UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Navegação exploratória baseada em Problemas de Valores de Contorno

por

EDSON PRESTES E SILVA JÚNIOR

Tese submetida a avaliação,
como requisito parcial para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Paulo M. Engel
Orientador

Prof. Dr. Marco A. P. Idiart
Co-orientador

Porto Alegre, fevereiro de 2003.

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Silva Júnior, Edson Prestes e

Navegação exploratória baseada em Problemas de Valores de Contorno / por Edson Prestes e Silva Júnior. — Porto Alegre: PPGC da UFRGS, 2003.

109 f.: il.

Tese (doutorado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientador: Engel, Paulo M.; Co-orientador: Idiart, Marco A. P..

1. Mapeamento de Ambientes. 2. Navegação Exploratória. 3. Problemas de Valores de Contorno. 4. Funções Harmônicas. 5. Robótica Móvel. I. Engel, Paulo M.. II. Idiart, Marco A. P.. III. Título.

"O desejo de nos preenchermos, de nos tornarmos alguma coisa, surge quando percebemos que nada somos. Porque sou nada, porque sou insuficiente, vazio, interiormente pobre, luto por "vir a ser"alguma coisa; exterior ou interiormente, luto por me preencher numa pessoa, numa coisa, numa idéia. Preencher esse vazio- nisso consiste todo o processo da nossa existência. Tendo consciência de que estamos vazios, de que somos pobres interiormente, lutamos com o fim de acumular coisas, exteriormente, ou de cultivar riquezas interiores"

—Krishnamurti

Agradecimentos

Gostaria neste espaço de expressar minha gratidão a todas as pessoas que participaram de mais este ciclo que se fecha.

Inicialmente agradeço todo o apoio que minha família tem me dado durante estes anos. Em especial, agradeço a minha mãe por ter me ensinado a ser perseverante, confiante e determinado; e a minha Verinha por ter me dado amor, carinho, por ter tido paciência com minha alteração constante de humor.

Agradeço aos professores Paulo Engel, Marco Idiart, Dante Barone, Carla Dal Sasso pela sua amizade e seu eterno apoio. Em especial gostaria de agradecer ao professor Marco Idiart por ter me incentivado e ensinado a realizar pesquisas avançadas, além de ter me dado forças quando em momentos críticos quis desistir.

Além disso, agradeço aos amigos e colegas que participaram direta ou indiretamente deste trabalho, em especial ao Marcelo Trevisan; e aos funcionários Ida, Beatriz, Henrique, Luis Otávio e Eliane pelo excelente serviço e presteza quando solicitados.

Ademais, agradeço ao CNPQ por financiar este trabalho e ao Laboratório de Robótica Inteligente(LRI) por fornecer a infraestrutura adequada para a realização desta tese.

Sou grato a todos vocês.

Sumário

Lista de Abreviaturas . . . 7

Lista de Figuras . . . 8

Lista de Tabelas . . . 12

Resumo . . . 13

Abstract . . . 14

1 Introdução . . . 15
1.1 Motivação . . . 16
1.2 Objetivos . . . 17
1.3 Organização . . . 17

2 Exploração de Ambientes . . . 18
2.1 Métodos de Navegação . . . 18
2.2 Planejamento Exploratório . . . 19
2.3 Componentes Básicos . . . 21
2.3.1 Representação do Ambiente . . . 22
2.3.2 Auto-Localização . . . 24
2.3.3 Mapeamento e Localização Concorrentes . . . 25
2.4 Trabalhos Relacionados . . . 27

3 Princípios da Navegação Exploratória baseada em Problemas de Valores de Contorno . . . 31
3.1 Campos potenciais harmônicos para o planejamento de caminhos 32
3.2 Além das Funções Harmônicas . . . 33
3.3 Contornos Dinâmicos e Exploração . . . 35
3.4 Exploração dirigida por características do contorno . . . 35
3.5 Mudança de Domínio . . . 37

4 Agente Exploratório Proposto . . . 38
4.1 Arquitetura do Agente . . . 38
4.2 Ingredientes Básicos . . . 41
4.2.1 Processamento Sensorial . . . 41
4.2.2 Localização . . . 41
4.2.3 Representação do Ambiente . . . 42
4.2.4 Atualização do Mapa . . . 44
4.2.5 Preenchimento de Lacunas . . . 45
4.2.6 Planejamento . . . 46
4.2.7 Condição de Parada . . . 48
4.2.8 Processamento da Ação . . . 52
4.3 Comportamento Exploratório . . . 54
4.3.1 Janela de Ativação . . . 56
4.3.2 Conhecimento local × conhecimento global . . . 56

5 Experimentos . . . 60
5.1 Robô Nomad . . . 60
5.2 Interface com o usuário . . . 61
5.3 Descrição Algorítmica do Processo Exploratório . . . 63
5.4 Experimentos de exploração em ambientes internos *densos* . . . 64
5.4.1 Explorando ambientes internos *densos* . . . 65
5.4.2 Comparando com outras estratégias . . . 67
5.4.3 Analisando os métodos de relaxação e a taxa de atualização . . . 70
5.4.4 Analisando o tempo de exploração . . . 71
5.4.5 Analisando o erro de odometria . . . 73
5.4.6 Explorando ambientes internos com obstáculos dinâmicos . . . 74
5.4.7 Exploração orientada a objetivos . . . 75
5.4.8 Usando a janela de ativação adaptativa . . . 79
5.4.9 Analisando o conhecimento global × conhecimento adaptativo . . . 81
5.5 Experimentos de exploração em ambientes internos *esparsos* . . . 84
5.5.1 Explorando ambientes críticos : ambientes internos *esparsos* . . . 84
5.5.2 Experimentos no robô simulado . . . 85
5.5.3 Avaliação do desempenho . . . 87
5.5.4 Experimentos no robô Nomad . . . 89

6 Conclusões . . . 91
6.1 Arquitetura do agente . . . 92
6.2 Módulo de Planejamento . . . 93
6.3 Discussão e sumário dos resultados . . . 93
6.3.1 A equação base e suas variações . . . 93
6.3.2 Formas de atualização do potencial . . . 95
6.3.3 Comparação com outras técnicas . . . 96
6.3.4 Tempo de exploração . . . 96
6.3.5 Erros de odometria . . . 96
6.3.6 Ambientes dinâmicos . . . 97
6.3.7 Janela Adaptativa . . . 97
6.4 Trabalhos Futuros . . . 97
6.5 Contribuições da Tese . . . 99

Bibliografia . . . 101

Lista de Abreviaturas

EDF	Equação de diferenças finitas
EDP	Equação diferencial parcial
EM	Expectation Maximization
HIMM	Histogramic In-Motion Mapping
IA	Inteligência Artificial
KS	Knowledge Source
LRI	Laboratório de Robótica Inteligente
MLC	Mapeamento e Localização Concorrentes
MLS	Mapeamento e Localização Simultâneos
PDF	Probability Density Function
PVC	Problemas de Valores de Contorno
SMPA	Sense-model-plan-act
SOR	Successive Over-Relaxation
SPAE	Sense-plan-act-explore
VFF	Virtual Force Field

Lista de Figuras

FIGURA 3.1 – Região do ambiente onde o PVC é calculado. 32
FIGURA 3.2 – Gradiente descendente da solução da equação (3.4) (flechas sólidas) e a equação de Laplace (flechas tracejadas) sujeita a (3.5) na região quadrada $L \times M$ onde $L = M = 1$. 34
FIGURA 3.3 – Regiões do ambiente onde o robô pode visitar dependendo de seu comportamento. Procurar regiões dentro dos quadrados pontilhados representa um comportamento exploratório *seguir paredes* e em regiões dentro dos círculos pontilhados representa um comportamento de *preenchimento de espaço*. 36

FIGURA 4.1 – Arquitetura de um agente SMPA 39
FIGURA 4.2 – Arquitetura de um agente SPAE (Sense-Plan-Act-Explore). . . 40
FIGURA 4.3 – Situação dos atributos (estado, certeza, potencial) da região marcada na representação do ambiente ilustrada na Figura 4.2 e o campo vetorial gerado a partir do atributo potencial. . . . 43
FIGURA 4.4 – Atualização do mapa. A figura mostra um diagrama esquemático da detecção de um objeto por um dos sensores do robô. As células em cinza tiveram seus atributos certeza incrementados de 3, enquanto que as demais dentro do cone de atualização tiveram seu atributo diminuído de 1. A informação (d,α) representa a informação fornecida pelos sensores do robô no sistema de coordenadas polar. 45
FIGURA 4.5 – Ambiente com concavidades. As células brancas denotam regiões pertencentes ao espaço livre, as pretas denotam obstáculos e as cinzas denotam regiões não exploradas. 49
FIGURA 4.6 – Propagação do atributo *região* para as células do espaço livre a partir da posição do robô. As células de cor preta representam obstáculos e as de cor branca representam o espaço livre. . . . 50
FIGURA 4.7 – Propagação do atributo região. As Figuras (a)-(j) ilustram o processo de propagação de incremento e de decremento do atributo região. As células de cor preta representam obstáculos, as de cor branca representam espaço livre, as de cor cinza escuro representam regiões não exploradas e as de cor cinza claro representam células de espaço livre inatingíveis pelo robô. . . 51
FIGURA 4.8 – Processo de Exploração: Acima, a trajetória seguida pelo agente. (a)-(c) mostram *snapshots* do campo e do mapa para cada posição correspondente na trajetória. 55
FIGURA 4.9 – Janela de ativação. As Figuras (a)-(c) mostram a influência de uma janela de ativação fixa no mapeamento de um ambiente com diferentes dimensões. A Figura (d) mostra a influência de uma janela de ativação de tamanho variável no processo de mapeamento de um ambiente. Os arcos de cor preta ilustram células marcadas como ocupadas no mapa do ambiente. 57

FIGURA 4.10 – Exploração orientada a conhecimento local. A janela local denota todas as células que são utilizadas na relaxação local. A janela global denota todas as células que são utilizadas na relaxação global. 58

FIGURA 5.1 – Robô NOMAD 200 60
FIGURA 5.2 – Interface visual desenvolvida em C++/Linux 62
FIGURA 5.3 – Seqüência de exploração realizada pelo simulador do Nomad: A trajetória seguida pelo robô. O ambiente tem $10,2\ m\ \times\ 8,4\ m$, e o robô se desloca com passo constante $\Delta d\ =\ 7,6\ cm$. O campo potencial é atualizado usando Gauss-Seidel em um taxa $r\ =\ 30$ iterações/passo. 66
FIGURA 5.4 – Seqüência de exploração realizada pelo simulador do Nomad: Instâncias parciais do campo vetorial e do mapa durante a exploração. O campo vetorial representa o gradiente descendente em cada célula do espaço livre e é normalizado para facilitar sua visualização. No mapa, o valor de certeza das células é representado em níveis cinza, onde branco corresponde ao valor 0 o preto ao valor 15. O tamanho do mapa é 80×80 células, onde cada célula corresponde a $15,2\ cm\ \times\ 15,2\ cm$ da região do espaço real. 67
FIGURA 5.5 – Seqüência de exploração realizada pelo simulador do Nomad: Ambiente poligonal. O ambiente é $10,2\ m\ \times\ 8,4\ m$, e o deslocamento do robô corresponde ao passo $\Delta d = 7,6\ cm$. O campo potencial é atualizado usando Gauss-Seidel com uma taxa r = 30 iterações/passo. 68
FIGURA 5.6 – Exploração de um ambiente *interno* pelo robô Nomad 200. a) Mapa obtido após a exploração. O tamanho do mapa é 100×100 células, onde cada célula corresponde a uma região quadrada de $15,2\ cm \times 15,2\ cm$. A janela de ativação possui raio igual a $2\ m$. b) A trajetória seguida durante a exploração. Seu comprimento total é $L\ =\ 36,6\ m$ e foi navegada com velocidade média igual a $12,7\ cm/s$. As leituras do sonar são também mostradas. Observe que elas são mais ruidosas que o mapa. Isto se deve ao fato de que as leituras fora da janela de ativação são descartadas e somente as leituras consistentes atualizam o atributo certeza das células. 69
FIGURA 5.7 – Seqüência de Exploração.a) Trajetória seguida usando o método Gauss-Seidel para a taxa $r\ =\ 10$ iterações/passo. b) Trajetória seguida usando o método SOR para a taxa $r\ =\ 10$ iterações/passo. c) Trajetória seguida usando o método Gauss-Seidel para a taxa $r\ =\ 30$ iterações/passo. d) Trajetória seguida usando o método SOR para a taxa $r\ =\ 30$ iterações/passo. O ambiente é $10,2\ m$ por $10,2\ m$, e o robô se move em passos de $\Delta d\ =\ 7,6\ cm$. O tamanho do mapa é 80×80 células, onde cada célula corresponde a uma região quadrada de $15,2\ cm \times 15,2\ cm$ do espaço real. A janela de ativação possui raio igual a $2\ m$. 71

FIGURA 5.8 – Experimento com o robô Nomad 200 em um labirinto. As dimensões do ambiente são aproximadamente $3,5\ m \times\ 7,8\ m$. A janela de ativação tem raio igual a $2\ m$, e a taxa de iteração é igual a $r\ =\ 20$ iterações/s usando o método Gauss-Seidel. . 72
FIGURA 5.9 – Experimento de exploração com o robô NOMAD em nosso Instituto. a) Mapa do ambiente e a trajetória seguida pelo robô NOMAD. b) Retorno do robô NOMAD ao seu ponto de partida 73
FIGURA 5.10 – Experimento em ambientes com obstáculos dinâmicos - parte I. A figura ilustra o processo de mapeamento de um ambiente realizado pelo robô NOMAD na presença de pessoas. As pessoas são ilustradas por um círculo pontilhado. 76
FIGURA 5.11 – Experimento em ambientes com obstáculos dinâmicos - parte II. A figura ilustra o processo de mapeamento de um ambiente realizado pelo robô NOMAD na presença de pessoas. As pessoas são ilustradas por um círculo pontilhado. 77
FIGURA 5.12 – Exploração orientada a objetivos: trajetória seguida pelo robô. O ambiente é $10,2\ m\ \times\ 10,2\ m$, e o robô se desloca a cada instante $\Delta d = 7,6\ cm$. O tamanho do mapa é 80×80 células, onde cada célula corresponde a uma região quadrada de $15,2\ cm \times 15,2\ cm$ do espaço real. A janela de ativação é um círculo com raio igual a $2\ m$. A taxa de iteração é igual a $r\ =\ 30$ iterações/passo usando Gauss-Seidel. 78
FIGURA 5.13 – Exploração orientada a objetivos: mapa e o campo vetorial. Como antes, o campo vetorial representa o gradiente descendente em cada célula do espaço livre. Ele é normalizado para facilitar a sua visualização. No mapa, o valor de certeza é representado em níveis de cinza, onde o valor 0 corresponde à célula de cor branca e o valor 15 à célula de cor preta. . . . 79
FIGURA 5.14 – Ambiente para teste da janela adaptativa. Suas dimensões externas são aproximadamente $6,8\ m \times\ 6,0\ m$. 80
FIGURA 5.15 – Experimento de exploração usando as janelas de ativação adaptativa e fixa. a) e b) usando janela adaptativa. c) e d) usando janela fixa com raio de ativação igual a $2\ m$. As dimensões externas do ambiente são aproximadamente $6,8\ m \times\ 6,0\ m$. 81
FIGURA 5.16 – Conhecimento global × conhecimento adaptativo. a) exploração orientada pelo conhecimento global. b) exploração orientada pelo conhecimento adaptativo. 82
FIGURA 5.17 – Quantidade de células atualizadas durante o processo de exploração de um ambiente interno típico. 83
FIGURA 5.18 – Seqüência de exploração nos ambientes esparsos usando o potencial harmônico. O ambiente **A** foi explorado seguindo um caminho de comprimento $L = 10,1\ m$, enquanto que no ambiente **B**, este caminho foi igual a $L = 40,1\ m$. 86
FIGURA 5.19 – Seqüência de Exploração usando a regra 1 no ambiente **A**. Para $\epsilon = 0,5$, $L = 5\ m$ e para $\epsilon = 0,8$, $L = 4,6\ m$. 87

FIGURA 5.20 – Seqüência de Exploração usando a regra 2 no ambiente **A**. Para $\epsilon = 0,5$, $L = 7,3\ m$ e para $\epsilon = 0,8$, $L = 7,6\ m$. 87
FIGURA 5.21 – Seqüência de Exploração no ambiente **B**. Para a regra 1 e $\epsilon = 0.8$, $L = 26,3\ m$ e para a regra 2 e $\epsilon = 0,8$, $L = 36,2\ m$. . 88
FIGURA 5.22 – Ambiente esparso para teste da direção do vetor **v** da regra 1. 89
FIGURA 5.23 – Seqüência de exploração usando as funções harmônicas e a regra 1 com $\epsilon = 0,7$. Para as funções harmônicas, $L = 194,9\ m$ e para a regra 1, $L = 156,0\ m$. 90

FIGURA 6.1 – Campo vetorial parcialmente relaxado utilizando o método SOR após 30 iterações de atualização. As setas indicam a posição dos obstáculos, objetivo, a presença de padrões distorcidos semelhantes à ondas, e de vetores que apontam em direção a um obstáculo. 95

Lista de Tabelas

TABELA 5.1 – Resultado da simulação para a exploração aleatória, *wall following* e as funções harmônicas no ambiente da Figura 5.7 sobre 30 execuções. 70
TABELA 5.2 – Resultado da simulação utilizando o simulador do Nomad no ambiente da Figura 5.7. As dimensões do ambiente são $10,2\ m \times 10,2\ m$, a janela de ativação tem raio igual a $2\ m$, e o desempenho médio foi extraído a partir de 10 execuções. . 72
TABELA 5.3 – Resultado médio extraído de 4 experimentos no labirinto da Figura 5.8 com o robô Nomad. As dimensões do ambiente são $3,5\ m \times\ 7,8\ m$. A janela de ativação tem raio igual a $2m$, e a taxa $r\ =\ 20$ iterações/s. 72
TABELA 5.4 – Tempo médio extraído de 6 experimentos realizados com o robô Nomad em nosso Instituto. 73
TABELA 5.5 – Discrepância e comprimento médio do caminho seguido pelo robô sobre 5 experimentos. 74
TABELA 5.6 – Quantidade média de células atualizadas durante todo o processo de exploração de um ambiente típico usando conhecimento global e adaptativo sobre 10 experimentos. 82
TABELA 5.7 – Quantidade média de células visitadas durante todo o processo de exploração de um ambiente típico usando conhecimento global e adaptativo sobre 10 experimentos. 83
TABELA 5.8 – O caminho médio seguido pelo robô durante todo o processo de exploração de dois ambientes típicos usando conhecimento global e adaptativo sobre 10 experimentos. 84
TABELA 5.9 – Resultado do desempenho médio e o desvio padrão correspondente a 6 diferentes julgamentos feitos com as duas principais regras e o potencial harmônico. 88
TABELA 5.10 – Resultados médio obtidos sobre 10 execuções com diferentes direções de $\mathbf{v}$. 89
TABELA 5.11 – Resultado médio de 4 execuções com o robô NOMAD 200 iniciando de diferentes posições no ambiente usando as duas principais regras. 90

Resumo

Este trabalho apresenta e discute uma estratégia inédita para o problema de exploração e mapeamento de ambientes desconhecidos usando o robô NOMAD 200. Esta estratégia tem como base a solução numérica de problemas de valores de contorno (PVC) e corresponde ao núcleo da arquitetura de controle do robô. Esta arquitetura é similar à arquitetura *blackboard*, comumente conhecida no campo da Inteligência Artificial, e é responsável pelo controle e gerenciamento das tarefas realizadas pelo robô através de um programa cliente. Estas tarefas podem ser a exploração e o mapeamento de um ambiente desconhecido, o planejamento de caminhos baseado em um mapa previamente conhecido ou a localização de um objeto no ambiente. Uma característica marcante e importante é que embora estas tarefas pareçam diferentes, elas têm em comum o mesmo princípio: solução de problemas de valores de contorno.

Para dar sustentabilidade a nossa proposta, a validamos através de inúmeros experimentos, realizados em simulação e diretamente no robô NOMAD 200, em diversos tipos de ambientes internos. Os ambientes testados variam desde labirintos formados por paredes ortogonais entre si até ambientes esparsos. Juntamente com isso, introduzimos ao longo do desenvolvimento desta tese uma série de melhorias que lidam com aspectos relacionados ao tempo de processamento do campo potencial oriundo do PVC e os ruídos inseridos na leitura dos sensores. Além disso, apresentamos um conjunto de idéias para trabalhos futuros.

Palavras-chave: Mapeamento de Ambientes, Navegação Exploratória, Problemas de Valores de Contorno, Funções Harmônicas, Robótica Móvel.

TITLE: "EXPLORATORY NAVIGATION BASED ON BOUDARY VALUE PROBLEM"

Abstract

This work presents and discusses a new strategy to the problem of exploration and mapping of an unknown environment using the NOMAD 200 robot. This strategy is based on the numerical solution of boundary value problems and corresponds to the core of the architecture of the robot. This architecture resembles the blackboard architecture, commonly known in the Artificial Intelligence. It is in charge of controlling and of managing the progress of the tasks performed by the robot through a client program. These tasks can be exploration and mapping of unknown environments, path planning based on map known previously or localization of objects in the environment. Even though those tasks can be different, they have the same principle: solution of boundary value problems.

To give support to our proposal, we validated it through several experiments performed in simulations and in the NOMAD robot. These experiments were achieved in indoor environments with maze-like structures and sparse environments. Besides, we introduced improvements related to potential computation and the treatment of noisy readings coming from the sensors. Furthermore, we present a set of ideas for future work.

Keywords: Environment Mapping, Exploratory Navigation, Boundary Value Problem, Harmonic Function, Mobile Robot.

1 Introdução

A robótica está deixando de ser apenas ficção científica passando a estar cada vez mais presente na vida das pessoas através de máquinas que auxiliam o homem na realização de tarefas. Hoje, a pesquisa em robótica está direcionada à criação de robôs capazes de realizar tarefas sem intervenção humana, onde o usuário define através de uma descrição de alto-nível o que o robô deve executar, ao invés de como executar.

Embora este comportamento seja o esperado, há algumas décadas o robô era definido como um equipamento multifuncional reprogramável, como observado na definição dada por Ullrich [ULL 87]:

> *um robô é um equipamento multifuncional e reprogramável, projetado para mover materiais, peças, ferramentas ou dispositivos especializados através de movimentos variáveis e programados para a execução de uma série de tarefas.*

Atualmente a definição de robô está intimamente ligada à definição de agente em Inteligência Artificial(IA). Do ponto de vista da IA, o robô é um agente que irá atuar em um dado ambiente realizando ações de acordo com a informação oriunda de seus sensores.

Existem diversas definições para agente [RUS 95], cada uma delas abordando uma característica desejável:

- **focalizando a sobrevivência**: um agente é um sistema capaz de manter a si próprio;
- **focalizando a percepção/ação**: um agente é alguma coisa que percebe através de seus sensores e atua através de seus atuadores;
- **focalizando a capacidade de aprendizado**: um agente é um sistema que muda seu comportamento como resultado de suas ações passadas.

