obstáculo e são guardados em alguma estrutura de dados eficiente para que não seja necessário realizar esta computação novamente.

3.1.1 Problemas em grades

Se as resoluções da grade forem ótimas para o problema a ser considerado, ou seja, a resolução é a menor possível de forma a garantir que o algoritmo seja resolução completa, o algoritmo terá um bom desempenho. Porém, se a resolução for alta demais o algoritmo pode não ser resolução completa e se a resolução for baixa demais terá um péssimo desempenho.

Existem diversas formas para corrigir o problema da resolução incompleta, seguindo o paradigma de iterativamente refinar a resolução até que uma solução seja encontrada. Uma solução bem ingênua seria a de, a cada iteração, caso não houvesse um retorno positivo de que o caminho exista, diminuir pela metade a resolução da grade, dobrando a quantidade de pontos por eixo. Se na primeira iteração, a grade tem 2^d pontos, na i-ésima iteração, a grade terá 2^{id} pontos. Mesmo que arestas e vértices possam ser reutilizados em um passo do refinamento da grade, a taxa de crescimento do tempo de busca cresceria muito rapidamente. Imagine que sejam $d = 10$ dimensões. A primeira grade terá $2^{id} = 1024$ pontos a serem buscados. Na segunda iteração, a quantidade de pontos aumentará para 1048576. O que mostra que a grade em dimensões elevadas não é uma boa estratégia a ser escolhida.

Há formas para que esse número de pontos não cresça tão rapidamente. Uma forma seria calcular a quantidade de pontos em cada eixo em função da iteração i do algoritmo, usando a fórmula i^d, em vez de diminuirmos pela metade a resolução da grade. Outra forma seria diminuir uma resolução por eixo a cada iteração. Enfim, várias estratégias podem ser criadas e até combinadas para evitar esse crescimento tão rápido no tempo de busca.

3.2 Guia probabilístico

Introduzido em KAVRAKI *et al.* [21], o guia probabilístico foi desenvolvido para resolver o problema da amostragem baseada em grade, devido ao número exponencial de pontos em função da dimensão. O método é executado em duas fases:

1. Fase de aprendizado: um guia, cuja representação é um grafo não-direcionado G topológico, é construído ao gerar repetidamente configurações livres do robô e conectando essas configurações usando um planejador de caminho simples,

mas eficiente, responsável por verificar se existe um caminho livre de colisão definido por um segmento de reta entre dois vértices. Desta forma, os espaços livres são os vértices de G e as conexões entre as configurações são as arestas.

2. Fase de consulta: uma consulta requer um caminho entre duas configurações livres do robô. Primeiro, o método faz um casamento entre um par de configurações livres e um par de vértices do guia. Segundo, o método faz uma busca no grafo e fornece as informações necessárias para descobrir uma sequência de vértices que conectem a dupla de vértices que representam a dupla de configurações livres. Essa sequência de vértices é um caminho viável para o robô.

3.2.1 Fase de aprendizado

A fase de aprendizado necessita construir uma estrutura que permita que múltiplas consultas sejam realizadas no mesmo espaço e tratadas eficientemente. Esta estrutura é o grafo retornado pelo Algoritmo 3, que ilustra o processo da fase de aprendizado.

O algoritmo executará até que sejam escolhidos k amostras que pertençam ao espaço de configuração livre C_F.

Em cada iteração do laço externo, a amostra se torna um vértice v do grafo G e, depois, tenta conectar esse vértice a cada vértice, digamos v', que esteja próximo de v e que já não se encontra na mesma componente conexa, verificado pela operação $\neg mesmaComponenteConexa(v, v')$, para evitar processamento extra com a operação $planejadorPodeComputarCaminho(v, v')$. Vale lembrar que esta última operação usa um método de planejamento local que verifica se existe um caminho sem colisões entre v e v'.

Além disso, podemos utilizar vários critérios para escolher N_v. Originalmente, os vértices que eram considerados como vizinhos eram ordenados por ordem crescente de distância de v. Isto faz sentido, pois é intuitivamente menos custoso, em relação aos demais, verificar se o caminho entre v e seu vizinho, se ele estiver bem próximo, é livre de colisão.

Algoritmo 3: Algoritmo da fase de aprendizado

Entrada: k: quantidade de amostras no espaço de configurações livres

Saída: $G(N, E)$: grafo não-direcionado

```
1  início
2      N ⟵ ∅;
3      E ⟵ ∅;
4      i ⟵ 0;
5      enquanto i < k faça
6          v ⟵ uma configuração livre escolhida aleatoriamente;
7          i ⟵ i + 1;
8          N_v ⟵ um conjunto de vizinhos candidatos de v escolhidos de N;
9          N ⟵ N ⋃ {v};
10         para cada v' ∈ N_v faça
11             se ¬mesmaComponenteConexa(v, v') ∧
               planejadorPodeComputarCaminho(v, v') então
12                 E ⟵ E ⋃ {(v, v')};
13     retorna G(N, E);
14 fim
```

Alguns outros critérios para escolher N_v são [22]:

- Os k vértices mais próximos: tentar conectar cada configuração v com os k vértices mais próximos do grafo G. Desta forma a comparação $\neg mesmaComponenteConexa(v, v')$ se transformaria na comparação $grau(v) \leq k$.

- k vértices de cada componente conexa: tentar obter até k elementos de cada componente conexa de G que estejam perto da nova configuração.

- Os vértices mais próximos em um raio r: tentar obter os vértices que estejam até uma distância r do vértice v. Se o parâmetro não for especificado, tentar usar um limite superior para não tentar fazer muitas conexões e diminuir o raio da bola caso o número de pontos seja alto, que pode ser baseado na dispersão, se esta informação estiver disponível para v.

- Visibilidade: um vértice é inútil quando pode ser conectado a apenas uma componente e é descartado. Apenas os vértices úteis são considerados [23].

Na Figura 3.4 podemos entender o passo-a-passo da fase de aprendizado do algoritmo de guia probabilístico.

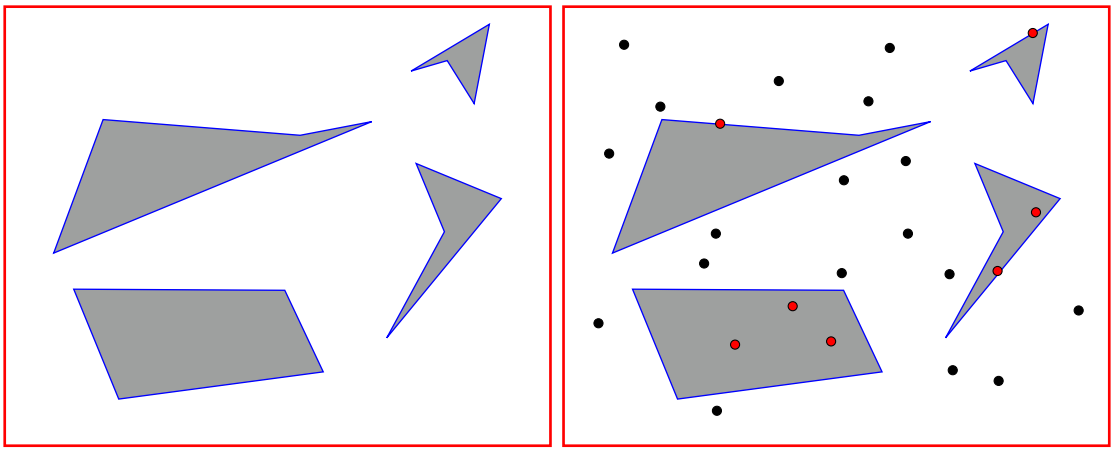
3.2.2 Fase de consulta

Quando a consulta é realizada, os dois pontos q_{ini} e q_{fim} que definem as configurações inicial e final, respectivamente, são tratados como dois novos nós no grafo G. Portanto, utilizamos o Algoritmo 3 para adicioná-los a G e tentarmos conectá-los ao restante do grafo. Após isto, realizamos uma busca no grafo para descobrir se há um caminho livre de colisão que liga q_{ini} a q_{fim}. Se houver uma resposta afirmativa, o caminho traçado no grafo G tem um caminho equivalente no espaço de configuração livre C_F. Porém, a resposta negativa pode ter duas interpretações: uma interpretação é que o caminho realmente não existe, enquanto a outra interpretação é que o caminho pode existir, mas o espaço não foi amostrado de modo a verificar que o caminho existe. Este problema é semelhante no caso em que as grades não têm uma resolução apropriada.

Note que a amostragem é baseada em grafos e em ambas as fases do guia probabilístico envolve-se busca em grafo. Algumas das buscas que podem ser utilizadas são as buscas em largura e profundidade [24, Capítulo 22], algoritmo de Dijkstra [24, Capítulo 24], A^*, "best-first", busca em profundidade iterativa e busca bidirecional [25, Capítulo 3].

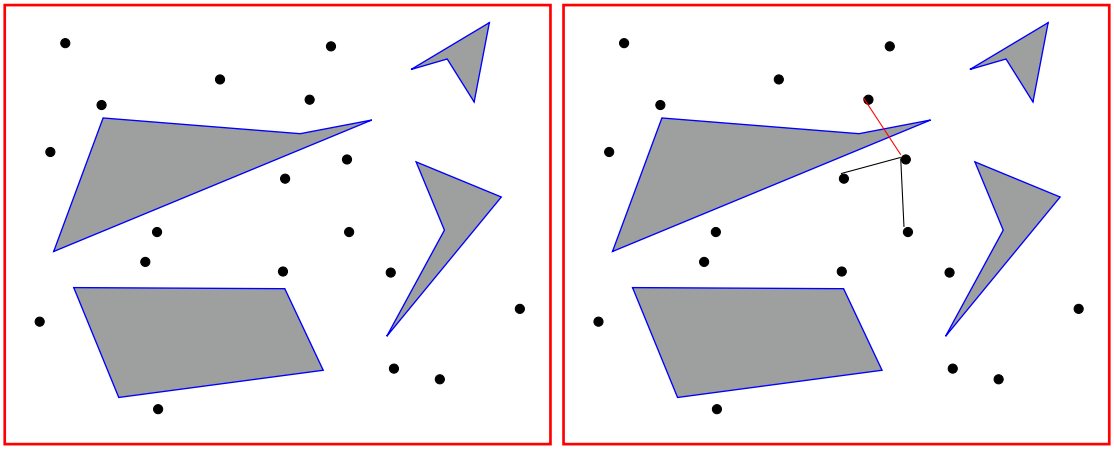
Podemos entender o passo-a-passo da fase de consulta do algoritmo de guia probabilístico na Figura 3.5 .

Na prática, o guia probabilístico é amplamente utilizado e foram propostas inúmeras variações ao longo de alguns anos, cuja comparações entre si se tornavam complicadas, por utilizarem implementações, bibliotecas e cenas diferentes para testes, além de terem sido executadas em máquinas de configurações diferentes. Um estudo comparativo de várias técnicas que foram testadas no mesmo ambiente de configuração, com as mesmas bibliotecas em comum e na mesma cena estão presentes em GERAERTS e OVERMARS [22].



