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

Navegação de Robôs Móveis baseada na Equação de Laplace: Uma nova abordagem utilizando Elementos Finitos

Luciano Cunha de Araújo Pimenta

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

Orientador: Prof. Guilherme Augusto Silva Pereira

Co-orientador: Prof. Renato Cardoso Mesquita

Belo Horizonte, fevereiro de 2005

Resumo

Este trabalho aborda o problema de navegação de robôs móveis. Mais especificamente, é proposta uma nova abordagem, no contexto de robótica, para a solução da equação Laplace visando a construção de funções de navegação. Esta nova abordagem consiste na aplicação do Método de Elementos Finitos, o que permite o tratamento de obstáculos e robôs de formatos complexos. O trabalho ainda propõe regras para a definição de condições de contorno para a solução da equação de Laplace, as quais tornam a metodologia proposta *completa*, isto é, caso exista um caminho possível, o robô sempre atinge o alvo num tempo finito, independentemente da posição e orientação iniciais. Uma nova condição de contorno, dentro do contexto de robótica, chamada Condição de Contorno Periódica, também é proposta neste trabalho, permitindo um tratamento fechado da orientação do robô. O tratamento da orientação do robô passa também pela construção de espaços de configurações em $\mathbb{R}^3$, utilizados quando a orientação de robôs navegando no plano é considerada. Esta dissertação propõe um novo algoritmo para uma construção aproximada desses espaços. Os resultados do trabalho são validados numa plataforma constituída de robôs holonômicos reais.

Abstract

This work addresses the mobile robot navigation problem. More specifically, we propose a novel approach, in the robotics context, for constructing navigation functions based on the Laplace's equation solution. This approach is based on Finite Elements Methods, which allows for complex shaped obstacles and robots. Also, we propose rules for attaching boundary conditions to the boundary domain, in order to solve the Laplace's Equation, thus guaranteeing completeness for the proposed methodology, *i.e.*, if a path exists the robot always reach the goal in a finite time, independently of its initial position and orientation. A new boundary condition, called Periodic Condition, is proposed and used to take into account the robot's orientation. Additionally, we propose an algorithm for constructing configurations spaces in $\mathbb{R}^3$, useful when three degrees of freedom, planar robots are considered. Our methodology is validated in actual, holonomic mobile robots.

Agradecimentos

Gostaria de agradecer primeiramente aos professores Guilherme Pereira e Renato Mesquita pela excelente orientação, disponibilidade, boa vontade, transmissão de conhecimento e amizade. Agradeço também aos professores Walmir Caminhas e Mário Campos, que no início deste trabalho foram meus orientadores, pela disponibilidade, pelo apoio e por todo o conhecimento que assimilei durante suas disciplinas.

Agradeço também à toda minha família pelo apoio. Agradeço especialmente à Carina pelo amor, carinho, compreensão e companheirismo. Gostaria de agradecer especialmente também aos meus pais, meu irmão e minha avó que sempre estiveram do meu lado e me deram todo o suporte necessário, amor e carinho. Agradeço ainda à família da Carina pelo apoio e amizade.

Queria agradecer também aos amigos e colegas que participaram desta trajetória. Aos amigos do GOPAC, do VERLAB, do GEP e do CPDEE por propiciarem ótimos ambientes de trabalho e pela amizade. Agradeço especialmente ao amigo Alexandre Fonseca pela ajuda fundamental nas implementações computacionais desenvolvidas neste trabalho, pelos momentos de descontração e pela amizade.

Devo agradecer ainda aos demais professores que me apoiaram e me transmitiram conhecimentos fundamentais: Elson Silva, Paulo Seixas, Porfírio Cortizo, Pedro Donoso, Marcos Severo, Antônio Braga e Fábio Jota.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

> Tudo que uma pessoa pode imaginar, outras podem tornar real.
>
> *Júlio Verne (1828–1905)*

Ao longo dos anos, *robôs autônomos* têm atraído a atenção de vários cientistas. O termo *robô* foi introduzido por Karel Čapek em sua peça R.U.R ("Rossum's Universal Robots") em 1923. Nas línguas tcheca ou polonesa, a palavra *robota* significa "trabalho", e *robotnik* significa "trabalhador". Por sua vez, o termo *autônomo* tem origem no grego, onde "auto+matos" significa *desejo próprio*. Em [Dudek and Jenkin, 2000], observa-se que o termo *robô autônomo* é um tanto quanto contraditório, uma vez que automato implica em um grau de vontade própria, a qual não é prevista na palavra robô. No contexto atual de robótica, sistemas robóticos são denominados autônomos se eles forem capazes de realizar suas tarefas sem intervenção humana e tomar suas próprias decisões diante de situações inesperadas. Devido à complexidade de tais sistemas, os pesquisadores que se propõem a estudá-los são provenientes das mais diversas áreas: engenharia, computação, matemática, física, biologia, etc. Esse grande interesse é motivado pela vasta gama de aplicações onde estas máquinas podem ser utilizadas. Alguns exemplos de aplicações são: exploração de ambientes inóspitos, desarmamento de minas terrestres, busca e resgate de sobreviventes em desastres, auxílio a pes-

soas idosas ou deficientes em atividades do cotidiano, realização de tarefas domésticas, etc.

Apesar de todo o desenvolvimento tecnológico observado nos últimos anos, são poucos os robôs autônomos encontrados fora dos ambientes de pesquisa ou de ficção científica. Os robôs que se observam inseridos na sociedade atual são em sua maioria robôs manipuladores industriais não-autônomos. A maior parte destes robôs consiste em máquinas programadas para seguir trajetórias fixas no espaço. Aproximar-se destas máquinas, quando as mesmas estão em operação, constitui um grande risco, uma vez que elas não têm a capacidade de elaborar trajetórias alternativas para evitar uma eventual colisão. Observa-se na natureza, que qualquer animal ou organismo inteligente possui a habilidade de se locomover evitando colisões. Assim, o primeiro passo para que os robôs possam se tornar autônomos e atingir, de fato, a sociedade, é o desenvolvimento de técnicas eficientes e robustas para a locomoção autônoma destas máquinas.

Esta dissertação se insere no contexto de robótica e propõe uma nova metodologia para a navegação de robôs em ambientes complexos. Entendem-se por complexos ambientes que apresentam grande quantidade de obstáculos, os quais estão dispostos neste ambiente em posições e orientações quaisquer, e além disso possuem formas geométricas, também, quaisquer.

1.1 Motivação

Uma aplicação em especial onde robôs autônomos podem ser utilizados é o salvamento de vítimas de desastres. Locais de desastre, normalmente, representam ambientes de alto risco para equipes de salvamento. O risco é devido à possibilidade de novos desastres similares ao primeiro (o que é

Figura 1.1: Foto de destruição causada pela onda Tsunami no sudeste da Ásia [von Feldt, 2004].

freqüente no caso de terremotos e deslizamentos de terra) ou à possibilidade de acidentes provocados pelas conseqüências do desastre, como, por exemplo, incêndios, desabamentos, etc. Além disso, o acesso a possíveis locais onde haja sobreviventes é extremamente difícil. Por sua vez, sabe-se que a chance de se encontrar sobreviventes diminui a cada hora após o desastre. Dessa forma, robôs autônomos, e que sejam capazes de navegar em ambientes complexos, são uma ótima opção para executar buscas por sobreviventes.

No dia 26 de dezembro de 2004, foi noticiado mais um grande desastre da história mundial, o qual atingiu diversos países do sudeste asiático. Uma onda Tsunami se formou no oceano Índico e causou a morte de mais de 150.000 pessoas, distribuídas em mais de cinco países (veja Figura 1.1). Um outro exemplo de desastre recente é o terremoto que arrasou boa parte da cidade de Kobe, no Japão, em 1995 causando mais de 6.400 mortes. Obviamente, os robôs não são soluções para evitar os desastres, mas algumas dessas vidas poderiam ter sido salvas se a busca por sobreviventes tivesse sido mais eficiente.

Desastres têm ocorrido em todo o planeta, e em diversas proporções ao

Figura 1.2: Foto de um robô durante competição na RoboCup Rescue League [Tadokoro, 2004].

longo da história. O Brasil não está imune a estas situações. Em março de 2004, inúmeras residências foram devastadas no litoral sul de Santa Catarina e litoral norte do Rio Grande do Sul, pelo, então, batizado Ciclone Catarina. Isto é uma forte justificativa para o desenvolvimento de pesquisa na área de robótica no país.

Inúmeras iniciativas têm sido tomadas pela comunidade de robótica. Um bom exemplo é a competição "RoboCup Rescue League", onde diversos times competem em arenas que simulam desastres, e vence o time cujos robôs encontrarem as vítimas em menor tempo. A Figura 1.2 apresenta uma foto de um robô durante a competição.

Uma arquitetura robótica que possa ser utilizada num ambiente real de desastre deve ser complexa. Esta arquitetura deve permitir a execução de diversas tarefas como: planejamento de trajetórias em ambientes entulhados, desvio de obstáculos dinâmicos e não modelados, localização, mapeamento, etc. Esta dissertação endereça o problema de navegação, que deve ser resolvido na execução da primeira tarefa citada. Supondo que se possua o mapa do ambiente onde o robô deve navegar, que este ambiente seja estático, e que se tenha uma dada posição alvo do espaço para onde o robô deva se deslocar,

pode-se definir o seguinte problema de navegação:

Definição 1.1 (Problema de Navegação) *Seja um robô de formato genérico e um mapa do ambiente com objetos estáticos, também, de formatos genéricos, leve o robô para uma posição final (alvo) sem colidir.*

1.2 Objetivos

O presente trabalho tem o objetivo de propor uma metodologia para resolver o problema de navegação de robôs móveis atendendo os seguintes requisitos:

- Por razões óbvias deseja-se a garantia de que num intervalo de tempo finito, desde que exista um caminho possível, o alvo sempre seja atingido;
- Para que o método seja robusto a pequenos erros de localização e não haja necessidade de replanejamento, deve-se obter como solução do problema, caminhos para se chegar ao alvo a partir de todos os pontos do ambiente;
- Para que a metodologia seja simples e geral deseja-se levar em conta a orientação do robô numa forma fechada.

1.3 Metodologia e Contribuições

A metodologia proposta neste trabalho consiste no uso de funções de navegação calculadas numericamente pelo método de elementos finitos. São utilizadas soluções da equação de Laplace, denominadas funções harmônicas,

como funções de navegação para controlar o robô em seu espaço de configurações. O método de elementos finitos é introduzido, então, para resolver numericamente esta equação. As principais contribuições deste trabalho são:

- Tratamento eficiente de contornos com geometrias genéricas por meio da introdução do método de elementos finitos (Capítulo 5);
- Definição de critérios para a definição de condições de contorno para a equação de Laplace, de forma a tornar a metodologia proposta *completa* (Capítulo 4);
- Tratamento da orientação do robô numa forma fechada através da imposição de condições de contorno periódicas e da utilização de uma única lei de controle para as velocidades de translação e rotação do robô (Capítulo 4);
- Um algoritmo para a construção de espaços de configurações tridimensionais (Capítulo 3).

1.4 Organização da dissertação

O Capítulo 1 apresentou a motivação, os objetivos, a metodologia e as contribuições desta dissertação. No Capítulo 2 é feita uma revisão bibliográfica com o objetivo de familiarizar o leitor com o problema tratado nesta dissertação. Além disso, são discutidos trabalhos que procuram resolver o problema de navegação das mais diversas formas, com ênfase naqueles que mais se aproximam da metodologia proposta, que se baseiam em *Funções de Navegação*. No Capítulo 3 são apresentados algoritmos eficientes para construção de espaços de configurações bidimensionais e tridimensionais para

robôs móveis de formas genéricas com dois e três graus de liberdade, respectivamente. No Capítulo 4 são apresentados a modelagem matemática e os aspectos de controle de uma solução baseada na equação de Laplace para o problema de navegação. No Capítulo 5 é apresentado o Método de Elementos Finitos baseado no método de Galerkin, que foi o método utilizado neste trabalho para gerar a solução numérica da equação de Laplace e que permite o tratamento de obstáculos e robôs de geometrias complexas. No Capítulo 6 são apresentados exemplos ilustrativos, onde aplicou-se o método proposto em robôs reais e em simulações, afim de mostrar o funcionamento e a praticabilidade do método. O Capítulo 7 apresenta as conclusões deste trabalho enfatizando as vantagens e limitações da metodologia proposta e os possíveis trabalhos futuros. O Apêndice A mostra detalhes da plataforma robótica utilizada para validar a metodologia proposta.

Capítulo 2

Revisão Bibliográfica

> É preferível conhecer alguma coisa sobre tudo do que tudo sobre apenas alguma coisa.
>
> *Blaise Pascal (1623–1662)*

Este capítulo dedica-se a familiarizar o leitor com o problema a ser tratado nesta dissertação, que é o problema de navegação robótica. Esta familiarização é feita por meio de uma revisão bibliográfica do assunto tratado e do estabelecimento de relações com a metodologia proposta na dissertação. São discutidos trabalhos que procuram resolver o problema de navegação das mais diversas formas, com ênfase naqueles que mais se aproximam da metodologia proposta, que se baseiam em *Funções de Navegação*. Uma boa referência, que procura mostrar uma grande variedade de métodos para resolver o problema de navegação é [Latombe, 1991].

Tradicionalmente, o primeiro passo para resolver o problema de navegação robótica é transformar o problema do *espaço de trabalho* para um problema no *espaço de configurações* do robô. O espaço de trabalho $\mathcal{W}$ de um robô R é a região do ambiente que pode ser atingida pelo robô. No caso de um robô móvel navegando em um plano, por exemplo, seu espaço de trabalho tem duas dimensões e é constituído de todo o plano (x, y). Além disso, o espaço de trabalho pode conter um conjunto de obstáculos $\mathcal{O} = \{P_1, \ldots, P_n\}$.

Por sua vez, proposto em [Lozano-Pérez and Wesley, 1979], o espaço de configurações constitui um formalismo matemático elegante que visa representar o robô e seu espaço de trabalho, de forma a simplificar o problema de navegação. Entende-se por *configuração*, q, de um robô R, o conjunto de estados ou parâmetros independentes do robô que permite definir completamente a sua localização no ambiente em relação a um dado sistema de coordenadas. O Espaço de Configurações, $\mathcal{C}$, portanto, é o conjunto de todas as configurações possíveis do robô. Neste espaço $\mathcal{C}$ o robô é representado como um ponto. O número de dimensões do espaço de configurações corresponde ao número de *graus de liberdade* do robô. Para um robô manipulador com duas juntas de revolução, por exemplo, tem-se um espaço de configurações bidimensional, $\mathcal{C} \subset \mathbb{R}^2$, cujas dimensões são os ângulos de suas juntas (θ_1, θ_2).

No caso de robôs móveis navegando no espaço tridimensional, define-se um ponto de referência G fixo ao robô e o espaço de configurações tem 6 dimensões $(x, y, z, \alpha, \beta, \theta)$. As três primeiras dimensões correspondem às coordenadas cartesianas do ponto de referência, G, as outras três são os ângulos *Roll*, *Pitch* e *Yaw*, respectivamente. Esses ângulos são bem conhecidos nas áreas náutica, aeronáutica e veicular, e definem rotações de um corpo em torno dos eixos x,y e z, respectivamente.

Para um robô que se move em um plano, tem-se um espaço de configurações em $\mathbb{R}^2$ (x, y) ou em $\mathbb{R}^3$ (x, y, θ), onde θ é o ângulo de Yaw do robô, o que corresponde à sua orientação. O espaço de configurações em $\mathbb{R}^2$, na verdade, é uma simplificação, e pode ser usado sem perda de informação, na representação do robô, se este tiver um formato circular ou se ao mesmo é permitido apenas transladar em seu espaço de trabalho. Se o robô tiver uma forma não-circular e tiver a possibilidade de executar rotações em

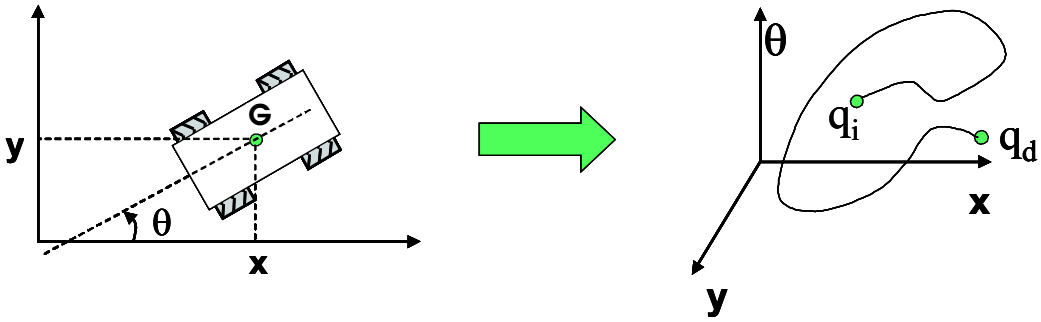

Figura 2.1: Representação e trajetória de um robô móvel em seu espaço de configurações.

torno de seu próprio eixo, então necessariamente tem-se um espaço em $\mathbb{R}^3$. A Figura 2.1 mostra a representação de um robô móvel em seu espaço de configurações e também a trajetória seguida neste espaço a partir de uma configuração inicial q_i até uma configuração final desejada ou alvo q_d. Neste contexto, entende-se por trajetória uma seqüência contínua de configurações em $\mathcal{C}$.

Nem todas as regiões do espaço de configurações são permitidas ao robô. Essas regiões são ditas proibidas, pois correspondem às áreas onde o robô intercepta um dos elementos do conjunto de obstáculos $\mathcal{O}$ no espaço de trabalho, ou seja, existe uma colisão. O conjunto dessas regiões proibidas, o qual constitui um mapeamento dos obstáculos do espaço de trabalho para o espaço de configurações, é chamado de *C-obstáculo*, e é representado por $\mathcal{C}_{obst}$. Por sua vez, o *espaço de configurações livre* $\mathcal{F}$ é aquele constituído de todos os pontos do espaço de configurações que restam, ao se retirar os pontos correspondentes ao C-obstáculo:

$$\mathcal{F} = \mathcal{C} \setminus \mathcal{C}_{obst}\,, \tag{2.1}$$

onde $\setminus$ é o operador de subtração de conjuntos.

Como se observa, no espaço de configurações o robô pode ser tratado como um ponto. Conclui-se, também, que se o robô se mover apenas em pontos de seu espaço de configurações livre $\mathcal{F}$ não há colisões com obstáculos de seu espaço de trabalho $\mathcal{W}$. O problema de navegação pode, então, ser traduzido para o seguinte problema no espaço de configurações:

Definição 2.1 (Problema de Navegação em $\mathcal{C}$) *Seja um robô R representado em seu espaço de configurações $\mathcal{C}$, e com configuração inicial $q_i \in \mathcal{F}$ no tempo $t = t_0$. Leve R até a configuração desejada (alvo) $q_d \in \mathcal{F}$ em algum tempo finito $t = t_f > t_0$, tal que $q \in \mathcal{F}$ $\forall t \in (t_0, t_f]$.*

Pela maior simplicidade de se planejar trajetórias para robôs puntuais, a maior parte das soluções propostas na literatura - inclusive a solução proposta nesta dissertação - trabalha no espaço de configurações. Além da simplicidade, um outro ponto interessante é que ao se elaborar algoritmos para o planejamento e controle de robôs puntuais num espaço de configurações $\mathbb{R}^n$, sendo n o número de graus de liberdade do robô, e que sejam independentes de n, esses algoritmos tornam-se genéricos para qualquer tipo de robô, seja ele móvel ou fixo. O algoritmo funcionará corretamente desde que o espaço de configurações tenha sido construído corretamente para o robô que se deseja utilizar. A construção de espaços de configurações é tratada no Capítulo 3 desta dissertação.

Após construir-se o espaço de configurações do robô, tipicamente, como é dito em [Rimon and Koditschek, 1992], o problema de navegação pode ser decomposto em três etapas:

1. Planejamento do caminho: conhecendo-se a geometria do espaço e o destino desejado, um caminho livre de colisão conectando as configurações iniciais e finais do robô é construído no espaço de configurações

livre, $\mathcal{F}$ [Latombe, 1991]. Nesta etapa a dinâmica do robô é deixada de lado;

2. Planejamento da trajetória: dado um caminho no espaço $\mathcal{F}$ resolve-se o problema da cinemática inversa do robô de forma a obter-se uma trajetória no espaço de estados, que inclui as derivadas temporais $\dot{q}$ e $\ddot{q}$, a ser seguida para percorrer o dado caminho;

3. Controle: mediante as variáveis de controle do robô, tenta-se fazer com que suas variáveis de estado sigam a trajetória gerada anteriormente.

Visto de um ângulo mais macroscópico, o problema de navegação é abordado via arquiteturas de controle [Arkin, 1998]:

1. Deliberativa: essa arquitetura segue o paradigma SPA (*sense, plan, act*), o que significa que um modelo do mundo é mantido pelo sistema, por meio de conhecimento prévio ou sensoreamento, e um planejamento é feito. Nessa arquitetura o problema de navegação é expresso exatamente pelas três etapas citadas anteriormente.

2. Reativa: o paradigma para essa arquitetura é chamado SA (*sense, act*). Não há planejamento e o robô apenas reage de forma direta aos estímulos vindos de seus sensores. Esse paradigma produz bons resultados quanto ao desvio de obstáculos mas não garante que o robô atinja o alvo. Uma vez que não necessita planejamento, a atuação do robô é realizada de forma rápida.

3. Híbrida: essa categoria engloba arquiteturas que tiram proveito das vantagens das duas apresentadas acima. De forma geral, um planejador global (camada deliberativa) é definido. Este planejador executa então

o papel da arquitetura deliberativa e realiza o planejamento de caminho. Uma camada inferior de planejamento local, com características mais reativas, é utilizada para simultaneamente seguir o caminho gerado pela camada superior e desviar de obstáculos não modelados.

Nas arquiteturas deliberativas tradicionais, o planejamento de caminhos utiliza algoritmos de buscas em grafos. Neste contexto, é gerado um grafo, onde os espaços livres são vértices, e as arestas indicam caminhos livres de colisão entre estes espaços. Após a geração do grafo, busca-se o caminho de menor custo até o alvo. Alguns exemplos de abordagens baseadas em grafos são: "Grafos de Visibilidade" [Lozano-Pérez and Wesley, 1979], "Cones Generalizados" [Brooks, 1983], "Diagrama de Voronoi" [Donald, 1984], "Subdivisão do espaço livre" [R.A.Brooks and Pérez, 1985], "*Roadmaps*" [Latombe, 1991] e "*Probabilistic Roadmaps*" [Kavraki et al., 1996].

A grande vantagem de abordagens que utilizam grafos, como dito anteriormente, é o fato de se procurar soluções com menor custo para se atingir o alvo. Isto pode, por exemplo, implicar em economia de energia durante a navegação. Entretanto, o custo computacional pode se tornar elevado em espaços de configurações de maior dimensão e em situações inesperadas, onde se necessita de replanejamento. Além disso, nessas metodologias, ainda é necessário que se faça, após o planejamento de caminhos, o planejamento de trajetória e o controle do robô.

Uma metodologia que procura tratar o problema de navegação de forma integrada é a baseada em campos potenciais artificiais. Em [Khatib, 1980] foi proposta uma expressão para uma função de potencial ϕ e foi sugerido o uso do seu gradiente descendente $-\nabla\phi$ como uma entrada de torque para o sistema. O planejamento nesse caso é baseado em analogias físicas: o robô é tratado como uma partícula sofrendo a ação de um potencial ϕ, o

qual é gerado para representar o espaço livre $\mathcal{F}$. Tipicamente, o robô é modelado como uma carga positiva e os obstáculos também, enquanto o alvo tem carga negativa. A repulsão evita a colisão com obstáculos e a atração faz o robô aproximar-se do objetivo. O controle é integrado ao planejamento, uma vez que a informação da trajetória que deve ser seguida é embutida no próprio controlador. A grande vantagem desta integração entre planejamento e controle é o fato do planejamento se tornar mais robusto em relação a ruídos nos sensores e atuadores e, também, em relação a perturbações externas. Além disso, como os potenciais artificiais são definidos para todo o espaço de configurações livre, trajetórias a partir de qualquer ponto de $\mathcal{F}$ para o alvo são automaticamente determinadas, o que evita, portanto, replanejamento.

A utilização de potenciais artificiais para navegação foi popularizada na década de 80. O principal problema das implementações tradicionais, tais como [Khatib, 1986, Borenstein and Y.Koren, 1989, M.D.Adams et al., 1990, R.B.Tilove, 1990] é a presença de mínimos locais. Uma vez nesses pontos, o robô cessa a navegação sem ter atingido o alvo (mínimo global). Por isso, muitas vezes os potenciais artificiais são utilizados em arquiteturas híbridas, de forma a realizar apenas o planejamento local.