O conjunto destas definições compõe um conceito importante em robótica chamado *autonomia*. Um sistema robótico é dito autônomo quando ele é capaz de estender seu comportamento a partir do seu conhecimento sobre o efeito de suas ações no ambiente [RUS 95]. Em alguns casos, um sistema pode definir sua atuação baseado apenas em seu conhecimento prévio embutido sem levar em consideração sua percepção. Neste caso, dizemos que este sistema não possui *autonomia*.

Um sistema móvel autônomo requer um sistema de controle que seja capaz de apresentar comportamento inteligente para resolver diferentes problemas e usar sua própria experiência de uma maneira inteligente [PET 97]. Além disso, esse sistema deve ser capaz de interagir e realizar tarefas em um ambiente dinâmico e desconhecido e necessita que sua funcionalidade degrade gradualmente no caso de ocorrência de falhas internas ou obstruções externas.

Durante muitos anos a comunidade de IA tem construído sistemas de controle que apresentam comportamento inteligente. Infelizmente, na maioria dos casos, estes sistemas atuam em um mundo simulado ou restrito. As pesquisas tradicionais de controle de tempo real têm desenvolvido sistemas de controle que atuam em um

mundo real, entretanto, na maioria das vezes sem comportamento inteligente ou autonomia verdadeira.

Dentre os diferentes campos da robótica, este trabalho destaca o campo da robótica móvel autônoma. Um dos componentes básicos de operação destes robôs é sua capacidade de movimentação em terrenos cujos modelos nem sempre são conhecidos.

A pesquisa em planejamento de movimentos começa na década de 60 durante o estágio inicial do desenvolvimento de robôs controlados por computadores. Apesar disso, a maioria dos trabalhos na área foi realizada durante a década de 80. Estes tipos de robôs possuem ampla aplicabilidade em diversas áreas, entre elas podemos citar: exploração espacial, manufatura, construção, assistência a paraplégicos, entre outros.

É importante salientar que o planejamento de movimento é muito mais que apenas checar e evitar colisões [LAT 93]. Ele envolve aspectos relacionados a: descoberta de caminhos livres de colisões; coordenação simultânea de múltiplos robôs; planejamento de caminhos estáveis e suaves; raciocínio sob incerteza baseado em informações sensoriais; tratamento de modelos com propriedades físicas tais como massa, gravidade e fricção; entre outros. Dessa forma, planejamento de movimento requer considerar tanto restrições geométricas quanto físicas e temporais. Além disso, a incerteza requer que o planejamento não seja limitado a comandos de movimento, exigindo que ele possua uma análise completa das informações sensoriais.

Neste trabalho, focalizamos o problema de exploração e mapeamento de ambientes desconhecidos utilizando o robô NOMAD 200. A estratégia de planejamento utilizada é baseada na solução de problemas de valores de contorno sobre um mapa representado por uma matriz bidimensional cujas células correspondem a posições específicas no ambiente e armazenam um valor computado a partir das informações oriundas dos sensores ultra-sônicos. Uma característica marcante neste trabalho é a união de diferentes estratégias - planejamento de caminhos livres de obstáculos em ambientes conhecidos e a aquisição de mapas através de um processo exploratório - através do mesmo princípio: solução de problemas de valores de contorno.

1.1 Motivação

A principal motivação para o desenvolvimento deste trabalho foi responder à seguinte pergunta: existe um princípio único capaz de fornecer a base para um algoritmo robusto de exploração e ao mesmo tempo ser uma ferramenta para planejamento de caminhos?

Esta motivação nos leva à investigação de uma teoria que possa unificar a compreensão e a solução de problemas relacionados a planejamento através de um princípio único, simples, robusto e eficiente.

O que nos levou a fazer esta pergunta foi nossa tentativa de evitar o chaveamento entre diferentes soluções algorítmicas, principalmente durante o processo de exploração. Neste caso, as abordagens apresentadas na literatura realizam isto através da união de diferentes estratégias [ROM 2000, ROM 2001a, SCH 99, THR 98]. Além disso, algumas delas estão limitadas ao comportamento exploratório não sendo possível serem utilizadas diretamente como planejadores de caminhos sobre o mapa aprendido.

Juntamente com a motivação inicial, está o fato de podermos realizar experimentos com o robô NOMAD 200. Este robô está disponível em nosso Instituto através do Laboratório de Robótica Inteligente (LRI). Uma outra fonte de motivação é a possibilidade de utilizar esta proposta em operações de resgate, transporte de materiais, recolhimento e entrega de alimento em hospitais, entre outros.

1.2 Objetivos

O objetivo desta tese consiste em desenvolver uma estratégia que integre a capacidade exploratória e o planejamento de caminhos através de um princípio único: campos potenciais oriundos da solução de problemas de valores de contorno(PVC). Esta estratégia é inserida em uma arquitetura de agente similar à arquitetura *blackboard*[CAR 92, PET 97]. Nosso ponto de partida foi o planejador de caminhos baseado em funções harmônicas proposto por Connolly [CON 93]. Neste trabalho, daremos mais atenção ao processo exploratório, ao invés do planejamento de caminhos, pois o último é conhecido e já foi validado anteriormente[CON 93].

Inicialmente esta integração será testada no ambiente de simulação desenvolvido pelo autor na plataforma C++/linux utilizando diferentes configurações de obstáculos. Isto objetiva extrair medidas estatísticas de desempenho a partir dos parâmetros do sistema.

Em seguida, a arquitetura será validada no robô NOMAD 200 em diferentes tipos de configuração de ambientes montados em nosso Instituto a fim de verificar o desempenho da estratégia e a qualidade dos resultados gerados. O robô será inserido em ambientes internos densos e esparsos. Os ambientes internos esparsos consistem em ambientes onde o robô navega durante um longo tempo sem perceber qualquer obstáculo. Devido à falta de locais suficientemente amplos, simularemos os ambientes esparsos diminuindo por software o alcance dos sensores do robô.

Finalmente, iremos comparar nossos resultados às abordagens tradicionais de exploração: aleatória e seguir paredes (*wall-following*).

1.3 Organização

Este trabalho está organizado da seguinte maneira. O Capítulo 2 introduz o problema de exploração de ambientes, apresentando um estudo sobre seus principais componentes e alguns trabalhos relevantes relacionados. O Capítulo 3 apresenta o formalismo matemático juntamente com a prova de convergência de nossa estratégia. O Capítulo 4 mostra o sistema desenvolvido do ponto de vista da IA. O Capítulo 5 apresenta os resultados obtidos utilizando simulação e o robô NOMAD. O último capítulo discute os resultados obtidos e apresenta as conclusões gerais e propostas de trabalhos futuros.

2 Exploração de Ambientes

O comportamento exploratório é fundamental para a autonomia de um robô. Ele dá suporte ao processo de tomada de decisão indicando qual é o conjunto de ações que permitirão ao robô conhecer e atuar eficientemente em um ambiente desconhecido previamente. As decisões são tomadas baseadas em todo conhecimento já adquirido sobre ambiente e o principal critério é, na maioria das vezes, a escolha de ações que minimizam a incerteza ou a ignorância sobre o ambiente. Os problemas mais comuns a serem resolvidos por este tipo de planejamento em robótica móvel são a descoberta de objetivos e a aquisição de mapas.

Este tipo de planejamento tem obtido crescente interesse especialmente em ambientes hostis ou inacessíveis [DUD 93], por exemplo: em uma exploração espacial, o atraso no envio de comandos da base na terra ao robô representa um problema importante de comunicação que dificulta a utilização de tele-operação. Problemas de comunicação afetam a aplicação com submarinos robôs, pois a utilização de cabos para a comunicação do robô com a base limita a região a ser explorada, enquanto que a sua eliminação pode facilmente deixar o robô perdido no ambiente [MCK 95].

Antes de discutirmos os principais componentes de um sistema de exploração iremos situá-lo entre os principais trabalhos relacionados à navegação. Na Seção 2.1 classificaremos os métodos de navegação comumente usados de acordo com a incerteza da aplicação, situando entre eles o processo de exploração. Na Seção 2.2, apresentaremos as principais estratégias utilizadas na exploração e aquisição de mapas de ambientes. Na Seção 2.3, discutiremos os componentes básicos para o desenvolvimento de um robô autônomo com capacidade exploratória. Finalmente, na Seção 2.4, apresentaremos alguns trabalhos relevantes encontrados na literatura.

2.1 Métodos de Navegação

A maioria dos trabalhos relacionados ao planejamento de movimentos está associada ao planejamento de caminhos, ou em ambientes conhecidos, como no caso de *roadmaps*, ou em ambientes desconhecidos utilizando apenas a informação local obtida através dos sensores e a informação global representada pela posição a ser alcançada.

Estes problemas são classificados como problemas navegacionais. A navegação é definida por McKerrow [MCK 95] como *a ciência ou arte de direcionar o curso de um robô móvel quando ele se desloca em um ambiente.* Na maioria das vezes, a navegação se refere simplesmente ao planejamento de um caminho viável entre duas posições em um espaço bidimensional conhecido, ou como em diversos casos, o planejamento é baseado apenas nos sensores sem conhecimento prévio sobre o ambiente.

Entretanto, é possível classificar os métodos de navegação em 3 tipos de acordo com a incerteza envolvida na aplicação[DAM 98]:

- **métodos globais** são baseados no conhecimento total sobre a estrutura do ambiente onde o robô está inserido e sua localização dentro dele. Neste caso, a tarefa de navegação se restringe ao planejamento de um caminho viável entre duas posições utilizando um mapa global. Estes tipos de métodos podem

ser divididos em três principais abordagens [LAT 93]: mapas de caminhos (*roadmaps*), decomposição celular e campo potencial[1];

- **métodos locais**, diferentemente dos métodos globais, o robô não possui informação alguma sobre o ambiente e sobre sua localização nele. A única informação que ele possui é um mapa local construído a partir da informação local oriunda de seus sensores. Em geral, métodos locais são chamados de *reativos* e são geralmente utilizados para tratar eventos inesperados.

  É possível estender sua funcionalidade adicionando um conhecimento global, por exemplo: a localização relativa da posição a ser alcançada. Neste caso, a tarefa de navegação é uma tarefa reativa orientada a um objetivo. Os métodos locais amplamente conhecidos são os campos potenciais propostos por Khatib [KHA 96], e a estratégia *seguir paredes* utilizada em alguns casos para explorar e mapear ambientes [CRO 85a, MEK 97] ou simplesmente manter um carro em uma autopista [POM 89].

- **métodos híbridos** são métodos que combinam informações locais e globais. Em diversos casos é comum o mapa ou a geometria do ambiente ser desconhecida, principalmente em ambientes naturais, também chamados *externos*. Neste caso, o planejamento básico de um caminho entre duas posições utilizando um método global se torna inútil, pois ele necessita de um modelo completo do ambiente para funcionar adequadamente, enquanto que a utilização de um método local não garante que o robô irá encontrar sua posição destino.

  Combinando os métodos locais e globais é possível gerar um método híbrido que herda e estende a aplicabilidade de ambos os métodos. Este novo método possibilita adquirir a estrutura do ambiente, através da integração de diferentes mapas locais em um mapa global; corrigir os erros de posicionamento do robô, realçando sua localização; corrigir informações inconsistentes armazenadas no mapa do ambiente; identificar rotas alternativas à posição objetivo, no caso de obstrução do caminho corrente; evitar colisões durante a navegação; enfim, realizar uma navegação mais robusta e precisa no ambiente.

O planejamento exploratório é um método híbrido, pois para ter sucesso ele precisa constantemente realizar leituras locais do ambiente e armazená-las em uma representação que possibilite facilmente identificar as regiões já visitadas pelo robô para que ele realize eficientemente sua exploração. Este comportamento elimina ou minimiza a necessidade de qualquer conhecimento prévio sobre a estrutura da área de atuação do robô facilitando sua readaptação em diferentes áreas de trabalho sem a necessidade de sua reprogramação.

2.2 Planejamento Exploratório

A construção de um mapa confiável e estável do mundo é feita através de um longo processo de interação do robô com o ambiente. O robô usa seus sensores para sensoriar o ambiente e estimar a posição dos obstáculos, e em seguida, adicionar ou substituir estas informações em sua representação interna. Esta estimativa é o

[1] É importante destacar que os campos potenciais utilizados como métodos globais são baseados na representação utilizada pelos métodos de decomposição celular [STE 94, RAT 95]

retorno mais importante do ambiente obtido pelos sensores. Um mapa pode ser criado através de navegação direcionada [BOR 91, ZEL 92, STE 95, ORI 97] ou através de um planejamento exploratório [PRE 2001, PRE 2002, YAM 98a].

As estratégias comumente usadas para exploração [ROM 2000] são divididas em duas classes: abordagem *reativa* e abordagem *baseada em modelo*. A abordagem reativa comumente utilizada é a estratégia *seguir paredes* [MEK 97, MAT 90, CRO 85a]. Ela consiste em guiar o robô paralelamente às paredes enquanto extrai o mapa topológico [MAT 90] ou geométrico [CRO 85a] do ambiente (mais detalhes sobre tipos de mapas, ver Seção 2.3.1). Esta estratégia funciona corretamente quando o ambiente a ser explorado possui obstáculos e paredes conectadas. Quando o ambiente é esparso, ou seja, o ambiente possui obstáculos cuja localização está além da capacidade perceptiva dos sensores do robô, esta estratégia deve ser mesclada com outras a fim de realizar o mapeamento completo do ambiente. Outro problema relacionado a esta abordagem é a facilidade de cair em mínimos locais[ROM 2000].

Por outro lado, as abordagens baseadas em modelo usam como núcleo do processo exploratório uma função que descreve quais são as ações preferenciais a serem executadas dado o estado corrente da exploração e o conhecimento obtido sobre o ambiente. Esta função é chamada *função de custo*, *função de recompensa* ou *função heurística*.

Existem diversas funções propostas na literatura. A maioria delas descreve uma maneira de alcançar a região desconhecida mais próxima [YAM 97, CHO 2000a]. Por exemplo, a exploração baseada em fronteiras de Yamauchi [YAM 97], onde o *limite* entre as regiões conhecidas e desconhecidas do ambiente, chamadas *fronteiras*, direcionam o robô através de um comportamento baseado no clássico algoritmo de busca em grafos chamado *busca em profundidade*. A informação de *Fischer* sobre as incertezas nas medidas sensoriais e navegacionais pode ser utilizada como uma função heurística [FED 99] durante a construção do mapa do ambiente. Neste caso, o robô tende a escolher ações que maximizem seu ganho de informação no sentido definido por esta função enquanto minimizam a incerteza da localização dos objetos e do robô no ambiente. Similarmente, Thrun [THR 98] desenvolveu uma estratégia onde uma função heurística é calculada iterativamente através de um algoritmo clássico da programação dinâmica chamado *iteração valorada*. Após a convergência, a função indica caminhos que robô deve seguir a fim de minimizar a quantidade de regiões desconhecidas do ambiente. Este algoritmo foi modificado por Romero [ROM 2000] a fim de dotar o robô do comportamento gerado pela estratégia *seguir paredes* produzindo uma abordagem híbrida que mantém o robô ao redor dos obstáculos enquanto ele realiza a exploração do ambiente.

Em relação à tarefa a ser realizada, o planejamento exploratório pode ser classificado em:

- **guiado por objetivo**: quando o robô não possui nenhum conhecimento prévio sobre a estrutura do ambiente e deve extrair informações sobre a localização de elementos dispostos no ambiente a partir unicamente da leitura de seus sensores. Este é um problema clássico de busca em um labirinto, onde um tesouro deve ser encontrado ou deve-se achar uma saída [RAO 93]. Na maioria das vezes, a solução deste problema é modelar o ambiente como um grafo e utilizar um algoritmo de busca conveniente;

  Um dos primeiros a resolver este tipo de problema foi Shannon em 1940 [RAO 93]. O rato de Shannon era capaz de descobrir um queijo armazenado

em um dos 25 quadrados de um labirinto utilizando um algoritmo que armazenava a direção de saída do rato para cada célula e calculava o próximo movimento do rato baseado nesta informação. Tarry [RAO 93] utilizou um algoritmo similar a busca em profundidade para resolver o problema do labirinto, o qual foi posteriormente melhorado por Fraenkel (para maiores detalhes ver [RAO 93]). Em alguns casos, haverá também a construção e o armazenamento de um mapa parcial do ambiente, além da informação da localização dos objetos de interesse[RAO 93, PRE 2001].

- **não-guiado**: quando o objetivo é a aquisição e o armazenamento da estrutura do ambiente em uma representação interna. Um exemplo é a estratégia *Sightseer* [RAO 93]. Ela considera que existe no mínimo um obstáculo visível ao robô. A estratégia basicamente corresponde a uma busca em profundidade, onde o robô deve visitar sempre o obstáculo mais próximo não visitado a partir de sua posição, contorná-lo e marcá-lo como visitado. Se todos os obstáculos ao redor do robô já estiverem sidos visitados, o robô retorna ao obstáculo visitado recentemente e repete o processo. Ao final, é gerado um grafo não direcionado mostrando a ordem de visita do grafo (para maiores detalhes ver [RAO 93]).

  Outros exemplos são a estratégia proposta por Choset e os algoritmos de Betke. Choset[CHO 2000a] propõe uma abordagem incremental para o aprendizado do diagrama de Voronoi generalizado hierárquico [CHO 2000] do ambiente utilizando uma navegação baseada em um algoritmo de busca em profundidade. Betke [BET 95] propõe dois algoritmos chamados *wavefront* e *ray* para realizar o aprendizado gradativo de ambientes constituídos de obstáculos retangulares. Os algoritmos são baseados nas técnicas de busca em grafos e levam em consideração restrições relacionadas ao tempo de duração da exploração e de combustível.

No processo exploratório, os obstáculos são os protagonistas. Eles definem regiões do ambiente que correspondem a pontos de atração ou repulsão para o robô. Além disso, eles impõem uma ordem de exploração que, na maioria das vezes, é baseada na distância entre eles e o robô. É possível também observar que o processo exploratório está intimamente relacionado com os algoritmos de busca em grafo.

2.3 Componentes Básicos

O planejamento exploratório é um problema navegacional. A maioria dos trabalhos considera situações ideais, poucos foram realmente implementados em robôs reais. Para que o sistema seja aplicado em um robô real, é necessário que os seguintes problemas sejam resolvidos:

- **percepção/representação** : qualquer decisão feita pelo robô tem que ser baseada em sua percepção local e em seu conhecimento global. O conhecimento global corresponde à união da percepção local em diferentes instantes durante a exploração, o qual é armazenado em uma representação que corresponde ao modelo do ambiente. Este modelo é utilizado no planejamento de movimentos para guiar o robô ao seu objetivo e durante a exploração para indicar ao robô os locais ainda não visitados;

- **auto-localização e mapeamento concorrente**: a construção de um modelo confiável do ambiente deve ser realizada considerando os erros gerados pelo deslocamento do robô. Estes erros correspondem a erros de odometria, que são observados através de comparações entre a percepção local e do conhecimento global sobre o ambiente, e devem ser reduzidos através de um processo de auto-localização;

- **planejamento de movimento**: o robô deve possuir um mecanismo para decidir qual é o caminho a seguir, localmente ou a médio prazo. Isto somente é possível devido ao seu conhecimento armazenado sobre o ambiente e de sua percepção.

2.3.1 Representação do Ambiente

Uma das questões fundamentais a ser considerada em robótica móvel é a forma como o ambiente é representado internamente pelo robô. As representações comumente utilizadas na literatura são grades de ocupação, objetos, lugares, rotas, etc [ROM 2001]. Em geral, estas representações podem ser divididas em dois grandes grupos: mapas métricos e mapas topológicos. Os mapas métricos podem ser subdivididos em aqueles que realizam *decomposição espacial* do ambiente, grades de ocupação, e aqueles que utilizam uma *representação geométrica*, mapas baseados em características e mapas probabilísticos.

As **grades de ocupação** [ELF 87, ELF 89] representam o ambiente como uma matriz de células, onde cada célula está geograficamente relacionada com uma região quadrada do ambiente e armazena um valor que indica a probabilidade de ocupação desta área. As principais vantagens deste tipo de mapa são as facilidades de representação e a possibilidade de integrar leituras de diferentes sensores.

O método pioneiro neste tipo de representação foi a *certainty grid* desenvolvida em Carnegie Mellon [MOR 85] para lidar com a incerteza espacial gerada pelos sensores do tipo *sonar*. Esta incerteza está relacionada à posição dos objetos e é descrita como uma distribuição espacial de probabilidades dentro da grade de ocupação. Quanto maior a incerteza maior será o número de células ocupadas pelo objeto observado.

Existem diversas variações deste tipo de representação. A principal diferença entre elas é a função utilizada para a atualização das células da grade. Esta função pode ser baseada em certeza Fuzzy [ORI 97], Bayes [HOW 96], heurística [ELF 87], gaussiana [ELF 89], freqüência [BOR 91, PRE 2001], entre outros.

Este tipo de representação possui problemas de granularidade, escalabilidade e extensibilidade [BAI 2001]. Estes estão relacionados ao tamanho fixo do mapa e de suas células e acarretam sobrecarga computacional e limites nas dimensões e na precisão do ambiente representado. Uma solução alternativa é utilizar estruturas do tipo *quadtree* ou *octrees* [FOL 90], as quais permitem utilizar células de tamanho variável. Outro problema é a dificuldade em representar entidades simbólicas como: portas, cadeiras, etc [FOX 99]. Além de não oferecer mecanismos para tratar obstáculos estáticos diferentemente de obstáculos dinâmicos.

Os **mapas baseados em características** representam o ambiente através da localização de todas as características relevantes do ambiente e suas propriedades geométricas (mapas de elevação, mapas poligonais, bolhas tridimensionais)[CRO 85, DEV 95, CHA 96].

Este tipo de representação foi inicialmente proposto por Chatila e Laumond [CHA 89] para ambientes *internos*. Eles representam o ambiente através de segmentos de linhas, arcos de círculos, pontos e poliedros a partir da informação oriunda dos sensores de proximidade juntamente com uma incerteza associada a sua localização. Esta formulação é geralmente utilizada em conjunto com o filtro de Kalman para minimizar o erro de estimativas dos objetos extraídos do ambiente e da posição e orientação do robô [CRO 85].

A principal vantagem da utilização de primitivas geométricas é a eficiência na representação do ambiente. Entretanto, este tipo de representação possui três principais problemas [ROM 2001]: perda de estabilidade, pois a representação pode ser prejudicada por pequenas variações das leituras dos sensores; dificuldade na localização, pois devido esta representação ser baseada em primitivas geométricas é possível que a representação não seja única; e perda de poder expressivo, decorrente da dificuldade em representar completamente todas as características do ambiente através de primitivas geométricas.

Em relação às grades de ocupação, Dam [DAM 98] faz algumas comparações importantes.