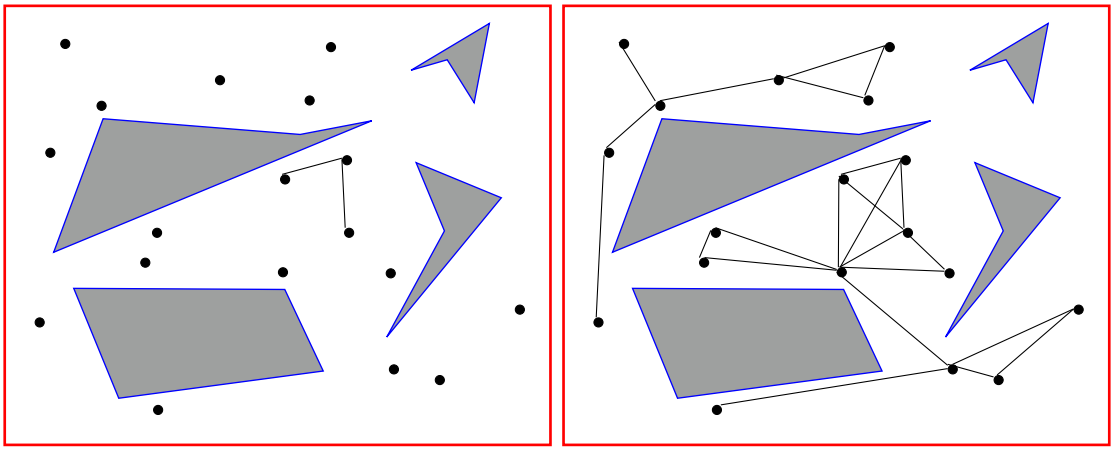
(a) Espaço de configuração

(b) k amostragens são realizadas



(c) Apenas configurações que não colidem são consideradas

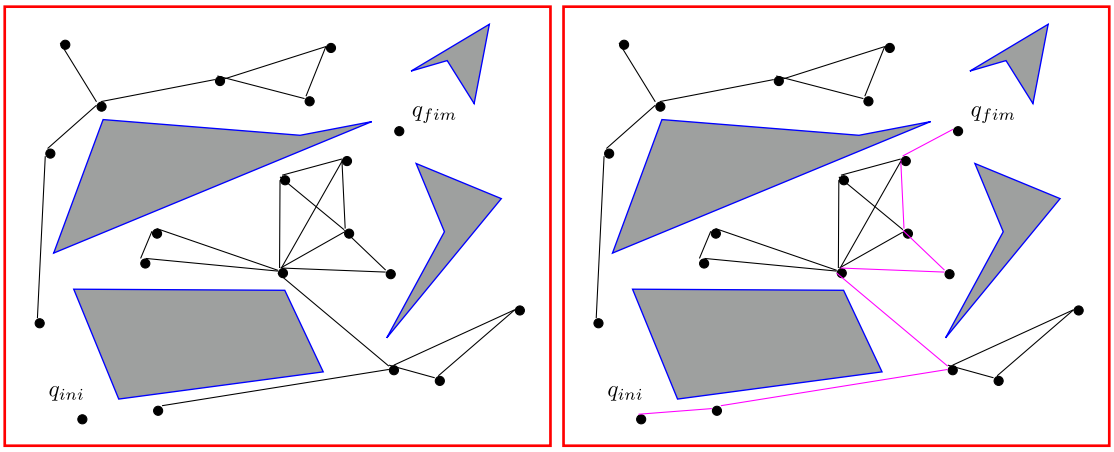
(d) Cada configuração é ligada diretamente aos vizinhos mais próximos



(e) Serão consideradas apenas as ligações sem colisão

(f) As ligações livres de colisão são os caminhos do guia probabilístico

Figura 3.4: Fase de aprendizado do guia probabilístico



(a) As configurações q_{ini} e q_{fim} são incluídas (b) O Guia probabilístico busca um caminho entre q_{ini} e q_{fim}

Figura 3.5: Fase de consulta do guia probabilístico

3.2.3 Guia probabilístico preguiçoso

O guia probabilístico monta um plano de consulta que responde de forma eficiente o problema de planejamento de caminho. Porém, o fato de construir um guia extenso tem um certo custo, o que para a realização de até uma quantidade de consultas pode não ser vantajoso, o que levou a uma variação em que a fase de aprendizado é mais rápida em responder uma única consulta. Além disso, o plano de consulta pode ser reusado para replanejar caminhos onde a configuração do espaço é dinâmica, ou seja, pode mudar.

No algoritmo de guia probabilístico, na fase de planejamento, ao tentar ligar, através de uma aresta, todo vértice v a um vizinho v', a aresta é verificada se está em colisão com os obstáculos. No algoritmo de guia probabilístico preguiçoso, essa verificação não é realizada. Apenas na fase de consulta, quando a busca está sendo realizada com o algoritmo de busca A^*, é que as arestas são verificadas se estão em colisão. As arestas que estiverem em colisão são retiradas do grafo G e a busca é refeita até que um caminho viável seja encontrado [26]. Uma variação do guia probabilístico preguiçoso seria, na fase de consulta, verificar se as arestas durante a busca estão em colisão e não apenas quando a solução é encontrada [27]. Observe que, se a solução não for encontrada, carece inserir novos nós ao guia e os algoritmos são executados novamente. A vantagem do guia probabilístico preguiçoso sobre o guia probabilístico é que a solução pode ser encontrada sem que todas as arestas

sejam verificadas quanto à colisão, mesmo que não seja improvável disto acontecer.

3.3 Desempenho do guia probabilístico

O guia probabilístico é um método largamente utilizado, pois funciona muito bem na prática, mostrando bom desempenho empiricamente e geralmente são probabilisticamente completos [28].

Note que na função $planejadorPodeComputarCaminho(v, v')$ do Algoritmo 3, Página 50, podemos dizer que v enxerga v' caso o retorno desta função seja verdadeiro. Logo, introduzimos o conceito de conjunto visibilidade:

Definição 3.14 (Conjunto visibilidade $V(M)$)**.** O conjunto visibilidade de um conjunto M de configurações em um espaço de configuração livre C_F é

$$V(M) = \bigcup_{v \in M} V(v),$$

onde $V(v) = \{v' \mid v' \in C_F \wedge planejadorPodeComputarCaminho(v, v') = verdadeiro\}$ é o conjunto de vértices v' que v enxerga.

Se alguns pontos têm um conjunto visibilidade relativamente grande, será fácil amostrar um conjunto de configurações que enxerguem, juntas, a maior parte do espaço de configuração livre C_F. Se todo ponto tem um conjunto de visibilidade de proporção ϵ do volume do espaço de configuração livre C_F, temos um espaço de configuração livre ϵ-bom [29]:

Definição 3.15 (Espaço de configuração ϵ-bom)**.** Seja $\epsilon \in (0, 1]$ uma constante e um espaço de configuração livre C_F. Um ponto $v \in C_F$ é ϵ-bom se enxerga pelo menos uma fração ϵ de C_F:

$$\mu(V(v)) \geq \epsilon\mu(C_F),$$

onde $\mu(S)$ denota o volume de um subconjunto $S \subseteq C_F$. Se todo ponto $v \in C_F$ é ϵ-bom, então C_F é ϵ-bom.

Temos um exemplo de um vértice 0,35-bom, na Figura 3.6.

Há de se ressaltar que o estudo realizado em KAVRAKI *et al.* [29] estima a taxa de convergência probabilística de uma variante do guia probabilístico, que admite

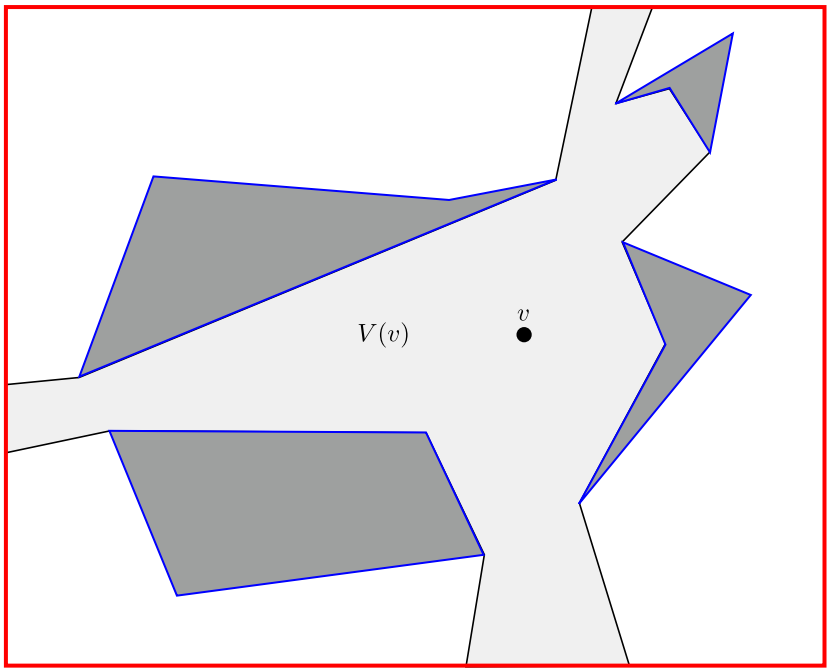


Figura 3.6: Exemplo de ϵ-bom: vértice v é 0,35-bom

que um planejador completo esteja disponível para melhorar a conectividade do guia. Entretanto, a hipótese que este planejador completo exista não é realista.

A definição de ϵ-bom é fraca para qualquer conclusão em relação à conectividade do guia construído, apenas nos diz que é fácil amostrar um conjunto de configurações que, juntas, conseguem ver muito do espaço de configuração C_F. Uma propriedade mais forte, chamada $(\alpha, \beta, \epsilon)$-expansivo [28], é necessária para obtermos maiores conclusões quanto à conectividade.

Definição 3.16 (Espaço de configuração $(\alpha, \beta, \epsilon)$-expansivo)**.** Sejam $\alpha, \beta, \epsilon \in (0, 1]$ constantes. O espaço de configuração livre C_F é dito $(\alpha, \beta, \epsilon)$-expansivo se cada uma de suas componentes conexas $C_{F_i} \subseteq C_F$ satisfaz as seguintes condições:

- Todo ponto $v \in C_{F_i}$ é ϵ-bom, ou seja, para todos os pontos $v \in C_{F_i}$, $\mu(V(v)) \geq \epsilon\, \mu\,(C_{F_i})$.

- Para qualquer subconjunto de pontos $S \subseteq C_{F_i}$, o conjunto β-espia$(S) = \{v \in S \mid \mu(V(v) \backslash S) \geq \beta \cdot \mu(C_{F_i} \backslash S)\}$ tem volume $\mu(\beta\text{-espia}(S)) \geq \alpha \cdot \mu(S)$.

