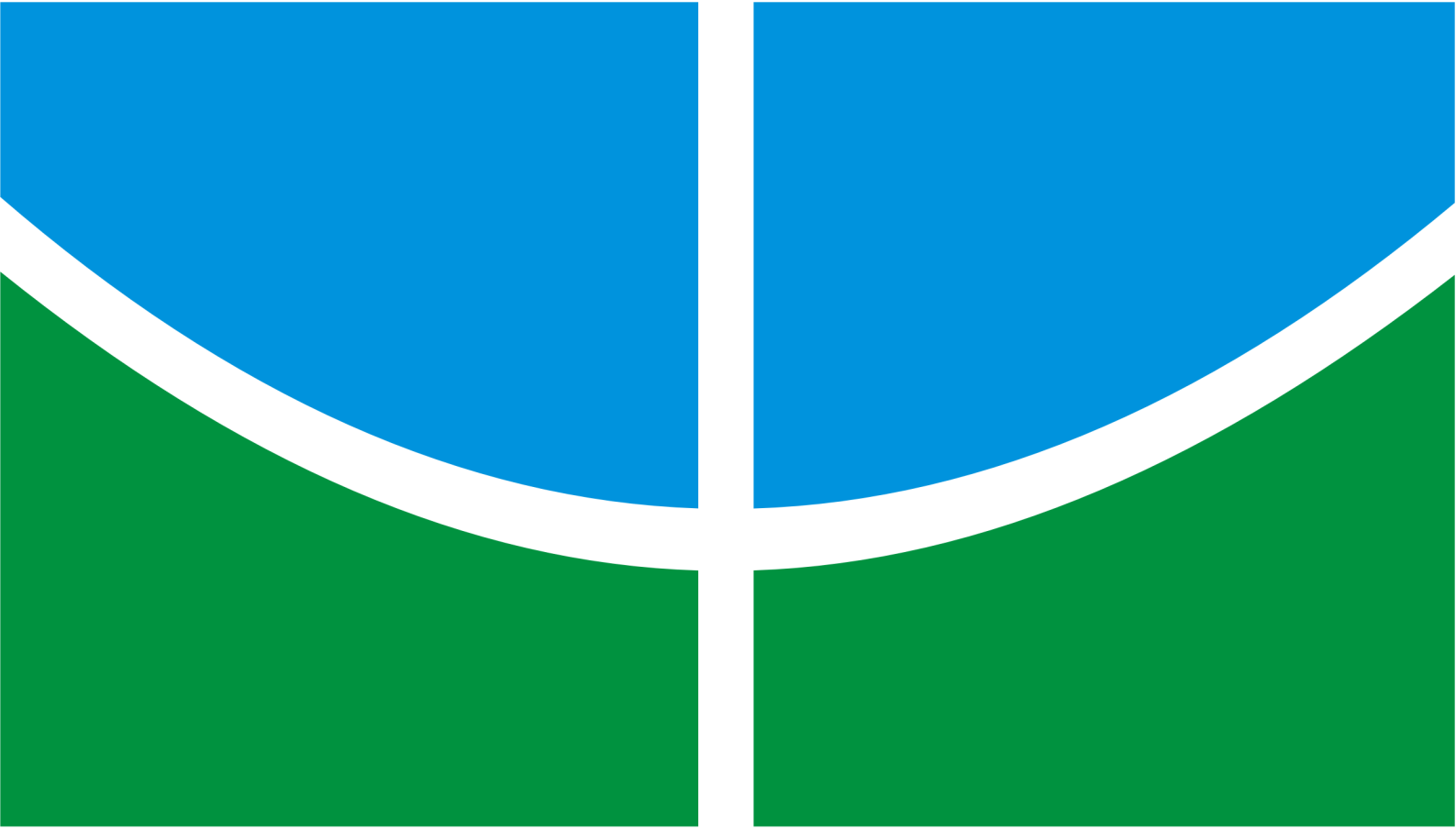


PLANEJAMENTO PROBABILÍSTICO DE ROTAS NO ESPAÇO DE CONFIGURAÇÃO E SUA APLICAÇÃO EM ROBÓTICA MÓVEL

BRUNO VILHENA ADORNO

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PLANEJAMENTO PROBABILÍSTICO DE ROTAS NO ESPAÇO DE CONFIGURAÇÃO E SUA APLICAÇÃO EM ROBÓTICA MÓVEL

BRUNO VILHENA ADORNO

DISSERTAÇÃO DE MESTRADO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA.

APROVADA POR:

_____

Prof. Geovany Araújo Borges, ENE/UnB
(Orientador)

_____

Prof. Guilherme Augusto Silva Pereira, DEE/UFMG
Examinador Externo

_____

Prof. Alexandre Ricardo Soares Romariz, ENE/UnB
Examinador Interno

BRASÍLIA, 05 DE SETEMBRO DE 2008.

FICHA CATALOGRÁFICA

ADORNO, BRUNO VILHENA
Planejamento probabilístico de rotas no espaço de configuração e sua aplicação em robótica móvel [Distrito Federal] 2008.
xi, 124p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2008).
Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica
1. Planejamento de Rotas 2. Espaço de Configurações
3. Robôs Holonômicos 4. Passeios aleatórios adaptativos
I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

ADORNO, B.V. (2008). Planejamento probabilístico de rotas no espaço de configuração e sua aplicação em robótica móvel, Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-351/08, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 124p.

CESSÃO DE DIREITOS

AUTOR: Bruno Vilhena Adorno
TÍTULO: Planejamento probabilístico de rotas no espaço de configuração e sua aplicação em robótica móvel.
GRAU: Mestre ANO: 2008

Bruno Vilhena Adorno
Departamento de Eng. Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

À minha filha Alice.

AGRADECIMENTOS

Esta foi a última parte da dissertação que foi escrita. É engraçado como ela será a primeira a ser lida e, mais engraçado ainda, é que a vejo como uma das mais importantes de todo o trabalho. Ela me faz lembrar das pessoas que me apoiaram durante todo o período do mestrado. Este foi um período bem intenso e, sem o apoio da família e dos amigos, ele poderia ter sido muito difícil. Assim, gostaria inicialmente de agradecer ao Professor Geovany pelo grande apoio na minha vida profissional, sendo a principal referência na minha carreira acadêmica. Sua competência, ética e cordialidade sempre me servirão de exemplo (Evidentemente, não poderia deixar de prestar meus agradecimentos aos sósias/clones do professor. Sim, isso mesmo. Como poderia ser explicado que ele saia do LARA às dezoito horas com algum material que lhe entreguei, responda aos meus e-mails às três horas da manhã, volte às oito da manhã do dia seguinte com o meu material respondido e, incrivelmente, tenha feito a mesma coisa com meus colegas do LARA?). Gostaria de agradecer ainda ao corpo docente do ENE pela minha formação, notadamente ao Professor Adolfo, sempre presente e prestativo, e ainda ao Professor Ishihara, que muito me influenciou no campo teórico. Agradeço à empresa Expansion Transmissão de Energia Elétrica S/A, que me financiou durante todo o período de mestrado.

Também não poderia me esquecer dos colegas do LARA e principalmente dos amigos da pós-graduação, que sempre foram companheiros tanto nos momentos de estudo, quanto nos momentos de discussões psico-sócio-filosóficas e também em alguns não tão raros momentos de lazer. Assim, gostaria de enfatizar os meus agradecimentos aos amigos Antônio, Alexandre e Ener, todos parceiros no projeto CARCARAH durante os dois anos de mestrado e que sempre foram ótimas companhias. As noites de pizza na casa do Alexandre sempre foram bons momentos fora do ambiente do laboratório, proporcionando oportunidades agradáveis para reencontrar os amigos. Além disso, o Antônio sempre foi um grande amigo dentro e fora do laboratório, inclusive estando na França. Sempre pude contar com ele. À Carlinha, que da França muito me ajudou. Ao Glauco, Mariana e Pedro, que apesar de trabalharem em projetos diferentes dos meus, sempre me ajudaram com idéias frutíferas e sempre estiveram dispostos a compartilhar seus conhecimentos. Aos amigos da capoeira, do grupo Beribazu, que sempre batalharam para manter um ambiente agradável e com muita energia. Principalmente ao Mestre Danilo, cujas aulas fizeram com que eu adorasse esta arte-luta. O seu empenho em manter a capoeira viva em todos seus alunos é admirável e fez com que eu me sentisse em casa toda vez que ia treinar.

Aos amigos de Goiânia, um agradecimento especial. Mesmo que eu passe anos sem vê-los, quando nos encontramos parece que tínhamos estado juntos no dia anterior. Ainda, não poderia me esquecer da Iolanda e Maria do Carmo, que muito me ajudaram.

Finalmente, mas não em último lugar, gostaria de prestar meus profundos agradecimentos aos familiares. Ao tio Rui e à tia Marilyn, que sempre agiram como se fossem meus pais, sendo sempre amáveis e carinhosos. Ao meu pai pelo grande apoio. Às minhas irmãs, que sempre pertenceram ao meu coração. Ao amigo e cunhado Daniel, pelas ajudas no violão e grandes partidas de Winning Eleven. Como não podia deixar de ser, um agradecimento mais que especial à minhas princesas Polyana e Alice. Sem a Polyana, não poderia ser quem sou. Já a Alice, aquela menininha do sorriso banguela que chegou para ficar, fez do meu mundo um mundo muito mais feliz.

RESUMO

PLANEJAMENTO PROBABILÍSTICO DE ROTAS NO ESPAÇO DE CONFIGURAÇÃO E SUA APLICAÇÃO EM ROBÓTICA MÓVEL

Autor: Bruno Vilhena Adorno

Orientador: Prof. Geovany Araújo Borges, ENE/UnB

Programa de Pós-graduação em Engenharia Elétrica

Brasília, 05 de setembro de 2008

Esta dissertação faz uma revisão e mostra a implementação dos principais métodos probabilísticos para planejamento de rotas no espaço de configurações de um robô e propõe um método incremental baseado em passeios aleatórios adaptativos. Visando a melhor compreensão do algoritmo proposto, é feita a análise teórica para uma classe de passeios aleatórios que não possuem homogeneidade espacial. A representação do problema no espaço de configurações permite tratar o robô como um ponto em um espaço abstrato, possibilitando a utilização de um mesmo algoritmo em robôs com diferentes formatos e ainda com diferentes modelos geométricos. A aplicação principal dos algoritmos abordados neste documento é em planejamento de rotas para robôs holonômicos que se movem em um plano. Porém, ela pode ser estendida para robôs com outros modelos geométricos e até mesmo sistemas multirrobôs. Uma avaliação por simulação do desempenho dos principais algorimos implementados é feita. Finalmente, é feita a comparação com o algoritmo proposto mostrando seu bom desempenho e boa qualidade das rotas resultantes.

ABSTRACT

PROBABILISTIC PATH PLANNING IN CONFIGURATION SPACE AND APPLICATIONS TO MOBILE ROBOTS

Author: Bruno Vilhena Adorno

Supervisor: Prof. Geovany Araújo Borges, ENE/UnB

Programa de Pós-graduação em Engenharia Elétrica

Brasília, 05th September 2008

This dissertation presents the implementation of the main probabilistic path planning algorithms in configuration space and presents a new incremental method based on adaptive random walks. In addition, a theoretical analysis about a class of random walks without spacial homogeneity is also presented. Configuration Spaces supply the framework necessary to represent the robot as a point in an abstract space. Thus, the path planning problem can be solved for robots with different shapes and geometric models. The analysis of simulated experiments presents the good perfomance of the proposed method over those presented in path planning literature.

SUMÁRIO

LISTA DE FIGURAS

Símbolos Latinos

$\mathcal{A}$	Robô poliédrico no espaço de trabalho
$\mathcal{C}$	Espaço de configuração
$\mathcal{CO}$	Obstáculos mapeados no espaço de configuração
$\mathcal{F}_{\mathcal{W}}$	Sistema de coordenadas fixo
$\mathcal{F}_{\mathcal{A}}$	Sistema de coordenadas anexado ao robô $\mathcal{A}$
$\mathcal{G}$	Grupo
$\mathbb{H}$	Grupo dos quatérnios
$\mathbf{H}$	Matriz de transformação homogênea
$\mathcal{O}$	Obstáculos no espaço de trabalho
q	Configuração em um espaço não especificado
$\mathbf{q}$	Configuração representada por um vetor em $\mathbb{R}^n$
$\mathbf{R}$	Matriz de rotação
$\mathbb{R}^n$	Espaço Euclidiano n-dimensional
$\mathcal{S}$	Espaço topológico
$\mathbb{S}^1$	Grupo do círculo
$SO\left(n\right)$	Grupo especial ortogonal de ordem N
$SE\left(n\right)$	Grupo especial euclidiano de ordem N
$\mathcal{T}$	Árvore de expansão
U	Potencial
$\mathcal{W}$	Espaço de trabalho
$\mathcal{X}$	Espaço de estados
$\mathbb{Z}^n$	Espaço n-dimensional dos números inteiros

Símbolos Gregos

Δ	Variação entre duas grandezas similares
$\mu(\cdot)$	Volume de um espaço
∇	Gradiente
ζ	Parâmetro de escala

Subscritos

att	atração
$free$	espaço livre
obs	obstáculos
$rand$	amostra aleatória
rep	repulsão
$start$	inicial
$goal$	destino

Sobrescritos

$\cdot$	Variação temporal
$-$	Valor médio

Siglas

ARW	Adaptive Random Walking
CAD	Computer Aided Design
EST	Expansive-Space Trees
iARW	Incremental Adaptive Random Walking
LARA	Laboratório de Robótica e Automação
LIRMM	Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
PFO	Planejador por Frente de Onda
PRM	Probabilistic Roadmap
RPP	Randomized Potential-Field Planner
RRT	Rapidly-Exploring Random Trees

Neste trabalho vetores são representados por letras minúsculas em negrito. Matrizes são representadas por letras maiúsculas em negrito. Já espaços e conjuntos em geral são representados por letras maiúsculas caligráficas. Por exemplo, tem-se o vetor linha $\mathbf{v} = [v_x,\ v_y]$, a matriz

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \tag{1}$$

e o conjunto $\mathcal{S}$.

Uma atenção especial deve ser dada às configurações q. Quando elas são representadas explicitamente por um vetor coluna, escreve-se $\mathbf{q}$. Quando elas representarem uma configuração em um espaço não especificado, escreve-se simplesmente q.

1 INTRODUÇÃO GERAL

A journey of a thousand miles
starts with a single step.
Lao-tsu

Em robótica móvel, três são as tarefas comumentes atribuídas aos robôs: sensoriamento e estimação; planejamento; e execução [1].

Um exemplo de tarefa de sensoriamento e estimação é a de localização e mapeamento simultâneos (SLAM, do inglês *Simultaneous Localization and Mapping*), na qual o robô recebe dados de diversos sensores e, após processamento adequado, estima sua localização no ambiente e o representa por meio de um mapa [2].

O planejamento consiste em determinar o conjunto de ações a serem tomadas para atingir um determinado objetivo. Por exemplo, um plano para que um robô saia de uma sala cuja porta está fechada pode ser: ir até a porta; girar a maçaneta; abrir a porta; e sair.

A etapa de execução consiste em "colocar em prática" o que foi planejado. No caso do robô cujo objetivo é sair da sala, o ato de se locomover até a porta, segurar a maçaneta e girá-la e, então, efetivamente se locomover para o lado de fora, constitui o ato de executar o plano.

Este trabalho tratará apenas da tarefa de planejamento de rotas, cuja descrição mais detalhada vem a seguir.

1.1 DEFINIÇÃO DO PROBLEMA

De acordo com o dicionário Aurélio da língua portuguesa, o termo planejar significa *"1. Fazer o plano ou a planta de; projetar, traçar"*, enquanto que o termo rota significa *"1. Caminho, rumo"*. Assim, planejamento de rotas pode ser considerado como o ato de planejar um caminho para ir de um lugar a outro.

O problema de planejamento de rotas no contexto de robótica consiste em, dadas as configurações inicial e final de um robô, descobrir uma seqüência de movimentos a ser executada pelo robô para que ele saia da primeira e chegue à segunda sem colidir com obstáculos. Assim, a configuração q de um robô representa a localização e orientação de todos os seus pontos no espaço de trabalho $\mathcal{W}$ e o espaço de configurações $\mathcal{C}$ representa o conjunto de todas as suas configurações. A dimensão do espaço de configurações é igual ao número de variáveis independentes utilizadas para representar a configuração do robô. Por exemplo, um robô holonômico (*i.e.*, que não possui restrições de movimento) com formato retangular no

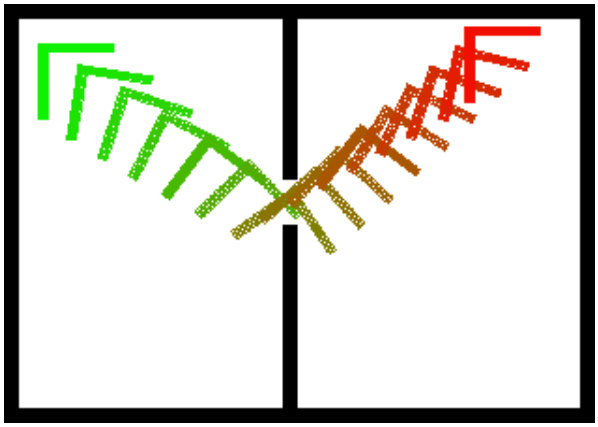


Figura 1.1: Configurações inicial e final à esquerda e à direita, respectivamente. $\mathcal{C}$ é tridimensional e $\mathcal{W}$ é bidimensional.

plano (x, y) possui três graus de liberdade, sendo dois para translação e um para orientação. Desta forma, seu espaço de configurações tem três dimensões e o espaço de trabalho em que ele se encontra é bidimensional.

O planejamento de rotas, em geral, ocorre no espaço de configurações do robô, e não no espaço de trabalho $\mathcal{W}$. É exigida assim, uma boa formulação para o problema, pois o alcance da solução pode consistir na busca em um espaço de dimensão arbitrária e muitas vezes não-Euclidiano.

De uma forma geral, a solução para o problema de planejamento de rotas é uma função contínua π tal que $\pi : [0, 1] \rightarrow \mathcal{C}_{free}$, em que $\pi(0) = q_{start}$, $\pi(1) = q_{end}$, $\mathcal{C}_{free}$ corresponde à porção do espaço de configurações que está livre de obstáculos e q_{start} e q_{end} correspondem às configurações inicial e final do robô, respectivamente [2].

A Figura 1.1 mostra um exemplo típico de planejamento de rotas, em que um robô com formato em L sai da configuração inicial no lado esquerdo e chega na configuração final no lado direito, passando por uma estreita passagem no meio do espaço de trabalho. As configurações intermediárias estão livres de colisão e portanto pertencem a $\mathcal{C}_{free}$.

1.2 PRINCIPAIS DIFICULDADES

Uma das maiores dificuldades encontradas no problema de planejamento de rotas é que sua solução requer um tempo que cresce exponencialmente com o número de graus de liberdade do robô [3], sendo que as soluções determinísticas e completas se mostraram computacionalmente impraticáveis para robôs com elevado número de graus de liberdade [3, 4]. Sendo assim, um dos focos das pesquisas tem sido em abordagens que aproximam o espaço de configurações livre por meio de amostras do mesmo, sendo que muitas das técnicas desenvolvidas desde a década de 90 se mostraram computacionalmente eficientes [5, 4, 6, 7, 8, 9].

As maiores dificuldades relacionadas ao problema de planejamento de rotas que aproximam $\mathcal{C}_{free}$ por meio de amostras são:

- a representação do problema em uma forma geral não é trivial e a determinação de uma representação eficiente de espaços de configurações mais complexos pode ser uma tarefa árdua. Notadamente, projetar planejadores que tenham bom desempenho computacional, produzam rotas com boa qualidade para uma ampla classe de espaços de configuração e que não sejam dependentes do espaço de trabalho é um grande desafio;

- passagens estreitas em $\mathcal{C}_{free}$ podem afetar consideravelmente o desempenho destes algoritmos. Sendo assim, muito esforço tem sido empreendido no sentido de amostrar mais seletivamente $\mathcal{C}_{free}$ [10, 11, 12, 13, 14];

- como a busca da solução consiste em realizar uma exploração no espaço de configurações, encontrar métodos eficientes para esta exploração ainda é um desafio, sendo que vários algoritmos foram projetados visando uma exploração uniforme de $\mathcal{C}_{free}$ [15, 6, 7, 4];

- como o problema de planejamento de rotas pode ser aplicado a robôs ou objetos com espaço de configurações com uma topologia arbitrária, definir métricas de distância e amostrar configurações aleatórias nestes espaços pode não ser trivial;

- finalmente, mas longe de ser a última dificuldade existente, geralmente os algoritmos baseados em amostragem probabilística do espaço de configurações geram rotas que possuem muitas redundâncias e ciclos desnecessários, havendo então a necessidade de suavização, o que nem sempre é fácil tendo em vista a topologia do espaço de configurações [16].

1.3 ESTADO DA ARTE EM PLANEJAMENTO DE ROTAS

O planejamento de movimento em robótica móvel é uma área consolidada e resultados significativos já foram obtidos tanto do ponto de vista teórico quanto de resultados práticos. O problema clássico conhecido como “problema do móvel generalizado” é tratado nos trabalhos de John Reif [17, 18] . Na análise mais básica, dado um poliedro inserido em um espaço euclidiano de duas ou três dimensões contendo obstáculos poliédricos, ele tem que ser levado de um ponto inicial e até um ponto final sem entrar em contato com os obstáculos [17]. Na generalização do problema, o móvel é substituído por um conjunto de poliedros ligados entre si [18]. Do ponto de vista prático, várias abordagens práticas são apresentadas em [19, 2], sendo as abordagens principais, no que se refere aos algoritmos probabilísticos, revisadas no Capítulo 3.

Devido ao bom desempenho dos algoritmos de planejamento de rotas desenvolvidos nos últimos anos, a sua utilização extrapolou o universo da robótica móvel e alcançou outras áreas de pesquisa, tais como animação de atores artificiais, biologia molecular, problemas de montagens complexas, planejamento de movimento para objetos flexíveis, planejamento multirrobôs e planejamento em ambientes com obstáculos que se movem [2].

A seguir é apresentado um levantamento do estado da arte de acordo com cada um desses segmentos.

1.3.1 Robótica móvel

Em robótica móvel terrestre, várias abordagens de planejamento de rotas foram implementadas com sucesso. Uma técnica que utiliza campos de potencial artificiais para robôs com vários graus de liberdade e que escapa de mínimos locais espúrios através de passeios aleatórios é apresentada em [5]. Uma abordagem que utiliza campos eletrostáticos artificiais e que resulta em apenas um mínimo localizado no destino é apresentada em [20].

Existem também abordagens que amostram o espaço de configurações e criam uma estrutura abstrata para representar as configurações livres de obstáculos, resolvendo de forma eficiente o problema de planejamento de rotas para ambientes estáticos [4, 6, 7, 8, 9]. Contudo, extensões em alguns desses algoritmos permitiram o seu uso em ambientes dinâmicos [21].

Em [22] é apresentado um algoritmo genético que utiliza cromossomos de tamanho variável e alguns operadores especializados que incorporam um conhecimento do espaço de trabalho, o que aumenta a velocidade de convergência do algoritmo quando comparado com algoritmos genéticos baseados apenas nos operadores clássicos.

Em robótica móvel existem ainda as abordagens ditas robustas. Nestas, em geral, o planejamento é feito em um espaço abstrato $\mathbb{I}$, chamado espaço de informações. O estado do ambiente não é conhecido e a única informação disponível e que compõe o espaço de informações é dada pelo histórico das observações dos sensores, as ações que foram aplicadas e as condições iniciais. A abordagem clássica para resolver este tipo de problema é usar toda a informação disponível para estimar o estado atual do sistema. Entretanto, em muitos casos é possível resolver a tarefa completamente apenas utilizando o espaço de informações [19].

Um exemplo de planejador robusto é apresentado em [23] e usa Processos de Decisões Markovianos para modelar um robô e sua interação com o ambiente. Assim, o processo é definido como uma quádrupla $< S, A, T, R >$ em que S é um conjunto finito dos estados que caracterizam o ambiente; A é um conjunto finito de ações que permite a transição entre os estados; T é a função de transição que determina a probabilidade de ir de um estado s_1 para um estado s_2 quando uma ação a é executada; e R é uma função de recompensa usada para especificar o objetivo a ser alcançado e as áreas perigosas do ambiente. A robustez é alcan-

çada ao minimizar as incertezas tanto na geração quanto na execução da rota, minimizando efeitos indesejáveis, como derrapagens, ao executar movimentos de rotação.

Um exemplo de aplicação em robótica móvel aérea consiste no planejamento de rotas para um veículo aéreo não tripulado em miniatura e de asa fixa [24]. Esta aplicação é particularmente difícil, pois além de haver grandes restrições dinâmicas, o sistema embarcado não possui grande poder de processamento por restrições na capacidade de carga do veículo.

Sendo assim, como o sistema não faz mapeamento, a informação sobre o ambiente é retirada de uma base de dados que não necessariamente contém a informação sobre todos os obstáculos. A rota é gerada a partir de um planejador baseado na amostragem do espaço de estados [15]. Um sistema de desvio local de obstáculos é necessário no momento da execução da trajetória para evitar objetos não catalogados na base de dados.

Na aplicação em sistemas multirrobôs, além de ser necessário encontrar rotas para cada robô, existe ainda a necessidade de garantir que dois robôs não entrem em colisão ao executarem a rota. Uma solução direta é a abordagem centralizada, na qual o espaço de configurações resultante é o produto cartesiano [1] do espaço de configurações de cada robô. A dimensão resultante será igual à soma das dimensões de cada espaço de configurações. Esta abordagem pode ocasionar uma grande dimensionalidade do espaço de busca final, porém o algoritmo utilizado neste caso não difere para o caso em que há apenas um robô.

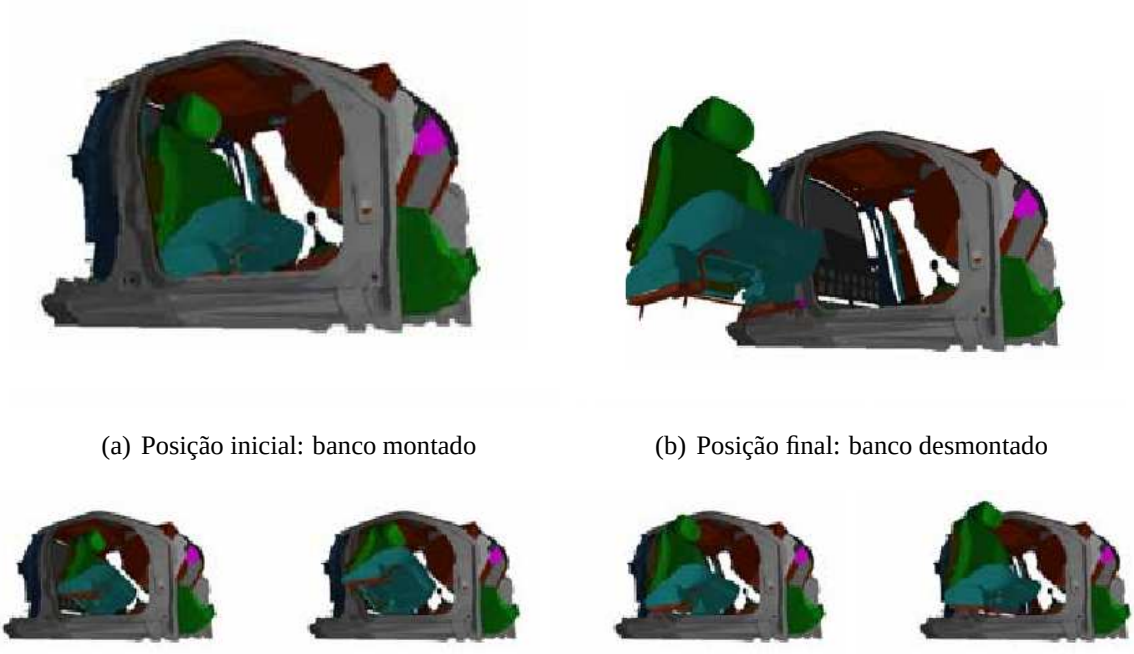
Já na abordagem desacoplada não há o aumento da dimensionalidade. Em uma primeira etapa são achadas rotas livres para cada robô e em uma segunda etapa calcula-se a velocidade relativa de cada robô ao longo da rota evitando colisão entre eles. O grande problema desta abordagem é que, mesmo que as duas etapas sejam completas [2], o algoritmo resultante não é completo, ou seja, pode ser que exista uma solução para o problema, mas as rotas geradas na primeira etapa poderiam gerar colisões entre dois robôs quando estes fossem executá-las. Neste caso, o algoritmo retornaria a ausência de solução, quando na verdade ela existia e o algoritmo que não foi capaz de encontrá-la. Alternativamente, cada robô pode ser processado individualmente em uma ordem pré-definida, sendo que cada robô passa a ser considerado um obstáculo que se move assim que sua rota é calculada [2].

1.3.2 Problemas de montagem e desmontagem

O objetivo do planejamento de rotas em problemas de montagem e desmontagem é testar a remoção e montagem de várias partes de um conjunto de peças com objetivo de fazer manutenção das mesmas [19]. Este tipo de algoritmo permite a utilização de modelos em programas computacionais específicos para projetos de peças e mecanismos (programas CAD,

[1] O produto cartesiano entre dois conjuntos $\mathcal{P}$ e $\mathcal{R}$ é o conjunto de todos os pontos (p, r), tal que $p \in \mathcal{P}$ e $r \in \mathcal{R}$.

[2] Um algoritmo é dito completo quando ele retorna a solução em tempo finito, caso ela exista ou, caso contrário, reporta a não existência da solução.



(a) Posição inicial: banco montado

(b) Posição final: banco desmontado

(c) Seqüência de movimentos para retirada do banco

Figura 1.2: Problema de montagem e desmontagem na indústria automobilística [25].

do inglês *Computer Aided Design*) em vez do teste em modelos físicos reais, diminuindo o custo e tempo de desenvolvimento. De fato, existem softwares usados pela indústria automobilística para resolver este tipo de problema.

No que se refere à montagem de carros, já existem células de manufatura que são projetadas em CAD, sendo que os modelos geométricos gerados em software são usados para planejamento de rotas. Em algumas tarefas relacionadas à selagem de automóveis (processo na qual uma substância é aplicada às frestas do veículo de forma a evitar a entrada de poeira e água) os engenheiros fornecem apenas comandos de alto nível para os robôs, que então calculam automaticamente seus movimentos.

A Figura 1.2 mostra um exemplo de resolução do problema existente na indústria automobilística de retirar um banco do interior de um automóvel [25]. A Figura 1.2(a) mostra o banco dentro do carro e a Figura 1.2(b) o mostra na posição final. A seqüência de movimentos que possibilita a desmontagem do banco é mostrada na Figura 1.2(c).

1.3.3 Animação de atores artificiais

O uso de algoritmos de planejamento de rotas em animação de atores artificiais permite o uso de comandos em alto nível, evitando tarefas tediosas como animação quadro a quadro [19]. Além disso, esses algoritmos se aplicam ao planejamento de movimento para manipulação de objetos, em que o objetivo é mover alguns objetos na cena de uma configuração inicial estável até uma configuração final também estável, evitando colisão com objetos na

cena.

Em [26] é mostrada a implementação baseada na amostragem do espaço de configurações por meio de uma árvore que explora de maneira eficiente o espaço de configurações livre e que, utilizando atalhos, forma um grafo utilizado para gerar movimentos para um ator artificial de 22 graus de liberdade. A amostragem é polarizada de forma a favorecer as melhores posturas para tarefas de manipulação de objetos e o mapa de rotas resultante ainda é atualizado dinamicamente quando os obstáculos mudam de posição. A Figura 1.3(a) mostra um exemplo de planejamento de movimento para um humanóide em um escritório virtual cuja mesa está cheia de objetos. A Figura 1.3(b) mostra a representação espacial dos nós gerados no espaço de configurações. Por último, a Figura 1.3(c) mostra o mapa de rotas com os nós e arcos representados no espaço de trabalho.

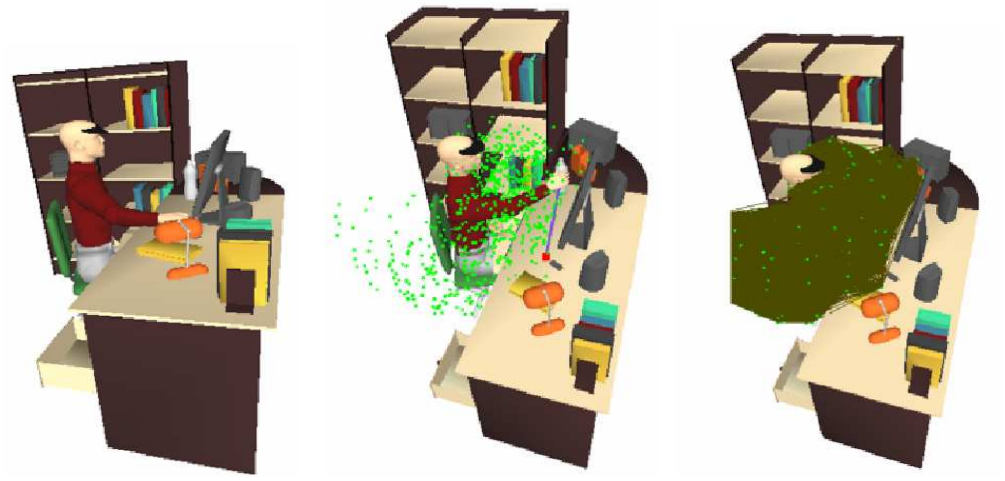
1.3.4 Planejamento de rotas para robôs humanóides

Uma abordagem para planejamento de movimento para robôs humanóides consiste em virtualizar o espaço de trabalho e, então, fazer o planejamento usando informações do ambiente virtual. Porém, a execução do movimento ocorre no espaço de trabalho real [27]. A árvore de exploração faz uma busca no espaço das configurações que são estaticamente estáveis para achar uma solução que também esteja em $\mathcal{C}_{free}$. Uma vantagem dessa abordagem é que o planejamento pode ser feito de maneira semelhante ao da animação de atores artificiais, pois o ambiente é virtualizado. Contudo, o espaço de busca é modificado, pois existem restrições relacionadas ao equilíbrio do humanóide real.

1.3.5 Aplicação em biologia computacional

Os algoritmos de planejamento de rotas também são utilizados em aplicações de *docking* e dobramento de proteínas. O problema de *docking* é particularmente difícil no que se refere à dimensionalidade do espaço de configurações, pois existem milhares de graus de liberdade [29]. Neste problema o objetivo é determinar se uma molécula flexível pode ser inserida dentro da cavidade de uma proteína, obedecendo limitações como baixa energia. A vantagem da simulação de *docking* de proteínas pode ser encontrada no desenvolvimento de novos medicamentos, uma vez que testes preliminares podem ser feitos em simulação e somente os componentes com melhores resultados são efetivamente testados, havendo uma economia de tempo e recursos financeiros [2].

No caso do estudo de dobramento de proteínas, o objetivo é caracterizar a seqüência de movimentos seguidos por uma proteína para que ela saia de uma forma desorganizada até uma configuração altamente ordenada. O caminho que a proteína segue para alcançar a sua configuração nativa é difícil de ser determinado experimentalmente porque geralmente os passos intermediários acontecem muito rapidamente, sendo de difícil detecção [2].



(a) Ator artificial (b) Representação das configurações geradas (c) Mapa de rotas resultante

Figura 1.3: Exemplo de planejamento de movimento para um ator artificial com 11 graus de liberdade [28].

Algoritmos comumente aplicados em robótica móvel têm sido utilizados para análise do movimento molecular de maneira eficaz. Em [30] um mapa de rotas é utilizado para representar as transições de várias configurações de proteínas. Desta forma, cada nó é amostrado aleatoriamente e representa um arranjo molecular. Cada borda direcionada ligando dois nós possui um peso que é a probabilidade da molécula sair do nó de origem da borda (arranjo inicial) até o nó de destino (arranjo final), sendo esta probabilidade dependente da diferença de energia entre os dois arranjos moleculares. Sendo assim, o grafo contém vários caminhos com probabilidades associadas, permitindo a análise do movimento de proteínas em todo o "cenário de energia".

1.3.6 Manipulação de objetos deformáveis

Na manipulação de objetos flexíveis deve-se levar em consideração as propriedades físicas dos objetos manipulados. Um exemplo de representação emprega um sistema massa-mola para descrever os objetos [31], sendo que o cálculo de seus formatos com respeito aos limites de manipulação é dado por meio da minimização da função de energia de deformação de cada objeto.

Dado um objeto deformável com características físicas conhecidas e um conjunto de maneiras que o objeto pode ser deformado, o objetivo é achar um caminho que o leva de uma configuração inicial até uma configuração final, sendo que estes caminhos consistem em rotas com deformações de baixa energia.

Em problemas de montagens, a flexibilidade pode ser levada em consideração para produzir montagens mais compactas. No contexto de projetos de medicamentos, o tratamento

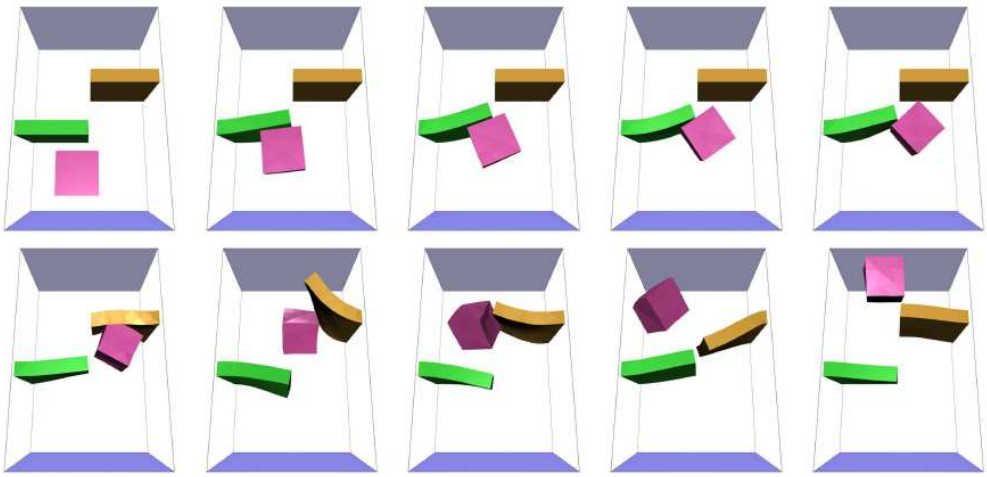


Figura 1.4: Exemplo de planejamento com objetos deformáveis. Tanto o cubo quanto as barreiras são flexíveis. A seqüência é mostrada da esquerda para a direita e de cima para baixo. [32].

rigoroso das propriedades físicas das moléculas é crucial para obter seqüências de baixa energia e que podem ser encontradas na natureza [2]. Assim, a rota resultante tem que ser livre de colisões e energeticamente praticável.

A Figura 1.4 mostra o planejamento de movimento para um cubo deformável em um ambiente cujos obstáculos também são deformáveis. O objetivo é levar o cubo da extremidade inferior do espaço de trabalho até a extremidade superior. O desafio consiste em atravessar uma passagem mais estreita que a largura do cubo situada entre as duas barreiras. Na formulação clássica de planejamento de rotas este problema não tem solução, pois o objeto móvel não pode entrar em contato com os obstáculos. Contudo, o modelamento da deformação dos objetos torna o problema solucionável.

1.4 OBJETIVOS E CONTRIBUIÇÕES DO TRABALHO

São objetivos deste trabalho estudar e avaliar técnicas de planejamento de rotas baseadas na amostragem do espaço de configurações e sua aplicação a um robô móvel omnidirecional que se move em um plano. O principal desafio é encontrar soluções que sejam praticáveis computacionalmente e sejam de boa qualidade (*e.g.*, sem ciclos e desvios necessários), visando a futura implementação em uma plataforma móvel omnidirecional pertencente ao Laboratório de Robótica e Automação da Universidade de Brasília [33].

Apesar desta aplicação ser um caso particular do problema clássico do móvel generalizado, este trabalho visa explorar não somente técnicas que resolvam o caso específico do robô omnidirecional em um plano, mas que também possam ser utilizadas em problemas

de maior complexidade e mais generalizados, como poliedros se movendo em um espaço tridimensional com obstáculos poliédricos ou mesmo sistemas multirrobôs.

As principais contribuições deste trabalho são:

1. análise teórica de uma classe de passeios aleatórios usados em algoritmos de planejamento de rotas; e
2. desenvolvimento de um algoritmo incremental de alto desempenho.

Até o presente momento, três artigos baseados neste trabalho foram publicados em conferências nacionais [33, 34, 35].

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho está organizado em sete capítulos. O Capítulo 2 apresenta conceitos relacionados à representação do espaço de configurações, sendo que o Capítulo 3 faz principalmente uma revisão dos principais métodos probabilísticos de planejamento de rotas. O Capítulo 4 apresenta métodos de amostragem do espaço de configurações e ainda alguns métodos de suavização de rotas. O Capítulo 5 apresenta o algoritmo desenvolvido nesta dissertação, assim como uma análise teórica do algoritmo no qual ele foi baseado. No Capítulo 6 são apresentados resultados obtidos por meio de simulação comparando entre si os principais algoritmos que compõem o estado da arte, assim como com o algoritmo proposto. O Capítulo 7 apresenta as conclusões e as propostas de continuidade para este trabalho. Por último, o Apêndice A mostra os ambientes de teste utilizados na avaliação experimental, enquanto o Apêndice B faz uma revisão dos principais conceitos de probabilidade usados nesta dissertação.

2 ESPAÇO DE CONFIGURAÇÕES

Geometry is not true, it is advantageous.
Henri Poincaré

O espaço de configurações é uma representação matemática que permite tratar o robô como um ponto em um espaço apropriado. Esta representação permite que problemas de planejamento que parecem diferentes em termos de geometria e cinemática sejam resolvidos pelo mesmo algoritmo [19]. Conceitos físicos como força e atrito também podem ser representados neste espaço como construções geométricas adicionais [36]. Além disso, para planejamento de rotas, faz-se necessário mapear os obstáculos neste espaço, mas não necessariamente de forma explícita.

Definição 2.0.1. *A configuração de um robô poliédrico arbitrário é a especificação da posição de todos os pontos desse robô relativa a um sistema de coordenadas fixo de referência. Sendo assim, a configuração q de um robô poliédrico $\mathcal{A}$ é a especificação da posição e orientação de um sistema de coordenadas $\mathcal{F}_{\mathcal{A}}$ fixado no robô em relação a um sistema de coordenadas fixo de referência $\mathcal{F}_{\mathcal{W}}$ [36].*

$\mathcal{A}(q)$ representa o volume ocupado no espaço de trabalho $\mathcal{W}$ pelo robô $\mathcal{A}$ quando ele se encontra na configuração q e o espaço de trabalho representa o ambiente onde o robô e os obstáculos se encontram. Ainda que a Definição 2.0.1 se aplique a robôs poliédricos, ela pode ser estendida para robôs com características geométricas mais gerais ou mesmo sistemas multirrobôs.

Definição 2.0.2. *O espaço de configurações $\mathcal{C}$ de um robô é o conjunto de todas as configurações possíveis para este robô.*

Tipicamente, cada grau de liberdade do robô corresponde a uma dimensão do espaço de configurações, sendo que este vai caracterizar todas as possíveis posições e orientações do robô independentemente dos obstáculos que se encontram no espaço de trabalho no qual o robô está inserido.

Todo obstáculo $\mathcal{O}_i$, $i = 1, \ldots, n$, no espaço de trabalho $\mathcal{W}$ pode ser mapeado em $\mathcal{C}$ por meio da região $\mathcal{CO}_i = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O}_i \neq \emptyset\}$. Sendo assim, o espaço de configurações livre $\mathcal{C}_{free}$ é dado por [36]

$$\mathcal{C}_{free} = \mathcal{C} \setminus \bigcup_{i=1}^{n} \mathcal{CO}_i = \left\{ q \in \mathcal{C} \mid \mathcal{A}(q) \cap \left(\bigcup_{i=1}^{n} \mathcal{O}_i \right) = \emptyset \right\}. \tag{2.1}$$

$\mathcal{C}_{free}$ representa as configurações em $\mathcal{C}$ na qual o robô não está em contato com obstáculos no espaço de trabalho. De fato, o planejamento de rotas em geral acontece em $\mathcal{C}_{free}$,

uma vez que se deseja encontrar uma rota livre de contato com obstáculos que ligue duas configurações quaisquer.

Exemplo 2.0.1. *Um robô pode estar contido em espaços de trabalho de duas ou três dimensões, mas o seu espaço de configurações pode ser de ordem mais alta e não Euclidiano. Por exemplo, um robô holonômico que possui movimentos de translação e rotação em um plano possui um espaço de configurações tridimensional, pois o robô possui três graus de liberdade (*x, y, θ*).* □

Se o robô é um objeto contido em um espaço de trabalho tridimensional e que pode se translacionar e rotacionar livremente, então ele possuirá um espaço de configurações ainda mais complexo, pois haverá três graus de liberdade para translação mais três graus de liberdade para rotação. Além disso, esta última pode ser representada de diversas maneiras, seja por meio de ângulos de Euler, matrizes de rotação ou quatérnios [19, 36, 2].

Faz-se necessário compreender melhor a estrutura de diferentes tipos de espaço de configurações uma vez que, quanto melhor a compreensão dessas estruturas, maior será o entendimento do problema de planejamento de rotas em geral. Para entender a estrutura de diversos tipos de espaços de configuração é necessário reconhecê-los como espaços topológicos. Topologia é o ramo da Matemática que considera as propriedades dos objetos que não mudam quando eles estão sujeitos a transformações contínuas arbitrárias [2].

Definição 2.0.3. *Um conjunto* $\mathbb{S}$ *é chamado de espaço topológico se existe uma coleção de subconjuntos de* $\mathbb{S}$*, chamados* ***conjuntos abertos****, para os quais os seguintes axiomas são válidos [19]:*

1. *A união de um conjunto contável de conjuntos abertos é um conjunto aberto;*
2. *A intersecção de um número contável de conjuntos abertos é um conjunto aberto;*
3. $\mathbb{S}$ *e* $\emptyset$ *são conjuntos abertos.* □

A Definição 2.0.3 é bastante geral e vai muito além dos espaços Euclidianos. Contudo, os espaços de configuração tipicamente tratados no problema de planejamento de movimento possuem mais restrições [2]. De fato, nos espaços topológicos comumentes utilizados em planejamento de rotas cada ponto tem uma vizinhança que se assemelha a um espaço euclidiano, mas a estrutura global de $\mathcal{C}$ pode ser bem mais complexa. Estes espaços são chamados de variedade. Entretanto, antes de uma definição mais formal de variedade faz-se necessária a definição de homeomorfismo [2]:

Definição 2.0.4. *Se o mapeamento* $\phi : \mathbb{S} \longrightarrow \mathbb{T}$ *é bijetor e* ϕ *e* ϕ^{-1} *são contínuas, então* ϕ *é um homeomorfismo e é dito que* $\mathbb{S}$ *e* $\mathbb{T}$ *são homeomorfas.*

Uma vez conhecida a Definição 2.0.4, segue a definição de variedade [2]:

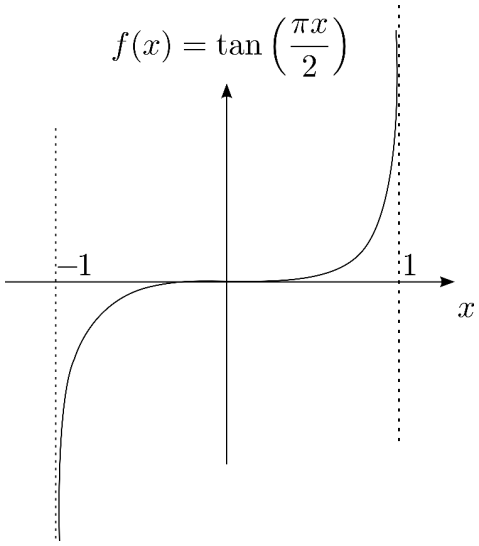


Figura 2.1: Mapeamento do intervalo aberto $(-1, 1)$ em $\mathbb{R}$.