- se o ambiente é desconhecido, um mapa baseado em características é muito mais complicado de ser construído;
- um mapa baseado em características é difícil de ser utilizado em ambientes não-estruturados, devido a dificuldade de modelar alguns obstáculos usando primitivas geométricas;
- a posição do robô pode ser calculada de forma mais eficiente utilizando mapas baseados em características do que utilizando grades de ocupação.
- a principal vantagem da representação baseada em grade é a possibilidade de representar tanto o espaço livre quanto o espaço ocupado do ambiente, os quais podem ser facilmente utilizados em algoritmos de navegação.

Os **mapas probabilísticos** foram propostos por Smith, Self e CheeseMan [SMI 87] como uma ferramenta para tratar as incertezas espaciais originadas das medidas sensoriais e dos movimentos do robô na representação espacial, ao invés de considerá-las como um problema geométrico à parte a ser tratado. Neste tipo de representação, tanto os obstáculos encontrados quanto o robô são representados como um conjunto de objetos descritos pela sua localização juntamente com uma matriz de covariância que descreve o relacionamento espacial entre eles [HEB 95].

Este tipo de representação é comumente utilizado com o filtro de Kalman estendido, dando origem a um algoritmo MLS (ver Seção 2.3.3). O filtro de Kalman foi proposto em 1960 por R. E. Kalman como uma ferramenta recursiva para o problema de filtragem linear de dados discretos [WEL 2001]. Sua principal característica é a capacidade de estimar estados passados, correntes e futuros mesmo quando a dinâmica do sistema é desconhecida. Esta propriedade é utilizada para atualizar e corrigir o estado do sistema e a sua incerteza associada. Esta estratégia tem sido utilizada em ambientes *externos*, como sub-aquático[SMI 97], e *internos* [FED 99, LEO 92].

O principal problema deste tipo de representação é a necessidade de se armazenar uma matriz de covariância para a convergência do mapa. Esta matriz

cresce quadraticamente com a quantidade de objetos observados resultando em alto custo computacional quando o ambiente a ser mapeado é grande, o que inviabiliza sua utilização em aplicações de tempo-real [DIS 2001].

Os **mapas topológicos** correspondem a um grafo composto de vértices e arestas, onde os vértices correspondem a locais distintos, os quais são locais de distância máximas entre os obstáculos, e as arestas correspondem a ligações entre diferentes locais contendo informações que permitirão ao robô navegar entre os vértices [KUI 91]. Estas informações correspondem ou ao conjunto de comportamentos que permitirão ao robô ir de um vértice a outro, ou a informações métricas sobre a posição relativa entre os vértices [FOX 99].

Os vértices representam locais distinguíveis no ambiente baseados no padrão observado nos dados sensoriais [BAI 2001]. Estes padrões são representações comuns de portas, interseções de corredores, etc. Em ambientes *internos*, as arestas correspondem a corredores e os *caminhos médios* correspondem a vértices.

De acordo com Dudek [DUD 93], a principal vantagem desta abordagem é a sua natureza qualitativa e sua relação com teorias de cognição humana. Além disso, os mapas topológicos são geralmente menos complexos que os mapas geométricos; muito mais eficientes no processo de planejamento de caminhos [SHA 97] e mais flexíveis à noção de estado.

2.3.2 Auto-Localização

O problema de auto-localização corresponde a uma área vasta da robótica móvel que tem recebido muita atenção durante as últimas décadas, pois se um robô não sabe sua posição, ele não pode eficientemente planejar movimentos, localizar ou alcançar objetivos [OLS 99].

Este problema consiste em determinar a localização do robô em relação as suas posições prévias ou/e em relação ao seu mapa interno. Este processo é comumente modelado como uma função que expressa a adequação de posições candidatas à posição real do robô em função do número e das distâncias dos *landmarks* observados [CHA 96]. Bailey [BAI 2001] destaca que um robô autônomo genuíno é aquele que consegue se auto-localizar em um ambiente usando apenas seus sensores embarcados sem necessitar que o ambiente seja artificialmente modificado.

Planejadores de movimentos clássicos contam apenas com odômetros para determinar sua posição. Este mecanismo chamado de *dead-reckoning* corresponde a integrar os dados fornecidos pelos odômetros das rodas do robô, os quais contam a quantidade de rotações realizadas por cada roda. Este processo provou ser suscetível a erros [THR 98] por diversas razões: insuficientes mecanismos de alinhamentos das rodas; derrapagem; erros no sinal dos sensores; variações na trajetória devido a diferença de superfícies; entre outros.

Em particular, os odômetros geram dois tipos de erros: sistemáticos e não-sistemáticos. Os erros *sistemáticos* são dependentes inteiramente das características da plataforma móvel, enquanto que os erros *não-sistemáticos* são gerados a partir da interação entre o robô e o ambiente. Os erros sistemáticos podem ser preditos; alguns são determinísticos (o diâmetro das rodas é menor do que aquele modelado pelo sistema, o sistema irá sempre superestimar a distância percorrida) , enquanto que alguns podem ser modelados por distribuição probabilística (a resolução finita do odômetro causa um erro com distribuição normal). Os erros não-sistemáticos sempre são modelados por funções probabilísticas e estão associados aos sensores do

robô.

As estratégias para a solução do problema de auto-localização podem ser divididas em dois grupos: técnicas de posicionamento relativo e absoluto. Posicionamento relativo é baseado nos odômetros, os quais estimam a posição corrente do robô integrando as revoluções de suas rodas, e na percepção local do robô [YAM 98a, DEV 95, BET 94]. O posicionamento absoluto é baseado em sensores absolutos como GPS, ou em *landmarks* [BET 94a, BET 97], cuja localização é conhecida no ambiente. Dependendo do deslocamento do robô, os GPS são mais precisos que os odômetros. Entretanto em determinados ambientes é impraticável seu uso, um exemplo é o ambiente sub-aquático [FED 99]. Neste caso, o robô deve vir à superfície para se localizar, o que é indesejável.

A maioria dos métodos de localização falham em estimar precisamente a posição do robô devido à limitação da capacidade perceptiva dos sensores de proximidade utilizados[ROY 99]. Em regiões abertas, sensores com alcance finito, como laser, não conseguem identificar nenhum ponto de referência. Câmeras podem falhar quando informações estruturais visuais não estão presentes, exemplo paredes brancas. A seguir relacionamos alguns trabalhos relevantes e recentes na literatura.

Roy [ROY 99] propõe uma técnica chamada *navegação costeira* que sugere uma navegação próxima às regiões que contém alto conteúdo informativo. Ele se baseia na navegação costeira realizada por barcos quando estes não possuem GPS. Esta técnica utiliza *localização markoviana* [THR 98] sobre uma representação baseada em grades. Ela guia o robô em direção àquelas regiões de maior conteúdo informativo, ou seja, regiões com maiores probabilidades de estarem ocupadas.

Olson [OLS 97, OLS 98] propõe uma técnica chamada *iconic matching* de auto-localização para o robô Rocky 7 utilizando visão estéreo. O mapa representa um ambiente natural e é modelado através de grades de ocupação tridimensional, onde o plano xy define a posição do robô, e o eixo z define a altura da superfície na posição(x,y) e a situação da célula ocupada ou não. A localização do robô é realizada utilizando a distância *Hausdorff* do mapa local e o mapa global prévio calculado.

Golfarelli[GOL 2001] propõe uma técnica chamada correção elástica para corrigir o erro posicional gerado durante a exploração de um ambiente. Esta técnica assume que o robô é capaz de identificar *landmarks* e que o conhecimento adquirido é representado em um mapa topológico. A correção elástica é baseada na analogia entre o modelo do ambiente e uma estrutura mecânica, onde cada rota é uma barra elástica e cada *landmark* um nó. Os erros são corrigidos como um resultado das deformações induzidas a partir de forças surgindo dentro da estrutura quando medidas inconsistentes são tomadas. Os parâmetros de elasticidade que caracterizam a estrutura são usados para modelar a incerteza na odometria.

A localização markoviana é também utilizada no problema de *localização global*, também chamado de o *problema do robô raptado*. Este problema foi proposto por Engelson [ENG 94] e consiste em determinar a posição do robô em um frame de referência global com total incerteza sobre sua posição.

Além destes podemos destacar [LU 97], [SHA 97], [BAI 2001], [BUG 99].

2.3.3 Mapeamento e Localização Concorrentes

A maioria das aplicações em robótica exige uma representação interna, na forma de um mapa, da área de atuação do robô a fim de que ele possa realizar suas tarefas eficientemente e com sucesso. É necessário que este mapa seja confiável

e representativo para que possa ser usado em planejamentos de caminhos seguros entre diferentes posições no ambiente. Para algumas aplicações, um mapa detalhado já é fornecido ao robô previamente, entretanto, em algumas não. Neste último caso, um robô que possua capacidade exploratória para realizar a aquisição de mapas de ambientes é essencial.

O processo de mapeamento de ambientes (*map-building*) é extremamente importante para todas as aplicações onde o ambiente é desconhecido. Ele consiste em determinar a localização das entidades de interesse, tais como: landmarks, obstáculos e objetos em um frame de referência global [FOX 99].

Este problema está estritamente relacionado ao problema de auto-localização: por um lado, o robô necessita saber sua posição para construir o mapa do ambiente, por outro o robô necessita deste mapa para se auto-localizar. Ambos os problemas levam a outro problema chamado *mapeamento e localização simultâneos* (**MLS**). É importante observar que alguns autores chamam este problema de *mapeamento e localização concorrentes* (**MLC**)[FED 99].

O mapeamento e auto-localização concorrentes eliminam a necessidade de uma infraestrutura artificial para a localização do robô e de conhecimento *a priori* topológico ou geométrico sobre o ambiente, além de reduzir ou eliminar os erros inseridos pelo *dead-reckoning*.

Um robô capaz de realizar o mapeamento de um ambiente desconhecido e se auto-localizar a partir de uma versão parcial do mapa é capaz de construir um mapa representativo e confiável de qualquer ambiente somente a partir de observações relativas deste ambiente.

A solução deste problema pode ser usada em diversos domínios. Entre eles podemos incluir: ambientes sub-aquáticos, exploração planetária, mineração, aplicações militares, construção, entre outros. Os ambientes sub-aquáticos são especialmente desafiadores para robótica. O principal problema é a sua estrutura. Neste tipo de ambiente é comum ocorrem oclusões de características devido a impossibilidade dos sensores de identificarem o ambiente completamente, além de que, as características extraídas podem sofrer anomalias na associação dos dados [MAJ 2000]. Neste caso, o desenvolvimento de robôs submarinos completamente autônomos é de fundamental importância a fim de que eles sejam capazes de navegar com limitada capacidade sensorial em um ambiente desconhecido e não estruturado como é um domínio sub-aquático [MAJ 2000].

De acordo com Dissanayake [DIS 2001], os trabalhos relacionados a MLS adotam uma das 3 principais filosofias:

1) a mais popular é a utilização do filtro de Kalman. Sua popularidade deriva do fato dele fornecer tanto uma solução recursiva ao problema de navegação como um mecanismo para calcular a incerteza sobre a localização dos *landmarks* e do veículo;

2) a segunda filosofia é a utilização de conhecimento qualitativo dos locais relativos dos *landmarks* e do veículo para construir o mapa para guiar o movimento do veículo [KUI 91];

3) A terceira utiliza grades de ocupação probabilísticas juntamente com um método de relaxação para a navegação e o MLS[THR 98, YAM 98].

O estudo de soluções ao problema MLS iniciou com o trabalho de Smith e Durrant-whyte [LEO 92], o qual estabeleceu uma base estatística para descrever o relacionamento entre *landmarks* e as incertezas geométricas manipuladas. O elemento chave deste trabalho foi mostrar que existe um alto grau de correlação entre as estimativas da localização dos *landmarks* e que essas correlações crescem à medida que novas observações são obtidas. Isto ocorre devido ao erro comum existente na estimativa da posição do veículo [SMI 87].

Leonard e Durrant-Whyte[LEO 92] propuseram um MLS utilizando uma representação baseada em mapa probabilístico para armazenar todas as características geométricas identificadas no ambiente juntamente com uma matriz de covariância associada. Dissanayake[DIS 2001], prova que se determinados requisitos forem obedecidos a consistência do filtro é mantida e o mapa converge monotonicamente a um mapa relativo com incerteza zero. Entretanto isto se torna intratável computacionalmente, pois a utilização completa da matriz de covariância em cada passo acarreta sérios problemas computacionais e a sua omissão resulta em inconsistências e soluções divergentes do mapa.

Em missões de longa duração, o número de landmarks aumenta e os recursos computacionais não são suficientes para manter a atualização do mapa em tempo real. Para resolver isto, Guivant[GUI 2001] propõe uma implementação de um algoritmo MLS que utiliza uma forma especial de matrizes e um novo filtro compressor que reduz significantemente os requisitos computacionais sem inserir qualquer penalidade na precisão dos resultados. Este algoritmo é utilizado no armazenamento e na manutenção de todas as informações obtidas em uma área local. A transferência da informação local para o mapa global é feita quando esta informação se tornar estável e confiável e com custo igual ao do MLS completo com a diferença de ser feito em somente uma iteração. A proposta de Guivant prevê que o usuário pode determinar o número máximo de *landmarks* de acordo com os recursos computacionais disponíveis, e o sistema irá selecionar aqueles que possuem maior poder informativo.

Fox [FOX 99] propõe uma abordagem estatística para estimativas em espaços de alta dimensionalidade baseada no algoritmo EM (*Expectation Maximization*), o qual no contexto das cadeias escondidas de Markov é chamado de algoritmo *alfa-beta* ou *Baum-Welch*. O algoritmo é composto basicamente de dois passos alternados: *passo de localização* e *passo de mapeamento*. O passo de localização é responsável por calcular a posição e orientação real (x, y, θ) do robô baseado no mapa previamente calculado e no conjunto de observações e ações realizadas. O passo de mapeamento é responsável por produzir o mapa mais provável baseado nas estimativas da posição calculada anteriormente. Esta estratégia é utilizada na geração de mapas topológicos [SHA 97] e baseados em grades [FOX 99] utilizando sensores do tipo sonar.

Além destes trabalhos, podemos destacar [MAJ 2000, CHO 2001, CSO 97, SCH 99].

2.4 Trabalhos Relacionados

Como já mencionamos, grande parte dos trabalhos relacionados à exploração e aquisição de modelos de ambientes está restrita à simulação, onde são assumidas condições ideais [RAO 93]. Esta seção apresenta alguns trabalhos relevantes encontrados na literatura.

Thrun [THR 98] propõe uma estratégia baseada em programação dinâmica, chamada *iteração valorada* aplicada sobre uma representação baseada em grades. As células do mapa armazenam um valor que representa o custo navegacional do robô e são atualizadas através de uma função baseada em probabilidade condicional obtida com o uso dos sensores sonar. A iteração valorada atua sobre este custo direcionando o robô para áreas de custo mínimo, as quais correspondem às células ainda não exploradas.

Romero [ROM 2001] cita alguns problemas relacionados a abordagem utilizada por Thrun:

- devido ao movimento do robô ser, em algumas ocasiões, próximo dos obstáculos, o risco de colisões tende a aumentar;
- o robô pode ficar perdido quando inserido em ambientes esparsos;
- o robô muda de direção freqüentemente. Isto tende a aumentar os erros odométricos.

É importante destacar que Thrun, de acordo com Romero [ROM 2001], trata alguns destes problemas empiricamente, entretanto, ele não comenta como é feito este tratamento. Para resolver os problemas supracitados, Romero [ROM 2001a, ROM 2000] propõe a modificação da função de custo para dotar o robô do comportamento gerado pela estratégia local *wall following* a fim de que ele atue em conjunto com o comportamento produzido pela iteração valorada. Neste caso, o robô integra as leituras oriundas dos sensores do sonar e laser, e atualiza o mapa. Em seguida, o robô visita as regiões de baixo custo, direcionado pela iteração valorada, e quando as atinge as explora utilizando a estratégia *seguir paredes*.

Após o processo exploratório, um diagrama similar ao de Voronoi é extraído e usado como base para a iteração valorada gerando um eficiente algoritmo de navegação.

Schultz [SCH 99] descreve como integrar exploração, localização, navegação e planejamento através do uso de uma representação baseada em grades chamada *evidence grid* e do uso de sensores laser e sonar. Esta estratégia é chamada exploração *baseada em fronteiras* e utiliza um processo semelhante à detecção de bordas e extração de regiões para identificar as *fronteiras*, as quais correspondem ao limite entre o espaço conhecido e desconhecido do ambiente até o momento visitado. Durante a exploração, o robô automaticamente se autolocaliza utilizando um mapa que corresponde a integração de diversos mapas locais. Em seguida, o mapa resultante é utilizado para corrigir a posição do robô relativa ao mapa global. Esta correção consiste em diversas rotações e translações ao redor da posição estimada corrente do robô para identificar aquela posição que melhor case o mapa integrado com o global.

As fronteiras são visitadas utilizando o algoritmo *Trulla* [MUR 96], semelhante aos planejadores baseados em *wavefront*. Trulla atua sobre um mapa de longa duração, onde obstáculos não modelados ou transientes geram erros levando o robô a possíveis colisões. Para tratar este problema é usado em conjunto um método reativo chamado tipo VFF (*Virtual Force Field*)[BOR 91a].

Dudek [DUD 93] descreve uma estratégia para exploração e extração de mapas topológicos em um ambiente ideal. Inicialmente, é assumido que o ambiente pode ser representado por um mapa topológico. Esta afirmativa diz respeito à esparsidade do

ambiente [2]. O processo exploratório cria incrementalmente uma árvore de exploração que permite facilmente identificar as áreas que já foram visitadas e as áreas que ainda são desconhecidas do ambiente. Nesta árvore, cada vértice possui uma assinatura única baseada em seus vértices vizinhos que serve como uma identificação daquele local correspondente no ambiente. Esta assinatura elimina os problemas associados ao crescimento ilimitado da árvore no caso de retorno a um vértice já visitado. A visita ao ambiente é realizada através de uma estratégia baseada em busca em largura[VEL 84].

Mataric [MAT 90] propõe uma estratégia baseada na estratégia *seguir paredes* para a extração de mapas topológicos de ambientes *internos*, onde os nós do mapa representam *landmarks* detectados durante o processo de seguir as paredes. Após a identificação de um nó, ele é inserido no mapa juntamente com algumas informações, tais como: parede esquerda, parede direita ou corredor; informações sobre a distância relativa aos outros nós que são utilizadas no processo de localização do robô. Em conjunto, um protocolo é utilizado para garantir que *landmarks* iguais não se tornem nós distintos no mapa.

Choset [CHO 2000a] propõe uma estratégia para a construção incremental do diagrama de Voronoi generalizado hierárquico. A principal característica desta estratégia é ser completo e permitir a exploração e mapeamento de ambientes multidimensionais. Inicialmente, o robô adquire informações sensoriais para detectar parte de uma aresta do diagrama para guiar sua exploração. Ele se move ao longo dela a fim de detectar sua continuidade e vértices do diagrama que darão origem a novas arestas. Estas arestas são visitadas sucessivamente utilizando um algoritmo de busca em profundidade até que o diagrama seja completamente construído.

Um grande inconveniente é gerado pelo *dead-reckoning*. É difícil decidir se um vértice corrente identificado é um vértice novo ou um antigo sendo revisitado. Para resolver este problema, Choset utiliza a vizinhança do vértice juntamente com um conjunto de vértices candidatos.

Geralmente, os trabalhos relacionados à exploração de ambientes assumem que o ambiente será explorado por um único robô. É possível aumentar a eficiência do processo utilizando um time de robôs, que trocam mensagens entre si, para explorar simultaneamente diferentes partes do ambiente. Além disso, é possível mesmo em ambientes com pouca informação topológica, obter um algoritmo poderoso de auto-localização e identificação de objetivos. Abaixo destacamos alguns trabalhos relacionados à exploração utilizando diversos robôs.

A estratégia utilizada por Amat [AMA 95] é fazer com que seus robôs se desloquem aleatoriamente pelo ambiente e extraiam informações sobre ele. Após um ciclo de tempo, estas informações são fornecidas a um computador central responsável por sua integração em uma representação geométrica. Além disso, os robôs agem cooperativamente trocando mensagens, quando se encontram, sobre as regiões que exploraram a fim de minimizar as perdas de informação de um robô caso este fique perdido no ambiente. É importante destacar que a exploração possui um tempo fixo de realização, o que não garante a exploração completa do ambiente.

Por outro lado, Cohen [COH 96] propõe uma estratégia utilizando um time de robôs constituído de um robô navegador e diversos cartógrafos. Durante o processo

[2]De acordo com Choset [CHO 2000a], estratégias de exploração baseadas em mapas topológicos não funcionam corretamente quando o robô está inserido em um ambiente esparso devido à ausência de informação topológica.

de mapeamento este time constrói um modelo do ambiente utilizando exploração aleatória. Em seguida, este mapa é utilizado pelo navegador para planejar caminhos de qualquer ponto do ambiente para uma posição objetivo. Uma consideração importante é que nenhum robô possui informação sobre sua posição absoluta no ambiente. No caso da descoberta de objetivos, quando um robô o encontra durante sua navegação, ele envia uma mensagem, que corresponde a um valor inteiro, aos robôs em seu campo de visão que a incrementam e a repassam sucessivamente até atingir o navegador. Quando o navegador recebe a mensagem ele planeja um caminho seguindo o gradiente descendente da mensagem até atingir o objetivo.

Um importante problema relacionado à abordagem multi-robôs corresponde ao chamado *rendezvous*, ou *pontos de encontro*. Este problema consiste em identificar regiões distinguíveis do ambiente, de acordo com a capacidade perceptiva dos robôs, que corresponderão a pontos de encontro entre robôs para troca de informações sobre o ambiente.

Roy [ROY 2001, ROY 97] propõe estratégias determinísticas e probabilísticas para realizar o *rendezvous* a fim de vencer limitações práticas relacionadas a comunicação entre robôs. A estratégia mais simples é navegar durante um tempo finito pelo ambiente e localizar bons pontos de encontro. Ao final deste tempo, se direcionar ao melhor ponto de encontro e esperar a chegada dos outros robos para realizar o troca de informações sobre o ambiente. A extração destes pontos, neste caso, é baseada na esparsidade do ambiente indicada pelos sensores do robô e o critério de seleção é baseada em uma função probabilística. Os critérios de seleção comumente utilizados podem ser a ordem de detecção dos pontos, uma seleção aleatória ou uma seleção baseada em uma função probabilística.

Outro trabalho interessante é o trabalho de Rekleitis [REK 97]. Rekleitis propõe um algoritmo para exploração cooperativa entre dois robôs móveis objetivando otimalidade e redução da incerteza posicional durante o processo de mapeamento do ambiente. Ele considera que o mundo é um polígono e os robôs são pontuais com movimento *omnidirecional*. A exploração é realizada concorrentemente pelos robôs, onde a cada instante apenas um robô se movimenta, e o outro fica parado o observando. Isto permite ao robô em movimento utilizar o robô parado como um *landmark* inteligente e corrigir sua posição com maior precisão que o simples *dead-reckoning*.