Para deixar bem claro a notação de β-espia(S), veja a Figura 3.7, onde S é o espaço de configuração abaixo da linha tracejada. Neste exemplo, o conjunto $V(v') \backslash S$ representa pelo menos 10% da componente conexa do espaço de confi-

guração livre que v' pertence, a menos de S. Logo, v' pertence a $0,1$-$espia(S)$. Já o conjunto $V(v)\backslash S$ representa menos de 10% da componente conexa do espaço de configuração livre que v pertence, a menos de S. Logo, v não pertence a $0,1$-$espia(S)$.

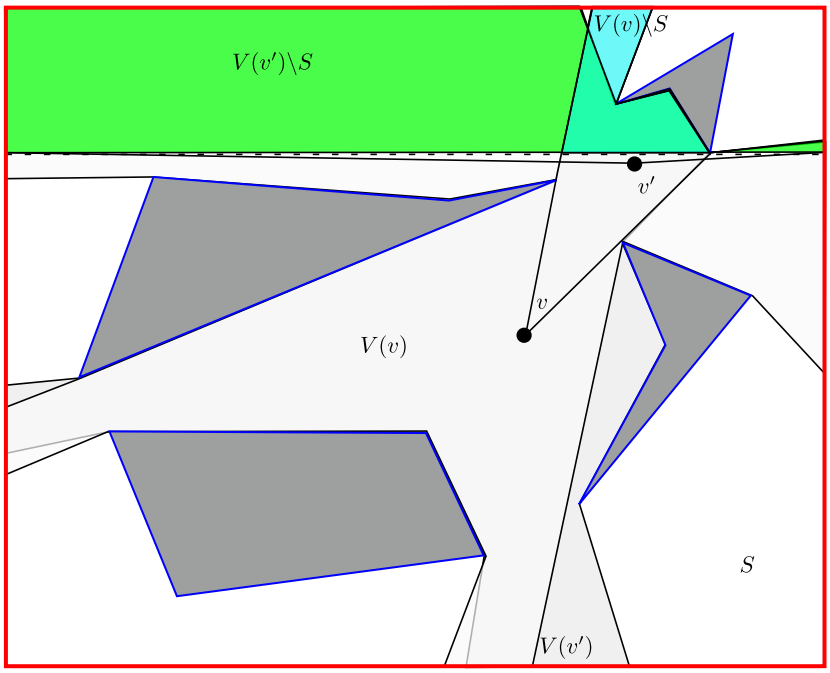


Figura 3.7: Exemplo de β-espia(S): $v' \in$ 0,1-espia(S), $v \notin$ 0,1-espia(S)

Assim, o Teorema 3.1 caracteriza o desempenho do guia probabilístico em um espaço de configuração $(\alpha, \beta, \epsilon)$-expansivo C e é provado em HSU *et al.* [28].

Teorema 3.1. *Sejam um espaço de configuração $(\alpha, \beta, \epsilon)$-expansivo C e o par de configurações $\langle q_{ini}, q_{fim} \rangle$ pertencentes à mesma componente de C. O Algoritmo 3, Página 50, com amostragem uniforme retorna um caminho entre q_{ini} e q_{fim} com probabilidade tendendo a 1 em uma taxa exponencial em função do crescimento de k (a quantidade de amostras no espaço de configuração livre C_F). Mais precisamente, o limite superior da probabilidade de falha é:*

$$\left(\frac{c_1}{\epsilon\alpha} e^{c_2\epsilon\alpha\left(-k+\frac{c_3}{\beta}\right)}\right),$$

onde c_1, c_2 e c_3 são constantes positivas.

Os valores de α, β e ϵ quando grandes, resultam em um tempo de computação mais rápido, pois k precisa ser pequeno para que as respostas das consultas sejam corretas. Em relação aos valores de α, β e ϵ quando pequenos, resulta em respostas incorretas, mesmo quando o guia é muito grande.

Teorema 3.2. *Para quaisquer $\epsilon > 0$, $k > 0$ arbitrariamente grandes e $\gamma \in (0, 1]$, existem α_0 e β_0 tais que se F não é $(\alpha, \beta, \epsilon)$-expansivo para $\alpha \geq \alpha_0$ e $\beta \geq \beta_0$, então existem q_{ini} e q_{fim} presentes na mesma componente conexa F tais que o Algoritmo 3, Página 50, falha para retornar um caminho entre q_{ini} e q_{fim} com probabilidade maior do que γ.*

Demonstração. Se F não é $(\alpha, \beta, \epsilon)$-expansivo, para $\alpha \geq \alpha_0$ e $\beta \geq \beta_0$, então existe uma componente conexa F_i de F e um subconjunto $S \subseteq F_i$ com um espia pequeno tal que

$$\mu(\beta_0\text{-espia}(S)) < \alpha_0 \mu(S). \tag{3.1}$$

Para extrair um limite inferior na probabilidade que o Algoritmo 3, Página 50, gera um guia que não contém um caminho ligando q_{ini} e q_{fim}, vamos utilizar as configurações q_{ini} e q_{fim} escolhidas uniformemente de S e seu complemento $F_i \backslash S$, respectivamente. Esse caminho deve conter dois nós consecutivos, digamos $v \in S$ e $v' \in F \backslash S$. Note que $v' \in V(v)$ e $v \in V(v')$. Sejam $L = \beta_0$-espia(S) e L' seu complemento, isto é, $L' = S \backslash L$. Temos, pela definição de probabilidade total:

$$\begin{aligned} P(v' \in V(v)) &= P(v' \in V(v) \mid v \in L) P(v \in L) + P(v' \in V(v) \mid v \in L') P(v \in L') \\ &< 1 \cdot P(v \in L) + P(v' \in V(v) \mid v \in L') \cdot 1 \end{aligned}$$

- Provar que $P(v \in L) < \alpha_0$:
  - Se $v = q_{ini}$. Então, $P(v \in L) = \frac{\mu(L)}{\mu(S)}$, se q_{ini} for escolhido uniformemente de S. Segue da Equação 3.1 que $P(v \in L) < \alpha_0$.
  - Se $v \neq q_{ini}$. Então v é um nó correspondente a uma configuração amostrada uniformemente de F. Logo, $P(v \in L) = \frac{\mu(L)}{\mu(F)} < \frac{\alpha_0 \mu(S)}{\mu(F)}$ e, como $S \subseteq F$ e $\frac{\mu(S)}{\mu(F)} \leq 1$, concluímos que $P(v \in L) < \alpha_0$.
- Provar que $P(v' \in V(v) \mid v \in L') < \beta_0$:
  - Se $v' = q_{fim}$. Como q_{fim} foi escolhido uniformemente de $F_i \backslash S$, então $P(v' \in V(v) \mid v \in L') = \frac{\mu(V(v) \backslash S)}{\mu(F_i \backslash S)}$. Pela definição de espia, temos para todo $v' \in L$: $\mu(V(v) \backslash S) < \beta_0 \mu(F_i \backslash S)$. Logo, $P(v' \in V(v) \mid v \in L') < \beta_0$.

– Seja $v' \neq q_{fim}$. Então $P(v' \in V(v) \mid v \in L') = \frac{\mu(V(v) \backslash S)}{\mu(F)} < \beta_0 \frac{\mu(F_i \backslash S)}{\mu(F)} < \beta_0$.

Com isso, até agora temos que $P(v' \in V(v)) < \alpha_0 + \beta_0 \leq 2 \max(\alpha_0, \beta_0)$. Escolha $\max(\alpha_0, \beta_0) \leq \frac{(1-\gamma)}{6k^2}$. Denotaremos por $q_{ini} \Rightarrow q_{fim}$ o evento em que existe um caminho, no guia, conectando q_{ini} e q_{fim}. Como o guia tem $k+2$ nós,

$$P(q_{ini} \Rightarrow q_{fim}) \leq \binom{k+2}{2} P(v' \in V(v)) \leq 3k^2 \cdot 2 \max(\alpha_0, \beta_0) \leq 1 - \gamma.$$

Sejam as configurações $q_{ini} \in S$ e $q_{fim} \in F_i \backslash S$ escolhidas uniformemente dos seus respectivos conjuntos. Existe pelo menos um par de configurações q'_{ini} e q'_{fim} tal que o Algoritmo 3, Página 50, falha em produzir um caminho entre q'_{ini} e q'_{fim} com probabilidade maior que γ. Por outro lado, se escolher α_0 e β_0 menores que $\frac{(1-\gamma)}{6k^2}$, todos os pares de configurações q'_{ini} e q'_{fim} vão fazer o Algoritmo 3, Página 50, falhar em produzir um caminho entre q'_{ini} e q'_{fim} com probabilidade maior que γ. □

3.4 Espectro entre grades e guia probabilístico

De um lado, temos a amostragem baseada em grade utilizando uma amostragem determinística. Por outro lado, temos a amostragem do guia probabilístico utilizando uma amostragem probabilística. Enquanto o guia probabilístico tem uma pobre distribuição dos pontos e uma estrutura irregular de vizinhança, a amostragem baseada em grade tem o problema do excesso de pontos amostrados. Entre essas abordagens, há outras abordagens que são apresentadas na Tabela 3.1, retirada de LAVALLE *et al.* [20], que resume as famílias de algoritmos de planejamento utilizando amostragem, mostrando qual a dimensão apropriada dessas famílias, a forma em que os pontos estão dispostos no espaço e a avaliação quanto ao critério de dispersão mencionado no início do capítulo.

Tabela 3.1: Família de algoritmos de planejamento utilizando amostragem

Algoritmo de Planejamento	Disposição dos pontos	Dimensão	Dispersão
Busca em grade	Muitos pontos por eixo	Proibitiva em altas dimensões	Ótima
Busca em grade subamostrada	Poucos pontos por eixo	Escalonada para dimensão moderada	Ótima
Guia reticulado	Estrutura de vizinhança regular em eixos não ortogonais	Arbitrário	Ótima
Guia quasi-aleatório	Estrutura de vizinhança irregular	Arbitrário	Ótima com constantes altas
Guia probabilístico	Estrutura de vizinhança irregular	Arbitrário	Não-ótima

Capítulo 4

Cenários Realistas

Em configurações típicas, os obstáculos tendem a não ter partes longas e estreitas. Assim, a tendência é de não termos uma aglomeração de obstáculos em um local, pois um obstáculo já ocupa boa parte desse local. Uma técnica explora este fato, onde espera-se que os obstáculos sejam "gordos" (há diversas definições para "gordo", embora haja uma equivalência provada entre algumas delas [30, Seção 5], mas intuitivamente queremos que os obstáculos não tenham partes longas e estreitas). Os obstáculos tendem a não ser muito próximos, no sentido de que o robô possa interagir com um número constante de obstáculos em uma porção do espaço físico ocupado pelo robô em um dado instante. Em DE BERG *et al.* [31] e [17, Capítulo 47] há referências para outros modelos realistas de cenas como entrada, tais como ambientes de baixa densidade de obstáculos.