Definição 2.0.5. *Um conjunto $\mathbb{S}$ é uma variedade k-dimensional se ele é localmente homeomorfo a $\mathbb{R}^k$, significando que cada ponto em $\mathbb{S}$ possui uma vizinhança que é homeomorfa a um conjunto aberto em $\mathbb{R}^k$.*

A Definição 2.0.5 implica no fato de que localmente um espaço topológico pode ser mapeado em um espaço Euclidiano.

Exemplo 2.0.2. *O intervalo aberto $(-1, 1)$ é uma variedade, pois localmente ele é homeomorfo a $\mathbb{R}$. Um exemplo de mapeamento que indica este fato é $f(x) = tan\left(\frac{\pi x}{2}\right),\ x \in (-1, 1)$. Este mapeamento é mostrado na Figura 2.1, em que o domínio é o intervalo aberto $(-1, 1)$, mas a imagem corresponde a todo o intervalo $\mathbb{R}$. De fato, todos intervalos abertos de $\mathbb{R}$ são variedades unidimensionais e são mapeados diretamente em $\mathbb{R}$ [19].* □

Definição 2.0.6. *Um mapeamento $\phi : U \to V$ é dito suave se todas as derivadas parciais de ϕ, de todas as ordens, são definidas. Já um mapeamento suave $\phi : U \to V$ é um difeomorfismo se ϕ é bijetora e ϕ^{-1} é suave. Quando ϕ existe, é dito que U e V são difeomorfas [2].*

Definição 2.0.7. *Um par (U, ϕ), tal que U é um conjunto aberto em uma variedade k-dimensional e ϕ é um difeomorfismo de U em algum conjunto aberto em $\mathbb{R}^k$, é chamado* ***carta*** *[2].*

Cartas são comumente associadas a sistemas de coordenadas, pois cada ponto de U tem um ponto correspondente em um espaço Euclidiano. O difeomorfismo inverso $\phi^{-1} : \mathbb{R}^k \to U$ é conhecido com parametrização da variedade [2, 19, 36].

Uma variedade k-dimensional sempre pode ser embutida em um espaço Euclidiano $\mathbb{R}^{2k+1}$. Para variedades suaves, pode ser mostrado que $\mathbb{R}^{2k}$ é suficiente [19].

Exemplo 2.0.3. *A variedade unidimensional $\mathbb{S}^1$ que representa o grupo do círculo [1] é embutida em $\mathbb{R}^2$ por meio da parametrização $\mathbb{S}^1 = \{x = (x_1, x_2) \in \mathbb{R}^2 | x_1^2 + x_2^2 = 1\}$.*

Contudo, se $\mathbb{S}^1$ é uma variedade unidimensional, então é possível mapeá-la localmente em um espaço euclidiano unidimensional $\mathbb{R}$. De fato, para o semicírculo superior, a seguinte carta é válida

$$U_1 = \left\{x \in \mathbb{S}^1 | x_2 > 0\right\}, \phi_1(x) = x_1. \tag{2.2}$$

com parametrização dada por $\phi_1^{-1}(z) = \left(z, (1 - z^2)^{\frac{1}{2}}\right)$. □

No Exemplo 2.0.3, apenas o semicírculo superior foi mapeado em $\mathbb{R}$. É possível fazer uma carta global que mapeie todo o círculo? Não, pois $\mathbb{S}^1$ não é globalmente difeomorfo a $\mathbb{R}$ [2]. Além disso, não é possível utilizar várias cartas arbitrárias para representar a topologia de $\mathbb{S}^1$. De fato, na região de fronteira e intersecção das cartas deve ser possível fazer uma transição suave entre uma e outra. Portanto, as duas cartas devem ter uma relação C^∞, cuja definição é [2]:

Definição 2.0.8. *Seja (U, ϕ) e (V, ψ) duas cartas em uma variedade k-dimensional. Seja X a imagem de $U \cap V$ dada a função ϕ e Y a imagem de $U \cap V$ dada a função ψ, ou seja,*

$$X = \left\{\phi(x) \in \mathbb{R}^k | x \in U \cap V\right\},$$
$$Y = \left\{\psi(y) \in \mathbb{R}^k | y \in U \cap V\right\}.$$

Sendo uma função denominada C^∞ quando todas as suas derivadas são definidas, é dito que as duas cartas possuem uma relação C^∞ se $\psi(\phi^{-1}(x)) : X \to Y$ e $\phi(\psi^{-1}(y)) : Y \to X$ são C^∞. □

Quando todas as cartas de uma determinada variedade são C^∞-relacionadas e a cobrem completamente, elas formam um atlas [36, 2]. Assim, em um atlas a transição entre as cartas, além de contínua, é suave.

Exemplo 2.0.4. *Ainda para $\mathbb{S}^1$, as cartas*

$$U_1 = \left\{x = (x_1, x_2) \in \mathbb{S}^1 | x_1^2 + x_2^2 = 1, x_2 > 0\right\}, \phi_1(x) = x_1 \tag{2.3}$$
$$U_2 = \left\{x = (x_1, x_2) \in \mathbb{S}^1 | x_1^2 + x_2^2 = 1, x_2 < 0\right\}, \phi_2(x) = x_1 \tag{2.4}$$
$$\tag{2.5}$$

[1] O nome círculo é utilizado no sentido topológico. Toda variedade homeomorfa a $\mathbb{S}^1$ é considerada como $\mathbb{S}^1$, apenas representada de uma outra maneira. Então, no sentido topológico, todo círculo, independentemente do raio e da posição, é representado por $\mathbb{S}^1$ [19].

com as parametrizações

$$\phi_1^{-1}(z) = \left(z, \left(1 - z^2\right)^{\frac{1}{2}}\right) \tag{2.6}$$

$$\phi_2^{-1}(z) = \left(z, -\left(1 - z^2\right)^{\frac{1}{2}}\right) \tag{2.7}$$

formam um atlas, pois $\phi_1\left(\phi_2^{-1}(z)\right) = \phi_2\left(\phi_1^{-1}(z)\right) = z$*, e portanto têm todas as derivadas definidas.* □

O exemplo anterior mostra que, apesar de ser possível mapear uma variedade k-dimensional em um espaço Euclidiano também de dimensão k, muitas vezes são necessárias várias cartas para representar globalmente a variedade. Uma vez que é possível embutir uma variedade suave k-dimensional em $\mathbb{R}^{2k}$, pode ser interessante usar representações globais mesmo que sejam necessárias mais dimensões para representar a variedade em questão. As operações matemáticas podem ser mais diretas e elegantes, sem a necessidade de ter que avaliar qual carta do atlas representa a região de interesse na variedade.

2.1 GRUPOS

Os espaços topológicos podem ser divididos em grupos, de forma que espaços de um mesmo grupo sejam equivalentes [19]:

Definição 2.1.1. *Um grupo é um conjunto* $\mathcal{G}$*, em conjunto com uma operação binária* $\circ$*, tal que os seguintes "axiomas de grupo são satisfeitos":*

1. *para todo* $a, b \in \mathcal{G}$*,* $a \circ b \in \mathcal{G}$*;*
2. *para todo* $a, b, c \in \mathcal{G}$*,* $(a \circ b) \circ c = a \circ (b \circ c)$*;*
3. *existe um elemento* $i \in \mathcal{G}$*, chamado de identidade, tal que para todo* $a \in \mathcal{G}$*,* $i \circ a = a$ *e* $a \circ i = a$*;*
4. *para todo elemento* $a \in \mathcal{G}$*, existe um elemento* $a^{-1} \in \mathcal{G}$*, chamado de inversa de* a*, para o qual* $a \circ a^{-1} = i$ *e* $a^{-1} \circ a = i$*.* □

Exemplo 2.1.1. *Os números inteiros* $\mathbb{Z}$ *formam um grupo com respeito à operação de* adição*. Sendo a identidade o número 0, a inversa de cada elemento* $i \in \mathbb{Z}$ *é o elemento* $-i \in \mathbb{Z}$*.* □

Dois grupos são particularmente úteis para planejamento de rotas: Grupo Especial Ortogonal $SO(n)$ e Grupo Especial Euclidiano $SE(n)$.

2.1.1 Grupo especial ortogonal $SO\,(n)$

O grupo especial ortogonal $SO\,(n)$ é um conjunto de matrizes que tipicamente são utilizadas para representar rotações e que possuem as seguintes características:

- são não-singulares;
- para cada $\mathbf{A} \in SO\,(n)$ existe um $\mathbf{A}^{-1} \in SO(n)$ tal que $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$;
- $\mathbf{A}^{-1} = \mathbf{A}^T$, ou seja, $\mathbf{A}$ é ortogonal;
- por último, as matrizes pertencentes a $SO\,(n)$ possuem colunas ortogonais e o determinante sempre é igual a 1.

Uma vez que n refere-se à dimensão dos elementos do grupo, em robótica tipicamente $n = \{2, 3\}$, uma vez que essas são as dimensões de $\mathcal{W}$. Portanto, para representar uma rotação no plano utiliza-se as matrizes do grupo $SO\,(2)$, sendo elas definidas como

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{2.8}$$

Já em 3 dimensões a matriz de rotação pode ser definida como [36]

$$\begin{pmatrix} n_x^2(1-c_\theta)+c_\theta & n_x n_y(1-c_\theta)-n_z s_\theta & n_x n_z(1-c_\theta)+n_y s_\theta \\ n_x n_y(1-c_\theta)+n_z s_\theta & n_y^2(1-c_\theta)+c_\theta & n_y n_z(1-c_\theta)-n_x s_\theta \\ n_x n_z(1-c_\theta)-n_y s_\theta & n_y n_z(1-c_\theta)+n_x s_\theta & n_z^2(1-c_\theta)+c_\theta \end{pmatrix}, \tag{2.9}$$

em que $\sin\theta = s_\theta$ e $\cos\theta = c_\theta$. Esta matriz representa a orientação depois de uma rotação θ em torno de um eixo com vetor unitário $\mathbf{n} = [n_x, n_y, n_z]^T$ representando sua orientação em relação ao sistema de coordenadas fixo $\mathcal{F}_\mathcal{W}$. A Figura 2.2 mostra a representação do vetor unitário no sistema de coordenadas fixo, formado pelos eixos x, y e z, e a rotação θ em torno deste eixo.

A grande vantagem de usar as matrizes do grupo $SO(n)$ é que, dada uma orientação inicial $\mathbf{R}_0^1$ do sistema de coordenadas $\mathcal{F}_\mathcal{A}$ do robô em relação ao sistema de coordenadas fixo $\mathcal{F}_\mathcal{W}$, rotações sucessivas $\mathbf{R}_{i-1}^i,\ i = 2, \ldots, k$, implicam em uma orientação final dada por $\mathbf{R}_0^k = \mathbf{R}_0^1 \cdot \mathbf{R}_1^2 \ldots \mathbf{R}_{k-1}^k$, em que $\mathbf{R}_{i-1}^i$ significa que a rotação i é feita em relação ao sistema de coordenadas móvel com orientação dada pela matriz $\mathbf{R}_0^{i-1}$.

É importante observar que a ordem da multiplicação altera a orientação final de $\mathcal{F}_\mathcal{A}$. Assim, dada uma matriz de rotação $\mathbf{R}_0^1$ que relaciona $\mathcal{F}_\mathcal{A}$ com $\mathcal{F}_\mathcal{W}$, se o objetivo é fazer uma rotação $\mathbf{R}$ não em relação ao eixo móvel, mas sim em relação ao eixo fixo, então a multiplicação é feita pela esquerda, ou seja, $\mathbf{R}_0^2 = \mathbf{R} \cdot \mathbf{R}_0^1$ [37].

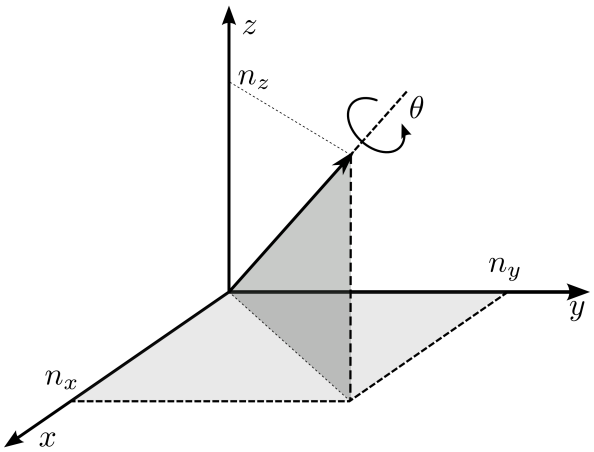


Figura 2.2: Rotação θ em torno de um eixo unitário arbitrário.

2.1.2 Grupo especial Euclidiano $SE(n)$

Dados o sistema de coordenadas fixo $\mathcal{F}_\mathcal{W}$ e o sistema de coordenadas $\mathcal{F}_\mathcal{A}$ associado a um robô poliédrico, a configuração final do robô que sofre simultaneamente translação e rotação pode ser dada por uma rotação, seguida de uma translação dada pelo vetor $\mathbf{d} \in \mathbb{R}^n,\ n = \{2, 3\}$, de $\mathcal{F}_\mathcal{A}$ em relação a $\mathcal{F}_\mathcal{W}$. Contudo, as matrizes pertencentes ao grupo $SE(n)$ já encapsulam as operações de rotação e translação em uma única matriz, dada por

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{2.10}$$

em que $\mathbf{R} \in SO(n)$, $\mathbf{d} \in \mathbb{R}^n$ é um vetor coluna de dimensão $n \times 1$ e $\mathbf{0}$ é um vetor linha de dimensão $1 \times n$.

A matriz $\mathbf{H}$ é chamada matriz de transformação homogênea e é utilizada para representar o movimento de um corpo rígido. De maneira análoga às matrizes de rotação do grupo $SO(n)$, as matrizes de transformação homogênea, quando multiplicadas sucessivamente a um sistema de coordenadas inicial móvel $\mathcal{F}_\mathcal{A}$, com posição e orientação em relação ao sistema de coordenadas fixo dadas por $\mathbf{H}_0^1$, implicam em uma orientação e posição final dadas por $\mathbf{H}_0^k = \mathbf{H}_0^1 \cdot \mathbf{H}_1^2 \ldots \cdot \mathbf{H}_{k-1}^k$. Assim como no caso das matrizes do grupo $SO(n)$, a ordem da multiplicação é importante. Sendo a transformação homogênea que relaciona $\mathcal{F}_\mathcal{A}$ com $\mathcal{F}_\mathcal{W}$ dada por $\mathbf{H}_0^1$, então se uma segunda transformação, dada por $\mathbf{H}$, for feita em relação a $\mathcal{F}_\mathcal{W}$, a orientação e posição final são dadas por $\mathbf{H}_0^2 = \mathbf{H} \cdot \mathbf{H}_0^1$ [37].

2.2 REPRESENTAÇÕES ALTERNATIVAS PARA ROTAÇÕES

Existem representações alternativas para rotações, além de $SO(n)$, que podem ser adotadas. O uso de cada uma é dado pela conveniência e ausência de restrições que inviabilizem

sua utilização. Por exemplo, uma rotação no plano pode ser representada por uma matriz do grupo $SO(2)$. Este grupo é homeomorfo a $\mathbb{S}^1$, que por sua vez está embutido em $\mathbb{R}^2$. Uma representação de menor cardinalidade [2] pode ser dada por um mapeamento de $\mathbb{S}^1 \to \mathbb{R}$ por meio de um processo de *identificação* [19].

Definição 2.2.1. *Um espaço topológico pode ser redefinido por meio de um processo de identificação, no qual alguns pontos previamente distintos passam a ser considerados iguais. Utiliza-se a notação $X/\sim$ para indicar que o espaço topológico X foi identificado e a indentificação é dada por $a \sim b$, indicando que os pontos a e b são equivalentes.* □

Exemplo 2.2.1. *Para a rotação em duas dimensões, a identificação $\mathbb{S}^1 = [0,1]/\sim$, com $0 \sim 1$ permite que um ângulo seja mapeado no intervalo $\theta \in [0,1]$, sendo o ponto 0 equivalente ao ponto 1. A parametrização $\theta \to (\cos(2\pi\theta), \sin(2\pi\theta))$ faz o mapeamento inverso $[0,1] \to \mathbb{S}^1$, que nada mais é que a parametrização por coordenadas polares normalizada no intervalo $[0,1]$.*

Exemplo 2.2.2. *Em robótica móvel, usualmente $\theta = 0 \neq \theta = 2\pi$. Embora no mapa seja a mesma postura, não é a mesma coisa para fins de controle e movimentação. Suponha que seja definida uma função de distância entre dois ângulos*

$$d = \| \theta_{end} - \theta_{start} \| . \tag{2.11}$$

Se $\theta_{end} = 179°$ e $\theta_{start} = -179°$, então $d = 179° - (-179°) = 358°$. Porém, se o espaço for identificado de tal forma que $\theta \in [-180°, 180°]/\sim$, $-180° \sim 180°$, então a Equação 2.11 é redefinida como

$$d = \begin{cases} \| \theta_{end} - \theta_{start} \| & \text{se } (\theta_{end} - \theta_{start}) \in [-180°, 180°], \\ \| \theta_{end} - \theta_{start} + 360° \| & \text{se } (\theta_{end} - \theta_{start}) < 180°, \\ \| \theta_{end} - \theta_{start} - 360° \| & \text{se } (\theta_{end} - \theta_{start}) > 180°. \end{cases} \tag{2.12}$$

Logo, se $\theta_{end} = 179°$ e $\theta_{start} = -179°$, então $d = 2°$. □

Pode-se definir o espaço de configurações para robôs com diferentes modelos geométricos[3] por meio do produto cartesiano de diferentes espaços de configurações, havendo ainda várias representações para o espaço de configurações de um mesmo robô.

Exemplo 2.2.3. *Um robô poliédrico que se translaciona em um plano mas não possui movimentos de rotação e cuja dinâmica é desconsiderada tem seu espaço de configurações pertencente a $\mathbb{R}^2$.*

[2] Cardinalidade representa o número de elementos de um conjunto.

[3] Muitos livros de robótica utilizam o termo "modelo cinemático" para designar o modelo geométrico de um robô (e.g., [37]).

Caso exista movimento de rotação, o espaço de configurações com representação de menor cardinalidade passa a ser $\mathbb{R}^2 \times \mathbb{S}$. *Como visto no Exemplo 2.2.1, a parametrização*

$$\theta \rightarrow (\cos(2\pi\theta), \sin(2\pi\theta)),$$

com $\theta = [0,1]/\sim$ *e* $0 \sim 1$ *pode ser utilizada para a representação de* θ.

Pode-se ainda representar o espaço de configurações como $\mathbb{R}^2 \times SO(2)$ *ou* $SE(2)$. *Sendo* $\mathbb{R}^2 \times \mathbb{S}$ *uma variedade tridimensional, pode-se representar o robô apenas por* $\xi_p = (x, y, \theta)$, *em que* ξ_p *é conhecida como a postura do robô. Contudo, o devido cuidado deve ser utilizado nos pontos de descontinuidade de* θ, *como já foi mostrado no Exemplo 2.2.2.* □

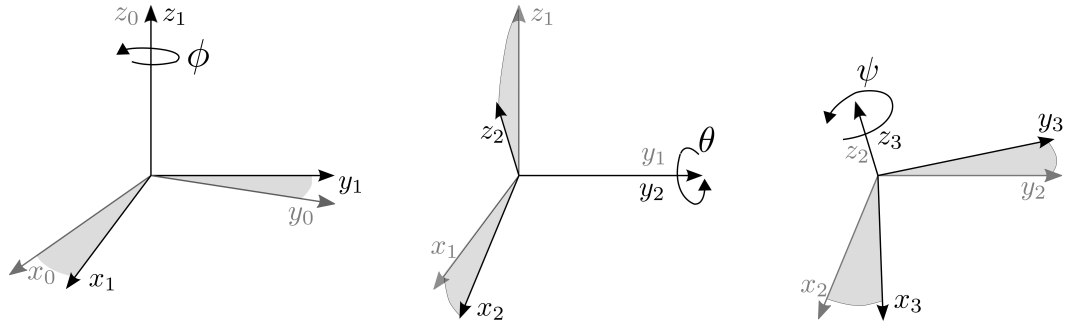
Exemplo 2.2.4. *Pode-se representar vários robôs, realizando simultaneamente movimento de translação no plano, com apenas um espaço de configurações. Se for utilizada uma aborgagem centralizada, isto é, um modelo global para vários robôs que somente se translacionam no plano, o espaço de configurações global é dado por* $\mathbb{R}^2 \times \ldots \times \mathbb{R}^2 = \mathbb{R}^{2n}$, *em que* n *é o número de robôs. Desta forma, a definição de espaço de configurações passa a ser mais ampla que a Definição 2.0.2, uma vez que vários sistemas de coordenadas* $\mathcal{F}_{\mathcal{A}_i}$ $(i = 1, \ldots, n)$ *associados um a cada robô são necessários para representar o espaço de configurações.* □

2.2.1 Ângulos de Euler

Para rotações em três dimensões, uma parametrização muito utilizada é por meio dos ângulos de Euler, em que a orientação é definida por meio de uma tripla (ϕ, θ, ψ). Esses três ângulos representam rotações sucessivas em diferentes eixos de $\mathcal{F}_{\mathcal{A}}$ partindo de uma orientação inicial de referência. A Figura 2.3 mostra essas três rotações partindo do sistema de coordenadas inicial (x_0, y_0, z_0). Na Figura 2.3(a), uma rotação ϕ em torno de z_0 é feita, resultando no sistema de coordenadas (x_1, y_1, z_1). Na Figura 2.3(b), uma rotação θ é feita em torno de y_1, resultando no sistema de coordenadas (x_2, y_2, z_2). Por último, uma rotação ψ em torno de z_2 é feita, implicando na orientação final dada por (x_3, y_3, z_3). A seqüência de eixos de rotação pode ser diferente, apenas implicando em cartas diferentes [36].

A matriz de rotação correspondente aos ângulos de Euler é dada por

$$\begin{aligned} E_{\phi\theta\psi} &= \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{pmatrix}, \end{aligned} \tag{2.13}$$



(a) Rotação de um ângulo ϕ em relação a z_0.

(b) Rotação de um ângulo θ em relação a y_1.

(c) Rotação de um ângulo ψ em relação a z_2

Figura 2.3: Orientação utilizando ângulos de Euler.

em que $\sin\alpha = s_\alpha$, $\cos\alpha = s_\alpha$ e $E_{\phi\theta\psi} \in SO(3)$. Fazendo

$$E_{\phi\theta\psi} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

com $\phi, \psi \in [0, 2\pi)$ e $\theta \in (0, \pi)$, tem-se que [36]

$$\begin{aligned} \phi &= atan2(r_{23}, r_{13}), \\ \theta &= atan2(\sqrt{r_{31}^2 + r_{32}^2}, r_{33}), \\ \psi &= atan2(r_{32}, -r_{31}). \end{aligned} \tag{2.14}$$

Um problema da parametrização por meio dos ângulos de Euler é que dois pontos diferentes $(\phi_1, \theta_1, \psi_1)$ e $(\phi_2, \theta_2, \psi_2)$ podem levar a uma mesma orientação (e.g. $(\frac{\pi}{4}, 0, \frac{\pi}{4})$ e $(\frac{\pi}{2}, 0, 0)$).

Exemplo 2.2.5. *Para um robô que se move em um espaço de trabalho tridimensional, se não houver movimento de rotação, o espaço de configurações é dado por $\mathbb{R}^3$. Caso haja rotação, ele é definido como $\mathbb{R}^3 \times SO(3)$ ou $SE(3)$. Com um atlas adequado pode-se representar $SE(3)$ por meio de um espaço Euclidiano $\mathbb{R}^6$ (e.g. $x, y, z, \phi, \theta, \psi$), em que ϕ, θ e ψ são os ângulos de Euler. Neste caso, comumente a configuração do robô é chamada* postura. *Assim como em $\mathbb{S}^1$, não é possível representar a rotação tridimensional por meio de ângulos de Euler usando apenas uma carta global [2].* □

2.2.2 Quatérnios

Os quatérnios são uma extensão do conceito de números complexos, uma vez que são constituídos de uma componente real e três componentes imaginárias [19]. Assim, dado o quatérnio $h \in \mathbb{H}$, tem-se que

$$h = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}, \tag{2.15}$$

em que $a, b, c, d \in \mathbb{R}$. Além disso, tem-se que

$$-\mathbf{ji} = \mathbf{ij} = \mathbf{k}, \tag{2.16}$$

$$-\mathbf{kj} = \mathbf{jk} = \mathbf{i}, \tag{2.17}$$

$$-\mathbf{ik} = \mathbf{ki} = \mathbf{j}, \tag{2.18}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1. \tag{2.19}$$

Os componentes do quatérnio formam uma base ortogonal em $\mathbb{R}^4$. Assim, um quatérnio pode ser representado pelo vetor $\mathbf{h} = [a, b, c, d]^T$, sendo o seu módulo definido como

$$\parallel h \parallel = \sqrt{a^2 + b^2 + c^2 + d^2} = \sqrt{\mathbf{h}^T\mathbf{h}}. \tag{2.20}$$

Os quatérnios formam um grupo quanto à multiplicação. Dados os quatérnios h_1 e h_2, a operação $h_3 = h_1 \cdot h_2$ resulta em um quatérnio dado por

$$\begin{aligned} h_3 =& h_1 \cdot h_2 \\ =& (a_0 + b_0\mathbf{i} + c_0\mathbf{j} + d_0\mathbf{k})(a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}) \\ =& (a_0a_1 - b_0b_1 - c_0c_1 - d_0d_1) + \\ & (a_0b_1 + b_0a_1 + c_0d_1 - d_0c_1)\mathbf{i} + \\ & (a_0c_1 - b_0d_1 + c_0a_1 + d_0b_1)\mathbf{j} + \\ & (a_0d_1 + b_0c_1 - c_0b_1 + d_0a_1)\mathbf{k}. \end{aligned} \tag{2.21}$$

Pela Eq. (2.19) nota-se que $ij = -ji,\ -kj = jk$ e $-ik = ki$, o que indica que a ordem da multiplicação é importante. Logo $h_1 \cdot h_2 \neq h_2 \cdot h_1$.

Os quatérnios podem ser reescritos, em notação compacta, como

$$h = (p_0, \mathbf{p}), \tag{2.22}$$

em que $p_0 = a_0$ e $\mathbf{p} = [b_0, c_0, d_0]^T$. Assim, a multiplicação dos quatérnios $h_1 = (p_0, \mathbf{p})$ e $h_2 = (r_0, \mathbf{r})$, com $r_0 = a_1$ e $\mathbf{r} = [b_1, c_1, d_1]^T$, pode ser reescrita como

$$\begin{aligned} h_3 = h_1 \cdot h_2 &= (s_0, \mathbf{s}), \\ s_0 &= p_0r_0 - \mathbf{p}^T\mathbf{r} \\ \mathbf{s} &= p_0\mathbf{r} + r_0\mathbf{p} + \mathbf{p} \times \mathbf{r}, \end{aligned} \tag{2.23}$$

e

$$\mathbf{p} \times \mathbf{r} = \det \begin{pmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ b_0 & c_0 & d_0 \\ b_1 & c_1 & d_1 \end{pmatrix} \tag{2.24}$$

é o produto vetorial entre $\mathbf{p}$ e $\mathbf{r}$.

Os quatérnios de módulo unitário podem ser usados para representar rotações θ em torno de um vetor unitário, assim como na representação eixo-ângulo dada pela Eq. (2.9). Assim, o quatérnio

$$h_r = (\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}), \tag{2.25}$$

representa a rotação de um ângulo θ em torno do eixo unitário $\mathbf{n}$ [19]. Sendo $\mathbf{n} = [n_x, n_y, n_z]^T$ um vetor unitário, h_r também é unitário, pois

$$\begin{aligned} \parallel h_r \parallel &= \cos^2\frac{\theta}{2} + n_x^2 \sin^2\frac{\theta}{2} + n_y^2 \sin^2\frac{\theta}{2} + n_z^2 \sin^2\frac{\theta}{2} \\ &= \cos^2\frac{\theta}{2} + \cancelto{1}{(n_x^2 + n_y^2 + n_z^2)} \sin^2\frac{\theta}{2} \\ &= 1. \end{aligned} \tag{2.26}$$

Sendo o quatérnio $h = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, a matriz de rotação correspondente pode ser dada por

$$\mathbf{R}(h) = \begin{pmatrix} 2(a^2+b^2)-1 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & 2(a^2+c^2)-1 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & 2(a^2+d^2)-1 \end{pmatrix}. \tag{2.27}$$

O contrário também é possível, ou seja, dada a matriz de rotação $\mathbf{R}$, as componentes do quatérnio correspondente podem ser dadas por

$$a = \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1}, \text{ e se } a \neq 0, \text{ então} \tag{2.28}$$

$$b = \frac{r_{32} - r_{23}}{4a}, \tag{2.29}$$

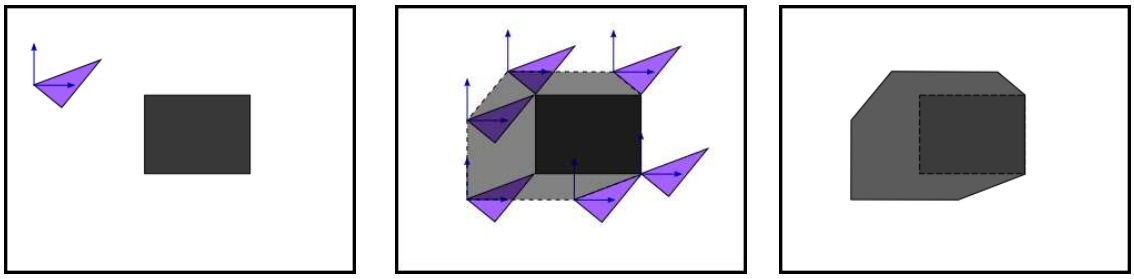
$$c = \frac{r_{13} - r_{31}}{4a}, \tag{2.30}$$

$$d = \frac{r_{21} - r_{12}}{4a}. \tag{2.31}$$

Dado um quatérnio h_0 que representa uma orientação inicial de um sistema de coordenadas, após $n-1$ rotações sucessivas a orientação final é dada por $h_n = h_0 \cdot h_1 \ldots h_{n-1}$. Assim como nas matrizes de rotações pertencentes a $SO(n)$ e nas matrizes homogêneas, a ordem da multiplicação é importante.

Exemplo 2.2.6. *O espaço de configurações de um robô poliédrico que se move em um espaço de trabalho tridimensional pode ser dado por* $\mathbb{R}^3 \times \mathbb{H}$. □

Exemplo 2.2.7. *Um sistema com m robôs, no qual cada robô é representado pelo seu espaço de configurações $\mathcal{C}_i$, pode ser representado de forma centralizada por meio do produto cartesiano entre esses vários espaços, ou seja,* $\mathcal{C}_{multi} = \mathcal{C}_1 \times \mathcal{C}_2 \times \ldots \times \mathcal{C}_m$.



(a) Espaço de trabalho contendo um obstáculo retangular e um robô triangular.

(b) Mapeamento do obstáculo no espaço de configurações do robô.

(c) Resultado do mapeamento: $\mathcal{C}_{obs}$ é dado pela região escura.

Figura 2.4: Mapeamento dos obstáculos no espaço de configurações.

Assim, supondo $\mathcal{C}_1 = \mathbb{R}^3 \times SO(3)$, $\mathcal{C}_2 = \mathbb{R}^3 \times \mathbb{H}$ *e* $\mathcal{C}_3 = SE(3)$ *os espaços de configuração correspondentes a três robôs que se movem livremente em um espaço tridimensional, o espaço de configurações resultante do sistema multirrobôs centralizado é dado por* $\mathcal{C}_{multi} = \mathbb{R}^6 \times \mathbb{H} \times SO(3) \times SE(3)$.

Observa-se, no entanto, que o espaço resultante tem uma grande dimensão, o que pode tornar o problema de planejamento de rotas computacionalmente impraticável, dependendo do algoritmo utilizado. □

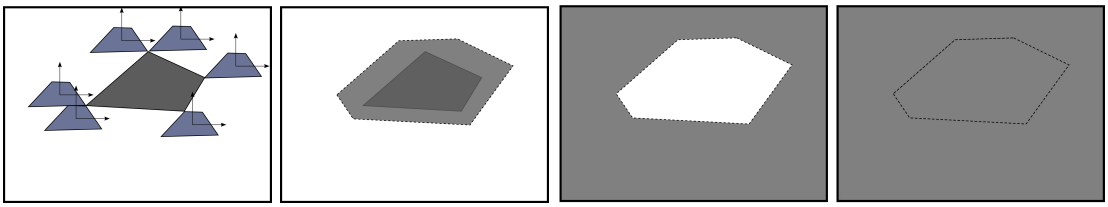
2.3 REPRESENTAÇÃO DOS OBJETOS NO ESPAÇO DE TRABALHO

A representação dos objetos no espaço de trabalho é crucial para muitos planejadores de rotas que necessitam de uma representação explícita do espaço de configurações livre, uma vez que esta representação explícita requer um mapeamento dos obstáculos $\mathcal{O}_i$ em $\mathcal{C}$ [19].

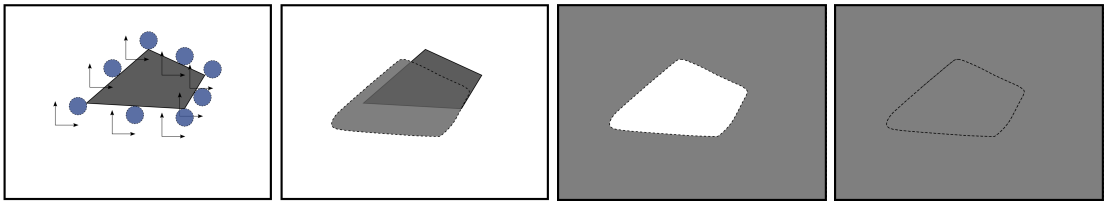
A Figura 2.4(a) mostra um robô triangular, que apenas se translaciona, e um obstáculo de formato retangular. A Figura 2.4(b) mostra o mapeamento do obstáculo no espaço de configurações do robô. Pela figura, é possível notar que este mapeamento depende do ponto de referência do robô no qual o seu sistema de coordenadas está anexado. Por último, a Figura 2.4(c) mostra o resultado do mapeamento, sendo que a região escura corresponde aos pontos do espaço de configurações que contém obstáculos, chamado $\mathcal{C}_{obs}$.

É imprescindível não confundir os conceitos de espaço de configurações $\mathcal{C}$ e espaço de configurações livre $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$. Enquanto este depende do formato do robô e dos obstáculos contidos no espaço de trabalho, aquele independe tanto de um quanto de outro. Este fato pode ser melhor ilustrado pela Figura 2.5, em que o espaço de configurações livre e o espaço de configurações com obstáculos são diferentes para dois robôs com formatos diferentes, mas $\mathcal{C} = \mathcal{C}_{obs} \cup \mathcal{C}_{free}$ é *igual* para os dois robôs.

Ainda que os planejadores abordados neste trabalho não façam uso de uma representação



(a) Mapeamento de um robô trapezoidal. (b) $\mathcal{C}_{obs}$ dentro da região tracejada. (c) $\mathcal{C}_{free}$ hachurado. (d) $\mathcal{C}_{obs} \cup \mathcal{C}_{free} = \mathcal{C}$.



(e) Mapeamento de um robô circular. (f) $\mathcal{C}_{obs}$ dentro da região tracejada. (g) $\mathcal{C}_{free}$ hachurado. (h) $\mathcal{C}_{obs} \cup \mathcal{C}_{free} = \mathcal{C}$.

Figura 2.5: Exemplo mostrando que o espaço de configurações independe do formato do robô.

explícita do espaço de configurações livre, e portanto sejam bem menos dependentes da representação dos objetos no espaço de trabalho, as principais representações serão revisadas visando manter uma continuidade deste trabalho.

2.3.1 Representação por polígonos convexos

A representação de objetos no plano pode ser dada por meio de polígonos convexos em $\mathbb{R}^2$ [19]. Um polígono pode ser representado pela seqüência $p_1 = (x_1, y_1),\ p_2 = (x_2, y_2), \ldots,\ p_m = (x_m, y_m)$ dos vértices em $\mathbb{R}^2$ no sentido anti-horário. A representação sólida deste polígono pode ser interpretada como a intersecção dos semiplanos formados por todos os pontos que situam-se em um dos lados de cada uma das retas que passam sobre os pontos $(p_1, p_2),\ (p_2, p_3), \ldots, (p_{m-1}, p_m), (p_m, p_1)$.

Definição 2.3.1. *Um subconjunto $X \subset \mathbb{R}^n$ é dito convexo se, e somente se, para qualquer $x_1, x_2 \in X$ e $\lambda \in [0, 1]$, $\lambda x_1 + (1 - \lambda)x_2 \in X$.* □

Dada a equação da reta

$$ax + by + c = 0 \tag{2.32}$$

e dois pontos (x_1, y_1) e (x_2, y_2), uma possível solução, dentre infinitas, para os parâmetros (a, b, c) em função dos dois pontos, pode ser dada por meio da resolução do seguinte sistema de equações lineares homogêneas

$$\mathbf{P} \cdot \mathbf{v} = \mathbf{0}_{3\times 1}, \tag{2.33}$$

em que $\mathbf{v} = [a,\ b,\ c]^T$ e

$$\mathbf{P} = \begin{pmatrix} x & y & 1 \\ x_2 & y_2 & 1 \\ x_1 & y_1 & 1 \end{pmatrix}. \tag{2.34}$$

Como o sistema admite a solução trivial, caso $\det \mathbf{P} \neq 0$, ou infinitas soluções caso $\mathbf{P}$ seja singular, a equação da reta é dada por $\det \mathbf{P} = 0$. Então,

$$\det \mathbf{P} = xy_2 + x_1 y + x_2 y_1 - x_1 y_2 - y_1 x - x_2 y \tag{2.35}$$

$$= (y_2 - y_1)x + (x_1 - x_2)y + x_2 y_1 - x_1 y_2 = 0. \tag{2.36}$$

Comparando a Eq. (2.36) com a (2.32), tem-se que

$$a = y_2 - y_1, \tag{2.37}$$

$$b = x_1 - x_2, \tag{2.38}$$

$$c = x_2 y_1 - x_1 y_2. \tag{2.39}$$

Além disso, é definida uma função $f(x, y) = ax + by + c$ tal que, sem perda de generalidade, os pontos que estão à esquerda da borda ligando (x_1, y_1) a (x_2, y_2) são dados por $f(x, y) = \det \mathbf{P} < 0$ [19], uma vez que a seqüência dos vértices do polígono convexo é dada sempre no sentido anti-horário.

Se o obstáculo $\mathcal{O}$ usa esta representação e possui m vértices, os semiplanos $H_i,\ (1 \leq i \leq m)$ são dados por $H_i = \{(x, y) \in \mathcal{W} \mid f_i(x, y) \leq 0\}$ e então o obstáculo é dado por

$$\mathcal{O} = H_1 \cap H_2 \ldots \cap H_m. \tag{2.40}$$

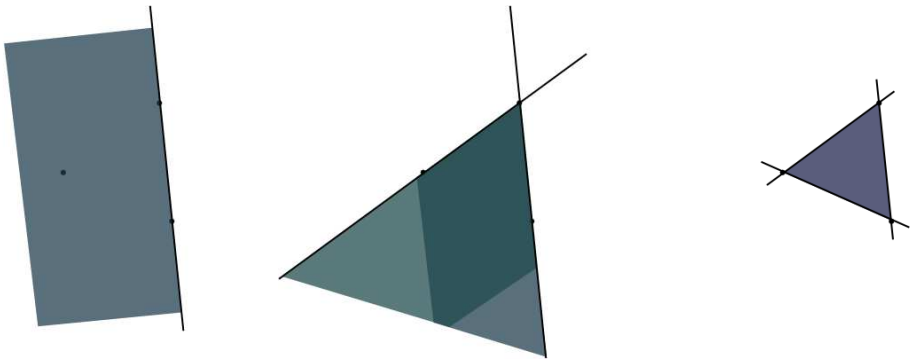
A grande vantagem desta representação é que o custo da detecção de colisão entre um ponto e um polígono qualquer é linear em relação ao número de vértices deste polígono.

A Figura 2.6 mostra como um triângulo pode ser construído pela intersecção de três semiplanos. Na Figura 2.6(a), o primeiro semiplano consiste em todos os pontos que estão do lado esquerdo da reta que passa entre os pontos inferior direito e superior da Figura. Já na Figura 2.6(b), a reta que define o segundo semiplano é dada pelos pontos superior e inferior esquerdo da Figura. Já a Figura 2.6(c) mostra a intersecção dos três semiplanos, o que resulta em um polígono na forma de um triângulo.

Polígonos não-convexos podem ser obtidos pela união de n polígonos convexos de tal forma que $\mathcal{O} = \mathcal{O}_1 \cup \ldots \cup \mathcal{O}_n$.

Seja uma função

$$e(x, y) = \begin{cases} 1 & , f(x, y) \leq 0 \\ 0 & , \text{caso contrário,} \end{cases} \tag{2.41}$$



(a) Construção do primeiro semiplano.

(b) Intersecção entre dois semiplanos.

(c) Triângulo resultante da intersecção entre três semiplanos.

Figura 2.6: Triângulo construído pela intersecção de três semiplanos.

então a função que define a fronteira e a região interior do polígono convexo de m lados é dada por

$$\alpha_m(x,y) = e_1(x,y) \cdot e_2(x,y) \ldots \cdot e_m(x,y) = \begin{cases} 0 & \text{, para } (x,y) \text{ fora do polígono,} \\ 1 & \text{, caso contrário.} \end{cases} \tag{2.42}$$

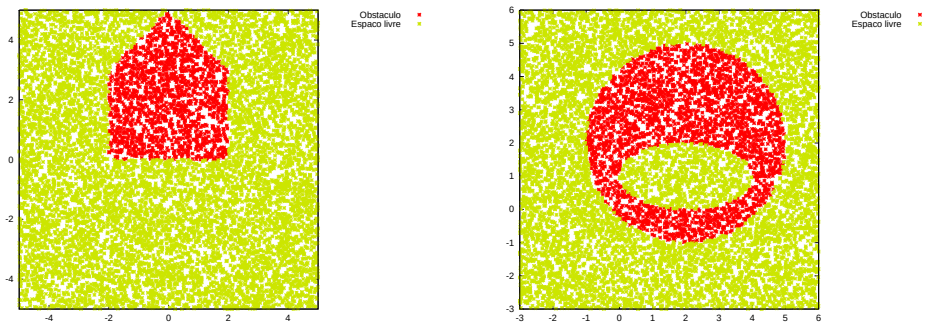
Desta forma, dada uma coleção de k obstáculos, é dito que o ponto (x,y) está em colisão se $\phi(x,y) = \alpha_1 + \ldots + \alpha_k$ for maior que zero.

A Figura 2.7(a) mostra uma simulação para detecção de colisão de um polígono dados pelos pontos $(2,0)$, $(2,3)$, $(0,5)$, $(-2,3)$ e $(-2,0)$ e seu interior dado pela Equação 2.40. Vários pontos foram gerados em todo o espaço de trabalho, sendo que aqueles no interior do polígono foram representados em cor mais escura, enquanto que os de fora foram representados em cor mais clara.

2.3.2 Modelos semi-algébricos

Uma representação que também pode ser utilizada para modelar os objetos do espaço de trabalho é aquela por modelos semi-algébricos. Um conjunto de pontos determinado por uma única primitiva polinomial é dito um conjunto algébrico. Já um conjunto que pode ser modelado por um número finito de uniões e intersecções de conjuntos algébricos é um conjunto semi-algébrico [19].

Desta forma, a representação sólida em $\mathbb{R}^2$ usando primitivas algébricas pode ser dada por $H = \{(x,y) \in \mathcal{W} \mid f(x,y) \leq 0\}$, sendo $f(x,y)$ um polinômio com coeficientes reais e variáveis x, y. A vantagem dessa representação é que os objetos modelados podem ser não convexos e inclusive conterem buracos em seu interior.



(a) Representação por polígonos convexos. (b) Representação por primitivas semi-algébricas.

Figura 2.7: Detecção de colisão usando representação por polígonos convexos e por primitivas semi-algébricas.

Por exemplo, dadas as equações da circunferência e da elipse

$$(x - x_1)^2 + (y - y_1)^2 = r^2, \tag{2.43}$$

$$\frac{(x - x_2)^2}{a^2} + \frac{(y - y_2)^2}{b^2} = 1 \tag{2.44}$$

o objeto da Figura 2.7(b) pode ser dado por

$$f_1(x, y) = (x - x_1)^2 + (y - y_1)^2 - r^2 \leq 0, \tag{2.45}$$

$$f_2(x, y) = -\left(\frac{(x - x_2)^2}{a^2} + \frac{(y - y_2)^2}{b^2} - 1\right) \leq 0, \tag{2.46}$$

para $(x_1, y_1) = (2, 2)$, $(x_2, y_2) = (2, 1)$, $a = 2$, $b = 1$ e $r = 3$. É importante notar que $f_2(x, y)$ corresponde à Eq. (2.44) com sinal negativo, uma vez que o interior da elipse corresponde a um espaço livre de obstáculos.

Outra vantagem de representar os obstáculos por primitivas semi-algébricas é que o custo para avaliar se um ponto e um obstáculo em específico estão em colisão é linear em relação ao número de polinômios usados para representar o obstáculo.

2.3.3 Grades de ocupação

Em robótica móvel, um tipo de representação freqüente para o espaço de trabalho é a grade de ocupação. Neste tipo de representação, o espaço de trabalho é dividido em células, sendo que um número $P(c_{x,y}) \in [0, 1]$, indicando a probabilidade de ocupação, é associado a cada célula $c_{x,y}$. A vantagem do uso das grades de ocupação é que as incertezas relacionadas ao mapeamento do espaço de trabalho são explicitadas na própria representação.

Para fins de planejamento de rotas, é conveniente uma representação em que cada célula tenha um valor binário 0 ou 1. Assim, 0 e 1 indicam a ausência e presença de obstáculos, respectivamente. Esta representação é conhecida como *bitmap* [19].

(a) Grade de ocupação.

(b) *Bitmap*.

Figura 2.8: Grade de ocupação transformada em *bitmap*.