Quando o ambiente é esparso, a estratégia de exploração é dividida em exploração local e global. O ambiente é dividido em diversas regiões no formato de um trapézio que são exploradas pelo método local através de um caminho seguido pelos robôs no formato de diamante e o chaveamento entre estas regiões é feito através da exploração global que corresponde a um algoritmo de uma busca em profundidade sobre elas. Quando o ambiente não é esparso, o movimento dos robôs é baseado em um algoritmo de triangularização de uma figura poligonal. Rekleitis também estende sua estratégia para a utilização de mais de um robô no processo exploratório.

3 Princípios da Navegação Exploratória baseada em Problemas de Valores de Contorno

Como foi visto anteriormente, os campos potenciais podem ser classificados tanto como métodos globais quanto como métodos locais. Nosso trabalho foi motivado pelo método global baseado em campos potenciais desenvolvido por Connolly e Grupen[CON 93] para ambientes onde objetos e paredes têm suas localizações conhecidas com precisão, assim como a posição a ser atingida.

A grande novidade do método de Connolly e Grupen em relação aos métodos de campo potencial é o uso de funções harmônicas para o campo potencial. Este campo é calculado por uma solução de um problema de valor de contorno (PVC) bem definido envolvendo a equação de Laplace.

$$\nabla^2 p(\mathbf{r}) \;=\; \frac{\partial^2 p(\mathbf{r})}{\partial x^2} + \frac{\partial^2 p(\mathbf{r})}{\partial y^2} \;=\; 0 \tag{3.1}$$

e, em geral, condições de contorno de *Dirichlet* [1] que indicam características globais do ambiente como obstáculos obstruindo movimento (*alto potencial*) e posições objetivo (*baixo potencial*), descritas adiante. Os caminhos navegacionais correspondem à *linhas de força* extraídas a partir do gradiente descendente do potencial, que direcionam o robô até a posição *objetivo*, concomitantemente o fazendo desviar dos obstáculos.

As soluções da equação de Laplace, chamadas funções harmônicas, não apresentam mínimos locais [COU 62], assim o objetivo sempre é alcançado seguindo uma linha de força, se um caminho até ele existir. Neste sentido o método é formalmente completo[2].

A extensão do método de funções harmônicas ao problema de exploração inicia considerando que durante a exploração, a área onde o PVC deve ser calculado corresponde unicamente à região do ambiente já explorada. Esta região é limitada por dois tipos de condições de contorno: superfícies de obstáculos e fronteiras de exploração. As fronteiras são limites do espaço livre onde nenhuma leitura sensorial foi adquirida ou considerada além daquele ponto (ver Figura 3.1). Esta informação está intrínseca na memória interna do robô e é dependente do histórico da exploração, da capacidade do robô de se autolocalizar e de seu modelo de sensores. Neste capítulo mostramos que o método é coerente e por ser uma extensão e generalização do método das funções harmônicas situa-se em sólidas bases matemáticas.

Para uma dada fronteira, a exploração é obtida resolvendo um PVC para condições de contorno que permitem uma navegação segura em sua direção. Dessa forma, nosso método de exploração consiste em uma navegação baseada em PVC em um contorno dinâmico. Neste método, a maneira com que a exploração é realizada está intrinsecamente dependente da configuração do contorno. As características ambientais influenciam diretamente no comportamento do robô. Portanto, estratégias diferentes para exploração podem ser criadas selecionando diferentes características no contorno.

[1] Condições de *Dirichlet* estabelecem o valor da função em pontos do contorno. Enquanto que as condições de *Neuman* estabelecem o valor das derivadas da função em pontos do contorno.

[2] Um método de planejamento é completo quando ele garante encontrar um caminho livre se existir ou, caso contrário, emitir um sinal de falha.

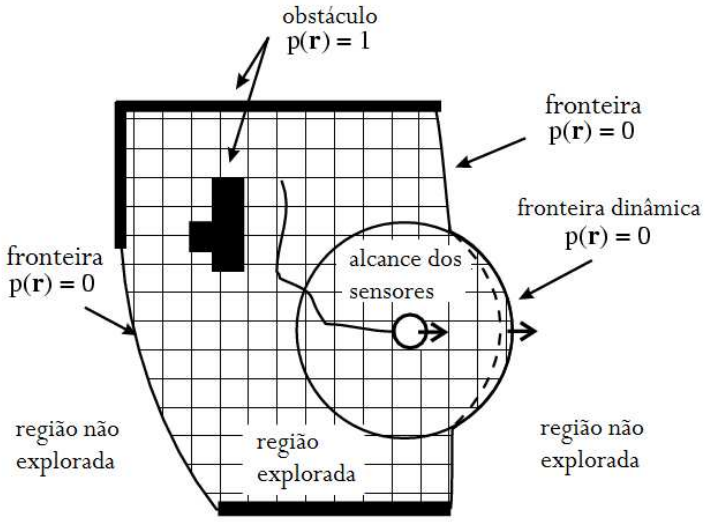


FIGURA 3.1 – Região do ambiente onde o PVC é calculado.

3.1 Campos potenciais harmônicos para o planejamento de caminhos

Para implementar o método para planejamento de caminhos baseado em campos potenciais harmônicos é necessário o completo conhecimento da localização dos obstáculos e da posição a ser alcançada. Em seguida o contorno do PVC no espaço de graus de liberdade independentes (*c-space*) é definido com o valor 1 para o potencial sobre a superfície dos obstáculos e 0 para o potencial sobre a superfície, ou ponto, que define a posição objetivo. Finalmente, a equação de Laplace é calculada, através de um método de relaxação similar à *iteração valorada* [SUT 98] que interpola o valor do potencial entre os obstáculos e o alvo, e a partir de um ponto inicial $\mathbf{r}(0) = \mathbf{R}_0$. Após a convergência, o planejador retorna como caminho a curva definida pelo gradiente descendente iniciando em R_0 e finalizando na posição objetivo.

Um ponto interno $\mathbf{r}^*$ *é um mínimo (máximo) se e somente se*

$$\nabla p(\mathbf{r}^*) = 0 \quad \text{e} \quad \frac{\partial^2 p}{\partial x_i^2}|_{\mathbf{r}^*} > (<) 0 \quad \forall i$$

onde x_i são coordenadas espaciais. A equação de Laplace 3.1 claramente afirma que mínimos (máximos) locais não são possíveis, pois para qualquer ponto interno $\mathbf{r}$, a soma das curvaturas $\frac{\partial^2 p}{\partial x_i^2}$ é igual a zero. Outras propriedades dos potenciais podem ser obtidas com a ajuda da Física. Considere que o robô segue o gradiente descendente com uma velocidade constante. Podemos escrever que

$$\frac{d\mathbf{r}}{dt} = -c\frac{\nabla p}{|\nabla p|}$$

onde c é uma constante que representa a velocidade. O ambiente é finito e delimitado por paredes ou obstáculos. O movimento do robô integra $\frac{d\mathbf{r}}{dt}$ gerando um caminho

suave em direção ao objetivo. Neste caso, o robô segue caminhos que são análogos às linhas de força na Teoria Eletromagnética

$$\mathbf{E} = -\nabla p.$$

A analogia feita entre os caminhos do robô com linhas elétricas de força é conveniente e válida, uma vez que ambas são definidas como gradiente descendente do potencial. Portanto, esses caminhos possuem propriedades análogas [FEY 72]

i) elas correspondem a curvas abertas, ou seja, elas não apresentam ciclos. Isto é claramente visto pois o campo vetorial é derivado a partir do potencial (*campo conservativo*) e por isso ciclos não existem $\nabla \times \mathbf{E} = 0$.

ii) elas conectam pontos na superfície dos obstáculos (*alto potencial*) aos pontos na fronteira (*baixo potencial*). Isto é conseqüência do fato de que mínimos ou máximos locais não são permitidos pela equação que define a equação diferencial parcial (EDP).

iii) duas linhas de força nunca se cruzam. Isto também é conseqüência do fato que o campo de força é derivado a partir de um potencial. O potencial é uma função analítica e tem gradiente univocamente definido em todos os seus pontos.

Além disso, as funções harmônicas podem ter diversas interpretações. Elas podem ser o resultado do princípio de energia mínima, ou mínima ação. Elas podem ter um significado probabilístico como a probabilidade de colisão de um *caminhante aleatório*, partindo de uma dada célula, com um obstáculo antes de atingir o objetivo desejado. Além disso, elas podem ser derivadas de processos de aprendizado como *diferenças temporais* [SUT 88, CON 94] ou o método Monte Carlo para o aprendizado por reforço [SUT 98].

3.2 Além das Funções Harmônicas

A equação de Laplace é um caso particular de um conjunto mais geral de equações que produzem campos potenciais sem mínimos locais.

Existe uma família de funções escalares geradas por um PVC que não apresenta mínimos locais.

É possível mostrar que a solução da família de equações definidas por

$$\nabla^2 p + F(\nabla p) = 0 \tag{3.2}$$

tal que F($\mathbf{v}$) é qualquer função contínua, com

$$F(\mathbf{0}) = 0$$

não tem mínimos (máximos) locais.

Para ver isto considere que $\mathbf{r}^*$ é um ponto onde o gradiente é nulo. A partir da Equação (3.2) temos que neste ponto também é verdade que

$$\nabla^2 p|_{\mathbf{r}^*} = 0$$

e portanto estão excluídos mínimos ou máximos locais no ponto $\mathbf{r}^*$.

A fim de ilustrarmos como o novo termo na equação afeta os caminhos resultantes, consideramos o caso mais simples: a função linear

$$F(\nabla p) \;=\; \epsilon\, \nabla p \cdot \mathbf{v} \tag{3.3}$$

onde ϵ é um escalar e $\mathbf{v}$ um vetor constante. Substituindo (3.3) em (3.2) temos o problema de Sturm-Liouville [KRE 66]

$$\nabla^2 p + \epsilon\, \nabla p \cdot \mathbf{v} = 0 \tag{3.4}$$

que, dependendo das condições de contorno, pode ser resolvido analiticamente. Nós resolvemos (3.4) dentro do intervalo $x \in [0, L]$ e $y \in [0, M]$ com $\mathbf{v} = (1, 1)$, $\epsilon = 1$ e sujeito a condições de contorno específicas dadas por

$$\begin{aligned} p(x,0) &= 1 \\ p(x,M) &= 0 \\ p(0,y) &= 0 \\ p(L,y) &= 0. \end{aligned} \tag{3.5}$$

Isto representa uma região quadrada com uma parede (obstáculo) em $(y = 0)$ e três fronteiras de exploração (limites do espaço livre) em $(y = M)$, à esquerda $(x = 0)$ e à direita $(x = L)$. A Figura 3.2 ilustra o gradiente descendente da solução da equação (3.4) (flechas sólidas) e o gradiente descendente da solução da equação de Laplace (3.1) (flechas tracejadas) sujeita a (3.5).

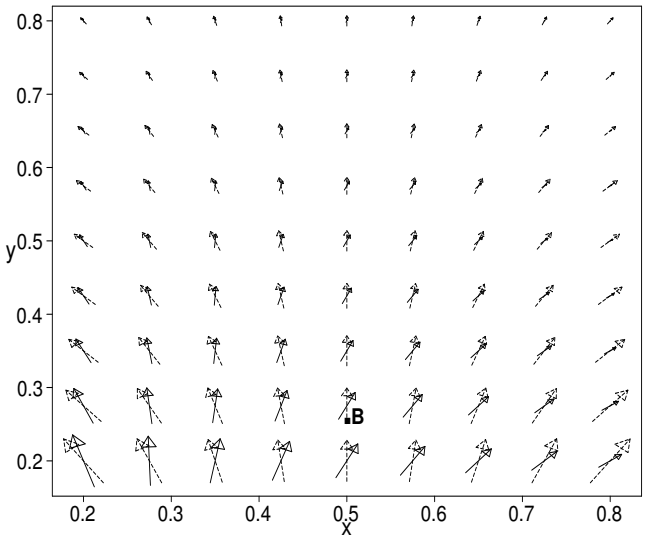


FIGURA 3.2 – Gradiente descendente da solução da equação (3.4) (flechas sólidas) e a equação de Laplace (flechas tracejadas) sujeita a (3.5) na região quadrada $L \times M$ onde $L = M = 1$.

A contribuição de (3.3) pode ser compreendida supondo que o robô está situado em um ponto $\mathbf{B}$ (ver Figura 3.2). Neste ponto o gradiente descendente da solução de Laplace aponta em direção ao topo do ambiente, assim ele guia o robô em um caminho linear ao topo. Nos vizinhos à esquerda e à direita de $\mathbf{B}$, o gradiente aponta ligeiramente para o lado esquerdo e direito do ambiente respectivamente. Este

comportamento simétrico do gradiente descendente da solução de Laplace relativo ao eixo $x = 0.5$ é completamente determinado por seu contorno. Agora, considere o gradiente descendente da solução da equação (3.4), nós observamos que em **B** o gradiente aponta ao lado direito do ambiente. Isto mostra que quando o termo é adicionado à equação de Laplace a simetria ao redor do eixo $x = 0.5$ é quebrada. Neste caso, a simetria da solução não é determinada exclusivamente pelo contorno mas também pela direção do vetor **v**.

3.3 Contornos Dinâmicos e Exploração

O método de navegação usando PVC com contornos dinâmicos consiste em um método de campo potencial onde o campo indica o caminho em direção às regiões não exploradas do espaço. Isto é feito considerando as fronteiras com regiões não exploradas como objetivos temporários, isto é, regiões cujo potencial é igual a 0. De forma semelhante ao método desenvolvido por Connolly, seguir o gradiente descendente sobre o potencial faz o robô evitar obstáculos e se aproximar destes sub-objetivos. As propriedades das linhas de força garantem que o robô sempre se movimentará em direção a uma fronteira com regiões não exploradas se ela existir. Uma vez que o robô atingiu seu objetivo, ele ultrapassa a fronteira, a região explorada cresce e novas fronteiras surgem.

O próximo ponto é perguntar como a dinâmica básica muda quando permitimos que as condições de contorno variem com a exploração das fronteiras. Em particular, estamos interessados em saber se mínimos locais ou ciclos no campo de força surgem como resultado dessa dinâmica. Neste contexto a seguinte afirmação é verdadeira:

Navegação em direção às regiões não exploradas garante uma completa exploração de um ambiente finito.

Nós provamos isto por contradição. Inicialmente, é necessário primeiro observar que o contorno sempre cresce. Em seguida, consideramos que o robô entrou em um atrator periódico, no qual ele segue uma trajetória fechada γ. Após, ter completado o primeiro *ciclo* o robô finalizará a exploração pois no *ciclo* seguinte ele apenas estará repetindo um caminho já seguido. Assim, nestas circunstâncias o limite é estático. Como discutimos na Seção 3.1, *ciclos* no contorno estático não aparecem, isto contradiz a afirmativa acima e por isso *ciclos* não são possíveis.

3.4 Exploração dirigida por características do contorno

Foi observado para roedores que exploram ambientes a procura de comida que as características ambientais são fontes importantes de informação espacial [O'KE 96]. Estas características que representam obstáculos móveis ou fixos, paredes e objetivos são determinantes geométricos do contorno do ambiente. Embora isto não pareça óbvio à primeira vista são bons pontos para orientar a exploração de ambientes desconhecidos.

Selecionando diferentes características no contorno podemos criar estratégias diferentes para a exploração completa do ambiente. Essas estratégias estão associadas com o estado interno do robô que determina como tem que ser o comportamento do robô em uma dada configuração de contorno usada no cálculo do potencial

definido por (3.2).

Dessa forma, os contornos podem ser tratados como um módulo separado que possibilita a redefinição de suas propriedades durante o processo de exploração. Isto dá origem a uma navegação exploratória onde as características ambientais são dinamicamente redefinidas com a exploração. Conseqüentemente, o comportamento do robô também muda, uma vez que ele se comporta de acordo com o contorno que ele encontra.

Por exemplo, suponha que o robô está em um ambiente como é representado na Figura 3.3. Regiões dentro do retângulo pontilhado são bordas de paredes. Quando o robô persegue estas regiões primeiro ele automaticamente assume um comportamento exploratório *seguir paredes*. Mas se o robô, ao invés de seguir as paredes, procurar preencher as regiões côncavas, como círculos pontilhados, nós dizemos que isto é um comportamento exploratório de *preenchimento de espaço*.

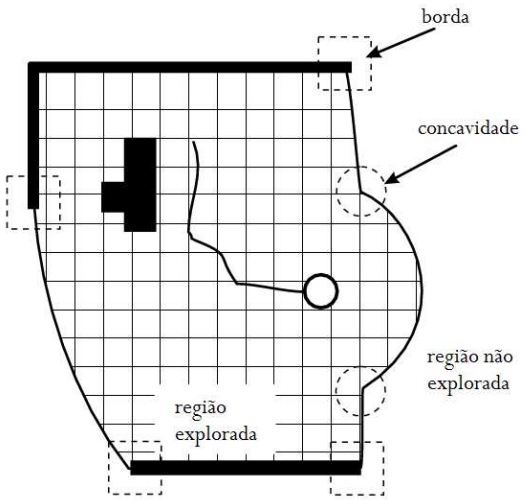


FIGURA 3.3 – Regiões do ambiente onde o robô pode visitar dependendo de seu comportamento. Procurar regiões dentro dos quadrados pontilhados representa um comportamento exploratório *seguir paredes* e em regiões dentro dos círculos pontilhados representa um comportamento de *preenchimento de espaço*.

Como afirmamos acima, o comportamento do robô é dado pelas condições de contorno, onde o contorno está situado no potencial definido por alguma função ao longo dele, isto é,

$$p(\mathbf{r}) = f(\mathbf{r}) \quad \text{para} \quad \mathbf{r} \in \partial\Gamma \tag{3.6}$$

onde $\partial\Gamma$ é o limite entre a região explorada e a não explorada do ambiente Γ. O comportamento de exploração *seguir paredes* é realizado apenas atribuindo baixo potencial ($f(\mathbf{r}) = 0$) na borda das paredes e em sua vizinhança. Caso contrário, o comportamento exploratório de preenchimento de regiões é realizado atribuindo baixo potencial nas regiões côncavas não exploradas.

3.5 Mudança de Domínio

Nas seções anteriores mostramos que a solução de uma família de equações definidas em (3.2) não tem mínimos (máximos) locais. Mas o que acontece quando discretizamos o sistema? Estes mínimos (máximos) irão aparecer se mudarmos o domínio de contínuo para discreto? No caso específico da equação (3.4) a resposta é que mínimos (máximos) locais não aparecem contanto que o potencial p no ponto (i, j) possa ser escrito como uma média ponderada de p sobre seus pontos vizinhos

$$p_{ij} = \sum_{\langle i',j' \rangle} w_{i'j'} p_{i'j'}$$

onde $\langle i', j' \rangle$ representa a soma dos locais vizinhos de (i, j) e $w_{i'j'}$ são pesos positivos. Como p_{ij} é um valor médio sobre os locais vizinhos de (i, j) (mais detalhes ver Seção 4.2.6 ou [CON 94, PRE 2002]) podemos escrever

$$p_{min} \leq p_{ij} \leq p_{max}$$

onde p_{min} e p_{max} são os valores mínimos e máximos de p na vizinhança de (i, j). Isto afirma que existem valores próximos de p menores ou maiores que o valor em (i, j). Em outras palavras, isto mostra que a discretização não introduz mínimos (máximos) locais no sistema.

4 Agente Exploratório Proposto

Do ponto de vista da IA existem duas classes de estratégias para exploração. Elas atuam em estruturas do tipo *grafo* e são classificadas como: buscas sistemáticas e buscas heurísticas. Em robótica, estas categorias podem ser comparadas, respectivamente, às estratégias reativas e baseadas em modelos (ver Seção 2). A busca sistemática ou estratégia reativa implica que nenhum conhecimento prévio sobre o ambiente é utilizado para direcionar o comportamento do robô. Além disso, este comportamento não é afetado pelo resultado da exploração. Por outro lado, a exploração baseada em modelos carrega consigo informações extras que são utilizadas para modificar o comportamento do robô de acordo com a distribuição dos obstáculos e com o progresso da exploração.

Um exemplo de busca sistemática já discutido é a estratégia *seguir paredes* [MAT 90, CRO 85a]. Sua principal limitação é a estrutura do ambiente onde será utilizada, dependendo da esparsidade do ambiente, ela necessita ser mesclada com outras estratégias para que a exploração seja realizada com sucesso. As estratégias baseadas em modelo, por outro lado, usam uma função de custo que carregam consigo um conhecimento extra definido previamente para direcionar o movimento do robô.

Neste capítulo apresentaremos a estrutura de um agente que utiliza uma função de custo representada pelo potencial calculado a partir da solução de problemas de valores de contorno, como apresentado na Seção 3. Este agente é definido e representado pelo robô NOMAD 200 fabricado pela empresa Nomadic (ver Seção 5.1).

4.1 Arquitetura do Agente

Do ponto de vista da IA, o robô é um agente que interage com o ambiente executando ações baseadas nas informações oriundas de seus sensores. Ele pode ser visualizado como a união entre uma estrutura física e um programa [RUS 95]. A estrutura física corresponde a um computador ou a um hardware específico que atua e mede propriedades de um dado ambiente real ou simulado através de seus sensores. Por outro lado, o programa corresponde a uma função que implementa o mapeamento da percepção do robô em ações que ele pode realizar.

A arquitetura de um agente descreve como diferentes sub-funções, ou módulos, são organizados em um programa para produzir o mapeamento final. Existem diversas arquiteturas ilustradas na literatura de IA, por exemplo NASREM [ALB 89], LAAS[ALA 98], Subsumption[BRO 86] , etc. Entretanto, não existe uma teoria de desenvolvimento de arquitetura amplamente aceita que possa ser usada para provar que uma arquitetura é melhor que a outra. A maioria delas tem princípios em comum que fazem qualquer tentativa de classificação parecer bastante confusa [MED 98, PET 97].

Para um agente autônomo, uma organização conveniente de funções é o paradigma *SMPA* (Sense- Model - Plan - Act)[NIL 98]. Ela pode ser considerada uma estrutura organizacional ampla sobre a qual muitas arquiteturas são baseadas.

Nesta estrutura, o agente sente o ambiente e processa a informação oriunda de seus sensores usando o módulo *processamento sensorial*, decodificando-a e

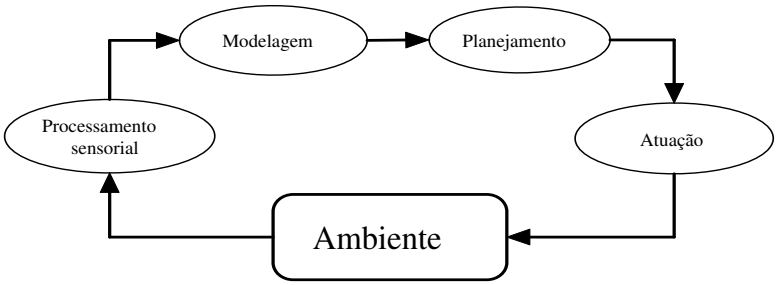


FIGURA 4.1 – Arquitetura de um agente SMPA

armazenando-a internamente em uma representação conveniente. Na seqüência, no módulo *modelagem*, o agente compara o estado do sistema esperado com o estado do sistema corrente, resolve os conflitos ou discrepâncias entre eles, e verifica o progresso em direção ao objetivo. Em seguida, no módulo *planejamento*, ele planeja a seqüência de ações a ser realizada por seus atuadores. No módulo *atuação*, as ações são executadas. Essas ações podem ser ações de controle de baixo nível como alterar a voltagem de seus motores, ou decisões de alto nível como escolher qual o objeto deve direcionar sua atenção.