Alguns trabalhos, como [Krogh, 1984, Barraquand and Latombe, 1990], abordam o problema dos mínimos locais, mas não oferecem garantia de funcionamento correto. Em [Rimon and Koditschek, 1988] introduziu-se o conceito de *Funções de Navegação*, que são potenciais artificiais que garantem, para um robô com n-graus de liberdade, a presença de um único mínimo na função, e que está localizado justamente no alvo q_d. Uma técnica analítica de construção destes potenciais é apresentada em [Rimon and Koditschek, 1992]. Pode-se dividir esta técnica em três passos: (i) cálculo de um difeomorfismo, h, entre um espaço composto apenas por esferas, $\mathcal{E}$, e o espaço de confi-

gurações, $\mathcal{C}$, do robô; (ii) cálculo de uma função de navegação, φ, sobre o espaço $\mathcal{E}$ utilizando uma expressão fechada; (iii) cálculo de uma função de navegação, $\hat{\varphi}$, sobre o espaço $\mathcal{C}$ transformando φ através do difeomorfismo h. As desvantagens desta metodologia são: (i) a implementação do método é difícil mesmo para situações onde a geometria do ambiente é simples; (ii) o funcionamento correto do método para ambientes genéricos não é garantida; e (iii) informações geométricas perfeitas do mundo real devem ser conhecidas.

Nos trabalhos desenvolvidos por Connolly *et al.* [Connnolly et al., 1990], [Connnolly, 1992], e [Connolly and Grupen, 1993] propõe-se o uso de *funções harmônicas*, que são soluções da *equação de Laplace*, como funções de navegação[1]. Diferentemente de [Rimon and Koditschek, 1992], onde buscam-se soluções analíticas para o problema de navegação, Connolly *et al.* propõem a integração numérica da equação de Laplace. Esta integração foi feita por meio da discretização do domínio de solução em uma malha retangular regular e do uso do *Método de Diferenças Finitas* [Burden et al., 1978]. Essa metodologia é explorada para o planejamento e controle de manipuladores. Depois de Connolly *et al.*, diversos trabalhos utilizaram funções de potencial harmônicas como funções de navegação [Kim and Khosla, 1992, Guldner et al., 1997, Júnior, 2003, Waydo and Murray, 2003].

Muitos autores têm utilizado funções de navegação e suas variantes para controlar seus robôs [Brock and Khatib, 1999, Esposito and Kumar, 2002, Tanner et al., 2003, Pereira, 2003, Pereira et al., 2004a]. Outros autores propuseram novas formas de se calcular numericamente funções de navegação [Konolige, 2000, Valavanis et al., 2000, Wang and Chirikjian, 2000]. O principal problema destas abordagens é que elas utilizam discretizações em ma-

[1]Os autores não utilizaram o nome *Função de Navegação*, mas Rimon e Koditschek caracterizaram esta abordagem como tal [Rimon and Koditschek, 1992].

lhas estruturadas, tal qual a abordagem de Connolly *et al.*. Em acordo com [Shewchuk, 1998] malhas estruturadas impõem um custo computacional excessivo em situações de geometria complexa. Além disso, apesar da maioria destas abordagens serem utilizadas para controlar robôs móveis e manipuladores, apenas algumas delas levam em consideração a orientação do robô. Como foi dito em [Wang and Chirikjian, 2000], a maioria dos autores evita este problema por meio da expansão dos obstáculos pelo raio máximo do robô.

No presente trabalho, propõe-se uma metodologia deliberativa similar à de Connolly, isto é, utilizam-se também funções harmônicas como potenciais artificiais. A principal contribuição deste trabalho é a introdução do Método de Elementos Finitos [Hughes, 2000] no contexto de navegação de robôs móveis. O método de elementos finitos permite a utilização de malhas não-estruturadas, e portanto o tratamento de geometrias genéricas pode ser feito de forma muito mais eficiente do que nas abordagens anteriores. Além disso, diferentemente de outros trabalhos, a metodologia proposta leva em conta a orientação do robô em uma forma fechada, isto é, a orientação do robô não é tratada à parte no planejamento. Isto é feito por meio da construção correta do espaço de configurações do robô e da inclusão de condições de contorno periódicas [Ida and Bastos, 1992]. Logo, a metodologia proposta caracteriza-se como uma solução interessante e livre de mínimos locais para o problema de navegação de robôs em ambientes conhecidos a priori, onde tanto a geometria do robô quanto a geometria do ambiente são genéricas.

Capítulo 3

Construção do Espaço de Configurações

> O que sabemos é uma gota, o que não sabemos é um oceano.
>
> *Isaac Newton (1642–1727)*

Neste capítulo, discute-se a construção do espaço de configurações. Conforme já apresentado, o primeiro passo da maioria das abordagens para a solução do problema de navegação robótica é a construção do espaço de configurações. No Capítulo 2 definiu-se o espaço de configurações e mostrou-se que neste espaço o robô pode ser tratado como um ponto, e por isso o problema de navegação se torna mais simples e os algoritmos de solução se tornam genéricos. No presente capítulo, são apresentados algoritmos eficientes para mapear os obstáculos do espaço de trabalho para o espaço de configurações, por meio da construção do C-obstáculo. Embora a idéia de se utilizar a equação de Laplace e o método de elementos finitos seja independente da dimensão do espaço, a metodologia proposta é instanciada na solução do problema para robôs móveis terrestres em ambientes planares. Em virtude disso, são abordados, neste capítulo, dois tipos de espaços de configurações: em $\mathbb{R}^2$ (x, y) e em $\mathbb{R}^3$ (x, y, θ).

Antes de tratar dos algoritmos de construção dos espaços propriamente

ditos, apresenta-se uma operação matemática que é a base dos algoritmos em questão. Essa operação é a *Soma de Minkowski* [de Berg et al., 2000], que possibilita a construção de C-obstáculos a partir dos vértices dos obstáculos e do robô. Além de ser utilizada no planejamento de caminhos em robótica, a soma de Minkowski também é utilizada em diversas outras aplicações como: projeto e manufatura assistido por computador e planejamento de montagem [Flato, 2000].

3.1 Soma de Minkowski

A soma de Minkowski é uma operação matemática que pode ser aplicada na solução de diversos problemas. Nesta seção, apresentam-se os aspectos principais desta operação com enfoque no problema de construção do C-obstáculo. Maiores detalhes, outras aplicações e outras propriedades da soma de Minkowski podem ser encontrados em [de Berg et al., 2000, Benson, 1966, Chew and Kedem, 1993, Guibas et al., 1983].

A soma de Minkowski de dois conjuntos $\mathcal{A} \in \mathbb{R}^n$ e $\mathcal{B} \in \mathbb{R}^n$, denotada por $A \oplus B$, é definida como:

$$\mathcal{A} \oplus \mathcal{B} \equiv \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}, \tag{3.1}$$

onde $\mathbf{a} + \mathbf{b}$ denota a soma vetorial dos vetores $\mathbf{a} = (a_1, \ldots, a_n)$ e $\mathbf{b} = (b_1, \ldots, b_n)$, ou seja:

$$\mathbf{a} + \mathbf{b} \equiv (a_1 + b_1, \ldots, a_n + b_n). \tag{3.2}$$

A soma de Minkowski pode ser aplicada a polígonos, desde que se considere que um dado polígono A é um conjunto constituído de todos seus pontos

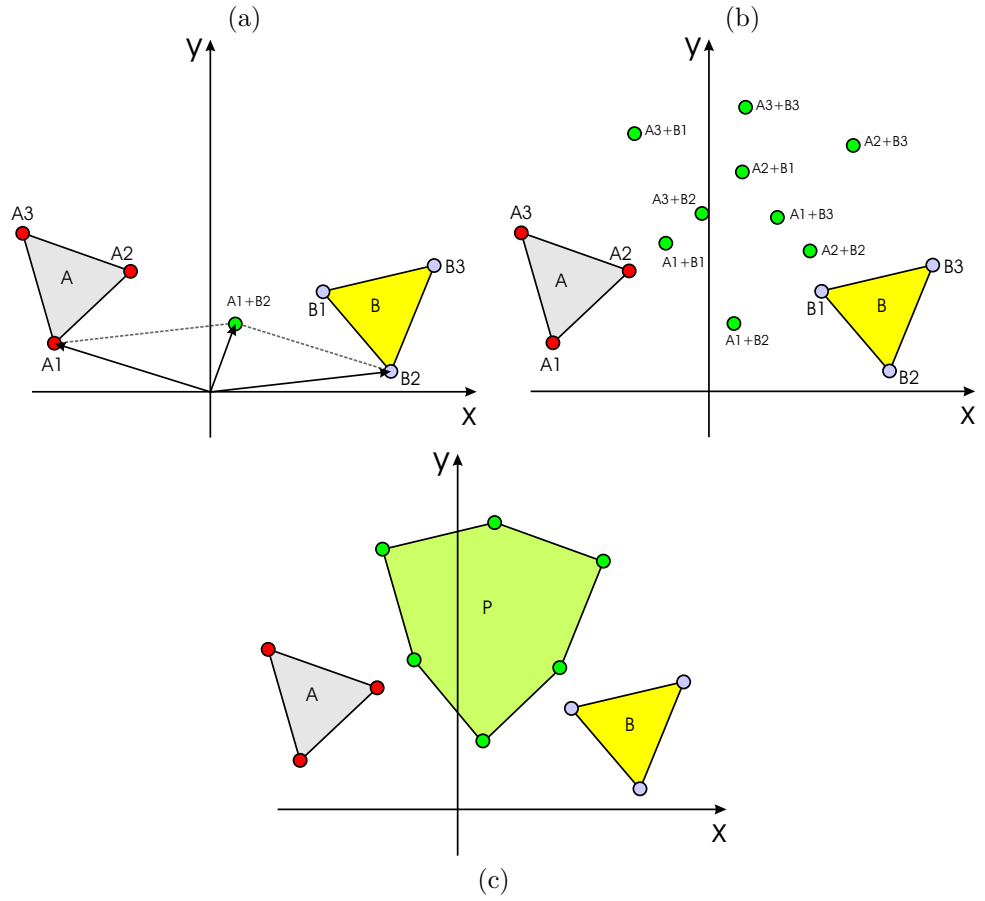

Figura 3.1: Soma de Minkowski de dois polígonos. (a) O ponto $A1 + B2$ é a soma vetorial dos vetores que definem os vértices $A1$ e $B2$. (b) Soma vetorial de todos os vetores que definem os vértices. (c) Polígono resultante da soma de Minkowski.

interiores e dos pontos localizados em sua fronteira. A Figura 3.1 ilustra a soma de Minkowski de dois triângulos. Conforme a definição expressa na Equação (3.1), a soma de Minkowski de dois polígonos A e B corresponde à soma vetorial de todos os pontos interiores e da fronteira de A com todos os pontos interiores e da fronteira de B. Pode ser feita a seguinte observação sobre os pontos extremos do polígono $A \oplus B$ [de Berg et al., 2000]:

Observação 3.1 *Sejam A e B dois polígonos no plano. Um ponto extremo numa dada direção $\overrightarrow{d}$ no polígono $A \oplus B$ é a soma dos pontos extremos, na*

direção $\overrightarrow{d}$*, de A e de B.*

Pela Observação 3.1 fica claro que cada vértice de $A \oplus B$ é o resultado da soma de um dos vértices de A com um dos vértices de B. Na Figura 3.1 (a) pode-se ver a soma vetorial de um par de vértices. As somas vetoriais de todos os vértices são apresentadas na Figura 3.1 (b). Na Figura 3.1 (c) apresenta-se a soma de Minkowski dos dois polígonos, e como esperado, cada vértice, extremo numa direção $\overrightarrow{d}$, deste polígono resultante é a soma vetorial de dois vértices que são os extremos, para a mesma direção $\overrightarrow{d}$, em cada um dos polígonos que estão sendo somados.

Afim de analisar a soma de Minkowski de dois polígonos quaisquer, vamos considerar, primeiramente, a soma entre dois polígonos convexos. Um teorema interessante sobre a soma de Minkowski de dois polígonos convexos é o seguinte [de Berg et al., 2000]:

Teorema 3.2 *Sejam A e B dois polígonos convexos com n e m arestas, respectivamente. Então a soma de Minkowski* $A \oplus B$ *é um polígono convexo com no máximo* $n + m$ *arestas.*

A prova do teorema 3.2 pode ser encontrada em [de Berg et al., 2000]. Com base na Observação 3.1 e no Teorema 3.2 pode-se utilizar o Algoritmo 3.1 para calcular a soma de Minkowski de dois polígonos convexos. A notação $\hat{A}ngulo(cd)$, presente no algoritmo, se refere ao ângulo formado pelo vetor $\overrightarrow{cd}$ com o eixo x-positivo. O algoritmo, através de uma varredura das direções no sentido anti-horário, procura calcular apenas as somas vetoriais que resultam em vértices da soma de Minkowski, isto é, apenas os vértices que são extremos numa mesma direção são somados. Pelo fato dos polígonos serem convexos, a comparação entre os ângulos, realizada no algoritmo, garante que todos os pares de vértices que são extremos numa mesma direção sejam encontrados.

algoritmo 3.1 Calcula a soma de Minkowski de dois polígonos convexos $A \oplus B$

Entradas: Um polígono convexo A com vértices $A_1, \ldots, A_n$ e um polígono convexo B com vértices $B_1, \ldots, B_m$, ambos no plano xy. Os vértices dos dois polígonos devem estar ordenados em sentido anti-horário, e A_1 e B_1 devem ser os vértices de menor coordenada y (e menor coordenada x em caso de empate).

Saídas: $P \leftarrow A \oplus B$

1: $i \leftarrow 1$; $j \leftarrow 1$
2: $A_{n+1} \leftarrow A_1$; $B_{m+1} \leftarrow B_1$
3: **enquanto** $i \neq n+1$ **e** $j \neq m+1$ **faça**
4: Faça $A_i + B_j$ ser um vértice de P
5: **Se** $\hat{A}ngulo(A_iA_{i+1}) < \hat{A}ngulo(B_jB_{j+1})$ **então**
6: $i \leftarrow i+1$
7: **senão**
8: **Se** $\hat{A}ngulo(A_iA_{i+1}) > \hat{A}ngulo(B_jB_{j+1})$ **então**
9: $j \leftarrow j+1$
10: **senão**
11: $i \leftarrow i+1$; $j \leftarrow j+1$
12: **fim Se**
13: **fim Se**
14: **fim enquanto**

Como o algoritmo passa por todos os n e os m vértices dos dois polígonos, e além disso, o algoritmo passa em cada vértice uma única vez, claramente, o algoritmo tem complexidade computacional linear $O(n+m)$ em relação ao tempo de cálculo.

Até o momento discutiu-se a soma de Minkowski entre polígonos convexos. Para generalizar esta operação para polígonos quaisquer deve-se resolver o problema em três etapas [Flato, 2000]:

1. Decomponha os polígonos A e B em sub-polígonos convexos $L_1, \ldots, L_s$ e $Q_1, \ldots, Q_t$, respectivamente;

2. Para cada $i \in [1, \ldots, s]$ e para cada $j \in [1, \ldots, t]$ calcule $P_{ij} = L_i \oplus Q_j$;

3. Calcule a união de todos os polígonos obtidos no passo anterior.

A decomposição de polígonos em sub-polígonos convexos pode ser feita por meio de *triangulação* dos polígonos [de Berg et al., 2000]. Formas alternativas para esta decomposição, baseadas em heurísticas ou otimização, inclusive mais eficientes, em termos de tempo de cálculo, que a triangulação, são apresentadas em [Flato, 2000]. O segundo passo na solução do problema pode ser realizado utilizando o Algoritmo 3.1. Por fim, para a união dos polígonos, existem basicamente três abordagens: algoritmo baseado em arranjos, algoritmo incremental e algoritmo dividir para conquistar. Uma discussão detalhada desses algoritmos foge ao escopo deste trabalho e é apresentada em [Flato, 2000].

Para finalizar esta seção, o seguinte teorema, cuja prova é apresentada em [de Berg et al., 2000], resume os aspectos de complexidade computacional para o cálculo da soma de Minkowski:

Teorema 3.3 *Sejam A e B dois polígonos quaisquer com n e m vértices, respectivamente. A complexidade computacional da soma de Minkowski $A \oplus B$ é limitada, no pior caso, em:*
(i) $O(n+m)$ se os dois polígonos forem convexos;
(ii) $O(nm)$ se um dos polígonos for convexo e o outro for não-convexo;
(iii) $O(n^2m^2)$ se ambos forem não-convexos.

3.2 Espaço de configurações em $\mathbb{R}^2$

Nesta seção, mostra-se como utilizar a soma de Minkowski, descrita na seção anterior, para calcular o espaço de configurações em $\mathbb{R}^2$ (x, y). Este espaço de configurações é utilizado para os casos onde não se necessita tratar

a orientação do robô. Estes casos são: robôs de formas circulares e robôs que não sofrem rotação.

Para se construir o espaço de configurações é necessário dilatar o contorno dos obstáculos do espaço de trabalho para as dimensões do robô, o que corresponde a construir o conjunto C-obstáculo. O seguinte teorema permite a dilatação de cada obstáculo do ambiente através da soma de Minkowski [de Berg et al., 2000]:

Teorema 3.4 *Seja um robô, $R(0,0)$, que executa apenas translação cujo ponto de referência no espaço de trabalho, que representará o robô no espaço de configurações, é o ponto (0,0). Seja também um obstáculo A no espaço de trabalho. Considere que $(-R(0,0))$ representa a reflexão de $R(0,0)$ em relação à origem. Então a soma de Minkowski $A \oplus (-R(0,0))$ corresponde ao C-obstáculo de A.*

Prova: Deve ser provado que $R(x,y)$, ou seja, o robô R com configuração (x,y), têm intersecção com o obstáculo A, se e somente se $(x,y) \in A \oplus (-R(0,0))$.

Suponha que A e $R(x,y)$ se interceptem e que $a = (a_x, a_y)$ seja um ponto desta intersecção. Se $a \in R(x,y)$ então $(a_x - x, a_y - y) \in R(0,0)$. Logo, $(-a_x + x, -a_y + y) \in -R(0,0)$. Como, também, $a \in A$, pela definição da Equação (3.1) tem-se que $(x,y) \in A \oplus (-R(0,0))$.

Para completar a prova, suponha agora que $(x,y) \in A \oplus (-R(0,0))$. Então existem pontos $(r_x, r_y) \in R(0,0)$ e pontos $(b_x, b_y) \in A$ tal que $(x,y) = (b_x - r_x, b_y - r_y)$, ou seja, $b_x = r_x + x$ e $b_y = r_y + y$, o que implica que $R(x,y)$ intercepta A. ∎

A Figura 3.2 ilustra o teorema 3.4. Na Figura 3.2 (a) mostra-se o resultado da operação $A \oplus (-R(0,0))$ para um obstáculo retangular e um robô triangular. Na Figura 3.2 (b) comprova-se que o resultado obtido pela operação

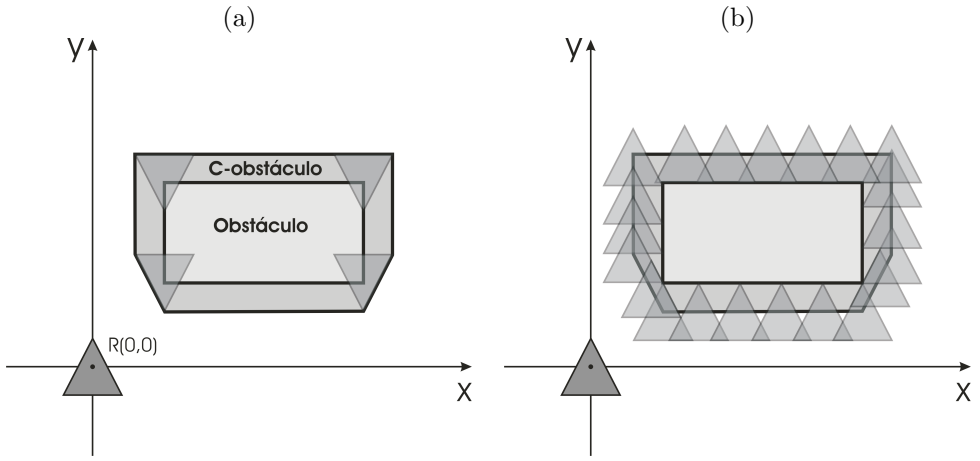

Figura 3.2: Construção de um C-obstáculo através de soma de Minkowski. (a) Crescimento de um obstáculo retangular para as dimensões de um robô triangular. (b) Confirmação de que o crescimento através da soma de Minkowski resulta no C-obstáculo esperado.

da letra (a) realmente permite a construção do C-obstáculo, uma vez que o ponto de referência do robô toca o C-obstáculo somente quando o robô real toca o obstáculo real.

Até o momento, tratou-se a dilatação de um único obstáculo. Para concluir esta seção, falta apenas mostrar como construir todo o C-obstáculo. Para tanto, basta realizar a operação de união entre todos os obstáculos já "dilatados". Portanto, para um ambiente constituído de um conjunto de obstáculos $\mathcal{O} = \{P_1, \ldots, P_n\}$, tem-se:

$$\mathcal{CO} = \bigcup_{i=1}^{n} \mathcal{C}P_i \,, \tag{3.3}$$

onde $\mathcal{CO}$ denota o C-obstáculo total, de todo o conjunto $\mathcal{O}$, e $\mathcal{C}P_i$ denota o resultado da operação $P_i \oplus (-R(0,0))$.

3.3 Espaço de configurações em $\mathbb{R}^3$

Para robôs de geometria complexa, e que sejam livres para executar rotações, o espaço de configurações deve ser definido em $\mathbb{R}^3$ (x, y, θ). Logo, os obstáculos que constituem o C-obstáculo são *poliedros* em $\mathbb{R}^3$. Uma definição formal para estes poliedros é a seguinte:

$$\mathcal{C}P_i = \{(x, y, \theta) \in \mathbb{R}^2 \times [0 : 2\pi) \mid R(x, y, \theta) \cap P_i \neq \emptyset\}, \tag{3.4}$$

onde $\mathcal{C}P_i$ denota o C-obstáculo do obstáculo P_i, e os pontos $(x, y, 0)$ são equivalentes aos pontos $(x, y, 2\pi)$.

Nesta seção, propõe-se um algoritmo para a construção de C-obstáculos no $\mathbb{R}^3$. No espaço (x, y, θ), cada plano $\theta = constante$ é constituído de um C-obstáculo em $\mathbb{R}^2$. Este, por sua vez, pode ser construído por meio da soma de Minkowski, como discutido na Seção 3.2. Logo, o C-obstáculo em $\mathbb{R}^3$ pode ser construído por meio de uma varredura em θ, onde um C-obstáculo em $\mathbb{R}^2$ é calculado para cada plano. O C-obstáculo em $\mathbb{R}^3$ seria calculado se houvesse uma varredura contínua no intervalo $[0 : 2\pi)$. Obviamente, uma varredura contínua é computacionalmente inviável, e por isso a metodologia proposta aproxima o poliedro desejado por meio de uma varredura discreta em l planos ou camadas. O algoritmo que será visto adiante propõe, então, que se realize uma triangulação entre camadas de forma a aproximar as laterais do poliedro. A razão de se escolher uma triangulação foi tornar a estrutura de dados que representa os poliedros compatível com a entrada do programa Tetgen [Si, 2004], o qual foi utilizado na discretização do espaço de configurações em uma malha de tetraedros. Conforme ficará claro nos Capítulos 4 e 5, este trabalho utiliza o método de elementos finitos, que necessita de tal discretização, para resolver a equação de Laplace no domínio

dado pelo espaço de configurações livre.

A Figura 3.3 ilustra a varredura discreta na construção de um C-obstáculo para um robô de formato triangular e um obstáculo de formato retangular. De fato, ilustra-se a construção da parte do C-obstáculo referente ao intervalo $[0 : \frac{\pi}{2}]$ a partir de três planos: $\theta = 0$, $\theta = \frac{\pi}{4}$ e $\theta = \frac{\pi}{2}$. As Figuras 3.3 (a), (b) e (c) apresentam, respectivamente, a construção dos C-obstáculos em $\mathbb{R}^2$ para os referidos planos. A Figura 3.3 (d) mostra os C-obstáculos construídos no $\mathbb{R}^2$ ocupando suas posições em $\mathbb{R}^3$, em seus respectivos planos, de forma que estejam empilhados. Por fim, na Figura 3.3 (e) realiza-se uma triangulação entre camadas de forma a "fechar" as laterais do poliedro.

A Figura 3.4 ilustra a idéia utilizada no algoritmo, que será apresentado adiante, para realizar a triangulação entre camadas. Na Figura 3.4, supõe-se que se tenha um par de camadas A, constituída dos vértices a e a_p, e B, constituída dos vértices b e b_p. O ponto mais próximo (segundo distância euclidiana) de a, na camada B, é o vértice b, portanto é razoável que eles formem um segmento de um triângulo. Existem duas possibilidades de vértices para se formar um triângulo com o segmento $\overline{ab}$, e essas possibilidades são os vértices a_p e b_p, que são vizinhos de a e b, respectivamente. Como critério de escolha utiliza-se a distância euclidiana ao vértice da outra camada, isto é, se a distância entre b e a_p, $\|b - a_p\|$, é menor que a distância entre a e b_p, $\|a - b_p\|$, então o triângulo a ser formado deve ser o triângulo (b, a, a_p). Em caso contrário, forma-se o triângulo (b, a, b_p).