A transformação da grade de ocupação em *bitmap* é feita por um processo de comparação das probabilidades $P(c_{x,y})$ com um valor limiar, em que a probabilidade de ocupação resultante de cada célula é dada por

$$P_m(c_{x,y}) = \begin{cases} 0, \ P(c_{x,y}) \leq P_{limiar} \\ 1, \ \text{caso contrário.} \end{cases} \tag{2.47}$$

A Figura 2.8(a) mostra uma grade de ocupação, representada em uma imagem em tons de cinza, com resolução de oito *bits*. Cada *pixel* corresponde a uma célula e, para um *pixel* cujo valor de tom de cinza é igual a 0 (cor preta), a probabilidade de ocupação correspondente é 1. Para um *pixel* com valor 255 (cor branca), a probabilidade de ocupação é 0. Qualquer valor intermediário dos *pixels* corresponde a valores intermediários de probabilidade de ocupação.

A Figura 2.8(b) mostra o *bitmap* derivado da Figura 2.8(a). O valor de P_{limiar} foi escolhido de tal maneira que fosse correspondente à probabilidade associada a um *pixel* com valor 254. Este é um critério bem conservador, uma vez que a menor probabilidade de ocupação da célula já é o suficiente para considerá-la ocupada. Valores pequenos para P_{limiar} geram um *bitmap* "otimista", em que apenas as células com forte probabilidade de estarem ocupadas são consideradas como tal. Porém, um valor muito baixo para P_{limiar} pode fazer com que o robô não tenha conhecimento de alguns obstáculos no espaço de trabalho.

A detecção de colisão usando *bitmaps* costuma ser feita *pixel* a *pixel*, ou seja, para o robô na configuração q, cada pixel a_i de $\mathcal{A}(q)$ deve ser avaliado se está em colisão ou não. Assim, o custo para a detecção de colisão de um robô poliédrico é linear em relação ao número de *pixels* utilizados para representá-lo.

3 MÉTODOS DE PLANEJAMENTO DE ROTAS

Assim como
Todas as portas são diferentes
Aparentemente
Todos os caminhos são diferentes
Mas vão dar todos no mesmo lugar
Raul Seixas

Os métodos atuais de planejamento de rotas comumente são divididos em três categorias principais [36]: algoritmos que utilizam mapa de rotas; algoritmos que fazem decomposição em células; e algoritmos que utilizam funções de potencial. Os métodos que utilizam mapa de rotas visam construir um mapa topológico que representa a conectividade de $\mathcal{C}_{free}$. Geralmente utiliza-se um grafo para armazenar as informações deste mapa topológico, sendo que os nós armazenam informações sobre o estado do robô e as bordas indicam a possibilidade de transição entre os estados. Os métodos que fazem decomposição em células decompõem $\mathcal{C}_{free}$ em uma coleção de regiões, chamadas células, que não se sobrepõem. Então, um grafo que armazena as informações de adjacência entre as células é utilizado para resolver o problema de planejamento de rotas. Por último, os algoritmos que utilizam funções de potencial representam o robô por uma partícula no espaço de configurações sujeita a forças que a guiam para o destino. Assim, os obstáculos provocam uma força de repulsão, enquanto a configuração de destino induz uma força de atração na partícula.

Os algoritmos de planejamento de rotas também podem ser divididos em outras duas categorias: de questionamento único e de múltiplos questionamentos. Aqueles que se encontram na primeira categoria, em geral, não fazem pré-processamento do espaço de configurações, sendo que a cada requisição o planejador faz a representação necessária de $\mathcal{C}_{free}$ para resolver o problema em questão. Já os métodos de múltiplos questionamentos fazem o pré-processamento do espaço de configurações visando representá-lo de maneira global. Assim, um tempo maior é destinado à construção da representação global de $\mathcal{C}_{free}$, mas os questionamentos feitos posteriormente podem ser solucionados rapidamente. Esta última categoria é especialmente útil no caso de espaços de trabalho estáticos.

Por último, uma classificação útil divide os métodos de planejamento de rotas em determinísticos e em probabilísticos. Dadas as configurações inicial e final, os planejadores determinísticos sempre retornam a mesma rota, enquanto que os planejadores probabilísticos podem retornar rotas diferentes. A vantagem óbvia dos métodos determinísticos é que, além do comportamento ser previsível, a depuração dos resultados é muito mais fácil que nos métodos probabilísticos. No entanto, um "oponente mal-intencionado" [1] pode criar situ-

[1] O termo "oponente mal-intencionado" é utilizado metaforicamente para indicar um agente que cria, pro-

ações que induzem o planejador a falhar. Já os métodos probabilísticos, além de serem mais robustos quanto a este problema, podem ser mais eficientes por fazerem representações do espaço de configurações por amostras do mesmo, sendo que uma amostragem mais seletiva pode capturar somente as informações relevantes de $\mathcal{C}_{free}$, diminuindo o tempo de execução do algoritmo.

A seguir é apresentado um método simples baseado em campos de potencial, sendo que nas seções seguintes uma ênfase maior é dada nos métodos probabilísticos baseados na amostragem do espaço de configurações.

3.1 CAMPOS DE POTENCIAL

O planejador de rotas baseado em campos de potencial artificiais foi uma das primeiras abordagens utilizadas na tentativa de resolver o problema de planejamento de rotas. Apesar de não ser baseado na amostragem do espaço de configurações, historicamente ele desempenhou um papel importante para o crescimento das pesquisas na área de navegação.

Ainda hoje abordagens híbridas que utilizam o princípio de campos de potencial artificiais juntamente com outras técnicas (*e.g.* amostragem do espaço de configurações) são exploradas. De fato, existem implementações que mostraram ser praticáveis para problemas com dimensões mais altas do espaço de configurações, como o algoritmo apresentado em Barraquand & Latombe [5]. Naquele trabalho, campos de potencial artificiais são utilizados para planejamento de rotas e passeios aleatórios são utilizados para escapar de mínimos locais.

Em Khatib [38], um planejador baseado em funções de potencial é aplicado em problemas de desvio de obstáculos em tempo real para robôs manipuladores, mas que pode facilmente ser estendido a robôs móveis. O problema de planejamento, tradicionalmente tratado como uma tarefa de alto nível, é reformulado de forma a ser integrado diretamente no sistema de controle de movimento do robô no chamado espaço operacional. Este espaço é um conjunto formado pelas configurações que descrevem a posição e orientação da garra do robô em relação a um sistema de coordenadas fixo. Uma arquitetura de controle de dois níveis é implementada: o nível de avaliação dos parâmetros, que atualiza os coeficientes dinâmicos da garra, a matriz jacobiana e o modelo geométrico a uma taxa de amostragem mais baixa; e um nível de controle dos servos do manipulador, com uma taxa de amostragem mais alta, que calcula o sinal de controle usando um estimador e os coeficientes dinâmicos atualizados.

Nesta Seção é apresentada a formulação mais simples para o planejamento de rotas baseado em funções de potencial. Assim, dadas as configurações inicial $\mathbf{q}_{start}$ e final $\mathbf{q}_{goal}$ da

positadamente ou não, situações nas quais o planejador pode falhar.

rota, define-se uma função de potencial

$$U(\mathbf{q}) = U_{att}(\mathbf{q}) + U_{rep}(\mathbf{q}), \tag{3.1}$$

em que $\mathbf{q} = [x_1, x_2, \ldots, x_n]^T$ é um vetor coluna que representa a configuração do robô, $U_{att}(\mathbf{q})$ representa o potencial de atração gerado pela configuração final e $U_{rep}(\mathbf{q})$ representa o potencial de repulsão gerado pelos obstáculos.

Nesta abordagem o movimento do robô é obtido por descida do gradiente de $U(\mathbf{q})$ e o seu movimento termina quando o gradiente se anula, ou seja, o robô atinge um ponto $\mathbf{q}^*$, em que $\nabla U(\mathbf{q}^*) = 0$. Tal ponto $\mathbf{q}^*$ é chamado ponto crítico de U.

Uma vantagem do planejador baseado em campos de potencial é o grande apelo intuitivo de que a configuração final do robô gera um potencial de atração, enquanto os obstáculos geram um potencial de repulsão. Assim, as rotas geradas possuem uma boa qualidade e geralmente passam a uma distância segura dos obstáculos. Porém, as funções de potencial devem ser determinadas criteriosamente de forma a evitar problemas de mínimos locais que não correspondem ao ponto de destino. Assim, muitas funções de potenciais levam a planejadores que não são completos [2] e muitas vezes a escolha da função de potencial é muito dependente da topologia do espaço de configurações [36].

$U_{att}(\mathbf{q})$ deve ser monotonicamente crescente com a distância de $\mathbf{q}$ a $\mathbf{q}_{goal}$. A escolha mais simples é o potencial cônico

$$U_{att}(\mathbf{q}) = \zeta \parallel \mathbf{q} - \mathbf{q}_{goal} \parallel, \tag{3.2}$$

em que ζ é um parâmetro de escala para o efeito do potencial atrativo e $\parallel \mathbf{q} - \mathbf{q}_{goal} \parallel$ a distância euclidiana entre as configurações $\mathbf{q}$ e $\mathbf{q}_{goal}$ [2].

O gradiente de atração é

$$\begin{aligned} \frac{\partial U_{att}(\mathbf{q})}{\partial \mathbf{q}} = \nabla U_{att}(\mathbf{q}) &= \left(\frac{\partial U_{att}(\mathbf{q})}{\partial x_1}, \frac{\partial U_{att}(\mathbf{q})}{\partial x_2}, \ldots, \frac{\partial U_{att}(\mathbf{q})}{\partial x_n} \right) \\ &= \zeta \, \frac{\mathbf{q} - \mathbf{q}_{goal}}{\parallel \mathbf{q} - \mathbf{q}_{goal} \parallel}. \end{aligned} \tag{3.3}$$

Sendo assim, o vetor gradiente aponta para fora do ponto de destino com magnitude ζ para todos os pontos do espaço de configurações, exceto para $\mathbf{q}_{goal}$, onde o gradiente é indefinido.

Devido a esta descontinuidade, é mais desejável utilizar uma função continuamente diferenciável. A função mais simples a fazer o potencial aumentar na medida em que aumenta a distância do destino e seja continuamente diferenciável é aquela que cresce quadraticamente com a distância a $\mathbf{q}_{goal}$:

$$U_{att}(\mathbf{q}) = \frac{1}{2} \zeta \parallel \mathbf{q} - \mathbf{q}_{goal} \parallel^2 \tag{3.4}$$

com o gradiente

$$\begin{aligned}\nabla U_{att}(\mathbf{q}) &= \nabla\left(\frac{1}{2}\zeta \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel^2\right) \\ &= \frac{1}{2}\zeta \nabla \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel^2 \\ &= \zeta(\mathbf{q}-\mathbf{q}_{goal})\end{aligned} \tag{3.5}$$

A vantagem desta função é que quanto mais longe o robô está do destino, mais rápido ele se aproxima do mesmo. Assim, na medida em que ele se aproxima da configuração final, sua velocidade diminui, o que ajuda a reduzir o sobrepasso que pode ser causado pela discretização do planejador.

Como as velocidades podem aumentar bastante quando $\parallel \mathbf{q}-\mathbf{q}_{goal} \parallel$ é muito grande, combinam-se as funções cônicas e quadráticas. Tal potencial pode ser definido como [2]

$$U_{att}(\mathbf{q}) = \begin{cases} \frac{1}{2}\zeta \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel^2 & , \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel \leq d^*_{goal}, \\ d^*_{goal}\zeta \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel -\frac{1}{2}\zeta(d^*_{goal})^2 & , \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel > d^*_{goal}. \end{cases} \tag{3.6}$$

Sendo assim, o gradiente de atração é dado por

$$\nabla U_{att}(\mathbf{q}) = \begin{cases} \zeta(\mathbf{q}-\mathbf{q}_{goal}) & , \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel \leq d^*_{goal}, \\ d^*_{goal}\zeta\frac{\mathbf{q}-\mathbf{q}_{goal}}{\|\mathbf{q}-\mathbf{q}_{goal}\|} & , \parallel \mathbf{q}-\mathbf{q}_{goal} \parallel > d^*_{goal}. \end{cases} \tag{3.7}$$

O termo d^*_{goal} inserido na Eq. (3.6) indica a fronteira de atuação das duas funções de potencial e garante que o potencial de atração seja contínuo nessa fronteira.

Já o potencial de repulsão pode ser definido em termos da distância $D(\mathbf{q})$ do obstáculo mais próximo:

$$U_{rep}(\mathbf{q}) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(\mathbf{q})}-\frac{1}{Q^*})^2 & , D(\mathbf{q}) \leq Q^*, \\ 0 & , D(\mathbf{q}) > Q^* \end{cases} \tag{3.8}$$

$Q^* \in \mathbb{R}$ é um fator que permite ao robô ignorar obstáculos suficientemente distantes e $\eta > 0$ é um ganho do gradiente de repulsão, sendo ambos determinados de forma empírica [2].

O gradiente do potencial de repulsão é dado por

$$\nabla U_{rep}(\mathbf{q}) = \begin{cases} \eta(\frac{1}{Q^*}-\frac{1}{D(\mathbf{q})})\frac{1}{D^2(\mathbf{q})}\nabla D(\mathbf{q}) & , D(\mathbf{q}) \leq Q^*, \\ 0 & , D(\mathbf{q}) > Q^*. \end{cases} \tag{3.9}$$

O planejamento de rotas utilizando funções de potencial é obtido por descida de gradiente e pode ser resumido pelo Algoritmo 1. Neste algoritmo, $\alpha(i)$ determina o passo na i-ésima

```
1 q(0) ← q_start;
2 i ← 0;
3 while | ∇U(q(i)) | > ε do
4     q(i + 1) ← q(i) + α(i)∇U(q(i));
5     i ← i + 1;
```

Algoritmo 1: Planejador que utiliza campos de potencial artificiais.

iteração e ε pode ser arbitrariamente pequeno, sendo determinado pelos requisitos da tarefa. Além disso, para uma escolha apropriada de α, pode ser mostrado que a descida de gradiente converge para um mínimo no campo. Porém, não há garantia de o mínimo ser global, não havendo então garantia que o robô pare na configuração final [2].

A Figura 3.1 mostra um exemplo típico no qual um planejador de rotas baseado nas funções de potencial dadas pelas Eq. (3.6) e (3.8) levam o robô para um mínimo local que se encontra na cavidade de um obstáculo não-convexo. Enquanto o ponto de destino impõe um potencial de atração, os obstáculos impõem um potencial de repulsão cuja resultante anula o potencial de atração, mantendo o robô parado. Contudo, uma função de potencial que permite com que o robô chegue ao destino é representada graficamente pelas linhas que contornam o obstáculo.

Exemplos de abordagens que visam eliminar o problema de mínimos locais espúrios são apresentados em [5, 20]. Em [5], o mecanismo usado para escapar de mínimos locais por meio de passeios aleatórios se mostrou eficiente mesmo para robôs com 31 graus de liberdade. Já em [20] foi projetado um planejador baseado em campos eletrostáticos no qual existe um único mínimo global localizado na configuração final.

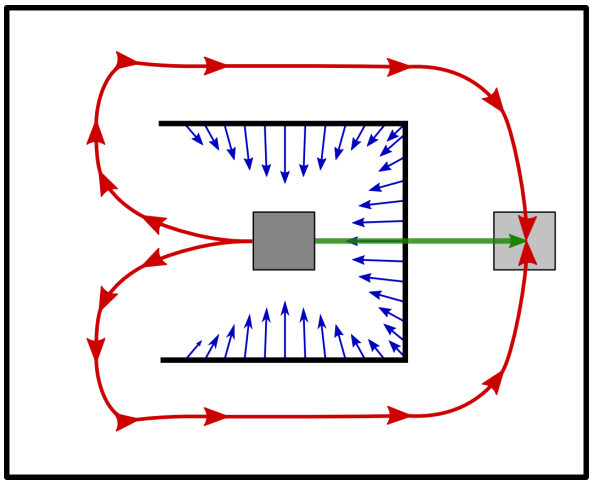


Figura 3.1: Planejador baseado em campos de potencial. A cavidade do obstáculo não-convexo constitui um mínimo local.

A grande vantagem dos planejadores baseados em funções de potencial é que as rotas geralmente possuem uma boa qualidade, passando a uma distância segura em relação aos obstáculos. Porém, existe a necessidade de se especificar uma função de potencial que contenha apenas um mínimo localizado no ponto de destino, o que pode não ser trivial para topologias mais complexas do espaço de configurações. Além disso, alguns planejadores fazem a representação explícita do espaço de configurações (*e.g.* [20]), o que se torna computacionalmente inviável para dimensões muito grandes.

Uma abordagem relativamente simples e determinística é o planejador por frente de onda, PFO (também conhecido como *Wavefront planner*, do inglês) [2, 19]. Neste algoritmo, uma grade é colocada no espaço de configurações. Então, rotulam-se as células ocupadas pelos obstáculos com 1 e o espaço livre em 0. A configuração final é rotulada com 2 e a partir dela as células vizinhas livres são numeradas com o valor de seu rótulo mais 1, ou seja, 3. O processo é repetido com todas as células vizinhas e assim por diante. Este procedimento é repetido até que a configuração inicial tenha sido alcançada. A solução é encontrada partindo da configuração inicial e seguindo a seqüência decrescente até a célula de rótulo 2 que corresponde à configuração final.

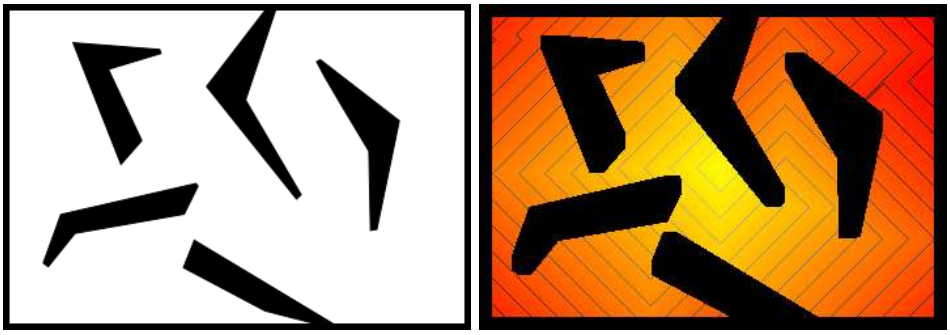
O nome deste algoritmo vem do fato de o processo de criação de rótulos se assemelhar com a propagação de uma onda no espaço de configurações. Além disso, o rótulo de uma célula pode representar um potencial proporcional à sua distância em relação ao ponto de destino. Assim, esse algoritmo pode ser encontrado na literatura sobre planejadores de rotas baseados em funções de potencial [2].

As vantagens do PFO são o fato dele ser de resolução completa [2] e só existir um único ponto de mínimo, localizado no ponto de destino. Além disso, a solução é ótima no sentido de ter a menor distância de Manhattan [5]. Ainda, este algoritmo é de fácil implementação, não necessitando de estruturas de dados complexas para armazenamento dos pontos intermediários da rota. Contudo, para espaços de configuração de ordem mais alta, o PFO pode ser tornar computacionalmente impraticável [2].

A Figura 3.2 mostra a onda de potencial que se propaga a partir do centro do ambiente. O espaço de configurações é bidimensional (não há rotação) e o robô é um quadrado de lado igual a 10 células. Cada célula é representada por um *pixel* na imagem e corresponde a um quadrado de $10cm$ de lado em $\mathcal{W}$. Assim, a parte mais clara do mapa indica o ponto de destino, enquanto que as linhas eqüipotenciais estão desenhadas em toda a figura. Ao longo de cada uma dessas linhas, encontram-se células com distâncias idênticas em relação ao ponto de destino (de acordo com a distância de Manhattan).

Este algoritmo possui outras duas desvantagens: as rotas são geradas muito próximas aos obstáculos, sendo que a segurança do robô pode ficar comprometida; além disso, o critério

[2] Um planejador é dito de resolução completa se, dado um espaço de configurações discretizado com uma resolução arbitrária, ele retorna a solução em tempo finito, caso ela exista, e retorna a ausência de solução, caso contrário.



(a) Espaço de trabalho. (b) Potencial gerado pela frente de onda em $\mathcal{C}_{free}$.

Figura 3.2: Potencial gerado pela frente de onda cujo mínimo global está localizado no centro do espaço de configurações.

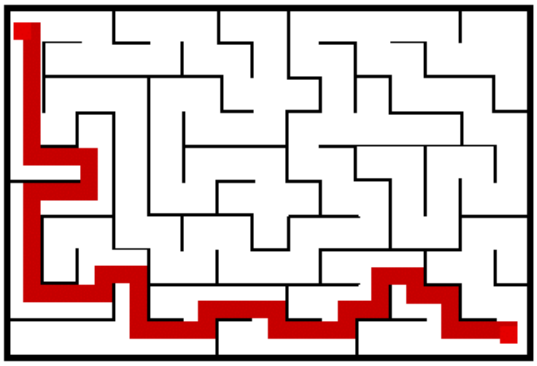


Figura 3.3: Rota gerada pelo planejador por frente de onda.

da distância de Manhattan não gera rotas ótimas segundo o critério da distância euclidiana. Assim, existem outros planejadores, que usam a distância euclidiana como métrica do espaço de configurações, cujas rotas geradas são mais curtas que aquelas geradas pelo PFO.

A Figura 3.3 mostra uma rota gerada pelo PFO para um espaço de trabalho na forma de um labirinto. Nota-se que a rota passa muito próxima dos obstáculos, o que na prática não é seguro e pode causar colisões devido a incertezas na representação do espaço de trabalho e mesmo incertezas na localização do robô. Contudo, o algoritmo apresentado na Seção 4.2.3 pode ser utilizado para afastar a rota em relação aos obstáculos.

3.2 MAPA DE ROTAS PROBABILÍSTICO

O Mapa de Rotas Probabilístico (também conhecido por PRM, sigla para *Probabilistic Roadmap*, do inglês) foi um dos primeiros algoritmos viáveis na prática, baseados na amostragem do espaço de configurações, a obter sucesso no problema de planejamento de rotas [39, 4].

Entrada: Número máximo de configurações a serem geradas em $\mathcal{C}_{free}$; número máximo de vizinhos com as quais uma configuração pode se conectar; e a distância máxima a ser avaliada para estes vizinhos.

Saída : Mapa de rotas.

```
1  i ← 0; N ← 0; E ← 0;
2  while i < nodes do
3      q_new ← configuração aleatória em C;
4      if q_new ∈ C_free then
5          i ← i + 1;
6          Adiciona q_new ao grafo por meio de um nó em N;
7          Seleciona n vizinhos q_j em torno de q_new que estejam a uma distância inferior
           a D_n;
8          for j = 1 to n do
9              if local_planner(q_new,q_j) and not mesmo componente then
10                 Adiciona uma borda em E conectando o nó referente a q_new ao nó
                   referente a q_j;
```

Algoritmo 2: Etapa de aprendizado do PRM

A estrutura básica deste algoritmo é um grafo $R(N, E)$ que se localiza no espaço livre $\mathcal{C}_{free}$ de um espaço de configurações de dimensão m, em que m é o número de graus de liberdade do robô e N e E são os nós e as bordas de R, respectivamente.

O algoritmo PRM é dividido em duas etapas: aprendizado e questionamento. A primeira consiste em expandir o grafo na região $\mathcal{C}_{free}$ para poder gerar rotas rapidamente na etapa seguinte. O objetivo é que um tempo maior seja usado no pré-processamento do espaço de configurações na etapa de aprendizado, de forma que as rotas requisitadas na etapa de questionamento sejam geradas rapidamente. Dessa forma, uma das premissas utilizadas no PRM padrão é que o espaço de trabalho seja estático.

Na etapa de aprendizado, geram-se sucessivamente várias configurações aleatórias no espaço livre $\mathcal{C}_{free}$. Cada nova configuração q_{new}[3] gerada é armazenada em um nó N do grafo R e então um planejador local, definido mais à frente, verifica os vizinhos q_j aos quais q_{new} pode se conectar. Caso não haja colisão, se q_{new} e q_j não estiverem em um mesmo componente, uma borda ligando esses dois nós é gerada no grafo R e acrescentada a E. Um componente é dado por um conjunto de nós N ligados entre si por meio de bordas E.

[3]A formulação do PRM permite abstrair a topologia do espaço de configurações, de forma que este pode ser considerado uma variedade arbitrária. Portanto, as configurações, em regra, não são escritas em negrito, pois não representam necessariamente vetores em $\mathbb{R}^n$.

O objetivo de não ligar o novo nó gerado a um novo vizinho, caso ambos estejam no mesmo componente, é evitar criar ciclos no grafo, pois além de facilitar as buscas por rotas na etapa de questionamento, há uma diminuição do número de chamadas ao planejador local, o que aumenta a eficiência do algoritmo. Contudo, a ausência de ciclos pode fazer com que o robô utilize um caminho bem maior que o necessário para alcançar o seu destino, de forma que existem trabalhos que exploram a adição de alguns ciclos úteis no mapa de rotas [40, 41].

O Algoritmo 2 mostra a etapa de construção do mapa de rotas, conhecida também como etapa de aprendizado. A Linha 2 mostra o parâmetro $nodes$ que é dependente da topologia de $\mathcal{C}_{free}$ e indica quantas amostras devem ser geradas.

Um número muito pequeno de amostras pode não garantir uma conectividade suficientemente boa do mapa de rotas, levando à incapacidade do algoritmo em achar uma solução na etapa de questionamento. Por outro lado, um número muito grande pode ser desnecessário, ocasionando em desperdício de recursos computacionais. Existem trabalhos que tratam da análise teórica do PRM e eles indicam que a probabilidade de sucesso do algoritmo em alcançar a solução (caso ela exista) cresce exponencialmente com o número de amostras geradas em $\mathcal{C}_{free}$ [2, 42], mas ainda assim depende de parâmetros que nem sempre são triviais de serem encontrados.

Na Linha 3 vários métodos podem ser utilizados para gerar as amostras, sendo que o desempenho do algoritmo pode depender fortemente do método de amostragem escolhido [10, 43, 14].

Por outro lado, a seleção dos vizinhos mais próximos que ocorre na Linha 7 nem sempre é trivial e às vezes é necessária uma estrutura eficiente de armazenamento e busca de vizinhos [44], uma vez que em grandes dimensões esta pode se tornar uma etapa computacionalmente onerosa.

O planejador local da Linha 9 mais simples a ser utilizado, caso $\mathcal{C} \in \mathbb{R}^n$, é dado pelo Algoritmo 3, porém planejadores mais complexos podem ser usados. Existem vários trabalhos mostrando que planejadores locais mais robustos podem aumentar o desempenho do algoritmo [45], havendo inclusive planejadores locais baseados em campos de potenciais artificiais [40].

Exemplo 3.2.1. *Um planejador local extremamente simples é aquele que discretiza o segmento de reta que liga uma configuração q_i a uma configuração q_j e coloca configurações intermediárias do robô para verificar se alguma delas colide com algum obstáculo. É válido lembrar que este segmento de reta localiza-se no espaço de configurações do robô, que não necessariamente é euclidiano (apenas localmente, pois assume-se que $\mathcal{C}$ é uma variedade). Assim, uma vez que a menor distância entre dois pontos é dada por um segmento de reta, a métrica para distância em $\mathcal{C}$ torna-se fundamental. Por exemplo, se $q_i, q_j \in \mathbb{S}^1$, a distância entre elas pode ser definida como [46]*

$$dist_{\mathbb{S}^1}(q_i, q_j) = \min\left(|q_i - q_j|, 1 - |q_i - q_j|\right). \tag{3.10}$$

Entrada: As configurações $\mathbf{q}_i, \mathbf{q}_j \in \mathcal{C}_{free}$ e $\alpha \in (0, 1)$

Saída : 1, se $\mathbf{q}_i$ puder ser conectado a $\mathbf{q}_j$ e 0, caso contrário.

```
1 q ← q_i; t ← 0;
2 while t < 1 do
3     if collision_detection(q) then
4         return 0;
5     else
6         t ← t + α;
7         q ← (1 − t)q_i + t q_j;
  // Se chegou até este ponto, é porque não houve colisão.
8 return 1;
```

Algoritmo 3: Planejador local simples em $\mathbb{R}^n$.

Assim, para $\mathbb{S}^1$, o planejador local tem que discretizar o segmento de reta baseado na distância retornada pela Eq. (3.10)*.* □

Exemplo 3.2.2. *Se o espaço de configurações é representado por $\mathbb{R}^n$, um planejador local de fácil implementação verifica se o par $\{\mathbf{q}_i, \mathbf{q}_j\}$ é válido se e somente se* [4]

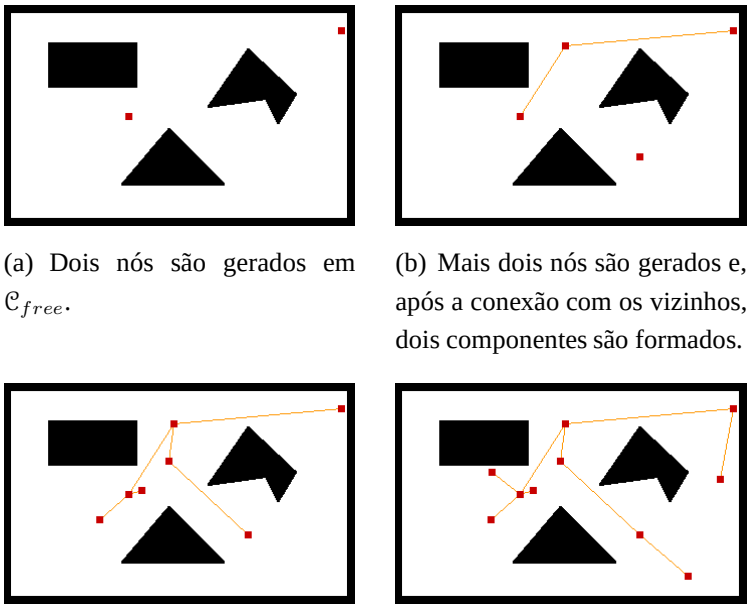
$$t\mathbf{q}_i + (1 - t)\mathbf{q}_j \in \mathcal{C}_{free} \tag{3.11}$$

para todo $t \in [0, 1]$. □

O Algoritmo 3 mostra o planejador local apresentado no Exemplo 3.2.2. O parâmetro α indica a resolução da discretização do segmento de reta entre os pontos $\mathbf{q}_i$ e $\mathbf{q}_j$. Para cada configuração intermediária, a detecção de colisão é feita. É dito que $\mathbf{q}_i$ "enxerga" $\mathbf{q}_j$ se nenhuma dessas configurações intermediárias estiver em colisão.

As Figuras 3.4(a)-(d) mostram vários passos intermediários da construção de um mapa de rotas com dez nós no espaço livre. A Figura 3.4(a) mostra dois nós gerados em $\mathcal{C}_{free}$. Como eles não podem ser ligados pelo planejador local, então eles formam dois componentes distintos. A Figura 3.4(b) mostra a geração de mais configurações. Nota-se que uma configuração intermediária foi suficiente para conectar os dois nós da figura anterior. O processo de geração de amostras continua nas Figuras 3.4(c) e 3.4(d), sendo que esta última mostra o mapa de rotas resultante após a geração de 10 configurações aleatórias. Este mapa de rotas é formado pelos nós, que indicam as configurações intermediárias, e as bordas ligando estes nós, indicando a transição entre as configurações por eles representadas.

[4] Como apontado no Exemplo 3.2.1, espaços de configuração não euclidianos podem ser utilizados, bastando apenas que métricas consistentes de distância sejam definidas.



(a) Dois nós são gerados em $\mathcal{C}_{free}$.

(b) Mais dois nós são gerados e, após a conexão com os vizinhos, dois componentes são formados.

(c) Mais nós são gerados e os dois componentes se fundem em apenas um.

(d) Mapa de rotas resultante após a geração de 10 configurações.

Figura 3.4: Evolução da construção do mapa de rotas no algoritmo PRM.

Ainda que o exemplo mostrado na Figura 3.4 tenha um forte apelo intuitivo e que tanto o espaço de trabalho quanto o espaço de configurações tenham duas dimensões e sejam euclidianos, os nós dos grafos poderiam armazenar muito mais informações além da mera localização espacial, como por exemplo arranjos moleculares.

Uma vez que o mapa de rotas é construído, as requisições de rotas podem ser feitas na fase de questionamento. Assim, dadas as configurações inicial q_{start} e final q_{end}, o algoritmo tenta ligá-las às configurações q'_{start} e q'_{end} mais próximas no grafo R, respectivamente. Uma primeira verificação é feita para garantir que q'_{start} e q'_{end} estejam no mesmo componente.

Caso estejam em componentes diferentes, significa que não há uma rota ligando a configuração de origem à de destino. Contudo, se estiverem em componentes iguais, então é feita uma busca no grafo e o algoritmo retorna a rota formada pela concatenação de q_{start} a q'_{start}, todos os nós ligando q'_{start} a q'_{end} e por último q'_{end} a q_{end}.

A grande vantagem do PRM é que o tempo de execução só é grande na etapa de construção, sendo que as rotas requisitadas na etapa de questionamento são retornadas rapidamente. Assim, o seu uso é indicado para ambientes estáticos ou cujas mudanças ocorrem lentamente. Entretanto, o PRM pode ser utilizado para resolver o problema de planejamento de rotas em espaços de configuração de dimensões muito grandes, pois uma vez que o mapa de rotas é construído, a informação relacionada à conectividade do espaço de configurações fica toda "compactada" no grafo, independente da dimensão do espaço de configurações.

Em relação à etapa de construção, alguns pontos negativos podem ser levantados. Pri-

meiramente, o número de nós a serem gerados tem que ser determinado *a priori*. Assim, um número pequeno pode não ser suficiente para capturar a conectividade do espaço de configurações livre. Já um número muito grande aumenta - e muito - o tempo de execução do algoritmo [10]. Além disso, a construção do mapa de rotas consiste em gerar nós uniformemente em $\mathcal{C}$, checar se eles estão em colisão e em seguida verificar os vizinhos mais próximos com os quais eles podem se conectar. Como a amostragem ocorre uniformemente no espaço de configurações e apenas as amostras que caem em $\mathcal{C}_{free}$ são aceitas, muitas amostras são desperdiçadas. Além disso, o método para verificar quais são os vizinhos mais próximos influencia fortemente o desempenho do algoritmo. A técnica mais simples, e também a mais ingênua, consiste em analisar todos os nós do grafo e verificar quais estão mais próximos do nó de referência. Apesar de ser de fácil implementação, o desempenho do algoritmo cai drasticamente quando o número de nós é muito grande. Técnicas mais eficientes para a busca de vizinhos mais próximos existem, mas elas são mais complicadas de se implementar, principalmente quando a topologia de $\mathcal{C}$ é não-euclidiana (*e.g.* algoritmos baseados em *kd-tree* [46]).

3.3 ÁRVORES ALEATÓRIAS PARA EXPLORAÇÃO RÁPIDA

O algoritmo Árvores Aleatórias para Exploração Rápida (também conhecido como RRT, do inglês *Rapidly-Exploring Random Trees*) foi desenvolvido inicialmente para planejamento kinodinâmico [5]. Isto é feito por meio da substituição dos conceitos relacionados ao espaço de configurações por conceitos no espaço de estados [15].

Este algoritmo visa realizar uma busca em espaços com grandes dimensões. Estes espaços possuem tanto restrições algébricas provenientes dos obstáculos, quanto restrições sob a forma de equações diferenciais provenientes da dinâmica do sistema, bem como restrições de movimento de robôs não-holonômicos.

Uma grande vantagem deste tipo de planejamento é que ele resolve não somente o problema de planejamento de rotas, como também o planejamento de trajetória, considerando assim as velocidades e acelerações do robô. Além disso, no planejamento kinodinâmico ainda é considerado o sinal de controle que leva o robô de uma configuração para outra.

A motivação para o desenvolvimento deste algoritmo é que um planejador puramente cinemático pode levar a um caminho não executável pelo robô (*e.g.* limitações de forças e torques no motor). Além disso, imprecisões no controle podem requerer um modelamento explícito da dinâmica do sistema para garantir trajetórias livres de colisões caso esta dinâmica seja muito rápida (*e.g.* helicópteros).

Sendo o estado $\mathbf{x}$ de um robô definido como $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$, o espaço de estados $\mathcal{X}$ consiste

[5] Planejamento kinodinâmico refere-se a planejamento de movimento que envolve dinâmica, ou seja, velocidades e acelerações [19].

Entrada: q_{start} e q_{end}

Saída : Caminho resultante

```
1  T_a.init(q_start);
2  T_b.init(q_end);
3  for k = 1 to K do
4      gera uma configuração aleatória q_rand em C de acordo com uma distribuição
       uniforme;
5      if RRT-expand(T_a,q_rand) not TRAPPED then
6          if RRT-connect(T_b,q_new) = REACHED then
7              return path(T_a,T_b);
8      else
9          swap(T_a,T_b);
10 return NULL;
```

Algoritmo 4: Planejador de rotas RRT.

no conjunto formado por todos os estados possíveis do robô. Assim, os obstáculos mapeados no espaço de estados correspondem aos estados $\mathbf{x}$ cujas configurações correspondentes não estão em $\mathcal{C}_{free}$. Este conjunto é dado por $\mathcal{X}_{obs}$.

Contudo, uma outra região indesejável em $\mathcal{X}$ é aquela de inevitável colisão (*e.g.*, para uma dada velocidade não existe sinal de controle que consiga evitar a colisão). Essa região onde há uma colisão inevitável juntamente com $\mathcal{X}_{obs}$ forma o conjunto mais abrangente $\mathcal{X}_{ric}$, sendo que [15] define $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{ric}$, o que leva a um planejador mais robusto.

Uma trajetória é então definida como um caminho contínuo parametrizado no tempo $\tau : [0, T] \rightarrow \mathcal{X}_{free}$ que satisfaz os limites holonômicos do robô. Sendo assim, a tarefa do planejador consiste em achar uma função de controle $u : [0, T] \rightarrow U$ representando uma entrada variante no tempo que, quando aplicada, move o sistema de $\mathbf{x}_{start}$ até $\mathbf{x}_{end}$, evitando os obstáculos.

Dados $\mathbf{x}_{start}$ e $\mathbf{x}_{end}$ os estados inicial e final do robô, o objetivo é criar duas árvores $\mathcal{T}_{start}$ e $\mathcal{T}_{end}$ cujos nós correspondem a estados aleatórios e que crescem de $\mathbf{x}_{start}$ rumo a $\mathbf{x}_{end}$ e de $\mathbf{x}_{end}$ rumo a $\mathbf{x}_{start}$, respectivamente. O objetivo, então, é tentar fusionar as árvores, o que caracteriza a solução do problema.

Apesar do RRT ter sido desenvolvido inicialmente para planejamento no espaço de estados, em [6] uma versão do RRT bidirecional específico para buscas no espaço de configurações foi apresentada. Esta versão é adequada para casos em que a dinâmica não é levada em consideração ou ainda nos casos em que ela é considerada em uma etapa posterior ao processo de planejamento de rotas.

Entrada: q_{rand} e a árvore $\mathcal{T}$

Saída : Status de q_{new}

1 $q_{near} \leftarrow$ `nearest_neighbor(`q_{rand},$\mathcal{T}$`);`

2 **if** `distance(`q_{near},q_{rand}`)` > ϵ **then**

3 Gera q_{new} a uma distância ϵ de q_{near} no sentido $q_{near}\overrightarrow{q_{rand}}$;

4 **else**

5 $q_{new} \leftarrow q_{rand}$;

6 **if** `straight_line_path_planner(`q_{near},q_{new}`)` **then**

7 $\mathcal{T}$`.add_vertex(`q_{new}`);`

8 $\mathcal{T}$`.add_edge(`q_{near},q_{new}`);`

9 **if** q_{rand} = q_{new} **then**

10 **return** REACHED;

11 **else**

12 **return** ADVANCED;

13 **else**

14 **return** TRAPPED;

Algoritmo 5: RRT-expand, utilizado para fazer a expansão das árvores aleatórias.

Uma vez que o problema passa a ser formulado no espaço de configurações, o objetivo é fazer com que duas árvores cresçam da configuração inicial e final do robô por meio de um processo de amostragem em $\mathcal{C}$ e expansão destas árvores em $\mathcal{C}_{free}$, sendo então que a abordagem do algoritmo passa a ser apenas geométrica.

O Algoritmo 4 mostra uma versão do planejador RRT. O objetivo deste algoritmo é explorar uniformemente todo o espaço de configurações livre visando encontrar a solução. Contudo, como a busca é feita de forma bidirecional, a solução em geral é encontrada bem antes do espaço ter sido todo explorado. Na Linha 5 a função RRT-expand faz a expansão de uma árvore $\mathcal{T}$ e seu funcionamento é melhor explicado no Algoritmo 5. Além disso, dado que as duas árvores puderam ser fundidas através da Linha 6, a solução consiste em concatenar todos os nós que ligam q_{start} a q_{end}. Se a solução não tiver sido encontrada, as árvores são trocadas e o processo de expansão seguida de tentativa de fusão (com alternância das árvores) é feito sucessivamente até que um número máximo de amostras tenham sido geradas.

O Algoritmo 5 mostra o mecanismo de expansão do RRT. O parâmetro ϵ em geral é determinado empiricamente e depende da estrutura de $\mathcal{C}_{free}$. Um valor muito pequeno faz com que o algoritmo demore a convergir. Já valores muito grandes podem fazer com que muitas amostras sejam descartadas devido ao grande incremento exigido, sendo que a tendência é

Entrada: q e a árvore $\mathcal{T}$

Saída : Status de q_{new}

```
1  q_near ← nearest_neighbor(q,T);
2  if greedy_path_planner(q_near,q,q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near,q_new);
5      if q = q_new then
6          return REACHED;
7      else
8          return ADVANCED;
9  else
10     return TRAPPED;
```

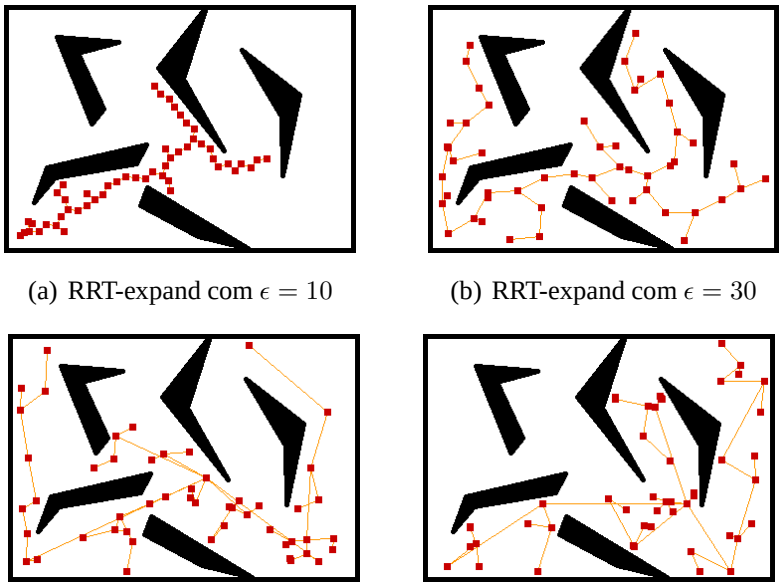
Algoritmo 6: RRT-connect, utilizado para fazer a fusão entre as duas árvores.

que somente as amostras que caírem a uma distância menor que ϵ do vizinho mais próximo serão aproveitadas.

O Algoritmo 6 mostra uma outra versão para o algoritmo de expansão do RRT, que também pode ser utilizado para realizar a fusão entre as duas árvores aleatórias. Na Linha 1 o vizinho mais próximo a q é selecionado. Em seguida, na Linha 2 um planejador local mais agressivo tenta ligar q_{near} a q_{rand}. Se ele conseguir, o algoritmo retorna REACHED. Caso contrário, se o planejador conseguir avançar mas encontrar depois de alguns incrementos algum obstáculo no meio do caminho, q_{new} recebe a configuração imediatamente anterior à colisão com o obstáculo e o algoritmo retorna ADVANCED. Se não for possível avançar (*e.g.*, o primeiro incremento do planejador já está em colisão), então o algoritmo retorna TRAPPED.

Outras versões para o Algoritmo 4 podem ser usadas. Ele pode ser implementado usando apenas o RRT-connect, o que leva a uma abordagem de exploração mais agressiva, ou então usando apenas o RRT-expand, cuja implementação é mais simples e que pode resultar em uma cobertura mais homogênea do espaço explorado.

As Figuras 3.5(a) – (c) mostram a expansão de apenas uma árvore cujo nó inicial encontra-se no canto inferior esquerdo do mapa. Para gerar estas árvores, foi utilizado o RRT-expand com diferentes valores para o parâmetro ϵ. Já a Figura 3.5(d) mostra o resultado da expansão da árvore usando o RRT-connect para a geração de 50 amostras. Para valores menores de ϵ, os nós da árvore ficam mais próximos uns dos outros e portanto a exploração é menor. Por outro lado, valores maiores de ϵ aumentam a explorabilidade, mas com o custo de obter uma árvore em que as bordas de ligação dos nós se cruzam, o que resulta em uma rota com zi-



(a) RRT-expand com $\epsilon = 10$ (b) RRT-expand com $\epsilon = 30$

(c) RRT-expand com $\epsilon = 200$ (d) RRT-connect

Figura 3.5: Exemplo de árvore criada pelo algoritmo RRT. Foram geradas 50 amostras.

guezagues. Assim, enquanto valores pequenos para ϵ resultam em rotas de maior qualidade, o tempo de execução do algoritmo pode ser muito maior, pois vários nós têm que ser gerados para alcançar a solução, enquanto que valores maiores podem ocasionar rotas com um baixo tempo de execução, mas que necessitem de um pós-processamento. Já o RRT-connect constrói árvores com uma boa explorabilidade, porém bem desorganizadas e com muitas sinuosidades.

Assim como o PRM, um problema do RRT é que a busca por vizinhos mais próximos é feita ao adicionar um novo nó na árvore de exploração. Assim, a implementação dessa busca é crucial para o bom desempenho do algoritmo, principalmente quando as árvores contém um grande número de nós. No entanto, uma grande vantagem do RRT em relação ao PRM é que, enquanto neste o número de nós tem que ser determinados *a priori*, naquele os nós são gerados de maneira incremental. Contudo, caso o RRT use o RRT-expand como mecanismo de exploração, o valor de ϵ tem que ser determinado, sendo que este parâmetro influencia o desempenho do algoritmo. Já se o RRT utilizar somente o RRT-connect, inclusive como mecanismo de exploração do espaço de configurações, o algoritmo passa a não ter parâmetro. Entretanto, a rota resultante, por conter ziguezagues, necessita de um pós-processamento mais rigoroso.

Apesar do algoritmo RRT ser de questionamento único, ou seja, a cada nova rota requisitada ele repete todo o processo de busca, as rotas geradas em requisições prévias e os nós adicionais gerados em cada busca podem ser mantidos de forma a serem aproveitados posteriormente. De fato, um algoritmo incremental que mantém as árvores criadas em buscas anteriores para utilizá-las em questionamentos subseqüentes foi desenvolvido em [47] e mostrou um bom desempenho. Além disso, nessa abordagem incremental o problema é

generalizado para espaços de trabalho em que os obstáculos se movem. As árvores são armazenadas em uma lista e podem ser fundidas ou divididas, de acordo com a atualização do espaço de trabalho. Além disso, um algoritmo para poda do grafo resultante é apresentado visando a manutenção e diminuição dos recursos computacionais utilizados.

3.4 ÁRVORES EM ESPAÇOS EXPANSIVOS

O algoritmo EST (do inglês, *Expansive-Spaces Trees*) [48, 49] também é baseado na amostragem do espaço de configurações e o conceito de expansividade é definido e utilizado na análise teórica do algoritmo. Ele é de questionamento único e, assim como o RRT, visa explorar de maneira uniforme o espaço de configurações. Duas árvores são geradas a partir das configurações iniciais q_{start} e q_{end} e o objetivo é que uma árvore seja alcançável pela outra, o que caracteriza a solução do problema.