Uma representação interna comum do mundo é o *grafo de espaço de estados*. Ele representa um conjunto discreto de configurações no mundo, chamados estados, que um agente pode encontrar durante seu movimento pelo ambiente e as ações que realizam a mudança entre essas configurações. Graficamente os estados são representados como nós no grafo e as ações, como arcos que conectam estes nós. Utilizando este formalismo, o módulo de planejamento é simplesmente um algoritmo convencional de busca em grafos.

A principal vantagem em usar estruturas tipo grafo é a redução do problema de planejamento para um problema de busca em grafos. Entretanto, a principal desvantagem é a existência de fortes requisitos sobre o problema, tais como: todas as configurações do mundo devem ser representadas pelo agente, o agente deve ter um modelo preciso de como as suas ações afetam a mudança de uma configuração em outra, e seu sistema perceptivo deve precisamente identificar a configuração atual. Em aplicações do mundo real, a maioria desses requisitos não é satisfeita pelo estado corrente da tecnologia e, em particular, para robôs móveis o espaço de estados é um espaço contínuo.

Para lidar com este problema, nós propomos a arquitetura ilustrada na Figura 4.2, onde o mundo é representado por estados discretos a partir dos quais comandos contínuos podem ser gerados. A representação do mundo é controlada por uma arquitetura similar à arquitetura *blackboard*[PET 97]. Esta arquitetura é composta por uma estrutura de dados chamada *blackboard* e diversos módulos especialistas, chamados *fontes de conhecimento* que atuam independentemente sobre os dados no *blackboard*, os atualizando e checando sua consistência.

Os dados armazenados no *blackboard* são compostos de dois componentes essenciais e complementares:

- as coordenadas do robô em um sistema de referência global;
- a representação baseada em grades do mundo, onde as propriedades do ambi-

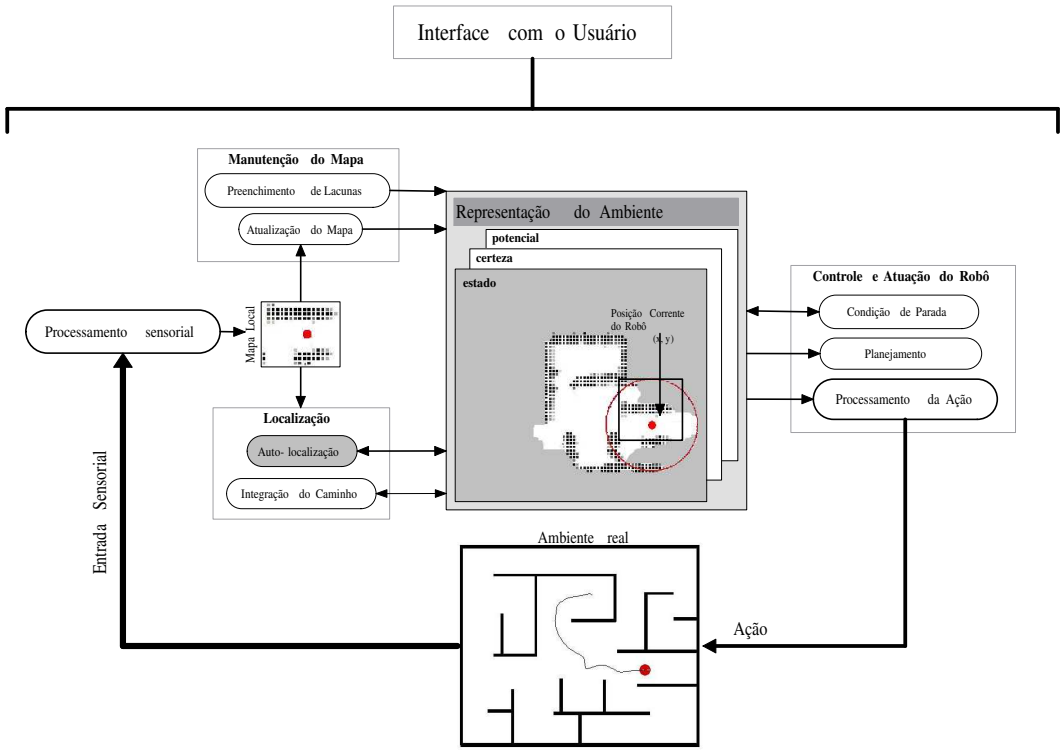


FIGURA 4.2 – Arquitetura de um agente SPAE (Sense-Plan-Act-Explore).

ente são armazenadas.

Os especialistas interagem com os dois componentes supracitados atualizando-os ou extraindo informações que são usadas nas fases subseqüentes de planejamento e ação. Esses especialistas realizam as seguintes funções:

- Para a atualização das coordenadas do robô (localização).
  - integração de caminho, responsável por realizar a localização do robô baseada nas informações fornecidas por seus odômetros;
  - auto-localização (módulo ainda não implementado), responsável por reduzir os erros de odometria gerados pelo módulo *integração de caminho*.
- Para a manutenção do mapa ambiente (*blackboard*):
  - atualização do mapa baseado nas leituras sensoriais;
  - preenchimento de lacunas.
- Para o controle e atuação do robô
  - planejamento, responsável por atualizar o campo potencial (função heurística) utilizando diferentes PDEs;
  - condição de parada, responsável por determinar quando o robô deve parar sua atuação a partir da identificação de que um objetivo foi alcançado (exploração completa de um ambiente, localização de um objeto, etc);

– processamento da ação. Cálculo do gradiente do potencial para determinar a ação que o robô deve realizar no ambiente.

Estes especialistas são ativados a partir do módulo de interface com o usuário. Este módulo é responsável por:

- especificar a tarefa a ser realizada. As tarefas que nosso agente pode realizar são: exploração e aquisição de mapas de ambientes desconhecidos, planejamento de caminhos baseado em mapas e localização de objetos[1] ;
- ajustar os paramêtros do sistema;
- enviar comandos para o robô.

4.2 Ingredientes Básicos

4.2.1 Processamento Sensorial

Em nossa aplicação, as duas grandes fontes de informações do ambiente são:

1. leitura dos odômetros. A informação fornecida pelos odômetros é utilizada para atualizar a posição corrente do robô no espaço bidimensional contínuo;
2. leitura do sonar. A partir da leitura do sonar, um mapa parcial é gerado para ser incorporado ao mapa global do ambiente pelo especialista responsável pela atualização do mapa.

4.2.2 Localização

Neste grupo de operações estão os módulos *integração de caminho* e *auto-localização*. No módulo *integração de caminho* o algoritmo usado é o *dead-reckoning* citado anteriormente na Seção 2.3.2 que consiste em integrar os dados fornecidos pelos odômetros das rodas do robô. Estes dados são os deslocamentos $(\Delta x, \Delta y)$ que são adicionados à posição atual do robô (p=(x, y)) resultando em sua posição corrente. Erros na variação do deslocamento pelo odômetro podem ocorrer por uma série de fatores. Esses erros são incorporados na trajetória fazendo que a posição real e a posição estimada do robô se afastem gradativamente. A forma de corrigir isto é através de um processo de auto-localização.

O módulo de auto-localização é portanto de grande importância para a construção de um mapa confiável do ambiente. Entretanto este módulo ainda não foi implementado, pois estes erros ainda não começaram a afetar de maneira significativa os resultados dos experimentos, devido à baixa velocidade de deslocamento do robô e as pequenas dimensões dos ambientes explorados (ver Seção 5.4.5). Além disso, porque o movimento do robô é calculado a partir de um gradiente contínuo (com será visto na Seção 4.2.8), sua trajetória é automaticamente suavizada. Isto resulta em um trajeto sem mudanças abruptas na orientação do robô, um dos principais fatores que aumenta nos erros de odometria.

[1] A base para o processo exploratório é a mesma para o planejamento de caminhos. Como o planejamento de caminhos já foi estudado anteriormente [CON 93], destacamos aqui apenas a exploração. No caso da localização de um objeto, o príncipio é o mesmo da exploração com exceção que o agente pára seu processamento assim que ele encontra o objeto que procura.

De qualquer forma é importante ressaltar que os erros gerados pelo *dead-reckoning* crescem ilimitadamente à medida que o robô se desloca pelo ambiente e é nitidamente visível em ambientes de grandes dimensões, sendo de grande importância um mecanismo de correção de erros. Além da correção dos erros de localização, é esperado que o módulo de auto-localização também realize a localização global do robô(ver Seção 2.3.2).

4.2.3 Representação do Ambiente

Em nossa implementação, o agente representa o ambiente internamente utilizando uma representação icônica baseada em uma matriz bidimensional $M_{L_x \times L_y}$, da mesma forma que o agente de Nilsson [NIL 98] representa. Esta representação em robótica é denominada mapa baseado em grades (ver Seção 2.3.1). A principal razão para sua utilização é a compatibilidade desta representação com aquela utilizada na solução de problemas de valores de contorno, pois ela fornece uma representação já discretizada, pronta para ser utilizada em conjunto com os métodos de relaxação para o cálculo da função potencial. Além disso, ela fornece uma maneira fácil para fundir diferentes sensores e representar o espaço livre do ambiente, o que corresponde ao cerne do processo exploratório.

Cada célula da grade está centrada nas coordenadas do mundo real $\mathbf{r}= (r_i, r_j)$ e corresponde a uma região quadrada do mundo real que armazena os seguintes atributos:

- **estado** (s_{ij}): indica o estado corrente da célula, o qual pode ser: *não explorado* , *espaço livre* e *ocupado*. O estado *não explorado* indica que a posição correspondente à célula de índice *ij* ainda não foi visitada e seu valor de potencial é igual a 0. O atributo *espaço livre* indica que o potencial da célula pode ser atualizado. O atributo *ocupado* indica que a célula está ocupada por um objeto no mundo real e seu valor de potencial é igual a 1 no caso de um obstáculo e 0 no caso de um objetivo;

- **certeza** (c_{ij}): dá uma medida de probabilidade de encontrar um obstáculo naquela célula;

- **potencial** (p_{ij}): se refere à função heurística, no espaço de estados discreto, que quando convertido ao espaço contínuo indica caminhos de navegação para o agente. Este potencial tem a mesma finalidade daquele apresentado no Capítulo 3;

- **região** (rg_{ij}): armazena um número inteiro e é utilizado apenas para identificar quando o agente deve parar o processo exploratório (ver Seção 4.2.7).

A Figura 4.3 ilustra uma instância particular do estado das células da região marcada na representação do ambiente ilustrada na Figura 4.2. A Figura 4.3 (a), ilustra o valor da propriedade *estado* das células da região em questão, assim como a posição do robô definida por **R**. As células nas cores preta, branca e cinza, representam, respectivamente, os valores *ocupado*, *espaço livre* e *não explorado*. A Figura 4.3 (b) ilustra o valor da propriedade *certeza* de cada célula em níveis de cinza, onde branco corresponde ao valor 0 e o preto ao valor máximo permitido. A Figura 4.3 (c) ilustra o valor da propriedade *potencial* de cada célula em níveis de cinza, onde

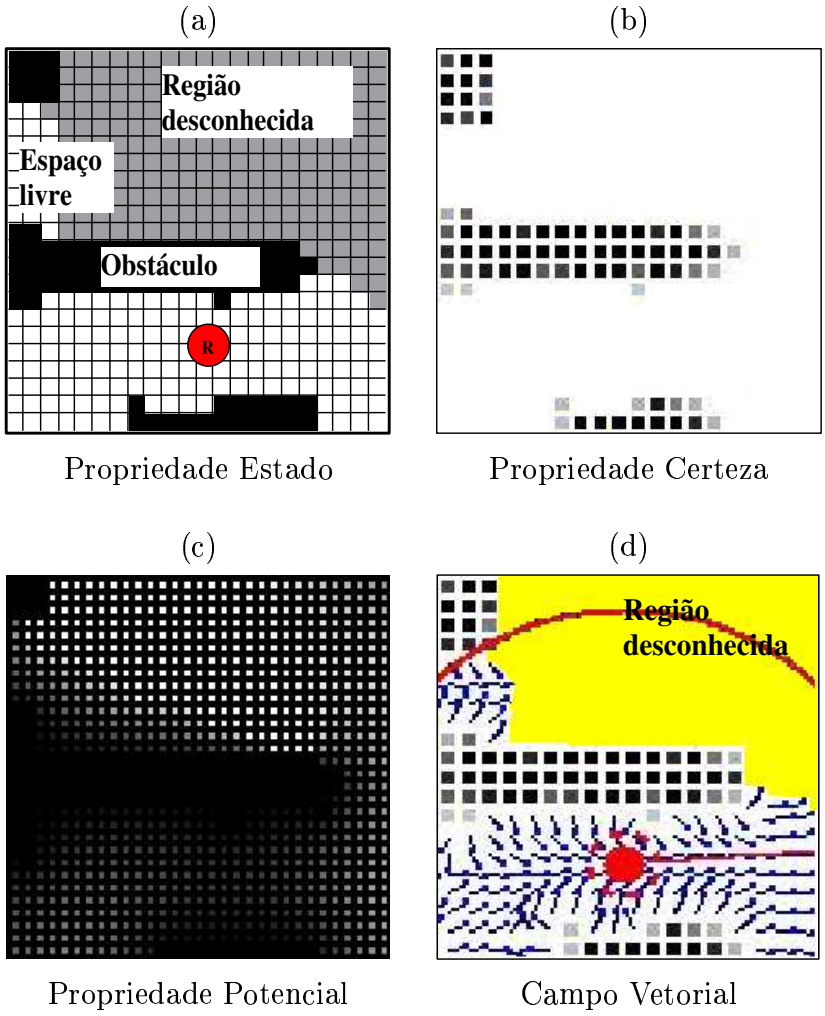


FIGURA 4.3 – Situação dos atributos (estado, certeza, potencial) da região marcada na representação do ambiente ilustrada na Figura 4.2 e o campo vetorial gerado a partir do atributo potencial.

branco e preto correspondem ao valor de potencial igual a 0 e 1, respectivamente. A propriedade *região* não foi ilustrada na Figura 4.3, por não representar uma característica do ambiente e ser utilizada apenas na heurística que determina quando o processo exploratório deve finalizar. A Figura 4.3 (d) mostra o campo vetorial gerado a partir do campo potencial que guia o agente durante sua exploração.

Quando a exploração inicia, os atributos de todas as células são modificados da seguinte maneira: estado recebe o valor *não explorado*, enquanto que certeza, potencial e região recebem os valores $0, 0, 5$[2], respectivamente. Nas seções seguintes apresentaremos o mecanismo utilizado para calcular o valor destas propriedades em cada ciclo do processo exploratório[3].

[2]A atribuição do valor 0 à propriedade certeza de uma célula indica que o sistema não possui qualquer conhecimento sobre a situação da posição real daquela célula no ambiente. Enquanto que a atribuição do valor 0 à propriedade potencial de uma célula indica que ela corresponde a uma célula de atração, a qual pode representar uma região objetivo ou uma região desconhecida. A atribuição do valor 5 à propriedade região é vista com maiores detalhes na Seção 4.2.7.

[3]Um ciclo do processo exploratório corresponde à leitura dos sensores, processamento das medi-

4.2.4 Atualização do Mapa

A atualização do mapa é feita em dois passos. Inicialmente, o mapa local é extraído a partir da leitura dos sensores. Este mapa indica uma região, ou janela, centrada na posição corrente robô, que contém células com seus valores para estado e certeza atualizados. A seguir, as células do mapa local são casadas com as células correspondes do mapa global, e estas últimas atualizadas.

A atualização é feita utilizando um método baseado na freqüência de observações chamado HIMM (*Histogramic In-Motion Mapping*) proposto por Borenstein e Koren [BOR 91]. Ele usa uma função linear que conta o número de observações sensoriais feitas para cada objeto no ambiente. A partir daí, ele marca as células que estão sobre o eixo acústico de cada sensor, pois estas produzem maior retorno dos obstáculos que aquelas células localizadas longe dele, usando a propriedade *certeza*. Se a taxa de observações positivas por unidade de tempo exceder um limiar predefinido, a célula tem sua propriedade estado alterada para *ocupado*, caso contrário, para *espaço livre*. Esses atributos podem variar quando o número de observações cresce.

Como esta taxa está escondida atrás de um limiar, HIMM produz uma versão binária do mapa. É importante destacar que esta função não espalha uma *pdf* (*probability density function*) ao redor dos obstáculos indicando a probabilidade de colisão como a função usada por Thrun[THR 98] e outros. Mas, a associação do mapa binário com a função heurística, descrita adiante, automaticamente gera a *pdf* representada pelo valor de potencial das células do espaço livre ao redor dos obstáculos.

Um caso particular é a função heurística para o cálculo do potencial harmônico. Neste caso, Connolly [CON 94] mostrou que no caso específico onde os obstáculos são representados por um valor de potencial fixo igual a 1 e o objetivo por um potencial igual a 0, o potencial harmônico em uma célula dá a probabilidade de um *caminhante aleatório* partindo de qualquer célula colidir com um obstáculo antes de atingir o alvo desejado. Neste caso, este potencial é chamado probabilidade de colisão (*hitting probability*).

Nos experimentos nós usamos sensores de sonar. Eles são disparados a cada 500ms e podem detectar objetos a uma distância d a partir do robô. Este intervalo é dependente da velocidade do robô. Se o robô se move a uma velocidade alta é importante que os sensores sejam disparados em intervalos curtos, a fim de que ele consiga reagir rapidamente a eventos inesperados. No caso de velocidades baixas, o intervalo do disparo dos sensores deve ser aumentado, para que seja minimizado o efeito colateral produzido pela reflexão especular.

Usando o conhecimento sobre sua posição e orientação, o robô incrementa de 3 o c_{ij} das células na região definida por $[d-\Delta d, d+\Delta d]$ no limite do cone de visão dos sensores (ver Figura 4.4). As células dentro deste cone tem seu atributo c_{ij} diminuído de 1. Uma célula tem seu atributo s_{ij} alterado de *espaço livre* para *ocupado*, quando seu c_{ij} é maior que 2, caso contrário, ela muda de *ocupado* para *espaço livre*. Este processo permite um fácil tratamento de obstáculos tanto dinâmicos quanto estáticos. Durante os experimentos, limitamos a propriedade certeza c_{ij} ao intervalo [0,15]. É possível através da interface desenvolvida alterar os parâmetros

das sensoriais, atualização das coordenadas do robô (localização), manutenção do mapa, verificação da condição de parada, planejamento e execução da ação a ser realizada pelo robô.

relacionados ao valor máximo, incremento ou decremento desta propriedade em tempo real.

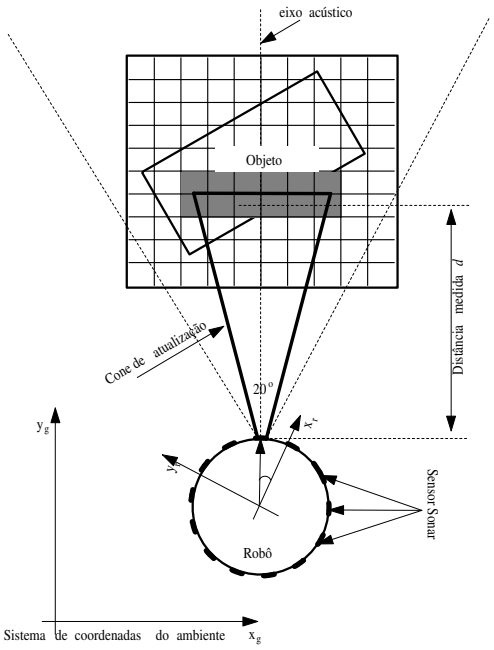


FIGURA 4.4 – Atualização do mapa. A figura mostra um diagrama esquemático da detecção de um objeto por um dos sensores do robô. As células em cinza tiveram seus atributos certeza incrementados de 3, enquanto que as demais dentro do cone de atualização tiveram seu atributo diminuído de 1. A informação (d,α) representa a informação fornecida pelos sensores do robô no sistema de coordenadas polar.

4.2.5 Preenchimento de Lacunas

Este processo é responsável por realizar um refinamento na representação do ambiente eliminando os ruídos provocados pela precisão finita da detecção e classificação das células do ambiente. O processo pode ser dividido em duas etapas:

- mudar células classificadas como *espaço livre* para *ocupado*. Neste caso, se uma célula ij possui células vizinhas $(i-1, j-1)$ e $(i+1, j+1)$, ou $(i+1, j-1)$ e $(i-1, j+1)$ classificadas como *ocupado* então a célula ij tem suas propriedades s_{ij},c_{ij} e p_{ij} alteradas para *ocupado*, o valor máximo definido pelo usuário e 1, respectivamente;

- mudar células classificadas como *não explorado* para *espaço livre*. Se a maioria das células vizinhas à célula ij tem sua propriedade *estado* diferente de *não explorado* então a célula ij tem suas propriedades s_{ij},c_{ij} e p_{ij} alteradas para, respectivamente, *espaço livre*, 0 e 1. Neste caso, é preferível alterar o estado de uma célula de *não explorado* para *espaço livre* do que para *ocupado*, pois no primeiro caso, existe a possibilidade do robô planejar um caminho que passe por ela e, assim, classificá-la convenientemente, entretanto, no último, o robô

somente realizará a classificação correta se for definido um objetivo próximo a ela, caso contrário, ele nunca a atingirá e a classificará corretamente.

4.2.6 Planejamento

O planejamento realizado pelo agente é baseado na solução de problemas de valores de contorno envolvendo a equação 3.2 considerando as condições de contorno de *Dirichlet* e o potencial (p) armazenado na representação construída pelo agente. Utilizamos a equação 3.2 com as seguintes alternativas para a função $F(\mathbf{r})$,

a) $F(\mathbf{r}) = 0, \ \forall \, \mathbf{r}$, resultando na equação de Laplace;

b) $F(\nabla p) = \epsilon \ \nabla p \ \mathbf{v}$, resultando na equação linear 3.3;

c) $F(\nabla p) = \epsilon \ (\nabla p)^2$;

onde $\mathbf{v} \in \Re^{\mathbf{2}}$ é um vetor constante, e $\epsilon \in [-1, 1]$ é um escalar que determina a influência de $\mathbf{v}$. Estas alternativas afetam de maneira crucial o comportamento do robô e podem ser escolhidas de acordo com a tarefa a ser executada, como será visto na Seção 5.