O algoritmo proposto é o Algoritmo 3.2. Para cada par de camadas, o algoritmo opera da seguinte forma: inicialmente, um vértice qualquer a da camada inferior A é escolhido. Encontra-se o vértice b da camada superior B mais próximo de a. Utiliza-se então o teste de distância explicado anteriormente para escolher, ou o vizinho de a, no sentido anti-horário, ou o

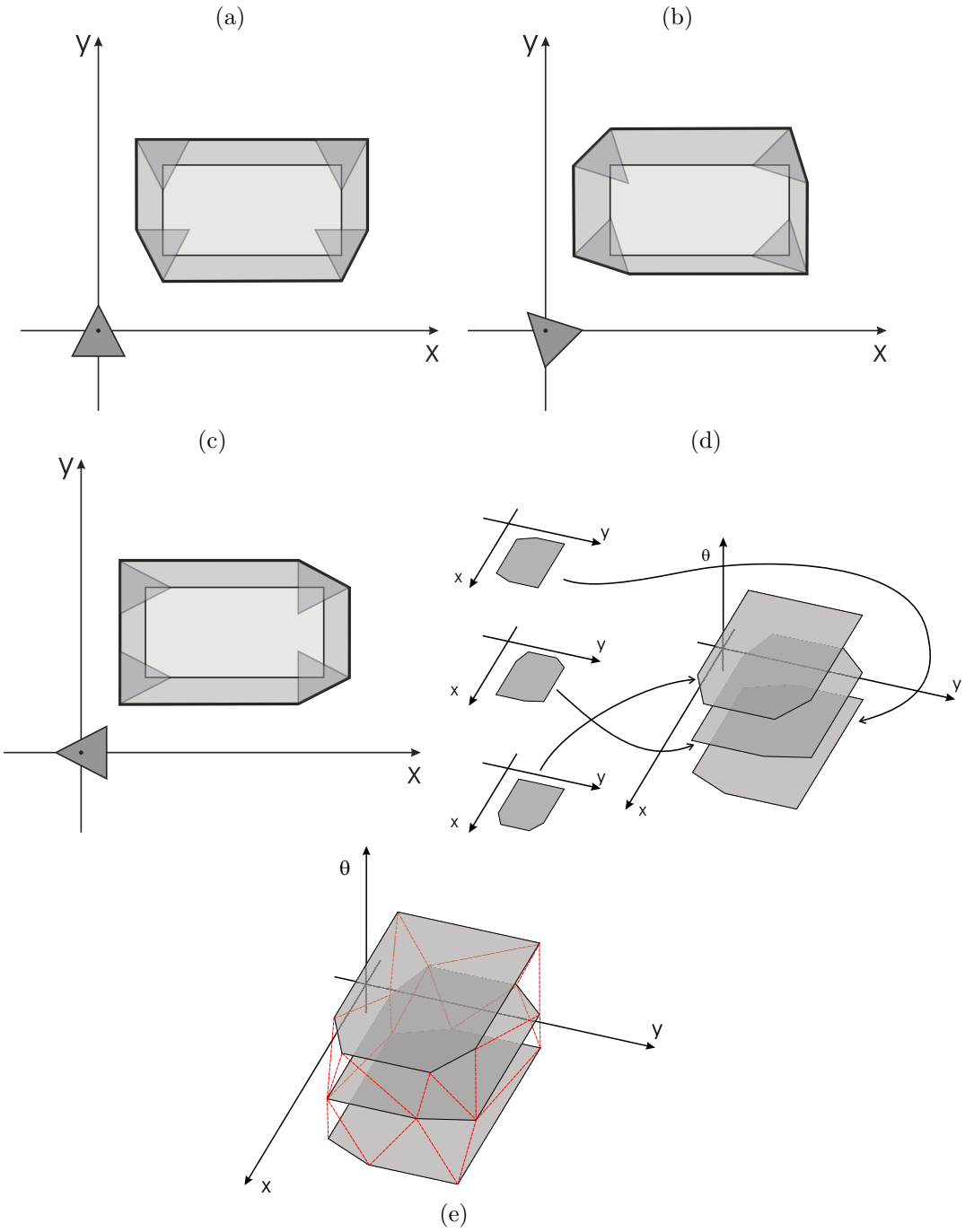

Figura 3.3: Passos na construção do C-obstáculo em $\mathbb{R}^3$. (a) Camada $\theta = 0$. (b) Camada $\theta = \frac{\pi}{4}$. (c) Camada $\theta = \frac{\pi}{2}$. (d) Empilhando as camadas. (e) Triangulação das laterais.

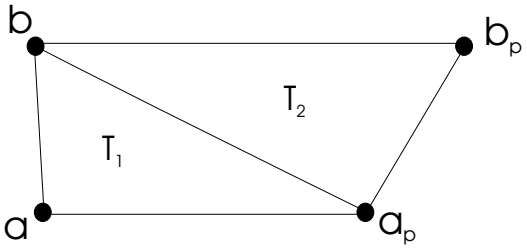

Figura 3.4: Ilustração do critério de distância que é utilizado para se escolher o triângulo a ser formado na triangulação entre camadas. O triângulo T_1 é formado porque $\|b - a_p\| < \|a - b_p\|$.

vizinho de b, também no sentido anti-horário, para formar um triângulo com o segmento $\overline{ab}$. O resultado do teste define então um novo segmento da triangulação, $\overline{ba_p}$ ou $\overline{ab_p}$, que conecta as duas camadas. Na próxima iteração, utiliza-se novamente o teste de distância para encontrar o vértice que formará um triângulo com aquele novo segmento encontrado na iteração anterior. O algoritmo percorre os vértices dos polígonos das duas camadas encontrando triângulos e só é finalizado quando o segmento encontrado na iteração anterior é o segmento inicial $\overline{ab}$, ou seja, realiza-se uma volta completa nos polígonos, de forma a fechar completamente as laterais.

Para se analisar o algoritmo proposto, deve-se notar que, para cada par de camadas (C_i, C_{i+1}), cada aresta dos polígono em C_i e em C_{i+1} é uma aresta de um triângulo. Portanto, o número de triângulos para cada par de camadas é igual à soma $k + h$, onde k é o número de vértices de C_i e h é o número de vértices de C_{i+1}. Para um número de camadas l tem-se um número de pares de camadas igual a $l-1$. Logo, verifica-se que a complexidade computacional do algoritmo é limitada por $O(lv)$, onde l é o número de camadas, e v é o número de vértices da camada que apresenta o maior número de vértices.

Um ponto, ainda importante de se discutir sobre o algoritmo, é que o algoritmo não é geral para se conectar um polígono qualquer A, de um plano,

algoritmo 3.2 Calcula o C-obstáculo em $\mathbb{R}^3$ a partir das camadas $\mathbb{R}^2$

Entradas: O conjunto C de todas as camadas em $\mathbb{R}^2$ $C_1 \dots C_n$, ordenadas, onde C_1 é a camada mais baixa e C_n é a camada do topo. Os vértices das camadas devem estar ordenados em sentido anti-horário.

Saídas: T armazena todos os triângulos que definem as faces do Poliedro em $\mathbb{R}^3$ que corresponde ao C-obstáculo.

1: $i \leftarrow 1$
2: **enquanto** $i \neq n$ **faça**
3: $A \leftarrow C_i$; $B \leftarrow C_{i+1}$
4: $a \leftarrow$ um vértice qualquer de A
5: $b \leftarrow$ o vértice de B mais próximo de a
6: $a_{inicial} \leftarrow a$; $b_{inicial} \leftarrow b$
7: **repita**
8: $a_p \leftarrow$ vértice de A seguinte ao vértice a
9: $b_p \leftarrow$ vértice de B seguinte ao vértice b
10: **Se** $\|b - a_p\| < \|a - b_p\|$ **então**
11: Adicione o triângulo (b, a, a_p) à saída T
12: $a \leftarrow a_p$
13: **senão**
14: Adicione o triângulo (b, a, b_p) à saída T
15: $b \leftarrow b_p$
16: **fim Se**
17: **até que** $a = a_{inicial}$ **e** $b = b_{inicial}$
18: $i \leftarrow i + 1$
19: **fim enquanto**

a um outro polígono qualquer B, de outro plano. Suponha que um vértice r do polígono A esteja muito distante de todos os vértices de B. Se os outros vértices de A estiverem próximos aos vértices de B, então o vértice r nunca será selecionado para formar um triângulo, a não ser que ele seja o escolhido na primeira iteração, de forma aleatória. Portanto, num caso onde um vértice nunca é escolhido, o algoritmo não consegue fechar as laterais. O fato do algoritmo funcionar na construção do C-obstáculo no $\mathbb{R}^3$ depende da variação de ângulo $\delta\theta$ entre camadas. Se a variação for pequena, existirá uma grande correlação entre o polígono da camada superior e o polígono

da camada inferior, numa dada iteração do algoritmo. Logo, o algoritmo funcionará corretamente.

Verificou-se, durante os testes realizados, que uma variação de ângulo dentro do intervalo em radianos $\frac{\pi}{180} < \delta\theta < \frac{\pi}{36}$ ($1^o < \delta\theta < 5^o$) pode ser considerada como uma variação pequena. Quanto menor $\delta\theta$, melhor está se aproximando o poliedro real, mas maior é o número de camadas. Em trabalhos como [Fuchs et al., 1977, Boissonat, 1988, Ekoule et al., 1991, Meyers et al., 1992] são propostos algoritmos mais genéricos e, também, mais complexos que o apresentado aqui, os quais poderiam ser utilizados nos casos mais complicados. Nestes trabalhos, trata-se inclusive a questão da conexão entre camadas de topologias distintas, que é um tópico ainda mais complicado. Este tópico é discutido mais adiante nesta seção.

A metodologia de construção de espaços de configurações em três dimensões apresentada aqui foi implementada utilizando a linguagem C++ e a biblioteca de geometria computacional CGAL (*Computational Geometry Algorithms Library* [CGAL, 2005]), versão 3.0.1. Esta biblioteca foi utilizada principalmente para implementar a soma de Minkowski e a união de polígonos em cada camada no $\mathbb{R}^2$. A Figura 3.5 apresenta um exemplo, onde reconstruiu-se um poliedro com sucesso, utilizando a implementação desenvolvida. Nas Figuras 3.5 (a) e (b) são mostrados, respectivamente, as dimensões de um robô retangular, e as dimensões de um obstáculo composto de três retângulos formando um U. Na Figura 3.5 (c), apresenta-se o resultado obtido na construção do C-obstáculo correspondente. Deve-se observar que foi mostrado apenas o intervalo $[0 : \pi]$, uma vez que o poliedro, neste caso de robô retangular, é simétrico. Na Figura 3.5 (c), ainda se observa um prisma retangular contornando o obstáculo. Este prisma foi incluído manualmente e representa um contorno externo, que tem a função de limitar

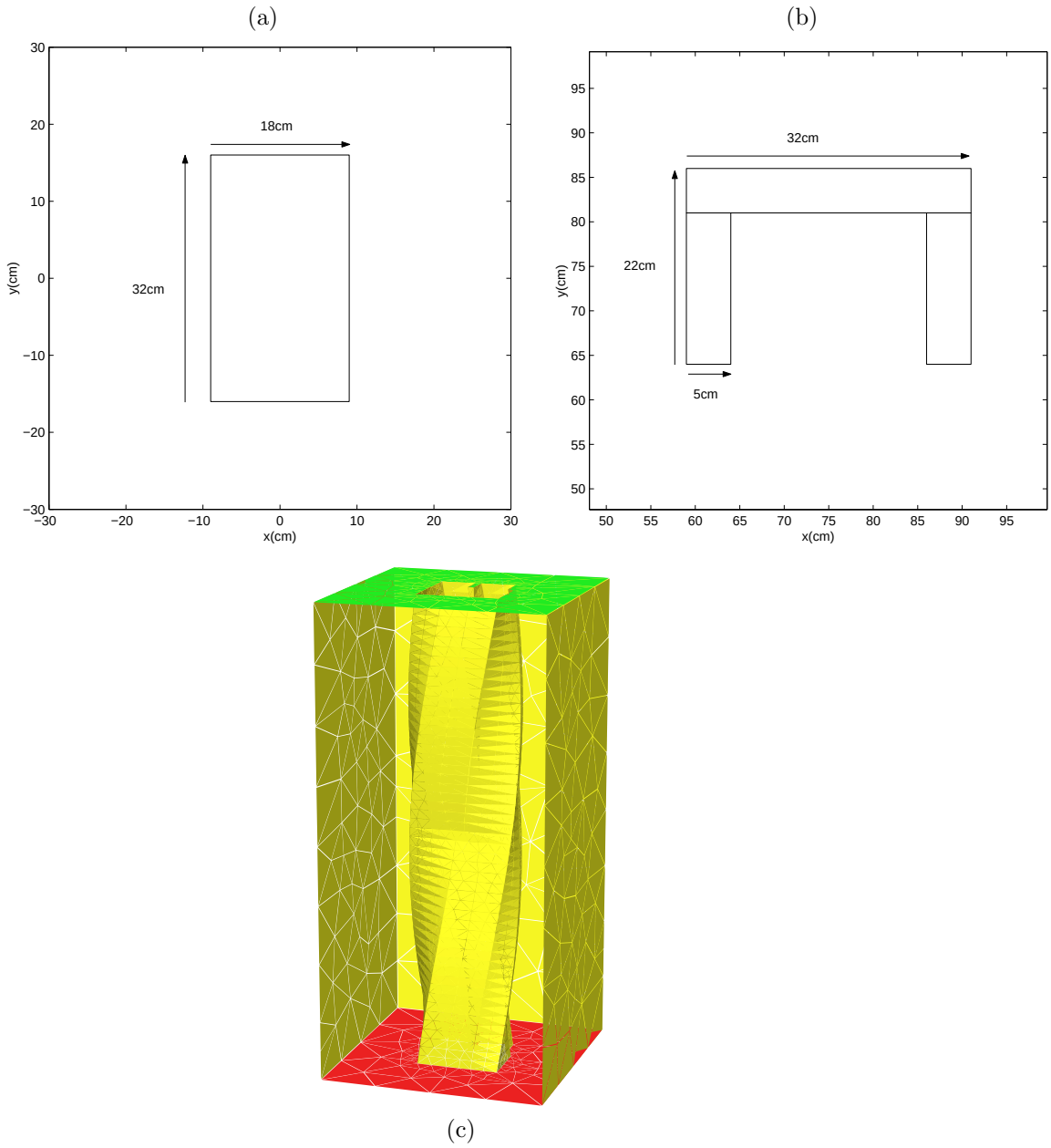

Figura 3.5: Aplicação do algoritmo proposto para construção de C-obstáculos em $\mathbb{R}^3$ para um robô retangular e um obstáculo em forma de U. (a) Dimensões do robô. (b) Dimensões do obstáculo. (c) Resultado obtido aplicando-se o Algoritmo 3.2.

a região do espaço onde o robô pode navegar. A necessidade deste prisma deve-se ao fato, conforme será visto no Capítulo 5, do método de elementos finitos trabalhar com domínios limitados.

Até o momento, foi discutida a construção de poliedros a partir de camadas contendo um único polígono, porém os espaços de configurações reais são compostos de diversos polígonos. Nestes espaços, pode ocorrer o problema de camadas adjacentes com topologias distintas. Mudanças de topologia podem ocorrer quando dois obstáculos originalmente distintos são transformados em um único polígono no C-obstáculo bidimensional, para apenas alguns planos $\theta = constante$. A Figura 3.6 ilustra o fato. Na letra (a) da Figura 3.6, pode-se observar um robô retangular com orientação de 0 radiano. Nesta condição, o robô consegue passar entre os dois obstáculos, e tem-se o C-obstáculo em $\mathbb{R}^2$ mostrado na Figura 3.6 (b). Por outro lado, na Figura 3.6 (c) o robô está rotacionado de $\frac{\pi}{2}$ radianos e não pode mais passar entre os obstáculos. O C-obstáculo em $\mathbb{R}^2$, apresentado na Figura 3.6 (d) reflete esta impossibilidade, uma vez que os dois polígonos inicialmente distintos, foram agora mapeados em um único. O Algoritmo 3.2 não prevê formas para a conexão entre as camadas das letras (b) e (d) da Figura 3.6. Por simplicidade, nos testes realizados, evitou-se este tipo de situação.

Para finalizar, propõe-se uma solução, a qual não foi implementada por fugir do escopo deste trabalho, para eliminar o problema das topologias distintas. Para que se construa o C-obstáculo completo de um conjunto de obstáculos $\mathcal{O}$, propõe-se a operação de união de poliedros, ao invés da união de polígonos. Como mostrado na Seção 3.1, realiza-se uma decomposição de cada polígono em sub-polígonos convexos. Portanto, o que se propõe é que se construa os poliedros provenientes dos sub-polígonos convexos utilizando o Algoritmo 3.2, e que depois seja realizada a união entre esses poliedros.

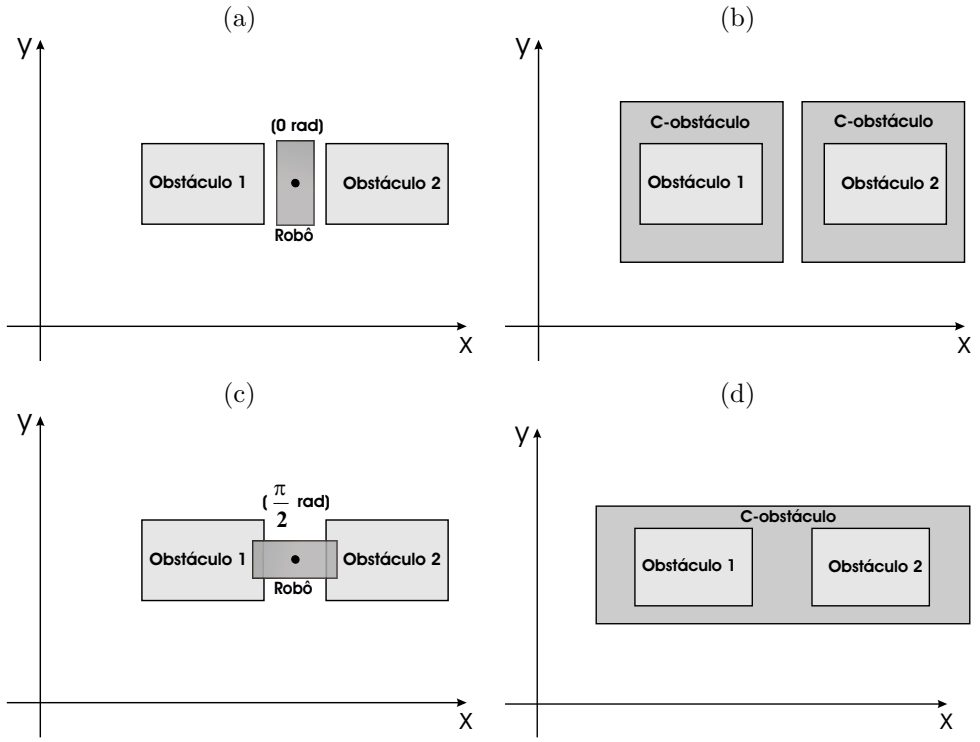

Figura 3.6: Ilustração de topologias de C-obstáculo distintas. (a) Robô com orientação de 0 radiano. (b) C-obstáculo correspondente à orientação de 0 radiano. (c) Robô com orientação de $\frac{\pi}{2}$ radianos. (d) C-obstáculo correspondente à orientação de $\frac{\pi}{2}$ radianos.

Assim, não haveria camadas com topologias distintas durante a construção dos poliedros. Esta solução não foi implementada por estar fora do escopo do presente trabalho mas deverá ser em um futuro próximo.

Capítulo 4

Equação de Laplace e Navegação de Robôs

> Não é paradoxo dizer que em nossos momentos mais teóricos podemos estar mais próximos de nossas aplicações mais práticas.
>
> *Alfred North Whitehead (1861–1947)*

Neste capítulo apresenta-se a equação de Laplace e as principais propriedades da sua solução. Uma forma de se modelar o problema de navegação de robôs por meio da equação de Laplace também é apresentada. Mais especificamente, esse modelo passa pela definição de condições de contorno, e portanto propõe-se uma metodologia para a atribuição dessas condições de contorno de maneira consistente. Aspectos de controle também são abordados e, pela verificação de estabilidade da lei de controle utilizada, prova-se que o método de navegação proposto é *completo*. Entende-se por completo o fato do robô sempre atingir o alvo num tempo finito, desde que exista um caminho.

É importante ressaltar que os resultados obtidos aqui são válidos para robôs holonômicos. Esta categoria de robôs é caracterizada por sofrer apenas restrições holonômicas em seu movimento. Restrições holonômicas são

restrições de igualdade sobre as configurações q de um sistema:

$$G(q) = 0. \tag{4.1}$$

Esse tipo de restrição não limita as direções de movimento que o robô pode executar no próximo instante. Por outro lado, restrições não-holonômicas são restrições também de igualdade, que alteram o número de velocidades independentes do sistema. Estas restrições são da forma:

$$G(q, \frac{dq}{dt}, \frac{d^2q}{dt^2}, \ldots) = 0\,, \tag{4.2}$$

onde q é a configuração do robô e $\frac{d^mq}{dt^m}$ é a m-ésima derivada temporal do movimento do robô que não pode ser integrada e reduzida à forma de restrição holonômica.

Veículos ou robôs não-holonômicos são aqueles sujeitos a restrições não-holonômicas. Um exemplo de veículo não-holonômico é o automóvel, o qual não pode ter velocidades perpendiculares à direção de movimento. Se o automóvel fosse holonômico, ele poderia se movimentar em qualquer direção e portanto não seriam necessárias manobras para estacioná-lo. Logo, o planejamento de trajetórias e controle para sistemas não-holonômicos é mais complexo que para sistemas holonômicos, e está fora do escopo do presente trabalho. Em [Pereira, 2003], apresenta-se um controlador para robôs não-holonômicos que pode rastrear trajetórias obtidas por meio de funções de navegação. Dessa forma, este controlador poderia ser utilizado para aplicar a metodologia proposta a robôs não-holonômicos.

No final do capítulo, são feitas analogias entre o problema de navegação e o problema de cálculo de campo em eletrostática. Essa analogia permite tirar conclusões sobre as trajetórias que podem ser executadas utilizando a

metodologia apresentada.

4.1 Funções Harmônicas

Seja um robô, R, mapeado como um ponto, q, em seu espaço de configurações, $\mathcal{C}$. O problema de navegação resume-se em levar o robô até uma configuração final desejada, q_d, a partir de uma configuração inicial, q_i, evitando configurações de colisão. Em outras palavras, o robô deve estar restrito ao seu espaço de configurações livre, $\mathcal{F} \subseteq \mathcal{C}$.

Funções harmônicas podem ser utilizadas para resolver o problema de navegação [Connnolly et al., 1990, Connnolly, 1992]. Funções harmônicas são soluções da equação de Laplace:

$$\nabla^2 \phi = 0\,, \tag{4.3}$$

onde ϕ é uma função harmônica, e ∇^2 é o operador *Laplaciano* [Macedo, 1988]. A Equação (4.3) é válida no domínio limitado $\Omega \subset \mathbb{R}^n$. Para o problema de navegação, o domínio é o espaço de configurações livre $\mathcal{F} \subset \mathbb{R}^n$. Conforme apresentado em capítulos anteriores, esta dissertação focaliza a navegação de robôs móveis no plano, e portanto os domínios são em $\mathbb{R}^2$ (x, y) ou em $\mathbb{R}^3$ (x, y, θ).

As seguintes propriedades das funções harmônicas são úteis para o caso de robótica:

1. Elas são suaves em Ω, isto é, têm derivadas segundas contínuas;

2. Elas obedecem o princípio max-min [Zachmanoglou and Thoe, 1986, Weinstock, 1974], isto é, seus máximos e mínimos estão localizados nas fronteiras do domínio, para qualquer região fechada;

3. Os pontos críticos no interior do domínio Ω são todos pontos de sela;

4. Elas são soluções únicas, para um dado conjunto de condições de contorno;

A primeira propriedade é importante pelo fato do robô ser controlado, tipicamente, para seguir o negativo do gradiente da função potencial, e portanto é desejável que se tenha suavidade. A segunda propriedade permite concluir que não existem mínimos locais no interior do domínio. Mínimos locais são pontos críticos, que apesar de não serem o alvo, são pontos de equilíbrio estáveis onde o robô pode ficar parado. Por outro lado, pontos de sela são pontos de equilíbrio instáveis, isto é, pequenas perturbações são suficientes para que o sistema abandone esses pontos. Assim, a terceira propriedade garante que os pontos críticos no interior do domínio, apesar de serem pontos onde os gradientes são nulos, desde que o robô tenha uma dada velocidade inicial, não podem parar o robô antes que o mesmo atinja o alvo. A quarta propriedade também é desejada porque garante repetibilidade e reprodutibilidade.