Neste trabalho, temos uma hipótese que consideramos realista:

1. O robô R é relativamente pequeno comparado aos obstáculos: R é no máximo $b \cdot r$, onde b é alguma constante positiva e r é o menor raio de todas as hiperesferas minimais que circunscrevem pelo menos um obstáculo.

Nosso objetivo é descartar os obstáculos que os robôs que são de tamanho ilimitado.

Uma definição importante que será utilizada no restante deste capítulo é a notação de duplo fatorial apresentada abaixo.

Definição 4.1 (Duplo fatorial $n!!$)**.** O duplo fatorial de n é representado por $n!!$ e

definidio recursivamente por

$$n!! = \begin{cases} 1, n = 0 \text{ ou } n = 1. \\ n \cdot (n-2)!!, n > 1. \end{cases}$$

4.1 "Gordo"

Introduziremos nesta seção a definição de gordo que utilizaremos no restante do texto, obtida a partir de VAN DER STAPPEN *et al.* [32]. Para tanto, precisamos de um pouco de notação que será útil: as regiões esféricas d-dimensionais centradas em um ponto com localização arbitrária em um objeto E (no nosso caso, um obstáculo P).

Definição 4.2 (Bola fechada $S_{m,r}$)**.** Sejam M um espaço métrico com métrica ρ, X um subconjunto de M e m um ponto de X. O conjunto de pontos de X que estão a uma distância de x igual ou inferior a r é chamado de bola fechada centrada em m de raio r, ou seja:

$$S(m, r) = \{y \in X : \rho(m, y) \leq r\},$$

onde ρ denota a métrica de M.

Observe que é uma definição bem geral, mas no nosso contexto o espaço métrico M que utilizamos é o espaço Euclidiano. Daqui em diante, para evidenciar a diferença, chamaremos a bola fechada $S_{m,r}$ no espaço métrico Euclidiano de região esférica fechada $S_{m,r}$.

Definiremos a seguir um conjunto sobre um objeto E que consiste de todas as hiperesferas centradas dentro de E e não totalmente contendo E, necessário para a definição de um objeto k-gordo.

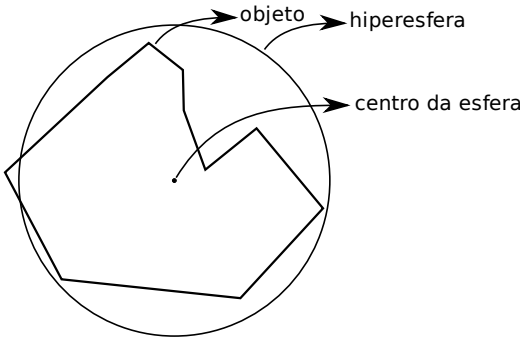
Definição 4.3 (U_E)**.** Sejam $E \subseteq \mathbb{R}^d$ um objeto e $r \in \mathbb{R}$. U_E consiste da união de todas as hiperesferas de raio r centradas dentro de E e não totalmente contendo E, isto é,

$$U_E = \bigcup_{m \in E} \left\{ S_{m,r} \subseteq \mathbb{R}^d \mid \exists r > 0, \partial S_{m,r} \bigcap E \neq \emptyset \right\}$$

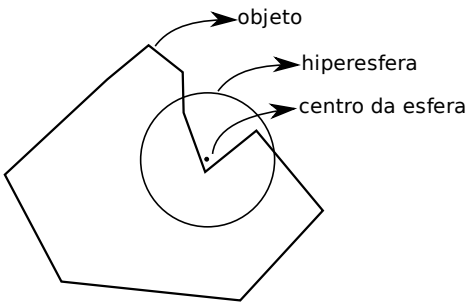
Agora, estamos aptos a definir um objeto k-gordo.

Definição 4.4 (k-gordo)**.** Seja $E \subseteq \mathbb{R}^d$ um objeto e $k \geq 1$, uma constante. O objeto E é k-gordo se para todas as hiperesferas $S \in U_E$

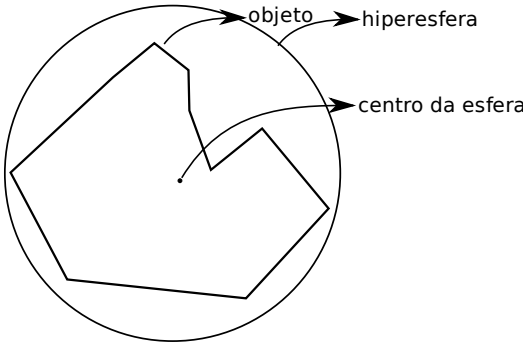
$$k \cdot volume(E \bigcap S) \geq volume(S)$$



(a) Hiperesfera $\in U_E$



(b) Hiperesfera $\notin U_E$



(c) Hiperesfera $\notin U_E$

Figura 4.1: Exemplos decorrentes da definição de U_E.

Em outras palavras, um objeto k-gordo E deve cobrir pelo menos $\frac{1}{k}$-ésimo de qualquer hiperesfera que seja centrada dentro de E e que não contenha E completamente (veja a Figura 4.1 para exemplos no espaço bidimensional).

A definição de k-gordo intuitivamente nos diz que para todo ponto $p \in E$, a vizinhança de p requer uma "alta densidade" de E, evitando que muitos outros objetos k-gordos de um certo tamanho mínimo estejam na vizinhança de p.

Na literatura existem diversas outras definições de k-gordo [33], mas consideraremos, no restante do texto, apenas a Definição 4.4.

Algumas observações podem ser feitas acerca da definição de k-gordo. Por exemplo, a utilização de regiões esféricas pode ser trocada por qualquer outra região compacta, tal como regiões limitadas por simplexos regulares, hipercubos fechados, etc.

Outras observações que podem ser feitas são acerca de propriedades, tais como: o limite inferior em k depende da dimensão e de que todo objeto k-gordo é k'-gordo, para $k' \geq k$. Sobre o limite inferior de k, note que podemos inferir, pela definição, que não podemos ter $k < 1$ em qualquer dimensão. Além disso, um objeto 5-gordo pode ser possível em 2 dimensões, mas é impossível em 3 dimensões. Esse exemplo é consequência do Teorema 4.1.

Teorema 4.1. *O limite inferior de k, de um objeto k-gordo, é dependente da dimensão d. Precisamente, $k \geq 2^d$.*

Demonstração. Suponha que temos um objeto k-gordo E de diâmetro δ, cujo volume é limitado superiormente pelo volume de uma hiperesfera de diâmetro δ. Como temos em E um diâmetro δ, isto significa que dois pontos de E, digamos p_1, p_2, estão a δ de distância entre eles. Note que a região esférica fechada $S_{p_1,\delta}$ pertence a U_E, pois $p_1 \in E$ e $p_2 \in \partial S_{p_1,r}$. Da mesma forma, a região esférica fechada $S_{p_2,\delta}$ pertence a U_E, pois $p_2 \in E$ e $p_1 \in \partial S_{p_2,r}$. Com ambas a observações, mostramos que U_E tem elementos de S com raio δ.

Temos até então:

$$volume(E \bigcap S) \leq volume(E). \tag{4.1}$$

$$Volume(E) \leq \begin{cases} \frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!} \cdot \frac{\delta}{2}^{\frac{d}{2}}, \ d \text{ par.} \\ \frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!} \cdot \frac{\delta}{2}^{d}, \text{ caso contrário.} \end{cases} \tag{4.2}$$

$$Volume(S) = \begin{cases} \frac{\pi^n}{n!} \cdot \delta^d, \ d \text{ par.} \\ \frac{2(2\pi)^n}{(2n+1)!!} \cdot \delta^d, \text{ caso contrário.} \end{cases} \tag{4.3}$$

A partir de 4.1, 4.2 e 4.3, temos que

$$volume(E \bigcap S) \leq volume(E) \leq 2^d \cdot volume(S).$$

Isto, em combinação com o fato de que $k \cdot volume(E \bigcap S) \geq volume(S)$, pela Definição 4.4, resulta em

$$k \geq 2^d.$$

. □

Em VAN DER STAPPEN e OVERMARS [30], foi provado que o número de objetos k-gordos intersectando uma determinada região, que não é muito maior comparado aos objetos, é constante. Segue o resultado no Lema 4.2.

Lema 4.2. *Sejam O um conjunto de objetos k-gordos em $\mathbb{R}^d$ em que a hiperesfera minimal que circunscreve cada objeto tenha raio pelo menos r e seja $c \in \mathbb{R}^+$ uma constante positiva. Se R for qualquer região em que a hiperesfera minimal que circunscreva essa região tenha raio $c \cdot r$, então o número de objetos $E \in O$ intersectando a região R é limitado superiormente pela constante $k \cdot (c+1)^d$.*

Demonstração. Sejam P_{min} o obstáculo com a hiperesfera minimal, digamos $S_{m,r}^{minimal}$, de todas as hiperesferas que circunscrevem os elementos de C e uma configuração q em que a região R esteja em contato com P_{min}. Veja que R fica completamente dentro de uma região esférica fechada $S_{m,r+c\cdot r}$. Note que m também é o centro de $S_{m,r}^{minimal}$.

Qualquer obstáculo, digamos P', que também tiver em contato com R terá interseção não-nula com a região esférica fechada $S_{m,r+c\cdot r}$. Logo, existe um ponto $m' \in P' \bigcap S_{m,r+c\cdot r}$. Seja a região esférica fechada $S_{m',r}$.

Vamos provar que $S_{m',r} \in U_{P'}$. Suponha, por contradição que $\partial S_{m',r} \bigcap P' \neq \emptyset$. Sabemos que $m' \in P'$, pois $m' \in P' \bigcap S_{m,r+c\cdot r}$. Como temos que $\partial S_{m',r} \bigcap P' \neq \emptyset$, isso implica que o objeto P' cabe todo dentro uma hiperesfera cujo raio é menor que r, o que é um absurdo.

Para cada obstáculo P' que tem interseção não-vazia com R, vamos construir uma região esférica fechada $S_{m',r}$. Todas essas regiões estarão totalmente dentro da região esférica fechada $S_{m,2\cdot r+c\cdot r}$. Note novamente que m também é o centro de $S_{m,r}^{minimal}$. Veja a Figura 4.2 para um exemplo bidimensional, com os ingredientes fornecidos nesta demonstração.