A equação 3.2 é resolvida numericamente sobre a região já explorada pelo robô através de um método de relaxação similar à *iteração valorada* [SUT 98]. Isto começa com a discretização do ambiente, que consiste em dividir a região de interesse em um conjunto de pontos. Somente nestes pontos é que as soluções da equação serão obtidas. Ao conjunto dos pontos discretos dá-se o nome de *malha*. Quanto maior for o número de pontos e conseqüentemente mais densa for a malha, mais fiel será o resultado numérico obtido. As malhas podem ser divididas em 2 grupos: estruturadas e não-estruturadas. As estruturadas apresentam regularidades na distribuição espacial dos pontos (mapas baseados em grades). As não-estruturadas não apresentam essa regularidade, entretanto permitem a discretização de domínios com geometria mais complexa de forma mais direta do que seria possível com malhas estruturadas.

Em seguida, os termos que aparecem nas equações são escritos em função dos valores das incógnitas em pontos discretos adjacentes. O resultado é um conjunto de equações algébricas. Estas equações são denominadas aproximações por diferenças finitas. O resultado final deste processo é um sistema de equações algébricas, denominadas equações de diferenças finitas (EDF).

As aproximações por diferenças finitas podem ser obtidas de diversas maneiras. As mais comuns são [OLI 2000]: expansão em série de Taylor e interpolação polinomial. A última é comumente utilizada quando o espaçamento na malha não é uniforme. As EDFs podem utilizar diferenças progressivas, quando utilizam na aproximação um ponto adiante do ponto que está sendo expandido; ou atrasadas, quando utilizam na aproximação um ponto atrás do ponto que está sendo expandido. Quando utilizamos ambos os pontos estamos realizando uma aproximação por diferenças centrais.

Existem dois métodos comumente utilizados na solução numérica de equações diferenciais [OLI 2000]: o método Gauss-Seidel e o método de Sucessivas Sobre-Relaxações (SOR). O método Gauss-Siedel é obtido a partir da discretização da equação diferencial por meio de diferenças centrais de segunda ordem[OLI 2000].

Considerando a equação 3.2,com $F(\mathbf{r}) = 0,\ \forall\ \mathbf{r}$, temos

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 0 \tag{4.1}$$

onde x e y representam as coordenadas espaciais no sistema de referência global. A discretização da equação acima por meio de diferenças centrais de segunda ordem usando séries de Taylor resulta em

$$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\Delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\Delta y)^2} = 0 \tag{4.2}$$

onde Δx e Δy correspondem ao espaçamento de cada ponto da malha no eixo x e no eixo y, respectivamente. Reescrevendo a equação anterior como

$$p_{i+1,j} - 2p_{i,j} + p_{i-1,j} + \left(\frac{\Delta x}{\Delta y}\right)^2 (p_{i,j+1} - 2p_{i,j} + p_{i,j-1}) = 0 \tag{4.3}$$

e definindo $\beta = \frac{\Delta x}{\Delta y}$, obtemos

$$p_{i+1,j} + p_{i-1,j} + \beta^2 p_{i,j+1} + \beta^2 p_{i,j-1} - 2(1+\beta^2)p_{i,j} = 0 \tag{4.4}$$

O método Gauss-Seidel surge quando reescrevemos a equação acima da seguinte maneira

$$p_{i,j} = \frac{1}{2(1+\beta^2)} \left(p_{i+1,j} + p_{i-1,j} + \beta^2 p_{i,j+1} + \beta^2 p_{i,j-1}\right) \tag{4.5}$$

Como o espaçamento é regular e igual em ambas as coordenadas, temos $\beta = 1$. Isto acarreta a equação

$$p_{i,j}^{t+1} \ = \ \frac{1}{4} \left(p_{i-1,j}^{t+1} + p_{i+1,j}^{t} + p_{i,j-1}^{t+1} + p_{i,j+1}^{t}\right) \tag{4.6}$$

Resultados teóricos e observações mostram que conforme progridem as iterações do método Gauss-Seidel, a diferença entre sucessivas aproximações p^{t+1} e p^t diminui. É possível aumentar deliberadamente essa pequena diferença extrapolando o valor de p^{t+1} de maneira que ele se aproxime mais rapidamente da solução numérica do sistema. Esta é a base para o método SOR. Dessa forma, o cálculo do potencial usando o método SOR é igual a :

$$p_{SOR}^{t+1} = p_{GS}^{t+1} + \lambda \left(p_{GS}^{t+1} - p^t\right) \tag{4.7}$$

considerando $w = 1 + \lambda$. Obtemos

$$p_{SOR}^{t+1} = w p_{GS}^{t+1} + (1-w)p^t \tag{4.8}$$

onde w é uma fator de relaxação. Para a convergência do SOR, $0 < w < 2$. Substituindo 4.6 em 4.8, obtemos

$$p_{i,j}^{t+1} \ = \ p_{i,j}^{t} + \frac{w}{4}(p_{i-1,j}^{t+1} + p_{i+1,j}^{t} + p_{i,j-1}^{t+1} + p_{i,j+1}^{t} - 4p_{i,j}^{t}) \tag{4.9}$$

O melhor valor para o parâmetro de relaxação w é definido em [PRE 92] como

$$w = \frac{2}{(1+\sqrt{1-\rho^2})} \tag{4.10}$$

onde

$$\rho = \frac{1}{2}\left[\cos\left(\frac{\pi}{L_x}\right) + \cos\left(\frac{\pi}{L_y}\right)\right] \quad . \tag{4.11}$$

e L_x e L_y correspondem à quantidade de colunas e linhas respectivamente da região discretizada. De maneira similar é possível escrever as variações da equação 3.2 com $F(\nabla p) = \epsilon\ \nabla p\ \mathbf{v}$ e $F(\nabla p) = \epsilon\ (\nabla p)^2$ usando Gauss-Seidel. Iremos considerar apenas Gauss-Seidel para estas variações por razões que ficarão claras na Seção 5.5.

Considere a equação 3.2 com $F(\nabla p) = \epsilon\ \nabla p\ \mathbf{v}$, obtemos a equação 3.4. A discretização desta equação diferencial por meio de diferenças centrais de segunda ordem resulta em

$$\begin{aligned} p_{i,j}^{t+1} \ &= \ \frac{1}{4}(p_{i,j-1}^{t+1} + p_{i-1,j}^{t+1} + p_{i+1,j}^{t} + p_{i,j+1}^{t}) - \\ &\frac{\epsilon}{4}((p_{i+1,j}^{t} - p_{i-1,j}^{t+1})v_x + (p_{i,j+1}^{t} - p_{i,j-1}^{t+1})v_y) \end{aligned} \tag{4.12}$$

onde v_x e v_y são as componentes do vetor $\mathbf{v}$. Com a equação 3.2 e $F(\nabla p) = \epsilon\ (\nabla p)^2$, obtemos

$$\nabla^2 p + \epsilon\ (\nabla p)^2 = 0 \tag{4.13}$$

De maneira similar, a discretização desta equação por meio de diferenças centrais de segunda ordem resulta em

$$\begin{aligned} p_{i,j}^{t+1} \ &= \ \frac{1}{4}(p_{i-1,j}^{t+1} + p_{i+1,j}^{t} + p_{i,j-1}^{t+1} + p_{i,j+1}^{t}) \\ &+\frac{\epsilon}{16}((p_{i,j+1}^{t} - p_{i,j-1}^{t+1})^2 + (p_{i+1,j}^{t} - p_{i-1,j}^{t+1})^2); \end{aligned} \tag{4.14}$$

4.2.7 Condição de Parada

A condição de parada é um módulo essencial para o comportamento do agente. Ele é responsável por decidir quando o agente deve parar de atuar no ambiente. Esta decisão é dependente da tarefa a ser realizada, por exemplo: no caso do planejamento de caminhos baseado em um mapa disponível, a condição de parada consiste em identificar quando o robô atingiu a posição destino determinada pelo usuário; no caso da localização de um objeto, a condição de parada corresponde ao reconhecimento do objeto procurado entre os objetos identificados no ambiente[4].

No caso do processo exploratório, a condição de parada consiste em verificar se não existe alguma célula *acessível ij* com s_{ij}=*espaço livre* ao lado de alguma célula *mn* com s_{mn}=*não explorado*. Se esta condição é verdadeira, o robô pára o processo exploratório, caso contrário, ele continua sua exploração até que esta condição seja verdadeira. Para decidir se uma célula explorada do tipo espaço livre é acessível ou não, usamos uma variação do algoritmo de crescimento de regiões[GON 92] que segmenta o espaço livre em uma região conectada(acessível) e outra desconectada (inacessível). Este algoritmo é executado concorrentemente com o processo de atualização do campo potencial.

Esta estratégia é mais robusta que considerar que o processo deve parar quando todas as células tiverem seus atributos ($s_{ij} \neq$ *não explorado*) pois, dependendo da

[4]Neste caso, o robô deve possuir mecanismos para o reconhecimento de objetos a partir de um sistema de visão. O módulo responsável pela manipulação da câmera do robô não foi implementado, entretanto nossa arquitetura suporta a tarefa relacionada à localização de objetos (ver Seção 5.4.7)

estrutura do ambiente, um subconjunto das células do mapa nunca será alcançado e o algoritmo falhará. Isto é ilustrado na Figura 4.5, onde as células de cor cinza nunca serão alcançadas e o algoritmo conseqüentemente falhará.

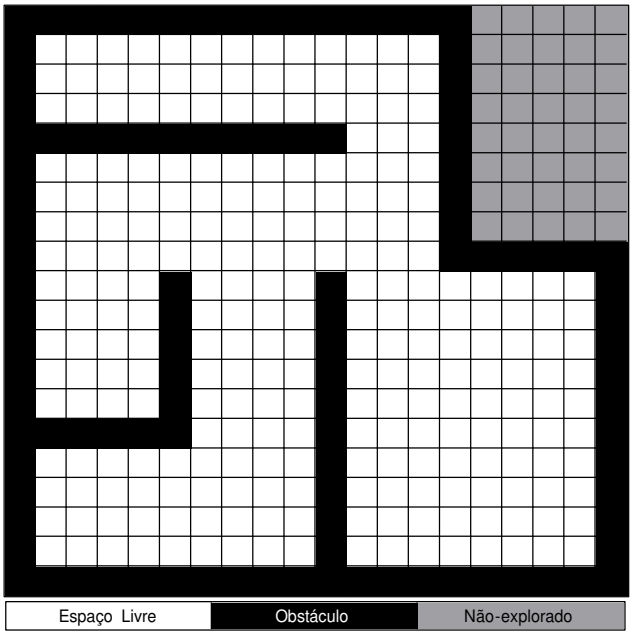


FIGURA 4.5 – Ambiente com concavidades. As células brancas denotam regiões pertencentes ao espaço livre, as pretas denotam obstáculos e as cinzas denotam regiões não exploradas.

Esta estratégia também é mais robusta que simplesmente procurar alguma célula ij com s_{ij}=*não explorado* ao lado de alguma célula mn com s_{mn}=*espaço livre*. Pois, ruidos inseridos na leitura dos sensores podem causar uma classificação errônea de células inacessíveis como mostra a seqüência de Figuras 4.7 (a), (b) e (c). Neste caso, sempre vai existir pelo menos uma célula do *espaço livre* ao lado de uma célula *não explorada*, conseqüentemente, o processo de exploração irá ser executado eternamente.

Para segmentar o espaço livre assumimos que cada célula possui um atributo chamado *região* que armazena valores entre 0 e 5. Inicialmente, todas as células são classificadas como pertencentes ao espaço livre, e têm seu atributo *região* igual a 5. Durante o movimento do robô, a célula que representa sua posição tem seu atributo *região* atualizado com o maior valor permitido, neste caso, este foi definido, empiricamente, igual a 5. Em seguida, este valor é propagado às demais células através de um processo de propagação semelhante aos métodos *wavefront*, chamado *propagação de incremento*. As células do espaço livre têm seu atributo *região* atualizado com o maior valor deste atributo de suas células vizinhas. Isto é realizado a fim de conectar as células que representam o espaço livre realizando a segmentação do ambiente.

Este procedimento é executado a partir da posição do robô em todas as direções a fim de que todas as células do espaço livre sejam atualizadas. A execução em diversas direções serve para agilizar o processo de segmentação do espaço livre, pois dependendo do formato do ambiente, a execução em apenas um sentido pode levar este processo a realizar mais iterações que o necessário. A Figura 4.6 ilustra o processo de *propagação de incremento*.

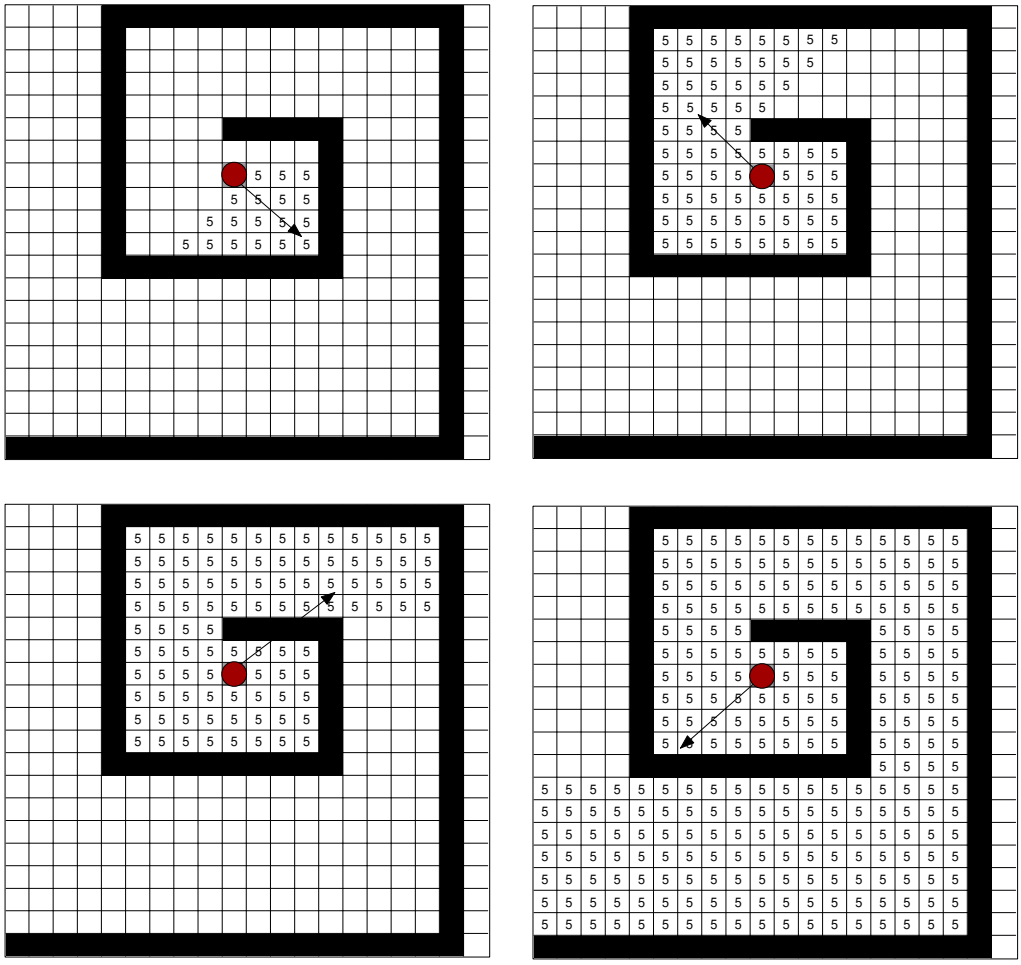


FIGURA 4.6 – Propagação do atributo *região* para as células do espaço livre a partir da posição do robô. As células de cor preta representam obstáculos e as de cor branca representam o espaço livre.

Além da *propagação de incremento*, é executado outro tipo de propagação, chamado *propagação de decremento*. Este processo é responsável por decrementar de 1 o valor do atributo *região* de todas as células do espaço livre. Este procedimento tem como objetivo desconectar aquelas células que não possuem um caminho viável até a posição do robô. Se a célula estiver desconectada ela não irá receber o valor propagado pela posição do robô, na fase de *propagação de incremento*, e terá sempre o valor de seu atributo *região* decrementado até atingir o valor igual a 0 (ver Figura 4.7 (d)).

Estes dois processos eliminam as células marcadas como pertencentes ao espaço livre através da redução do valor de seu atributo *região*. Todas as células pertencentes ao espaço livre terão valor de região maior que zero. Enquanto que aquelas marcadas pelo sinal ruidoso dos sensores terão valor igual à 0. Os dois processos são combinados na Figura 4.7. O algoritmo é simples e é executado concorrentemente com o sistema de maneira incremental.

Após a execução dos dois processos de propagação, é verificada a existência de alguma célula classificada como *espaço livre* ao lado de uma célula classificada como *não explorado*. Esta verificação leva em consideração tanto a classificação da célula quanto sua propriedade *região*. Aquelas células classificadas como *espaço livre* e com

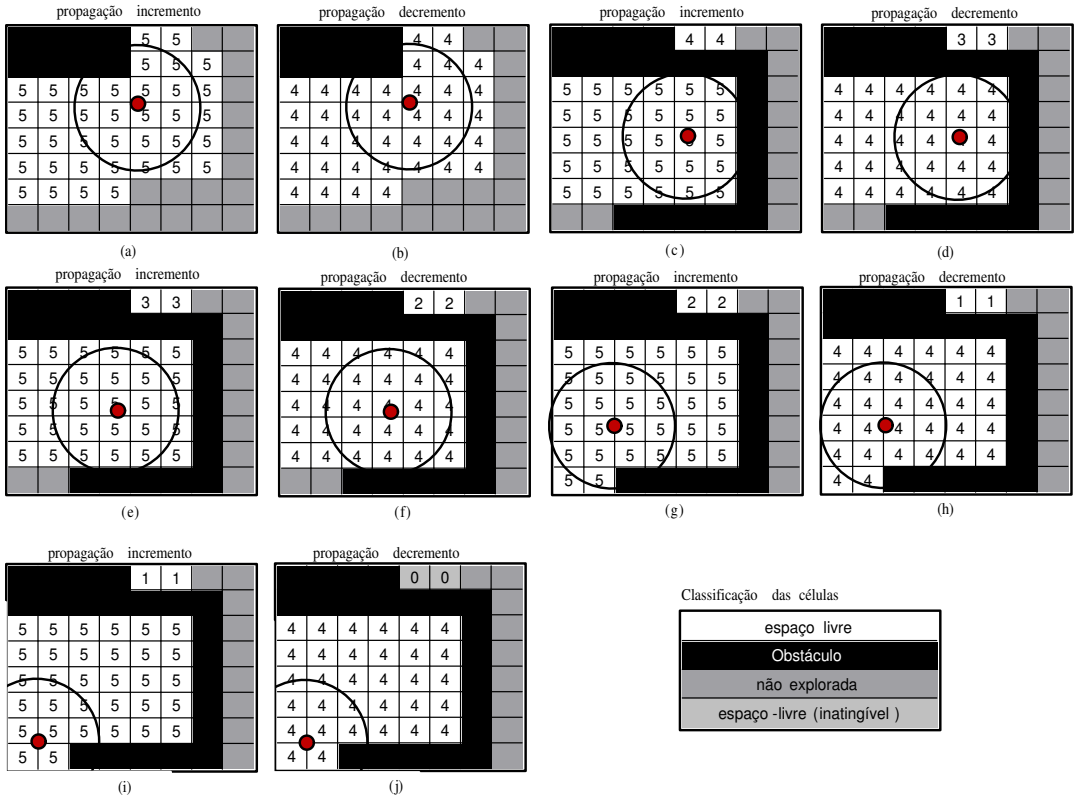


FIGURA 4.7 – Propagação do atributo região. As Figuras (a)-(j) ilustram o processo de propagação de incremento e de decremento do atributo região. As células de cor preta representam obstáculos, as de cor branca representam espaço livre, as de cor cinza escuro representam regiões não exploradas e as de cor cinza claro representam células de espaço livre inatingíveis pelo robô.

valor igual a 0 em sua propriedade região são descartadas, pois elas são consideradas inatingíveis pelo robô. O robô continua a explorar o ambiente até não existir célula classificada como *não explorado* ao lado de uma célula classificada como *espaço livre* e com atributo região maior que 0, ou seja, até não existir regiões desconhecidas que possam ser visitadas pelo robô.

É importante observar o valor máximo que o atributo região pode possuir. Ele foi obtido empiricamente, entretanto, é possível dar-lhe um significado. Ele pode ser visto como um valor que indica quantas vezes as células do espaço livre estarão exposta ao processo de *propagação de decremento* mantendo ainda sua participação na verificação feita no processo de parada. Sua magnitude está diretamente relacionada à complexidade do ambiente; quanto mais complexa for a estrutura do ambiente, maior deve ser esta magnitude para que todas as células do espaço livre recebam o valor propagado pelo robô e continuem a participar da verificação.

Células que estiverem expostas à *propagação de decremento* por um período maior que o definido pela sua propriedade região, sem receberem propagação alguma a partir do robô, não participarão do processo de parada. Lembre-se que a exploração termina quando não houver mais células não exploradas ao lado de células espaço livre. Isto pode fazer com que aquelas células do espaço livre próximas às regiões não exploradas, em ambientes muito complexos, com corredores em forma de espirais,

fiquem muito tempo expostas ao processo de *propagação de decremento* e tenham seu atributo decrementado até zero. Neste caso, estas células não participarão do processo e o robô conseqüentemente não explorará todo o ambiente.

Observe que como o valor máximo do atributo região é dependente da estrutura do ambiente ele deve ser estipulado antes do robô começar a adquirir informações sobre sua área de atuação. Para resolver este problema é possível fazer com que este valor máximo seja adaptativo a fim de que ele se ajuste automaticamente à complexidade do ambiente que está sendo explorado.

4.2.8 Processamento da Ação

As ações são extraídas a partir do potencial gerado e armazenado na grade de ocupação. Para controlar o robô, existe um protocolo bem definido que permite programas clientes acessarem o estado completo do robô (leituras sensoriais, velocidade, posição relativa, entre outros) e controlar sua movimentação. A movimentação do robô é realizada através de um único comando que define sua velocidade linear v e seu ângulo de rotação φ. Este comando permite realizar um movimento combinado, rotação em conjunto com a translação, gerando trajetórias suaves.

O ângulo de rotação φ é calculado utilizando a direção corrente do robô θ e a nova direção ϕ, computada através do gradiente descendente na célula $(p_{i,j})$ contendo a posição corrente do robô. O cálculo do gradiente é feito sobre uma representação discreta e resulta em um vetor com valores contínuos. O robô ajusta a direção de seu movimento ao ângulo ϕ dado por

$$\phi = \arctan(p_{i-1,j} - p_{i+1,j}, p_{i,j-1} - p_{i,j+1}) \tag{4.15}$$

onde $\arctan(x, y)$ é a tangente inversa tomada no intervalo $[-\pi, \pi]$.

O ângulo φ é calculado através da seguinte fórmula:

$$\varphi = \theta(-)\phi \tag{4.16}$$

onde (-) define um operador para dois operandos, α e β (em graus), que é usado na forma c=α (-) β; e c é o menor arco entre α e β. Assim, c está sempre no intervalo $-180^0 < c < 180^0$.