Para expandir uma árvore $\mathcal{T}$ de maneira homogênea, um nó q_{rand} é escolhido de acordo com uma probabilidade $\pi(q)$ e a partir deste nó uma nova amostra q_{new} é gerada em sua vizinhança seguindo uma distribuição uniforme. Caso o planejador local consiga conectar q_{rand} a q_{new}, então q_{new} é adicionada à árvore e uma borda conectando as duas amostras é inserida em $\mathcal{T}$.

A escolha adequada do tamanho do intervalo de amostragem na vizinhança do nó q_{rand} pode ter um impacto significativo no desempenho do algoritmo. Se o intervalo for muito grande, áreas que não são relevantes para a solução podem ser amostradas. Por outro lado, se este for muito pequeno o algoritmo pode demorar a convergir.

A probabilidade $\pi(q)$ pode ser definida de forma a polarizar as amostras nas regiões de $\mathcal{T}$ que são menos densas. Uma maneira direta é atribuir a cada configuração q um peso $w(q)$ que corresponde ao número de configurações dentro de sua vizinhança. Se $\pi(q)$ for definida como sendo inversamente proporcional a $w(q)$, então regiões menos densas terão maior chance de serem amostradas.

O Algoritmo 7 resume o mecanismo de construção das árvores. A idéia é que o EST seja usado de maneira bidirecional, sendo que uma árvore cresce da configuração inicial q_{start} e outra cresce da configuração final q_{end}. Já o Algoritmo 8 mostra o mecanismo de fusão entre as duas árvores. Este mecanismo consiste em tentar ligar os nós de cada uma das árvores que estejam suficientemente próximos entre si. Isto ocorre porque para distâncias muito grandes a probabilidade de um nó enxergar o outro pode ser muito pequena. Desta forma, há uma redução no número de chamadas ao planejador local, o que aumenta o desempenho do algoritmo. Na linha 4 o caminho retornado é dado pela concatenação de todos os nós que ligam q_{start} a q_{end}.

Apesar do objetivo deste algoritmo ser explorar todo o espaço de configurações de ma-

Entrada: q_0 e N

Saída : Árvore $\mathcal{T}$ com nó raiz q_0

```
1 Adiciona q_0 à árvore T;
2 for i = 1 to N do
3     q_rand ← conf. aleatória escolhida em T com probabilidade π(q_rand);
4     q_new ← conf. aleatória na vizinhança de q_rand, tal que dist(q_rand,q_new) < d;
5     if local_planner(q_rand,q_new) then
6         adiciona q_new a T;
7         adiciona uma borda ligando q_rand a q_new;
8 return T;
```

Algoritmo 7: Build-EST, usado para construção das árvores.

Entrada: $\mathcal{T}_{start}$ e $\mathcal{T}_{end}$

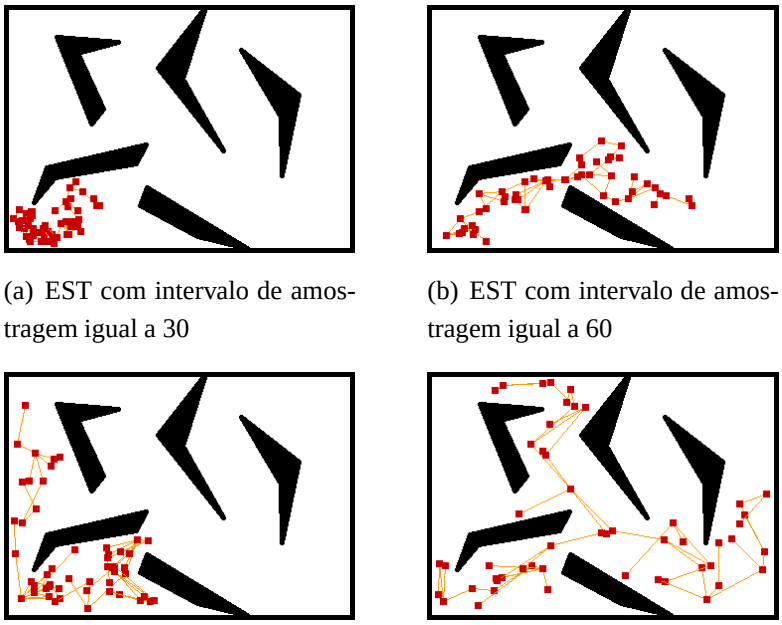
```
1 forall q_a ∈ T_start and q_b ∈ T_end do
2     if dist(q_a,q_b) < l then
3         if local_planner(q_a,q_b) then
4             return PATH;
```

Algoritmo 8: merge-EST, utilizado para fazer a fusão entre as árvores.

neira uniforme, pode ser mostrado que a solução é encontrada antes que ocorra a exploração completa de $\mathcal{C}_{free}$ [49].

Um outro fator que pode causar um impacto significativo no desempenho deste algoritmo é a maneira como é calculado $\pi(q)$ [2]. Por exemplo, um método bastante direto (e também ingênuo) para calcular $\pi(q)$ é enumerar todas as configurações de $\mathcal{T}$ e analisar quais são vizinhas a q. O custo disto é linear em relação ao número de nós na árvore e passa a ser proibitivo se esta for muito grande.

Uma alternativa, implementada neste trabalho, que se propõe para este método ingênuo de calcular $\pi(q)$ é impor uma grade com resolução arbitrária em $\mathcal{C}$, sendo que a cada nó gerado apenas as células mais próximas são consideradas para o cálculo da vizinhança (a distância máxima pode ser a mesma utilizada na geração uniforme das amostras). Para cada uma dessas células vizinhas é feita a atualização dos nós que estão nelas contidas, de forma que não é preciso percorrer toda a árvore para fazer atualização da estrutura de vizinhança. Porém, este método provavelmente possui baixo desempenho em dimensões muito altas do espaço de configurações. Além disso, a escolha da função de probabilidade $\pi(q)$ pode ser crucial para o desempenho do algoritmo.



(a) EST com intervalo de amostragem igual a 30

(b) EST com intervalo de amostragem igual a 60

(c) EST com intervalo de amostragem igual a 90

(d) EST com intervalo de amostragem igual a 120

Figura 3.6: Árvore gerada pelo algoritmo EST. Foram geradas 50 amostras.

A Figura 3.6 mostra a árvore de expansão gerada pelo EST. Cada nó é escolhido de acordo com a probabilidade $Pr\{q_i\} = \frac{1}{e^{n_i}}$, sendo n_i o número de vizinhos do nó q_i. Esta função de massa polariza fortemente as amostras para zonas inexploradas do espaço de configurações. Nota-se que quanto maior o intervalo de amostragem da distribuição uniforme, maior é a exploração do algoritmo. Porém, valores muito altos fazem com que regiões do espaço de configurações que não são relevantes para o questionamento de rota em questão sejam amostrados, aumentando o tempo de execução do algoritmo. A determinação deste parâmetro não é trivial e depende da topologia do espaço de configurações, assim como do formato de $\mathcal{C}_{free}$.

Outra dificuldade inerente a este algoritmo diz respeito à escolha da função $\pi(q)$. Se esta função de massa resultar em uma distribuição que polariza fortemente a amostragem rumo a zonas pouco exploradas, pode haver problema de mínimos locais, e então a evolução da construção da árvore pode ficar prejudicada. Por outro lado, se não existir polarização, ou seja, se a distribuição for uniforme, então a tendência é que mais configurações sejam geradas em áreas já muito exploradas. Isto porque, como a probabilidade de escolha de um nó é igual para todos os nós da árvore, então se uma região contêm apenas um nó enquanto que outra contém 100 nós, a probabilidade de um nó desta região ser escolhido é 100 vezes maior que a probabilidade de escolher o único nó da outra região.

O algoritmo EST gera árvores com qualidade inferior àquelas geradas pelo RRT, pois independente dos parâmetros utilizados, as bordas que ligam os nós das árvores se sobrepõem de tal maneira que as rotas resultantes sempre contêm ziguezagues.

3.5 PASSEIO ALEATÓRIO ADAPTATIVO

O algoritmo ARW (*Adaptive Random Walk*, do inglês) [9] encontra-se na categoria dos planejadores de rotas de questionamento único. Assim, dadas as configurações inicial e final de um robô, um caminho ligando as configurações é feito em $\mathcal{C}_{free}$ sem a etapa de pré-processamento que ocorre em alguns planejadores de múltiplos questionamentos, como o PRM. A ênfase do ARW é na eficiência de tempo de execução, sendo que há a garantia de convergência para a solução [9]. Este algoritmo utiliza um passeio aleatório para explorar o espaço de configurações livre até que a última amostra gerada possa se conectar ao ponto de destino por meio de um planejador local. Todas as amostras geradas em $\mathcal{C}_{free}$ são armazenadas, formando uma cadeia de configurações. Assim, quando a última amostra do passeio é ligada ao ponto de destino, faz-se a concatenação de todas essas configurações da cadeia, e então é dito que a solução do problema de planejamento de rotas foi encontrada. A definição do passeio aleatório adaptativo vem a seguir.

Seja uma configuração $\mathbf{q} = [q_0, q_1, \ldots, q_{n-1}]^T$, em que n é o número de dimensões do espaço de configurações. O ARW é dado por um processo estocástico de tempo discreto, caracterizado por um passeio aleatório que evolui a partir de uma configuração de origem $\mathbf{q}_{start}$, com o objetivo de alcançar a configuração de destino $\mathbf{q}_{goal}$. A evolução deste passeio aleatório adaptativo é dado pelas seguintes equações recursivas [9]

$$\begin{aligned} \mathbf{q}_0 &= \mathbf{q}_{start} \\ \mathbf{q}_k &= \mathbf{g}\left(\mathbf{q}_{k-1}, \mathbf{v}_k\right), \text{para } k = 1, 2, 3 \ldots \end{aligned} \tag{3.12}$$

A função $\mathbf{g} : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ é usada para avaliar se duas configurações $\mathbf{q}_i$ e $\mathbf{q}_j$ podem ser conectadas. Ela é definida tal que [6]

$$\mathbf{g}(\mathbf{q}_i, \mathbf{q}_j) = \begin{cases} \mathbf{q}_i + \mathbf{q}_j & \text{, se } \mathbf{q}_i \text{ pode ser ligado a } \mathbf{q}_i + \mathbf{q}_j \text{ pelo planejador local.} \\ \mathbf{q}_i & \text{, caso contrário.} \end{cases} \tag{3.13}$$

Além disso, $\mathbf{v}_k$ é gerada seguindo

$$\mathbf{v}_k \sim N(\mathbf{0}, \mathbf{\Sigma}_k), \tag{3.14}$$

em que $\mathbf{\Sigma}_k$ é a matriz de covariâncias adaptativa do processo, dada por

$$\mathbf{\Sigma}_k = \max\left(\mathbf{P}_k, \mathbf{\Sigma}_{min}\right) \tag{3.15}$$

com

$$\mathbf{P}_k = \left(\frac{1}{H} \sum_{i=k-H}^{k-1} \mathbf{q}_i \cdot \mathbf{q}_i^T\right) - \bar{\mathbf{q}}_H \cdot \bar{\mathbf{q}}_H^T, \tag{3.16}$$

[6] Ainda que a operação "+" neste caso seja realmente uma soma, pois considera-se que as configurações sejam representadas por vetores em $\mathbb{R}^n$, pode ser definida uma operação de concatenação. Assim, a configuração $\mathbf{q}$ passa a ser uma configuração q em um espaço de configurações arbitrário.

sendo $\bar{\mathbf{q}}_H$, a média das H últimas amostras, dada por

$$\bar{\mathbf{q}}_H = \frac{1}{H} \sum_{i=k-H}^{k-1} \mathbf{q}_i. \tag{3.17}$$

O termo $\mathbf{\Sigma}_{min}$ da Eq. (3.15) garante uma variância mínima para a geração de novas amostras. Valores muito altos para $\mathbf{\Sigma}_{min}$ podem ocasionar a diminuição (ou mesmo eliminar) a adaptabilidade do passeio aleatório. Já o parâmetro $H > 0$ define o número de elementos da cadeia gerada pelo processo que serão utilizados para a atualização da matriz de covariâncias. O ajuste deste parâmetro é feito de forma empírica e depende da topologia de $\mathcal{C}_{free}$, sendo que grandes mudanças em seu valor não promove mudanças significativas no desempenho do algoritmo [9]. Contudo, a existência de adaptabilidade atualiza a matriz de covariâncias e direciona o algoritmo para uma melhor amostragem.

O objetivo principal da atualização da matriz de covariâncias é fazer com que a amostragem de $\mathbf{v}_k$ seja polarizada pela história H do processo e a geração de amostras seja adequada ao formato da região de $\mathcal{C}_{free}$ onde se localiza a última configuração da cadeia. É essa atualização da matriz de covariâncias que determina a adaptabilidade do passeio aleatório.

Apesar do algoritmo ARW já ser eficiente na tarefa de exploração do espaço de configurações, sua eficiência em termos de tempo de execução pode ser melhorada por meio de uma implementação bidirecional. Nesta implementação, dois passeios aleatórios são gerados, de forma que um inicia na configuração de origem e o outro inicia na de destino. Caso os dois passeios se encontrem, a rota final será a união dos dois passeios. Caso o passeio originado no ponto $\mathbf{q}_{start}$ encontre a configuração final, ou caso o passeio originado em $\mathbf{q}_{goal}$ encontre a configuração inicial, então uma solução também terá sido alcançada.

Assim como na maioria dos planejadores baseados na amostragem do espaço de configurações, um problema do ARW é a qualidade muito baixa da rota resultante. Em geral ela possui muitos ciclos e desvios desnecessários. Uma maneira de amenizar este problema é fazer um pós-processamento com o objetivo de suavizá-la. Este pós-processamento é apresentado na Seção 4.2.

A Figura 3.7 mostra a evolução de um ARW, dado pela Eq. (3.12), sendo que a elipse 3σ é desenhada em várias regiões do ambiente. Nota-se que tanto a inclinação da elipse 3σ quanto o tamanho de seus eixos se adaptam de acordo com a região do ambiente em que a última configuração gerada se encontra. Esta adaptabilidade é importante, pois para espaços mais confinados o traço da matriz de covariâncias é menor e menos amostras são geradas em $\mathcal{C}_{obs}$. Isto aumenta a eficiência do algoritmo, uma vez que menos amostras são desperdiçadas. Por outro lado, em espaços mais abertos, o traço da matriz de covariâncias aumenta, permitindo que amostras sejam geradas a distâncias maiores, aumentando a explorabilidade do algoritmo.

O Algoritmo 9 resume o funcionamento do planejador de rotas ARW unidirecional. O planejador local utilizado nas Linhas 3 e 6 é dado pelo Algoritmo 3. Já a Linha 10 indica que

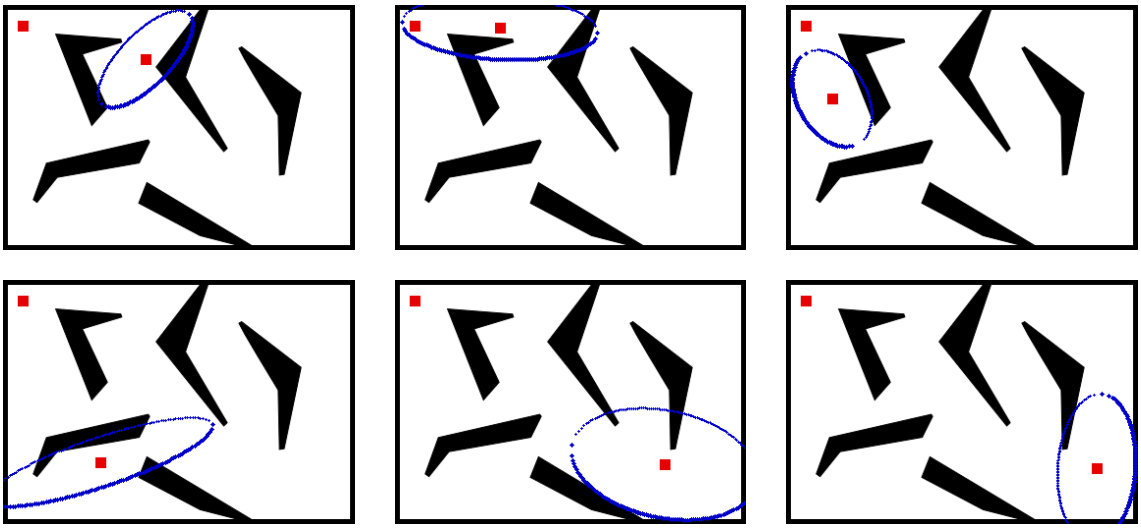


Figura 3.7: Evolução do passeio aleatório com a adaptação da elipses 3σ em diversos pontos do espaço de configurações $\mathcal{C} \in \mathbb{R}^2$.

todas as novas configurações geradas e não rejeitadas são armazenadas em uma lista. Além desta lista armazenar as configurações que compõem a solução do problema, as últimas H configurações armazenadas são utilizadas para o cálculo da matriz de covariâncias.

O algoritmo ARW oferece grandes vantagens quando comparado com o PRM, RRT e EST. Primeiramente, sua implementação é a mais simples dentre os quatro algoritmos. Além disso, por não fazer busca de configurações vizinhas mais próximas, o custo de gerar uma nova amostra também é muito baixo. Outra vantagem é que o ARW é o único a se adaptar à estrutura do espaço de configurações para gerar uma nova amostra. Esta adaptação permite que, implicitamente, o algoritmo extraia informações sobre a região onde o passeio aleatório se localiza. Por exemplo, é fácil identificar quando uma região possui um volume pequeno, pois o traço da matriz de covariâncias tende a ser pequeno nessas regiões. Além disso, esta adaptabilidade permite que menos configurações sejam desperdiçadas, pois a amostragem é polarizada em $\mathcal{C}_{free}$.

Já a maior desvantagem do ARW é que a rota resultante possui uma péssima qualidade, com muitas sinuosidades e ciclos desnecessários. Assim, as rotas geradas precisam de um pós-processamento rigoroso, visando a eliminação de redundâncias e suavização da rota resultante.

3.6 DISCUSSÃO

No contexto de robótica móvel, todos os algoritmos abordados neste Capítulo possuem bom desempenho. Isto porque, em geral o espaço de configurações tem uma dimensão pequena, ou seja, duas ou três dimensões. Assim, todos os cinco algoritmos possuem bom

Entrada: $\mathcal{W}$, configurações inicial e final.

Saída : Lista que contém a rota ligando $\mathbf{q}_{start}$ a $\mathbf{q}_{goal}$.

```
1  k ← 0; q_k ← q_start;
2  Inicia a matriz de covariâncias Σ_0 com Σ_min;
3  while not local_planner(q_k,q_goal) do
4      Gera uma nova configuração aleatória v_k ∈ N(0, Σ_k);
5      s ← q_k +v_k;
6      if local_planner(q_k,s) then
7          k ← k +1;
8          q_k ← s;
9          Atualiza a matriz de covariâncias Σ_k usando a Eq. (3.15);
10         Insere q_k na lista de configurações intermediárias;
```

Algoritmo 9: ARW simples.

desempenho no que se refere ao tempo de execução e à explorabilidade do espaço de configurações.

Quando $\mathcal{C} \in \mathbb{R}^2$, o planejador por frente de onda possui uma imensa vantagem em relação aos algoritmos probabilísticos. Além dele ser determinístico, completo e ótimo de acordo com o critério da menor distância de Manhattan, a sua implementação é bem simples, não necessitando de estruturas de dados complexas ou de algoritmos adicionais (planejador local, algoritmo para busca de vizinhos mais próximos, etc). Além disso, o único pós-processamento necessário consiste em afastar a rota dos obstáculos. No entanto, a facilidade de implementação desaparece quando $\mathcal{C} \in \mathbb{R}^2 \times \mathbb{S}^1$ e provavelmente a implementação se torna inviável para $\mathcal{C} \in \mathbb{R}^2 \times SO(2)$ ou $\mathcal{C}_{free} \in SE(2)$.

Por outro lado, os quatro algoritmos probabilísticos, além de possuírem bom desempenho em problemas de planejamento de rotas para robótica móvel, também possuem bom desempenho para espaços de configuração bem mais complexos, como os encontrados na animação de atores artificiais, em aplicações de biologia molecular, etc, conforme visto no Capítulo 1. No entanto, eles são apenas probabilisticamente completos, não sendo capazes de retornar a não existência de solução. Além disso, as rotas resultantes possuem uma baixa qualidade.

Dentre os quatro algoritmos probabilísticos, o ARW é o que possui a implementação mais simples, porém gera as piores rotas. Já o RRT é o algoritmo mais elegante, pois provê uma cobertura uniforme de $\mathcal{C}_{free}$, sendo incremental e ainda gerando rotas de boa qualidade. Além disso, o seu parâmetro tem uma relação intuitiva com o formato de $\mathcal{C}_{free}$, ou seja, quanto maior o volume das diversas regiões de $\mathcal{C}_{free}$, maior o valor de ϵ. Entretanto, para

topologias complexas é complicado dizer se o volume das diversas regiões de $\mathcal{C}_{free}$ é grande ou pequeno.

Conforme será mostrado no Capítulo 6, que corresponde aos resultados experimentais, a etapa de pós-processamento consome um tempo ínfimo quando comparado com o tempo total de execução do algoritmo, além das rotas finais serem praticamente iguais tanto para os algoritmos probabilísticos, quanto para o planejador por frente de onda. Evidentemente, as diferenças entre os algoritmos tendem a se acentuar quando a topologia do espaço de configurações é complicada ou quando sua dimensão é muito grande.

No que se refere aos planejadores que amostram o espaço de configurações, uma vantagem é que eles possuem uma grande independência em relação ao modelo dos obstáculos, assim como em relação à geometria do robô. De fato, o único conhecimento que o planejador tem é da topologia do espaço de configurações. A conexão entre a geometria do robô e o espaço de configurações livre é feita pelo detector de colisões. Dessa forma, o planejador tem a função de gerar amostras no espaço de configurações, enquanto o detector de colisão tem a função de verificar se elas devem ser rejeitadas ou aceitas. Então, para o planejador de rotas não interessa se os obstáculos são representados por grades de ocupação, primitivas semi-algébricas ou polígonos convexos, havendo apenas a necessidade da existência de um detector de colisões capaz de verificar se uma amostra gerada está ou não em colisão. Assim, é o detector de colisões que conhece o modelo geométrico e o formato dos obstáculos e do robô.

4 MÉTODOS DE AMOSTRAGEM E SUAVIZAÇÃO DE ROTAS

Do not go where the path may lead,
go instead where there is no path
and leave a trail.
Ralph Waldo Emerson

Com o grande sucesso dos planejadores de rotas baseados na amostragem do espaço de configurações na década de 1990, algumas questões inevitavelmente foram levantadas ao longo do tempo: seria possível amostrar seletivamente o espaço de configurações? Mesmo que a amostragem seja uniforme em todo o espaço de configurações, ela necessariamente tem que ser aleatória ou existem alternativas determinísticas [1]?

Vários trabalhos tentaram responder a primeira questão, principalmente no contexto do PRM e suas variantes. Em [11], é feita uma decomposição em células do espaço de trabalho visando extrair informações que sejam úteis para uma amostragem mais seletiva de $\mathcal{C}_{free}$. Em [10, 12, 50] são usadas funções densidade de probabilidade específicas para aumentar a amostragem de passagens estreitas no espaço de configurações. Mais especificamente, em [10] e [50], amostras são geradas perto dos obstáculos, verificando-se a melhoria da amostragem em passagens estreitas. Em [13] a amostragem é concentrada no eixo médio de $\mathcal{C}_{free}$. Por último, em [14] é feita uma avaliação entre vários métodos de amostragem comumente utilizados na literatura referente ao PRM. Assim, estes trabalhos mostraram que é possível amostrar seletivamente $\mathcal{C}_{free}$, mas nenhum apresentou uma técnica definitiva que funciona bem pelo menos na grande maioria dos casos. No que se refere à segunda pergunta, em [51] são avaliadas técnicas determinísticas para a amostragem global do espaço de configurações. Foi mostrado que estes métodos podem ocasionar uma cobertura mais homogênea de $\mathcal{C}_{free}$.

Ainda que grande parte das técnicas de amostragem exploradas sejam desenvolvidas visando a aplicação no PRM, muitas delas podem ser estendidas ou adaptadas para planejadores de questionamento único, notadamente o EST e o ARW.

4.1 AMOSTRAGEM DETERMINÍSTICA

Em Lavalle *et al.* [52] é mostrado um estudo consistente comparando várias técnicas de amostragem determinística e amostragem probabilística uniforme. Para avaliar e comparar os diferentes métodos, foram utilizados conceitos como discrepância e dispersão. Assim, seja $X = [0,1]^d \in \mathbb{R}^d$ o espaço no qual serão gerados os pontos, $P = \{p_0, \ldots, p_{N-1}\}$ o

[1] O termo amostragem na literatura de planejamento de rotas é usado para indicar que um espaço é aproximado por um conjunto de pontos, ou seja, discretizado de maneira não necessariamente homogênea.

conjunto finito de N pontos em X e $\mathcal{R}$ a coleção de subconjuntos de X chamada faixa. A discrepância é definida como [52]

$$D\left(P,\mathcal{R}\right)=\underset{R\in\mathcal{R}}{sup}\left|\frac{\mid P\cap R\mid_c}{N}-\mu\left(R\right)\right|, \tag{4.1}$$

em que $\mid\ \cdot\ \mid_c$ é aplicada a um conjunto finito e denota sua cardinalidade (quantidade de elementos em um conjunto discreto), e μ é a medida de Lebesgue (generalização do conceito de volume) de um conjunto. Sendo assim, a discrepância mede o maior erro de estimação de volume em todos os conjuntos em $\mathcal{R}$.

Exemplo 4.1.1. *Se para um conjunto de pontos é reinvindicado que ele provê uma discrepância nula, então dado que N amostras são geradas em $X=[0,1]^2$, o número de amostras dentro de todo subconjunto R, tal que $\mu(R)=\frac{1}{N}$, é igual a* 1. □

Ainda em [52] o conceito de dispersão é definido como

$$\delta\left(P,\rho\right)=\underset{x\in X}{sup}\left(\underset{p\in P}{min}\,\rho\left(x,p\right)\right), \tag{4.2}$$

em que ρ denota qualquer métrica (*e.g.* distância euclidiana). Dispersão pode ser considerada como o raio da maior bola em X que não contém algum elemento de P. Dessa forma, a dispersão pode ser utilizada como uma métrica para quantificar qual método provê um conjunto de pontos com maior cobertura de $\mathcal{C}_{free}$.

Além disso, em [52] é mostrado que uma baixa discrepância implica em baixa dispersão. Sendo assim, naquele trabalho foram explorados como métodos de baixa discrepância a seqüência de Halton, o conjunto de Hammerseley e uma generalização de grades denominada reticulados [53].

A seqüência de Halton para um espaço de dimensão d é dada da seguinte forma: d números primos entre si são escolhidos (em geral os d primeiros primos maiores ou iguais a 2 são escolhidos). Sabendo que um número pode ser representado em uma base p qualquer por meio da sequência $i_p=a_0+pa_1+p^2a_2+p^3a_3\ldots$, em que $a_j\in\{0,1,\ldots,p-1\}$ representa os dígitos do número (*e.g.* 123 na base decimal é igual a $3+2\cdot10+1\cdot10^2$), define-se a função

$$r_p\left(i\right)=\frac{a_0}{p}+\frac{a_1}{p^2}+\frac{a_2}{p^3}+\ldots, \tag{4.3}$$

tal que $r_p\left(i\right)\in[0,1]$ e a i-ésima amostra é dada por $\left(r_{p1}\left(i\right),r_{p2}\left(i\right),\ldots,r_{pd}\left(i\right)\right)$. Um algoritmo simples para gerar $r_p\left(i\right)$ é apresentado em [54].

O conjunto de pontos de Hammerseley é uma variação da seqüência de Halton, porém necessita da informação *a priori* sobre o número de amostras a serem geradas. Sendo assim, para gerar N amostras em um espaço de d dimensões são necessários $d-1$ primos e o conjunto é gerado de acordo com $\left(i/N,r_{p1}\left(i\right),\ldots,r_{p_{d-1}}\left(i\right)\right),i=0,1,\ldots,N-1$.

Por último, as grades não ortogonais, ou reticulados, também são conjuntos de pontos e N tem que ser determinado *a priori*. Para gerar a i-ésima de um total de N amostras em um espaço de d dimensões utiliza-se

$$\left(\frac{i}{N}, \{i\alpha_1\}, \ldots, \{i\alpha_{d-1}\}\right), i = 0, 1, \ldots, N-1, \tag{4.4}$$

em que α_i é um número positivo irracional e $\{\cdot\}$ representa a parte decimal do número real. A vantagem de usar reticulados é que a estrutura de vizinhança é regular, o que pode representar um ganho considerável de desempenho no PRM, uma vez que a busca de vizinhos no PRM clássico é uma etapa computacionalmente onerosa [52].

Já um método que gera um conjunto de amostras de baixa dispersão é a grade de Sukharev. Ele é baseado no critério de Sukharev que indica que, para qualquer conjunto de pontos P, a dispersão $\delta(P) \geq \frac{1}{2N^{\frac{1}{d}}}$, o que implica que para manter a dispersão fixa, o número de amostras necessárias cresce exponencialmente com a dimensão. Sendo assim, a grade de Sukharev possui dispersão $\delta(P) = \frac{N^{-\frac{1}{2}}}{2}$ e pode ser construída particionando $[0,1]^d$ em N hipercubos e colocando uma amostra no centro de cada hipercubo [52].

Um dos problemas associados à amostragem por conjunto de pontos é que o número N a ser definido *a priori* nem sempre é fácil de ser determinado. Porém, em [55] é apresentado um método para criação de um reticulado de forma incremental.

A grande vantagem dos métodos determinísticos de amostragem, em relação aos probabilísticos, é evidenciada quando o objetivo é fazer uma amostragem homogênea do espaço de configurações com a menor dispersão possível. Esta vantagem é ainda maior se a estrutura de vizinhança for regular, uma vez que não há necessidade da utilização de algoritmos para busca de vizinhos mais próximos. Entretanto, em boa parte das aplicacões, as rotas requisitadas precisam de amostras em apenas algumas partes de $\mathcal{C}_{free}$. Sendo assim, vários métodos estocásticos para amostrar seletivamente $\mathcal{C}_{free}$ foram desenvolvidos, como mostrado a seguir.

Em Sun *et al.* [12] é apresentado um método de amostragem seletiva cujo objetivo é fazer uma filtragem das amostras de forma a aumentar a probabilidade de amostrar as passagens estreitas. O método consiste em gerar uma amostra q_1 de acordo com uma distribuição uniforme em $\mathcal{C}_{obs}$ e a partir dela gerar uma amostra q_2 de acordo com uma distribuição gaussiana de média nula e matriz de covariâncias determinada pela largura das passagens estreitas. Caso q_2 também esteja em $\mathcal{C}_{obs}$ então a configuração q_m no ponto médio entre q_1 e q_2 é avaliada, sendo que se ela estiver em $\mathcal{C}_{free}$ ela é armazenada no mapa de rotas.

A idéia é que q_1 e q_2 formem os pilares e q_m seja a ponte sobre a passagem estreita. Este método simples se mostrou eficiente ao capturar a conectividade de passagens estreitas, sendo conhecido como *teste da ponte*.

Os métodos de amostragem gaussiana com seleção de amostras e de amostragem no eixo médio de $\mathcal{C}_{free}$ são descritos com mais detalhes a seguir.

4.1.1 Amostragem gaussiana com seleção de amostras

A técnica de amostragem gaussiana com seleção de amostras foi proposta com o objetivo de melhorar o desempenho do PRM no problema da passagem estreita [10]. Ela é mais onerosa que a amostragem uniforme, mas é justificada pelo princípio de que os dois passos computacionalmente mais onerosos do algoritmo PRM correspondem à geração de amostras e aos testes de conexão entre cada configuração aleatória e seus vizinhos por meio do planejador local. O tempo total de execução do PRM pode ser aproximado por [10]

$$T = n(T_s + T_a), \tag{4.5}$$

em que n corresponde ao número de amostras necessárias para resolver o problema, T_s o tempo necessário para gerar uma amostra em $\mathcal{C}_{free}$ e T_a o tempo necessário para adicionar a configuração ao mapa de rotas (o que inclui checar a conexão com os vizinhos).

Na implementação padrão T_s é muito menor que T_a, de forma que a idéia central deste algoritmo é gerar as configurações aleatórias de uma maneira muito mais criteriosa, diminuindo o número de amostras necessárias para resolver o problema. Sendo assim, mesmo que T_s aumente, há um menor número de nós gerados e portanto um menor número de chamadas ao planejador local. Conseqüentemente T_a diminui drasticamente, reduzindo o tempo global gasto na etapa de construção do mapa de rotas.

A estratégia de amostragem gaussiana apresenta bom desempenho em situações onde o espaço de configurações tem grandes áreas sem obstáculos e algumas poucas passagens estreitas. Sendo assim, o algoritmo amostra basicamente os locais difíceis de $\mathcal{C}_{free}$, evitando colocar amostras em grandes áreas abertas.

O seu funcionamento consiste em inicialmente gerar uma amostra $\mathbf{q}_1$ de acordo com uma distribuição uniforme em $\mathcal{C}$. Em seguida, uma configuração $\mathbf{q}_2$ é gerada seguindo

$$\mathbf{q}_2 \sim N(\mathbf{q}_1, \mathbf{\Sigma}). \tag{4.6}$$

Sendo assim, se $\mathbf{q}_1 \in \mathcal{C}_{free}$ e $\mathbf{q}_2 \notin \mathcal{C}_{free}$, então $\mathbf{q}_1$ é adicionada ao grafo. Caso contrário, se $\mathbf{q}_1 \notin \mathcal{C}_{free}$ e $\mathbf{q}_2 \in \mathcal{C}_{free}$, então $\mathbf{q}_2$ é adicionada ao grafo. Se nenhuma das condições anteriores forem atendidas, então ambas configurações são descartadas. O processo é repetido até que um número pré-determinado de amostras em $\mathcal{C}_{free}$ seja gerado. Este número depende da topologia do espaço de configurações, assim como do formato do espaço de trabalho. Dessa forma, nem sempre é fácil determiná-lo.

Este procedimento de geração de configurações pode ser resumido pelo Algoritmo 10. Apesar deste método ter sido inicialmente desenvolvido para ser utilizado com o PRM, ele pode ser facilmente adaptado para ser utilizado em algoritmos de questionamento único tal como o ARW ou o EST. Observa-se que um dos parâmetros do algoritmo é a variância da distribuição gaussiana. Quanto menor esta variância, maior será o número de amostras geradas perto de $\mathcal{C}_{obs}$. Conseqüentemente, um número maior de amostras serão geradas

Entrada: Número N de amostras a serem geradas, variância $\mathbf{\Sigma}$ da distribuição gaussiana.

Saída : Mapa de rotas.

```
1  i ← 0;
2  while i < N do
3      q1 ← é gerada de acordo com uma distribuição uniforme em C_free;
4      q2 ← é gerada de acordo com a Eq. (4.6);
5      if q1 ∈ C_free and q2 ∉ C_free then
6          adiciona q1 ao mapa de rotas;
7          i ++;
8      else if q1 ∉ C_free and q2 ∈ C_free then
9          adiciona q2 ao mapa de rotas;
10         i ++;
```

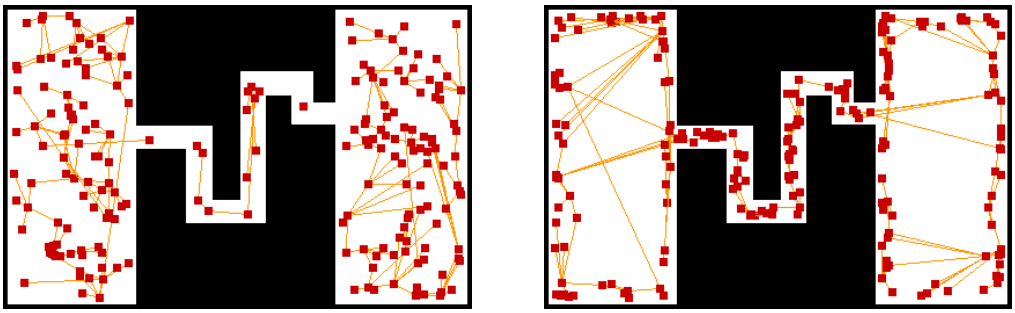
Algoritmo 10: Amostragem gaussiana com seleção de configurações aleatórias.

em passagens estreitas. Entretanto, se esta variância for muito pequena, um número maior de amostras serão rejeitadas, pois a amostra $\mathbf{q}_2$ será gerada próxima à amostra $\mathbf{q}_1$. Assim, haverá uma grande probabilidade de ambas estarem em $\mathcal{C}_{obs}$ ou ambas estarem em $\mathcal{C}_{free}$, o que implica na rejeição das duas amostras. Por outro lado, se a variância for muito grande, áreas irrelevantes de $\mathcal{C}_{free}$ podem ser amostradas.

A Figura 4.1 mostra um tipo de ambiente no qual a amostragem gaussiana, na média, possui um desempenho bem superior à amostragem uniforme. Com parâmetros idênticos, mudando apenas a distribuição utilizada para amostragem, a distribuição gaussiana com seleção de amostras consegue capturar a conectividade do corredor central bem antes da distribuição uniforme. Este fato é comprovado por meio dos resultados das simulações do Capítulo 6. Isto acontece porque, no PRM uniforme, a probabilidade de uma amostra cair em uma região é proporcional ao seu volume. Então, a probabilidade de uma amostra ser gerada em uma passagem estreita é pequena devido ao seu pequeno volume. Já o PRM gaussiano foi projetado visando justamente capturar a conectividade de passagens estreitas. Assim, como nesta figura existe apenas uma passagem estreita, que consiste em um corredor separando duas grandes regiões, o algoritmo que polariza a amostragem rumo ao corredor consegue capturar melhor a conectividade de $\mathcal{C}_{free}$.

4.1.2 Amostragem no eixo médio

Visando também aumentar o desempenho do PRM em espaços de configuração com passagens estreitas, foi desenvolvido em [13] um método de amostragem em que as amostras



(a) Amostragem uniforme.

(b) Amostragem gaussiana com seleção de amostras.

Figura 4.1: Amostragem uniforme vs. amostragem gaussiana no algoritmo PRM.

geradas no espaço de configurações são levadas para o eixo médio $MA(\mathcal{C}_{free})$ de $\mathcal{C}_{free}$ sem a necessidade de seu cálculo explícito [13].

Além de haver um aumento da distância das amostras em relação aos obstáculos, em [13] é mostrado que a conectividade em passagens estreitas pode ser melhor capturada com este método e o aumento desta conectividade não depende do volume destas passagens, mas sim das características dos obstáculos ao redor.

A Figura 4.2 mostra a representação do eixo médio para $\mathcal{C} \in \mathbb{R}^2$. O polígono sólido em forma de U, assim como os limites da figura, representam $\mathcal{C}_{obs}$. O eixo médio é representado pela linha tracejada. Observa-se que cada ponto do eixo médio é eqüidistante a pelo menos dois pontos de $\mathcal{C}_{obs}$.

Uma grande vantagem de usar o eixo médio do espaço de configurações livre é que $MA(\mathcal{C}_{free})$ tem dimensão menor que $\mathcal{C}_{free}$ mas ainda é uma representação completa para os propósitos de planejamento de rotas, uma vez que mantém a estrutura topológica de $\mathcal{C}_{free}$ [13]. É importante notar que no caso de polígonos no plano toda configuração amostrada no espaço de configurações pode ser levada para o eixo médio [13], o que significa que a técnica pode ser facilmente aplicada em problemas de robótica móvel. Um exemplo é mostrado na Seção 4.2.3, em que as configurações geradas em $\mathcal{C}_{free}$ são levadas para para o eixo médio, visando o afastamento do robô em relação aos obstáculos.

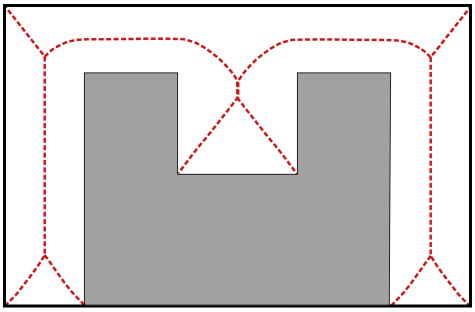


Figura 4.2: Eixo médio em $\mathcal{C} \in \mathbb{R}^2$ representado pela linha pontilhada.

Entrada: Número N de nós a serem gerados

Saída : Mapa de rotas localizado em $\mathcal{C}_{free}$

1 **repeat**
2 Gera uma amostra q em $\mathcal{C}$ de acordo com uma distribuição uniforme;
3 $q_n \leftarrow$ `nearest_configuration(`q`)` na fronteira de $\mathcal{C}_{free}$;
4 **if** $q \in \mathcal{C}_{free}$ **then**
5 O sentido $\vec{v}$ do deslocamento é $\overrightarrow{q_n\, q}$ e o ponto de partida s do deslocamento é q;
6 **else**
7 O sentido $\vec{v}$ do deslocamento é $\overrightarrow{q\, q_n}$ e o ponto de partida s do deslocamento é q_n;
8 Move s na direção $\vec{v}$ até que q_n não seja a única configuração mais próxima na fronteira de $\mathcal{C}_{free}$;
9 **until** *N nós sejam gerados em* $\mathcal{C}_{free}$;
10 **forall** *combinação dois a dois dos nós gerados* **do**
11 **if** `local_planner(`q_i,q_j`)` **then**
12 Insere uma borda no grafo conectando q_i a q_j

Algoritmo 11: Amostragem no eixo médio de $\mathcal{C} \in \mathbb{R}^2$.

O Algoritmo 11 resume o funcionamento da amostragem do eixo médio para $\mathcal{C} \in \mathbb{R}^2$. A idéia é que toda amostra gerada em $\mathcal{C}$ seja levada para o eixo médio de $\mathcal{C}_{free}$. Assim, enquanto na Linha 4 as amostras geradas em $\mathcal{C}_{free}$ são levadas para seu eixo médio, na Linha 6 observa-se que as amostras que caem em $\mathcal{C}_{obs}$ também são aproveitadas e levadas para o eixo médio do espaço de configurações livre. Sendo assim, ainda que o processo de amostragem seja mais oneroso computacionalmente, não há desperdício de amostras e é necessário gerar um número menor de amostras para capturar a conectividade de $\mathcal{C}_{free}$. Já na Linha 8, partindo do ponto s pequenos incrementos são dados na direção $\vec{v}$. A cada um desses incrementos, verifica-se se existe algum outro ponto na fronteira de $\mathcal{C}_{free}$ cuja distância seja igual àquela entre v e q_n[2]. Caso exista, então a configuração s encontra-se no eixo médio. Este processo pode ser bastante oneroso computacionalmente, pois mesmo para o caso $\mathcal{C} \in \mathbb{R}^2$, o tempo necessário para encontrar pontos na fronteira de $\mathcal{C}_{free}$ depende fortemente da implementação do algoritmo utilizado para encontrar objetos mais próximos.

[2]Na verdade, para evitar problemas devido ao efeito da discretização, verifica-se se existe algum outro ponto cuja distância d_1 é *aproximadamente* igual àquela d_2 entre s e q_n, ou seja, $d_1 = d_2 \pm \delta$.

4.2 COMPONENTES PARA SUAVIZAÇÃO DE ROTAS

Conforme discutido nas Seções anteriores, os métodos probabilísticos geralmente retornam rotas de baixa qualidade. Assim, um pós-processamento é necessário para retirada de nós redundantes, ciclos e desvios desnecessários. Pelo que mostra a literatura de planejamento de rotas, ainda não foi desenvolvido um método único de suavização que faça essas três coisas concomitantemente. Dessa forma, vários passos intermediários são necessários para a completa suavização da rota. Os algoritmos que representam estes passos intermediários são apresentados a seguir.

4.2.1 Dividir para conquistar

No algoritmo Dividir para Conquistar [9], dadas as configurações inicial e final de uma lista contendo as configurações intermediárias de uma rota qualquer, o objetivo é tentar ligar a primeira configuração à ultima utilizando um planejador local. Caso isto seja possível, os nós intermediários são removidos. Caso contrário, a lista é dividida em duas e o processo é repetido recursivamente nas listas menores.

Este procedimento pode ser resumido pelo Algoritmo 12 e deve ser chamado [3] repetidamente até que a rota não possa mais ser simplificada. Isto é determinado quando, em uma nova chamada do algoritmo, não há remoção de qualquer configuração da rota. Além disso, este algoritmo possui um excelente desempenho e em poucas chamadas ele atende ao critério de parada [9, 34]. Uma outra vantagem deste método é que ele é bastante geral e se aplica a qualquer tipo de espaço de configurações.

A Figura 4.3 mostra uma rota após quatro chamadas ao algoritmo Dividir para Conquistar. A rota original foi gerada pelo algoritmo ARW e possui muitos nós desnecessários, sendo mostrada na Figura 4.3(a). As Figuras 4.3(b)-(c) mostram a primeira e a segunda chamada ao algoritmo. Observa-se que já na primeira chamada a rota sofre uma grande simplificação, sendo que na segunda a rota está com bem menos nós que a rota original. Porém, existem ainda alguns nós redundantes, de tal maneira que mais chamadas ao algoritmo são necessárias para que o critério de parada seja atendido. A Figura 4.3(d) mostra a quarta chamada, em que a rota já está bem simplificada, com apenas 10 nós.

4.2.2 Utilização de atalhos e remoção de ciclos

Após a aplicação sucessiva do Algoritmo 12 apresentado na Seção 4.2.1, podem existir alguns nós remanescentes que causam desvios e ciclos desnecessários na rota. Uma boa

[3] É importante notar que uma chamada refere-se ao nível mais alto do algoritmo, sem considerar a recursividade. Uma vez que o algoritmo é chamado, ele é encarregado de suavizar a rota recursivamente, mas quem o chamou não precisa saber que ele funciona de maneira recursiva.

Entrada: q_{start}, q_{end} e a lista contendo a rota a ser suavizada

Saída : Lista que contém a rota suavizada

```
1 if (q_start = q_end) or (q_start+1 = q_end) then
      /* A rota não pode mais ser dividida, então interrompe
         esta execução.                                        */
2     return;
3 else if local_planner(q_start,q_end) then
4     remove_nodes_interval(q_start+1,q_end-1);
5 else
      /* Faz chamadas recursivas ao algoritmo dividir para
         conquistar                                            */
6     divide_and_conquer(q_start,q_(start+end)/2);
7     divide_and_conquer(q_(start+end)/2+1,q_end);
8 return Rota suavizada
```

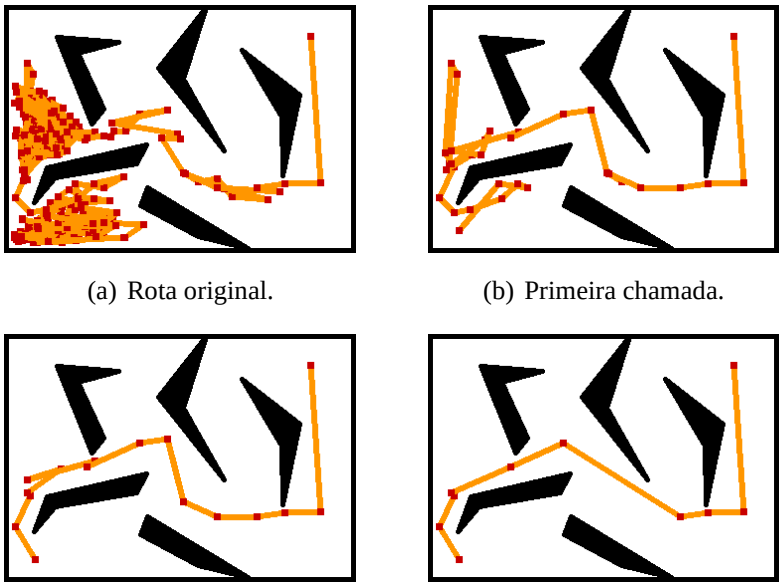
Algoritmo 12: Dividir para conquistar.

estratégia para eliminar esses nós é utilizar atalhos na rota. Assim, um algoritmo simples e geral baseado naquele proposto por Geraerts & Overmars [16] é utilizado para a utilização de atalhos e remoção de ciclos. Assim como o algoritmo Dividir para Conquistar, este algoritmo funciona para espaços de configuração arbitrários.