Em [Rimon and Koditschek, 1992] propõem-se funções de potencial artificiais, que não têm mínimos locais, denominadas *funções de navegação*. Uma função de navegação é definida como um mapeamento $\varphi : \mathcal{F} \rightarrow [0, 1]$ que é:

1. Suave em $\mathcal{F}$;

2. Polar em q_d, isto é, existe um único mínimo, localizado em q_d, no subconjunto conectado de $\mathcal{F}$ que contém q_d;

3. Admissível em $\mathcal{F}$, isto é, na fronteira de $\mathcal{F}$ é uniformemente máxima;

4. Uma função de Morse [Milnor, 1970];

De acordo com [Connnolly et al., 1990] pode-se construir funções de navegação através de funções harmônicas. Conforme apresentado anteriormente, suavidade é uma propriedade de toda função harmônica, e portanto o primeiro item das funções de navegação é automaticamente atendido.

Para avaliar se uma função harmônica é polar e admissível, considere uma configuração desejada q_d. Considere, também, que a fronteira de $\mathcal{F}$ é constituída de pontos p, os quais pertencem ao contorno dos obstáculos ou a um contorno externo que abrange toda a área de trabalho do robô. Aplicando-se ao alvo a condição $\phi(q_d) = 0$, e à fronteira de $\mathcal{F}$ a condição $\phi(p) = 1$, os ítens 2 e 3 são atendidos. Para se chegar a essa conclusão, basta lembrar que a função harmônica atende ao princípio min-max. Funções de Morse são funções cujas hessianas (matriz das derivadas segundas) não são singulares nos pontos críticos. Uma forma mais simples de se entender as funções de Morse é que as mesmas não apresentam pontos críticos degenerados. Como os únicos pontos críticos que podem aparecer no interior do domínio para funções harmônicas são pontos de sela, a condição de ser função de Morse também é atendida pelas funções harmônicas. Logo, funções harmônicas podem ser funções de navegação globais.

4.2 Condições de Contorno

A fronteira do domínio Ω, lembrando que o domínio Ω corresponde aqui ao espaço de configurações livre $\mathcal{F}$, é definida pelos contornos do C-obstáculo, pelo contorno externo do espaço de trabalho do robô e pelo contorno externo da região de alvo. Em nossa abordagem, não necessariamente uma configuração final q_d necessita ser definida. Na verdade, uma região no espaço Ω_d conexa e fechada pode ser definida como alvo, ou seja, um dado conjunto de

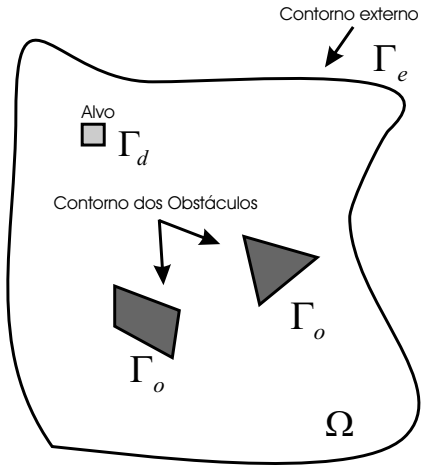

Figura 4.1: Ilustração do domínio Ω e seus contornos $\Gamma = \Gamma_d \cup \Gamma_o \cup \Gamma_e$.

configurações é definido como alvo. A Figura 4.1 ilustra o domínio onde a equação de Laplace deve ser resolvida e seus contornos.

As fronteiras de Ω são modeladas utilizando condições de contorno para a Equação (4.3). A definição de condições de contorno é o que efetivamente garante a unicidade da solução. Tradicionalmente, dois tipos de condição de contorno são definidos para a equação de Laplace: Dirichlet constante e Neumann homogêneo.

A condição de contorno de Dirichlet constante:

$$\phi_D = \phi|_\Gamma = c\,, \tag{4.4}$$

onde c é um valor constante, implica que o gradiente é normal à fronteira Γ onde esta condição é definida. Isto pode ser provado através da definição da diferencial da função ϕ:

$$d\phi = \nabla\phi \cdot d\mathbf{l}\,, \tag{4.5}$$

onde $d\mathbf{l}$ é um deslocamento elementar. O valor de ϕ é constante e igual a c ao longo do contorno onde foi definida a condição de Dirichlet. Logo $d\phi = 0$

neste contorno, o que só pode ocorrer descartando a solução trivial $d\mathbf{l} = 0$, se:

$$\nabla\phi \perp d\mathbf{l}\,, \tag{4.6}$$

onde $d\mathbf{l}$ neste caso é o deslocamento tangente ao contorno de Dirichlet, e $\perp$ indica perpendicularidade. Pela Equação (4.5), pode-se concluir também que a máxima variação de ϕ ocorre na direção do gradiente, ou seja, o sentido do gradiente aponta para o crescimento máximo da função. Conforme será apresentado na Seção 4.3, o robô segue o negativo do gradiente do potencial (também chamado campo potencial). Dessa forma, as superfícies dos obstáculos definidas com condição de Dirichlet constante irão repelir o robô como mostrado na Figura 4.2 (a).

A condição de contorno de Neumann homogêneo:

$$\phi_N = \frac{\partial\phi}{\partial\mathbf{n}}|_\Gamma = 0\,, \tag{4.7}$$

onde $\mathbf{n}$ é um vetor normal ao contorno, força o gradiente ser tangente a este contorno. Isto pode ser mostrado facilmente utilizando a definição da derivada de ϕ na direção de $\mathbf{n}$:

$$\frac{\partial\phi}{\partial\mathbf{n}} = \nabla\phi\cdot\mathbf{n}. \tag{4.8}$$

De acordo com (4.7), a expressão em (4.8) deve se anular. Logo, o gradiente deve ser perpendicular a $\mathbf{n}$, ou seja, ser tangente ao contorno. Assim, o robô seguindo o gradiente descendente tangencia os obstáculos cujos contornos são definidos com condição de Neumann homogêneo. Veja Figura 4.2 (b).

Uma terceira possibilidade de condição de contorno é a superposição das

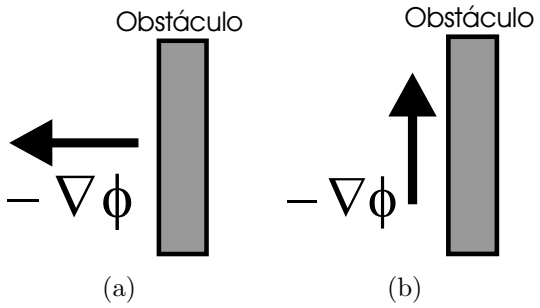

Figura 4.2: Comportamento do gradiente diante de condições de contorno. (a) Condição de contorno de Dirichlet constante. (b) Condição de Neumann homogêneo.

condições anteriores chamada Condição Mista:

$$\phi = k_1\phi_D + k_2\phi_N \,, \tag{4.9}$$

onde k_1 e k_2 são constantes definidas de forma que o sistema exiba um determinado comportamento desejado e $k_1 + k_2 = 1$.

Juntamente com as condições de contorno anteriores, uma outra é necessária quando a orientação do robô deve ser levada em conta. Esta condição é chamada *Condição de Contorno Periódica* [Ida and Bastos, 1992]. Acredita-se que este trabalho é o primeiro a utilizar tal condição no contexto de robótica. Como já foi dito anteriormente, a inclusão da orientação do robô é necessária quando o robô apresenta uma geometria complexa ou quando é desejável que o robô atinja o alvo com uma dada orientação. O domínio, para este caso, tem três dimensões (x, y, θ), onde o eixo θ é periódico no intervalo de $\frac{2\pi}{k}$ radianos e $k \geq 1$. O valor de k depende do eixo de simetria do robô. Para um robô totalmente assimétrico, tem-se periodicidade em intervalos de 2π radianos, onde $k = 1$. A periodicidade é facilmente representada limitando o domínio $0 \leqslant \theta < \frac{2\pi}{k}$ e impondo a seguinte condição aos planos $\theta = 0$

e $\theta = \frac{2\pi}{k}$:

$$\phi(x, y, 0) = \phi(x, y, \frac{2\pi}{k}) \,, \tag{4.10}$$

a qual é chamada Condição de Contorno Periódica.

A definição das condições de contorno, feita de forma adequada, possibilita o sucesso da navegação a partir de qualquer configuração inicial no espaço de configurações livre. Assim, as condições de contorno devem seguir as seguintes restrições:

- As condições de contorno do alvo, Γ_d, são todas iguais do tipo Dirichlet e possuem valor zero;
- As condições de contorno da fronteira que define o espaço de trabalho do robô, Γ_e, e demais obstáculos do ambiente, Γ_o, são iguais do tipo Dirichlet e com valores positivos;
- No caso de haver eixos correspondentes a ângulos, devem-se definir condições de contorno periódicas como apresentado na Equação (4.10).

Supondo que o alvo é atingível, ao se respeitar as condições anteriores pode-se garantir que: (1) não há mínimos locais no potencial artificial gerado; (2) o controlador sugerido na próxima seção é estável; e (3) o tempo que o robô leva para atingir o alvo é finito independentemente da configuração inicial – como provado na Seção 4.3.

4.3 Controlador

Supondo-se, que o movimento do robô holonômico, R, que está representado em seu espaço de configurações, $\mathcal{C} \subset \mathbb{R}^n$, é descrito por um modelo

cinemático simples da forma:

$$\dot{q} = u(q)\,, \tag{4.11}$$

onde u é o vetor de entradas do sistema, e $\dot{q}$ é o vetor velocidade do robô.

Dada uma função harmônica $\phi(q)$ calculada sob as condições de contorno estabelecidas na Seção 4.2, isto é, o alvo é submetido à condição de contorno de Dirichlet constante com valor zero, o contorno do C-obstáculo e o contorno externo são submetidos a condições, também, de Dirichlet constante. Mas esse valor constante possui valor idêntico positivo e ainda existem condições de contorno periódicas. Propõe-se a seguinte lei de controle para resolver o problema de navegação:

$$u(q) = \begin{cases} -\mathbf{K} \cdot \frac{\nabla\phi(q)}{\|\nabla\phi(q)\|^{\alpha}} & \text{se } \nabla\phi(q) \neq 0 \\ 0 & \text{se } \nabla\phi(q) = 0 \end{cases}, \tag{4.12}$$

onde $\nabla\phi(q)$ é o gradiente de $\phi(q)$, e o operador $\|.\|$ representa a norma euclidiana. Como no interior da região alvo Ω_d tem-se $\nabla\phi = 0$, a lei de controle força explicitamente o robô a parar quando o alvo é atingido. A constante $\alpha \in \mathbb{N}_+$, quando maior que 1, implica que a velocidade seja inversamente proporcional ao valor da norma do gradiente. Como o gradiente é maior na região em torno do alvo, a velocidade torna-se menor nesta região e portanto o robô pode atingir seu alvo de forma mais suave. Para $\alpha = 1$, tem-se o gradiente normalizado, o que determina uma velocidade constante ao longo de toda a trajetória do robô. Este caso particular foi utilizado nos experimentos que são apresentados nesta dissertação (veja Capítulo 6).

Em (4.12), a matriz diagonal $\mathbf{K}$ tem dimensões $n \times n$, onde n é a dimensão do espaço de configurações, e é utilizada para compatibilizar a variável de

controle u com valores de velocidade do robô. A matriz $\mathbf{K}$ é da forma:

$$\mathbf{K} = \begin{pmatrix} k_x & 0 & 0 & \dots & 0 \\ 0 & k_y & 0 & \dots & 0 \\ 0 & 0 & k_\theta & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_n \end{pmatrix}, \tag{4.13}$$

onde $k_n, k_y, k_\theta, \dots, k_n$ são constantes não-negativas. Algumas destas constantes escalonam velocidades de translação e outras escalonam velocidades rotacionais. Ao se considerar robôs móveis que se movem no plano, tem-se $k_x = k_y$ escalonando a velocidade de translação v, medida em metros por segundo, e k_θ escalonando a velocidade de rotação ω, medida em radianos por segundo.

Uma característica importante de se analisar em um método de planejamento de trajetórias é se este método é *completo*. Isto significa que se deve provar que $q \to q_d \in \Omega_d$, num tempo finito t_f. Uma forma de se obter esta prova, para métodos baseados em leis de controle, é avaliar a estabilidade da malha fechada resultante do sistema. De forma intuitiva, um sistema ser estável significa que pequenas perturbações nas entradas do sistema produzem pequenas mudanças nas saídas. O método direto de Lyapunov permite determinar a estabilidade de um sistema sem integrar explicitamente a equação diferencial que descreve o sistema. Um conceito importante que é utilizado na teoria de estabilidade de Lyapunov é o conceito de *funções definidas positivas*:

Definição 4.1 (Funções definidas positivas) *Uma função diferenciável* $V : \mathbb{R}^N \times \mathbb{R}_+ \to \mathbb{R}$ *é definida positiva numa região* Ω *contendo a origem se:*

1. $V(0) = 0$;

2. $V(x(t)) > 0$, *se* $x \in \Omega$, $x \neq 0$ *e* $\forall t \geq 0$.

Para a avaliação da estabilidade de um sistema, pode-se utilizar o teorema de Lyapunov [Murray et al., 1994]:

Teorema 4.2 (Teorema de Lyapunov) *Seja um sistema dinâmico definido por* $\dot{x} = f(x(t))$. *Seja, também,* $V(x(t))$ *uma função não-negativa com derivada temporal* $\dot{V}(x(t))$ *ao longo das trajetórias do sistema. Se* $V(x(t))$ *é definida positiva e decrescente, e* $-\dot{V}(x(t))$ *é definida positiva, então a origem* $x = 0$ *do sistema é assintoticamente estável.*

A prova do Teorema de Lyapunov pode ser encontrada em [Sastry, 1999]. O fato do ponto de equilíbrio $x = 0$ ser assintoticamente estável significa que o sistema converge para este ponto. Com base no Teorema 4.2, faz-se a seguinte proposição sobre a navegação realizada através de funções harmônicas:

Proposição 4.3 *Se um robô holonômico R navega numa área de trabalho fechada, o alvo é submetido às condições de contorno de Dirichlet constante de valor igual a zero, o contorno do C-obstáculo e o contorno externo são submetidos a condições de contorno de Dirichlet constante com valor idêntico positivo e ainda condições de contorno periódicas, conforme estabelecido anteriormente, então o controlador (4.12) garante solução completa para o problema de navegação.*

Prova: Suponha que o sistema de coordenadas seja tal que o alvo esteja na origem. Para provar que $q \rightarrow q_d \in \Omega_d$ num tempo finito t_f, basta provar que a malha fechada resultante do controlador (4.12) é assintoticamente estável em relação à origem. Uma forma de se obter esta prova é verificar

que a função harmônica $\phi(q)$ obedece às propriedades da função $V(x,t)$ do Teorema de Lyapunov. Pelas propriedades das funções harmônicas apresentadas na Seção 4.1 e pelas condições de contorno que foram impostas, pode-se afirmar diretamente que a função $\phi(q)$ é definida positiva e decrescente. O controlador em (4.12) força que $\dot{\phi}(\Omega_d) = 0$, e portanto resta provar que $\dot{\phi}(q) < 0$ fora do alvo:

$$\begin{aligned}\dot{\phi}(q) =& \frac{d\phi}{dq} \cdot \frac{dq}{dt} = \nabla\phi(q) \cdot \dot{q} = \nabla\phi(q) \cdot u = -\frac{\nabla\phi(q) \cdot \mathbf{K}\nabla\phi(q)}{\|\nabla\phi(q)\|^{\alpha}} \\ =& -\frac{k_x(\nabla\phi_x)^2 + \ldots + k_n(\nabla\phi_n)^2}{\|\nabla\phi(q)\|^{\alpha}} < 0\,,\end{aligned}$$

para $\|\nabla\phi(q)\| \neq 0$, e $k_x, k_y, k_\theta, \ldots, k_n$ constantes positivas. ∎

Como dito anteriormente, o sistema não apresenta mínimos locais e os pontos de gradiente nulo no interior do domínio são pontos de sela. Pelo fato destes pontos de sela serem pontos de equilíbrio instáveis, qualquer ruído do sistema impede que o mesmo fique parado neles. Além disso, como será visto mais adiante nesta dissertação, antes de se resolver a equação de Laplace faz-se uma discretização do domínio em elementos. Como o gradiente é constante no interior de cada elemento, e o cálculo é feito com base em alguns valores de potenciais discretos em torno do elemento, a possibilidade de se ter um gradiente realmente nulo no interior do domínio é remota. Ainda, se para uma dada discretização do domínio houver um elemento com gradiente nulo, pode-se gerar uma nova discretização de forma a eliminar o problema. Logo, o sistema é assintoticamente estável e o alvo é sempre atingido num tempo finito. Deve-se ressaltar que uma prova mais rigorosa deveria observar o fato da lei de controle em (4.12) possuir um chaveamento na fronteira da região de alvo, e portanto existe uma descontinuidade nesta fronteira. A abordagem desta descontinuidade está fora do escopo desta dissertação, uma vez que

este fato não acarreta em problemas para a navegação do robô.

4.4 Analogia com a Eletrostática

A inspiração dos potenciais artificiais apresentados vêm do eletromagnetismo, e pode-se fazer uma analogia perfeita entre o problema de navegação e o problema de eletrostática num meio dielétrico isotrópico [Macedo, 1988]. O potencial artificial para navegação pode ser visto como um potencial escalar elétrico, uma vez que a equação para o cálculo do potencial escalar elétrico é a equação de Laplace (4.3). O campo elétrico $\mathbf{E}$ é definido por:

$$\mathbf{E} = -\nabla\phi\,, \tag{4.14}$$

onde ϕ é o potencial escalar elétrico. Assim, a velocidade do robô faz uma analogia perfeita com o campo elétrico. Uma vez definidas as condições de contorno iguais para os dois problemas, a solução do problema eletromagnético é a mesma do problema robótico a não ser por um fator de escala. Uma conclusão interessante que se chega através desta analogia é que a metodologia proposta é imune ao problema de ciclos limite. Este problema aparece em algumas abordagens como [Fox et al., 1997, Brock and Khatib, 1999] e é resolvido em trabalhos como [Xu, 1999]. Um ciclo limite é uma trajetória fechada onde o robô permanece navegando infinitamente, e portanto não atingindo o alvo. Sabe-se, pela Lei de Faraday [Macedo, 1988], que o campo elétrico é irrotacional para o caso eletrostático. Por irrotacional entende-se:

$$\nabla \times \mathbf{E} = 0\,, \tag{4.15}$$

onde $\nabla \times \mathbf{E}$ é o rotacional do campo elétrico. O rotacional do campo em um dado ponto p pode ser interpretado fisicamente como uma medida da circulação, por unidade de área, do campo em torno deste ponto p. Portanto, a circulação ser zero para todos os pontos do domínio indica a não existência de trajetórias fechadas neste campo e conseqüentemente a não existência de ciclos limite.

Até o momento, nesta dissertação, o problema de navegação foi apresentado e uma solução para o mesmo foi proposta. A abordagem proposta é baseada na solução da equação de Laplace, e portanto necessita-se de um método para resolvê-la. O Método de Elementos Finitos tem sido aplicado com sucesso em diversas áreas onde os problemas são modelados por meio de equações diferenciais parciais, como em engenharia mecânica, civil e eletromagnetismo. Portanto, a aplicação deste método na metodologia proposta é promissora. No próximo capítulo discutem-se as características deste método, que, pelo conhecimento do autor, tem neste trabalho sua primeira utilização em navegação robótica.

Capítulo 5

Método de Elementos Finitos

> Nas questões matemáticas não se compreende a incerteza nem a dúvida, assim como tampouco se podem estabelecer distinções entre verdades médias e verdades de grau superior.
>
> *David Hilbert (1862–1943)*

No Capítulo 4 apresentou-se uma metodologia para resolver o problema de navegação de robôs móveis baseada na solução da equação de Laplace. Portanto, para que esta metodologia possa ser aplicada é necessário que se tenha um método para resolver esta equação. No presente capítulo apresenta-se o *Método de Elementos Finitos* (MEF) [Hughes, 2000] que se constitui em um método numérico poderoso para a solução de equações diferenciais parciais. A principal vantagem deste método frente aos demais, como, por exemplo, o *Método de Diferenças Finitas* (MDF) [Burden et al., 1978], é o fato de que o MEF permite uma representação exata mais eficiente de domínios que têm contornos com geometrias complexas. Ao se desejar que robôs móveis naveguem em ambientes reais, a questão do tratamento de geometrias complexas é inerente ao problema.

Este capítulo procura mostrar, de forma resumida, uma seqüência lógica para a construção da solução aproximada da equação de Laplace via o MEF. Inicialmente, transforma-se o problema de sua forma forte para sua forma fraca. Depois, aplica-se o método de Galerkin para transformar o problema

na forma fraca para uma nova versão colocada num espaço de funções de dimensão finita. Após a discretização do domínio em elementos, o MEF oferece então uma maneira de se selecionar estas funções. Por fim, um sistema linear é obtido e ao resolvê-lo tem-se uma aproximação da solução desejada. No final do capítulo, um algoritmo eficiente de localização do robô nos elementos é proposto, para que se consiga aplicar os Elementos finitos na navegação de robôs, e portanto controlar o robô em tempo real.

5.1 Forma Forte e Forma Fraca

A Figura 5.1 ilustra um domínio $\Omega \subset \mathbb{R}^n$ e seu contorno $\Gamma = \Gamma_h \cup \Gamma_g$ onde se deseja resolver a equação de Laplace (4.3). Atribui-se condição de contorno de Neumann homogêneo (4.7) à Γ_h e condição de contorno de Dirichlet constante (4.4) à Γ_g. A formulação do problema a ser resolvido, na *forma forte*, é a seguinte:

(S) Determinar a função $\phi : \Omega \rightarrow \mathbb{R}$ que satisfaz:

$$\nabla^2 \phi = 0 \quad \text{em } \Omega \tag{5.1}$$

$$\phi = c \quad \text{em } \Gamma_g \tag{5.2}$$

$$\frac{\partial \phi}{\partial \mathbf{n}} = 0 \quad \text{em } \Gamma_h \tag{5.3}$$

Para solucionar a formulação apresentada, usando técnicas clássicas, a função ϕ deve ser contínua e, também, deve possuir derivadas parciais de primeira e segunda ordem contínuas. Por isso, essa forma é chamada forma forte. O Método de Diferenças Finitas, por exemplo, trabalha diretamente

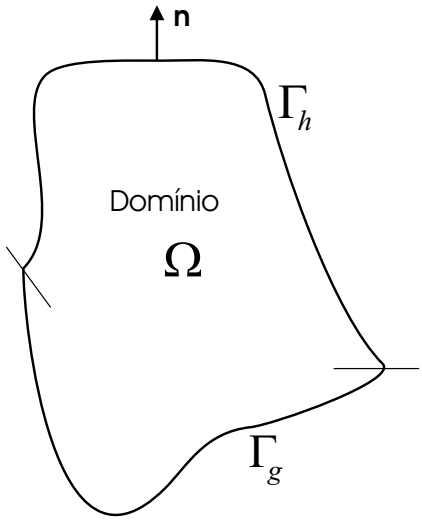

Figura 5.1: Domínio onde se deseja resolver a equação de Laplace e seus contornos. Aos contornos Γ_h e Γ_g são atribuídos, respectivamente, condição de contorno de Neumann homogêneo e condição de contorno de Dirichlet constante. O vetor $\mathbf{n}$ é normal ao contorno e aponta para fora do domínio.

com a forma forte do problema de contorno. Já o Método de Elementos Finitos requer uma formulação diferente, chamada *forma fraca*, que admite soluções com restrições mais fracas sobre suas derivadas.

Dois espaços de funções são utilizados na construção da forma fraca: o espaço das funções admissíveis S e o espaço das funções de teste U:

$$S = \{\phi \mid \phi \in H^1(\Omega) \ , \ \phi = c \text{ em } \Gamma_g\} \tag{5.4}$$

$$U = \{w \mid w \in H^1(\Omega) \ , \ w = 0 \text{ em } \Gamma_g\} \, , \tag{5.5}$$

onde $H^1(\Omega)$ é uma classe de funções onde elas e suas derivadas primeiras são quadraticamente integráveis. Esta classe de funções é um exemplo de *espaço de Sobolev* [Adams, 1975].

A forma fraca, também chamada de *variacional*, pode ser obtida utilizando o método dos resíduos ponderados [Becker et al., 1981]. Este método

consiste em montar o resíduo para a equação a ser resolvida no domínio, isto é, fazer o lado direito da equação ser zero, e então integrar este resíduo, ponderado por funções de peso, w:

$$\int_{\Omega}(\nabla^2\phi)w d\Omega = 0 \ , \qquad \forall w \in U \tag{5.6}$$

Como apresentado em [Macedo, 1988], o operador Laplaciano consiste do divergente do gradiente, isto é, $\nabla \cdot \nabla$. Considere a seguinte identidade:

$$w\nabla \cdot (\nabla\phi) = \nabla \cdot (w\nabla\phi) - \nabla w \cdot \nabla\phi \, , \tag{5.7}$$

onde w é uma função escalar.