A velocidade v é calcula da seguinte maneira:

$$v_t = \eta\ \hat{v}_t + (1 - \eta)v_{t-1} \tag{4.17}$$

onde v_{t-1} representa a velocidade no instante $(t-1)$; $\hat{v}_t$ representa o valor estimado para a velocidade do robô no instante t e η define a suavidade na transição entre velocidades calculadas em intervalos de tempos subseqüentes. Nos experimentos foi usado $\eta = 0,5$. A velocidade estimada $\hat{v}_t$ é baseada principalmente na rotação realizada pelo robô, sendo que $\hat{v}_t$ diminui linearmente com o ângulo φ. Ela é calculada de duas maneiras diferentes:

- **dentro da região de colisão** : a região de colisão corresponde a uma região circular de raio r_{max} centrada no robô. Esta região define uma área com alta probabilidade de colisão, por exemplo: passagens estreitas. Quando algum obstáculo é identificado a uma distância $d \leq r_{max}$, o robô começa a sofrer uma influência $\rho = d\ /\ r_{max}$ proporcional a sua distância ao mais próximo

obstáculo. Esta influência causa um decremento na velocidade do robô de acordo com sua proximidade aos objetos a sua volta. Dessa forma, a velocidade $\hat{v_t}$ é calculada utilizando:

$$\hat{v_t} = v_{max}(0,2 + 0,8\frac{180 - |\varphi|}{180}\ \rho) \tag{4.18}$$

- **fora da região de colisão** : quando o robô está fora da região de colisão a influência ρ gerada pelos obstáculos é igual a 1, resultando em

$$\hat{v_t} = v_{max}(0,2 + 0,8\frac{180 - |\varphi|}{180}) \tag{4.19}$$

Em ambos os casos, v_{max} representa a velocidade máxima (v) definida no programa cliente, com $0 < v \leq 25,4\ cm/s$; e $r_{max} = 1,3\ m$.

O cálculo de (4.18) e o valor de seus parâmetros foram escolhidos heuristicamente para propiciar um movimento razoavelmente suave. Para compreendermos estes parâmetros, vamos transformar a equação 4.18 para sua forma geral

$$\hat{v_t} = v_{max}(\alpha + (1 - \alpha)\frac{180 - |\varphi|}{180}\ \rho) \tag{4.20}$$

onde $0 \leq \alpha \leq 1$. Considerando que o robô não está realizando um movimento próximo aos obstáculos presentes no ambiente temos $\rho = 1$. Assim a equação 4.20 se reduz a equação 4.19. Assumindo o caso mais simples com $\alpha = 0$, temos a seguinte equação

$$\hat{v_t} = v_{max}(\frac{180 - |\varphi|}{180}) \tag{4.21}$$

Observe que a velocidade estimada $\hat{v_t}$ é a velocidade máxima permitida proporcional ao ângulo de rotação do robô $0 \leq \varphi \leq 180$. Quanto maior φ menor é a velocidade estimada $\hat{v_t}$ do robô. Isto serve para controlar o ângulo de abertura da curva que o robô irá seguir.

Assumindo o ângulo de rotação máximo $\varphi = 180$, temos $\frac{180-|\varphi|}{180} = 0$, conseqüentemente, $\hat{v_t} = 0$. Neste caso, o comando enviado ao robô irá fazê-lo parar, pois estamos dizendo a ele para se deslocar com velocidade linear nula. Para resolver este problema definimos uma velocidade base que corresponde a 20% da velocidade máxima do robô ($\alpha = 0,2$). Os outros 80% são determinados pela rotação a ser realizada pelo robô. Isto é ilustrado em 4.19 através dos fatores $0,2$ e $0,8$, respectivamente. No caso do ângulo de rotação ser igual a $\varphi = 0$, o robô irá se deslocar com a velocidade máxima permitida, pois $\frac{180-|\varphi|}{180} = 1$.

Além da rotação é importante considerar a velocidade de deslocamento do robô quando ele está muito próximo a obstáculos, por exemplo, quando ele está se deslocando por um corredor estreito ou quando ele está tentando passar por uma porta aberta. No último caso em que o robô deve passar por uma porta, imagine que ele esteja de certa forma alinhado perpendicularmente com a abertura da porta. Ele apenas deve se mover sem realizar qualquer rotação. Nesta situação, o ângulo de rotação definido pelo campo vetorial é igual a 0, $\varphi = 0$. Conforme foi mostrado anteriormente isto fará com que o robô se desloque com a maior velocidade possível.

Esta situação é crítica pois no caso de uma colisão iminente, decorrente de deslizamentos, o robô terá pouco tempo para reagir convenientemente a ela devido

a sua velocidade. Tendo em mente que o efeito desta colisão é proporcional à velocidade do robô, dependendo da velocidade máxima estabelecida, esta colisão pode causar sérias avarias no equipamento.

Dessa forma, consideramos que a velocidade estimada do robô é influenciada tanto pelo seu ângulo de rotação, quanto pela sua proximidade aos obstáculos presentes no ambiente. Sendo assim, inserimos um fator ρ na equação 4.19 produzindo 4.18.

4.3 Comportamento Exploratório

No planejamento de caminhos, uma célula pode estar em um dos três possíveis estados: livre, ocupado por um obstáculo ou por um alvo. Para a exploração, nós introduzimos uma quarta possibilidade que correspondem ao estado *não explorado*. As células não exploradas equivalem às células objetivo com a diferença que elas mudam seu estado quando estão dentro da região definida pelo campo de visão dos sensores do agente.

Este mecanismo gera um comportamento exploratório autônomo pois estes objetivos são colocados nas fronteiras do espaço explorado e atuam como pontos de atração que guiam o agente para regiões ainda não exploradas. O agente persegue estas células até que todas sejam alcançadas, quando isto ocorre é assumido que o robô visitou todo o ambiente.

O processo de exploração inicia redimensionando o mapa interno do agente para que ele possa representar o maior ambiente possível a ser explorado. Inicialmente, o agente está em um estado sem conhecimento, e todas as células do seu mapa são classificadas como *não exploradas*. Conforme o seu movimento, as células visitadas têm seus estados alterados de acordo com a resposta dos sensores.

O agente tem sensores de proximidade que detectam obstáculos no ambiente. Estes sensores alimentam o módulo *processamento sensorial*, o qual produz uma região chamada *janela de ativação* que tem aproximadamente o tamanho do alcance médio dos sensores do robô. Ela é usada para atualizar a representação interna do agente e indicar as células do mapa que são recrutadas para participarem do cálculo do potencial. Se uma célula está em qualquer instante dentro da janela de ativação, ela se torna parte do espaço explorado conseqüentemente podendo participar do cálculo do potencial durante o processo exploratório. Nós chamamos *região de ativação* ou *região potencial* ao conjunto de células ativas que compõem o espaço explorado[5]. O limite da região de ativação corresponde ao limite entre as regiões conhecidas e desconhecidas do ambiente.

O campo potencial para a região explorada corrente é calculado com as células não exploradas como objetivos temporários. Observe que o valor do potencial para células objetivo é igual a 0 (ver Seção 4.2). Essas células irão atrair o agente e quando elas forem alcançadas, elas mudarão seu estado para *ocupado* ou espaço-livre dependendo de sua resposta sensorial, dessa forma, tornando-se parte da região de

[5]É importante destacar que a região explorada corresponde a um subconjunto de células do mapa $M_{x_{max} \times y_{max}}$ do ambiente. Sendo assim, para agilizar o processamento apenas atualizamos este subconjunto, o qual é definido em uma região retangular (x_{inf}, y_{inf},x_{sup},y_{sup}), chamada janela global, onde x_{inf} e y_{inf} representam os menores valores encontrados para a coordenada x e y das células do espaço explorado, respectivamente; e x_{sup} e y_{sup} representam os maiores valores encontrados para a coordenada x e y, respectivamente

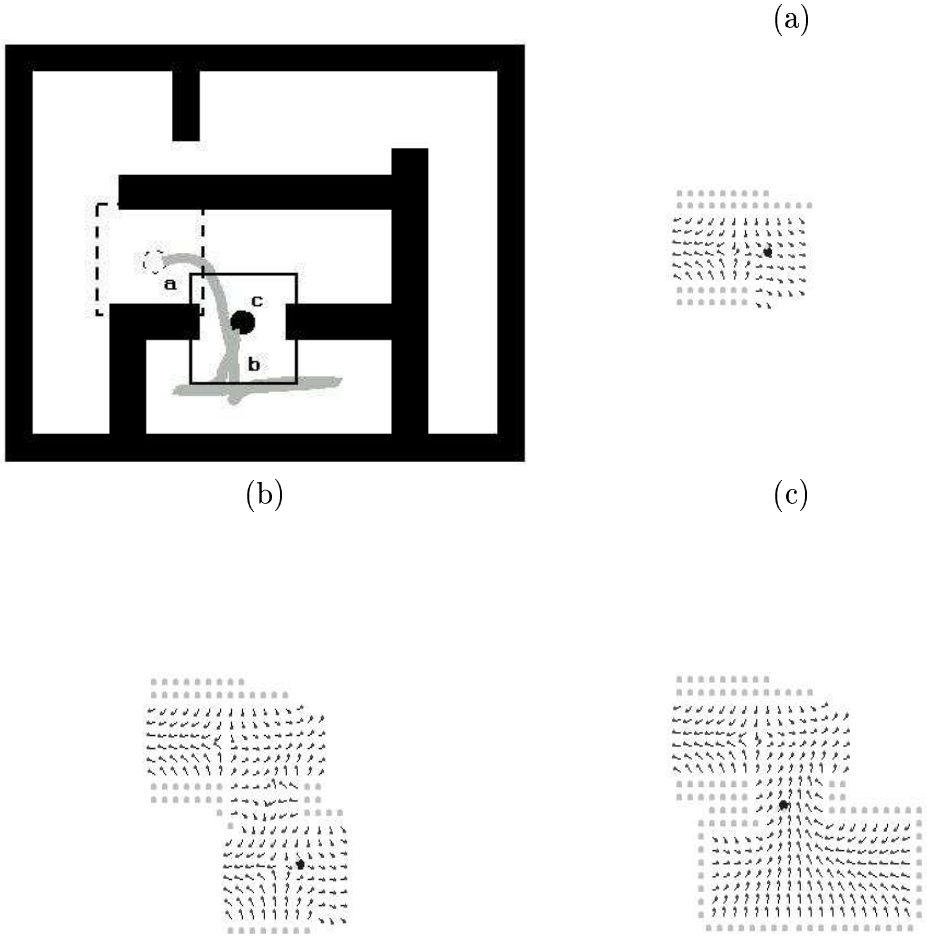


FIGURA 4.8 – Processo de Exploração: Acima, a trajetória seguida pelo agente. (a)-(c) mostram *snapshots* do campo e do mapa para cada posição correspondente na trajetória.

ativação, e conseqüentemente outras células *não exploradas* se tornarão atrativas e o processo executará novamente.

A *ação* gerada pela arquitetura corresponde ao deslocamento do agente seguindo o gradiente do potencial em sua posição corrente no mapa interno, o qual inicialmente corresponde ao centro da janela de ativação. Neste caso, o gradiente apontará para a direção da região mais próxima desconhecida ou para a maior no caso de existirem duas ou mais regiões desconhecidas à mesma distância. Se nenhum obstáculo for detectado, o limite de sua região de ativação é igual a zero e o potencial é igual a zero em todas as células. Neste caso, ou em qualquer situação onde não existe gradiente para guiar o agente, ele simplesmente segue em linha reta para frente.

A Figura 4.8 [6] ilustra o comportamento exploratório gerado pelo nosso agente. Ele seqüencialmente ocupa as posições *a,b* e *c*, com a posição final indicada pelo círculo escuro. A Figura 4.8 também mostra *snapshots* do mapa interno nas posições correspondentes. Observe que o campo vetorial sempre aponta para a região não

[6] É importante destacar que esta figura é meramente ilustrativa. Em todos os experimentos foram usadas janelas circulares de ativação.

explorada mais próxima no ambiente. Na Figura 4.8 (c), vemos como o campo vetorial muda puxando o agente de um beco sem saída.

4.3.1 Janela de Ativação

Nos experimentos iniciais utilizamos uma janela de ativação circular com raio, chamado *raio de ativação*, fixo e igual a 2 m. Observamos que manter o raio de ativação fixo ocasionava alguns problemas, entre eles a dificuldade de identificar passagens estreitas (ver Figura 4.9(a)), portas(ver Figura 4.9(c)), etc. pois, a quantidade de células atualizadas como *ocupadas* na identificação de um obstáculo aumenta com o comprimento deste raio. Assim, dependendo deste valor, portas não eram identificadas, devido às células próximas a sua entrada serem erroneamente classificadas como ocupadas.

Além deste problema, ambientes que possuem corredores ou salas de diferentes dimensões, maiores que o alcance dos sensores ou do comprimento do raio de ativação, são muito complexos para serem modelados com uma janela de tamanho fixo, pois neste caso, o ambiente se torna esparso, aumentando a complexidade do processo de tomada de decisão (ver Figura 4.9(b)).

Uma solução encontrada foi ajustar, empiricamente, o valor do raio da janela de ativação de acordo com o retorno dos sensores, o qual dependerá da largura de seus corredores ou das dimensões de sua sala (ver Figura 4.9(d)). Dessa forma, a primeira proposta é o ajuste automático do raio de ativação r utilizando a menor entre as medidas dos sensores $\{s_i\}$, do robô a cada instante desde que esta não seja menor que um valor crítico r_{min}, que define a janela mínima, ou seja, considerando $P = \{p_i \mid 0 \leq\ i \leq 15\}$ o conjunto ordenado das medidas sensorias $\{s_i\}$, onde $p_i < p_j$ se e somente se $i < j$. Esta proposta pode ser escrita como

$$r\ =\ \max(p_0, r_{min}). \tag{4.22}$$

Uma segunda proposta seria o ajuste do raio ativação através da média das 3 menores medidas dos sensores do robô, ou seja,

$$r\ =\ \max(\frac{(p_0 + p_1 + p_2)}{3}, r_{min}) \tag{4.23}$$

Uma vantagem da janela ajustável é o refinamento automático do mapa. Os experimentos realizados mostram que um mapa de melhor qualidade é obtido, pois a incerteza na detecção das paredes é reduzida ao mínimo aceitável definido pelo menor retorno dos sensores. Observe as Figuras 4.9(c) e 4.9(d), a primeira mostra a atualização do mapa utilizando uma janela de raio de ativação fixo e a última mostra a atualização do mapa utilizando uma janela de raio de ativação adaptativo. No primeiro caso, observamos que a entrada para a sala **A** foi fechada devido a células próximas a ela serem classificadas como ocupadas decorrente da detecção da porta. No segundo caso, observamos que a porta não foi detectada, pois a região delimitada pelo raio de ativação não a incluia, conseqüentemente, foi mantida a abertura para a sala **A**.

4.3.2 Conhecimento local × conhecimento global

Esta subseção discute um refinamento da exploração baseada em PVC. Sabemos que a região explorada pelo agente corresponde a um subconjunto de células do

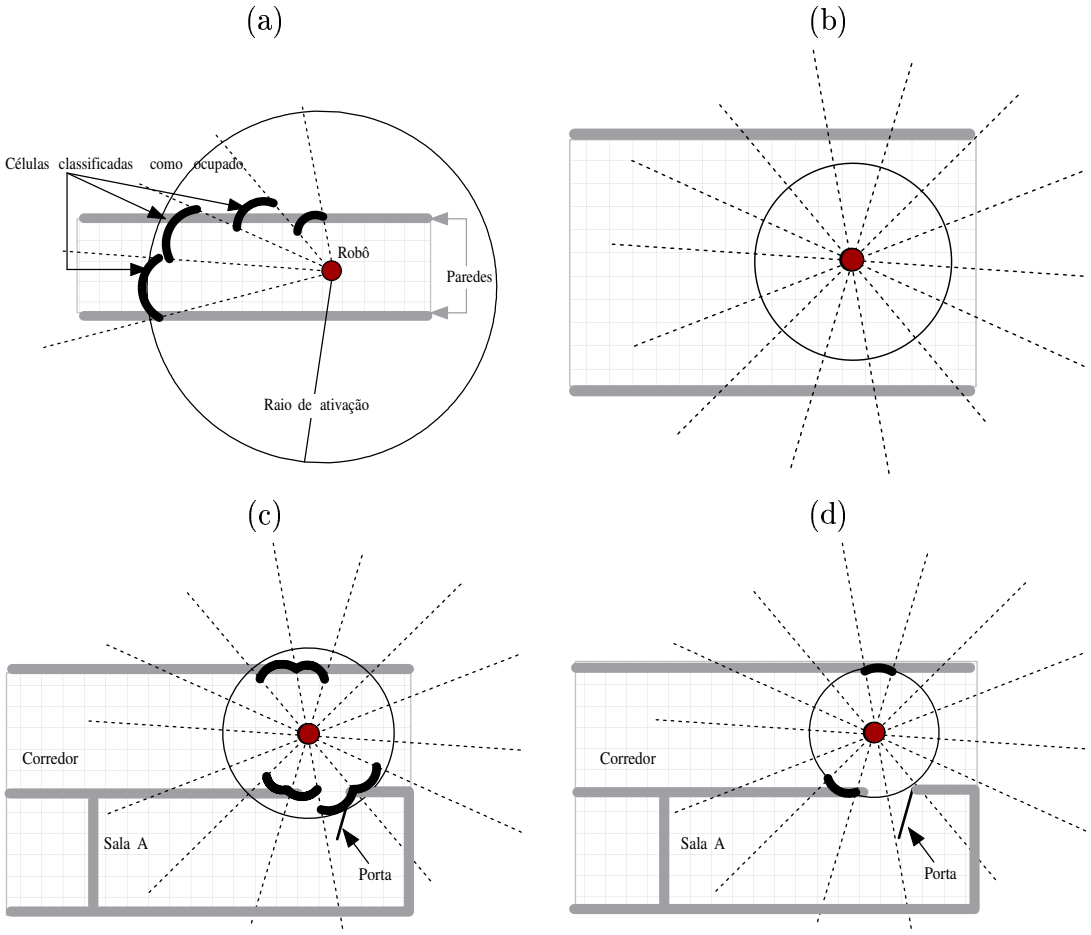


FIGURA 4.9 – Janela de ativação. As Figuras (a)-(c) mostram a influência de uma janela de ativação fixa no mapeamento de um ambiente com diferentes dimensões. A Figura (d) mostra a influência de uma janela de ativação de tamanho variável no processo de mapeamento de um ambiente. Os arcos de cor preta ilustram células marcadas como ocupadas no mapa do ambiente.

mapa do ambiente. Para agilizar o processamento do cálculo do potencial definimos uma região retangular, chamada janela global, que circunscreve este subconjunto, e apenas atualizamos as células desta região com um número finito de iterações (relaxação parcial). Veja a Figura 4.10.

Por que a relaxação parcial funciona convenientemente com poucas iterações? Ela funciona corretamente porque na maioria das vezes o robô está próximo de uma região desconhecida. Nestes casos, poucas iterações são necessárias para que esta região comece a atrair o robô em sua direção. Com isto estamos ignorando a situação das células distantes da célula que possui a posição do robô. Simplesmente não estamos levando em consideração se elas estão relaxadas completamente ou não.

Baseado nesta idéia por que não atualizar apenas as células que estão dentro da janela local do robô ? Ela possui todas as informações necessárias para que ele realize sua exploração, por exemplo, células que representam regiões desconhecidas e a própria posição do robô. Considerando isto, obtemos um ganho expressivo no desempenho do sistema pois a quantidade de células atualizadas na janela local, por

iteração, é expressivamente menor que a quantidade de células pertencentes à janela global na maior parte do tempo[7]. Isto pode ser visto na Figura 4.10.

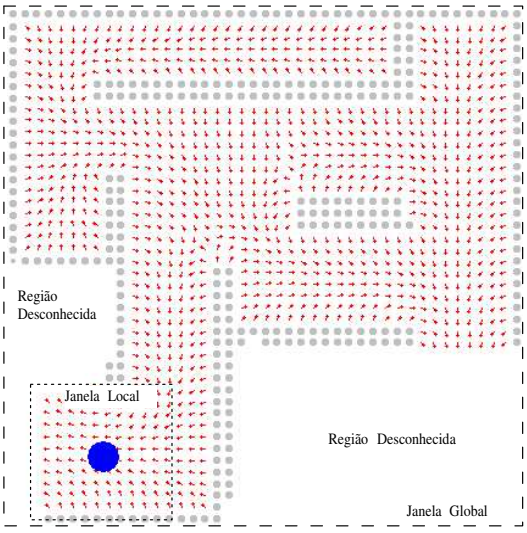


FIGURA 4.10 – Exploração orientada a conhecimento local. A janela local denota todas as células que são utilizadas na relaxação local. A janela global denota todas as células que são utilizadas na relaxação global.

Na maior parte do tempo apenas a relaxação local baseado na janela local pode ser usada, entretanto, em alguns casos é necessário a relaxação global. Por exemplo, imagine o caso, onde o robô está explorando um ambiente com diversos corredores longos e em um dado determinado instante o robô escolhe um corredor C. A relaxação local pode perfeitamente ser utilizada, pois temos na borda da janela local regiões desconhecidas que irão atrair o robô até o final do corredor C. Considere agora que o robô chegou no final do corredor C e descobre que ele não possui saída. Neste caso, a relaxação local se torna inútil pois não existe dentro da janela local ou em sua borda regiões desconhecidas que possam atrair o robô. Isto fará com que o robô fique preso em uma espécie de mínimo local.

Este mínimo local é criado da seguinte maneira. Considere que o robô está em um corredor sem saída que é paralelo ao seu eixo X, conseqüentemente, o robô pode se mover apenas para a direita ou para a esquerda. Quando o robô encontra o final do corredor ele não terá mais o auxílio das regiões desconhecidas para continuar sua exploração, sendo assim, ele será atraído para a região de mais baixo potencial.

Como ele pode se mover apenas para a direita ou para a esquerda, se a região de mais baixo potencial estiver a sua esquerda, ele se moverá para a esquerda. Isto fará com que as células a sua esquerda no instante t sejam atualizada, aumentando seu valor de potencial, de maneira que as células a sua direita sejam consideradas como possuindo potencial menor que as células à esquerda em um instante $t+1$, por não terem sido atualizadas no instante t. O efeito resultante é um comportamente oscilatório do robô da esquerda para a direita.

[7]É importante destacar que na maior parte do tempo isto é verdade, a não ser no início da exploração onde a quantidade de células da janela local é aproximadamente igual à quantidade de células na janela global. Com o progresso da exploração a diferença entre ambas vai aumentando.

Dessa forma, faz-se necessário um chaveamento da janela local para a janela global a fim de que o robô seja atraído para uma região desconhecida mais próxima. Este chaveamento é feito verificando se existe ao lado de uma célula *ij*, dentro de sua janela local, com s_{ij}=*espaço livre* e atributo região maior que 0 ao lado de uma célula *mn* com s_{mn}=*não explorado*. Se esta condição é verdadeira, o robô utiliza apenas informações em sua janela local, caso contrário usa informações oriundas da janela global. Iremos a partir deste ponto chamar este chaveamento de conhecimento adaptativo. Na Seção 5.4.9 apresentamos alguns resultados de experimentos onde comparamos o desempenho do processo exploratório usando o conhecimento global e o conhecimento adaptativo.