Partindo do nó inicial, o objetivo é percorrer a rota de trás para frente para tentar ligá-lo a um nó q_j o mais próximo possível do nó final. Caso exista este nó, os nós intermediários são removidos e o processo é repetido a partir de q_j. Este processo é repetido até que o penúltimo nó da rota seja alcançado. O Algoritmo 13 resume este procedimento. Para que haja uma suavização mais agressiva, às vezes se faz necessário interpolar nós intermediários antes de fazer a remoção de ciclos.

Em Geraerts & Overmars [16] a varredura é feita no sentido direto, enquanto neste trabalho ela é realizada no sentido reverso. Esta diferença, ainda que sutil, permite um ganho de desempenho, pois trechos maiores são eliminados e menos passos são necessários para a suavização da rota.

A Figura 4.4(a) mostra o resultado de uma rota processada pelo algoritmo Dividir para Conquistar. Observa-se que há um grande ciclo desnecessário que não pôde ser eliminado com este método. Assim, o algoritmo Remoção de Ciclos é utilizado, gerando a rota da Figura 4.4(b). Para ter este resultado, foi necessária a interpolação de configurações intermediárias na rota, uma vez que com mais nós há um aumento da possibilidade de conexões, o que implica em uma maior suavização.



(a) Rota original. (b) Primeira chamada.

(c) Segunda chamada. (d) Quarta chamada.

Figura 4.3: Algumas chamadas ao algoritmo Dividir para Conquistar.

4.2.3 Aumento da distância em relação a obstáculos

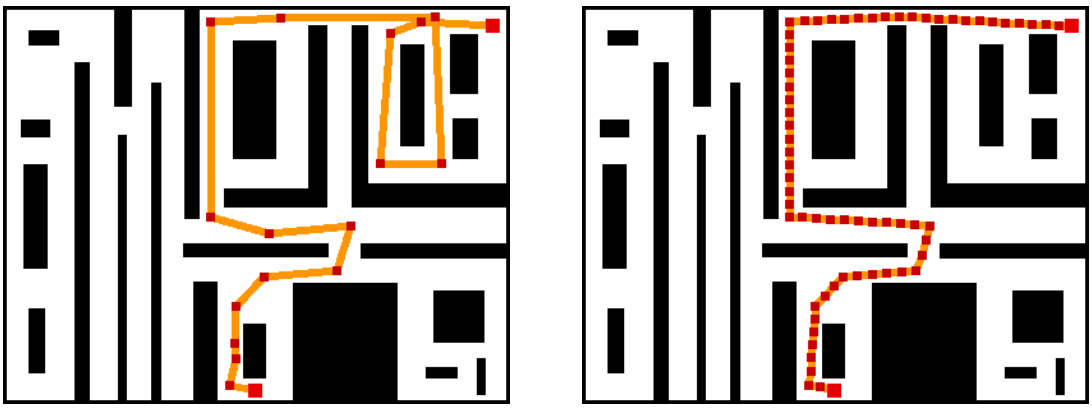
Todos os métodos de suavização anteriores não levam em consideração a distância em relação aos obstáculos, uma vez que eles tendem a otimizar a distância total da rota (no caso limite da otimização a rota pode passar arbitrariamente perto da fronteira dos obstáculos). Em robótica móvel isto pode ser inviável, uma vez que as representações do ambiente tendem a conter incertezas associadas. Assim, os obstáculos podem ter uma ligeira variação da posição real em relação à posição representada no mapa. Desta forma, rotas muito próximas aos obstáculos podem gerar colisões quando o robô for executá-las.

A distância em relação aos obstáculos considerada segura é dada pelos requisitos da tarefa. Contudo, existem situações em que o robô tem que passar por caminhos muito estreitos,

Entrada: q_{start}, q_{goal} e lista com a rota a ser suavizada
Saída : Lista que contém a rota suavizada

```
1 for q_i ← q_start to q_goal−1 do
2     for q_j ← q_goal to q_i+1 do
3         if local_planner(q_i, q_j) then
4             remove_nodes_interval(q_i+1,q_j−1);
5             q_i ← q_j;
              /* Interrompe o laço mais interno da linha 2    */
6             break;
```

Algoritmo 13: Remove ciclos e utiliza atalhos



(a) Antes da remoção dos ciclos. (b) Depois da remoção dos ciclos.

Figura 4.4: Remoção de ciclos na rota.

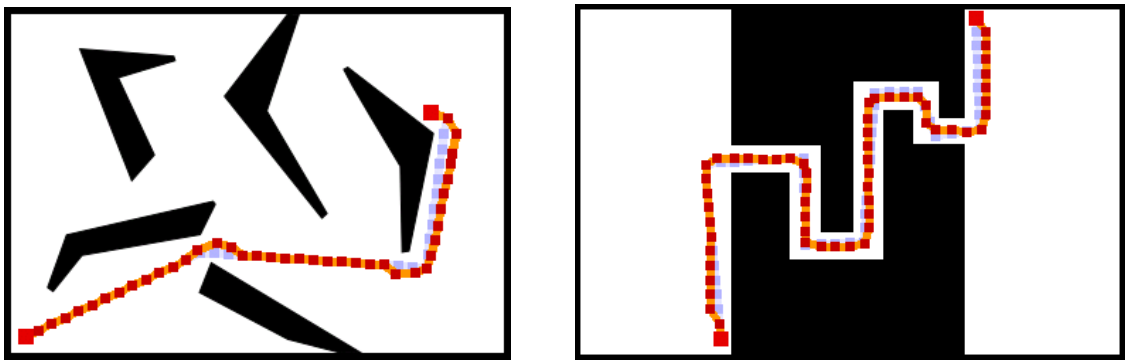
sendo que o critério da distância mínima não pode ser satisfeito. Nessas situações, o mais adequado é levar o ponto de passagem para o eixo médio de $\mathcal{C}_{free}$, maximizando a segurança do robô.

O Algoritmo 11, utilizado originalmente no contexto do PRM para gerar amostras no eixo médio, pode ser simplificado de forma a aumentar a distância dos nós em relação aos obstáculos para $\mathcal{C} \in \mathbb{R}^2$. Para tanto, basta eliminar a etapa de geração de amostras e conexão e então utilizar somente o trecho que corresponde ao aumento da distância em relação aos obstáculos. Este trecho corresponde às Linhas 4 a 8. Assim, inicialmente a rota é interpolada com nós intermediários e estes são afastados dos obstáculos se estiverem a uma distância menor que a distância mínima desejada. Eles devem ser deslocados até a distância mínima ou, caso isto não seja possível, afastados até que atinjam o eixo médio do espaço de configurações.

A Figura 4.5 mostra dois ambientes nos quais a rota original está desenhada em tons claros e a rota afastada dos obstáculos está desenhada em tons escuros. Nota-se que muitos nós atendiam ao critério da distância mínima e não precisaram ser afastados. Na Figura 4.5(a) a parte da rota mais à direita foi afastada até que satisfizesse o critério da distância de segurança e não chegou ao eixo médio. Já na Figura 4.5(b) o corredor central é muito estreito, de tal forma que não é possível fazer com que os nós daquela região atendam ao critério da distância mínima. Sendo assim, os nós são levados para o eixo médio, maximizando a distância da rota em relação aos obstáculos.

4.2.4 Remoção de ramos

O mapeamento de pontos do espaço de configurações no eixo médio não é um homeomorfismo. Isto faz com que os nós distintos situados perto de concavidades e vértices de obstáculos possam ser mapeados no mesmo ponto, ocasionando “ramos” nas rotas. A Figura 4.6 mostra um exemplo de três nós diferentes no espaço de configurações que são mapeados



(a) Rota gerada em um ambiente simples. (b) Rota gerada em um corredor estreito.

Figura 4.5: Rota original (tons claros) e rota afastada dos obstáculos (tons escuros).

no mesmo ponto do eixo médio.

O algoritmo 14 retira os ramos das rotas de acordo com um parâmetro de distância s_{min} [16]. Dado um nó de referência q_i, se entre os nós q_{i-1} e q_{i+1} a distância for menor que s_{min} e se o planejador local conseguir ligar estes dois nós, q_i é retirado da rota. O nó de referência passa a ser q_{i-1}, se ele não for o primeiro nó da rota ou, caso contrário, ele passa a ser q_{i+1}. Este método é bastante geral e funciona para espaços de configuração arbitrários.

A Figura 4.7(a) mostra uma rota que foi levada ao eixo médio. Uma vez que várias configurações foram mapeadas em um mesmo ponto de $\mathcal{C}_{free}$, a rota ficou com grandes ramos. Estes são caracterizados por trechos das rotas em que os pontos de passagem se sobrepõem, fazendo com que o robô, ao executar a rota, passe por esses pontos duas vezes, porém em sentidos opostos. A Figura 4.7(b) mostra a rota suavizada. É importante notar que somente os ramos são removidos, sendo o restante da rota preservada.

4.2.5 Discussão

Como visto no Capítulo 3, os métodos probabilísticos retornam rotas de baixa qualidade. Isto implica na necessidade de um pós-processamento das rotas. Os métodos apresentados

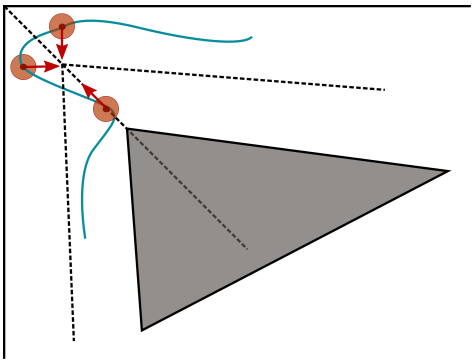
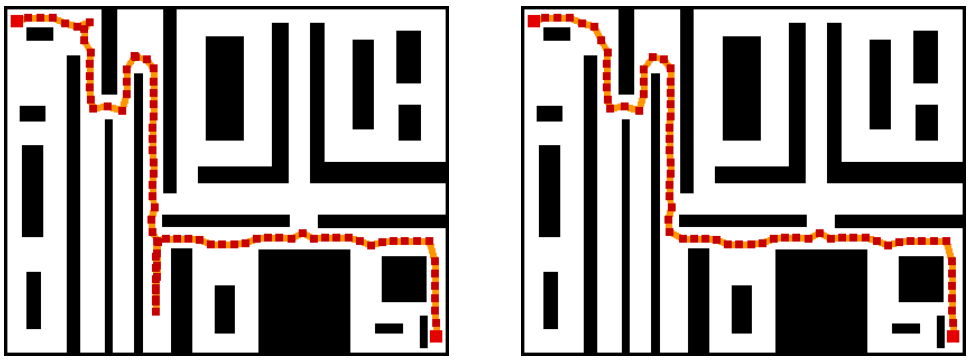


Figura 4.6: Vários nós mapeados no mesmo ponto do eixo médio.

```
Entrada: q_start e s_min
1  q_i ← q_start+1;
2  while q_i+1 not Nil do
3      if distance(q_i-1, q_i+1) < s_min then
4          if local_planner(q_i-1, q_i+1) then
5              aux ← q_i;
6              if q_i-1 not q_start then
7                  q_i ← q_i-1;
8              else
9                  q_i ← q_i+1;
10             remove(aux);
               /* Volta para o início do while                    */
11             continue;
12     q_i ← q_i+1;
```

Algoritmo 14: Remove os ramos da rota.

na Seção 4.2 são eficientes, porém individualmente não são algoritmos fechados, que por si só já retornam uma boa rota. Assim, se faz necessária a aplicação dos vários métodos à rota visando sua suavização completa. Com exceção do método que afasta os nós dos obstáculos, todos os métodos são bastante gerais e podem ser aplicados em espaços de configuração arbitrários. Além disso, os algoritmos são simples, sendo de fácil implementação. Contudo, o simples fato de ser necessário o uso de vários algoritmos intermediários, mesmo que simples, torna o método global não muito elegante. Assim, seria desejável que houvesse uma solução fechada, mas até o ponto conhecido pelo autor, ainda não foi desenvolvido um método elegante, de solução fechada e que seja computacionalmente eficiente, que resolva o problema de suavização de rotas.



(a) Antes da remoção de ramos. (b) Após a remoção de ramos.

Figura 4.7: Remoção de ramos

5 REVISITANDO O PASSEIO ALEATÓRIO ADAPTATIVO

Do not worry about your difficulties in mathematics,
I assure you that mine are greater.
Albert Einstein

Na Seção 3.5 foi apresentado um algoritmo de questionamento único, o ARW, cujo mecanismo de exploração do espaço de configurações é dado por um passeio aleatório. A variação do único parâmetro a ser ajustado – a história H do processo – não implica em um impacto muito significativo no desempenho do algoritmo [9]. Além disso, a literatura mostra que este algoritmo é bastante eficiente quando comparado com os algoritmos de questionamento único que compõem o estado da arte de planejamento de rotas [9]. Porém, uma desvantagem deste algoritmo está na sua dificuldade em sair de longos corredores[1] do espaço de configurações [56].

Tendo em vista essas características do ARW, algumas questões ficam pendentes: seria possível alterar a função densidade de probabilidade de forma a maximizar a explorabilidade? Além disso, qual é a função densidade de probabilidade final do algoritmo na forma em que ele foi projetado? Existe alguma maneira de polarizar a amostragem rumo a áreas menos exploradas do espaço de configurações?

Nas seções seguintes essas questões são discutidas. Na Seção 5.1 é feita uma caracterização da função densidade de probabilidade resultante do ARW e, em seguida, na Seção 5.2 a condição de convergência para o algoritmo em um espaço discretizado é apresentada. Já na Seção 5.3 é feita uma comparação entre a distribuição uniforme e a distribuição gaussiana. Em seguida, na Seção 5.4 um método de fácil implementação usando grades no espaço de configurações é proposto visando aumentar a explorabilidade do algoritmo. Por último, a Seção 5.5 propõe um algoritmo incremental baseado em passeios aleatórios adaptativos. As análises teóricas que vêm a seguir, assim como o algoritmo incremental resultante desenvolvido neste trabalho, não estão presentes na literatura, e portanto constituem a principal contribuição desta dissertação.

[1] Um corredor no espaço de configurações, de dimensão arbitrária, para um robô holonômico é dado por regiões em que o robô pode se mover, sem muita restrição espacial, apenas em uma direção.

5.1 CARACTERIZAÇÃO DA FUNÇÃO DENSIDADE DE PROBABILIDADE DO ARW

O passeio aleatório adaptativo, já descrito na Seção 3.5, é replicado a seguir. Dadas as configurações inicial $\mathbf{q}_{start}$ e final $\mathbf{q}_{goal}$, tem-se que

$$\mathbf{q}_0 = \mathbf{q}_{start}, \tag{5.1}$$

$$\mathbf{q}_k = \mathbf{g}(\mathbf{q}_{k-1}, \mathbf{v}_k). \tag{5.2}$$

As amostras $\mathbf{v}_k$ são geradas seguindo uma distribuição uniforme de média $\mathbf{q}_{k-1}$ e matriz de covariâncias adaptativa. A função $\mathbf{g}(\mathbf{q}_{k-1}, \mathbf{v}_k)$ avalia se $\mathbf{q}_{k-1}$ pode ser ligada a $\mathbf{v}_k$. Como pôde ser visto na Seção 3.5, esta função, quando implementada na forma de um algoritmo, é chamada de planejador local. Assim, visando uma análise mais aprofundada do algoritmo, nesta Seção é caracterizada a função densidade de probabilidade das amostras geradas no algoritmo ARW.

Inicialmente, assume-se que o espaço de configurações seja uma variedade. De fato, na grande maioria dos problemas práticos em robótica esta assertiva é válida [36, 2, 19]. Sendo $\mathcal{C}$ uma variedade, então localmente ele é homeomorfo a um espaço Euclidiano, e conseqüentemente $\mathcal{C}_{free}$ também o é. Assim, uma vez que há um homeomorfismo entre o espaço de configurações livre e um espaço Euclidiano, então existe uma função que retorna o volume de $\mathcal{C}_{free}$ que é homeomorfa a uma função que retorna o volume do espaço Euclidiano correspondente. Uma vez que $\mathbb{R}^k$ é homeomorfo ao conjunto aberto $(0,1)^k$ e este tem volume unitário, então assume-se, sem perda de generalidade, que $\mathcal{C}_{free}$ tenha volume unitário. Esta mesma consideração sobre o volume de $\mathcal{C}_{free}$ já foi utilizada em outros trabalhos para caracterizar distribuições não uniformes no contexto do PRM [12]. Assim, a função densidade de probabilidade para a primeira amostra em $\mathcal{C}_{free}$ é dada por[2]

$$p(\mathbf{q}_0) = \begin{cases} 1 & \mathbf{q}_0 \in \mathcal{C}_{free} \\ 0 & \text{caso contrário,} \end{cases} \tag{5.3}$$

e $\int_{\mathcal{C}_{free}} p(\mathbf{q}_0) d\mathbf{q}_0 = 1$. Assim, a distribuição da primeira amostra do passeio aleatório é uniforme em todo o domínio de $\mathcal{C}_{free}$. Isto porque não há restrição sobre onde o passeio aleatório possa começar. Então, como o espaço de configurações livre depende do formato do espaço de trabalho e este pode ser arbitrário, a distribuição da primeira configuração tem que necessariamente ser uniforme, pois não existe informação privilegiada sobre onde o passeio aleatório deve começar.

O objetivo é partir da amostragem inicial do ARW e da operação de aceitação/rejeição que ocorre em seqüência e, então, encapsular estas duas operações em uma distribuição que caracterize a evolução do passeio aleatório adaptativo.

[2] Visando uma notação mais simples, a função densidade de probabilidade $p_X(x)$ nesta Seção é escrita apenas como $p(x)$.

Assim, a amostragem inicial, dada a amostra $\mathbf{q}_{k-1} \in \mathcal{C}_{free}$, é caracterizada por

$$\lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) = N(\mathbf{q}_{k-1}, \mathbf{\Sigma}_{\mathbf{q}_{k-1}}), \tag{5.4}$$

em que a amostra $\mathbf{q}_k$ é gerada por meio da distribuição gaussiana $\lambda(\mathbf{q}_k, \mathbf{q}_{k-1})$, chamada distribuição de base. Porém, ela só é aceita se estiver na região de visibilidade de $\mathbf{q}_{k-1}$, cuja definição vem a seguir.

Definição 5.1.1. *Seja $\pi(\cdot)$ uma função contínua qualquer com domínio $[0, 1]$. A região de visibilidade a partir do ponto $\mathbf{q}$ é dada por*

$$\mathcal{V}(\mathbf{q}) = \{\mathbf{q}_v \in \mathcal{C}_{free} \mid \pi : [0, 1] \rightarrow \mathcal{C}_{free},\ \pi(0) = \mathbf{q},\ \pi(1) = \mathbf{q}_v\}, \tag{5.5}$$

□

Define-se ainda a função de pertinência

$$I(\mathbf{q}_k, \mathbf{q}_{k-1}) = \begin{cases} 1 & \text{, se } \mathbf{q}_k \in \mathcal{V}(\mathbf{q}_{k-1}), \\ 0 & \text{, caso contrário .} \end{cases} \tag{5.6}$$

Esta função de pertinência geralmente é avaliada, em nível de implementação, pelo planejador local, para se certificar que $\mathbf{q}_k$ é visível a partir de $\mathbf{q}_{k-1}$.

A função densidade de probabilidade condicional, que caracteriza a geração de amostras $\mathbf{q}_k$ em função de uma configuração $\mathbf{q}_{k-1}$ de referência, é dada por

$$p(\mathbf{q}_k|\mathbf{q}_{k-1}) = \frac{\lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) I(\mathbf{q}_k, \mathbf{q}_{k-1})}{\int_{\mathcal{C}_{free}} \lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) I(\mathbf{q}_k, \mathbf{q}_{k-1}) d\mathbf{q_k}}. \tag{5.7}$$

É importante notar que k é o índice da k-ésima amostra gerada em $\mathcal{C}_{free}$.

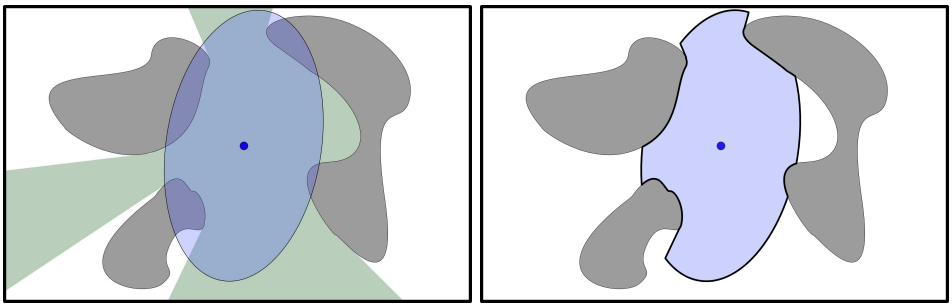
A Figura 5.1 mostra uma representação visual para a construção da função densidade de probabilidade condicional dada pela Eq. (5.7). A Figura 5.1(a) mostra um ponto no centro do mapa com a respectiva elipse 3σ da distribuição de base e a região de visibilidade a partir do ponto central. Para este exemplo, assume-se que todos os pontos da região de visibilidade podem ser conectados ao ponto central $\mathbf{q}$ de referência por meio de segmentos de retas, ou seja, $\mathcal{V}(\mathbf{q}) = \{\mathbf{q}_v \in \mathcal{C}_{free} \mid t\mathbf{q} + (1-t)\mathbf{q}_v \in \mathcal{C}_{free},\ t \in [0, 1]\}$. Já a Figura 5.1(b) mostra o resultado da intersecção da função densidade de probabilidade da distribuição de base e a função de pertinência, que por sua vez é relacionada com a região de visibilidade.

Pelas Equações B.18 e B.22 do Apêndice B nota-se que

$$p(\mathbf{q}_k) = \int_{\mathcal{C}_{free}} p(\mathbf{q}_k|\mathbf{q}_{k-1}) p(\mathbf{q}_{k-1}) d\mathbf{q}_{k-1}. \tag{5.8}$$

Substituindo 5.7 em 5.8, tem-se que

$$p(\mathbf{q}_k) = \int_{\mathcal{C}_{free}} \frac{\lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) I(\mathbf{q}_k, \mathbf{q}_{k-1})}{\int_{\mathcal{C}_{free}} \lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) I(\mathbf{q}_k, \mathbf{q}_{k-1}) d\mathbf{q_k}} p(\mathbf{q}_{k-1}) d\mathbf{q}_{k-1} \tag{5.9}$$



(a) Elipse 3σ da distribuição gaussiana, região de visibilidade do robô (tons claros) e obstáculos (tons escuros).

(b) Distribuição $p(\mathbf{q}_k|\mathbf{q}_{k-1})$ resultante.

Figura 5.1: Caracterização da distribuição condicional $p(\mathbf{q}_k|\mathbf{q}_{k-1})$ do ARW gaussiano.

Fazendo

$$f(\mathbf{q}_k, \mathbf{q}_{k-1}) = \int_{\mathcal{C}_{free}} \lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) I(\mathbf{q}_k, \mathbf{q}_{k-1}) d\mathbf{q}_\mathbf{k}, \tag{5.10}$$

então, tem-se que a distribuição que caracteriza o ARW é dada por

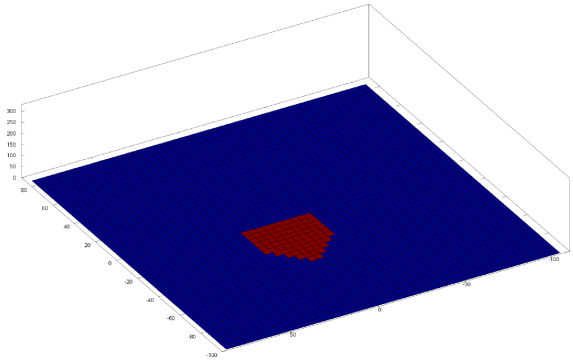
$$p(\mathbf{q}_k) = \int_{\mathcal{C}_{free}} \frac{\lambda(\mathbf{q}_k, \mathbf{q}_{k-1}) I(\mathbf{q}_k, \mathbf{q}_{k-1})}{f(\mathbf{q}_k, \mathbf{q}_{k-1})} p(\mathbf{q}_{k-1}) d\mathbf{q}_{k-1}. \tag{5.11}$$

Os incrementos do passeio aleatório são dados por $\mathbf{u}_k = \mathbf{q}_k - \mathbf{q}_{k-1}$, em que $\mathbf{q}_k$ é gerada seguindo a distribuição dada pela Eq. (5.11). Assim, o passeio aleatório pode ser reescrito como uma soma de variáveis aleatórias, ou seja,

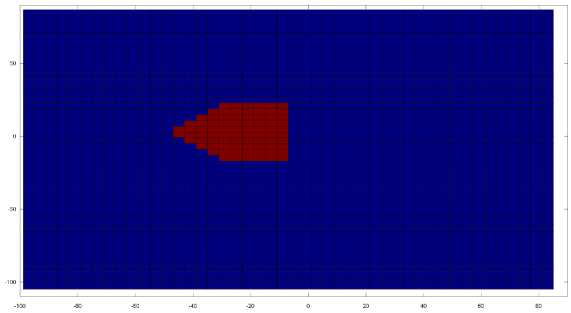
$$\mathbf{q}_k = \mathbf{q}_0 + \sum_{n=1}^{k} \mathbf{u}_n. \tag{5.12}$$

A principal vantagem de reescrever o problema como a Eq. (5.12) é que há uma bibliografia consolidada sobre passeios aleatórios escritos nesta forma, como por exemplo o livro clássico de Spitzer [57]. Contudo, conforme pode ser visto na Seção 5.2, a classe de passeios aleatórios abordada nesta dissertação é uma generalização da formalização de Spitzer.

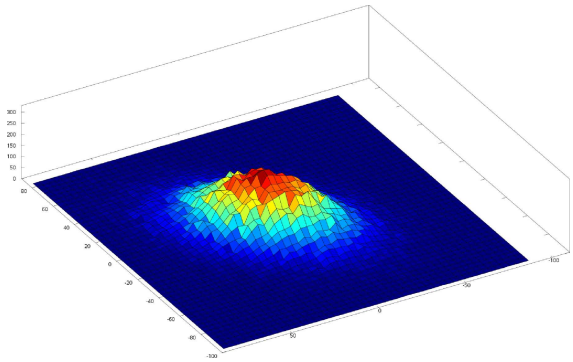
A Figura 5.2 mostra um exemplo de histograma da distribuição $p(\mathbf{q}_k)$. Para esta simulação, a região de visibilidade foi definida como $\mathcal{V}(\mathbf{q}) = \mathbf{q}_v \in \mathcal{C}_{free}$. Isto significa que toda amostra em $\mathbf{C}_{free}$ é visível a partir do ponto de referência. A Figura 5.2(a) mostra um obstáculo mapeado no espaço de configurações. $\mathcal{C}_{obs}$ é representado em vermelho, enquanto que $\mathcal{C}_{free}$ é representado em azul. A Figura 5.2(c) mostra o histograma da distribuição de base gerado para a configuração de referência $\mathbf{q} = [0, 0]^T$. O histograma da distribuição resultante é dado pela Figura 5.2(e). Notam-se alguns aspectos importantes. O primeiro é que curva do histograma da distribuição final está mais elevada que o da distribuição de base nas partes referentes a $\mathcal{C}_{free}$. Isto acontece porque as amostras que foram geradas em $\mathcal{C}_{obs}$ pela *distribuição de base* foram todas rejeitadas e reamostradas em $\mathcal{C}_{free}$, aumentando a frequência nesta região correspondente da *distribuição final*. Assim, como esperado, nenhuma amostra



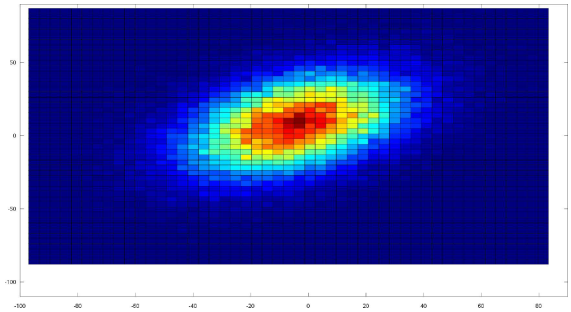
(a) Exemplo de $\mathcal{C} \in \mathbb{R}^2$, sendo $\mathcal{C}_{free}$ e $\mathcal{C}_{obs}$ representados em azul e em vermelho, respectivamente.



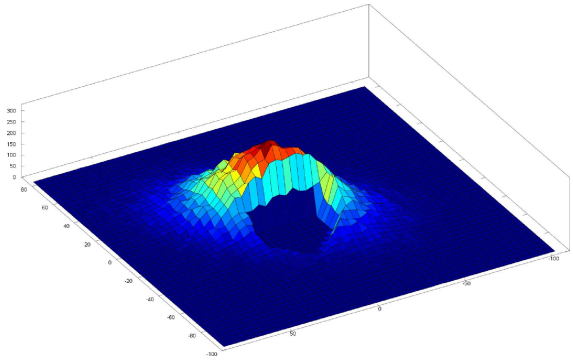
(b) Plano do espaço de configurações visto de cima.



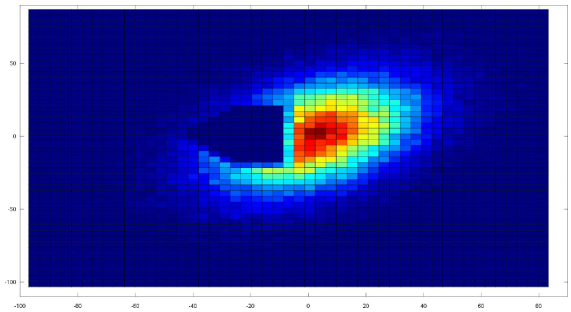
(c) Distribuição de amostras gaussianas com média nula.



(d) Distribuição gaussiana bivariada vista de cima.



(e) Representação tridimensional da distribuição resultante.



(f) Distribuição resultante vista de cima.

Figura 5.2: Exemplo de histograma da distribuição $p(\mathbf{q}_k)$.

na *distribuição final* foi gerada em $\mathcal{C}_{obs}$, pois esta não está na zona de visibilidade da configuração de referência. Outro aspecto importante pode ser melhor visto nas Figuras 5.2(d) e 5.2(f), em que se nota um deslocamento do valor médio da distribuição final para a direita. Isto acontece, especificamente para este exemplo, porque a região que está fora da visibilidade de $\mathbf{q}$ está concentrada mais à esquerda na Figura 5.2(f). Assim, o processo de reamostragem da distribuição de base valoriza a zona de visibilidade, deslocando a amostragem para a direita do espaço de configurações, e conseqüentemente deslocando o valor médio da distribuição final.

5.2 CONDIÇÕES PARA CONVERGÊNCIA DO ALGORITMO ARW EM UM ESPAÇO DISCRETIZADO

As dúvidas que surgem naturalmente ao utilizar um algoritmo estocástico de planejamento de rotas é se ele converge para a solução e quais são as condições para a convergência. No que se refere à convergência, já foi mostrado que o passeio aleatório adaptativo gaussiano converge para a solução, caso ela exista [9].

Nesta seção, será apresentada uma condição suficiente para convergência de passeios aleatórios adaptivos em um espaço euclidiano discretizado, porém cujas distribuições de base não precisam ser necessariamente gaussianas.

Assim, assume-se que $\mathcal{C} \rightarrow \mathbb{Z}^d$, em que d é o número de dimensões de $\mathcal{C}$. Define-se $\mathcal{C}_{free}$ como o conjunto de pontos do espaço de configurações mapeados em $\mathbb{Z}^d$ que estejam livres de obstáculos.

Definição 5.2.1. *Seja $P(\mathbf{x}, \mathbf{y})$ a probabilidade de sair de um ponto $\mathbf{x}$[3] arbitrário e chegar a um ponto $\mathbf{y}$ qualquer, então*

$$P(\mathbf{x}, \mathbf{y}) \geq 0, \ \forall \ \mathbf{y} \in \mathcal{C}_{free}, \tag{5.13}$$

$$P(\mathbf{x}, \mathbf{y}) = 0, \ \mathbf{y} \notin \mathcal{V}_x, \tag{5.14}$$

$$\sum_{\mathbf{y} \in \mathcal{C}_{free}} P(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{y} \in \mathcal{V}_x} P(\mathbf{x}, \mathbf{y}) = 1, \tag{5.15}$$

com $\mathcal{V}_x$ dado pela Eq. (5.5). □

$P(\mathbf{x}, \mathbf{y})$ representa uma função de transição que indica a probabilidade de sair de x e *estar* em y após um incremento. É importante notar algumas propriedades desta função de transição. Saindo de um ponto qualquer, só é possível ir para outro ponto que esteja em sua região de visibilidade, caso apenas um incremento seja dado. Além disso, a Eq. (5.15) indica que a probabilidade de sair de um ponto $\mathbf{x} \in \mathcal{C}_{free}$ e estar em um ponto $\mathbf{y}$ qualquer, em sua região de visibilidade, é 1. Isto torna-se claro ao imaginar uma partícula em um espaço com restrições e que se move aleatoriamente em $\mathcal{C}_{free}$. Independente da distribuição que caracteriza seu movimento, ela não só estará, após apenas um incremento, em algum lugar qualquer de $\mathcal{C}_{free}$, como também estará, com probabilidade 1, na região de visibilidade da sua posição anterior.

Uma formulação semelhante para a função de transição $P(\mathbf{x}, \mathbf{y})$ é feita no livro clássico de Spitzer [57] sobre passeios aleatórios. Porém, em [57] existe uma restrição de homegeneidade espacial estabelecendo $P(\mathbf{x}, \mathbf{y}) = P(\mathbf{0}, \mathbf{y} - \mathbf{x})$. Esta condição não pode ser imposta ao algoritmo de planejamento de rotas que utiliza passeios aleatórios, uma vez que as restrições

[3] Assume-se que $\mathbf{x}$ e $\mathbf{y}$ são duas configurações $\mathbf{q}_i$ e $\mathbf{q}_j$ quaisquer. Isto ajuda a simplificar a notação ao longo da Seção.

do espaço de configurações fazem com que a função densidade de probabilidade a partir de um ponto $\mathbf{x}$ seja determinada pela sua visibilidade, como já mostrado na Seção 5.1.

Neste sentido, a formulação para a função de transição deste trabalho, dada pelas Equações (5.13), (5.14) e (5.14), é uma generalização daquela encontrada em [57].

O bom senso indica que se não é possível, em apenas um incremento, sair de um ponto $\mathbf{x}$ e chegar em um ponto $\mathbf{y}$ fora de sua região de visibilidade $\mathcal{V}(\mathbf{x})$, então mais incrementos são necessários. Obviamente, uma partícula contida em $\mathcal{J} \in \mathcal{C}_{free}$ nunca poderá chegar em uma região $\mathcal{K} \in \mathcal{C}_{free}$ que não seja alcançável a partir da região de origem, ou seja, se não houver qualquer seqüência, de qualquer tamanho, que permita sair de $\mathcal{J}$ e chegar a $\mathcal{K}$, então a função de transição deve ser zero. A função de transição que representa a probabilidade de sair de $\mathbf{x}$ e *estar* em $\mathbf{y}$ após n passos é dada por $P_n(\mathbf{x}, \mathbf{y})$.

Definição 5.2.2. *Sendo* $\mathbf{x}, \mathbf{y} \in \mathcal{C}_{free}$

$$P_0(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x}, \mathbf{y}) = \begin{cases} 1, \mathbf{x} = \mathbf{y} \\ 0, \textit{caso contrário} \end{cases} \tag{5.16}$$

$$P_1(\mathbf{x}, \mathbf{y}) = P(\mathbf{x}, \mathbf{y})$$

$$P_n(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{x}_1 \in \mathcal{C}_{free}} \sum_{\mathbf{x}_2 \in \mathcal{C}_{free}} \dots \sum_{\mathbf{x}_{n-1} \in \mathcal{C}_{free}} P(\mathbf{x}, \mathbf{x}_1) \cdot P(\mathbf{x}_1, \mathbf{x}_2) \dots P(\mathbf{x}_{n-1}, \mathbf{y}),\ n \geq 2.$$

□

Em cada somatório são avaliadas todas as amostras de $\mathcal{C}_{free}$, o que indica que esta função de transição avalia a probabilidade associada a todos os caminhos possíveis para sair de $\mathbf{x}$ e estar em $\mathbf{y}$ em n passos.

Uma partícula que se desloca em $\mathcal{C}_{free}$, independente do número de incrementos dados, sempre estará em $\mathcal{C}_{free}$. A Proposição 5.2.1 mostra, de maneira mais formal, este fato.

Proposição 5.2.1. *Para* $\mathbf{x} \in \mathcal{C}_{free}$

$$\sum_{\mathbf{y} \in \mathcal{C}_{free}} P_n(\mathbf{x}, \mathbf{y}) = 1.$$

PROVA Da Definição 5.2.2, tem-se que

$$\sum_{\mathbf{y} \in \mathcal{C}_{free}} P_n(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{y} \in \mathcal{C}_{free}} \sum_{\mathbf{x}_1 \in \mathcal{C}_{free}} \dots \sum_{\mathbf{x}_{n-1} \in \mathcal{C}_{free}} P(\mathbf{x}, \mathbf{x}_1) \dots P(\mathbf{x}_{n-1}, \mathbf{y})$$

Pela Eq. (5.15), tem-se que

$$= \sum_{\mathbf{x}_1 \in \mathcal{C}_{free}} \dots \sum_{\mathbf{x}_{n-1} \in \mathcal{C}_{free}} P(\mathbf{x}, \mathbf{x}_1) \dots P(\mathbf{x}_{n-2}, \mathbf{x}_{n-1}) \cancelto{1}{\sum_{\mathbf{y} \in \mathcal{C}_{free}} P(\mathbf{x}_{n-1}, \mathbf{y})}$$

e então

$$= \sum_{\mathbf{x}_1 \in \mathcal{C}_{free}} \dots \sum_{\mathbf{x}_{n-2} \in \mathcal{C}_{free}} P(\mathbf{x}, \mathbf{x}_1) \dots P(\mathbf{x}_{n-3}, \mathbf{x}_{n-2}) \sum_{\mathbf{x}_{n-1} \in \mathcal{C}_{free}} P(\mathbf{x}_{n-2}, \mathbf{x}_{n-1}) \to 1$$

$$\vdots$$

$$= 1 \qquad \square$$

A Proposição 5.2.1 indica que a probabilidade de sair de um ponto no espaço de configurações livre e, em n passos, estar em um ponto *qualquer* de $\mathcal{C}_{free}$, é igual a 1. Ou seja, todo passeio aleatório que começa em $\mathcal{C}_{free}$ sem destino pré-determinado, após n passos claramente vai se encontrar em $\mathcal{C}_{free}$.

Como visto na Definição 5.2.2, $P_n(\mathbf{x}, \mathbf{y})$ indica a probabilidade de sair de $\mathbf{x}$ e estar em $\mathbf{y}$ em n passos. Porém, em planejamento de rotas usando passeios aleatórios, a solução é caracterizada pela primeira vez que o passeio atinge o ponto desejado. Assim, é necessário definir uma função que represente este evento.

Definição 5.2.3.

$$\begin{aligned} F_0(\mathbf{x}, \mathbf{y}) &= 0 \\ F_1(\mathbf{x}, \mathbf{y}) &= P(\mathbf{x}, \mathbf{y}) \\ F_n(\mathbf{x}, \mathbf{y}) &= \sum_{\mathbf{x}_1 \in \mathcal{C}_{free} \setminus \{\mathbf{y}\}} \dots \sum_{\mathbf{x}_{n-1} \in \mathcal{C}_{free} \setminus \{\mathbf{y}\}} P(\mathbf{x}, \mathbf{x}_1) \dots P(\mathbf{x}_{n-1}, \mathbf{y}) \end{aligned}$$

$\square$

$F_n(\mathbf{x}, \mathbf{y})$ representa a probabilidade de sair de $\mathbf{x}$ e *chegar* em $\mathbf{y}$ em n passos. A diferença crucial entre as funções $P_n(\mathbf{x}, \mathbf{y})$ e $F_n(\mathbf{x}, \mathbf{y})$ é que a primeira representa uma função de transição em n passos, enquanto que a segunda, conforme poderá ser visto na Proposição 5.2.3, representa a função de probabilidade acumulada do ARW em um espaço discretizado.

A proposição a seguir apresenta uma outra formulação de $F_n(\mathbf{x}, \mathbf{y})$ que será útil na prova da Proposição 5.2.3 e ainda na prova do Teorema 5.2.1, que estabelece a condição suficiente para convergência do passeio aleatório adaptativo.

Proposição 5.2.2. *Seja Ω_n o conjunto formado pelas seqüências ω tal que*

$$\omega = \{\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n-1}, \mathbf{x}_n\},\ \mathbf{x}_i \in \mathcal{C}_{free},\ i = 1, \dots, n. \tag{5.17}$$

e que, para cada ω em Ω_n, é associada a medida de probabilidade

$$Pr(\omega) = P(\mathbf{x}, \mathbf{x}_1) P(\mathbf{x}_1, \mathbf{x}_2) \dots P(\mathbf{x}_{n-1}, \mathbf{x}_n). \tag{5.18}$$

O conjunto Ω_n contém todas as seqüências possíveis, a partir de $\mathbf{x}$, utilizando n passos.

Define-se ainda os conjuntos A_k tais que

$$A_k = [\omega | \omega \in \Omega_n; \mathbf{x}_1 \neq \mathbf{y}, \mathbf{x}_2 \neq \mathbf{y}, \ldots, \mathbf{x}_{k-1} \neq \mathbf{y}, \mathbf{x}_k = \mathbf{y}],\ 1 \leq k \leq n. \tag{5.19}$$

Portanto, os conjuntos A_k são subconjuntos de Ω_n que contém todas as seqüências nas quais uma partícula que sai de $\mathbf{x}$, chega na posição especificada $\mathbf{y}$ em exatamente k passos. Então,

$$F_k(\mathbf{x}, \mathbf{y}) = \sum_{\omega \in A_k} Pr(\omega), 1 \leq k \leq n. \tag{5.20}$$

PROVA Como $\omega \in A_k$, então $\omega = \{\mathbf{x}, \mathbf{x}_1 \neq \mathbf{y}, \mathbf{x}_2 \neq \mathbf{y}, \ldots, \mathbf{x}_k = \mathbf{y}, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_n\},\ 1 \leq k \leq n$. Logo,

$$Pr(\omega) = P(\mathbf{x}, \mathbf{x}_1) \ldots P(\mathbf{x}_{k-1}, \mathbf{y}) P(\mathbf{y}, \mathbf{x}_{k+1}) \ldots P(\mathbf{x}_{n-1}, \mathbf{x}_n).$$

Sendo $\mathcal{C}^*_{free} = \mathcal{C}_{free} \setminus \{\mathbf{y}\}$ e usando a notação abreviada $\sum_{\substack{\mathbf{x}_i, \\ i=1,\ldots,n}} = \sum_{\mathbf{x}_1} \ldots \sum_{\mathbf{x}_n}$, tem-se que

$$\begin{aligned}
\sum_{\omega \in A_k} Pr(\omega) &= \sum_{\substack{\mathbf{x}_i \in \mathcal{C}^*_{free}, \\ i=1,\ldots,k-1}} \sum_{\substack{\mathbf{x}_j \in \mathcal{C}_{free}, \\ j=k+1,\ldots,n}} P(\mathbf{x}, \mathbf{x}_1) \ldots P(\mathbf{x}_{k-1}, \mathbf{y}) P(\mathbf{y}, \mathbf{x}_{k+1}) \ldots P(\mathbf{x}_{n-1}, \mathbf{x}_n) \\
&= \sum_{\substack{\mathbf{x}_i \in \mathcal{C}^*_{free}, \\ i=1,\ldots,k-1}} P(\mathbf{x}, \mathbf{x}_1) \ldots P(\mathbf{x}_{k-1}, \mathbf{y}) \sum_{\substack{\mathbf{x}_j \in \mathcal{C}_{free}, \\ j=k+1,\ldots,n}} P(\mathbf{y}, \mathbf{x}_{k+1}) \ldots P(\mathbf{x}_{n-1}, \mathbf{x}_n) \\
&= \sum_{\substack{\mathbf{x}_i \in \mathcal{C}^*_{free}, \\ i=1,\ldots,k-1}} P(\mathbf{x}, \mathbf{x}_1) \ldots P(\mathbf{x}_{k-1}, \mathbf{y}) \sum_{\mathbf{x}_n \in \mathcal{C}_{free}} \cancelto{1}{P_{n-k}(\mathbf{y}, \mathbf{x}_n)} \\
&= F_k(\mathbf{x}, \mathbf{y})
\end{aligned} \tag{5.21}$$

□

■

A Proposição 5.2.2 associa uma medida de probabilidade para a função $F_n(\mathbf{x}, \mathbf{y})$ que será útil na Proposição seguinte, que a caracteriza como uma função de probabilidade acumulada.

Proposição 5.2.3.

$$\sum_{k=1}^{n} F_k(\mathbf{x}, \mathbf{y}) \leq 1. \tag{5.22}$$

PROVA Seja ω, $Pr(\omega)$ e A_k dados pelas Equações (5.17), (5.18) e (5.19), respectivamente. Além disso, considera-se o fato de que se $\mathcal{C}_{free}$ é contável, então o conjunto Ω_n também é contável [57].