Mais a frente, precisamos do volume de uma hiperesfera I com raios de valor r, dado pela fórmula

$$volume(I) = \begin{cases} \frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!} \cdot \prod\limits_{i \in \{1,\ldots,d\}} r, \; d \text{ par.} \\ \frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!} \cdot \prod\limits_{i \in \{1,\ldots,d\}} r, \text{ caso contrário.} \end{cases}$$

Para concluir a prova, como cada obstáculo P' que tem interseção não-vazia com R é k-gordo e $S_{m',r} \in U_{P'}$, P' tem pelo menos uma parte do volume $\frac{1}{k} \cdot volume(S_{m',r})$

dentro de $S_{m,2\cdot r+c\cdot r}$, onde

$$\frac{1}{k}\cdot volume(S_{m',r}) = \begin{cases} \frac{1}{k}\cdot\frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!}\cdot(2\cdot r)^d, \text{ } d \text{ par.} \\ \frac{1}{k}\cdot\frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!}\cdot(2\cdot r)^d, \text{ caso contrário.} \end{cases} \tag{4.4}$$

Já o volume da região $S_{m,2\cdot r+c\cdot r}$, cujo diâmetro é limitado superiormente por $(c+1)(2\cdot r)$, é

$$volume(S_{m,2\cdot r+c\cdot r}) = \begin{cases} \frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!}\cdot((c+1)(2\cdot r))^d, \text{ } d \text{ par.} \\ \frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!}\cdot((c+1)(2\cdot r))^d, \text{ caso contrário.} \end{cases} \tag{4.5}$$

Desta forma, divindo a Equação 4.5 pela Equação 4.4, temos o limite superior de objetos $E \in O$ intersectando a região R:

$$\frac{volume(S_{m,2\cdot r+c\cdot r})}{volume(S_{m',r})} = k\cdot(c+1)^d$$

□

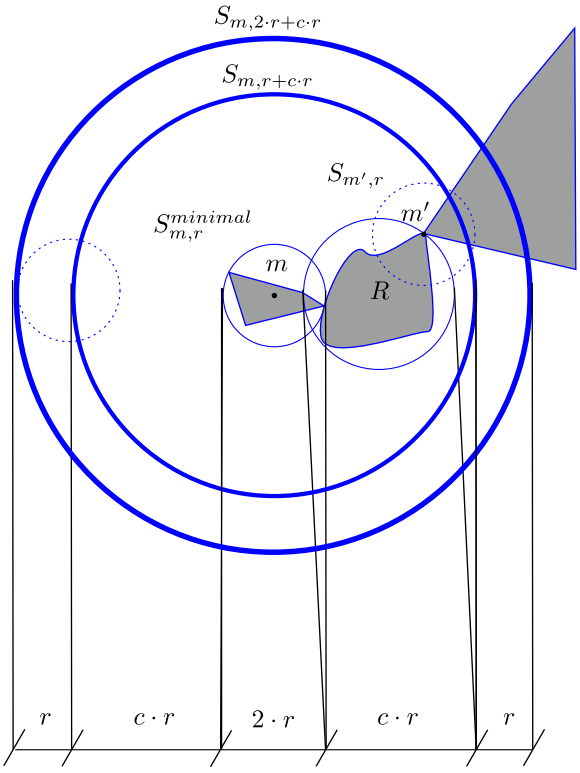


Figura 4.2: Exemplo dos ingredientes usados no Lema 4.2

4.2 Localização de pontos em objetos k-gordos

Utilizando amostragem baseada em grade, queremos resolver o problema de localização de pontos. Para tanto, vamos utilizar uma grade ortogonal regular, definida na Página 44. O objetivo desta seção é encontrar resolução de forma que o algoritmo de amostragem baseada em grade seja de resolução completa.

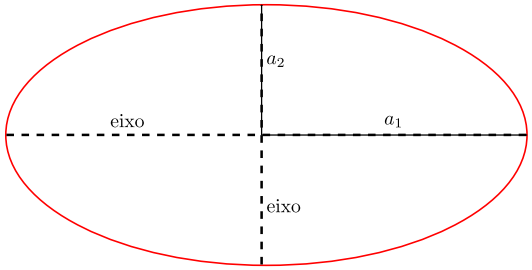
Será necessário a utilização de elipsóide e hiperesfera em posição padrão, que introduziremos a seguir.

Definição 4.5 (Elipsóide L em posição padrão)**.** Um elipsóide L na posição padrão é descrito usando a fórmula

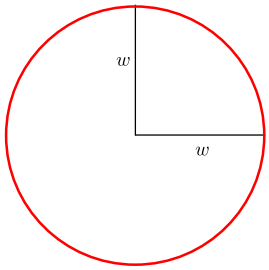
$$L : \sum_{i \in \{1,\ldots,d\}} \frac{x_i^2}{a_i^2} \leq 1, \tag{4.6}$$

onde $a_i, 1 \leq i \leq d$, são constantes.

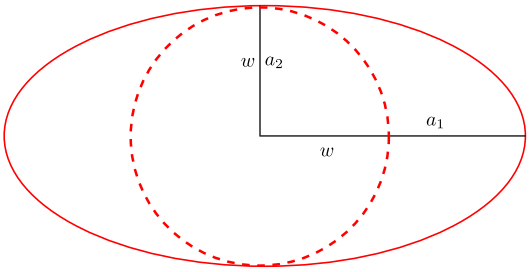
Veja a Figura 4.3(a) para um exemplo bidimensional de elipsóide.



(a) Elipsóide L



(b) Hiperesfera I



(c) Hiperesfera I contida na Elipsóide L

Figura 4.3: Elipsóides e hiperesferas

Definição 4.6 (Hiperesfera I em posição padrão)**.** Uma hiperesfera I na posição padrão é descrito usando a fórmula

$$I : \sum_{i \in \{1,\ldots,d\}} \frac{x_i^2}{w^2} \leq 1,$$

onde w é constante.

Veja a Figura 4.3(b) para um exemplo bidimensional de hiperesfera.

Se $w \leq a_i$, $1 \leq i \leq d$, então a hiperesfera I está contida no elipsóide L, como pode ser visto na Figura 4.3(c).

O volume de um elipsóide L é dado pela fórmula

$$volume(L) = \begin{cases} \frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!} \cdot \prod\limits_{i \in \{1,\ldots,d\}} a_i, \; d \text{ par.} \\ \frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!} \cdot \prod\limits_{i \in \{1,\ldots,d\}} a_i, \text{ caso contrário.} \end{cases} \tag{4.7}$$

Para encontrarmos a solução apropriada precisamos introduzir alguns resultados básicos. Começamos a busca pela resolução apropriada definindo para cada obstáculo P uma grande hiperesfera que esteja contida em P, de tal forma que pelo menos um ponto da grade atinja essa hiperesfera. Isto é facil usando a Propriedade 4.3.

Propriedade 4.3. *Sejam r a resolução da grade e d a quantidade de dimensões. Qualquer hiperesfera com raio pelo menos $\frac{r\sqrt{d}}{2}$ contém pelo menos um ponto da grade ortogonal regular.*

A hiperesfera será determinada mais adiante. Por enquanto, vamos focar nos elipsóides, que serão úteis na construção da hiperesfera. É possível identificar uma grande hiperesfera na parte convexa de um objeto. No nosso caso, como os obstáculos são k-gordos, seus elipsóides contém, de fato, grandes hiperesferas.

Para mostrar que um elipsóide L contém uma grande hiperesfera, um limite inferior no volume não é suficiente, pois é possível construir elipsóides longos e estreitos. Ao inserir um limite superior no diâmetro do elipsóide L, evitamos de construir tais elipsóides longos e estreitos.

Lema 4.4. *Sejam um elipsóide $L \subseteq \mathbb{R}^d$ com $volume(L) \geq V$ e δ um limite superior no diâmetro de L. Logo, L contém uma hiperesfera com raio pelo menos*

$$\begin{cases} \frac{V}{\left(\frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!} \cdot \frac{2}{\delta}^{d-1}\right)}, \; d \textit{ par.} \\ \frac{V}{\left(\frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!} \cdot \frac{2}{\delta}^{d-1}\right)}, \; \textit{caso contrário.} \end{cases}$$

Demonstração. Sem perda de generalidade, vamos utilizar a fórmula de um elipsóide L na posição padrão (Equação 4.6) e a fórmula do seu volume (Equação 4.7). O limite superior de δ no diâmetro implica $a_i \leq \frac{\delta}{2}$, $1 \leq i \leq d$.

Suponha por contradição que L contém uma hiperesfera com um raio, digamos a_j menor que

$$\begin{cases} \frac{V}{\left(\frac{\pi^{\frac{d}{2}}}{\frac{d}{2}!} \cdot \frac{2}{\delta}^{d-1}\right)}, \; d \text{ par.} \\ \frac{V}{\left(\frac{2(2\pi)^{\frac{d-1}{2}}}{(d)!!} \cdot \frac{2}{\delta}^{d-1}\right)}, \text{ caso contrário.} \end{cases}$$

Para facilitar a notação, seja γ_d a constante multiplicativa da fórmula de volume do elipsóide na d-ésima dimensão. Com o limite superior de δ no diâmetro, temos

$$\begin{aligned} volume(L) &= \gamma_d \cdot a_j \cdot \prod_{i \in \{1,\ldots,d\}, i \neq j} a_i \\ &< \gamma_d \cdot \frac{V}{\gamma_d} \cdot \left(\frac{2}{\delta}\right)^{d-1} \cdot \prod_{i \in \{1,\ldots,d\}, i \neq j} a_i \\ &\leq \gamma_d \cdot \frac{V}{\gamma_d} \cdot \left(\frac{2}{\delta}\right)^{d-1} \cdot \left(\frac{\delta}{2}\right)^{d-1} \\ &= V, \end{aligned}$$

o que é um absurdo, pois $volume(L) \geq V$. □

A Propriedade 4.3 com o Lema 4.4 são utilizados em OVERMARS e VAN DER STAPPEN [34] para encontrar uma resolução apropriada para que o algoritmo de amostragem em grade seja resolução completa, resultando no Lema 4.5.

Lema 4.5. *Sejam C um conjunto de objetos k-gordos convexos $E \subseteq \mathbb{R}^d$ com hiperesferas minimais que os circunscrevem tendo raio pelo menos r e $R \subseteq \mathbb{R}^d$ uma região com diâmetro $h \cdot r$, para alguma constante h. O conjunto $Q(R)$ de objetos $E \in C$ intersectando R pode ser encontrado por consultas de localização de pontos com pontos da grade ortogonal regular com resolução $2k^{-1}d^{-(d+\frac{1}{2})}r$ limitada pela diferença de Minkowski da região R com a região esférica fechada no zero euclidiano com raio r: $G(2k^{-1}d^{-(d+\frac{1}{2})}r) \bigcap (R \ominus S_{0,r})$.*

4.3 Método proposto

4.3.1 Busca de Região entre objetos gordos

Estrutura de Dados para a localização de pontos

A nossa idéia é utilizar uma quadtree compactada, definida na Página 11, para obter o resultado abaixo:

Teorema 4.6. *Um conjunto C de $O(n)$ objetos k-gordos de complexidade constante e disjuntos em $\mathbb{R}^d$ pode ser armazenado em uma estrutura de dados de tamanho $O(n)$, tal que, para um ponto de consulta $p \in \mathbb{R}^d$, podemos retornar em tempo $O(\log n)$ o objeto $E \in C$ que contenha p, ou vazio caso não exista tal objeto.*

Demonstração. Seja c a constante que define o tamanho máximo de uma lista de conflito, formalizada na Seção 1.2.3, Página 11, onde mostramos como a quadtree é construída. Depois de construída uma quadtree, para cada pai de folha, digamos $\square$, da árvore, vamos associar $\square$ ao menor objeto que tenha interseção com a região delimitada pelo hipercubo que $\square$ representa. Logo, temos uma função γ que leva de $\square$ para 1 objeto do nosso conjunto de obstáculos C. Precisamos provar que um determinado objeto é, utilizando a função γ, a imagem de um conjunto de pais das folhas, cuja cardinalidade é constante. Veja a Figura 4.4 para um exemplo bidimensional, com os ingredientes fornecidos nesta demonstração.