Substituindo (5.7) em (5.6) obtém-se:

$$\int_{\Omega} \nabla \cdot (w\nabla\phi) d\Omega - \int_{\Omega}(\nabla w \cdot \nabla\phi) d\Omega = 0 \ , \qquad \forall w \in U \tag{5.8}$$

Pode-se manipular o primeiro termo de (5.8) considerando o *Teorema da Divergência* [Becker et al., 1981]. Este teorema estabelece a seguinte relação:

$$\int_{\Omega} \nabla \cdot \sigma d\Omega = \int_{\Gamma} \sigma \cdot \mathbf{n} d\Gamma \, , \tag{5.9}$$

onde σ é um campo vetorial.

Lembrando que $w\nabla\phi$ é um campo vetorial, aplica-se o teorema da divergência em (5.8):

$$\int_{\Gamma_g}(w\nabla\phi)\cdot\mathbf{n}d\Gamma + \int_{\Gamma_h}(w\nabla\phi)\cdot\mathbf{n}d\Gamma - \int_{\Omega}(\nabla w\cdot\nabla\phi)d\Omega = 0 \ , \qquad \forall w \in U \tag{5.10}$$

Uma vez que $w \in U$, o primeiro termo de (5.10) se anula. Tem-se também

que:

$$\nabla\phi \cdot \mathbf{n} = \frac{\partial\phi}{\partial\mathbf{n}} \tag{5.11}$$

Portanto, considerando as Equações (5.11) e (5.3), o segundo termo de (5.10) também se anula. A formulação na forma fraca é , então, a seguinte:

(W) Determinar a função $\phi \in S$ que satisfaz:

$$\int_\Omega (\nabla w \cdot \nabla\phi) d\Omega = 0 \ , \qquad \forall w \in U \tag{5.12}$$

Uma forma alternativa, porém mais complexa, de se obter a equação da forma fraca para problemas de contorno é através da minimização de funcionais [Carey and Oden, 1981c]. No caso da equação de Laplace, pode-se utilizar o seguinte funcional:

$$F = \int_\Omega \|\nabla\phi\|^2 d\Omega \,, \tag{5.13}$$

Uma última observação é que o fato de trabalhar com a forma fraca não altera o problema. Como apresentado em [Carey and Oden, 1981c] as formulações nas formas fraca e forte são equivalentes.

5.2 Método de Galerkin

O *Método de Galerkin* [Hughes, 2000] consiste de um método para se obter soluções aproximadas para problemas de contorno formulados na forma fraca. Mais especificamente, o objetivo do método é transformar o problema da forma fraca para a forma de Galerkin, onde o problema é colocado num

espaço de funções de dimensão finita. A seguir, aplica-se sobre a forma fraca (5.12) os três passos necessários para transformá-la para a forma de Galerkin:
Passo 1: Construir uma aproximação de dimensão finita para os espaços S e $U \Rightarrow S^h$ e U^h, onde

$$S^h \subset S \tag{5.14}$$

$$U^h \subset U \tag{5.15}$$

Se $\phi^h \in S^h$ e $w^h \in U^h$, então:

$$\phi^h(\Gamma_g) = c \tag{5.16}$$

$$w^h(\Gamma_g) = 0 \tag{5.17}$$

Passo 2: Assuma uma coleção de funções de U^h conhecidas. Para cada membro $y^h \in U^h$, construir uma função $\phi^h \in S^h$, usando:

$$\phi^h = y^h + c^h\,, \tag{5.18}$$

onde c^h satisfaz a condição de Dirichlet:

$$c^h = c \qquad \text{em } \Gamma_g \tag{5.19}$$

Então:

$$\phi^h = y^h + c^h = 0 + c = c \qquad \text{em } \Gamma_g \tag{5.20}$$

Portanto, (5.18) constitui uma definição de S^h. É interessante observar que, a não ser por c^h, S^h e U^h são compostos por coleções idênticas de funções.

Passo 3: Reescrever a forma fraca em termos de ϕ^h e w^h.

$$\int_\Omega (\nabla w^h \cdot \nabla \phi^h) d\Omega = 0 \ , \qquad \forall w^h \in U^h \tag{5.21}$$

Substituindo (5.18) em (5.21) chega-se à forma de Galerkin:

(G) Determinar $\phi^h = y^h + c^h$, onde $y^h \in U^h$ e $c^h \in S^h$, tal que:

$$\int_\Omega (\nabla w^h \cdot \nabla y^h) d\Omega = -\int_\Omega (\nabla w^h \cdot \nabla c^h) d\Omega \ , \qquad \forall w^h \in U^h \tag{5.22}$$

Note que a forma de Galerkin é apenas uma versão da forma fraca colocada em termos de uma coleção de funções de dimensão finita U^h.

5.3 Solução via Elementos Finitos

Para se obter uma solução aproximada de um problema de contorno, formulado na forma fraca, via o método de Galerkin, necessita-se de alguma maneira sistemática de determinação das funções de U^h e S^h. O Método de Elementos Finitos fornece esta maneira de forma adequada. O primeiro passo do MEF é a divisão do domínio de solução em pequenos sub-domínios chamados elementos finitos. Esse processo de discretização pode ser feito utilizando elementos de diferentes tamanhos, o que torna o método poderoso e flexível. Regiões onde ocorrem maiores variações do gradiente da solução podem ser representadas por uma maior densidade de elementos que regiões onde a variação é menor. Dessa forma, ambientes de geometria complexa são melhor representados por este método que por outros métodos como o de Diferenças Finitas. O segundo passo do MEF é a construção de um sistema

linear, obtido a partir da forma de Galerkin do problema e da discretização, cuja solução aproxima a solução desejada do problema em alguns pontos do domínio. O último passo do MEF consiste da obtenção da solução em todos os pontos do domínio, por meio de interpolação.

A subseção a seguir discute a questão da discretização do domínio, que do ponto de vista prático pode determinar se o método encontrará uma boa aproximação da solução ou não. Subseções posteriores apresentam a construção do sistema linear e as funções de interpolação que são utilizadas pelo método. Por fim, a última subseção descreve como adicionar condições de contorno periódicas ao método.

5.3.1 Discretização do Domínio

Embora diferentes formas de elementos possam ser utilizadas no método, os elementos triangulares, para domínios bidimensionais, e os elementos tetraédricos, para domínios tridimensionais, são os elementos mais utilizados. Encontra-se, tanto na literatura [Shewchuk, 1997], quanto em implementações computacionais gratuitas [Shewchuk, 1996, Si, 2004], algoritmos eficientes e robustos para a divisão do domínio de solução em elementos triangulares e tetraédricos. Por esta razão, optou-se neste trabalho por utilizar estes dois tipos de elementos.

O conjunto de todos os elementos que compõem a discretização do domínio é chamado de *malha*. Por sua vez, uma intersecção entre arestas de elementos da malha é chamada de *nó*. A Figura 5.2 mostra uma malha de triângulos para um domínio em duas dimensões, onde se observa paredes, um obstáculo em forma de U no centro, e uma região denominada alvo. Este ambiente foi utilizado em alguns experimentos realizados no presente trabalho, cujos resultados são apresentados no Capítulo 6.

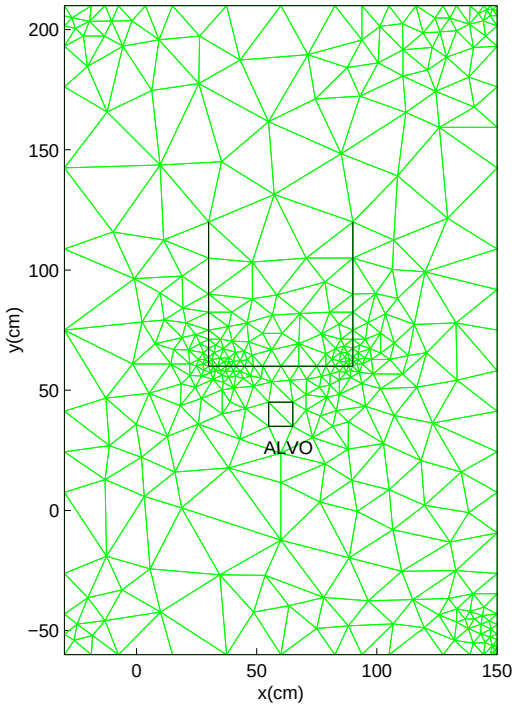

Figura 5.2: Discretização típica de um ambiente bidimensional em uma malha de triângulos.

Conforme apresentado em [Shewchuk, 2002], a qualidade da aproximação ϕ^h obtida utilizando MEF, e a própria convergência do método, estão fortemente vinculadas à qualidade da malha. Mais especificamente, uma malha torna-se ruim ao conter elementos com ângulos muito pequenos ou muito grandes. Em [de Berg et al., 2000] mostra-se que triangulações ótimas de um conjunto de pontos no plano, do ponto de vista de maximizar o mínimo ângulo de toda a triangulação, são *Triangulações de Delaunay*. Este tipo especial de triangulação é definido abaixo [de Berg et al., 2000]:

Definição 5.1 (Triangulação de Delaunay) *Dado um conjunto de pontos $\mathcal{Q}$, no plano, define-se $\mathcal{D}$ como Triangulação de Delaunay de $\mathcal{Q}$ se e somente se o círculo circunscrito a cada triângulo de $\mathcal{D}$ não contém qualquer outro ponto de $\mathcal{Q}$ em seu interior.*

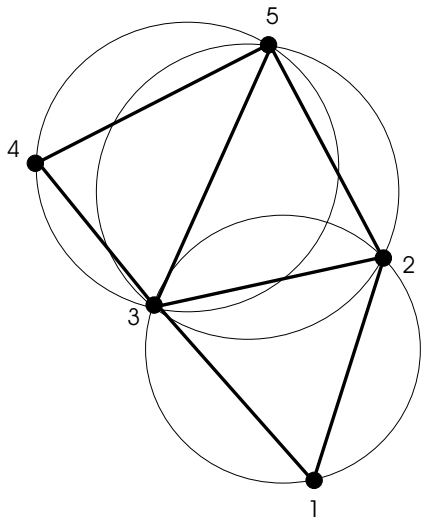

Figura 5.3: Triangulação de Delaunay para um conjunto de 5 pontos no plano.

A Figura 5.3 ilustra uma triangulação de Delaunay para um conjunto de 5 pontos no plano. Nesta figura, é possível observar os círculos circunscritos a cada triângulo e verificar que não há nenhum ponto no interior destes círculos. A extensão direta da triangulação de Delaunay para domínios em três dimensões é a *Tetraedralização de Delaunay*. Neste caso, os pontos não estão mais no plano e ao invés de se ter círculos circunscritos a triângulos, tem-se esferas circunscritas a tetraedros. Analogamente, não pode haver pontos do conjunto no interior das esferas. É interessante observar que as propriedades de otimalidade das triangulações de Delaunay são preservadas nestas tetraedralizações.

Malhas adequadas para o MEF não podem ser obtidas apenas por simples triangulações ou tetraedralizações de Delaunay. O problema é que os domínios onde se deseja utilizar o método de elementos finitos não são constituídos de um conjunto de pontos, mas sim de fronteiras compostas de faces, segmentos e curvas. Estas fronteiras têm que ser respeitadas pelos elemen-

tos, isto é, um elemento não pode atravessá-las. A solução deste problema é dada por métodos de triangulação ou tetraedralização de Delaunay com restrições [Shewchuk, 1997] que, geralmente, são implementadas utilizando o *Refinamento de Delaunay* [Carey and Oden, 1981b, Shewchuk, 1997]. Este refinamento, além de garantir que as restrições sejam satisfeitas, pode ser, também, utilizado para melhorar a qualidade da malha, isto é, fazer com que os elementos sejam o mais próximo de equiláteros quanto possível. Mais especificamente, o refinamento de Delaunay consiste em manter uma triangulação ou tetraedralização de Delaunay ao mesmo tempo em que se insere vértices adicionais na malha. As posições onde se inserem estes vértices são escolhidas de forma a forçar a conformidade com as fronteiras e melhorar a qualidade da malha. Dessa forma, malhas não-estruturadas adequadas podem ser obtidas.

Os principais conceitos envolvidos nas técnicas de discretização, que são utilizadas por métodos de elementos finitos, foram apresentados aqui, nesta subseção. A apresentação de algoritmos e de maiores detalhes sobre este assunto fogem ao escopo desta dissertação, e podem ser encontrados em [Carey and Oden, 1981c, Shewchuk, 1996, Shewchuk, 1997, Si, 2004].

Para finalizar, um último conceito, que é útil quando se deseja comparar o grau de discretização de diferentes malhas, é o chamado *tamanho de "grid"*. Esta grandeza corresponde ao valor do raio máximo dos círculos ou esferas circunscritos aos elementos da malha.

5.3.2 Equações Matriciais

A subseção anterior apresentou conceitos importantes para a discretização eficiente do domínio. Após a realização desta discretização, o MEF permite que se calcule, através da solução de um sistema linear, o valor da solução

aproximada ϕ^h em cada nó da malha. Nesta subseção, apresenta-se a construção deste sistema linear para resolver o problema na forma de Galerkin (5.22).

Seja η o conjunto de todos os nós da malha. Seja, também, η_g o conjunto de nós que estão dispostos na fronteira Γ_g, onde condições de Dirichlet são impostas. Assim, $\eta - \eta_g$ é o conjunto de nós onde não se conhece o valor das incógnitas, isto é, nós onde não estão impostas condições de Dirichlet. Um membro típico de U^h tem a forma:

$$w^h = \sum_{t \in \{\eta - \eta_g\}} z_t \psi_t(\mathbf{q}) \,, \tag{5.23}$$

onde $\mathbf{q}$ é um vetor, de dimensão n, que denota um ponto do domínio $\Omega \subset \mathbb{R}^n$. Tem-se, também, que z_t é uma constante e $\psi(\mathbf{q}) \in U^h$ é uma função de $\mathbf{q}$ chamada *função de base*, *função de forma*, ou *função de interpolação*. Conclui-se a partir de (5.23) que U^h tem dimensão igual ao número de nós pertencentes ao conjunto $\eta - \eta_g$.

Conforme (5.18), um membro de S^h é obtido pela soma de y^h e c^h, onde:

$$y^h = \sum_{t \in \{\eta - \eta_g\}} d_t \psi_t(\mathbf{q}) \tag{5.24}$$

$$c^h = \sum_{t \in \{\eta_g\}} c_t \psi_t(\mathbf{q}) \,, \tag{5.25}$$

onde c_t é o valor da condição de Dirichlet imposta ao nó t.

Substituindo as Equações (5.23), (5.24) e (5.25) na equação de Galerkin

(5.22) obtém-se:

$$\int_{\Omega}[(\sum_{t\in\{\eta-\eta_g\}} z_t\nabla\psi_t)\cdot(\sum_{l\in\{\eta-\eta_g\}} d_l\nabla\psi_l)]d\Omega = \\ = -\int_{\Omega}[(\sum_{t\in\{\eta-\eta_g\}} z_t\nabla\psi_t)\cdot(\sum_{l\in\{\eta_g\}} c_l\nabla\psi_l)]d\Omega\ , \qquad \forall z_t\in\mathbb{R} \quad (5.26)$$

Manipulando (5.26) chega-se a:

$$\sum_{t\in\{\eta-\eta_g\}} z_t \sum_{l\in\{\eta-\eta_g\}} [\int_{\Omega}(\nabla\psi_t\cdot\nabla\psi_l)d\Omega]d_l = \\ = -\sum_{t\in\{\eta-\eta_g\}} z_t \sum_{l\in\{\eta_g\}} [\int_{\Omega}(\nabla\psi_t\cdot\nabla\psi_l)d\Omega]c_l\ , \qquad \forall z_t\in\mathbb{R} \quad (5.27)$$

Como a expressão (5.27) deve ser válida para $\forall z_t \in \mathbb{R}$:

$$\sum_{l\in\{\eta-\eta_g\}} [\int_{\Omega}(\nabla\psi_t\cdot\nabla\psi_l)d\Omega]d_l = \\ = -\sum_{l\in\{\eta_g\}} [\int_{\Omega}(\nabla\psi_t\cdot\nabla\psi_l)d\Omega]c_l\ , \qquad \forall t\in\eta-\eta_g \qquad (5.28)$$

Considerando que a dimensão de $\eta-\eta_g$ é Λ, a expressão (5.28) corresponde ao sistema linear:

$$\mathbf{Md} = \mathbf{f}\,, \qquad (5.29)$$

onde a matriz $\mathbf{M}$ tem dimensão $\Lambda\times\Lambda$, e os vetores $\mathbf{d}$ e $\mathbf{f}$ têm dimensão Λ. Cada posição $\mathbf{M}_{tl}$ da matriz, a qual está associada a um par de nós $(t\in\eta-\eta_g, l\in\eta-\eta_g)$, é dada pela equação:

$$\mathbf{M}_{tl} = \int_{\Omega}(\nabla\psi_t\cdot\nabla\psi_l)d\Omega\,, \qquad (5.30)$$

Por sua vez, o vetor $\mathbf{f}$ é quem sofre a influência das condições de contorno

de Dirichlet e seus elementos são da forma:

$$\mathbf{f}_t = -\sum_{l \in \{\eta_g\}} [\int_\Omega (\nabla \psi_t \cdot \nabla \psi_l) d\Omega] c_l. \tag{5.31}$$

Uma vez determinada a solução $\mathbf{d}$ do sistema (5.29), a solução final ϕ^h do problema na forma de Galerkin (5.22) pode ser determinada por:

$$\phi^h = \sum_{t \in \{\eta - \eta_g\}} d_t \psi_t + \sum_{t \in \{\eta_g\}} c_t \psi_t. \tag{5.32}$$

O Método de Elementos Finitos possibilita que o sistema mostrado em (5.29) seja resolvido de forma eficiente. Isso se deve às funções de base ψ que o método fornece. Essas funções são tais que a matriz $\mathbf{M}$ é esparsa. Além disso, em [Hughes, 2000] prova-se que esta matriz também é simétrica e positiva definida. Devido a todas essas características, o sistema linear pode ser resolvido de forma eficiente utilizando o *Método dos Gradientes Conjugados* [Silvester, 1990, Shewchuk, 1994].

As duas próximas subseções destinam-se a descrever as referidas funções de base para elementos triangulares e para elementos tetraédricos, que, como dito anteriormente, foram os elementos utilizados no desenvolvimento dos experimentos deste trabalho. Em [Carey and Oden, 1981a] discutem-se outros tipos de elementos.

5.3.3 Elementos triangulares

Como visto anteriormente, através do sistema linear (5.29) e da Equação (5.32) obtém-se o valor de ϕ^h nos nós da malha. O MEF permite que se calcule o valor de ϕ^h no interior de cada elemento através de interpolação. Nesta subseção, considera-se que o domínio Ω do problema é bidimensional e

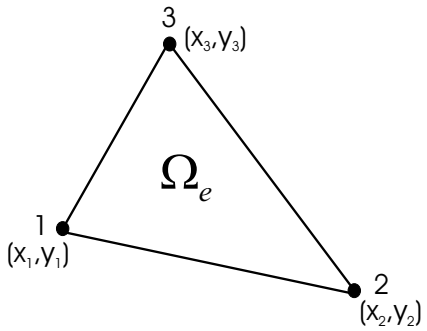

Figura 5.4: Ilustração de um elemento triangular. Define-se Ω_e como o domínio no interior do elemento.

aborda-se o elemento triangular. A Figura 5.4 ilustra tal elemento. Para realizar a interpolação no interior de um dado elemento, utilizam-se as funções de base ψ associadas a cada um dos nós do elemento da seguinte forma:

$$\phi^h(\Omega_e) = \sum_{l=1,2,3} \phi_l^h \psi_l \,, \tag{5.33}$$

onde ϕ_l^h é o valor de ϕ^h no nó l, e ψ_l é a função de base associada ao nó l.

Para que (5.33) seja satisfeita, o valor da função de base deve ser 1 no nó ao qual ela está associada, e deve ser 0 nos demais nós, isto é:

$$\psi_l(\mathbf{q}_l) = 1 \,, \tag{5.34}$$

$$\psi_l(\mathbf{q}_t) = 0 \,, \tag{5.35}$$

onde $\mathbf{q}_l$ é um vetor com as coordenadas do nó l, e $\mathbf{q}_t$ é um vetor com as coordenadas de um nó $t \neq l$ qualquer.

As funções de base podem ser polinômios de qualquer ordem. A construção destes polinômios pode ser feita utilizando-se *polinômios de Lagrange* [Becker et al., 1981]. O grande problema, como dito em [Hughes, 2000], é que quanto maior a ordem, maior é o custo computacional do MEF. Por outro

lado, utilizando-se polinômios de ordens superiores, aproximações ϕ^h mais precisas da solução desejada ϕ são obtidas. No presente trabalho, elementos triangulares lineares, com funções de base de primeira ordem, são utilizados. Estes elementos foram os primeiros a serem concebidos [Courant, 1943]. Esta escolha se deve ao tipo de aplicação à qual o trabalho se dedica. Para o caso de robótica, não se necessita de grande precisão da aproximação, uma vez que a solução do problema não representa um fenômeno físico real, onde se desejaria saber o resultado com a maior precisão possível. Portanto, optou-se por trabalhar na situação de mais baixo custo computacional.

A Figura 5.5 ilustra uma função de base linear para um elemento triangular. A equação que descreve as funções de base lineares, associadas aos nós 1, 2 e 3, para o elemento de domínio Ω_e, é:

$$\psi_l(\mathbf{q}) = \begin{cases} \alpha_l + \beta_l x + \gamma_l y & \text{se } \mathbf{q} \in \Omega_e \\ 0 & \text{se } \mathbf{q} \notin \Omega_e \end{cases} , \tag{5.36}$$

onde $l = 1, 2, 3$. As constantes α_l, β_l e γ_l podem ser calculadas ao se impor as restrições (5.34) e (5.35) à Equação (5.36). Mais especificamente, obtém-se um sistema com três equações e três incógnitas, que correspondem às constantes. Para ψ_1, as constantes são:

$$\alpha_1 = \frac{1}{D}(x_2 y_3 - x_3 y_2) , \tag{5.37}$$

$$\beta_1 = \frac{1}{D}(y_2 - y_3) , \tag{5.38}$$

$$\gamma_1 = \frac{1}{D}(x_3 - x_2) , \tag{5.39}$$

e para ψ_2 e ψ_3 as constantes podem ser obtidas pelas mesmas expressões anteriores, bastando fazer a permutação cíclica dos índices $(1, 2, 3)$. A constante

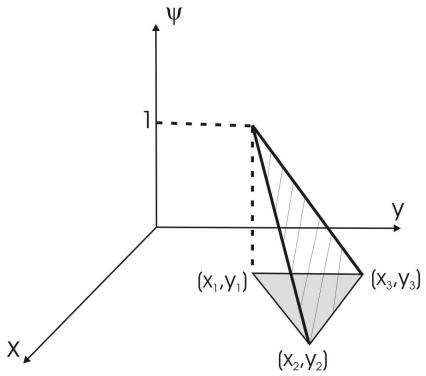

Figura 5.5: Função de base linear para um elemento triangular.

D, que aparece nas expressões, corresponde ao determinante:

$$D = det \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}. \tag{5.40}$$

Deve-se observar que as funções dadas por (5.36) são calculadas por elemento, e portanto têm validade apenas no elemento em questão, de domínio Ω_e. Nos demais elementos da malha, $\psi_l = 0$. Portanto, a construção do sistema linear (5.29) torna-se extremamente eficiente. As posições da matriz $\mathbf{M}_{tl}$ são calculadas por integrais no domínio Ω, de produtos de pares de funções de base associadas a pares de nós (t, l), conforme (5.30). Devido às características das funções de base apresentadas, não se necessita integrar todo o domínio para se obter $\mathbf{M}_{tl}$, e sim apenas os domínios Ω_e correspondentes aos elementos que contribuem para o cálculo, isto é, os que possuem o par de nós (t, l). De forma análoga, as integrais em (5.31), utilizadas para o cálculo do vetor $\mathbf{f}$, só precisam ser integradas nos elementos que possuem os pares de nós correspondentes. Além disso, as posições da matriz $\mathbf{M}$, relativas a pares de nós que não compartilham nenhum elemento, são nulas o

que torna a matriz esparsa, como dito na Subseção 5.3.2.