Então,

$$\sum_{[\omega|\omega\in\Omega_n,\mathbf{x}_n=\mathbf{y}]} Pr(\omega) = P_n(\mathbf{x},\mathbf{y}) \tag{5.23}$$

e

$$\sum_{\omega\in\Omega_n} Pr(\omega) = \sum_{\mathbf{y}\in\mathcal{C}_{free}} P_n(\mathbf{x},\mathbf{y}) = 1. \tag{5.24}$$

Sendo A_k conjuntos disjuntos de Ω_n e $(A_1 \cup A_2 \cup \ldots \cup A_n) \subset \Omega_n$, pois para A_n tem-se que $\mathbf{x}_n = \mathbf{y}$, então,

$$\sum_{k=1}^{n} \sum_{\omega\in A_k} Pr(\omega) \leq \sum_{\omega\in\Omega_n} Pr(\omega) = 1. \tag{5.25}$$

Substituindo a Eq. (5.20) em (5.25),

$$\sum_{k=1}^{n} F_k(\mathbf{x},\mathbf{y}) \leq 1. \tag{5.26}$$

□

A Eq. (5.22) pode ser interpretada como a probabilidade de sair de $\mathbf{x}$ e chegar em $\mathbf{y}$ em k passos ou menos, o que resulta no seguinte teorema:

Teorema 5.2.1. *Seja $\mathcal{C}_{free}$ conexo, ou seja, para todo par $\{\mathbf{a},\mathbf{b}\} \in \mathcal{C}_{free}$ existe uma seqüência de tamanho arbitrário $\{\mathbf{x}_0 = \mathbf{a}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n = \mathbf{b}\}$, $\mathbf{x}_i \in \mathcal{C}_{free}$, $n \in \mathbb{Z}^+$, tal que $\mathbf{x}_i \in \mathcal{V}_{\mathbf{x}_i-1}$, $i = 1\ldots, n$. Então, é dito que um passeio aleatório converge se*

$$\lim_{n\to\infty} \sum_{k=1}^{n} F_k(\mathbf{x},\mathbf{y}) = 1, \tag{5.27}$$

para $\forall \mathbf{x},\mathbf{y} \in \mathcal{C}_{free}$

PROVA Seja Ω_n o conjunto formado por todos $\omega = \{\mathbf{x}_0 = \mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Para $n \to \infty$, $\Omega_n \to \Omega$, em que Ω é o conjunto de todas as seqüências possíveis em $\mathcal{C}_{free}$. $Pr\{\omega\} = P(\mathbf{x}_0,\mathbf{x}_1)P(\mathbf{x}_1,\mathbf{x}_2)\ldots$ e

$$B_k = [\omega|\omega\in\Omega_n; \mathbf{x}_1 \neq \mathbf{y}, \mathbf{x}_2 \neq \mathbf{y}, \ldots, \mathbf{x}_{k-1} \neq \mathbf{y}, \mathbf{x}_k = \mathbf{y},\ \mathbf{x}_i \in \mathcal{V}_{\mathbf{x}_{i-1}}],\ 1 \leq k \leq n. \tag{5.28}$$

Nota-se que $B_k \subset A_k$ e $A_{k,\emptyset} = A_k - B_k$, ou seja, $A_{k,\emptyset}$ representa todas as sequências de A_k que contém pelo menos um par $\{\mathbf{x}_i, \mathbf{x}_{i+1}\}$ tal que $\mathbf{x}_{i+1} \notin \mathcal{V}_{\mathbf{x}_i}$. O que implica

$$Pr(A_{k,\emptyset}) = \sum_{\omega\in A_{k,\emptyset}} Pr(\omega) = 0. \tag{5.29}$$

Partindo da Eq. (5.27), tem-se que

$$\lim_{n\to\infty}\sum_{k=1}^{n} F_k(\mathbf{x},\mathbf{y}) = 1$$

$$\sum_{k=1}^{\infty}\sum_{\omega\in A_k} Pr(\omega) = 1$$

$$\sum_{k=1}^{\infty}\sum_{\omega\in(B_k\cup A_{k,\emptyset})} Pr(\omega) = 1.$$

Como $B_k \cap A_{k,\emptyset} = \emptyset$, então

$$\sum_{k=1}^{\infty}\left(\sum_{\omega\in B_k} Pr(\omega) + \cancelto{0}{\sum_{\omega\in A_{k,\emptyset}} Pr(\omega)}\right) = 1.$$

$$\sum_{k=1}^{\infty}\sum_{\omega\in B_k} Pr(\omega) = 1. \tag{5.30}$$

Uma vez que $\bigcup_{k=1}^{\infty} B_k$ forma a partição do conjunto das possíveis soluções e

$$Pr\left(\bigcup_{k=1}^{\infty} B_k\right) = \sum_{k=1}^{\infty}\sum_{\omega\in B_k} Pr(\omega), \tag{5.31}$$

então

$$Pr\left(\bigcup_{k=1}^{\infty} B_k\right) = 1, \tag{5.32}$$

o que resulta na convergência do passeio aleatório. □

5.3 DISTRIBUIÇÃO UNIFORME VERSUS DISTRIBUIÇÃO GAUSSIANA EM $\mathbb{R}^2$

Em [9] a distribuição de base utilizada no ARW foi a distribuição gaussiana. Contudo, conforme visto na Seção 5.1, a princípio não há uma indicação clara de qual distribuição deve ser utilizada. Sendo assim, um motivo de escolha da gaussiana seria dado talvez pela facilidade de gerar com baixo custo computacional amostras desta distribuição. Mas se o motivo fosse realmente este - facilidade de geração de amostras com baixo custo computacional - então uma outra distribuição a ser considerada seria a distribuição uniforme.

Para uma comparação justa entre as duas distribuições, os mesmos parâmetros devem ser utilizados, ou seja, média nula e matriz de covariâncias dada por $\mathbf{\Sigma_x}$. Assim, é necessário gerar uma distribuição uniforme multivariada, tal que $\mathbf{v} \sim U(\mathbf{0}, \mathbf{\Sigma_x})$. Para tanto, inicialmente gera-se um vetor $\mathbf{u}$ de elementos independentes u_x e u_y, tal que

$$\mathbf{u} = (\mathbf{0}, \mathbb{I}_2), \tag{5.33}$$

ou seja,

$$u_x, u_y \sim U(-\alpha, \alpha),$$
$$P_{\mathbf{u}} = \mathbb{I}_2,$$

em que α deve ser determinado de forma a garantir que $\sigma_u^2 = 1$.

A distribuição de $\mathbf{u}$ é dada por

$$p_{\mathbf{u}}(u) = \begin{cases} \frac{1}{2\alpha}, & -\alpha \leq u < \alpha, \\ 0, \text{ caso contrário.} \end{cases}, \; E\{u\} = 0,$$

e as variâncias de u_x e u_y por

$$\begin{aligned} \sigma_u^2 &= \int_{-\infty}^{\infty} (u - E\{u\})^2 p_{\mathbf{u}}(u) du \\ &= \int_{-\alpha}^{\alpha} u^2 \cdot \frac{1}{2\alpha} du \\ &= \frac{1}{2\alpha} \left(\frac{u^3}{3} \right) |_{-\alpha}^{\alpha} \\ &= \frac{\alpha^3}{3} \end{aligned}$$

Se $\sigma_u^2 = 1 \Longrightarrow \alpha = \sqrt{3}$ e, então, o vetor $\mathbf{u}$ é gerado de acordo com a Eq. (5.33). Em seguida é feita a decomposição de Cholesky para achar uma matriz $\mathbf{A}$ tal que $\mathbf{A}^T\mathbf{A} = \mathbf{\Sigma_x}$. Então tem-se que

$$\mathbf{v} = \mathbf{A}^T \mathbf{u}. \tag{5.34}$$

Contudo, resta saber se $\mathbf{v}$ segue uma distribuição uniforme. Uma vez que a decomposição de Cholesky é dada por

$$\begin{aligned} \mathbf{\Sigma_x} &= \mathbf{A}^T\mathbf{A} \\ \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} &= \begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} a_{11} & a_{21} \\ 0 & a_{22} \end{pmatrix} \\ &= \begin{pmatrix} a_{11}^2 & a_{11}a_{21} \\ a_{21}a_{11} & a_{21}^2 + a_{22}^2 \end{pmatrix} \\ &\Longrightarrow \mathbf{A}^T = \begin{pmatrix} \sigma_x & 0 \\ \frac{\sigma_{xy}}{\sigma_x} & \sqrt{\frac{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2}{\sigma_x^2}} \end{pmatrix} \end{aligned} \tag{5.35}$$

Substituindo 5.35 em 5.34 tem-se que

$$\mathbf{v} = \begin{pmatrix} \sigma_x u_x \\ \frac{\sigma_{xy}u_x + \sqrt{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2}u_y}{\sigma_x} \end{pmatrix} \tag{5.36}$$

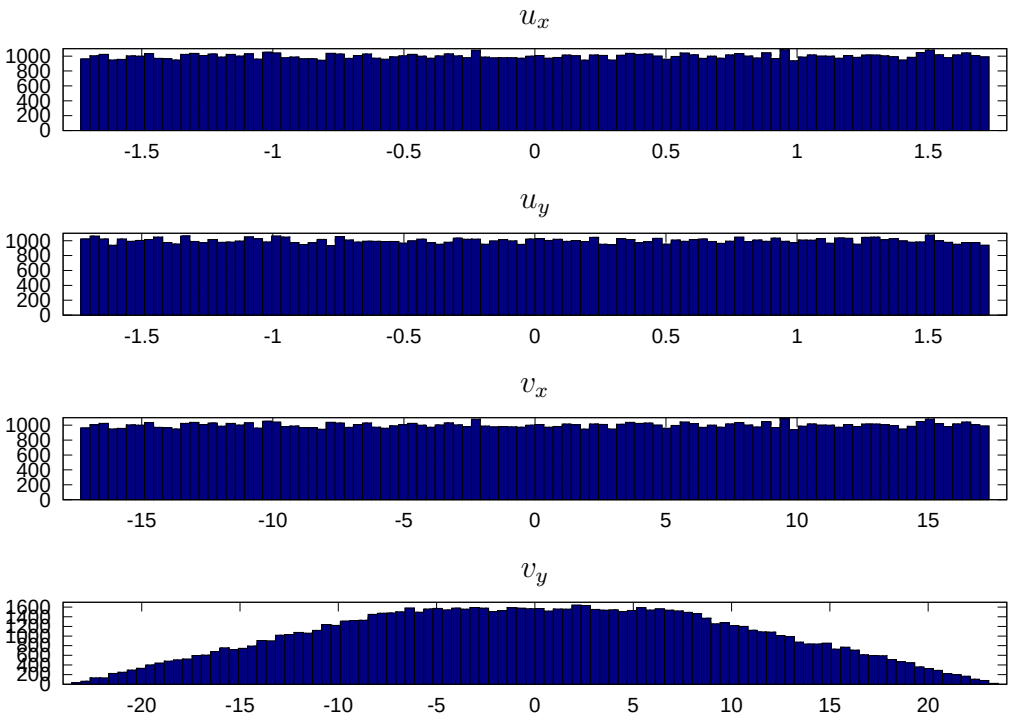


Figura 5.3: Variável aleatória $\mathbf{u}$ e sua influência na variável aleatória $\mathbf{v}$.

Pela Eq. (5.36) é possível notar que v_x segue a distribuição uniforme mas não v_y, pois o resultado da soma de duas variáveis aleatórias uniformes não é uma variável aleatória uniforme. A Figura 5.3 mostra o resultado da operação dada pela Eq. (5.34). Nota-se claramente que v_y não segue uma distribuição uniforme.

A Figura 5.4 mostra uma comparação entre as distribuições uniforme e gaussiana cujas amostras foram geradas utilizando a mesma matriz de covariâncias. A elipse 3σ também foi desenhada nas duas figuras. Observa-se que a distribuição gaussiana, apesar de concentrar as amostras geradas no centro da distribuição, ainda assim consegue gerar algumas amostras mais longe de seu centro do que a distribuição uniforme. Além disso, a distribuição uniforme é um paralelogramo, e não um retângulo rotacionado. Esta distorção, em conjunto com o menor alcance das amostras geradas, pode implicar em uma amostragem menos eficiente por parte da distribuição uniforme. No entanto, a adaptação da matriz de covariâncias, em função da região onde o passeio se encontra, não é exata. Pelo contrário, é uma aproximação dada pela história do processo, de forma que uma afirmação conclusiva sobre qual distribuição é mais eficiente só é possível por meio de ensaios experimentais. Estes resultados são apresentados no Capítulo 6.

A distorção da distribuição "uniforme", além ser explicada pela Eq. (5.36), também pode ser explicada pelo fato do vetor $\mathbf{u}$ ser multiplicado por uma matriz que não pertence ao grupo $SO(2)$, sendo assim a multiplicação $\mathbf{A}^T\mathbf{u}$ não representa uma rotação.

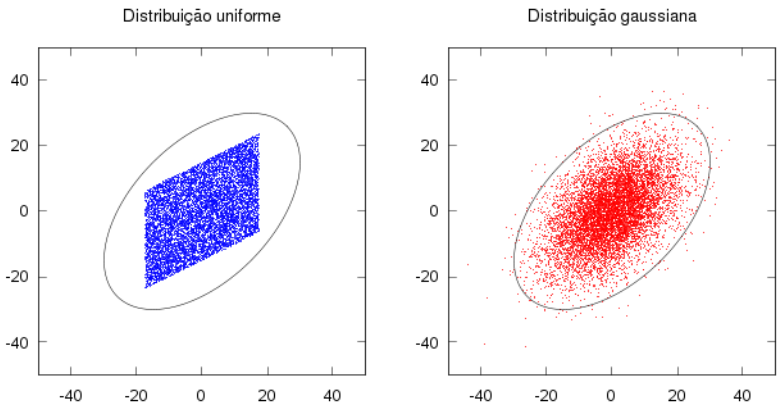
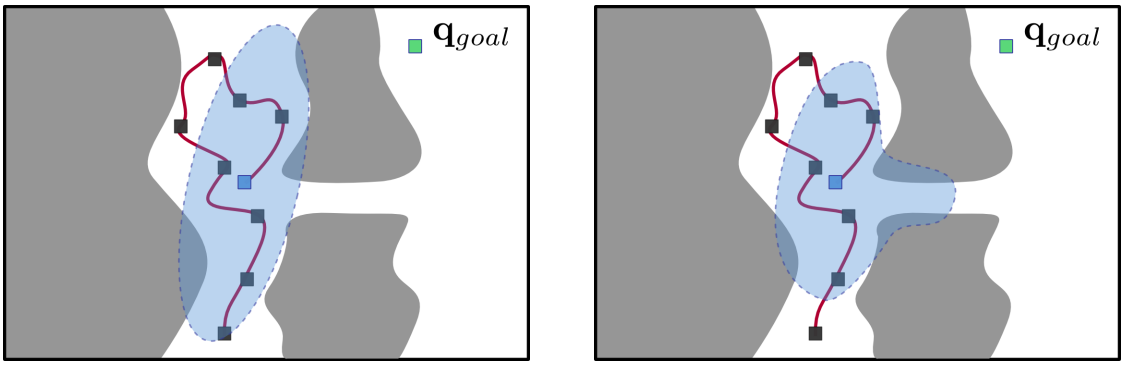


Figura 5.4: Distribuições uniforme (esquerda) e gaussiana (direita) bivariadas com os mesmos parâmetros de entrada.

5.4 SELEÇÃO DE CANDIDATOS

Como visto na Seção 5.1, a distribuição do ARW é dada basicamente por uma distribuição de base, no caso a distribuição gaussiana, multiplicada por uma função de pertinência. Esta função de pertinência implica na rejeição de amostras que não estejam na zona da última configuração gerada pelo passeio. Assim, a adaptabilidade da matriz de covariâncias, além de minimizar o número de amostras geradas em $\mathcal{C}_{obs}$, favorece a amostragem na região de visibilidade do passeio. Por outro lado, esta adaptação pode dificultar a amostragem em algumas passagens estreitas adjacentes a regiões de grande visibilidade. A Figura 5.5(a) ilustra este fato. A elipse 3σ da distribuição gaussiana concentra a geração de amostras na grande região de visibilidade no meio do mapa, mas não na pequena passagem estreita que conecta esta região àquela onde encontra-se $\mathbf{q}_{goal}$. Assim, visando aumentar o desempenho do ARW, nesta dissertação é proposta uma estratégia de polarização da amostragem rumo a regiões de $\mathcal{C}_{free}$ que são menos exploradas. A idéia é que a distribuição de base fique mais parecida com a da Figura 5.5(b), ou seja, que áreas inexploradas sejam favorecidas pela distribuição final. Uma maneira de se fazer isto é por meio da geração de várias amostras candidatas e seleção daquela que for gerada na região considerada menos explorada. Para tanto, é necessário definir uma métrica de explorabilidade.

Pode-se definir uma métrica de explorabilidade por meio da imposição de uma grade no espaço de configurações. Seja $\mathcal{C}$ um espaço de configurações de d dimensões de tal forma que uma grade $\mathcal{M}$ d-dimensional fique alinhada com os eixos de $\mathcal{C}$. Então, as coordenadas de uma célula da grade é dada por $\mathbf{x} = (x_1, \ldots, x_d)$ e o número de amostras na célula é dado por $f(\mathbf{x}) = f(x_1, \ldots, x_d)$. Assim, uma célula $\mathbf{x}_1$ é considerada mais explorada que a célula $\mathbf{x}_2$ se $f(\mathbf{x}_1) > f(\mathbf{x}_2)$.



(a) Elipse 3σ da distribuição gaussiana de base (b) Distribuição de base polarizada

Figura 5.5: Efeito da polarização das amostras rumo a regiões pouco exploradas.

A idéia da seleção de candidatos utilizando grades, no contexto do algoritmo ARW gaussiano, consiste em uma pequena extensão do algoritmo. Ao invés de gerar apenas uma amostra em $\mathcal{C}_{free}$ e verificar se ela está na região de visibilidade da última configuração do passeio, são geradas várias amostras em $\mathcal{C}_{free}$ e aquela que estiver na célula menos explorada (e concomitantemente na região de visibilidade do passeio), é aceita.

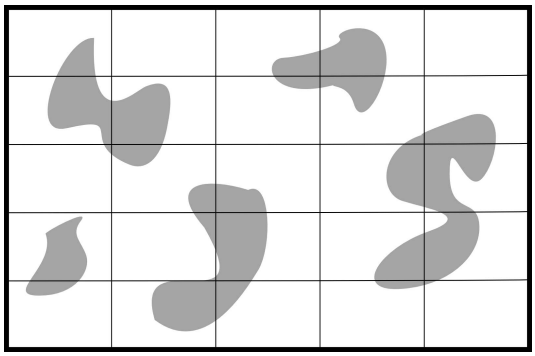
A Figura 5.6 mostra a utilização de uma grade em $\mathcal{C} \in \mathbb{R}^2$ para avaliar o quanto uma região foi explorada visando a polarização da distribuição de base (e conseqüentemente a distribuição final). A Figura 5.6(a) mostra a grade no espaço de configurações, enquanto a Figura 5.6(b) mostra o passeio aleatório com sua respectiva elipse 3σ. A Figura 5.6(c) mostra o número de amostras dentro de cada célula por onde o passeio já passou, assim como o número de amostras das células que estão sob a elipse 3σ. A probabilidade de gerar uma amostra na região hachurada aumenta na medida em que várias amostras candidatas são geradas. Uma vez que a candidata escolhida é aquela que estiver em uma região menos explorada, então a distribuição é polarizada rumo à região hachurada, resultando na distribuição da Figura 5.6(d).

5.4.1 Relação entre o volume da célula e a dimensionalidade de $\mathcal{C}$

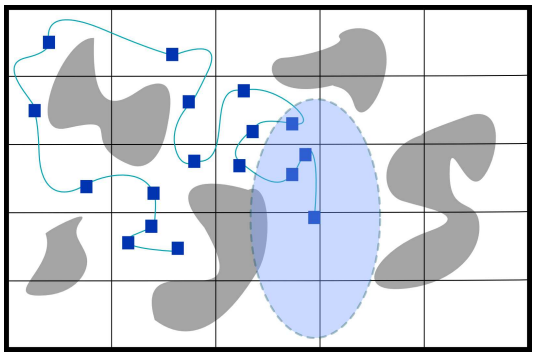
Seja a grade $\mathcal{M} \in \mathbb{Z}^d$, em que d é a dimensão do espaço de configurações. Assim, a grade pode ser decomposta pelo produto cartesiano $\mathcal{M} = \mathcal{M}_1 \times \ldots \times \mathcal{M}_d$, $\mathcal{M}_i \in \mathbb{Z},\ i = 1, \ldots, d$. Sendo a cardinalidade $\mid \mathcal{S} \mid_c$ de um conjunto discreto $\mathcal{S}$ o seu número de elementos, faz-se $\mid \mathcal{M}_i \mid_c = s$ e $s \geq 2$, em que s é o número de células em $\mathcal{M}_i$. Assim, o volume de uma célula $m \in \mathcal{M}$ é dado por

$$\mu(m) = \frac{\mu(\mathcal{C})}{s^d}. \tag{5.37}$$

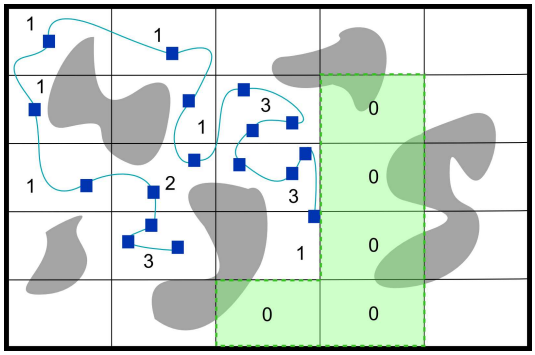
Isto significa que o volume de uma célula diminui exponencialmente com o aumento da dimensão d do espaço de configurações. Dessa forma, espera-se que esta métrica não funcione muito bem para dimensões muito altas. A Figura 5.7 mostra a relação entre a dimensão do



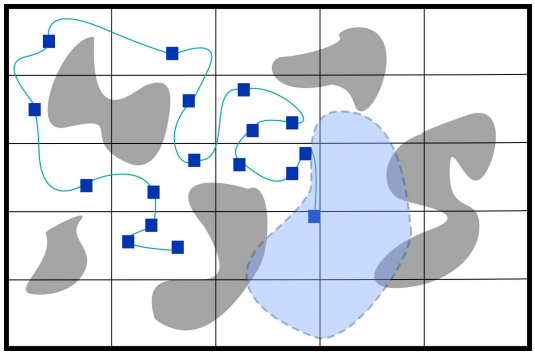
(a) Grade no espaço de configurações.



(b) Passeio aleatório e a elipse 3σ da distribuição gaussiana.



(c) Valor de $f(\mathbf{x})$ para cada célula e a região menos explorada hachurada.



(d) Distribuição de base polarizada.

Figura 5.6: Utilização de grades como métrica de explorabilidade para polarização da distribuição de base do ARW.

espaço de configurações e do volume de cada célula. Nota-se que, *independente* da cardinalidade da grade, o volume de cada célula diminui exponencialmente com o aumento da dimensão do espaço de configurações.

Exemplo 5.4.1. *Ainda analisando a grade da Figura 5.6(a), nota-se que* $s = 5$ *e* $d = 2$*. Fazendo com que o volume do espaço de configurações seja unitário, tem-se que* $\mu(m) = {}^{1}/_{25}$. □

O fato do volume de uma célula diminuir exponencialmente com o aumento da dimensão de $\mathcal{C}$ mostra que, para dimensões muito altas, *todos* os planejadores tendem a sub-amostrar o espaço de configurações.

5.5 PASSEIOS ALEATÓRIOS ADAPTATIVOS INCREMENTAIS

Uma grande vantagem dos algoritmos de múltiplo questionamento como o PRM, por exemplo, é que uma vez que eles amostraram suficientemente $\mathcal{C}_{free}$, as requisições de rotas são solucionadas muito rapidamente. Isto acontece basicamente porque, independentemente

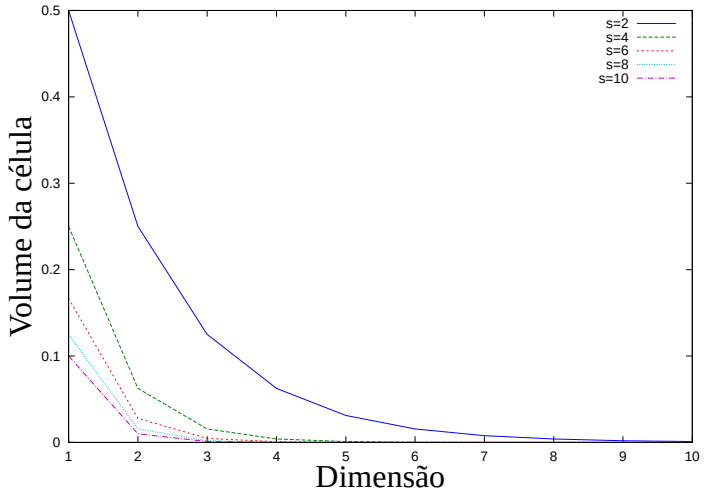


Figura 5.7: Relação entre o volume da célula e a dimensão de $\mathcal{C}$.

da dimensão do espaço de configurações, o grafo é uma representação discreta e que não depende da dimensão ou topologia de $\mathcal{C}$. Sendo assim, um grafo pode ser considerado como uma "compactação" do espaço de busca.

Por exemplo, em um mapa de rotas dois vértices que estão conectados por uma borda indicam que a configuração cujas coordenadas estão armazenadas no primeiro vértice pode ser ligada à configuração cujas informações estão no outro vértice, não importando se o espaço de configurações representa um robô móvel que translaciona no plano ou uma proteína com 10000 graus de liberdade.

Dessa forma, um algoritmo que faz as buscas no espaço de configurações como um algoritmo de questionamento único, mas que também usa um mapa de rotas para compactar e armazenar soluções que sejam relevantes e que possam ser usadas posteriormente, pode ter um bom desempenho computacional quando comparado tanto com algoritmos de questionamento único ou múltiplo questionamento.

O ARW possui a característica de boa explorabilidade, porém as rotas geradas não são armazenadas para uso posterior. No caso de múltiplas requisições de rotas, mesmo que idênticas, o algoritmo tem que gerar toda a rota novamente, implicando em desperdício de recursos computacionais. Nesse sentido, propõe-se uma forma incremental para o ARW, mostrada no Algoritmo 15. Este algoritmo, chamado iARW (do inglês, *Incremental Adaptive Random Walks*), utiliza o ARW gaussiano com seleção de amostras para exploração do espaço de configurações.

A idéia central desse algoritmo é aproveitar ao máximo as rotas que foram geradas em questionamentos anteriores. Sendo assim, inicialmente o algoritmo se comporta como o ARW e, na medida em que o espaço de configurações vai sendo muito explorado, o algoritmo passa a se comportar como o algoritmo PRM na fase de questionamento. Dessa forma, apenas as porções relevantes de C_{free} são armazenadas no mapa de rotas, diminuindo em

muito o número de amostras a serem armazenadas. De fato, na Linha 2 os dois passeios aleatórios tentam se conectar um ao outro, de tal forma que a última configuração de cada passeio tenta se conectar à configuração de destino ou então entre si. Caso haja a fusão, então uma solução foi encontrada e ela é armazenada no mapa de rotas. É conveniente processar a rota com o algoritmo Dividir para Conquistar da Seção 4.2.1 antes de armazená-la para reduzir o número de nós no mapa de rotas. Alternativamente, os nós poderiam ser armazenados sem que houvesse este processamento, aumentando a probabilidade de fusão em questionamentos subseqüentes. O único cuidado a ser tomado é com o número de nós no mapa de rotas. Se este número for muito grande, o desempenho final do algoritmo pode ser prejudicado em questionamentos posteriores. Além disso, devido ao fato do algoritmo ARW gerar rotas redundantes, com muitos ciclos e ziguezagues, algumas configurações podem não contribuir para o aumento da visibilidade do mapa de rotas por já existirem outras configurações na mesma região, caso não haja suavização das rotas armazenadas. Na Linha 5 é verificado para cada passeio quais são as configurações no mapa de rotas que estão na região de visibilidade de sua última configuração. Dessa forma, a última amostra de cada passeio é ligada a todos os *novos componentes* que estejam em sua região de visibilidade.

Em seguida é verificado se existe algum componente em comum no qual os dois passeios aleatórios conseguiram se conectar. Caso exista, uma busca é feita no mapa de rotas visando achar uma rota que ligue $\mathbf{q}'_{start}$ a $\mathbf{q}'_{goal}$, em que $\mathbf{q}'_{start}$ e $\mathbf{q}'_{goal}$ são as configurações do grafo que foram ligadas aos passeios inicial e final, respectivamente. Sendo assim, uma solução foi encontrada e é dada pela concatenação dos nós que ligam $\mathbf{q}_{start}$ a $\mathbf{q}'_{start}$, $\mathbf{q}'_{start}$ a $\mathbf{q}'_{goal}$ e $\mathbf{q}'_{goal}$ a $\mathbf{q}_{goal}$.

Caso a solução ainda não tenha sido descoberta, a evolução de cada processo é feita por meio da geração de novas amostras gaussianas com seleção de candidatos.

O Algoritmo 16 mostra em detalhes a busca por novos componentes. Na Linha 4, uma vez que existe um novo componente na zona de visibilidade de um dado passeio, o trecho do passeio aleatório que ainda não foi armazenado é suavizado pelo método Dividir e Conquistar e então armazenado no mapa de rotas. A configuração de entrada do mapa de rotas, isto é, a última configuração do passeio aleatório que foi gerada até o momento, é armazenada. Isto porque, caso este passeio aleatório seja conectado a outro componente do mapa de rotas, somente a porção da rota referente a $\mathbf{q}_{entrance}$ até a última amostra do passeio é suavizada e armazenada.

Pelos Algoritmos 15 e 16 é possível notar que o objetivo é fazer com que, além de buscar a solução da requisição de rota em questão, cada passeio se ligue ao maior número de componentes do mapa de rotas. Com isso, o mapa de rotas é expandido rapidamente e rotas que foram requisitadas anteriormente podem ser utilizadas de maneira eficiente.

A Figura 5.8 mostra um exemplo com a representação da evolução de dois passeios aleatórios e a suas conexões nos mapas de rotas, visando tanto a expansão do mesmo quanto o

Entrada: Configurações inicial e final e o mapa de rotas.

Saída : Lista com as configurações que compõem a rota resultante.

```
1  for at most k configurations do
2      if merge_ARW(ARW_start,ARW_end) then
3          store(path) and return path
4      find_new_component(ARW_start,roadmap,components_start);
5      find_new_component(ARW_end,roadmap,components_end);
6      if exist_common(components_start,components_end) then
7          return path
8      else
9          generate_new_configuration(ARW_start);
10         generate_new_configuration(ARW_end);
```

Algoritmo 15: iARW

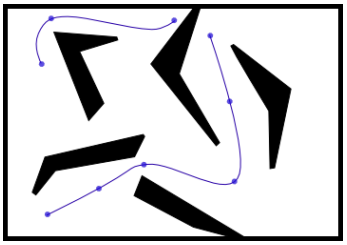
seu aproveitamento para a solução final do planejamento da rota requisitada. A Figura 5.8(a) mostra um mapa de rotas já com dois componentes armazenados. Estes foram criados em questionamentos prévios utilizando o algoritmo ARW gaussiano com seleção. Já a Figura 5.8(b) mostra a evolução de dois passeios que exploram o ambiente visando a resolução de uma nova requisição de rota. Nota-se que eles não necessariamente possuem a mesma quantidade de nós, uma vez que possuem execuções independentes. A visibilidade dos passeios é mostrada na Figura 5.8(c). É importante notar que a visibilidade de um passeio é definida somente para a sua última configuração, pois o custo de fazer a tentativa de fusão com todas as configurações do passeio pode se tornar impraticável caso o número de nós intermediários seja muito grande [35]. Na Figura 5.8(d) os passeios se conectam aos componentes que estão nas suas respectivas regiões de visibilidade. A conexão com cada componente é feita através de uma única configuração (a primeira encontrada ou, alternativamente, a mais próxima), que esteja na região de visibilidade do passeio. Além disso, uma suavização prévia com o algoritmo Dividir para Conquistar é feita antes de cada rota ser armazenada no mapa de rotas. Como os passeios não estão conectados em um componente comum, então novas amostras são geradas visando a continuidade da exploração do ambiente. Esta evolução pode ser vista na Figura 5.8(e). Observa-se que o passeio que se originou à esquerda do mapa se conecta em um componente no qual o outro passeio já estava conectado. Logo, uma solução foi encontrada e apenas os trechos relevantes da rota são armazenados. Por último, a Figura 5.8(f) mostra, em hachurado, a rota resultante.

Entrada: Configuração $\mathbf{q}$, mapa de rotas e lista dos componentes conectados a $\mathbf{q}$.

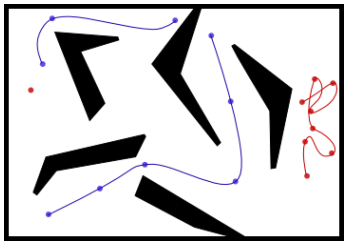
Saída : 1 se tiver se conectado ao mapa de rotas. 0 caso contrário.

```
1 find ← 0;
2 forall nodes q_i on roadmap do
3     if not same component and straight_line_local_planner(q,q_i) then
4         if not find then
5             entrance_node ← store_partial_path();
6             find ← 1;
7         update_list(components_list);
8         add_edge(entrance_node,q_i);
9 return find
```

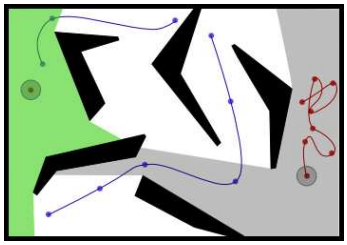
Algoritmo 16: Busca um novo componente no mapa de rotas.



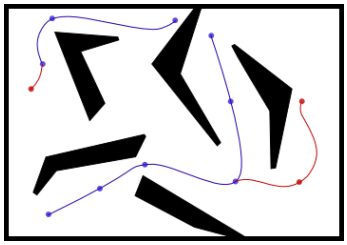
(a) Mapa de rotas previamente armazenado.



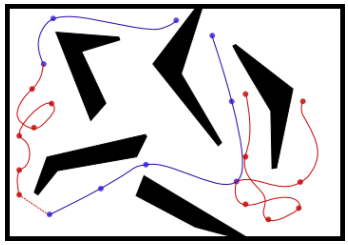
(b) Dois passeios aleatórios são iniciados à esquerda e à direita.



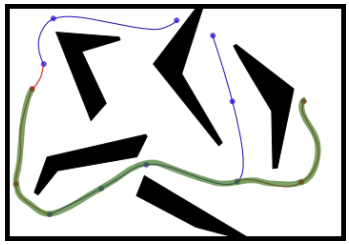
(c) Visibilidade do último nó de cada passeio.



(d) Os passeios são conectados aos componentes do mapa de rotas que estavam dentro dos respectivos campos de visão. Os nós intermediários desnecessários foram removidos.



(e) A busca ocorre até que um passeio se conecte ao outro ou que os dois se conectem ao mesmo componente do mapa de rotas.



(f) A rota resultante é dada pelo trecho hachurado. Nota-se que os trechos relevantes de cada passeio foram adicionados ao mapa de rotas.

Figura 5.8: Evolução do iARW.

6 AVALIAÇÃO DOS PLANEJADORES DE ROTAS

However beautiful the strategy,
you should occasionally look at the results.
Winston Churchill

Neste capítulo é feita uma avaliação quantitativa dos planejadores apresentados nos Capítulos 3 e 5 e dos métodos de suavização apresentados no Capítulo 4. Para todos os ensaios $\mathcal{W} \in \mathbb{R}^2$ e $\mathcal{C} \in \mathbb{R}^2$. Os testes são divididos entre os algoritmos de questionamento único e os algoritmos de múltiplo questionamento. Para cada algoritmo de questionamento único é avaliado seu desempenho em função dos seus parâmetros. Primeiramente, dada a geração de um número fixo de amostras, são avaliadas a explorabilidade e o tempo de execução correspondente. O objetivo deste primeiro teste é avaliar qual planejador consegue melhor explorar o espaço de configurações em função do número de amostras geradas. É importante verificar o tempo de execução, pois um planejador pode realizar uma maior exploração de $\mathcal{C}_{free}$ do que os outros, mas em um tempo de execução bem maior. Assim, neste primeiro teste é dado apenas o ponto de origem para o planejador, sendo que as amostras são geradas para explorar o ambiente, sem um destino pré-determinado.

No segundo teste são avaliados o tempo médio para, dados dois pontos no espaço de configurações, achar uma rota conectando-os. A diferença crucial deste teste com o primeiro é que, mesmo que um planejador seja um explorador menos eficiente, se ele tiver uma grande eficiência computacional em termos de baixo tempo de execução, então ele pode encontrar a solução mais rapidamente. Contudo, o número de amostras geradas para alcançar a solução pode ser muito maior. Além disso, neste segundo teste é realmente feita a avaliação do planejador, não a sua capacidade exploratória.

Como visto na Seção 5.4, a explorabilidade é mensurada por meio de uma grade imposta no espaço de configurações. Assim, nas simulações desta seção, cada célula possui um volume que corresponde a 1% do volume total do espaço de configurações, ou seja, $\mu(m_i) = 0,01\mu(\mathcal{C})$.

Desta forma, a explorabilidade global do espaço de configurações, chamada ϵ, é dada por

$$\epsilon = \sum_{i=0}^{99} I(i), \quad I(i) = \begin{cases} \frac{1}{100} & , n_i > 0, \\ 0 & , \text{caso contrário} \end{cases} \tag{6.1}$$

em que n_i é o número de amostras na célula i. É importante notar que a explorabilidade dada pela Eq. (6.1) refere-se à exploração global de $\mathcal{C}$, não $\mathcal{C}_{free}$. Assim, mesmo que $\mathcal{C}_{free}$ seja completamente explorado, ϵ pode nunca corresponder a 100% [1]. Esta primeira etapa

[1] Mesmo que $\mathcal{C}_{free} < \mathcal{C}$, ou seja, $\mathcal{C} \setminus \mathcal{C}_{free} \neq \emptyset$, pode acontecer de ϵ ser igual a 1, pois as grades constituem uma aproximação bastante imprecisa do espaço de configurações.

é utilizada para ajustar os parâmetros de cada algoritmo. Os espaços de trabalho utilizados nesta primeira etapa são mostrados na Figura A.1 e descritos no Apêndice A.

Depois do ajuste dos parâmetros dos algoritmos de questionamento único eles são comparados entre si. Os critérios utilizados para a comparação são o tempo de execução e a distância final percorrida. Para efeitos de comparação, o planejador por frente de onda apresentado na Seção 3.1 é utilizado, uma vez que ele é determinístico, ótimo em relação à distância da rota gerada (de acordo com a distância de Manhatan) e seu desempenho independe do número de obstáculos no espaço de trabalho, dependendo somente da distância dos pontos inicial e final.

No planejador por frente de onda, dado um nó qualquer, seus vizinhos estão a uma distância igual à largura da célula e o número de vizinhos deste nó é igual a 2^d, sendo d o número de dimensões do espaço de configurações. Sendo assim, como os outros planejadores avaliados utilizam a distância euclidiana em suas etapas de suavização, pode ocorrer da distância final desses algoritmos ser menor que a distância da rota gerada pelo algoritmo por frente de onda. Desta forma, as etapas de suavização também são aplicadas neste algoritmo. Um outro motivo para a necessidade de suavização das rotas geradas pelo planejador por frente de onda é que elas são geradas muito próximas aos obstáculos, sendo essencial a etapa de afastamento dos nós em relação aos obstáculos.

Os testes foram realizados em um computador com arquitetura Pentium 4, *clock* de 3,06GHz, com 1GB de memória RAM.

6.1 EXPLORABILIDADE DOS ALGORITMOS DE QUESTIONAMENTO ÚNICO

6.1.1 Algoritmo ARW e suas extensões

6.1.1.1 Influência do parâmetro H

O primeiro parâmetro a ser avaliado no algoritmo ARW é a história H do processo. Este parâmetro está intimamente ligado com a variância do processo por meio da Eq. (3.16). Quanto maior o valor de H, maior será o número de nós levados em consideração para o cálculo da matriz de covariâncias, aumentando o seu traço. Quanto maior o traço da matriz de covariâncias para um dado instante k, maior (probabilisticamente) será o "salto" da amostra $\mathbf{q}_{k+1} \sim N(\mathbf{q}_k, \mathbf{\Sigma_k})$ em relação à amostra $\mathbf{q}_k$. Contudo, na Seção 5.1 foi mostrado que, para uma dada amostra $\mathbf{q}_k$, uma amostra $\mathbf{q}_{k+1}$ é gerada de acordo com uma distribuição que *depende da visibilidade* de $\mathbf{q}_k$. Isto implica que a variância da distribuição de base só tem influência *dentro* da região de visibilidade. Desta forma, espera-se que a influência do parâmetro H seja saturada em um determinado valor que depende dos parâmetros de visibilidade do espaço de configurações no qual o passeio aleatório esteja inserido, assumindo que este espaço seja limitado (o que implica em uma visibilidade limitada).

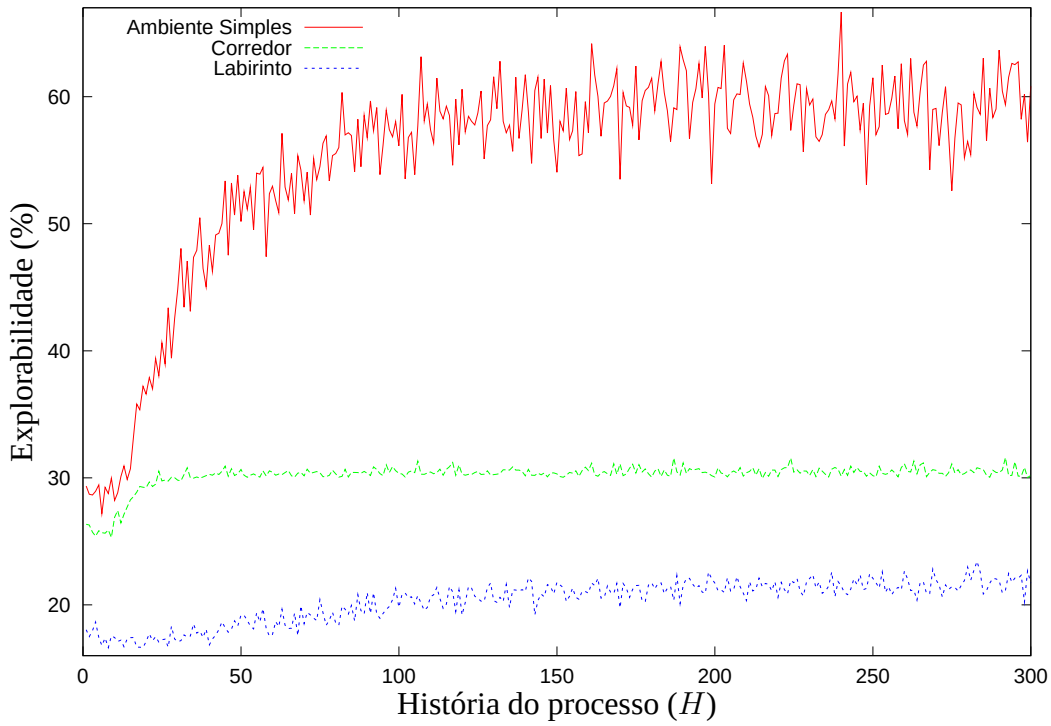


Figura 6.1: Influência da história do processo na explorabilidade (100 execuções com 500 amostras geradas em cada uma).

A Figura 6.1 mostra o resultado da explorabilidade para o algoritmo ARW padrão em função do parâmetro H nos ambientes mostrados na Figura A.1. Para cada valor de H ocorrem 100 execuções do algoritmo, com 500 amostras geradas a cada execução. Assim, a curva corresponde ao valor médio dentre essas 100 execuções. Estas curvas mostram uma evidência experimental da saturação do parâmetro H, pois para os três ambientes, a explorabilidade não aumentou a partir de um determinado valor de H para 500 amostras geradas em $\mathcal{C}_{free}$. De fato, tanto para o Ambiente Simples quanto para o Labirinto, as respectivas explorabilidades não aumentaram para valores de H maiores que 100. Para o Corredor, com H maior que 25, a explorabilidade do espaço de configurações ficou saturada em 30%. Isso porque, amostrar dentro do corredor é difícil sem que haja uma polarização da amostragem, de forma que, se um passeio inicia-se em um dos ambientes grandes, passar para o outro ambiente só é possível com um número de amostras muito grande.

6.1.1.2 Influência da seleção de candidatas

Para um valor fixo de H, a polarização da distribuição por meio da seleção de candidatas rumo a regiões menos exploradas, descrita na Seção 5.4, tende a aumentar a explorabilidade do algoritmo ARW.

A Figura 6.2 mostra a explorabilidade do algoritmo ARW gaussiano com seleção de candidatas. As curvas correspondem ao valor médio entre 100 execuções, sendo que a cada

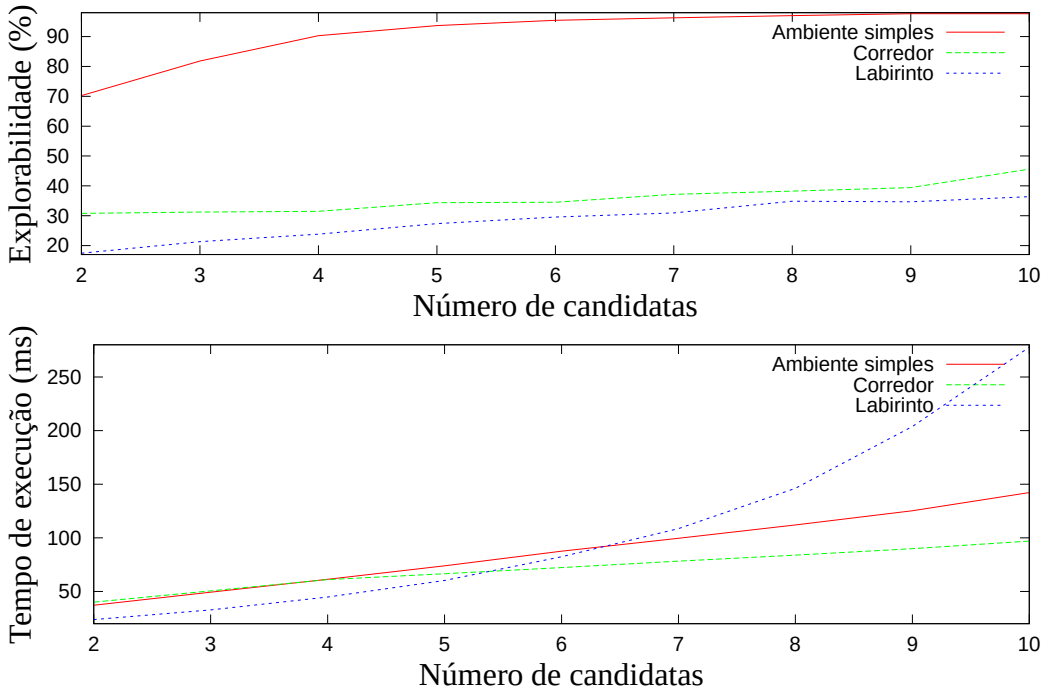


Figura 6.2: Influência das candidatas na explorabilidade do ARW. Foram realizadas 100 execuções, com 500 amostras geradas em cada uma delas e $H = 50$.

execução 500 amostras são geradas. O valor do parâmetro H foi igual a 50. Esta figura mostra, ainda, o tempo necessário para gerar estas 500 amostras.

Comparando a Figura 6.2 com a Figura 6.1 é possível notar que, já para duas candidatas e $H = 50$, a explorabilidade do Ambiente Simples passou de 50% para 70%. Para cinco candidatas, as explorabilidades do Ambiente Simples, Corredor e Labirinto passaram de aproximandamente 50%, 30% e 18% para 90%, 35%, 25%, respectivamente.