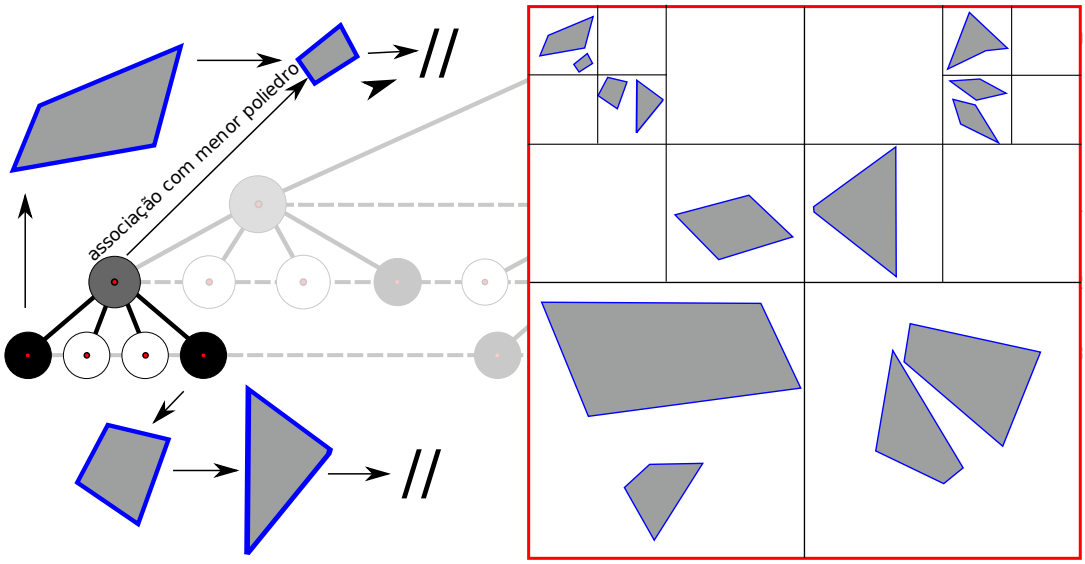


Figura 4.4: Exemplo dos ingredientes usados no Teorema 4.6

Atribuímos a c o limite superior de objetos que interceptam a região definida pelo hipercubo que $\square$ representa, digamos $\square_R$, dado pelo Lema 4.2, em que a região $\square_R$ tem a hiperesfera minimal que a circunscreve com raio δ. Temos pelo menos um objeto em que a hiperesfera minimal tenha raio menor que δ, pois se todos os objetos tivessem a hiperesfera minimal com raio maior ou igual a δ, poderíamos ter uma quantidade de objetos interceptando essa região menor ou igual a c e $\square$ não seria pai de folha. Logo, sabemos que o menor objeto certamente tem uma hiperesfera minimal que o circunscreva com raio menor que δ. Além disso, todas as folhas que se associam a esse objeto têm as hiperesferas minimais que os circunscrevem com raio maior ou igual a δ. Logo, o número de folhas com que o objeto tem interseção é constante.

Desta forma, podemos provar que o número de folhas é linear, pois cada objeto está associado a um número constante de folhas. Sejam κ_i a quantidade de folhas associadas ao objeto P_i, $1 \leq i \leq n$, e j o índice do objeto cuja quantidade de folhas associadas é a maior dentre os objetos de C. Logo, o número de folhas é dado por

$$\begin{aligned} \sum_{i=1}^{n} \kappa_i &\leq \sum_{i=1}^{n} \kappa_j \\ &= n \cdot \kappa_j \\ &= O(n). \end{aligned} \tag{4.8}$$

Como as listas de conflitos estão associadas às folhas, o tamanho total das listas de conflito é dado pelo número de folhas vezes o tamanho máximo da lista de conflito, onde já sabemos que o número de folhas é linear, ou seja, $O(n)$ e a lista de conflito é c, resultando em uma estrutura de dados de tamanho $O(n)$.

Agora, precisamos provar que o tempo de consulta é $O(\log n)$. Para tanto, vamos construir uma quadtree com hierarquia de separadores, definida na Página 12. $\square$

O resultado mais próximo que encontramos [34] desenvolveu uma estrutura de dados com o resultado presente no Teorema 4.7.

Teorema 4.7. *Um conjunto C de $O(n)$ objetos k-gordos de complexidade constante e disjuntos em $\mathbb{R}^d$ pode ser armazenado em uma estrutura de dados de tamanho $O(n \log^{d-1} n)$, tal que, para um ponto de consulta $p \in \mathbb{R}^d$, podemos retornar em tempo $O(\log^{d-1} n)$ o objeto $E \in C$ que contenha p, ou vazio caso não exista tal objeto.*

Portanto, nossa proposta reduz as complexidades de busca de $O(\log^{d-1} n)$ para $O(\log n)$ e de tamanho de $O(n \log^{d-1} n)$ para $O(n)$, onde n é a quantidade de vértices dos poliedros.

Busca de Região por localização de pontos

A estrutura de localização de pontos pode ser usada para resolver a busca de regiões em tempo de consulta na mesma ordem de complexidade da localização de pontos, no caso em que C seja um conjunto arbitrário de objetos convexos. A solução usa propriedades locais de objetos convexos, vistas no Lema 4.2. Isto torna possível descobrir qualquer objeto ou parte de objeto com um certo tamanho mínimo, sem a necessidade de saber sua localização exata, com pelo menos um ponto de um padrão suficientemente denso, mas não tão grande, de pontos amostrados. Isto não seria possível se os objetos não fossem gordos, pois a chance de atingir um segmento de reta por um padrão de pontos extremamente denso seria praticamente zero. Este padrão de pontos será uma grade ortogonal regular.

O próximo passo é encontrar a resolução da grade, tal que o menor subconjunto da grade ortogonal regular garantidamente atinja qualquer objeto E tendo intersecção não-vazia com a região de consulta R.

Na Seção 4.2, mostramos que uma resolução alta o suficiente que atinge todos os objetos E que pertencem à região de consulta é $G(2k^{-1}d^{-(d+\frac{1}{2})}r) \bigcap (R \ominus S_{0,r})$, que denotaremos por H.

Pelo Teorema 4.6, podemos fazer consultas de localização de pontos entre uma quantidade de objetos k-gordos com todos os pontos em H em tempo $O(|H| \log n)$. O resultado das consultas têm, no máximo, um número $|H|$ de elementos. Para cada elemento do resultado, há a verificação de interseção entre o obstáculo dado como retorno e o robô. Como ambos têm complexidade constante, a verificação da interseção é feita em tempo constante. Desta forma, o tempo total da computação é dominado por $O(|H| \log n)$, consequência das consultas de localização de pontos.

Precisamos saber qual a cardinalidade de H. A diferença de Minkowski $R \ominus S_{0,\rho}$ cabe completamente no hipercubo

$$[x_1^R - \rho, x_1^R + (h+1)\rho] \times \cdots \times [x_d^R - \rho, x_d^R + (h+1)\rho],$$

onde x_i^R, $1 \leq i \leq d$, é a x_i-coordenada minimal em R. Assim, o número de elementos

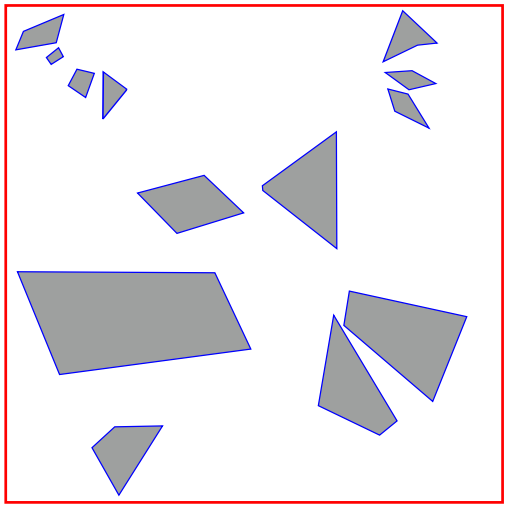
em H é limitado pelo número de pontos na grade que pertencem ao hipercubo que circunscreve a grade. A explicação desse parágrafo leva à Equação 4.9, que mostra que a cardinalidade de H é constante.

$$
\begin{aligned}
H &\leq |G(2k^{-1}d^{-(d+\frac{1}{2})}r) \bigcap \\
&\quad \underbrace{\underbrace{[x_1^R-\rho, x_1^R+(h+1)\rho]}_{(h+2)\rho} \times \cdots \times \underbrace{[x_d^R-\rho, x_d^R+(h+1)\rho]}_{(h+2)\rho}}_{((h+2)\rho)^d} | \qquad (4.9)\\
&= \frac{((h+2)\rho)^d}{(2k^{-1}d^{-(d+\frac{1}{2})}\rho)^d} \\
&= \left(\frac{1}{2}kd^{d+\frac{1}{2}}(\lfloor h \rfloor + 2)\right)^d \\
&= O((kd^dh)^d) \\
&= O(1), \text{ pois } k, h \text{ e } d \text{ são constantes.}
\end{aligned}
$$

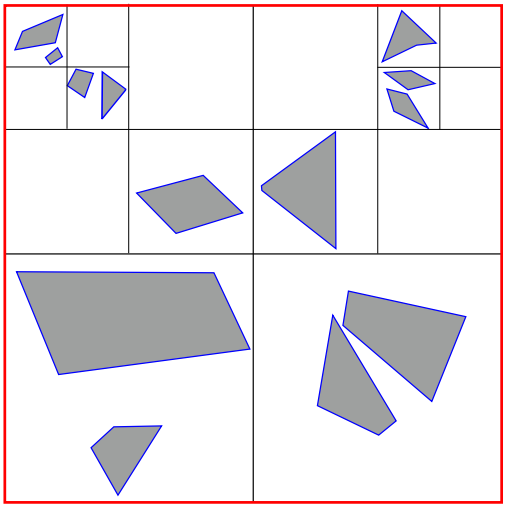
Como o tempo total da computação é dominado por $O(|H| \log n)$, consequência das consultas de localização de pontos e H tem cardinalidade constante, temos então que o tempo para fazer busca de região por localização de pontos é $O(\log n)$. Este resultado é resumido no Teorema 4.8.