Para finalizar esta subseção, discute-se o gradiente de ϕ^h. Pelo fato de se utilizar uma interpolação linear no interior de cada elemento, o gradiente será constante ao longo de um dado elemento. Derivando-se a Equação (5.33), chega-se aos componentes do gradiente, $\nabla\phi^h = (\frac{\partial\phi^h}{\partial x}, \frac{\partial\phi^h}{\partial y})$, que no interior de um dado elemento de domínio Ω_e, são dados por:

$$\frac{\partial\phi^h}{\partial x}(\Omega_e) = \sum_{l=1,2,3} \beta_l \phi_l^h \,, \tag{5.41}$$

$$\frac{\partial\phi^h}{\partial y}(\Omega_e) = \sum_{l=1,2,3} \gamma_l \phi_l^h. \tag{5.42}$$

5.3.4 Elementos tetraédricos

Como exposto anteriormente, este trabalho também utiliza domínios em três dimensões (x, y, θ). Os elementos tetraédricos lineares são extensões diretas dos elementos triangulares lineares, apresentados na subseção anterior, para domínios tridimensionais. Todas as observações feitas para elementos triangulares lineares são válidas para os elementos tetraédricos lineares, inclusive a razão de utilizá-los ao invés de utilizar elementos de ordem superior. A Figura 5.6 apresenta um elemento tetraédrico.

Analogamente à Equação (5.33), tem-se uma expressão para a aproximação ϕ^h no interior do tetraedro:

$$\phi^h(\Omega_e) = \sum_{l=1,2,3,4} \phi_l^h \psi_l \,, \tag{5.43}$$

onde ϕ_l^h é o valor de ϕ^h no nó l, e ψ_l é a função de base associada ao nó l.

As restrições (5.34) e (5.35) também são válidas. Considerando um dado elemento, tem-se a seguinte expressão para as funções de base associadas aos

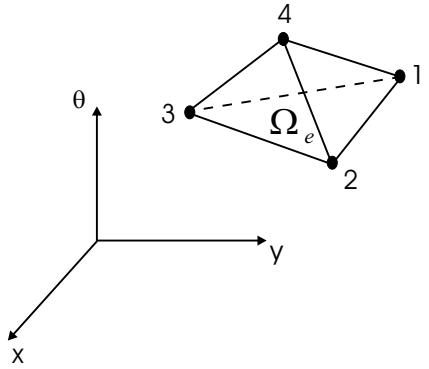

Figura 5.6: Ilustração de um elemento tetraédrico. Define-se Ω_e como o domínio no interior do elemento.

nós 1, 2, 3 e 4:

$$\psi_l(\mathbf{q}) = \begin{cases} \alpha_l + \beta_l x + \gamma_l y + \upsilon_l \theta & \text{se } \mathbf{q} \in \Omega_e \\ 0 & \text{se } \mathbf{q} \notin \Omega_e \end{cases} , \tag{5.44}$$

onde $l = 1, 2, 3, 4$.

O cálculo das constantes em (5.44) é feito pelas expressões:

$$\begin{aligned} \alpha_i &= \frac{(-1)^j}{D}[x_j(y_m\theta_n - y_n\theta_m) + x_m(y_n\theta_j - y_j\theta_n) \\ &\quad + x_n(y_j\theta_m - y_m\theta_j)] , \end{aligned} \tag{5.45}$$

$$\beta_i = \frac{(-1)^i}{D}[(y_m\theta_n - y_n\theta_m) + (y_n\theta_j - y_j\theta_n) + (y_j\theta_m - y_m\theta_j)] , \tag{5.46}$$

$$\gamma_i = \frac{(-1)^j}{D}[(x_m\theta_n - x_n\theta_m) + (x_n\theta_j - x_j\theta_n) + (x_j\theta_m - x_m\theta_j)] , \tag{5.47}$$

$$\upsilon_i = \frac{(-1)^i}{D}[(x_m y_n - x_n y_m) + (x_n y_j - x_j y_n) + (x_j y_m - x_m y_j)] , \tag{5.48}$$

onde $(i, j, m, n) \rightarrow (1, 2, 3, 4)$ ou uma permutação cíclica. A constante D é

dada por um determinante:

$$D = det \begin{pmatrix} 1 & x_1 & y_1 & \theta_1 \\ 1 & x_2 & y_2 & \theta_2 \\ 1 & x_3 & y_3 & \theta_3 \\ 1 & x_4 & y_4 & \theta_4 \end{pmatrix}. \tag{5.49}$$

A expressão (5.44), tal qual a expressão (5.36), é válida em um dado elemento. Tem-se, novamente, as mesmas conseqüências, já discutidas na subseção anterior, relacionadas com a matriz **M**, que é esparsa, simétrica e positiva definida.

Por fim, de forma análoga, o gradiente é constante no interior de cada tetraedro e tem-se as seguintes expressões para seu cálculo:

$$\frac{\partial \phi^h}{\partial x}(\Omega_e) = \sum_{l=1,2,3,4} \beta_l \phi_l^h, \tag{5.50}$$

$$\frac{\partial \phi^h}{\partial y}(\Omega_e) = \sum_{l=1,2,3,4} \gamma_l \phi_l^h, \tag{5.51}$$

$$\frac{\partial \phi^h}{\partial \theta}(\Omega_e) = \sum_{l=1,2,3,4} \upsilon_l \phi_l^h. \tag{5.52}$$

5.3.5 Adição de Condição de Contorno Periódica

Até o momento, a solução ϕ^h apresentada resolve um problema com condições de contorno de Dirichlet e Neumann homogêneo. Conforme o Capítulo 4, ao se tratar o problema em $\mathbb{R}^3$ (x, y, θ), devem ser impostas condições de contorno periódicas (4.10) aos planos $\theta = 0$ e $\theta = \frac{2\pi}{k}$. Esta condição pode ser facilmente adicionada ao MEF [Ida and Bastos, 1992], se a malha gerada garantir pares de nós periódicos, isto é, para cada nó em $(x, y, 0)$ existir um nó em $(x, y, \frac{2\pi}{k})$.

Suponha que existam pares de nós periódicos (t, t'). Para considerar a condição de periodicidade no problema, basta que sejam feitas algumas alterações na montagem da matriz $\mathbf{M}$ e do vetor $\mathbf{f}$ no sistema (5.29). Como os nós são periódicos, tem-se o mesmo valor de solução para os dois, isto é, $\phi^h(t) = \phi^h(t')$. Portanto, calcula-se a solução para apenas um dos nós, por exemplo t, e no final de todo cálculo atribui-se o valor calculado ao nó t'. As dimensões $\Lambda \times \Lambda$ de $\mathbf{M}$ e Λ de $\mathbf{f}$ são, então, modificadas para $(\Lambda - N_{par}) \times (\Lambda - N_{par})$ e $(\Lambda - N_{par})$, respectivamente, onde N_{par} é o número de pares periódicos existentes.

Como o cálculo agora é feito apenas para o nó t, e não para seu par t', então, deve-se modificar as Equações (5.30) e (5.31), referentes à montagem da matriz $\mathbf{M}$ e do vetor $\mathbf{f}$, de forma que a contribuição dos elementos que possuem o nó t' apareçam nas posições referentes ao nó t. Tem-se as seguintes expressões modificadas, para as posições da matriz e do vetor, referentes a nós periódicos:

$$\mathbf{M}_{tl} = \sum_{i=t,t'} \sum_{j=l,l'} \int_{\Omega} (\nabla \psi_i \cdot \nabla \psi_j) d\Omega \,, \tag{5.53}$$

onde t' é o par periódico de t, e l' é o par periódico de l. A Eq.(5.53) trata o caso mais geral, onde se tem t e l periódicos. Se apenas um dos nós for periódico, por exemplo t, obviamente tem-se $j = l$ e o somatório em j desaparece.

$$\mathbf{f}_t = - \sum_{i=t,t'} \sum_{l \in \{\eta_g\}} [\int_{\Omega} (\nabla \psi_i \cdot \nabla \psi_l) d\Omega] c_l \,, \tag{5.54}$$

onde c_l é o valor da condição de Dirichlet imposta ao nó $l \in \eta_g$.

Deve-se ressaltar que posições de $\mathbf{M}$ e $\mathbf{f}$, que não estão associadas a nós periódicos, continuam sendo calculadas por (5.30) e (5.31). A conclusão que se tira, do cálculo utilizando condições periódicas em $\theta = 0$ e $\theta = \frac{2\pi}{k}$, é que

o resultado obtido é o mesmo que se obteria se os planos $\theta = 0$ e $\theta = \frac{2\pi}{k}$ se tocassem e formassem, portanto, um só plano.

5.4 Aplicação dos Elementos Finitos para o Controle de Robôs

Tudo o que foi apresentado neste capítulo, até este momento, pode ser encontrado na literatura, e portanto, não é contribuição do presente trabalho. Esta seção apresenta, então, uma contribuição do trabalho: como aplicar os resultados obtidos anteriormente, via Elementos Finitos, na navegação robótica.

A metodologia de navegação proposta neste trabalho pode ser dividida em duas etapas:

1. Cálculo do potencial artificial ϕ^h e seu gradiente $\nabla\phi^h$: nesta etapa, constrói-se o espaço de configurações conforme o Capítulo 3, e resolve-se a equação de Laplace, utilizando o MEF conforme Capítulo 5, sujeita às condições de contorno apresentadas no Capítulo 4;

2. Aplicação da lei de controle (4.12) em tempo real: a lei de controle deve ser implementada na forma de um algoritmo de controle digital, onde a cada tempo de amostragem, t_a, a velocidade de referência, $u(q)$, do robô, é calculada por (4.12).

Até este ponto da dissertação, apresentou-se a implementação da etapa 1 da metodologia. Uma vez que o gradiente do potencial é calculado para cada elemento, conforme subseções 5.3.3 e 5.3.4, a implementação da etapa 2 consiste, basicamente, da solução do seguinte problema:

algoritmo 5.1 Encontra o elemento da malha de elementos finitos, no qual uma configuração q está inserida

Entradas: A configuração atual, q, do robô, a malha de elementos finitos ξ, e um elemento de busca inicial Y_0.
Saídas: O elemento Y, da malha ξ, onde q está inserido.

1: $T \leftarrow q$
2: $Y \leftarrow Y_0$
3: **enquanto** T é ponto externo a Y **faça**
4: $O \leftarrow$ um ponto interior qualquer de Y
5: $Y_1 \leftarrow$ um elemento de ξ, *vizinho* de Y, que é interceptado por $\overline{OT}$ {$\overline{OT}$ é o segmento que liga os pontos O e T (veja exemplo na fig. 5.7)}
6: $Y \leftarrow Y_1$
7: **fim enquanto**

Problema 5.2 *Dada uma configuração q do robô, obtida em tempo real para um tempo t_a, encontrar o elemento, Y, da malha de elementos finitos, no qual o robô está em t_a.*

Após a solução deste problema, o que se deve fazer é óbvio: utilizar o gradiente calculado na etapa 1, para o elemento Y, no cálculo da referência $u(q)$, através de (4.12). A localização do robô, ou seja, a obtenção precisa da configuração q do robô em t_a é , também, importante, mas está fora do escopo desta dissertação. Várias técnicas podem ser utilizadas para obter q, dependendo dos sensores disponíveis no robô [Dudek and Jenkin, 2000]. Supondo, então, que q é conhecido, propõe-se o Algoritmo 5.1, para resolver o Problema 5.2 de maneira eficiente para domínios em $\mathbb{R}^2$ e em $\mathbb{R}^3$.

Para um melhor entendimento do Algoritmo 5.1, a Figura 5.7 apresenta um exemplo em duas dimensões da primeira iteração do algoritmo. Deve ficar claro que elementos *vizinhos* são aqueles que compartilham uma aresta, para o caso de triângulos, ou uma face, para o caso de tetraedros. Assim, o número máximo de vizinhos de um triângulo é 3, e o de um tetraedro é 4. Este algoritmo, faz algo mais inteligente do que simplesmente testar se q é

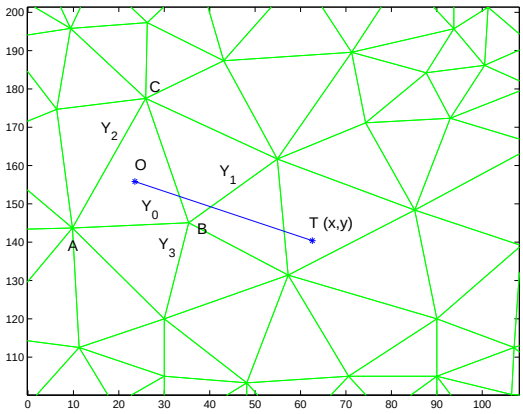

Figura 5.7: Exemplo num domínio bidimensional: Iteração inicial do Algoritmo 5.1, para localização do elemento da malha triangular, onde a configuração $q = (x, y)$ está. O elemento inicial de busca é Y_0 e a variável T armazena q.

interior para todos elementos da malha. Ao invés disso, o algoritmo caminha ao longo da malha testando apenas elementos que estejam sempre na direção de q, de forma a convergir rapidamente para o elemento procurado. Mais especificamente, dado um elemento de busca atual, por exemplo Y_0, e um ponto qualquer O, no interior deste elemento, o algoritmo avança na próxima iteração para o vizinho, Y_1, de Y_0 que seja interceptado por $\overline{OT}$ (veja Figura 5.7). Como $\overline{OT}$ é um segmento de reta, que une um ponto do interior do elemento atual à configuração q, o algoritmo sempre caminha na direção do elemento procurado.

O Algoritmo 5.1 é executado a todo intervalo de amostragem. Um elemento inicial Y_0 necessita ser fornecido para iniciar o processo. Ao se executar o algoritmo pela primeira vez, um elemento inicial Y_0 pode ser escolhido de forma aleatória. Nas execuções seguintes, entretanto, sugere-se escolher como Y_0 o elemento que foi encontrado como resposta na última execução. Esta escolha se baseia no fato do intervalo de amostragem ser pequeno quando comparado com os tempos envolvidos no deslocamento de um robô real. Dessa

forma, o elemento inicial quase sempre, ou coincidirá com o elemento desejado, ou estará próximo do mesmo, e portanto o processo de busca se torna ainda mais rápido. Esse algoritmo foi bem sucedido em todos os experimentos realizados. O próximo capítulo apresenta alguns destes experimentos.

Capítulo 6

Resultados Experimentais

> São fúteis e cheias de erros as ciências que não nasceram da experimentação, mãe de todo conhecimento.
>
> *Leonardo da Vinci (1452–1519)*

Este capítulo apresenta os resultados obtidos ao se aplicar a metodologia proposta, que é baseada na solução da equação de Laplace (4.3) via método de elementos finitos, na navegação de robôs em espaços de configurações no $\mathbb{R}^2$ (x, y) e no $\mathbb{R}^3$ (x, y, θ). Para os experimentos em duas dimensões utilizou-se o programa *Triangle* [Shewchuk, 1996] para gerar as malhas de triângulos, e o programa FEMM (Finite Element Method Magnetics [Meeker, 2004]), versão 4.0, para calcular as soluções de elementos finitos. Por sua vez, nos experimentos em três dimensões, as malhas de tetraedros foram geradas a partir do programa Tetgen [Si, 2004], versão 1.3. Já o cálculo de elementos finitos foi feito por um programa desenvolvido no *framework* de aplicação orientado a objetos BRFEM (Brazil Finite Elements [BRFEM, 2005]), versão 1.1. Este *framework* permite que se desenvolvam programas de elemento finitos reutilizáveis baseados na linguagem C++.

Como a metodologia foi desenvolvida para a navegação em ambientes estacionários, os mapas dos ambientes utilizados nos exemplos apresentados eram conhecidos a priori. Além disso, nenhum obstáculo presente nos mapas tinha a habilidade de se locomover. Para a obtenção da maior parte

dos resultados, utilizou-se uma plataforma de testes baseada em robôs reais holonômicos. As principais características desta plataforma são discutidas na Seção 6.1 (maiores detalhes são apresentados no Apêndice A). A lei de controle utilizada para controlar os robôs nos experimentos corresponde à Equação (4.12), quando $\alpha = 1$. Assim, tem-se o negativo do gradiente normalizado como velocidade de referência. As Seções 6.2 e 6.3 apresentam exemplos em duas e em três dimensões, respectivamente. No endereço http://www.cpdee.ufmg.br/∼lucpim/dissertacao podem-se encontrar vídeos com alguns dos resultados apresentados.

6.1 Plataforma de Testes

Os resultados apresentados neste capítulo foram obtidos utilizando uma plataforma robótica de baixo custo e alta versatilidade. Esta plataforma foi desenvolvida no Laboratório de Visão e Robótica (VERLAB) da Universidade Federal de Minas Gerais. O sistema é constituído de robôs holonômicos controlados remotamente. Cada robô holonômico tem três rodas omnidirecionais montadas numa base de acrílico. As rodas omnidirecionais possuem rolamentos montados de tal forma que estas possam rolar livremente em direções perpendiculares ao seu eixo e possam ter a sua velocidade controlada na direção em que estejam apontando. Por isso, os robôs holonômicos podem se mover, instantaneamente, em qualquer direção, e seus movimentos são descritos por modelos cinemáticos simples da forma $\dot{q} = u(q)$, tal qual Equação (4.11). Um servo de aeromodelismo, modificado para rotação, traciona cada roda. Os três servos recebem sinais de controle de um computador remoto a partir de um par transmissor/receptor. Os sinais provenientes do computador são convertidos para sinais PWM (*"pulse-width-modulation"*)

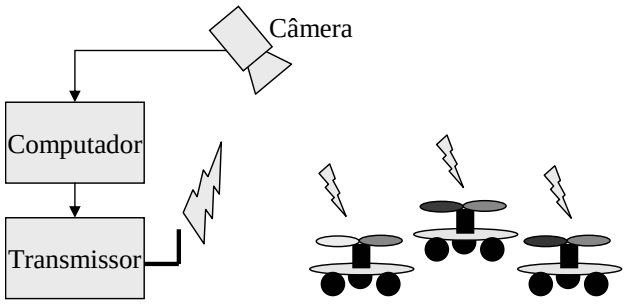

Figura 6.1: Diagrama do sistema de testes.

através da interface de um microcontrolador PIC®, o qual comunica com o computador por meio da porta serial RS232.

Os robôs não têm sensores locais, portanto a localização dos mesmos é feita por um sistema de visão computacional (veja Figura 6.1), o qual visualiza o espaço de trabalho do robô. Mais especificamente, uma transformação, da posição (u, v) do robô no plano de imagem da câmera para o referencial (x, y) do espaço de trabalho, é calculada via uma matriz de homografia. Esta matriz é obtida via calibração [Forsyth and Ponce, 2003]. Para simplificar o processo de localização, cada robô tem dois marcadores coloridos, colocados lado a lado, que permitem, ainda, a estimação da orientação θ dos robôs. A Figura 6.2 apresenta uma foto dos robôs. A velocidade máxima dos robôs é de aproximadamente 7 cm/s, e o erro de localização é menor que 2 cm. Maiores detalhes deste sistema são apresentados no Apêndice A e também em [Pereira et al., 2004b].

Nos exemplos apresentados a seguir, o sistema de controle de robôs foi implementado em linguagem C++. O computador utilizado foi um Pentium II, 300 MHz e 128 MB de memória RAM, com sistema operacional MS Windows 2000. Para a aquisição das imagens utilizou-se uma placa Matrox Meteor II® e uma câmera CCD colorida com resolução de 640×480 pixels. Nesta configuração, o sistema é executado a 10 quadros por segundo. É im-

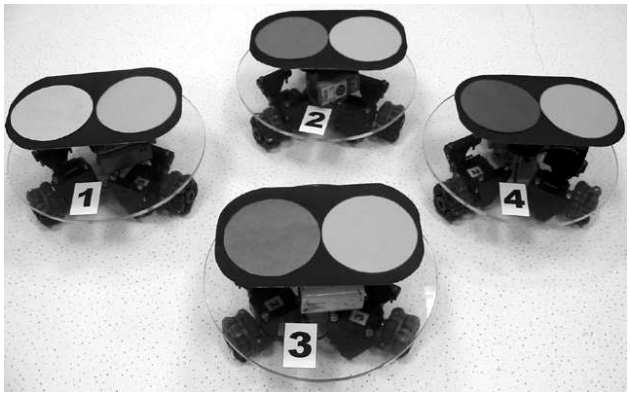

Figura 6.2: Foto dos robôs holonômicos utilizados.

portante ainda ressaltar que a localização dos robôs nas malhas de elementos finitos foi feita, com sucesso, pela implementação do Algoritmo 5.1 proposto na Seção 5.4.

6.2 Resultados para Robôs com 2 Graus de Liberdade

A grande vantagem da plataforma utilizada para a realização dos experimentos é sua versatilidade quanto a forma do robô. A base de acrílico, na qual as rodas são montadas, definem o corpo do robô e portanto sua forma geométrica. Assim, através da simples confecção de novas bases, são obtidos robôs de novas geometrias. Para os resultados em $\mathbb{R}^2$, utilizou-se uma base circular de raio igual a 10 cm, tal qual apresentado na foto da Figura 6.2. Nesta seção, apresentam-se as trajetórias desenvolvidas, por um dos robôs holonômicos, em dois espaços de trabalho diferentes. Em acordo com o Capítulo 4, para ambos os espaços de trabalho, as condições de contorno do alvo foram definidas com valor igual a zero, enquanto as demais condições, situadas no contorno dos obstáculos, foram definidas com valor igual a 100.

É importante mencionar também, que para os experimentos desta seção não se preocupou em construir espaços de configurações. Tratou-se os espaços de trabalho como os próprios espaços de configurações, e portanto, as trajetórias apresentadas se referem às trajetórias do centro do robô.

6.2.1 Exemplo 1

Neste exemplo, adotou-se um espaço de trabalho constituído de quatro paredes e um obstáculo em forma de U no centro. O alvo foi posicionado bem próximo ao obstáculo, o que para muitas metodologias baseadas em potencial artificial constitui um grave problema. Este problema é a já mencionada presença de mínimos locais na função de potencial. Conforme foi dito anteriormente nesta dissertação, este problema não aparece na atual metodologia e os resultados apresentados nesta seção confirmam este fato. A Figura 6.3 (a) apresenta o espaço de trabalho do robô.

A Figura 6.3 (b) mostra uma trajetória típica realizada pelo centro do robô quando o espaço de trabalho é discretizado com tamanho de grid igual a $0,3$ metros. Lembre-se do Capítulo 5 que o tamanho do grid corresponde ao raio máximo, dentre os raios dos círculos circunscritos aos triângulos da malha. Portanto, esta medida fornece uma idéia do grau de discretização do ambiente. Além disso, a malha gerada neste caso apresentou 386 nós. Observa-se na Figura 6.3 (b), setas no interior de cada triângulo. Estas setas são o negativo do gradiente normalizado, que, conforme foi mostrado Capítulo 5, é constante no interior de cada elemento. A figura em questão é didática, pois apresenta de forma exata como o método de navegação funciona: a trajetória seguida pelo robô dentro de cada elemento tem sua tangente paralela às setas de gradiente. Em alguns momentos, esta observação sobre a trajetória parece não ser válida, mas isso se deve à dinâmica do robô e ruídos

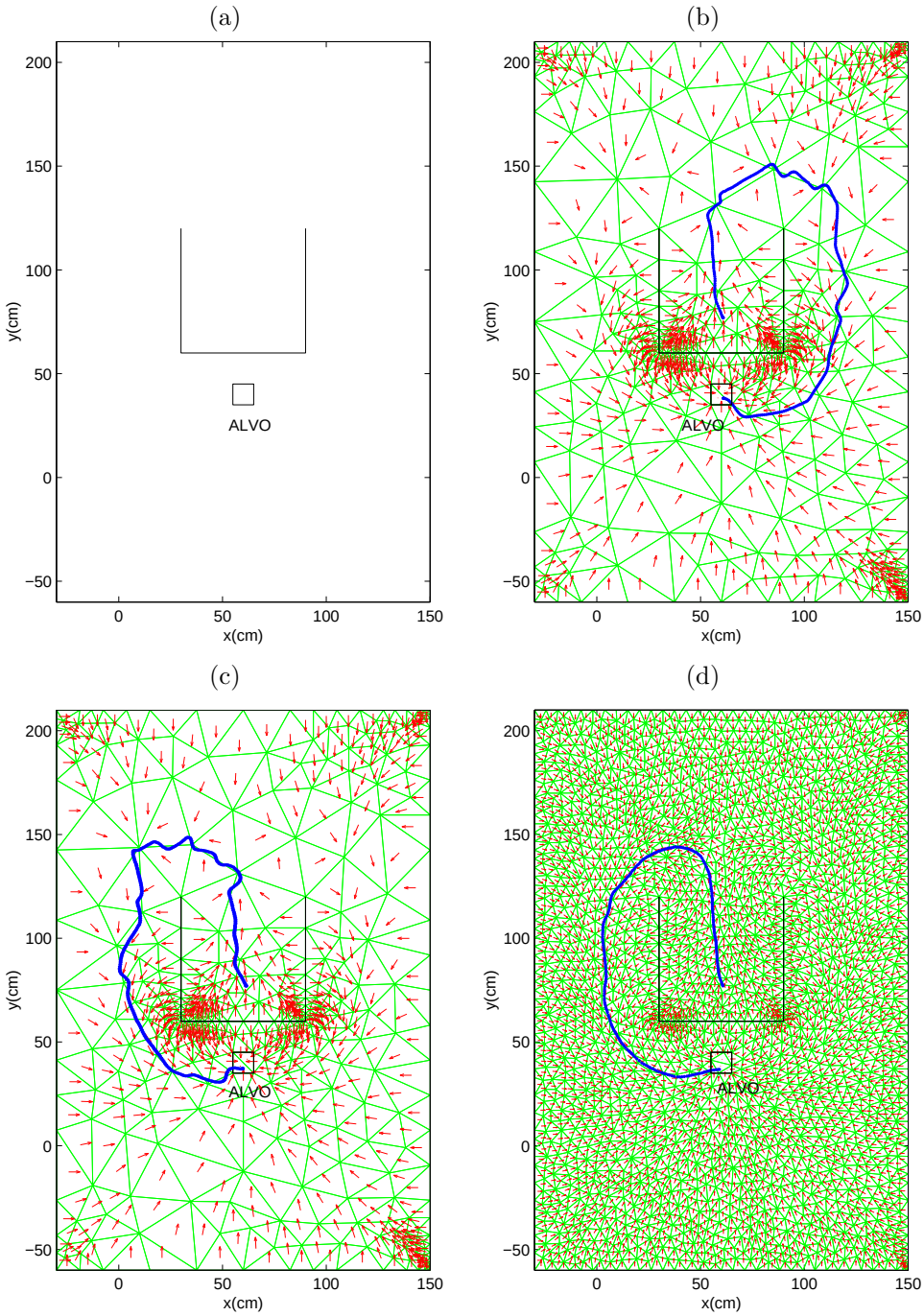

Figura 6.3: Navegação num espaço de trabalho retangular com um obstáculo em forma de U. (a) Ilustração do espaço de trabalho e do alvo. (b) Trajetória real para o robô, quando o tamanho do grid é igual a $0,3$ metros. (c) Outra trajetória real para o robô, quando o tamanho do grid é igual a $0,3$ metros. (d) Trajetória real para o robô, quando o tamanho do grid é igual a $0,05$ metros. As trajetórias do robô são paralelas às setas (gradiente descendente normalizado) dentro de cada elemento.

no sistema de localização. A dinâmica do robô implica no fato do mesmo, ao passar de um elemento para outro, demandar um certo tempo para que sua velocidade assuma a velocidade de referência do elemento atual. Por sua vez, pequenos erros de localização podem levar o robô a seguir o gradiente do elemento errado. Apesar destes problemas, o sistema se mostra robusto à dinâmica do robô e aos erros de localização. É óbvio que esta robustez não é válida para erros que façam o sistema acreditar que o robô está no interior de obstáculos, o que significaria um abandono do domínio válido do espaço de configurações livre.