Porém, a Figura 6.2 também mostra que o tempo de amostragem com seleção de candidatas cresce numa ordem maior que linear. Em alguns ambientes, ao utilizar mais de 10 candidatas, este tempo pode crescer de tal maneira a inviabilizar o uso do algoritmo. Isto se deve ao fato que a seleção entre muitas candidatas polariza fortemente a amostragem rumo a regiões inexploradas. Esta polarização pode fazer com que o passeio aleatório fique preso em alguns mínimos locais criados por esse potencial atrativo gerado pelas zonas menos exploradas. Porém, na medida em que muitas amostras são geradas nessas regiões, suas influências na polarização da amostragem diminuem e então o passeio aleatório consegue sair destes mínimos. Este tempo em que ele fica preso é o suficiente para inviabilizar a utilização de um número muito grande de candidatas. Contudo, pode-se tirar vantagem deste algoritmo para amostrar regiões muito difíceis, nas quais apenas uma grande polarização da amostragem torne viável a exploração com um número reduzido de amostras. Por outro lado, verificar de forma robusta se uma região é de difícil acesso (e definir quais são essas regiões) e estimar o número de candidatas necessárias de forma a maximizar o desempenho não é uma tarefa

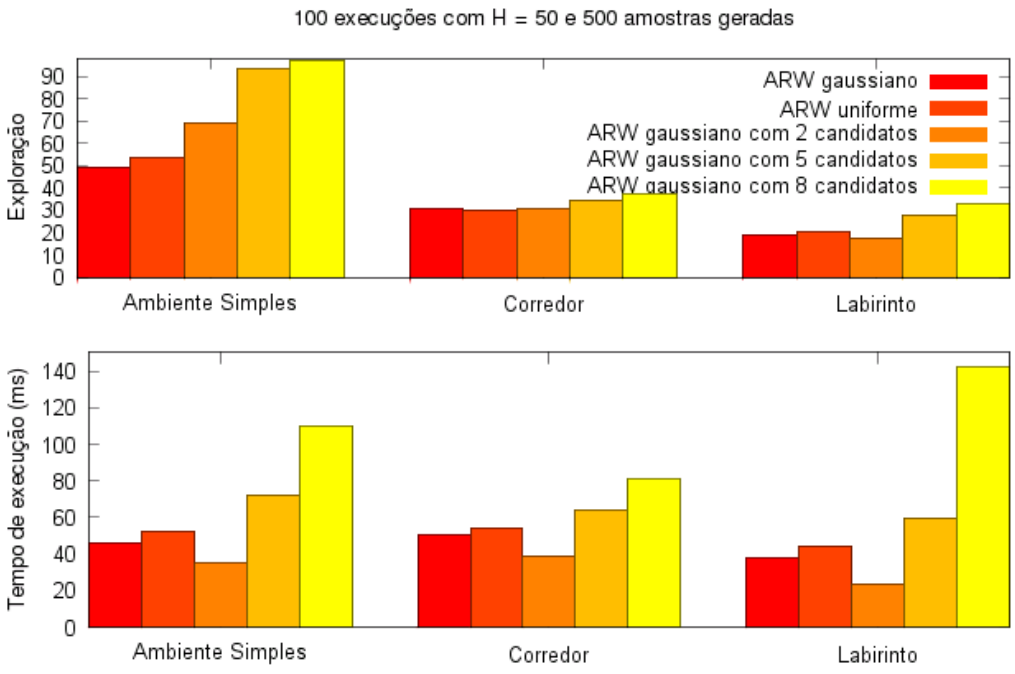


Figura 6.3: Avaliação da influência da distribuição de probabilidade no ARW.

trivial.

6.1.1.3 Influência da função densidade de probabilidade no desempenho do ARW

A Figura 6.3 mostra a influência da distribuição na explorabilidade média do ARW. Foram feitas 100 execuções, sendo que em cada uma delas foram geradas 500 amostras, e então o resultado mostrado corresponde à média entre estas execuções. Além disso, o valor de H foi igual a 50. O tempo de execução na figura refere-se ao tempo médio necessário para gerar as 500 amostras.

Nesta avaliação os resultados continuaram coerentes com os resultados da Figura 6.2, ou seja, quanto maior o número de amostras candidatas, maior é a explorabilidade do algoritmo. Além disso, é feita a comparação com o ARW padrão que utiliza como base a distribuição gaussiana e uma versão com a distribuição uniforme como distribuição de base.

A polarização da distribuição ocasionou um custo elevado de tempo de execução quando o número de candidatas passou de dois para oito. Um fato notável é que o uso de duas candidatas, além de em geral ter provocado um aumento na explorabilidade, em todos os ambientes obteve o menor tempo de execução dentre todas as distribuições.

Mais interessante ainda é o fato de que com duas candidatas, o tempo de execução foi menor que utilizando apenas uma, que é o caso do ARW gaussiano. Inicialmente, seria de esperar que o tempo de execução fosse aproximadamente o dobro, pois o número de amostras geradas é o dobro no caso da seleção entre duas candidatas. Contudo, como a amostragem

é polarizada rumo a regiões livres que foram menos exploradas, o algoritmo com seleção passa a ter um melhor desempenho, pois menos amostras são descartadas.

6.1.2 Algoritmo EST

Para o algoritmo EST, descrito na Seção 3.4, foram avaliadas as influências do parâmetro da função de massa na explorabilidade e no tempo de execução e ainda a influência da largura da janela de amostragem a partir de uma configuração de referência.

A função de massa utilizada foi $\pi(q) = \frac{1}{(n_v+1)^x}$, com $x = 1, \ldots, 5$ e n_v é o número de vizinhas da configuração q. Já a largura da janela de amostragem foi de 30 a 120 unidades e a distribuição utilizada para gerar configurações a partir de q foi a uniforme.

Os resultados da avaliação estão resumidos nas Figuras 6.4, 6.5 e 6.6. Foram realizadas 100 execuções, sendo que para cada uma foram geradas 500 amostras, de forma a avaliar a explorabilidade e o tempo médio de execução para gerar as 500 amostras. Em todos os ambientes a explorabilidade foi aumentada com a polarização das amostras rumo a regiões menos exploradas. Contudo, nota-se que uma grande polarização pode ocasionar um grande tempo de execução, pois o algoritmo pode ficar preso durante algum tempo em alguns mínimos locais. Isto pode ser notado por meio dos picos no tempo de execução quando $x = 5$.

A largura da janela de amostragem também influenciou tanto na exploração quanto no tempo para geração das 500 amostras. Uma maior janela de amostragem faz com que a exploração seja mais intensa. Porém, a amostragem em regiões de maior abrangência aumenta a chance da amostra ser gerada fora da região de visibilidade da configuração de referência, de forma que muitas amostras são descartadas antes da geração de uma amostra válida e então o tempo de execução pode aumentar bastante.

6.1.3 Algoritmo RRT

A Figura 6.7 mostra o desempenho do RRT-connect nos três ambientes de teste da Figura A.1 para 100 amostras geradas. Para este teste, foram realizadas 100 execuções e então avaliadas as explorabilidade e tempo para geração das amostras. Nota-se que, como não existe ajuste de parâmetro, os valores nas abscissas correspondem a cada uma das execuções do algoritmo (diferentemente dos outros algoritmos, cujas curvas eram obtidas por meio da média entre as várias execuções). Nota-se que tanto o tempo de execução quanto a exploração do ambiente seguiram uma mesma tendência para praticamente todas as execuções. Existe um pico no começo do gráfico que mostra o tempo de execução do RRT-connect. Este pico pode ser explicado por alguma interferência, a nível de sistema operacional, de outro processo concorrente ao processo do planejador de rotas. De fato, nos sistemas operacionais modernos, em que há um grande número de processos rodando de forma concorrente, não é possível que um programa, no caso o planejador de rotas, utilize a CPU de forma exclu-

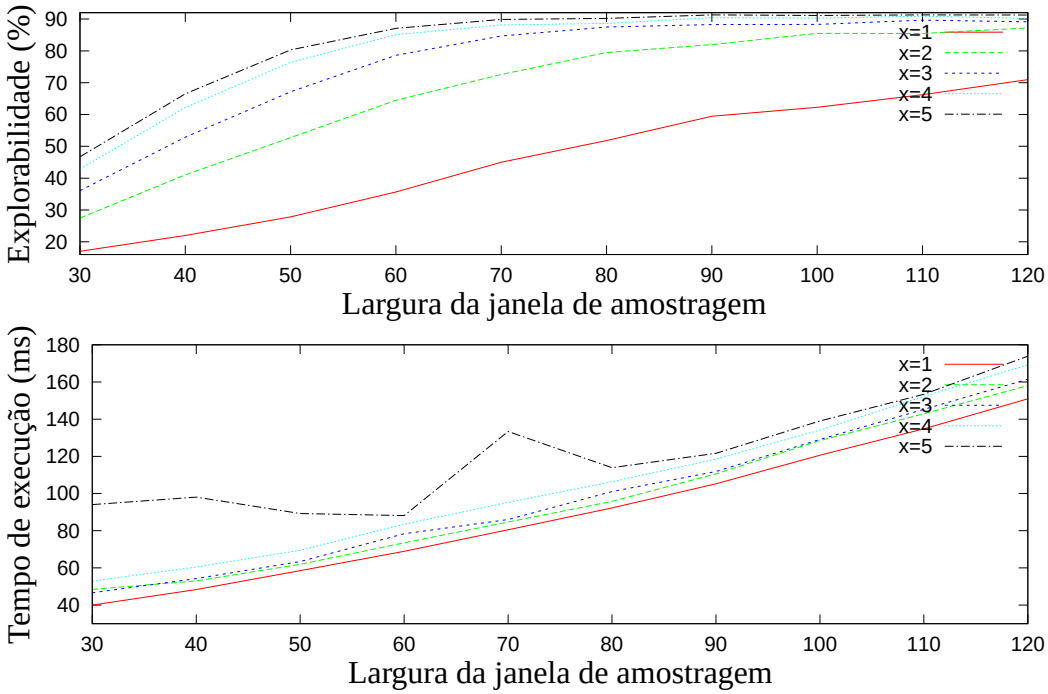


Figura 6.4: Desempenho do EST no ambiente simples. Foram realizadas 100 execuções com a geração de 500 amostras em cada uma delas.

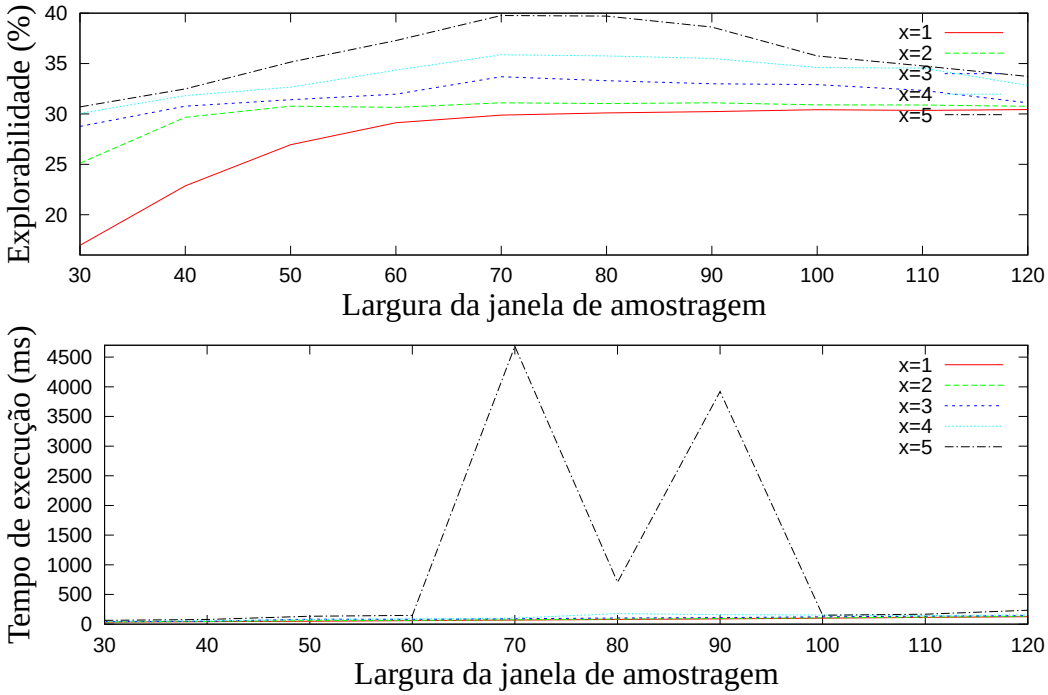


Figura 6.5: Desempenho do EST no corredor. Foram realizadas 100 execuções com a geração de 500 amostras em cada uma delas.

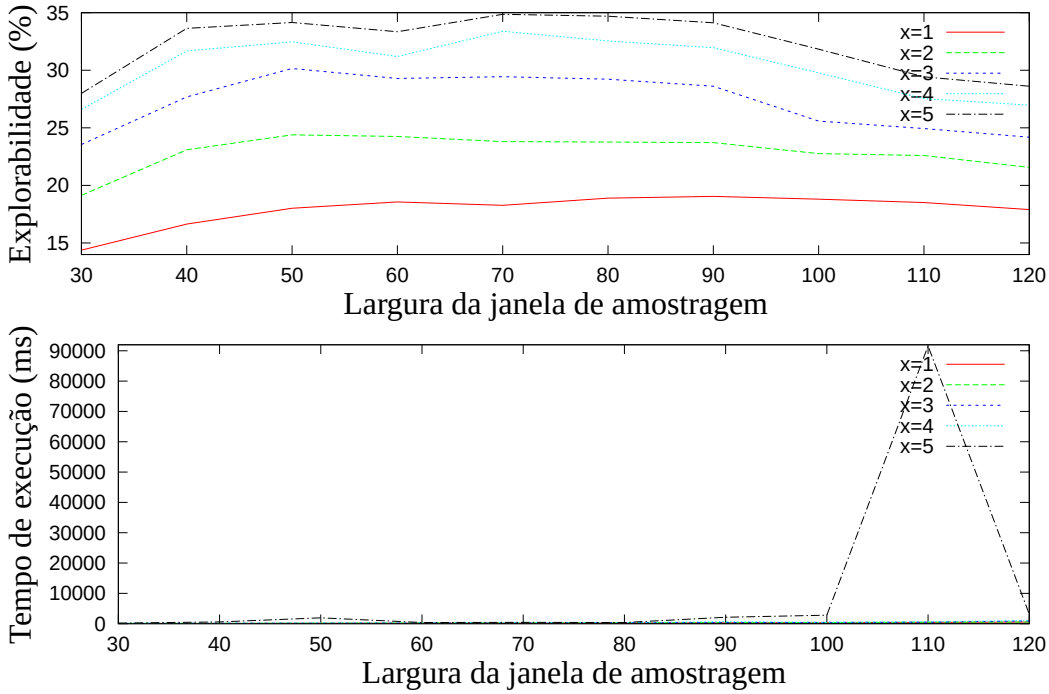


Figura 6.6: Desempenho do EST no labirinto. Foram realizadas 100 execuções com a geração de 500 amostras em cada uma delas.

siva. Porém, como a prioridade de execução do processo referente ao planejador é alta, estas interferências acontecem raramente.

Já a Figura 6.8 mostra o resultado da variação do parâmetro ϵ tanto no tempo de execução quanto na explorabilidade do RRT-expand. Nota-se que um aumento do valor de ϵ provoca um aumento na explorabilidade do planejador, porém a custo de um maior tempo de execução. Este efeito foi mais acentuado no Ambiente Simples e no Corredor. Já no Labirinto este parâmetro teve uma menor influência no desempenho do algoritmo. Isto pode ser atribuído ao grande número de passagens estreitas neste ambiente, de forma que, a partir de qualquer amostra, a região de visibilidade é muito pequena, sendo que para um determinado valor ϵ_0, qualquer valor acima fará com que passos muito grandes sejam dados, colocando as configurações geradas fora do campo de visibilidade da amostra de referência. Sendo assim, existe uma saturação na exploração. Já o tempo de execução fica maior para $\epsilon > \epsilon_0$, uma vez que muitas amostras são desperdiçadas.

6.2 TEMPOS DE SUAVIZAÇÃO

Antes da avaliação final entre os vários planejadores de questionamento único, é feita uma análise sobre os tempos de cada etapa de suavização, comparando-as entre si, assim como com o tempo necessário para exploração de um ambiente no contexto do algoritmo

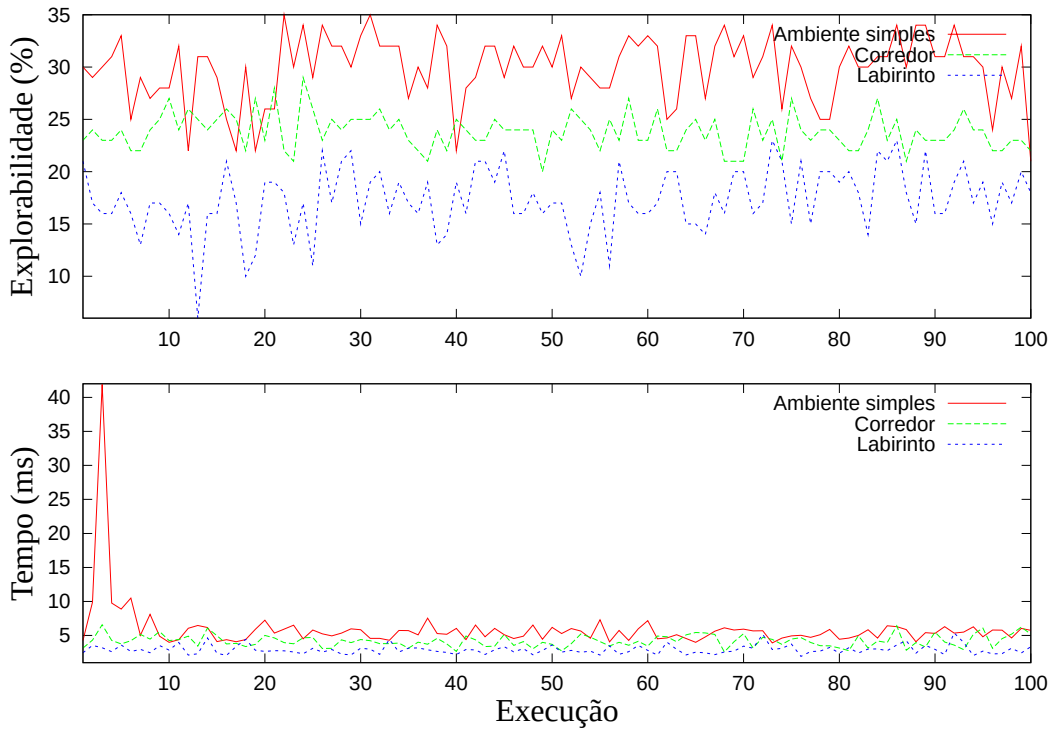


Figura 6.7: Avaliação do RRT-connect. Foram realizadas 100 execuções com 100 amostras geradas a cada execução.

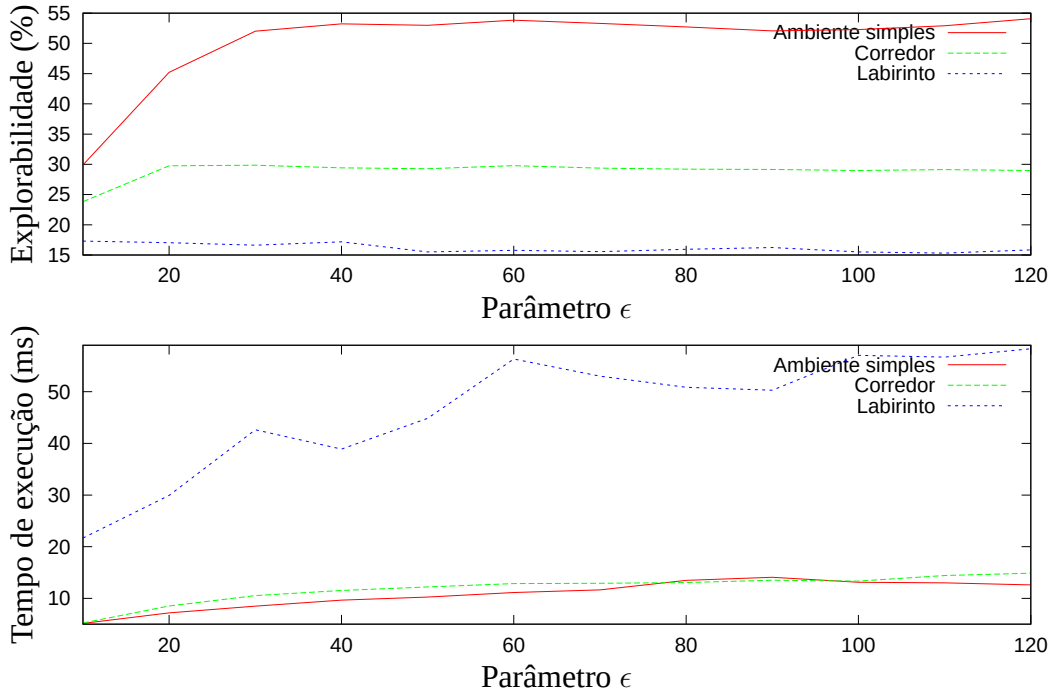
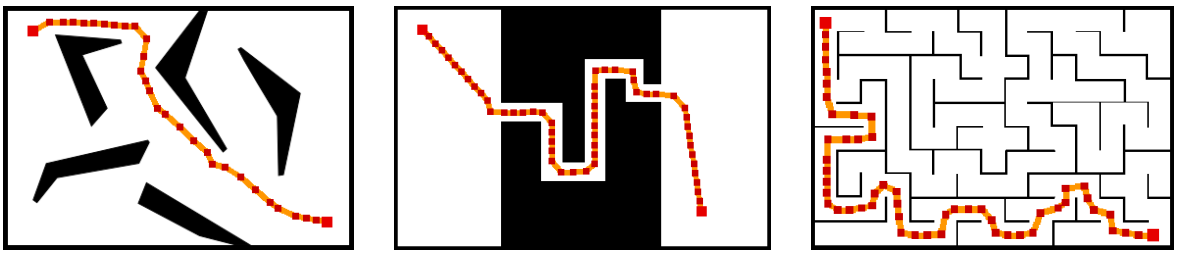


Figura 6.8: Influência do parâmetro ϵ no desempenho do RRT-expand. Foram realizadas 100 execuções com 100 amostras geradas a cada execução.



(a) Ambiente Simples. (b) Corredor. (c) Labirinto.

Figura 6.9: Rotas típicas para as requisições feitas nas avaliações das rotinas de suavização e dos planejadores de questionamento único.

ARW gaussiano bidirecional padrão. Este algoritmo foi escolhido por tipicamente gerar as rotas com pior qualidade dentre todos os algoritmos avaliados. Assim, espera-se que o tempo de suavização seja maior para este algoritmo.

Para avaliar os tempos de suavização em relação ao tempo de exploração do ARW, uma requisição não trivial de rota foi feita em cada um dos três ambientes da Figura A.1. As Figuras 6.9(a)-(c) mostram uma realização típica para cada uma dessas requisições nos ambientes Simples, Corredor e Labirinto, respectivamente.

A Figura 6.10 mostra dois gráficos. Eles foram obtidos por meio da média entre 100 execuções e fazendo $H = 50$. O gráfico superior mostra a comparação entre o tempo de execução para as várias etapas de suavização. Tanto no Corredor quanto no Labirinto, o método Dividir para Conquistar levou mais tempo para ser executado que os demais métodos, pois a rota de entrada continha muitos nós redundantes. Assim, como os métodos são aplicados sucessivamente, cada etapa é realizada em uma rota já tratada pela etapa anterior. Assim, a rota sucessivamente fica menos redundante e mais "bem comportada", então os respectivos tempos de execução são menores. No Ambiente Simples, a etapa de Remoção de Ciclos foi mais onerosa que a etapa Dividir para Conquistar. Como este ambiente possui somente regiões de grande visibilidade, a solução é encontrada utilizando-se menos amostras que nos outros ambientes. Assim, a etapa Dividir para Conquistar não necessariamente é a que consome maior tempo de execução.

No segundo gráfico da Figura 6.10 são mostrados os tempos para encontrar a solução e para a suavização da rota resultante. *É imprescindível notar que a escala de tempo é logarítmica*. Observa-se claramente que o tempo de suavização é muito menor que o tempo necessário para achar a solução. De fato, em ambientes difíceis, como o Corredor, o tempo médio de suavização, um pouco mais de 60 milisegundos, é ínfimo quando comparado com o tempo médio para achar a solução, que é aproximadamente igual a 20 segundos.

A Figura 6.11 mostra um exemplo para as várias etapas de suavização aplicadas sucessivamente a uma rota gerada pelo algoritmo ARW. A Figura 6.11(a) mostra a solução sem nenhuma suavização. A rota possui várias sinuosidades, assim como nós redundantes, sendo de péssima qualidade e cuja execução é inviável. A Figura 6.11(b) mostra que o algoritmo

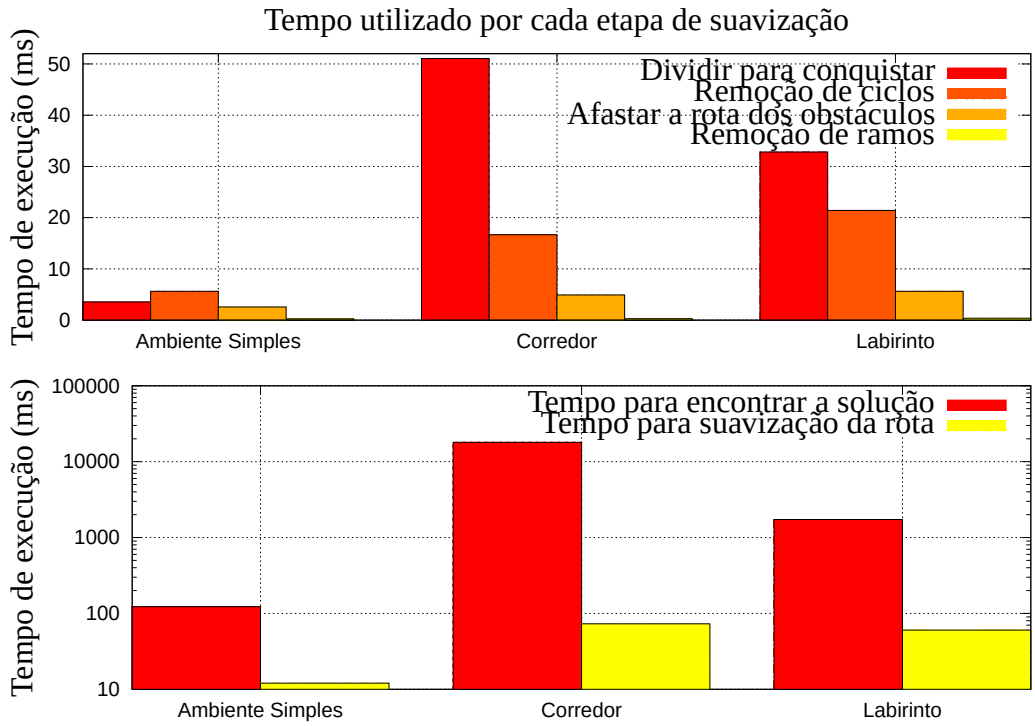
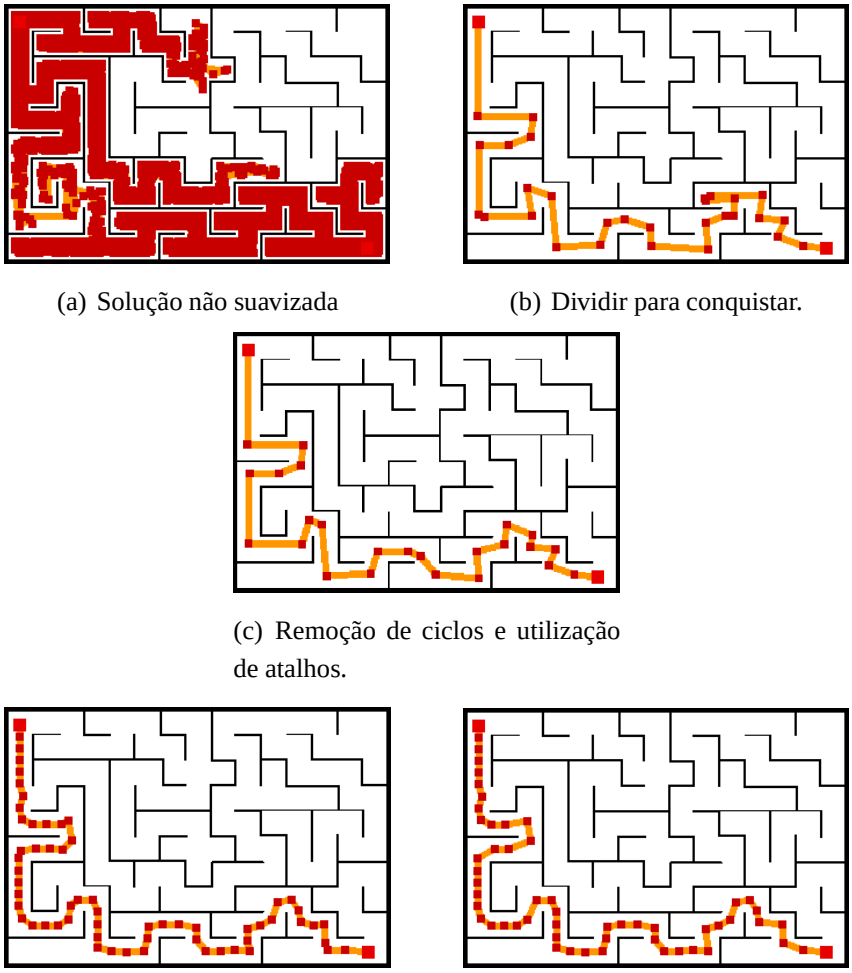


Figura 6.10: Comparação entre o tempo de exploração e o tempo de suavização no ARW gaussiano padrão. Foram realizadas 100 execuções com a história do processo ajustada como $H = 50$.

Dividir para Conquistar, apesar de já eliminar boa parte dos nós redundantes, retorna ainda uma rota de baixa qualidade, com algumas sinuosidades e que passa arbitrariamente próxima aos obstáculos. A Figura 6.11(c) mostra o resultado do algoritmo para remoção de ciclos e utilização de atalhos. A rota não possui nós desnecessários, mas em alguns poucos pontos ela passa arbitrariamente perto dos obstáculos. Assim, a Figura 6.11(d) mostra a rota afastada dos obstáculos. Como os corredores são estreitos, as configurações são levadas para o eixo médio. Nota-se que a rota não possui ramos, de forma que o algoritmo para remoção dos ramos não faz alterações significativas, sendo a rota final mostrada na Figura 6.11(e).

6.3 ALGORITMOS DE QUESTIONAMENTO ÚNICO

Nas Seções 6.1.1.3, 6.1.2 e 6.1.3 os algoritmos de questionamento único foram avaliados individualmente quanto ao critério de explorabilidade e tempo necessário para gerar um número fixo de amostras. A primeira etapa, além de verificar o desempenho segundo os critérios mencionados, também serviu para ajustar os parâmetros de cada algoritmo. Assim, nesta Seção é feita uma comparação entre os planejadores de questionamento único – já com os respectivos parâmetros ajustados – para avaliar os seus desempenhos quando a tarefa consiste em achar uma solução para o problema de planejamento de rotas.



(a) Solução não suavizada

(b) Dividir para conquistar.

(c) Remoção de ciclos e utilização de atalhos.

(d) Afasta a rota dos obstáculos.

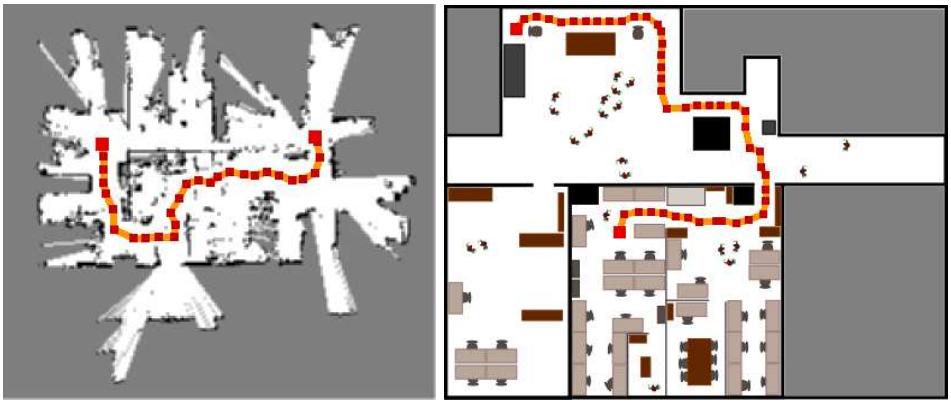
(e) Remoção dos ramos.

Figura 6.11: Passos seqüenciais para a suavização final de uma rota no Labirinto.

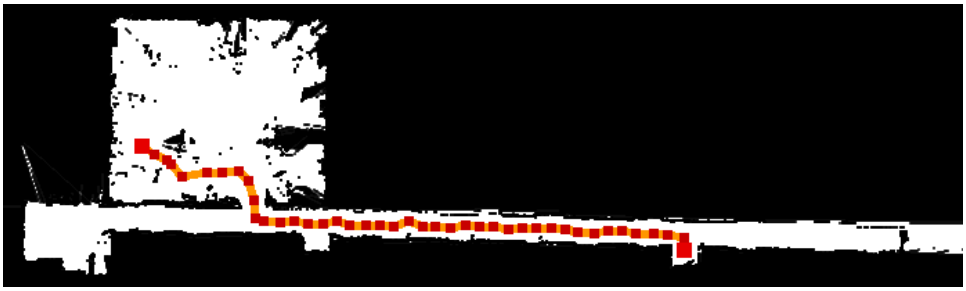
Nessa avaliação foram utilizados os ambientes das Figuras A.1 e A.2. O ajuste do ARW e suas variantes foi feito com base nos resultados da Seção 6.1.1. O valor de H foi escolhido com base na Figura 6.1. Foi notado que para valores de H maiores que 50, não houve melhoria na explorabilidade tanto no Labirinto quanto no Corredor e houve uma pequena melhoria no Ambiente Simples. Desta forma, o valor escolhido foi $H = 50$.

Para as variantes do ARW que utilizaram seleção de candidatas, a escolha do número de candidatas foi guiada pelas Figuras 6.2 e 6.3. Analisando essas duas Figuras, nota-se que o o uso de mais de 10 candidatas torna-se inviável pelo grande tempo de execução. Sendo assim, foram escolhidas versões com 2, 5 e 8 candidatas. *É importante ressaltar que o ARW e todas as variantes foram utilizadas na configuração bidirecional.*

O EST foi ajustado para $x = 3$ e a largura da janela de amostragem igual a 70 unidades. Já o RRT foi ajustado com $\epsilon = 40$ para a etapa de exploração, sendo que a etapa de fusão das árvores é feita pelo RRT-connect, exatamente como descrito na Seção 3.3 pelo Algoritmo 4. O planejador por frente de onda suavizado foi utilizado para efeitos de comparação.



(a) LIRMM 1. (b) LARA.



(c) LIRMM 2.

Figura 6.12: Rotas típicas para as requisições feitas na avaliação dos planejadores de questionamento único.

Para os testes foram dados os pontos inicial e final, registrados os tempos que cada algoritmo levou para achar a solução e suavizar a rota e ainda foi registrada sua distância final. As rotas típicas para a requisição em cada ambiente são mostradas nas Figuras 6.9 (já utilizadas nos testes da Seção 6.2) e 6.12. O resultado da avaliação pode ser resumido pela Figura 6.14. Foram realizadas 100 execuções para cada algoritmo, sendo que os resultados são dados pela média entre estas execuções. A Figura é dividida em três gráficos: o mais acima mostra o tempo médio necessário para achar a solução em cada ambiente. *É importante notar que a escala de tempo para este gráfico é logarítmica*; o do meio mostra o tempo médio necessário para a suavização da rota; e o terceiro mostra a distância final percorrida pela rota.

O primeiro fato a ser notado é que os algoritmos ARW com 2, 3 e 8 candidatas tiveram um desempenho bem superior aos gaussiano e uniforme com apenas uma amostra, indicando que a polarização da amostragem rumo a regiões pouco exploradas é eficaz quando comparado com o algoritmo original proposto em [9].

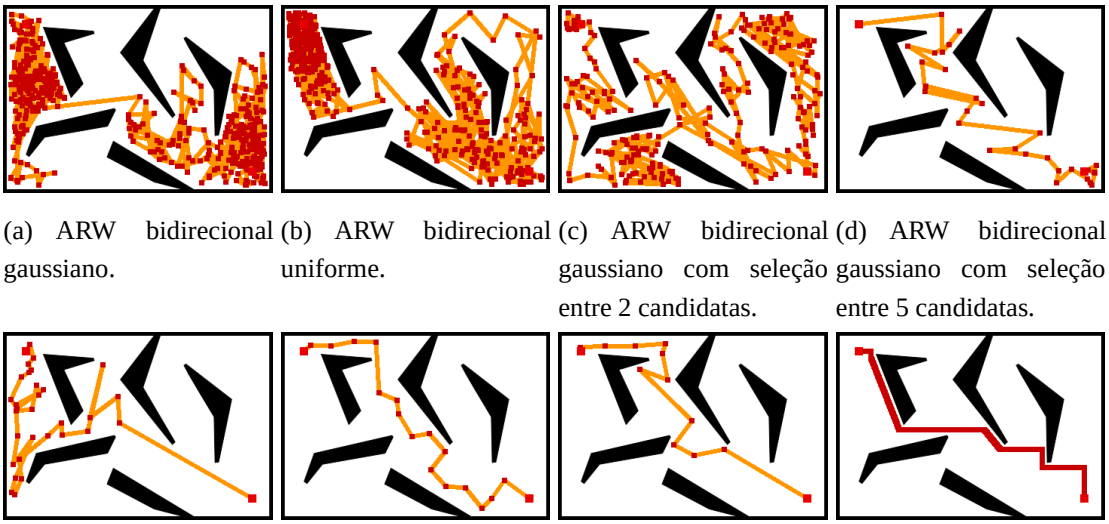
Com exceção do Ambiente Simples e do Corredor, o planejador por frente de onda teve um desempenho superior ao dos algoritmos estocásticos. Isso já era esperado, uma vez que a avaliação foi feita para $\mathcal{C} \in \mathbb{R}^2$, sendo que para baixas dimensões o planejador por frente de onda tem bom desempenho [19]. Porém, no Ambiente Simples os desempenhos de todos os algoritmos estocásticos foram melhores, pois enquanto estes planejadores conseguiam se

conectar ao destino tão logo começavam a execução (este ambiente possui grandes regiões com alta visibilidade), o planejador por frente de onda tinha que fazer todo o processo de propagação da sua onda de potencial entre os pontos final e inicial. Isto indica que o PFO foge à classificação de planejador de questionamento único enquanto $\mathbf{q}_{goal}$ é constante, pois ele acha a solução que conecta todas as configurações de $\mathcal{C}_{free}$ à configuração de destino. Isto é particularmente útil para o caso de haver necessidade de um replanejamento local devido ao aparecimento de um obstáculo não previsto (*e.g.*, uma pessoa ou outro objeto móvel). No entanto, para uma seqüência de questionamentos em que o ponto de destino sempre muda, o PFO funciona como um algoritmo de questionamento único, ou seja, não é possível aproveitar informações dos questionamentos anteriores.

O RRT obteve um péssimo desempenho no mapa LIRMM 1, o que talvez pode ser explicado pelo valor $\epsilon = 40$. De fato, este é um ambiente com muitas passagens bem estreitas e talvez um menor valor para ϵ tivesse aumentado o desempenho do algoritmo.

Pode-se notar também que, principalmente no Corredor e Labirinto, o tempo de suavização foi maior para o ARW gaussiano e ARW uniforme. Isto já era esperado, uma vez que o ARW padrão gera muitas amostras em sua etapa de exploração, aumentando o tempo de processamento na etapa de suavização. O tempo de suavização dos algoritmos EST, RRT e planejador por frente de onda foram os menores, uma vez que tanto o RRT quanto EST são algoritmos com grande ênfase na explorabilidade por meio da geração de um número relativamente reduzido de amostras. Já o planejador por frente de onda é um algoritmo ótimo de acordo com o critério da distância de Manhatan, o que implica que a suavização consiste em basicamente pegar alguns atalhos e afastar a rota dos obstáculos, não havendo necessidade de retirar nós redundantes. A Figura 6.13 mostra rotas não suavizadas típicas para os planejadores de questionamento único avaliados. As Figuras 6.13(a) e 6.13(b), que correspondem aos ARW gaussiano e uniforme, respectivamente, são as que contém um maior número de nós na rota. Observa-se que as Figuras 6.13(c)-(e) possuem gradativamente menos nós na medida em que o número de candidatas vão aumentando, o que corresponde exatamente aos resultados da Seção 6.1.1.2. Já as Figuras 6.13(f) e 6.13(g) mostram que a as rotas geradas pelos algoritmos EST e RRT, mesmo quando não suavizadas, são de boa qualidade e bem superiores àquelas geradas pelo ARW e variantes. Por último, a rota gerada pelo PFO, como já esperado, ainda que com a melhor qualidade dentre todas as rotas, é gerada muito próxima aos obstáculos.

Um fato notável que comprova a eficiência da etapa de suavização é a distância final das rotas geradas por todos os planejadores. Ainda que as rotas geradas na etapa de exploração tivessem um número muito diferente de configurações intermediárias para os diversos planejadores (indo de algumas dezenas para o planejador por frente de onda até algumas milhares para o ARW gaussiano), as rotas resultantes ficaram praticamente com a mesma distância para todos os planejadores, como pode ser visto no gráfico mais abaixo da Figura 6.14.



(a) ARW bidirecional gaussiano.

(b) ARW bidirecional uniforme.

(c) ARW bidirecional gaussiano com seleção entre 2 candidatas.

(d) ARW bidirecional gaussiano com seleção entre 5 candidatas.

(e) ARW bidirecional gaussiano com seleção entre 8 candidatas.

(f) EST.

(g) RRT.

(h) Planejador por frente de onda.

Figura 6.13: Rotas típicas não suavizadas para os planejadores de questionamento único.

6.4 ALGORITMOS DE MÚLTIPLOS QUESTIONAMENTOS

Nesta seção são avaliados os algoritmos PRM padrão, PRM gaussiano e iARW. Para o PRM padrão os parâmetros utilizados foram: número máximo de vizinhos igual a 10; e distância máxima para conexão com os vizinhos igual a 60 unidades. Para o PRM gaussiano foram usados esses mesmos parâmetros e ainda $\sigma_q = 50$. Para o iARW, o algoritmo utilizado para exploração foi o ARW gaussiano com seleção entre três amostras. Já o valor da história do processo foi $H = 50$.

Em todos os ambientes das Figuras A.1 e A.2 foram escolhidas para teste quatro rotas. Os pontos inicial e final destas rotas nos espaços de trabalho são mostrados na Figura A.3. Os números escritos em tons claros correspondem ao ponto inicial da rota, sendo que os escritos em tons escuros correspondem ao ponto final.

Para o iARW, as requisições de rotas foram feitas uma a uma de forma que o próprio algoritmo construísse e utilizasse o mapa de rotas sem a necessidade de pré-processamento e determinação *a priori* do número de amostras a serem geradas. Já para os algoritmos PRM uniforme e gaussiano, uma vez que o número de nós a serem gerados na etapa de construção tem que ser determinado *a priori*, a estratégia adotada em cada ambiente foi a seguinte. Inicialmente, 200 nós eram gerados. Caso eles não fossem suficientes para capturar a conectividade de $\mathcal{C}_{free}$, o que se caracteriza pela falha em encontrar pelo menos uma das quatro rotas de teste, o mapa de rotas era apagado e o processo de amostragem reiniciado com o dobro de amostras da tentativa anterior. O processo era repetido até que as quatro rotas fossem encontradas, sendo que as amostras se mantinham constantes de uma execução para a outra se não houvesse falha em gerar as quatro rotas.

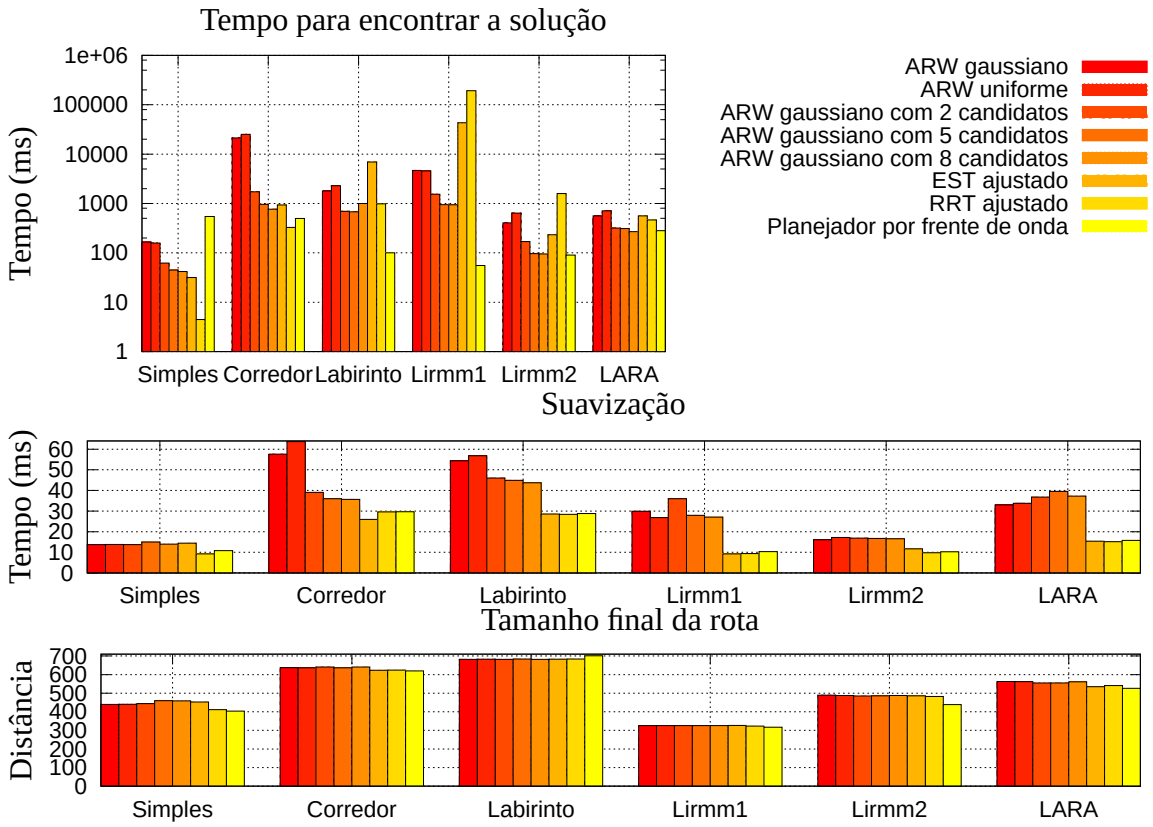


Figura 6.14: Avaliação dos algoritmos de questionamento único. Foram realizadas 100 execuções para encontrar uma rota de referência.

Os resultados estão resumidos na Figuras 6.15, 6.16 e 6.17. O tempo total para os algoritmos PRM uniforme e gaussiano consiste no tempo necessário para gerar as amostras na etapa de construção mais o tempo necessário para achar as quatro rotas na etapa de questionamento. Já para o iARW o tempo total é basicamente o somatório dos tempos necessários para achar cada uma das quatro rotas. *É importante observar que a escala de tempo adotada é logarítmica*. Nota-se que o iARW precisou de menos tempo para achar as quatros rotas que os demais algoritmos em todos os ambientes de teste, com exceção do Labirinto, no qual o iARW teve o mesmo desempenho que o PRM gaussiano.