Teorema 4.8. *Seja C um conjunto de objetos k-gordos convexos de complexidades constantes $E \subseteq \mathbb{R}^d$ com hiperesferas minimais que os circunscrevem tendo raio pelo menos r e seja $R \subseteq \mathbb{R}^d$ uma região com diâmetro $h \cdot r$, para alguma constante $h \geq 0$ e $k \geq 1$. Uma busca de região utilizando $R \subseteq \mathbb{R}^d$ de diâmetro no máximo $h \cdot r$ nos objetos em C executa em tempo $O(\log n)$.*

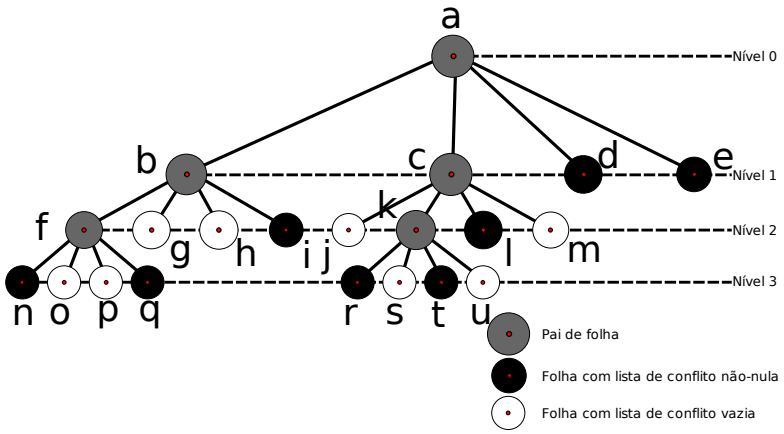
Para um exemplo do Teorema 4.8, veja o espaço de configuração na Figura 4.5(a). O primeiro passo é montar a quadtree sobre esse espaço de configuração C, ilustrado na Figura 4.5(b). A representação hierárquica da quadtree está ilustrada na Figura 4.5(c). O último passo é montar a hierarquia de separadores em cima da quadtree da Figura 4.5(c), resultando na Figura 4.5(d), para garantir as complexidades de consulta de localização de pontos.



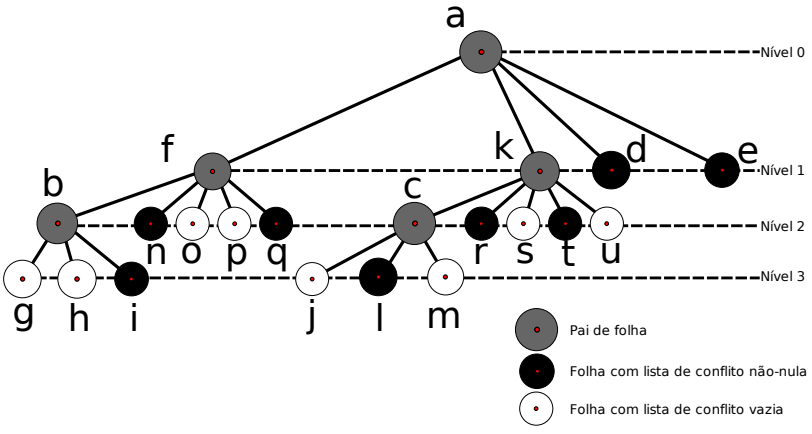
(a) Exemplo de espaço de configuração C



(b) Quadtree com $c = 2$ montada sobre o espaço de configuração C



(c) Representação hierárquica da quadtree



(d) Representação hierárquica da hierarquia de separadores sobre a quadtree

Figura 4.5: Exemplo de montagem da estrutura de dados para consulta de localização de pontos no espaço de configuração C

Agora, só precisamos utilizar o algoritmo de amostragem baseada em grade com os seguintes pontos para consultas de localização de pontos que representam uma região R: $G(2k^{-1}d^{-(d+\frac{1}{2})}r) \bigcap (R \ominus S_{0,r})$. Como os objetos são k-gordos, temos a garantia de que pelo menos um ponto sempre intersecta um poliedro. Assim, para qualquer região R que consultarmos, nós saberemos se essa região R colide com algum obstáculo. Dessa forma, sabemos sempre quando um posicionamento de um robô colide com algum obstáculo.

4.4 Considerações finais

Propomos o uso da hierarquia de separadores em cima da quadtree compactada utilizando a lista de conflito. No cenário realista de objetos "gordos", nossa proposta diminui as complexidades atuais na solução desenvolvida em OVERMARS e VAN DER STAPPEN [34], onde foi desenvolvida uma estrutura de dados com o resultado presente em VAN DER STAPPEN e OVERMARS [30]. A redução das complexidades de busca foram de $O(\log^{d-1} n)$ para $O(\log n)$ e de tamanho de $O(n \log^{d-1} n)$ para $O(n)$ na estrutura de dados, onde n é a quantidade de vértices dos poliedros.

Além disso, a solução reúne as melhores características das duas filosofias ao retornar a resposta correta em tempo finito, característica geralmente encontrado na filosofia da construção explícita, e funcionar bem em qualquer dimensão, característica geralmente encontrado na filosofia da amostragem.

Capítulo 5

Conclusão

Abordamos o problema de planejamento como a necessidade de converter as informações de um espaço físico W, ambiente em que temos objetos que devemos evitar colisão, na descrição de como movimentar-nos.

A formulação mais básica do problema de planejamento é conhecida como problema do movimento de piano. As aplicações do problema de planejamento fazem parte do nosso dia-a-dia, inclusive em casa: mover um móvel de um lugar para outro ou retirar um alimento da geladeira.

O material, da forma em que foi apresentado, cobriu os tópicos fundamentais relacionados ao tema e precisamente apresentando cada ingrediente utilizado na formulação e resolução do problema do planejamento. Entretanto, não nos prendemos apenas ao básico dos principais elementos que fazem parte do que envolve o problema de planejamento, aprofundamos em alguns pontos:

- No Capítulo 1, apresentamos as estruturas de dados utilizadas ao longo do texto.
- No Capítulo 2, fomos precisos sobre todo o desenvolvimento do algoritmo para resolver o problema de translação utilizando construção explícita do espaço bidimensional.
- No Capítulo 3, apresentamos a análise sobre o funcionamento do algoritmo de amostragem em um espaço de configuração C.
- No Capítulo 4, desenvolvemos uma nova forma de resolver, utilizando amostragem, o cenário realista em que os objetos são considerados gordos.

- Em cada capítulo, indicamos referências para o leitor se aprofundar em algoritmos mais eficientes, recentes e informações adicionais sobre os mais variados ingredientes que estão envolvidos diretamente com o problema e com as variadas soluções.

Discorremos sobre as duas principais filosofias do problema de planejamento. A filosofia da construção explícita e a filosofia da abordagem baseada em amostragem, que é dividida em determinística e probabilística.

A construção explícita do espaço de configuração C, trabalhado no Capítulo 2, reúne todas as informações dos objetos, codificando-as com o objetivo de obter um caminho livre de colisão eficientemente com base nas infomações disponíveis. Geralmente, os algoritmos que usam construção explícita são completos, retornando corretamente a solução, se ela existir.

A especificação dos elementos de entrada na filosofia de construção explícita, como a convexidade dos objetos ou a limitação de apenas um tipo de movimento, pode induzir a construção de algoritmos eficientes, elegantes e práticos do ponto de vista da implementação. Se não houver qualquer especificação dos elementos de entrada temos a dificuldade de conciliar o conceito de algoritmo completo e soluções eficientes, elegantes e práticas. Como geralmente os algoritmo de construção explícita não recorrem à aproximação, são mencionados como algoritmos exatos.

Em contraste com os métodos de construção explícita, os métodos baseados em amostragem, trabalhados no Capítulo 3, usam informações obtidas diretamente de um detector de colisão à medida que amostras são coletadas do espaço de configuração C.

Sobre a filosofia da amostragem, a primeira observação a ser feita é a independência sobre os elementos dados como entrada do problema. Agora, por exemplo, não é importante saber se o poliedro é convexo, pois o detector de colisão é uma caixa preta que responde se há colisão entre quaisquer elementos dados como entrada. A segunda observação é sobre a facilidade de obter a solução, pois com a amostragem temos uma abordagem bem mais simples que a construção explícita, além do desenvolvimento de bons algoritmos de detecção de colisão para construir algoritmos de planejamento capazes de serem eficientes para resolver instâncias do problema de planejamento com um número alto de graus de liberdade. Porém, al-

goritmos completos são mais difíceis de obter. É mais fácil obtermos algoritmos de resolução completa e/ou probabilisticamente completos.

A maioria dos algoritmos baseados em amostragem são de amostragem aleatória, que geralmente são probabilisticamente completos. Já os algoritmos de amostragem determinística geralmente são de resolução completa.

Para o cenário realista de objetos gordos, trabalhado no Capítulo 4, utilizamos a amostragem determinística. O interessante a ser mencionado sobre a abordagem é que o algoritmo obtido é completo, retornando a solução correta se existir. Não é necessário fazer a construção explícita do espaço, pois a restrição sobre a entrada do problema permite que possamos resolver o problema utilizando apenas a amostragem. Finalizamos em uma solução que reúne as vantagens do método baseado em amostragem aliada às vantagens que são mais comuns aos métodos de construção explícita. A solução para o problema também melhora as complexidades atuais da solução do problema, utilizando uma estrutura de dados bastante conhecida e de fácil implementação do que a presente na atual solução.

Na Seção 5.1 motivaremos o estudo das outras filosofias para o problema de planejamento e variações que podem ser melhoradas.

5.1 Trabalhos Futuros

O material dos capítulos anteriores concentra os esforços nas soluções baseadas na construção explícita e na abordagem baseada em amostragem. Campos potenciais constituem outra filosofia para resolver o problema do planejamento.

Na filosofia de campos potenciais, o espaço de configuração C é tratado como um campo potencial que combina a configuração final como a força atrativa e os obstáculos como as forças repulsivas. Na Figura 5.1, pode ser visto um exemplo. Temos vantagens e desvantagens com esse método. A vantagem é que podemos traçar o movimento com pouco custo computacional. A desvantagem é que o robô pode ficar preso em um mínimo local do campo potencial e não encontrar a solução. O algoritmo chamado campo potencial randomizado [35] utiliza o campo potencial como filosofia e passeio aleatório para evitar os mínimos locais quando o robô fica preso enquanto a busca está sendo executada.

O leitor interessado em como unir soluções de mais de uma filosofia, pode consultar em MAZER *et al.* [36] para um exemplo de um método utilizando amostragem e que aproveitou como inspiração idéias do campo potencial.