A Figura 6.3 (c) mostra uma trajetória diferente, porém, também, típica para o caso da mesma malha da Figura 6.3 (b). Apesar da posição inicial do robô ser aparentemente a mesma, as trajetórias seguiram caminhos diferentes. Este resultado é devido aos ruídos na localização. Pela característica de simetria do espaço de trabalho em questão, e pela posição inicial do robô estar aproximadamente sobre a linha de simetria, o ruído pode levar o robô a tomar tanto um caminho à esquerda quanto à direita desta linha.

As trajetórias apresentadas nas Figuras 6.3 (b) e (c) apresentam uma característica ruim: não-suavidade. A não-suavidade é causada pela descontinuidade do gradiente na interface de elementos vizinhos. Trajetórias não-suaves levam a um maior desgaste dos atuadores do robô. Uma maneira de minimizar este problema é utilizar uma malha mais fina. A Figura 6.3 (d) mostra uma trajetória típica obtida quando se utiliza uma malha com tamanho de grid igual a $0,05$ metros e com 2148 nós. Observa-se que esta trajetória é bem mais suave que as anteriores. Entretanto, sabe-se que existe um compromisso entre a suavidade da trajetória e o custo computacional. Ao se utilizar malhas mais finas, a suavidade aumenta, por outro lado, o número de nós também. Conforme visto no Capítulo 5, o maior número de

nós implica no aumento do tamanho do sistema linear a ser resolvido (veja Equação (5.29)). Este trabalho não se preocupou em realizar medições precisas do intervalo de tempo gasto para o cálculo do MEF. Vale ressaltar que para as malhas utilizadas neste exemplo, o cálculo foi feito em menos de 1 segundo.

Uma forma de se suavizar as trajetórias, sem alterar a malha, é a utilização de técnicas de *suavização de campo* [Zienkiewicz and Zhu, 1992]. Estas técnicas fazem um pós-processamento para calcular o valor do gradiente nos nós da malha, e então utilizar uma interpolação para se obter o gradiente no interior de um elemento. O cálculo do gradiente nos nós é feito a partir de uma média ponderada entre os valores de gradiente, pré-calculados pelo método de elementos finitos, dos elementos que compartilham o nó em questão. A implementação destas técnicas garantiria continuidade do gradiente na interface de elementos e é uma proposta de trabalho futuro.

Para finalizar, um outro fato interessante a ser observado é que para malhas mais finas o efeito dos ruídos na variabilidade da trajetória diminui. Isto pode ser comprovado pelos resultados da Tabela 6.1. Esta tabela mostra os resultados de distância média percorrida e desvio padrão após a realização de 20 experimentos para cada malha. Para cada experimento calculou-se a distância total percorrida. Verifica-se que o desvio padrão foi menor para a malha mais fina, e portanto a variabilidade é menor. Além disso, observou-se também que para todos os experimentos realizados com a malha da Figura 6.3 (d), ao contrário do observado com a malha das Figuras 6.3 (b) e (c), obteve-se trajetórias extremamente similares, com o robô sempre deixando o U pelo lado esquerdo.

Tabela 6.1: Comparação de distância média percorrida e desvio padrão entre malhas de grid diferente.

	Grid $= 0,05m$	Grid $= 0,3m$
Média (m)	$2,489$	$2,836$
Desvio Padrão (m)	$4,085 \times 10^{-2}$	$5,367 \times 10^{-2}$

6.2.2 Exemplo 2

Para verificar o funcionamento da metodologia de navegação em ambientes mais complexos, utilizou-se um espaço de trabalho na forma de um labirinto. A Figura 6.4 apresenta uma trajetória típica neste espaço de trabalho, para uma discretização com tamanho de grid de valor igual a 0,05 metros e 856 nós. Não se justifica, para este caso, fazer testes com malhas menos densas, uma vez que geometrias mais complexas, como é o caso neste ambiente, necessitam de malhas mais finas para serem bem representadas. Como esperado, novamente, não se observam pontos de mínimo local, ou pontos de sela, e o robô atinge seu alvo sem qualquer problema. O fato do desaparecimento dos pontos de sela se deve fundamentalmente à discretização. Pode-se verificar, que em nenhum elemento da malha obteve-se um gradiente de valor nulo. De acordo com as expressões derivadas no Capítulo 5, para um dado elemento da malha ter em seu interior um gradiente nulo, necessariamente todos os nós do elemento deveriam ter o mesmo valor de potencial, o que na prática não ocorre.

Antes de finalizar esta seção, é interessante observar a irregularidade das malhas apresentadas. Pode-se notar nas Figuras 6.3 e 6.4 que as malhas possuem elementos de tamanhos muito menores nos locais onde a geometria é mais complicada. O método de elementos finitos confirma, neste contexto, seu funcionamento em situações de malhas não-estruturadas.

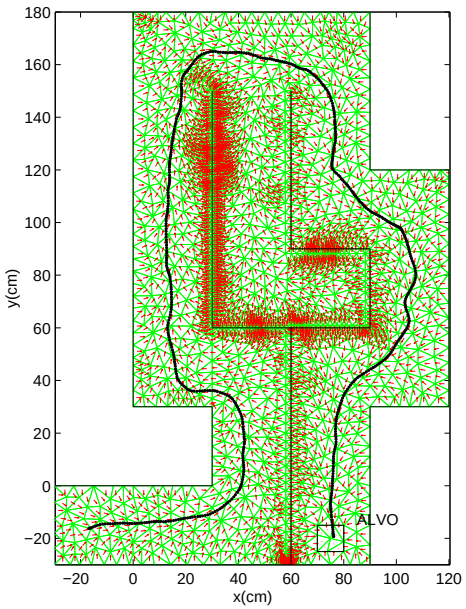

Figura 6.4: Trajetória real num espaço de trabalho em forma de labirinto, com tamanho de grid de $0,05$ metros.

6.3 Resultados para Robôs com 3 Graus de Liberdade

Esta seção se dedica a apresentar resultados para situações onde a orientação do robô deve ser levada em conta. Relembrando, estas situações são duas: (i) o robô necessita chegar ao alvo com uma dada orientação; (ii) o robô não tem formato circular. Os dois exemplos a seguir ilustram exatamente estas duas situações. No primeiro exemplo o robô está num espaço de configurações simples, mas necessita chegar ao alvo com uma dada orientação. Por sua vez, no segundo exemplo, além de haver uma orientação desejada para se atingir o alvo, o robô tem formato retangular. Além das definições de condições de Dirichlet nos contornos dos obstáculos e do alvo, foram definidas condições de contorno periódicas.

6.3.1 Exemplo 1

Um subproduto desta dissertação é um programa para plataforma MS Windows, desenvolvido em linguagem C++, e com a utilização da biblioteca gráfica OpenGL, que permite visualizar e simular trajetórias de robôs em espaços de configurações no $\mathbb{R}^3$. A simulação admite que o controle do robô é perfeito, isto é, não há erros na localização, e a velocidade final do robô segue instantaneamente a velocidade de referência. Esta subseção procura ilustrar o funcionamento da condição de contorno periódica $\phi(x, y, 0) = \phi(x, y, 2\pi)$ (veja Seção 4.2). Assim, construiu-se um espaço de configurações (x, y, θ) simples, constituído de paredes externas e uma região de alvo, a qual indica que o robô só pode chegar com uma dada orientação ($\approx 270^o$ ou $\frac{3\pi}{2}$ rad.), veja Figura 6.5. Após a solução da equação de Laplace neste domínio, utilizou-se o programa desenvolvido para visualizar e simular a trajetória do robô a partir de uma dada configuração inicial. A Figura 6.5 mostra o resultado obtido. Conforme esperado, a condição de contorno periódica permite que o robô atinja o alvo pelo caminho que for mais curto, segundo o eixo θ, que no exemplo é aquele que passa através dos planos $\theta = 0$ e $\theta = 2\pi$, onde se definiu a periodicidade.

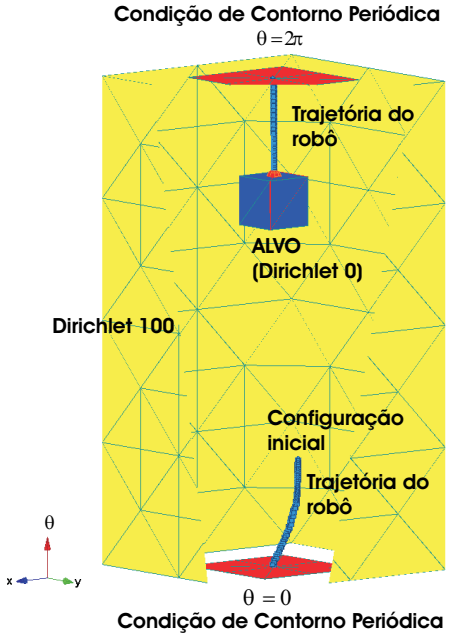

Figura 6.5: Trajetória simulada num espaço de configurações (x, y, θ) com condições de contorno periódicas.

6.3.2 Exemplo 2

Neste exemplo, utilizou-se a plataforma de testes da Seção 6.1, com um robô holonômico retangular de dimensões 18×32 cm. Além disso, definiram-se um alvo com uma orientação fixa em torno de $\frac{\pi}{2}$ e condições de contorno periódicas nos planos $\theta = \pi$ e $\theta = -\pi$.

O espaço de trabalho criado é tal que os obstáculos formam um corredor, com uma largura de ≈ 28 cm, e portanto, devido às dimensões do robô, e para que não haja colisões, a orientação do robô deve ser controlada. Devido à dificuldade de se conseguir malhas de tetraedros de boa qualidade, para espaços de configurações construídos pelas técnicas descritas no Capítulo 3, utilizou-se um espaço de configurações simplificado. Este espaço de configurações foi construído manualmente e é apresentado na Figura 6.6. Na Figura 6.6 (a), observa-se o corte para o plano xy do espaço de configurações

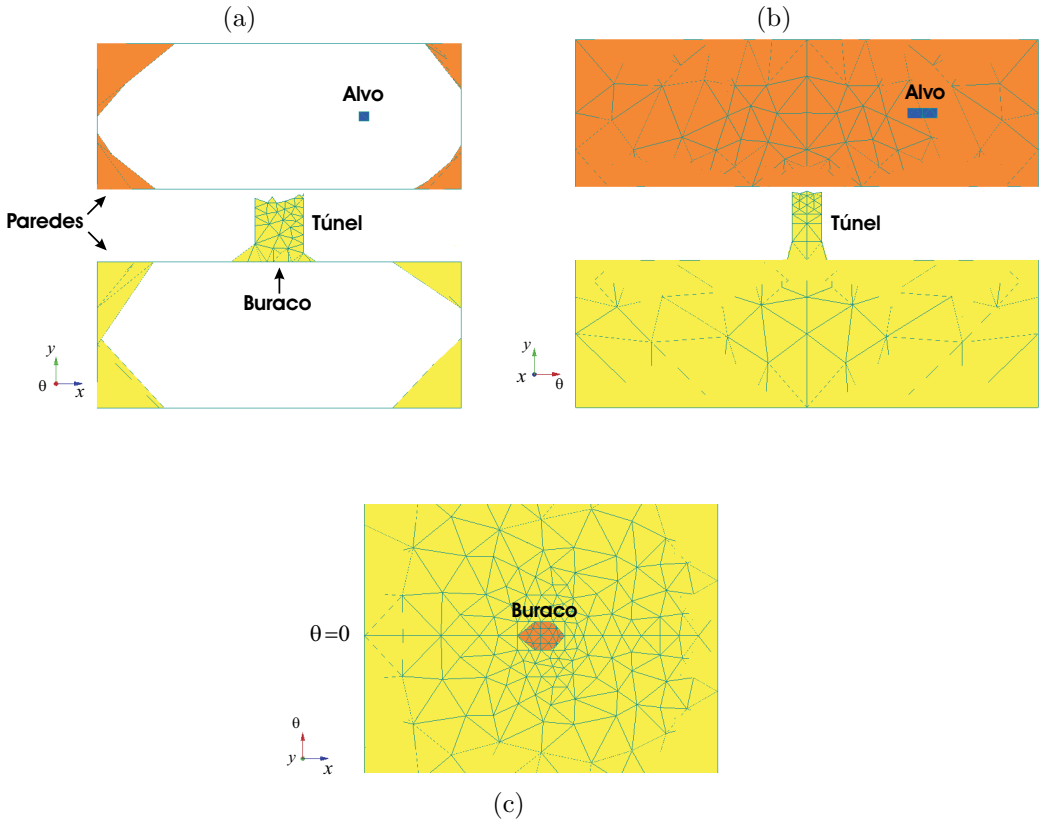

Figura 6.6: Espaço de Configurações simplificado construído manualmente. (a) Corte no plano xy. (b) Corte no plano θy. (c) Corte no plano $x\theta$.

construído. Esta é a mesma vista que se tem para o espaço de trabalho apresentado na Figura 6.7 (a). Observando as duas figuras pode-se deduzir o que foi feito: criou-se um túnel no espaço de configurações, de forma a representar o corredor do espaço de trabalho. Por sua vez, o alvo foi definido como uma região em forma de paralelepípedo, tal que:

$$108 \leq \quad x \quad \leq 112 \text{ cm} \ , \tag{6.1}$$

$$118 \leq \quad y \quad \leq 122 \text{ cm} \ , \tag{6.2}$$

$$\frac{4\pi}{9} \leq \quad \theta \quad \leq \frac{5\pi}{9} \text{ rad } (80^o \leq \theta \leq 100^o). \tag{6.3}$$

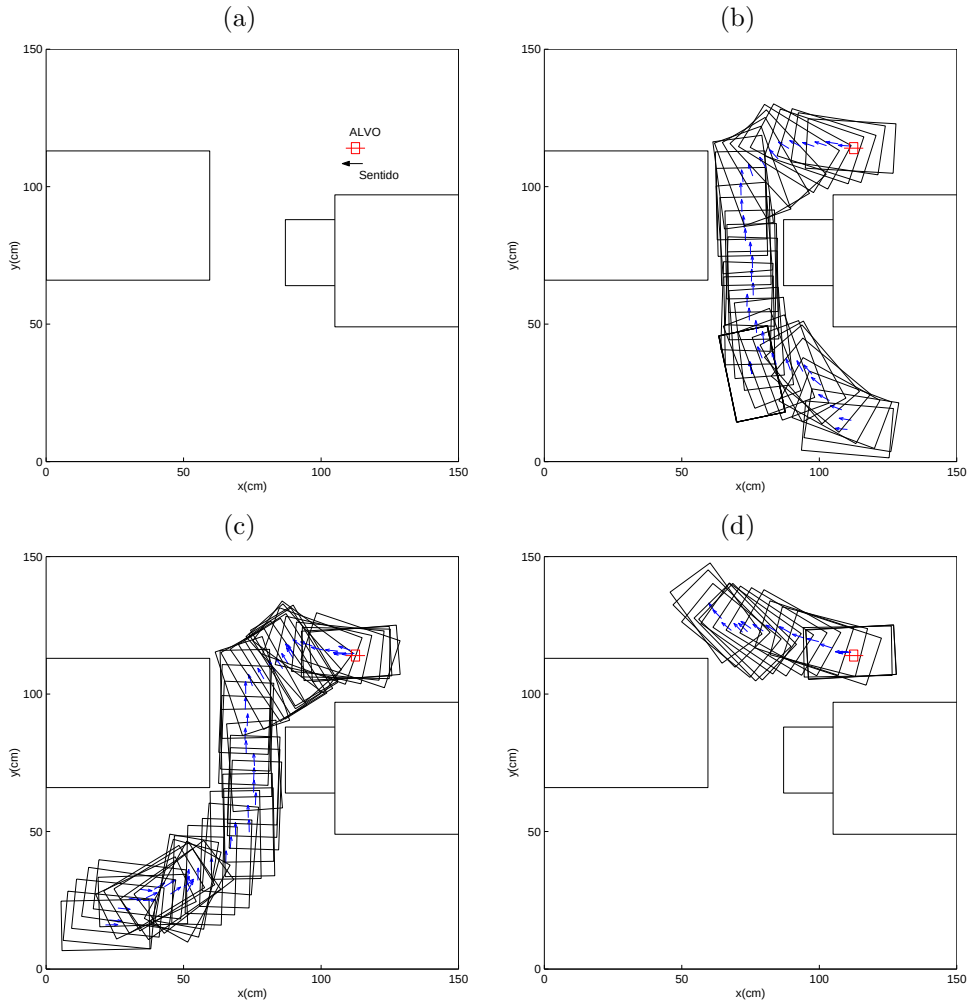

Figura 6.7: Navegação de um robô retangular num espaço de trabalho onde existe um corredor. (a) Ilustração do espaço de trabalho, do alvo, e também da orientação que o robô deve ter no alvo. (b) Trajetória real para o robô com dada configuração inicial. (c) Outra trajetória real para o robô, com outra configuração inicial. (d) Trajetória a partir de uma configuração onde o robô não é obrigado a passar pelo corredor.

A expressão (6.3) indica que o robô é forçado a atingir o alvo com uma orientação de $\frac{\pi}{2} \pm \frac{\pi}{18}$ $(90^o \pm 10^o)$. Na Figura 6.6 (b) pode-se notar a extensão do alvo ao longo da direção de θ. Não se definiu o alvo exatamente na orientação $\frac{\pi}{2}$ para garantir robustez quanto a erros de localização.

Na Figura 6.6 (c) apresenta-se um "zoom" a partir do corte no plano $x\theta$, que permite visualizar o buraco de entrada do túnel definido no espaço de configurações. Este buraco tem dimensões de 20 cm na direção do eixo x e $\frac{\pi}{30}$ (6^o) na direção do eixo θ. Além disso, o centro do buraco está localizado no plano $\theta = 0$. Assim, o robô pode passar pelo buraco com orientações de $0 \pm \frac{\pi}{60}$ $(0^o \pm 3^o)$. O motivo de não se definir apenas a orientação 0 é o mesmo da definição do alvo, ou seja, robustez a erros de localização. Definindo condições de Dirichlet de valor igual a 100 para as paredes do espaço de configurações, e condições de Dirichlet de valor igual a zero para o alvo, o robô é forçado a entrar neste buraco e passar pelo túnel para atingir o alvo, caso sua configuração inicial esteja na parte de baixo do espaço mostrado na Figura 6.6 (a). Nota-se que o robô, geometricamente, poderia passar pelo corredor do espaço de trabalho com orientação igual a π. Por simplicidade, e para garantia de obtenção de uma malha de qualidade para o espaço de configurações livre, definiu-se que o robô só deveria atravessar o corredor com orientação de ≈ 0.

As Figuras 6.7 (b), (c) e (d) mostram os resultados obtidos. Para facilitar a visualização da trajetória seguida pelo robô nas figuras, afixou-se uma seta ao centro da representação do robô, tal que sua orientação coincide com o eixo y, quando o robô tem orientação igual a 0. As figuras correspondem às trajetórias obtidas a partir de diferentes configurações iniciais. Em acordo com a Proposição 4.3, conclui-se pelas figuras, que o robô sempre atinge o alvo, a partir de qualquer configuração inicial, desde que exista um caminho.

Capítulo 7

Conclusões e Trabalhos Futuros

> Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim.
>
> *Chico Xavier (1910–2002)*

Este trabalho endereçou o problema de navegação de robôs em ambientes estáticos e completamente conhecidos, para situações onde tanto as formas do robô, quanto as dos obstáculos presentes no ambiente, são complexas. A solução proposta utiliza o Método de Elementos Finitos (MEF) para o cálculo de funções de navegação. Tradicionalmente, como por exemplo em [Rimon and Koditschek, 1992], a obtenção de funções de navegação é feita analiticamente. O problema deste tipo de tratamento é a extrema dificuldade em aplicá-lo quando as geometrias envolvidas são complexas. Com o objetivo de simplificar a obtenção de funções de navegação, a metodologia proposta adota uma abordagem similar à utilizada nos trabalhos de Connolly *et al.* [Connnolly et al., 1990], [Connnolly, 1992], e [Connolly and Grupen, 1993], onde utilizaram-se funções harmônicas calculadas de forma numérica como funções de navegação. Funções harmônicas são soluções da equação de Laplace, e então, o presente trabalho propôs o método de elementos finitos para obter soluções numéricas desta equação. Acredita-se que este é o primeiro trabalho a utilizar o método de elementos finitos no contexto de navegação

robótica. O MEF permite a utilização de malhas não-estruturadas, e portanto o tratamento de geometrias complexas pode ser feito de forma muito mais eficiente do que em abordagens anteriores, que também calculavam funções harmônicas, mas se baseavam em malhas regulares. O grande gargalo do MEF, nos dias de hoje, é a dependência de malhas não-estruturadas de qualidade. Conforme [Shewchuk, 2002], a má qualidade das malhas pode causar a não convergência do MEF. O desenvolvimento de algoritmos que gerem boas malhas está fora do escopo deste trabalho. Para a realização dos experimentos aqui apresentados, utilizaram-se geradores de malha de distribuição gratuita [Shewchuk, 1996, Si, 2004]. Isto não constituiu um problema para domínios em $\mathbb{R}^2$, mas para o caso em $\mathbb{R}^3$, utilizaram-se domínios simplificados, obtidos manualmente, devido a grande dificuldade em se gerar malhas de boa qualidade para os espaços de configurações obtidos segundo as técnicas mostradas no Cápítulo 3. Isto não invalida o trabalho aqui realizado, uma vez que os objetivos principais – que foram atingidos com sucesso – eram a proposta da metodologia e a comprovação de que a mesma poderia ser utilizada para controlar robôs reais. Além disso, estes problemas de implementação deverão ser resolvidos num futuro próximo, com os constantes avanços na área de Geometria Computacional.

Como foi dito em [Wang and Chirikjian, 2000], a maioria dos autores evita tratar o problema da orientação de robôs móveis. No presente trabalho, propõe-se uma forma para levar em conta a orientação dos robôs numa forma fechada. Esta forma é baseada na construção correta do espaço de configurações do robô, na imposição de condições de contorno periódicas sobre a equação de Laplace, e na utilização de uma única lei de controle para as velocidade de translação e rotação do robô. Apesar da metodologia ser geral e poder ser aplicada para robôs com qualquer número de graus de liber-

dade, nesta dissertação instanciou-se o problema para robôs com 2 e 3 graus de liberdade. Mais especificamente, tratou-se o problema de navegação de robôs móveis holonômicos no plano. Dessa forma, os espaços de configurações utilizados foram definidos em $\mathbb{R}^2$ (x, y) e em $\mathbb{R}^3$ (x, y, θ).