Além disso, o mapa de rotas resultante do iARW contém um número muito menor de amostras que os respectivos mapas de rotas dos algoritmos PRM uniforme e gaussiano. Isso acontece porque o iARW armazena somente as amostras de $\mathcal{C}_{free}$ que foram relevantes para solucionar o problema de planejamento de rotas em questão, enquanto que os PRMs amostram porções irrelevantes de $\mathcal{C}_{free}$. Ainda que o PRM gaussiano faça uma amostragem mais seletiva do espaço de configurações livre, o iARW faz uma seleção ainda maior.

Em geral, para o iARW a primeira rota leva mais tempo para ser gerada, sendo que as rotas subseqüentes, por usarem o mapa de rotas, possuem tempos de execução muito inferiores. A Figura 6.17 mostra que, com exceção do Labirinto, isto acontece em todos os ambientes de teste. De fato, as rotas requisitadas no Labirinto não tinham muitos pontos de

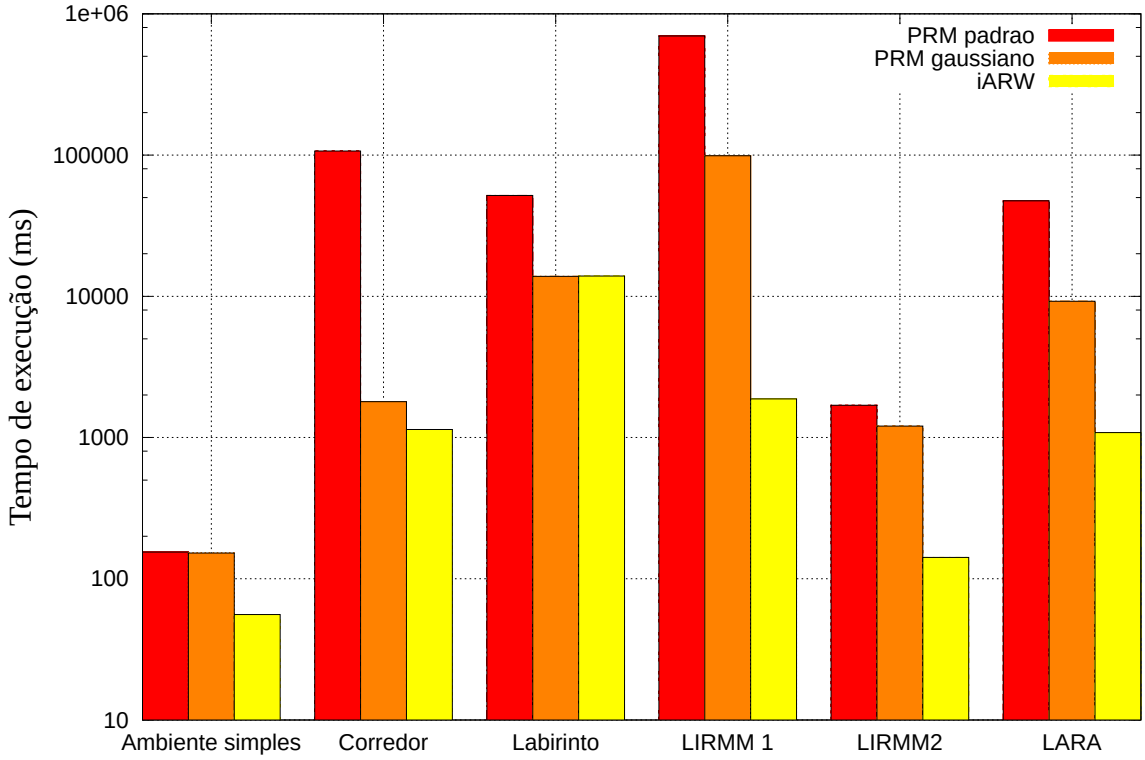


Figura 6.15: Tempo total necessário para encontrar quatro rotas pré-determinadas na avaliação dos algoritmos de múltiplos questionamentos.

sobreposição, de forma que nem sempre era possível ao iARW fazer um uso muito eficiente do mapa de rotas. Contudo, considerando o tempo total para achar as quatro rotas, o iARW obteve o mesmo desempenho do PRM gaussiano neste ambiente.

O iARW mostrou-se um algoritmo computacionalmente eficiente para resolver o problema de planejamento de rotas. Isto se deve, primeiramente, ao fato dele usar o algoritmo ARW gaussiano com seleção de amostras, que já se mostrou com alto desempenho, para exploração do ambiente. Em segundo lugar, a utilização de um mapa de rotas para utilizar caminhos que já haviam sido criados em questionamentos anteriores aumenta o desempenho do iARW na medida em que muitas rotas são requisitadas. Assim, quando comparado com algoritmos de questionamento único, o seu desempenho pode ser aproximado nas primeiras requisições de rotas pelo desempenho do ARW gaussiano com seleção de amostras, pois este é o algoritmo utilizado para exploração. Porém, na medida em que muitas rotas são geradas, o iARW passa a ter um desempenho semelhante ao do PRM na fase de questionamento, o que implica em uma grande eficiência.

Quando comparado com algoritmos PRM uniforme e gaussiano, além de possuir menor tempo de execução e armazenar um número menor de configurações no mapa de rotas, o iARW apresenta vantagens por ser um algoritmo incremental, sendo que o número de amostras a serem geradas não precisam ser determinadas *a priori*. Além disso, ele é de

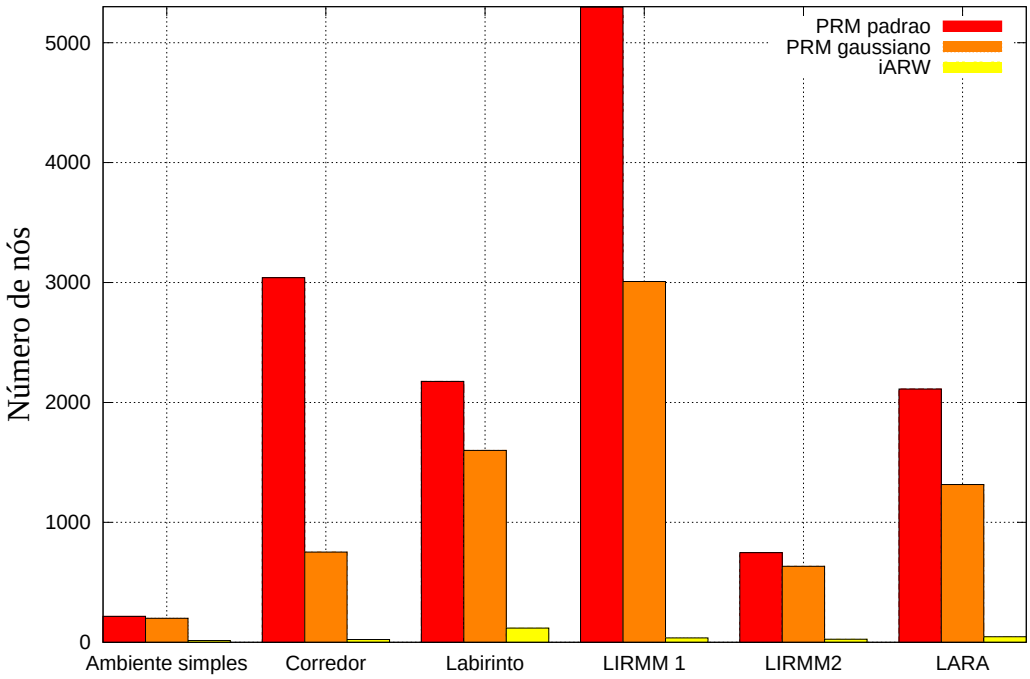


Figura 6.16: Número de nós resultantes nos mapas de rotas da avaliação dos algoritmos de múltiplos questionamentos.

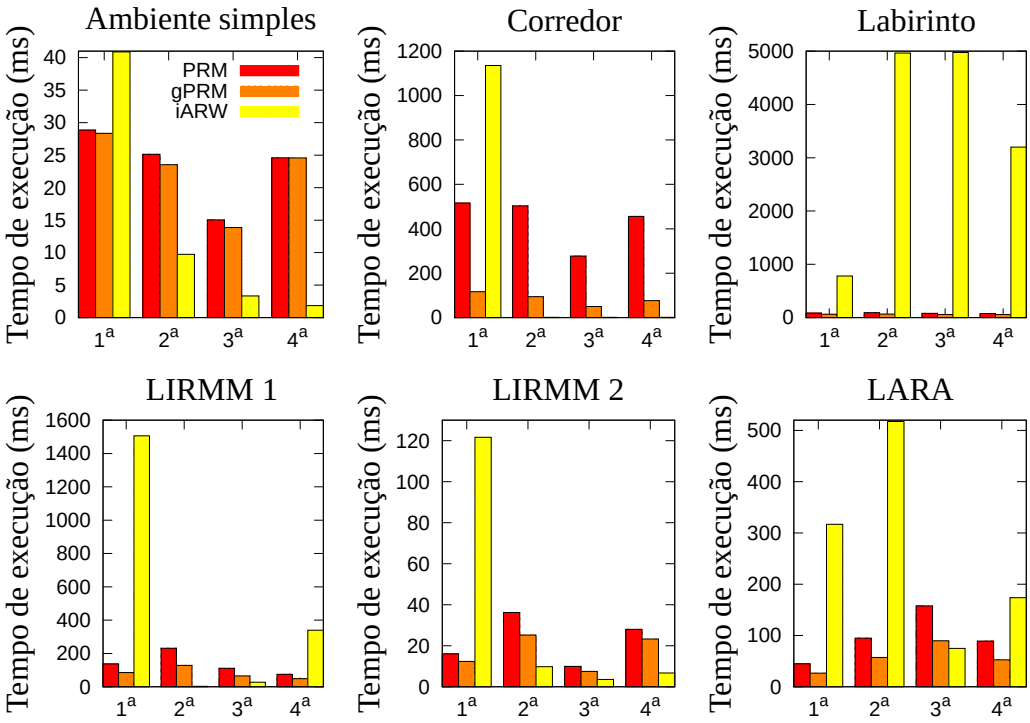


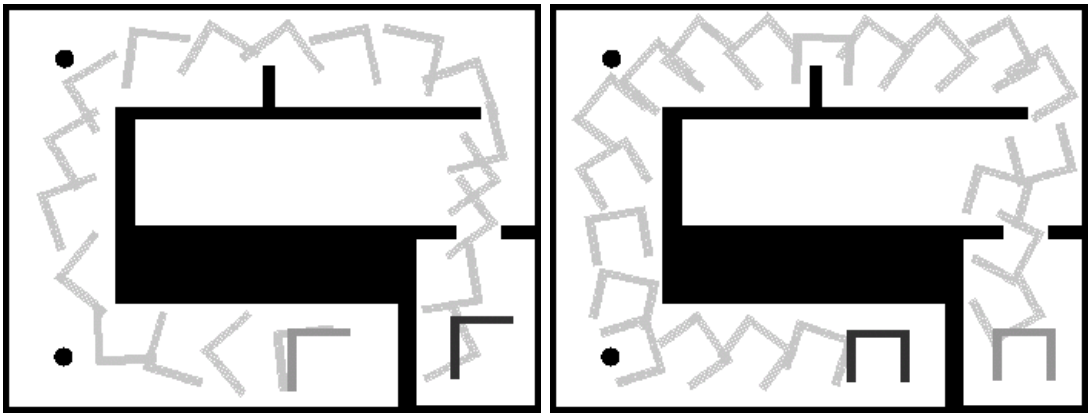
Figura 6.17: Tempo de execução de cada uma das quatro rotas na avaliação dos algoritmos de múltiplos questionamentos.

implementação simples e não utiliza estrutura de dados complexa, sendo que uma grande vantagem, quando comparado com os algoritmos RRT e PRM, é que não há necessidade de fazer a busca por vizinhos mais próximos.

6.5 PLANEJAMENTO DE ROTAS PARA $\mathcal{C} \in \mathbb{R}^2 \times \mathbb{S}^1$

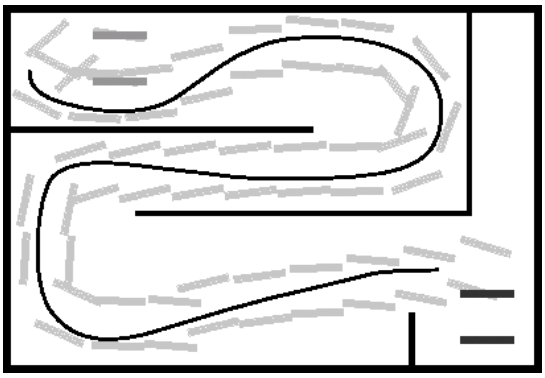
Ainda que neste trabalho a ênfase tenha sido dada a espaços de configuração euclidianos de duas dimensões, os planejadores apresentados podem ser aplicados a espaços de configuração com topologias mais complexas. De fato, em robótica móvel um espaço bem comum é dado por $\mathcal{C} \in \mathbb{R}^2 \times \mathbb{S}^1$. Assim, para exemplificar a aplicação neste espaço que permite rotação, foram geradas as três rotas da Figura 6.18. O algoritmo utilizado foi o ARW uniforme com seleção entre duas candidatas. Porém, qualquer um dos outros algoritmos probabilísticos apresentados poderiam ser utilizados. Nas Figuras 6.18(a) e 6.18(b), um robô em formato de L e U, respectivamente, foram utilizados para achar rotas não triviais. O ponto de partida está representado em tons escuros, enquanto a rota está representada em tons claros. Por último, o ponto de chegada está representado em uma tonalidade intermediária. Nota-se que muitos planejadores baseados em funções de potencial falhariam neste mapa para a situação da Figura 6.18(b), pois o ponto de partida localiza-se em uma grande concavidade, sendo que o ponto de destino está bem próximo deste ponto de partida. Assim, se o planejador for semelhante ao apresentado na Seção 3.1, em que o cálculo da função de potencial de atração usa a distância euclidiana entre o robô e o ponto de destino, sem considerar os obstáculos, então o ponto de partida da Figura 6.18(b) será um ponto de mínimo local. Este problema não existe nos planejadores probabilísticos apresentados neste trabalho. Por último, a Figura 6.18(c) mostra um robô com formato atípico, formado por duas barras paralelas. O ponto de partida é representado por tons escuros e encontra-se no canto inferior direito da Figura. A rota está representada em tons claros e o ponto de chegada está representado em uma coloração intermediária e localiza-se no canto superior esquerdo. Nota-se que o único caminho ligando os dois pontos faz com que os obstáculos passem entre as duas barras paralelas.

O mais interessante desses três exemplos é que eles usam exatamente o mesmo planejador. De fato, como já visto na Seção 3.6, o único conhecimento que o planejador tem é da topologia do espaço de configurações, mas não da geometria do robô. A conexão entre a geometria do robô e o espaço de configurações livre é feita por um detector de colisões. Esta abstração permite que o planejador de rotas seja independente do modelo geométrico utilizado para representar o robô, assim como da representação dos obstáculos. Assim, o planejador tem a função de gerar amostras no espaço de configurações, enquanto o detector de colisões tem a função de verificar se elas devem ser rejeitadas ou aceitas. Esta é a grande vantagem dos planejadores baseados na amostragem do espaço de configurações, pois $\mathcal{C}_{free}$ não precisa ser calculado explicitamente. Outra vantagem é que estes planejadores funcionam muito bem para espaços de configurações com dimensões maiores que três, sendo que



(a) Robô com formato em L.

(b) Robô com formato em U.



(c) Robô com formato atípico.

Figura 6.18: Planejamento para $\mathcal{C} \in \mathbb{R}^2 \times \mathbb{S}^1$. Pontos iniciais em tom escuro, rotas em tom claro e pontos finais com uma cor intermediária.

se a topologia do espaço for aproximada por um espaço Euclidiano, a dimensão passa a ser apenas um parâmetro para o planejador de rotas, sendo que o algoritmo a ser utilizado continua a ser exatamente o mesmo que aquele utilizado em problemas com dimensões menores.

7 CONCLUSÕES

L'homme libre est celui qui
n'a pas peur d'aller
jusqu'au bout de sa pensée.
Léon Blum

7.1 DISPOSIÇÕES FINAIS

Este trabalho apresentou técnicas para planejamento de rotas no contexto de robótica móvel. A ênfase foi em algoritmos probabilísticos com bom desempenho e que, com um pós-processamento adequado, retornam rotas de boa qualidade. Apesar do estudo ter sido feito visando a aplicação em um robô omnidirecional no plano, os algoritmos estudados podem ser aplicados em problemas mais gerais, como animação de atores artificiais ou sistemas multirrobôs.

Inicialmente, foi apresentada a definição do problema de planejamento de rotas para um robô poliédrico em um espaço de trabalho genérico. Então, as principais aplicações foram mostradas, mostrando o estado da arte nesta área de pesquisa. Em seguida, a formulação do espaço de configurações foi apresentada, viabilizando um tratamento unificado para robôs com diferentes formatos e modelos geométricos. Então, visando manter a continuidade do trabalho, foram revisados modelos para representação do robô e obstáculos no espaço de trabalho.

No Capítulo 3, vários métodos probabilísticos que atualmente compõem o estado da arte em planejamento de rotas foram discutidos. Foi mostrado que os algoritmos probabilísticos, em geral, são computacionalmente eficientes, mas retornam rotas com baixa qualidade. Esses métodos, além de aproximarem o espaço de configurações por amostras, não necessitam da representação explícita do espaço de configurações livre. Porém, existe a necessidade de uma função que avalia se uma configuração está ou não em colisão. Estas funções são disponíveis na forma de detectores de colisão. Um algoritmo simples baseado em funções de potencial, o planejador por frente de onda, foi apresentado para efeitos comparativos. Os métodos probabilísticos de questionamento único – RRT, EST, ARW e variantes – foram comparados qualitativamente entre si e com duas versões do algoritmo PRM de múltiplos questionamentos. Em seguida, foram apresentadas metodologias de amostragem determinística, bem como técnicas de amostragem não uniforme no contexto do PRM. Então, técnicas de suavização de rotas foram apresentadas, visando melhorar a qualidade das rotas resultantes.

Uma contribuição teórica para o algoritmo ARW foi realizada. Assim, sua distribuição foi caracterizada em função de uma região de visibilidade. Esta caracterização foi útil no sentido de possibilitar a reformulação do algoritmo como apenas uma soma de variáveis aleatórias. Assim, as condições de convergência – inspirada na teoria clássica de Spitzer sobre passeios aleatórios em espaços discretos – foram estabelecidas para passeios aleatórios que seguem a distribuição caracterizada neste trabalho. Contudo, diferentemente dos passeios aleatórios de Spitzer, os analisados neste trabalho não satisfazem a condição de homogeneidade espacial.

Outra contribuição deste trabalho foi a concepção de um algoritmo híbrido que utiliza passeios aleatórios adaptivos, com distribuição gaussiana e seleção entre amostras candidatas, para exploração do espaço de configurações, e que, de maneira incremental, constrói um mapa de rotas para armazenar rotas que foram utilizadas em questionamentos anteriores. Foi mostrado que o desempenho do algoritmo é satisfatório quando comparado tanto com os algoritmos de questionamento único, quanto com os algoritmos de múltiplos questionamentos.

7.2 SUGESTÕES PARA TRABALHOS FUTUROS

A área de pesquisa em planejamento de rotas é muito vasta, tanto no que se refere a desenvolvimento de algoritmos e avaliação experimental em robôs com diferentes modelos geométricos, como em análise teórica.

Em relação ao desenvolvimento de algoritmos e avaliação experimental, algoritmos probabilísticos poderiam ser aplicados em robôs mais complexos, como o robô humanóide do LARA[1], por exemplo. A incorporação do modelo dinâmico no planejador local possibilitaria tanto a geração de movimento quanto o desvio de obstáculos, assumindo que o modelo do espaço de trabalho esteja disponível. Além disso, o algoritmo iARW poderia ser estendido para robôs móveis não-holonômicos. De fato, apenas a extensão para o algoritmo ARW seria suficiente para se caracterizar como uma contribuição na área de planejamento de rotas, uma vez que a literatura mostra aplicações deste algoritmo apenas para robôs holonômicos. Esta extensão poderia ser feita por meio da reformulação do problema no espaço de estados, tal como já foi realizada no contexto do algoritmo RRT [15].

No que se refere à análise teórica, novas distribuições para o ARW poderiam ser estudadas, assim como a determinação da taxa de convergência do algoritmo em função dos parâmetros do espaço de configurações. Além disso, um método praticável para estimar a probabilidade de alcançar o ponto de destino, em função do número de amostras geradas, poderia ser pesquisado.

[1] O Laboratório de Robótica e Automação foi o local onde este trabalho foi desenvolvido.

[1] R. R. Murphy, *Introduction to AI Robotics*. The MIT Press, 2000.

[2] H. Choset, K. M. Lynch, S. Hutchinson, W. B. G. Kantor, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[3] J. Barraquand, L. Kavraki, J. C. Latombe, and T. Li, "A random sampling scheme for path planning," *The International Journal of Robotics Research*, vol. 16, pp. 759–774, 1997.

[4] L. E. Kavraki, M. N. Kolountzakis, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, 1996.

[5] J. Barraquand and J. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, pp. 628–649, 1991.

[6] J. J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

[7] D. Hsu, "Randomized single-query motion planning in expansive spaces," Ph.D. dissertation, Department of Computer Science, Stanford University, 2000.

[8] K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki, "Multiple query motion planning using single query primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, Nevada, USA, 2003, pp. 656–661.

[9] S. Carpin and G. Pillonetto, "Motion planning using adaptive random walks," *IEEE Transactions on Robotics*, vol. 21, no. 1, 2005.

[10] V. Boor, M. H. Overmars, and A. F. Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," 1999, pp. 1018–1023.

[11] J. P. Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," 2004, pp. 453–460.

[12] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmaps," *IEEE Transaction on Robotics*, vol. 21, no. 6, pp. 1105–1115, 2005.

[13] S. A. Wilmarth, N. M. Amato, and P. E. Stiller, "Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 1024–1031.

[14] R. Geraerts and M. Overmars, "Sampling techniques for probabilistic roadmap planners."

[15] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proceedings IEEE International Conference on Robotics and Automation*, 1999, pp. 473–479.

[16] R. Geraerts and M. H. Overmars, "Clearance based path optimization for motion planning," New Orleans, LA, 2004, pp. 2386–2392.

[17] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual IEEE Symposium on Foundations of Computer Science*, 1979, pp. 421–427.

[18] ——, *Complexity of the Generalized Mover's Problem*. Norwood, NJ: Ablex Pub., 1987, ch. 11, pp. 267–281.

[19] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[20] L. C. A. Pimenta, A. R. Fonseca, G. A. S. Pereira, R. C. Mesquita, E. J. Silva, W. M. Caminhas, and M. F. M. Campos, "Robot navigation based on electrostatic field computation," *IEEE Transactions on Magnetics*, vol. 42, no. 4, pp. 1459–1462, April 2006.

[21] J. P. Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," 2004, pp. 1598–1605.

[22] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," 2004, pp. 4350–4355.

[23] J. Burlet, O. Aycard, and T. Fraichard, "Robust motion planning using markov decision processes and quadtree decomposition," 2004, pp. 2820–2825.

[24] S. Griffits, J. Saunders, A. Curtis, B. Barber, T. McLain, and R. Beard, "Maximizing miniature aerial vehicles," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 34–43, 2006.

[25] E. Ferré, J.-P. Laumond, G. Arechavaleta, and C. Estevès, "Progresses in assembly path planning," in *International Conference on Product Lifecycle Management*, Lyon, France, Juillet 2005, pp. 373–382.

[26] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning collision-free reaching motions for interactive object manipulation and grasping," in *Eurographics*, vol. 22, no. 3, 2003.

[27] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, “Motion planning for humanoid robots,” in *Proceedings International Symposium on Robotics Research*, 2003.

[28] C. Ughini and L. P. Nedel, “Adaptable deterministic roadmaps for motion planning of bodies with many dofs,” in *Proceedings of the 20th International Conference on Computer Animation and Social Agents (CASA*, vol. 1, Hasselt University, 2007, pp. 19–26.

[29] M. L. Teodoro, G. N. P. Jr, and L. E. Kavraki, “Molecular docking: A problem with thousand of degrees of freedom,” in *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, 2001, pp. 960–965.

[30] M. Apaydin, D. Brutlag, C. Guestrin, and D. H. J.-C. Latombe, “Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion,” 2002, pp. 12–21.

[31] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki, “Deformable volumes in path planning applications,” in *IEEE International Conference on Robotics and Automation*, vol. 3.

[32] S. Rodríguez, J.-M. Lien, and N. M. Amato, “Planning motion in completely deformable environments,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA), 2006*, May 2006, pp. 2466–2471.

[33] B. V. Adorno, C. S. R. Aguiar, and G. A. Borges, “Planejamento de trajetória para o robô omni usando o algoritmo mapa de rotas probabilístico,” in *Simpósio Brasileiro de Automação Inteligente*, 2005, pp. 1–8.

[34] B. V. Adorno and G. A. Borges, “Um método de planejamento de trajetória para robôs móveis através de passeios aleatórios adaptativos e mapa de rotas,” in *XVI Congresso Brasileiro de Automática*, 2006, pp. 1–6.

[35] ——, “Planejamento de caminho usando bi-arw melhorado e mapa de rotas,” in *Simpósio Brasileiro de Automação Inteligente*, 2007, pp. 1–6.

[36] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[37] M. W. Spong, S. Huthinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons, Inc, 2006.

[38] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[39] L. E. Kavraki, “Random networks in configuration space for fast path planning,” Ph.D. dissertation, Stanford University, 1995.

[40] R. Geraerts, “Sampling-based motion planning: Analysis and path quality,” Ph.D. dissertation, Utrecht University, 2006.

[41] D. Nieuwenhuisen and M. H. Overmars, “Useful cycles in probabilistic roadmap graphs,” in *Proceedings IEEE International Conference on Robotics & Automation*, 2004, pp. 446–452.

[42] A. Ladd and L. E. Kavraki, “Measure theoretic analysis of probabilistic path planning,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 229–242, April 2004.

[43] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, “On the relationship between classical grid search and probabilistic roadmaps,” *International Journal of Robotics Research*, vol. 23, no. 7/8, pp. 673–692, July/August 2004.

[44] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,” *Journal of the ACM*, vol. 45, pp. 891–923, 1998.

[45] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, “Choosing good distance metrics and local planners for probabilistic roadmap methods,” *IEEE Transactions on Robotics and Automation*, vol. 16, no. 4, pp. 442–447, 2000.

[46] A. Atramentov and S. M. LaValle, “Efficient nearest neighbor searching for motion planning,” in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, May 2002, pp. 632–637.

[47] T.-Y. Li and Y.-C. Shie, “An incremental learning approach to motion planning with roadmap management,” in *Proceedings IEEE International Conference on Robotics & Automation*, 2002.

[48] D. Hsu, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997, pp. 2719–2796.

[49] D. Hsu, “Randomized single-query motion planning in expansive spaces,” Ph.D. dissertation, Stanford University, May 2000.

[50] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, “Obprm: an obstacle-based prm for 3d workspaces,” in *Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics: the algorithmic perspective*, 1998, pp. 155–168.

[51] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang, “Quasi-randomized path planning.”

[52] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7/8, pp. 673–692, July/August 2004.

[53] X. Wang and F. J. Hickernell, "An historical overview of lattice point sets," in *Monte Carlo and Quasi-Monte Carlo Methods 2000*, K.-T. Fang, F. Hickernell, and H. Niederreiter, Eds. Springer-Verlag, 2002, pp. 158–167.

[54] J. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, no. 1, pp. 84–90, December 1960.

[55] S. R. Lindemann and S. M. LaValle, "Incremental low-discrepancy lattice methods for motion planning," in *Proceedings of the 2003 IEEE Internetional Conference on Robotics & Automation*, September 2003, pp. 2920–2927.

[56] S. Carpin and G. Pillonetto, "Merging the adaptive random walks planner with the randomized potential field planner," *Fifth International Workshop on Robot Motion and Control*, pp. 151–156, 2005.

[57] F. Spitzer, *Principles of Random Walk*, second edition ed. Springer-Verlag, 1976.

[58] G. A. Borges, "Cartographie de l'environnement et localisation robuste pour la navigation de robots mobiles," Ph.D. dissertation, Université Montpellier II, LIRMM, 161 rue ADA, 34392, Montpellier, Cedex 5, France., May 2002.

[59] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, fourth edition ed. Tata McGraw-Hill, 2002.

APÊNDICES

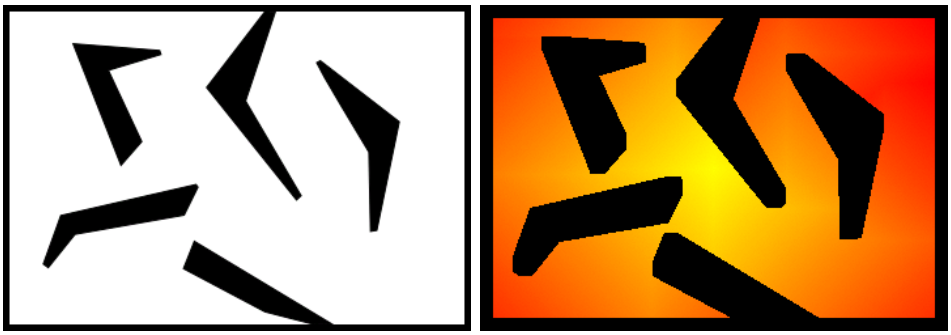
A. AMBIENTES DE TESTE UTILIZADOS NA AVALIAÇÃO DOS PLANEJADORES DE ROTAS

A Figura A.1 mostra os ambientes de teste utilizados para avaliação dos planejadores de questionamento único. Além disso, a alcançabilidade no espaço de configuração, a partir do ponto mais central, é mostrada para cada ambiente de teste. A Figura A.1(a) mostra um ambiente simples e com grande áreas de visibilidade. Já a Figura A.1(c) mostra dois grandes ambientes separados por um longo corredor e, apesar de ser trivial para um ser humano, costuma ser um ambiente difícil para os planejadores baseados na amostragem do espaço de configuração. No caso dos planejadores que amostram o espaço de configuração globalmente e de maneira uniforme, como o PRM, a probabilidade de uma amostra cair em uma região é diretamente proporcional ao volume desta região. Sendo assim, como o volume do corredor é pequeno, igualmente é pequena a probabilidade de conter um número de amostras que seja suficiente para capturar a conectividade de $\mathcal{C}_{free}$. Por outro lado, no caso de planejadores que fazem amostragem local (e.g. ARW), a dificuldade ocorre principalmente em conseguir gerar uma amostra na entrada do corredor e, a partir desta amostra, gerar outra amostra dentro do mesmo (e então a visibilidade tem uma grande influência no sucesso do planejador em gerar amostras em passagens estreitas).

O labirinto mostrado na Figura A.1(e) é um ambiente que visa impor aos planejadores uma situação na qual existem somente longos corredores e passagens estreitas interconectando-os. Dentre os três ambientes, é o mais difícil.

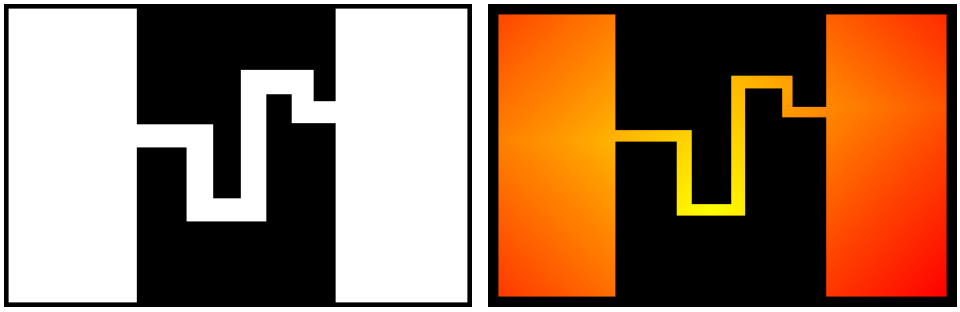
Os espaços de trabalho da Figura A.2 são utilizados para testes comparativos entre os planejadores de questionamento único, assim como para testes comparativos entre os planejadores de múltiplos questionamentos. As Figuras A.2(a) e A.2(c) mostram mapas de ambientes reais gerados pelo robô Omni [58]. A binarização destas duas grades de ocupação, conforme descrita na Seção 2.3.3, é feita de maneira conservadora. Assim, o menor valor correspondente à probabilidade de ocupação é o suficiente para considerar a célula como ocupada. Assim, o ambiente A.2(c) é particularmente difícil, pois além de ser não estruturado, as passagens são bem estreitas, o que é acentuado pelo *bitmap* conservador usado para a detecção de colisão. Já a Figura A.2(e) representa o Laboratório de Robótica e Automação em sua configuração atual e os ambientes adjacentes.

A Figura A.3 mostra, para todos os mapas de teste, os pontos inicial e final das quatro rotas requisitadas na avaliação dos planejadores de múltiplos questionamentos.



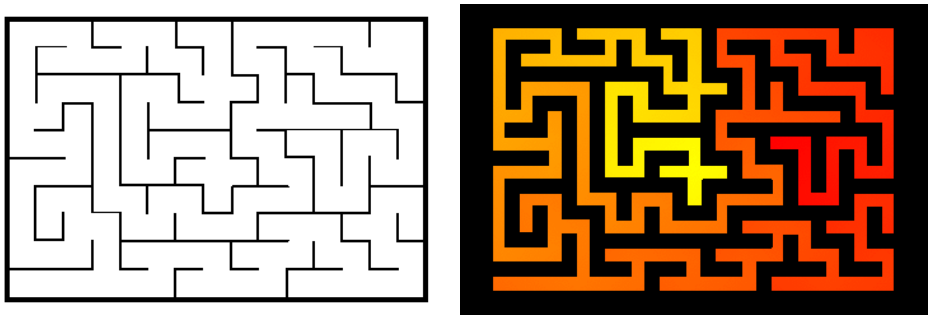
(a) Ambiente simples.

(b) Espaço de configuração e a alcançabilidade partindo do ponto central.



(c) Corredor.

(d) Alcançabilidade a partir do ponto central no espaço de configuração.

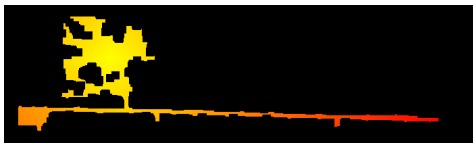


(e) Labirinto.

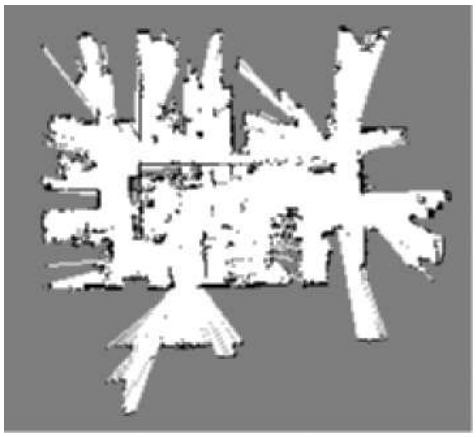
(f) Espaço de configuração.

Figura A.1: Primeiro grupo de ambientes de teste.

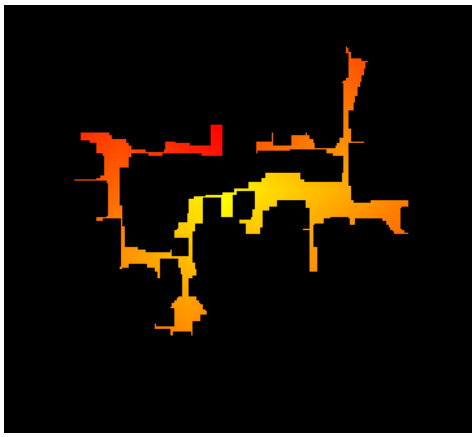
(a) Lirmm1.



(b) Alcançabilidade a partir do ponto central no espaço de configuração.

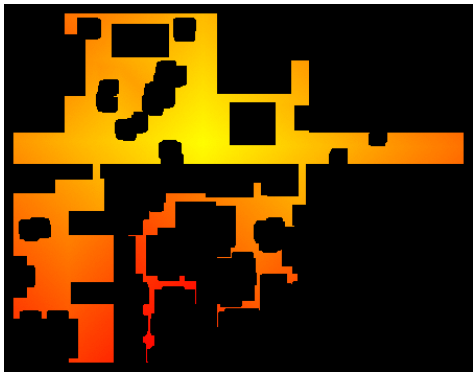


(c) Lirmm2.



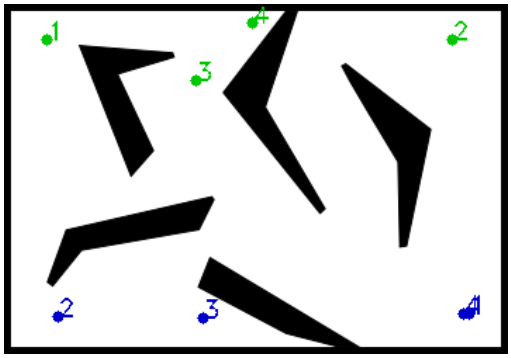
(d) Alcançabilidade a partir do ponto central no espaço de configuração.

(e) LARA.

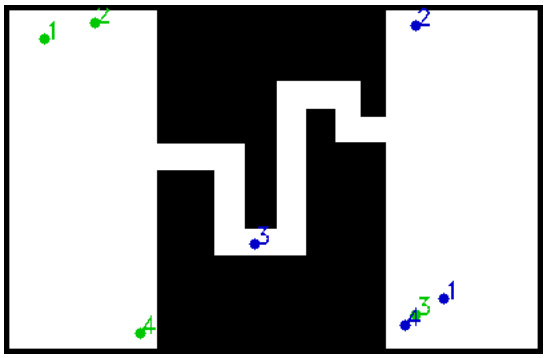


(f) Alcançabilidade a partir do ponto central no espaço de configuração.

Figura A.2: Segundo grupo de ambientes de teste.

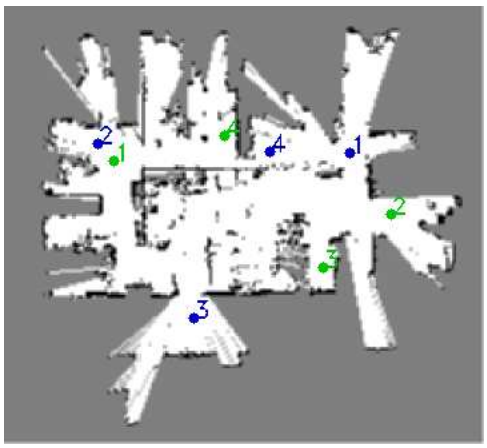


(a) Ambiente Simples

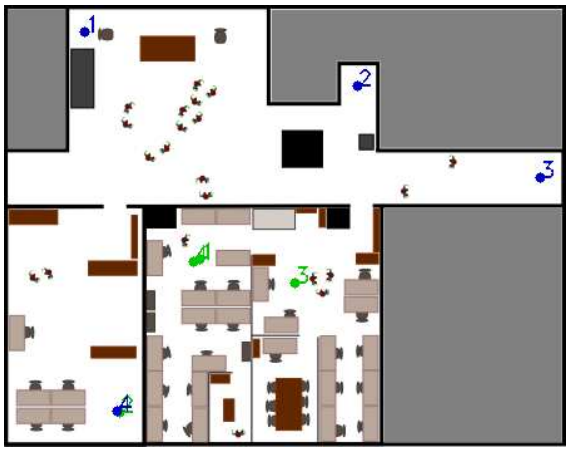


(b) Corredor.

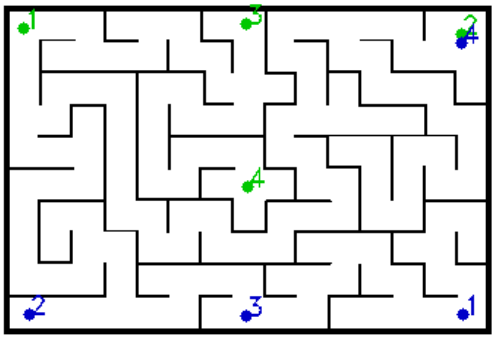
(c) LIRMM 1.



(d) LIRMM 2.



(e) LARA.



(f) Labirinto.

Figura A.3: Mapas com os pontos inicial (tons claros) e final (tons escuros) referentes às quatro rotas requisitadas pelos algoritmos de múltiplos questionamentos.

B.1 INTRODUÇÃO

Nesta Seção são apresentados conceitos básicos sobre probabilidade. O objetivo não é discutir exaustivamente cada tópico apresentado, tampouco oferecer provas de teoremas ou exemplos. Assim, sua principal função é fazer uma breve revisão dos tópicos que podem ser úteis para a compreensão deste trabalho. Papoulis [59] apresenta uma abordagem completa sobre o assunto e foi a principal referência para esta Seção.

Definição B.1.1. *A definição axiomática de probabilidade é dada por três postulados:*

I *A probabilidade $Pr\{A\}$ de um evento A é um número não negativo associado a esse evento:*

$$Pr(A) \geq 0;$$

II *Sendo Ω o conjunto de todos os possíveis eventos,*

$$Pr(\Omega) = 1,$$

em que Ω é conhecido como espaço amostral;

III *Se os eventos A e B são mutuamente exclusivos, então*

$$Pr(A \cup B) = Pr(A) + Pr(B).$$

III.a *(Axioma da aditividade infinita) Se os eventos A_1, A_2, ... são mutuamente exclusivos, então*

$$Pr(A_1 \cup A_2 \cup \ldots) = Pr(A_1) + Pr(A_2) + \ldots \qquad \square$$

Seja $\alpha(x)$ uma função tal que

$$\int_{-\infty}^{\infty} \alpha(x)dx = 1,\ \alpha(x) \geq 0. \tag{B.1}$$

A probabilidade do evento $x \leq x_i$ é dada por

$$Pr\{x \leq x_i\} = \int_{-\infty}^{x_i} \alpha(x)dx. \tag{B.2}$$

B.2 PROBABILIDADE CONDICIONAL

A probabilidade de ocorrer um evento A assumindo que o evento B já ocorreu (e então $P(B) > 0$) é dada por

$$Pr(A|B) = \frac{Pr(AB)}{Pr(B)}, \tag{B.3}$$

em que $Pr(AB) = Pr(A \cap B)$.

B.2.1 Independência

Dois eventos A e B são chamados independentes se $Pr(AB) = Pr(A) \cdot Pr(B)$. Então

$$Pr(A|B) = \frac{Pr(A) \cdot Pr(B)}{Pr(B)} \Rightarrow Pr(A|B) = Pr(A). \tag{B.4}$$

B.3 VARIÁVEIS ALEATÓRIAS

Dada a variável aleatória X, sua função de distribuição acumulada é dada por

$$F_X(x) = P\{X \leq x\}, \text{ para todo } x \in [-\infty, \infty], \tag{B.5}$$

e $F_X(\infty) = 1$ e $F_X(-\infty) = 0$

A função densidade de probabilidade é

$$p_X \triangleq \frac{dF_X(x)}{dx}, \tag{B.6}$$

então

$$F_X(x) = \int_{-\infty}^{x} p_X(u)du. \tag{B.7}$$

Como $F_X(\infty) = 1$, então

$$\int_{-\infty}^{\infty} p_X(x)dx = 1 \tag{B.8}$$

e

$$P\{x_1 < X \leq x_2\} = F_X(x_2) - F_X(x_1) = \int_{x_1}^{x_2} p_X(x)dx. \tag{B.9}$$

B.3.1 Distribuição condicional de uma variável aleatória

Dado o evento M, a distribuição condicional $F_X(x|M) = P\{X \leq x|M\} = \frac{P\{X \leq x, M\}}{P(M)}$, em que $\{X \leq x, M\}$ é a interseção dos eventos $\{X \leq x\}$ e M, ou seja, o evento consistindo de todas as realizações ξ tal que $X(\xi) \leq x$ e $\xi \in M$. Segue que

$$p_X(x|M) = \frac{dF_X(x|M)}{dx}. \tag{B.10}$$

B.3.2 Valor esperado e variância

O valor esperado, ou média, de uma variável aleatória X é dado por

$$E\{X\} = \int_{-\infty}^{\infty} x p_X(x) dx. \tag{B.11}$$

Sendo $E(X) = \mu$, a variância de X é dada por

$$\begin{aligned} \sigma_x^2 &= var(x) \\ &= E\{(x-\mu)^2\}. \end{aligned} \tag{B.12}$$

B.4 DUAS VARIÁVEIS ALEATÓRIAS

Sendo duas variáveis aleatórias X e Y, a distribuição conjunta $F_{XY}(x, y)$ é a probabilidade do evento $\{X \leq x, Y \leq y\}$.

B.4.1 Densidade conjunta

A densidade conjunta de X e Y é, por definição, a função

$$p_{XY} = \frac{\partial^2 F_{XY}(x, y)}{\partial x \partial y}. \tag{B.13}$$

Então

$$F_{XY}(x, y) = \int_{-\infty}^{x} \int_{-\infty}^{y} p_{XY}(u, v) du dv \tag{B.14}$$

B.4.2 Distribuição e densidade marginais

$F_X(x)$ e $p_X(x)$ são a distribuição e densidade marginais de X, respectivamente. As marginais de X determina suas estatísticas, mas não a estatística conjunta de X e Y.

Assim, as estatísticas de X e Y em termos da distribuição conjunta $F_{XY}(x, y)$ são dadas por

$$F_X(x) = F_{XY}(x, \infty), \tag{B.15}$$

$$F_Y(y) = F_{XY}(\infty, y). \tag{B.16}$$

Já as densidades marginais são dadas por

$$p_X(x) = \int_{-\infty}^{\infty} p_{XY}(x, y) dy, \tag{B.17}$$

$$p_Y(y) = \int_{-\infty}^{\infty} p_{XY}(x, y) dx \tag{B.18}$$

B.4.3 Distribuição condicional

Dado o evento M,

$$\begin{aligned} F_{XY}(x,y|M) &= Pr\{X \leq x, Y \leq y|M\} \\ &= \frac{Pr\{X \leq x, Y \leq y, M\}}{P\{M\}}. \end{aligned} \tag{B.19}$$

- Para $M = \{X \leq x\}$,

$$F_Y(y|X \leq x) = \frac{Pr\{X \leq x, Y \leq y\}}{Pr\{X \leq x\}} = \frac{F_{XY}(x,y)}{F_X(x)} \tag{B.20}$$

e

$$p_Y(y|X \leq x) = \frac{1}{F_X(x)} \frac{\partial F_{XY}(x,y)}{\partial y}. \tag{B.21}$$

- Para $M = \{X = x\}$,

$$p_Y(y|X = x) = \frac{p_{XY}(x,y)}{p_X(x)} = p_{Y|X}(y|x) \tag{B.22}$$

Como $p_{X|Y} = \frac{p_{XY}(x,y)}{p_Y(y)}$,

$$p_{X|Y}(x|y) = \frac{p_{Y|X}(y|x)p_X(x)}{p_Y(y)}. \tag{B.23}$$

B.4.4 Covariância

Para $\eta_x = E\{X\}$ e $\eta_Y = E\{Y\}$, a covariância σ_{XY} entre duas variáveis aleatórias X e Y é dada por

$$\sigma_{XY} = E\{(x - \eta_x)(x - \eta_y)\}. \tag{B.24}$$