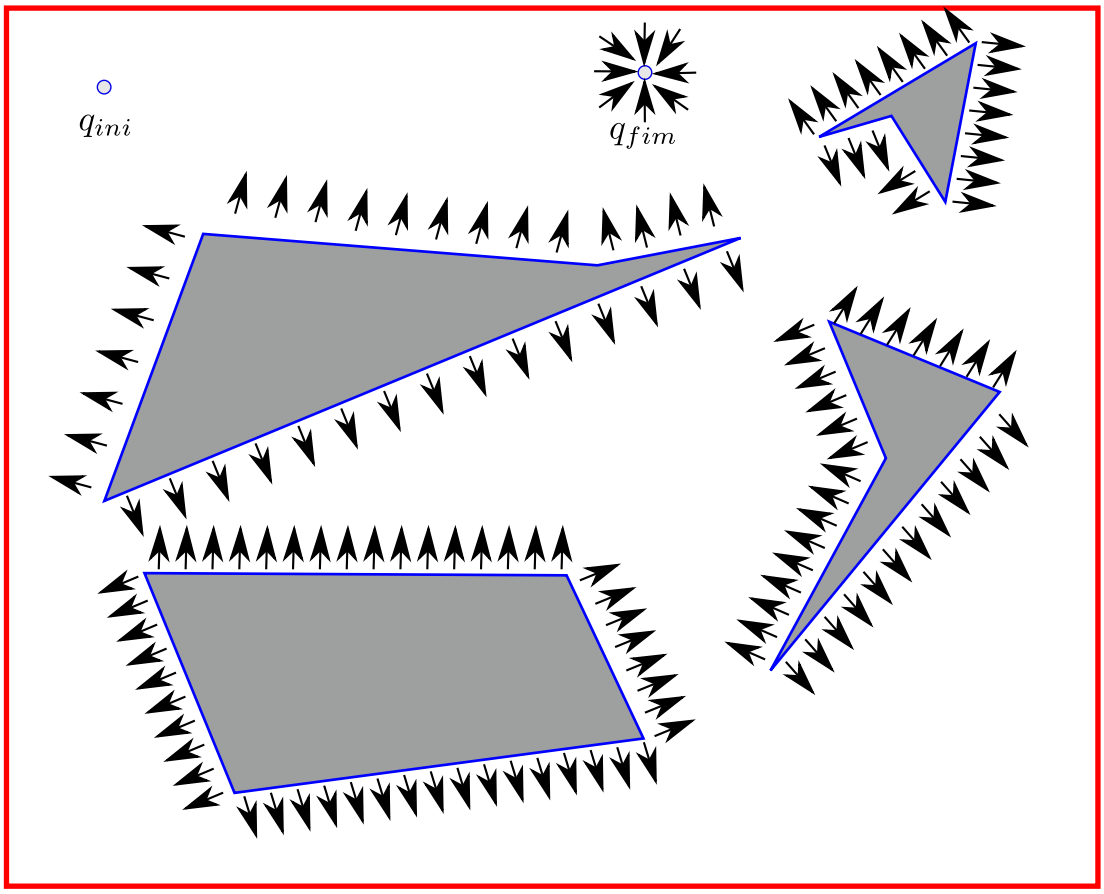


Figura 5.1: Exemplo utilizando a filosofia de campo potencial

Outro ponto a ser pesquisado é o estudo de mais variações do planejamento de movimento e buscar soluções para o problema em específico. Um exemplo bastante comum é o problema de planejamento que varia com o tempo. Vimos até agora um espaço físico estático, onde o robô pode se locomover enquanto os objetos permanecem parados, resultando em um espaço de configuração estacionário. Agora, imagine que os objetos podem se deslocar de acordo com o tempo, e não podemos ter a certeza do próximo estado do espaço de configuração C, apesar de sabermos o estado atual. Mais além, alguns próximos estados podem ser modelados probabilisticamente [37] utilizando conceitos da teoria de controles estocástica [38].

Referências Bibliográficas

[1] KOLTUN, V. "Pianos are not flat: Rigid motion planning in three dimensions". In: *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, p. 514. Society for Industrial and Applied Mathematics, 2005.

[2] SCHWARTZ, J., SHARIR, M. "On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers", *Communications on Pure and Applied Mathematics*, v. 36, n. 3, pp. 345–398, 1983.

[3] SEIDEL, R. "A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons", *Computational Geometry: Theory and Applications*, v. 1, pp. 51–64, 1991.

[4] MULLER, D. E., PREPARATA, F. P. "Finding the intersection of two convex polyhedra", *Theoretical Computer Science*, v. 7, n. 2, pp. 217–236, 1978.

[5] FINKEL, R., BENTLEY, J. L. "Quad trees: a data structure for retrieval on composite keys", *Acta Informatica*, v. 4, n. 1, pp. 1–9, 1974.

[6] SAMET, H. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, 2006.

[7] HAR-PELED, S. "Randomized incremental construction of compressed quadtrees", *CoRR*, v. abs/0907.0907, 2009.

[8] HAR-PELED, S. *Geometric approximation algorithms*.

[9] SIFRONY, S., SHARIR, M. "A new efficient motion-planning algorithm for a rod in two-dimensional polygonal space", *Algorithmica*, v. 2, n. 1, pp. 367–402, 1987.

[10] DE BERG, M., VAN KREVELD, M., OVERMARS, M., et al. *Computational Geometry: Algorithms and Applications*. Third ed. , Springer-Verlag, 2008.

[11] HSU, D., LATOMBE, J.-C., KURNIAWATI, H. "On the probabilistic foundations of probabilistic roadmap planning", *International Journal of Robotics Research*, v. 25, n. 7, pp. 627–643, 2006. ISSN: 0278-3649. doi: http://dx.doi.org/10.1177/0278364906067174.

[12] JAVGAL, P. S. *On the Minkowski sum of a terrain and a sphere*. Tese de Mestrado, Technische Universiteit Eindhoven, 2006.

[13] PACH, J., AGARWAL, P. K., SHARIR, M. In: *Surveys on Discrete and Computational Geometry - Twenty Years Later*, Providence, RI.

[14] CHAZELLE, B. "Triangulating a simple polygon in linear time", *Discrete & Computational Geometry*, v. 6, n. 5, pp. 485–524, 1991.

[15] BAUSU, S., POLLACK, R., ROY, M.-F. *Algorithms in Real Algebraic Geometry*. Springer-Verlag, 2006. ISBN: 978-3-540-33098-1, 3-540-33098-4.

[16] BAUSU, S., POLLACK, R., ROY, M.-F. *The Complexity of Robot Motion Planning*. Second ed. Cambridge, MA, MIT Press, 1987. ISBN: 0-262-03136-1.

[17] GOODMAN, J. E., O'ROURKE, J. *Handbook of Discrete and Computational Geometry*. Boca Raton, FL, Chapman & Hall/CRC, 2004.

[18] LAVALLE, S. M. *Planning Algorithms*. New York, NY, USA, Cambridge University Press, 2006. ISBN: 0521862051, 9780521862059. doi: 10.2277/0521862051.

[19] NIEDERREITER, H. *Random number generation and quasi-Monte Carlo methods*. Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, 1992. ISBN: 0-89871-295-5.

[20] LAVALLE, S. M., BRANICKY, M. S., LINDEMANN, S. R. "On the relationship between classical grid search and probabilistic roadmaps", *International Journal of Robotics Research*, v. 23, n. 7/8, pp. 673–692, July/August 2004. doi: 10.1177/0278364906067174.

[21] KAVRAKI, L., SVESTKA, P., LATOMBE, J., et al. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. Relatório técnico, 1994.

[22] GERAERTS, R., OVERMARS, M. H. "A comparative study of probabilistic roadmap planners". In: *In Proceedings Workshop on the Algorithmic Foundations of Robotics (WAFR'02)*, 2002.

[23] SIMÉON, T., LAUMAND, J.-P., NISSOUX, C. "Visibility-based probabilistic roadmaps for motion planning", *Advanced Robotics*, v. 14, n. 6.

[24] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al. *Introduction to Algorithms*. Third ed. Cambridge, Massachussets, The MIT Press, 2009. ISBN: 9780262533058, 9780262033848. doi: 10.1036/0070131511.

[25] RUSSELL, S., NORVIG, P. *Artificial Intelligence: A Modern Approach*. Second ed. Upper Saddle River, New Jersey, Pearson Education, 2003. ISBN: 0137903952.

[26] BOHLIN, R., KAVRAKI, L. E. "Path planning using lazy PRM". In: *IEEE Proceedings of International Conference on Robotics and Automation*, v. 1, pp. 521–528, 2000. ISBN: 0-7803-5886-4. doi: 10.1109/ROBOT.2000.844107.

[27] BRANICKY, M. S., LAVALLE, S. M., OLSON, K., et al. "Quasi-randomized path planning". In: *IEEE Proceedings of International Conference on Robotics and Automation*, v. 2, pp. 1481–1487, 2001. ISBN: 0-7803-6576-3.

[28] HSU, D., LATOMBE, J.-C., MOTWANI, R. "Path planning in expansive configuration spaces", *International Journal of Computational Geometry and Applications*, v. 9, n. 4/5, pp. 495–, 1999. Disponível em: ¡`http://citeseer.ist.psu.edu/article/hsu99path.html`¿.

[29] KAVRAKI, L. E., LATOMBE, J.-C., MOTWANI, R., et al. "Randomized query processing in robot path planning". In: *STOC '95: Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, pp. 353–362, New York, NY, USA, 1995. ACM. ISBN: 0-89791-718-9. doi: http://doi.acm.org/10.1145/225058.225159.

[30] VAN DER STAPPEN, A. F., OVERMARS, M. H. "Motion planning amidst fat obstacles (extended abstract)". In: *SCG '94: Proceedings of the Tenth Annual Symposium on Computational Geometry*, pp. 31–40, New York, NY, USA, 1994. ACM. ISBN: 0-89791-648-4. doi: http://doi.acm.org/10.1145/177424.177453.

[31] DE BERG, M., KATZ, M., OVERMARS, M., et al. "Models and motion planning", *Computational Geometry: Theory and Applications*, v. 23, n. 1, pp. 53–68, 2002.

[32] VAN DER STAPPEN, A. F., HALPERIN, D., OVERMARS, M. H. "The complexity of the free space for a robot moving amidst fat obstacles", *Computational Geometry: Theory and Applications*, v. 3, pp. 353–373, 1993.

[33] ALT, H., FLEISCHER, R., KAUFMANN, M., et al. "Approximate motion planning and the complexity of the boundary of the union of simple geometric figures", *Algorithmica*, v. 8, pp. 391–406, 1992.

[34] OVERMARS, M. H., VAN DER STAPPEN, A. F. "Range searching and point location among fat objects", *Journal of Algorithms*, v. 21, n. 3, pp. 629–656, 1996.

[35] BARRAQUAND, J., LATOMBE, J.-C. "A Monte-Carlo algorithm for path planning with many degrees of freedom". In: *IEEE Proceedings of International Conference on Robotics and Automation*, v. 3, pp. 1712–1717, 1990. ISBN: 0-8186-9061-5. doi: 10.1109/ROBOT.1990.126256.

[36] MAZER, E., AHUACTZIN, J. M., BESSIÈRE, P. "The Ariadne's clew algorithm", *International Journal of Artificial Intelligence*, v. 9.

[37] ZHOU, Y., CHIRIKJIAN, G. "Probabilistic models of dead-reckoning error in nonholonomic mobile robots". In: *IEEE Proceedings of International Conference on Robotics and Automation*, v. 2, pp. 1594–1599, 2003. ISBN: 0-7803-7736-2.

[38] KUMAR, P., VARAIYA, P. *Stochastic Systems: Estimation, Identification and Adaptive Control*. Prentice Hall.

Índice Remissivo