Para a construção de espaços de configurações (x, y, θ) este trabalho propôs um algoritmo simples, que é apresentado no Capítulo 3 (Algoritmo 3.2). Este algoritmo baseia-se na reconstrução tridimensional do C-obstáculo através de triangulação. Após a decomposição dos polígonos que representam os obstáculos e o robô em sub-polígonos convexos, e utilizando-se algoritmos clássicos de soma de Minkowski [de Berg et al., 2000], constrói-se C-obstáculos bidimensionais para planos $\theta = c$, onde c é uma constante no intervalo $[0 : 2\pi)$. O algoritmo proposto, permite, então, que estas camadas planas sejam conectadas de forma a obter uma aproximação do C-obstáculo tridimensional. Embora este algoritmo não seja genérico, verificou-se, durante os testes realizados, que se a distância $\delta\theta$ entre as camadas estiver no intervalo em radianos $\frac{\pi}{180} < \delta\theta < \frac{\pi}{36}$ $(1^o < \delta\theta < 5^o)$, uma boa aproximação dos poliedros reais pode ser obtida. É importante deixar claro também, que os testes foram realizados em espaços de configurações "simples", onde não havia mudança de topologia de uma camada para a outra. Essa mudança de topologia pode ocorrer quando dois obstáculos originalmente distintos são transformados em um único polígono no C-obstáculo bidimensional, para apenas alguns planos $\theta = c$. A implementação que foi realizada pode apresentar problemas quanto a este fato, pois, primeiramente, constrói-se o C-obstáculo em $\mathbb{R}^2$ completo de cada camada, através de soma de Minkowski e união de polígonos, tal qual proposto em [de Berg et al., 2000], e depois é feita a reconstrução do C-obstáculo em $\mathbb{R}^3$ por meio do Algoritmo 3.2. Como o Algoritmo 3.2 não prevê a conexão entre camadas topologicamente

distintas, não se pode garantir seu funcionamento. Em [Fuchs et al., 1977, Boissonat, 1988, Ekoule et al., 1991, Meyers et al., 1992], são propostos algoritmos genéricos, que resolvem este problema de reconstrução tridimensional a partir de planos. No final do Capítulo 3, propõe-se a união de poliedros, a qual não foi implementada por fugir do escopo do trabalho. Acredita-se que se realizada esta implementação, o problema será resolvido, uma vez que o Algoritmo 3.2 será responsável por reconstruir apenas os poliedros provenientes dos sub-polígonos convexos, evitando portanto camadas com topologias distintas. Provavelmente, esta implementação, também, permitirá a geração de uma malha de elementos finitos correta, pois a geometria obtida será bem mais simples e compatível com os geradores de malha utilizados.

No Capítulo 4 mostrou-se que, se certas restrições forem obedecidas para a definição das condições de contorno, tem-se a garantia de que um robô holonômico sempre atingirá o alvo num tempo finito, a partir de qualquer configuração inicial do espaço de configurações livre, $\mathcal{F}$, desde que exista um caminho. Nos experimentos do Capítulo 6 pôde-se observar claramente esta propriedade da metodologia, que na verdade é uma propriedade de qualquer função de navegação [Rimon and Koditschek, 1992]. Esta propriedade garante robustez a pequenos erros de localização e a ruídos nos atuadores do robô. Além disso, esta propriedade garante imunidade em relação a replanejamento, que é um problema clássico de metodologias de planejamento de caminhos baseadas em grafos. Note que a robustez a ruídos e a imunidade a replanejamento são obtidas para os casos onde os erros na estimação da posição e orientação do robô não implicarem em um abandono do espaço de configurações livre, $\mathcal{F}$. Este tipo de erro seria um problema para a maioria das metodologias de navegação, pois corresponderia ao sistema de localização informar, num dado instante, que o robô está dentro de um obstáculo, ou

mesmo fora de seu espaço de trabalho, o que não faria sentido.

Em todos os experimentos do Capítulo 6, observaram-se trajetórias não suaves. Isto se deve à descontinuidade do gradiente da função potencial na interface entre elementos. Esta não suavidade implicaria em desgaste dos atuadores do robô a longo prazo. Além disso, esta característica pode ser ainda mais crítica se os robôs forem não-holonômicos. Em [Pereira, 2003] apresenta-se o controle de robôs não-holonômicos através de um controlador não linear, que tem o objetivo de rastrear trajetórias obtidas via funções de navegação. Isto poderia ser feito utilizando como função de navegação, a função harmônica calculada pela metodologia proposta no presente trabalho. O controlador não linear receberia como entrada, a velocidade de referência calculada pela expressão (4.12). Durante a navegação, nos momentos onde o robô passasse de um elemento para outro da malha, a descontinuidade do gradiente implicaria na aplicação de degraus na entrada deste controlador, o que produziria trajetórias de péssima qualidade. A utilização de técnicas de *suavização de campo* [Zienkiewicz and Zhu, 1992], como um pós-processamento, seria uma possível solução para estas questões, pois garantiria a continuidade do gradiente na interface de elementos.

Resumindo as vantagens da metodologia apresentada, tem-se:

- Trajetórias livres de mínimos locais e ciclos limites são obtidas independentemente das configurações iniciais e finais;
- A metodologia é *completa*;
- Funções de potencial para espaços de configurações complexos podem ser obtidas;
- A orientação do robô é levada em consideração numa forma fechada;

- Robustez relacionada a pequenos erros de localização e dinâmica do robô é intrínseca à própria metodologia.

Por sua vez, as principais desvantagens são:

- Deve-se ter um bom conhecimento do ambiente, isto é, todos os contornos dos obstáculos devem ser modelados previamente;
- Apenas obstáculos estáticos são tratados;
- Restrições não-holonômicas não são abordadas;
- A lei de controle resultante não é suave;

Com o objetivo de sanar as desvantagens citadas da metodologia, os problemas da implementação realizada e, também, estender os resultados já obtidos, propõem-se os seguintes trabalhos futuros:

- Consideração de obstáculos não modelados e dinâmicos, através da implementação de um método de elementos finitos em tempo real ou de um método sem malha;
- Investigação de maneiras de se acrescentar restrições não-holonômicas, baseando-se em propriedades de meios anisotrópicos;
- Suavização das trajetórias obtidas, através do uso de técnicas de Suavização de Campo;
- Construção correta dos espaços de configurações em $\mathbb{R}^3$, com a implementação das operações de união de poliedros, e melhoria do Algoritmo 3.2, no sentido de torná-lo geral;

- Extensão dos resultados para dimensões maiores, como, por exemplo, dimensão 6, onde se inserem robôs manipuladores, aéreos, subaquáticos, e de exploração, por meio da implementação de métodos de elementos finitos para dimensões superiores.

O presente trabalho deu origem a uma publicação em congresso internacional e outra em congresso nacional:

- Pimenta, L. C. A., Fonseca, A. R., Pereira, G. A. S., Mesquita, R. C., Silva, E. J., Caminhas, W. M., Campos, M. F. M. (2005). On Computing Complex Navigation Funtions. In *Proc. of the IEEE International Conference on Robotics and Automation*, (Barcelona, Spain).

- Pimenta, L. C. A., Pereira, G. A. S., Mesquita, R. C., Caminhas, W. M., and Campos, M. F. M. (2004). Elementos finitos na navegação de robôs móveis. In *Proc. of XV Congresso Brasileiro de Automática*, (Gramado, RS).

Além disso, um outro artigo foi submetido a um outro congresso internacional:

- Pimenta, L. C. A., Fonseca, A. R., Pereira, G. A. S., Mesquita, R. C., Silva, E. J., Caminhas, W. M., Campos, M. F. M. (2005). Robots navigation based on electromagnetic fields. In *Proc. of The 15th Conference on the Computation of Electromagnetic Fields*, (Shenyang, China) (submitted).

Referências Bibliográficas

[Adams, 1975] Adams, R. A. (1975). *Sobolev Spaces*. Academic Press, New York.

[Arkin, 1998] Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge, MA.

[Barraquand and Latombe, 1990] Barraquand, J. and Latombe, J. C. (1990). A Monte-Carlo algorithm for path-planning with many degrees of freedom. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 1712–1717.

[Becker et al., 1981] Becker, E. B., Carey, G. F., and Oden, J. T. (1981). *Finite Elements - An Introduction*, volume 1 of *The Texas Finite Element Series*. Prentice Hall Inc.

[Benson, 1966] Benson, R. V. (1966). *Euclidean Geometry and Convexity*. McGraw-Hill.

[Boissonat, 1988] Boissonat, J.-D. (1988). Shape reconstruction from planar cross-sections. *Comp. Vision Graph. Image Process.*, 44:1–29.

[Borenstein and Y.Koren, 1989] Borenstein, J. and Y.Koren (1989). Real-time obstacle avoidance for fast mobile robots. *TSMC*, 19(5):1179–1187.

[BRFEM, 2005] BRFEM (2005). Brazil finite elements. `http://www.ead.eee.ufmg.br/~renato/brfem`. Acessado em fevereiro, 2005.

[Brock and Khatib, 1999] Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 341–346.

[Brooks, 1983] Brooks, R. (1983). Solving the find-path problem by good representation of free space. *IEEE Trans. on Systems, Man, and Cybernetics*, 13(2):190–197.

[Burden et al., 1978] Burden, R. L., Faires, J. D., and Reynolds, A. C. (1978). *Numerical Analysis*. Prindle, Weber and Schmidt, Boston.

[Carey and Oden, 1981a] Carey, G. F. and Oden, J. T. (1981a). *Finite Elements - A Second Course*, volume 2 of *The Texas Finite Element Series*. Prentice Hall Inc.

[Carey and Oden, 1981b] Carey, G. F. and Oden, J. T. (1981b). *Finite Elements - Computational aspects*, volume 3 of *The Texas Finite Element Series*. Prentice Hall Inc.

[Carey and Oden, 1981c] Carey, G. F. and Oden, J. T. (1981c). *Finite Elements - Mathematical Aspects*, volume 4 of *The Texas Finite Element Series*. Prentice Hall Inc.

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

[Chew and Kedem, 1993] Chew, L. P. and Kedem, K. (1993). A convex polygon among polygonal obstacles: Placement and high-clearance motion. *Comput. Geom.*, 3:59–89.

[Connnolly, 1992] Connnolly, C. I. (1992). Applications of harmonic functions to robotics. In *Proc. of the IEEE Int'l. Symposium on Intelligent Control*, pages 498–502.

[Connnolly et al., 1990] Connnolly, C. I., Burns, J. B., and Weiss, R. (1990). Path planning using Laplace's equation. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 2102–2106.

[Connolly and Grupen, 1993] Connolly, C. I. and Grupen, R. A. (1993). On the applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):931–946.

[Courant, 1943] Courant, R. (1943). Variational methods for the solution of problems of equilibrium and vibration. *Bulletin of the American Mathematical Society*, 49.

[de Berg et al., 2000] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (2000). *Computational Geometry Algorithms and Applications*. Springer-Verlag, second edition.

[Donald, 1984] Donald, B. (1984). Motion planning with six degrees of freedom. Technical Report AIM-791, MIT Artificial Intelligence Laboratory.

[Dudek and Jenkin, 2000] Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge University Press, 1 edition.

[Ekoule et al., 1991] Ekoule, A. B., Peyrin, F. C., and Odet, C. L. (1991). A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, 10(2):182–199.

[Esposito and Kumar, 2002] Esposito, J. M. and Kumar, V. (2002). A method for modifying closed-loop motion plans to satisfy unpredictable dynamic constraints at runtime. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 1691–1696.

[Flato, 2000] Flato, E. (2000). Robust and efficient construction of planar Minkowski sums. Master's thesis, Tel-Aviv University.

[Forsyth and Ponce, 2003] Forsyth, D. A. and Ponce, J. (2003). *Computer Vision: A Modern Approach*. Prentice Hall, New Jersey.

[Fox et al., 1997] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33.

[Fuchs et al., 1977] Fuchs, H., Kedem, Z. M., and Uselton, S. P. (1977). Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702.

[Guibas et al., 1983] Guibas, L. J., Ramshaw, L., and Stolfi, J. (1983). A kinetic framework for computational geometry. In *24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 100–111.

[Guldner et al., 1997] Guldner, J., Utkin, V. I., and Hashimoto, H. (1997). Robot obstacle avoiding in n-dimensional space using planar harmonic artificial potential fields. *Journal of Dynamics, Measurement and Control*, 119:160–166.

[Hughes, 2000] Hughes, T. J. R. (2000). *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*. Dover Publications,Inc.

[Ida and Bastos, 1992] Ida, N. and Bastos, J. P. A. (1992). *Electromagnetics and Calculation of Fields*. Springer-Verlab.

[Júnior, 2003] Júnior, E. P. E. S. (2003). *Navegação Exploratória baseada em Problemas de Valores de Contorno*. PhD thesis, Universidade Federal do Rio Grande do Sul.

[Kavraki et al., 1996] Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Trans. on Robotics and Automation*, 12(4):566–580.

[Khatib, 1980] Khatib, O. (1980). *Commande dynamique dans l'espace opérational des robots manipulateurs en présence d'obstacles*. PhD thesis, École nationale Supérieure de l'Aeéronatique et de l'Espace (ENSAE), France.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int'l. Journal of Robotics Research*, 5(1):90–98.

[Kim and Khosla, 1992] Kim, J.-O. and Khosla, P. K. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Trans. on Robotics and Automation*, 8(3):338–349.

[Konolige, 2000] Konolige, K. (2000). A gradient method for realtime robot control. In *Proc. of the IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems*, pages 639–646.

[Krogh, 1984] Krogh, B. H. (1984). A generalized potential field approach to obstacle avoidance control. In *Proc. SME Conf. Robotics Res.*

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

[Lozano-Pérez and Wesley, 1979] Lozano-Pérez, T. and Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570.

[Macedo, 1988] Macedo, A. (1988). *Eletromagnetismo*. Editora Guanabara.

[M.D.Adams et al., 1990] M.D.Adams, H.Hu, and P.J.Roberts (1990). Towards a real-time architecture for obstacle avoidance and path planning in mobile robots. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 584–589.

[Meeker, 2004] Meeker, D. (2004). *Finite Element Method Magnetics - User's Manual*. version 4.0.

[Meyers et al., 1992] Meyers, D., Skinner, S., and Sloan, K. (1992). Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258.

[Milnor, 1970] Milnor, J. (1970). *Annals of Mathematics Studies*, volume 51, chapter Morse Theory. Princeton University Press.

[Murray et al., 1994] Murray, R. M., Li, Z., and Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press LLC.

[Pereira, 2003] Pereira, G. A. S. (2003). *Navegação e Controle de Robôs Móveis Cooperativos: Uma Abordagem baseada em Conectividade de Grafos*. PhD thesis, Universidade Federal de Minas Gerais.

[Pereira et al., 2000] Pereira, G. A. S., Campos, M. F. M., and Aguirre, L. A. (2000). Data based dinamical model of vision observed small robots. In *Proc. of the IEEE Int'l. Conf. on Systems, Man and Cybernetics*, pages 3312–3317.

[Pereira et al., 2004a] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2004a). Decentralized algorithms for multi-robot manipulation via caging. *International Journal of Robotics Research*, 23(7):783–795.

[Pereira et al., 2004b] Pereira, G. A. S., Torres, F. B., and Campos, M. F. M. (2004b). Desenvolvimento de robôs holonômicos de baixo custo para o estudo de robótica móvel. In *Anais do XV Congresso Brasileiro de Automática (CBA'04)*, (Gramado, RS).

[R.A.Brooks and Pérez, 1985] R.A.Brooks and Pérez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *TSMC*, 15(2):225–233.

[R.B.Tilove, 1990] R.B.Tilove (1990). Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 566–571.

[Rimon and Koditschek, 1988] Rimon, E. and Koditschek, D. E. (1988). Exact robot navigation using cost functions: The case of distinct spherical boundaries in E^n. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 1791–1796.

[Rimon and Koditschek, 1992] Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potencial functions. *IEEE Trans. on Robotics and Automation*, 8(5):501–517.

[Sastry, 1999] Sastry, S. (1999). *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag.

[Shewchuk, 1994] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University.

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

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

[Shewchuk, 1998] Shewchuk, J. R. (1998). Tetrahedral mesh generation by delaunay refinement. In *Proc. Symp. on Comput. Geometry*, pages 86–95.

[Shewchuk, 2002] Shewchuk, J. R. (2002). What is a good linear element? interpolation, conditioning, and quality measures. *Eleventh International Meshing Roundtable*, pages 115–126.

[Si, 2004] Si, H. (2004). Tetgen - a quality tetrahedral mesh generator and three dimensional delaunay triangulator. Technical Report 9, Weierstrass Institute for Applied Analysis and Stochastics, Berlin.

[Silvester, 1990] Silvester, P. P. (1990). *Finite elements for electrical engineers*. Cambridge University Press.

[Tadokoro, 2004] Tadokoro, S. (2004). RoboCupRescue – Background, History and Objectives. `http://www.dis.uniroma1.it/~multirob/camp04/info.html`. Acessado em fevereiro, 2005.

[Tanner et al., 2003] Tanner, H. G., Loizou, S. G., and Kyriakopoulos, K. J. (2003). Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Trans. on Robotics and Automation*, 19(1):53–64.

[Valavanis et al., 2000] Valavanis, K. P., Hebert, T., Kolluru, R., and Tsourveloudis, N. (2000). Mobile robot navigation in 2-D dynamic environments using an electrostatic potential field. *IEEE Trans. on Systems, Man, and Cybernetics—Part A*, 30(2):187–196.

[von Feldt, 2004] von Feldt, R. (2004). Waveofdestruction.org. `http://www.waveofdestruction.org`. Acessado em fevereiro, 2005.

[Wang and Chirikjian, 2000] Wang, Y. and Chirikjian, G. S. (2000). A new potential field method for robot path planning. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 977–982.

[Waydo and Murray, 2003] Waydo, S. and Murray, R. M. (2003). Vehicle motion planning using stream functions. In *Proc. of the IEEE Int'l. Conf. on Robotics Automation*, pages 2484–2491.

[Weinstock, 1974] Weinstock, R. (1974). *Calculus of Variations with Applications to Physics and Engineering*. Dover Publications, Inc.

[Xu, 1999] Xu, W. L. (1999). A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behaviour-based mobile robot. *Robotics and Autonomous Systems*, (6):1–10.

[Zachmanoglou and Thoe, 1986] Zachmanoglou, E. C. and Thoe, D. W. (1986). *Introduction to Partial Differential Equations with Applications*. Dover Publications, Inc.

[Zienkiewicz and Zhu, 1992] Zienkiewicz, O. C. and Zhu, J. Z. (1992). The superconvergent patch recovery and a posteriori estimates, part 1: the recovery technique. *International Journal for Numerical Methods in Engineering*, 33:1331–1364.

Apêndice A

Plataforma Robótica do VERLAB

> Faça as coisas mais simples que você puder, porém não se restrinja às mais simples.
>
> *Albert Einstein (1879–1955)*

Este apêndice descreve com maiores detalhes a plataforma robótica utilizada para validar a metodologia proposta (veja os resultados no Capítulo 6). Na Seção A.1 é detalhado o sistema de localização global baseado em visão computacional. Por sua vez, na Seção A.2 o controle e a mecânica do robô holonômico são abordados.

A.1 Sistema de Visão Computacional

Esta seção descreve o sistema de visão utilizado como único sensor na localização do robô (veja Figura 6.1). O sistema utiliza uma câmera externa presa ao teto do laboratório com um ângulo de inclinação de aproximadamente 45^o, de forma a maximizar o campo de visão. Esta câmera visualiza o espaço de trabalho do robô e permite a estimativa de posição e orientação do mesmo. Para tanto, o robô possui dois marcadores em forma de círculos coloridos, dispostos lado a lado (veja Figura 6.2). Por meio da determinação

da posição dos dois círculos no ambiente obtém-se a posição e a orientação do robô.

Utilizando-se processos básicos de processamento de imagens, tais como segmentação, binarização e rotulação [Forsyth and Ponce, 2003], as posições, (u, v), dos marcadores do robô são facilmente e eficientemente encontradas no plano de imagem. A transformação entre estas coordenadas no plano da imagem e as posições reais dos marcadores no ambiente é executada utilizando-se uma transformação homográfica ou projetiva. Assim,

$$\begin{bmatrix} s\,x \\ s\,y \\ s \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} , \tag{A.1}$$

onde a matriz de homografia, $\mathbf{H}$, é obtida via calibração. Esta matriz inclui parâmetros intrínsecos da câmera (distância focal, distorção da lente, etc.), os parâmetros extrínsecos (posição e orientação) e outras constantes como a altura do robô. A matriz $\mathbf{H}$ é construída comparando pelo menos quatro pontos conhecidos do ambiente com suas projeções na imagem. A única restrição é que todos os pontos do ambiente estejam no mesmo plano, forçando a transformação a ser 2D para 2D. Para respeitar esta restrição, supõe-se um piso liso no laboratório.

A qualidade da estimação de posição e orientação do robô calculada por meio da matriz de homografia pode ser estimada pela matriz de covariância desta transformação. O cálculo analítico desta matriz involve o Jacobiano da matriz de parâmetros intrínsecos e extrínsecos da câmera. Deve-se ainda estimar a covariância do processo de visão responsável por detectar os marcadores, o que não é simples. Assim, alternativamente, foram realizados testes experimentais de forma a determinar que as covariâncias para x e y não são

maiores que 2 cm.

A.2 Robô Holonômico

O robô holonômico tem três rodas omnidirecionais montadas numa base de acrílico. A Figura A.1 mostra o sistema montado numa base circular. As rodas omnidirecionais possuem rolamentos montados de tal forma que estas possam rolar livremente em direções perpendiculares ao seu eixo e possam ter a sua velocidade controlada na direção em que estejam apontando. Por isso, o robô holonômico pode se mover, instantaneamente, em qualquer direção. Um servo de aeromodelismo, modificado para rotação, traciona cada roda. Os três servos recebem sinais de controle de um computador remoto a partir de um par transmissor/receptor. Os sinais provenientes do computador são convertidos para sinais PWM (*"pulse-width-modulation"*) através da interface de um microcontrolador PIC®, o qual comunica com o computador por meio da porta serial RS232.

Para entender como os sinais de controle são calculados, considere a Figura A.1. Deseja-se que o robô siga o vetor u_i calculado em relação a um sistema de referência fixo $\{U\}$. O primeiro passo é então transformar u_i para o sistema de referência do robô $\{R\}$ por meio de uma transformação (rotação e translação) simples. A matriz de rotação é determinada pela orientação do robô, que por sua vez é estimada pelo sistema de visão descrito na Seção A.1. Uma vez que u_i é transformado para o sistema de referência do robô, a velocidade linear de cada roda é calculada projetando-se u_i sobre os vetores unitários perpendiculares aos eixos de cada roda (F_1, F_2 e F_3 na Figura A.1). Assim, a velocidade angular de cada roda é calculada como:

$$\omega_j = (u_i \cdot F_j)/r\,, \qquad 1 \leq j \leq 3\,, \tag{A.2}$$

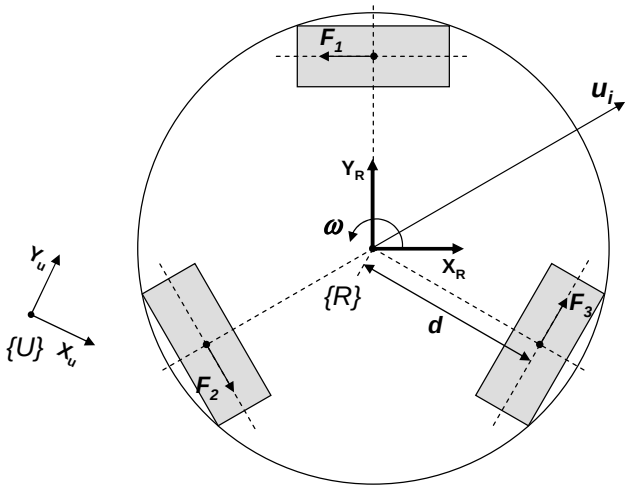

Figura A.1: Esquemático do robô holonômico. Os retângulos sombreados representam as rodas omnidirecionais.

onde $F_1 = [-1, 0]^T$, $F_2 = [1/2, -\sqrt{3}/2]^T$, $F_3 = [1/2, \sqrt{3}/2]^T$, r é o raio da roda e "$\cdot$" representa o produto interno de dois vetores. Pode-se ainda controlar a velocidade angular, ω, do robô reescrevendo a Equação (A.2) como:

$$\omega_j = (u_i \cdot F_j + d\,\omega)/r\,, \qquad 1 \leq j \leq 3\,, \tag{A.3}$$

onde d é a distância entre o centro de cada roda e o centro do robô, conforme Figura A.1.

Por simplificação, supõe-se que os sinais de controle são proporcionais às velocidades angulares das rodas. Em [Pereira et al., 2000] mostra-se que este não é sempre o caso devido a não-linearidades, dinâmicas e atrasos presentes no sistema. Entrentanto, porque os robôs holonômicos têm em geral velocidades lineares pequenas se comparadas com a velocidade de resposta do sensor (câmera), a malha fechada de controle é capaz de corrigir pequenos erros devidos a aproximações.