5 Experimentos

Este capítulo apresenta os resultados obtidos utilizando nosso agente exploratório em uma série de experimentos onde diferentes aspectos do problema de exploração e mapeamento são ilustrados e discutidos. Ele é dividido da seguinte maneira. Na Seção 5.1 apresentamos o robô Nomad 200. Na Seção 5.2 mostramos a interface desenvolvida para facilitar a interação do usuário com o o robô. Na Seção 5.3 descrevemos através de um algoritmo o funcionamento do nosso agente exploratório. Na Seção 5.4 apresentamos alguns experimentos de exploração em ambiente internos *densos*, tanto com o robô simulado quanto com o robô Nomad; realizamos comparações com as estratégias aleatória e *wall following*; analisamos o desempenho da nossa estratégia face aos seus parâmetros; analisamos o tempo de exploração e os erros de odometria produzidos; realizamos experimentos em ambientes com obstáculos dinâmicos; mostramos uma estratégia para aumentar a eficiência do processo de relaxação; apresentamos resultados preliminares da utilização da janela adaptativa no processo de construção de mapas e finalmente apresentamos um experimento relacionado à exploração orientada a um objetivo. Finalmente na Seção 5.5 apresentamos alguns experimentos em ambientes esparsos realizados com diferentes regras de atualização do potencial, comparamos os resultados obtidos com a utilização destas regras, e mostramos alguns resultados utilizando o robô Nomad.

5.1 Robô Nomad

O robô NOMAD 200, ilustrado na Figura 5.1, é um sistema integrado com 5 módulos sensoriais: tátil, infravermelho, ultrasom, visão e bússola. Além disso, ele possui uma base constituída de 3 rodas, as quais estão alinhadas e podem transladar e rotacionar sincronamente permitindo obter uma rotação de 360^0 sobre seu próprio eixo. Este robô é fabricado pela empresa Nomadic e está disponível em nosso Instituto através do Laboratório de Robótica Inteligente (LRI).

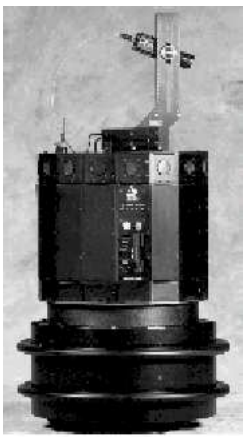


FIGURA 5.1 – Robô NOMAD 200

Dentre os módulos supracitados, utilizamos somente o sistema sensorial ultrasom (sonar), o qual é composto de 16 canais que fornecem informação no alcance

de 43 cm a 6,5 m com 1% de precisão resultando em uma cobertura de 360^0 do ambiente. Embora tenha bons alcance e cobertura, este tipo de sensor possui três principais problemas [BOR 89]:

- **pobre direcionalidade**: embora, estes sensores forneçam bom alcance, eles são incapazes de informar a direção do objeto identificado;
- **passíveis a ruídos**: a leitura de um sensor sofre influência de ruídos externos e de reflexões perdidas dos seus sensores vizinhos (*crosstalk*);
- **reflexão especular**: a leitura dos sensores sofre influência do ângulo de incidência do sinal sonoro nos objetos identificados, da estrutura da superfície de incidência, da refletividade e da distância do objeto em relação ao sensor. Isto acarreta uma falsa identificação, e em algumas ocasiões, na total ausência dela.

Apesar destes problemas, este tipo de sensor é amplamente utilizado em robôs móveis, na maioria das vezes, combinado com sensores do tipo infravermelho ou laser. As principais razões para isto são seu baixo custo e por serem fáceis de operar. Entretanto como visto, possuem sérios problemas que limitam sua utilização direta em aplicações de mapeamento de ambiente [RAS 90].

Os problemas relacionados à pobre direcionalidade e a reflexão especular são tratados através da função de mapeamento descrita na Seção 4.2.4. Esta função define uma distribuição de incerteza baseada em freqüência que facilita a localização dos obstáculos encontrados no ambiente.

O problema *crosstalk* é minimizado controlando o disparo dos sensores. Borenstein [BOR 89] propõe realizar o disparo de grupos de 4 sensores perpendiculares em intervalos de tempo constantes. Dessa maneira, considerando que o sonar consegue detectar qualquer objeto a uma distância de 6,47 m, o tempo gasto entre o disparo e a recepção do sinal sonoro é igual a $t = 2 * 6,5 \ / \ v_{som} = 38,2 \ ms$, onde $v_{som} = 340m/s$ corresponde a velocidade do som, e o fator 2 corresponde ao trajeto de ida e volta da onda sonora. No caso de serem disparados 4 grupos de 4 sensores, o tempo máximo necessário para a cobertura de 360^0 do ambiente é $4t = 152,8ms$.

É importante observar que a aquisição dos sinais sonoros é feita com o robô em movimento. Isto gera um mapeamento atrasado em ms, devido, a posição do robô no momento do disparo e no momento da recepção do sinal serem diferentes. Não estamos tratando este atraso, pois o erro resultante na localização do objeto é muito menor que as células que discretizam o ambiente.

5.2 Interface com o usuário

Para facilitar a interação do usuário com o robô desenvolvemos uma interface visual em C++/Linux, ilustrada na Figura 5.2. Esta interface permite:

- realizar a comunicação com o robô. Ela permite a comunicação direta com o robô através do envio/recepção de mensagens utilizando TCP/IP;
- especificar a tarefa a ser realizada, a qual pode ser: planejamento de caminhos em um ambiente conhecido; exploração e aquisição de mapas de ambientes

desconhecidos e a localização de objetos[1];

- ajustar os paramêtros do sistema. É possível definir a EDP que gerará o campo potencial que irá guiar o robô, definir o tamanho máximo do mapa do ambiente, definir a velocidade do robô, etc;
- visualizar o progresso do robô na realização de uma determinada tarefa, visualizar os mapas construídos, etc.

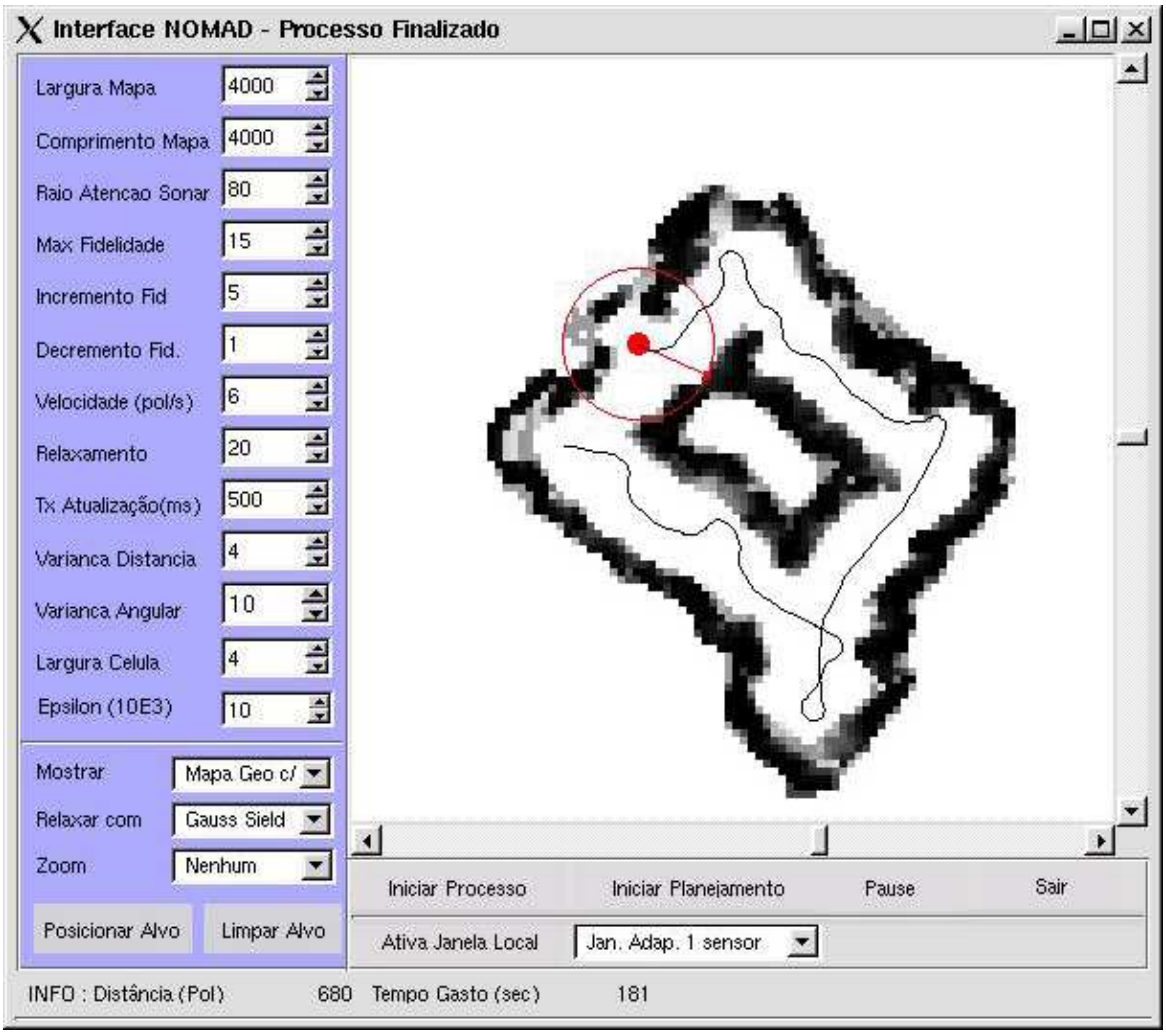


FIGURA 5.2 – Interface visual desenvolvida em C++/Linux

[1] A localização de objetos foi implementada apenas em simulações. Esta tarefa corresponde a descobrir a localização de um objeto em um ambiente desconhecido previamente(mais detalhes ver a Seção 5.4.7).

5.3 Descrição Algorítmica do Processo Exploratório

No capítulo anterior apresentamos a arquitetura do nosso agente exploratório e seus principais módulos. Entretanto, não descrevemos em baixo nível como os módulos interagem gerando o comportamento exploratório. Sabemos que a arquitetura *blackboard* possui como característica a execução assíncrona de seus especialistas. Portanto, para fins didáticos podemos impor uma ordem de execução dos módulos, o que facilita a visualização da interação do sistema como um todo. Assim, esta seção tem como objetivo descrever o processo exploratório sobre a perspectiva de um algoritmo seqüencial, dando ênfase principalmente à implementação do módulo de planejamento.

Quando a exploração inicia o robô tem sua posição (x, y) modificada para $(0, 0)$, o que corresponde ao centro do mapa bidimensional. Além disso, os atributos de todas as células no mapa são modificados da seguinte maneira: estado é atualizado com o valor *não explorado*, enquanto que certeza, potencial e região são atualizados com os valores $0, 0, 5$, respectivamente.

Durante o processo de exploração o robô executa o seguinte algoritmo:

1. ativa e obtém a leitura dos sensores sonar;
2. realiza a atualização local do mapa baseada na informação contida na janela de ativação;
3. atualiza o atributo *potencial* das células da região potencial.
4. calcula o vetor gradiente descendente a partir da sua posição corrente no mapa;
5. se desloca seguindo a direção definida por este gradiente;
6. repete-se os passos de 1 a 5 até que todo o ambiente seja completamente explorado, ou seja, não exista nenhuma célula classificada como *espaço livre* ao lado de uma célula classificada como *não explorada* (ver Seção 4.2.7).

O robô aciona seus sensores para obter informações locais sobre o ambiente. A informação adquirida por cada sensor i é expressa em coordenadas polares (ver Figura 4.4), na forma de um escalar d_i que representa a distância entre o sensor e o objeto mais próximo em seu campo de visão, e um ângulo α_i que determina a orientação do sensor em relação ao robô. Este ângulo é descrito em relação ao eixo x do sistema de coordenadas do robô.

Utilizando estas informações juntamente com o conhecimento sobre sua posição e orientação, o robô atualiza os valores certeza, estado e potencial de todas células sobre o cone de visão (ver Figura 4.4) de cada sensor. É importante destacar que as leituras fora deste cone são descartadas.

Após a atualização do mapa, a propriedade potencial de todas as células da *região potencial* é atualizada. Inicialmente mostraremos os resultados obtidos com a utilização das equações 4.6 e 4.9 apresentadas na Seção 4.2.6. Estas equações correspondem a utilização da equação 3.2, com $F(\mathbf{r}) = 0, \ \forall \ \mathbf{r}$. A atualização do potencial utilizando as equações 4.12 e 4.14 será apresentada adequadamente na Seção 5.5.

Uma simples atualização do campo potencial pode consistir na atualização de todas células dentro da região potencial ou apenas das células dentro da janela de

ativação (veja Seção 4.3.2 para mais detalhes sobre o uso do conhecimento global ou adaptativo). O procedimento é repetido diversas vezes antes do movimento do robô. Quando as funções harmônicas são utilizadas no planejamento de caminho, o potencial é atualizado até sua estabilização dentro de uma precisão predefinida $\epsilon < 10^{-p}$. Somente então, o robô é colocado em movimento [CON 93, CON 94]. Isto pode levar a um número considerável de iterações, geralmente da ordem de pM para Gauss-Seidel e $p\sqrt{M}$ para SOR, onde M é o número de células a serem atualizadas[PRE 92]. Isto é feito principalmente porque eles objetivam computar um caminho suave e estável entre duas posições quaisquer baseado em um mapa estático global do ambiente conhecido previamente.

Em nosso caso, o mapa do ambiente é desconhecido e deve ser construído gradativamente. Nestas circunstâncias, esperar a relaxação completa do potencial seria muito oneroso para o desempenho do sistema. Felizmente, nós descobrimos que a relaxação completa do potencial não é um requisito para a descoberta de caminhos bons e suaves [PRE 2002] (ver Seção 5.4.3). Sendo assim, este relaxamento não é mais condição necessária para que o robô comece a se movimentar pelo ambiente.

Para implementar esta idéia, trabalhamos com uma taxa de iteração fixa r. No simulador, como a inércia não é prejudicial ao comportamento do robô, nós paramos o robô durante o processo de atualização do potencial e em seguida, o colocamos em movimento. Neste caso, a taxa de atualização r é melhor expressa em *iterações/passo*. Por outro lado, no robô real, o processo de leitura dos sensores, atualização do mapa é realizado em uma freqüência de tempo fixa, assim, a taxa de atualização r é expressa em *iterações/segundo*. Em seguida, o gradiente descendente é calculado e o robô se move ajustando a direção de seu movimento ao ângulo ϕ definido pela equação 4.15.

Nos experimentos de simulação, o procedimento acima é calculado alternadamente com o movimento do robô, ou seja, após o cálculo do gradiente, o robô é deslocado a um passo fixo Δd. Em contrapartida, nos experimentos realizados no robô Nomad, devido a sua inércia, este procedimento é calculado simultaneamente com o movimento do robô. Devido ao robô estar sempre em movimento, ele executa o procedimento acima, e em seguida ajusta a direção do seu movimento usando a equação 4.16, e a sua velocidade de acordo com a equação 4.17.

5.4 Experimentos de exploração em ambientes internos *densos*

Esta seção apresenta os experimentos de exploração realizados pelo nosso agente[2] exploratório utilizando a equação de Laplace[3] em ambientes internos *densos*[4] compostos por diversos obstáculos formando labirintos. Estes experimentos são

[2]Utilizaremos os termos agente e robô como sinônimos durante todo o texto.

[3]A equação de Laplace é a variação fundamental do PVC estudado, e se aplica, principalmente, à exploração de ambientes internos densos, como será visto na Seção 5.5.1.

[4]Este tipo de ambiente tem estrutura similar àquela encontrada em escritórios. Ele é composto por paredes ortogonais entre si, móveis, chão nivelado, etc. Aqui classificamos os ambientes internos como densos e esparsos. Os ambientes internos densos são caracterizados por possuírem uma distribuíção espacial de objetos no ambiente que faz com que sempre exista no mínimo um obstáculo dentro do campo de visão dos sensores do robô durante seu deslocamento, ao contrário dos ambientes internos *esparsos*.

realizados tanto no simulador do robô quanto no robô Nomad para testar a robustez do algoritmo em situações reais.

Além disso, esta seção apresenta comparações realizadas entre a nossa abordagem e as técnicas de exploração padrão, alguns resultados comparativos realizados entre os 2 métodos de atualização para a função potencial, apresentados anteriormente utilizando o simulador do Nomad, e a influência da taxa de iteração r em seus desempenhos. Ademais, esta seção mostra análises relacionadas ao tempo de exploração e os erros de odometria produzidos, assim como experimentos em ambientes com obstáculos dinâmicos, experimentos usando uma janela de tamanho adaptativo e uma exploração orientada a objetivo, onde o robô deve explorar o ambiente para localizar um objeto. Finalizando é apresentada uma estratégia para aumentar a eficiência do processo de relaxação.

5.4.1 Explorando ambientes internos *densos*

Esta seção apresenta alguns experimentos realizados no simulador do Nomad. A Figura 5.3 mostra o progresso de um experimento de exploração realizado pelo nosso agente. Neste caso, o robô está inserido em um ambiente retangular de $10,2\ m\ \times\ 8,4\ m$ com diversos obstáculos formando a estrutura de um labirinto. Seu mapa interno é representado usando uma grade bidimensional de $80\ \times\ 80$ células, onde cada célula é um quadrado de $15,2\ cm\ \times\ 15,2\ cm$. Conseqüentemente, este mapa pode representar no máximo uma área de $12,2\ m\ \times\ 12,2\ m$, o que é adequado para esta simulação em particular.

Neste experimento, a janela de ativação foi mantida fixa com raio igual a $2\ m$. O passo do robô corresponde ao deslocamento $\Delta d = 7,6\ cm$ e o campo potencial é atualizado em toda a região explorada (*globalmente*) a uma taxa $r = 30$ iterações/passo com atualização por Gauss-Seidel.

Para cada instante de simulação na Figura 5.3, a Figura 5.4 mostra o mapa parcial correspondente juntamente com os obstáculos detectados e o campo vetorial que guia o robô em sua exploração. Observe que na vizinhança do robô, o campo vetorial aponta para a região mais próxima ainda não explorada. É importante notar que diversos mínimos locais aparecem no campo vetorial decorrente do fato de que o potencial não relaxou globalmente à função harmônica com 30 iterações/passo.

Algumas dificuldades surgem quando a região não explorada está muito longe da posição atual do robô, em geral, quando o robô encontra um beco sem saída. Neste caso, o potencial necessita ser completamente relaxado para indicar caminhos ótimos, ocasionando um decaimento no desempenho. Isto pode ser observado no canto superior direito da Figura 5.3 (d) onde a trajetória do robô oscila. Tal comportamento não é visível nas Figuras 5.3 (a), (b) e (c), pois nestes casos o robô está relativamente próximo às regiões não exploradas. Uma possível solução a este problema é considerar uma taxa de atualização dinâmica que aumenta quando a velocidade média do robô atingir um limiar inferior predefinido.

A Figura 5.5 mostra uma seqüência exploratória para um ambiente simulado com obstáculos poligonais e paredes com orientações aleatórias. As dimensões do ambiente e os parâmetros utilizados são os mesmos da Figura 5.3. Isto demonstra a robustez do método quando utilizado em ambientes mais genéricos. Algumas abordagens na literatura têm bom desempenho apenas em certas configurações de obstáculos e formas [THR 96, KUI 91, KUI 88, LEE 96]. Aqui esta limitação não é encontrada.

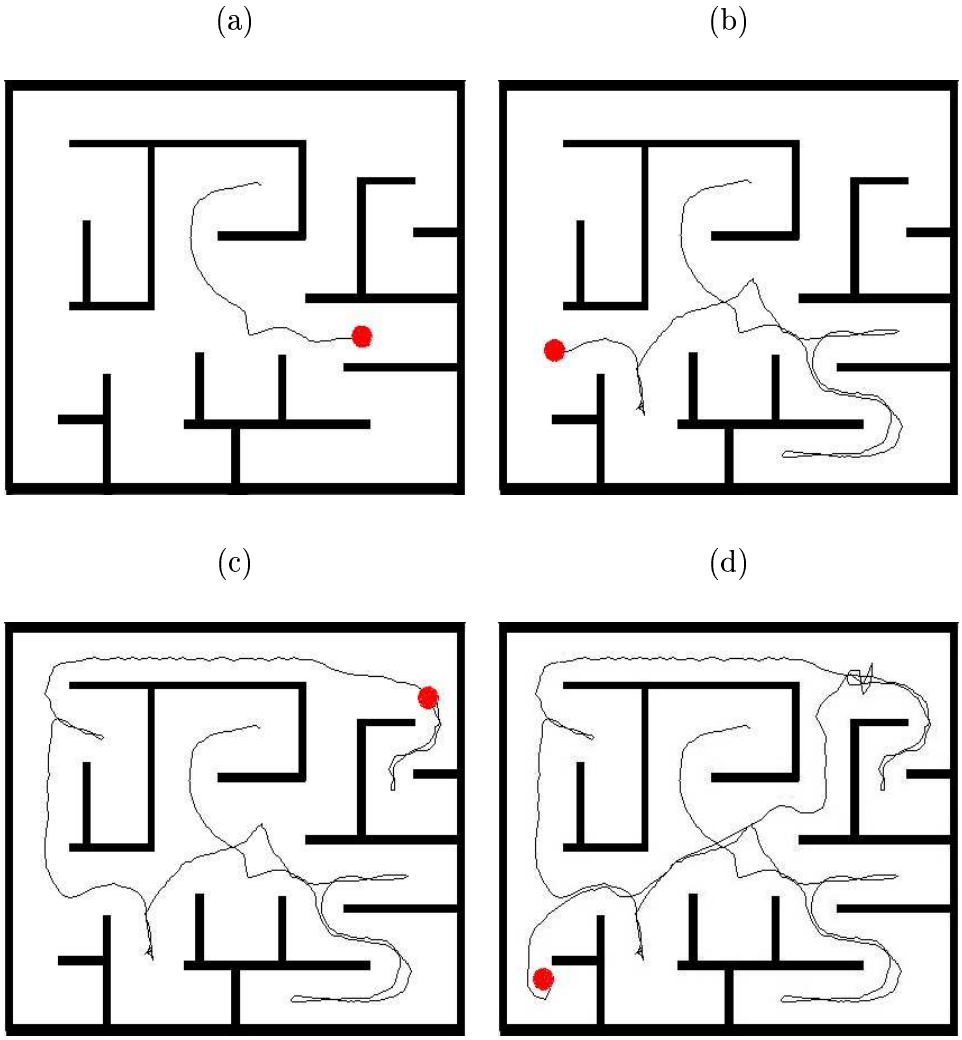


FIGURA 5.3 – Seqüência de exploração realizada pelo simulador do Nomad: A trajetória seguida pelo robô. O ambiente tem $10,2\ m\ \times\ 8,4\ m$, e o robô se desloca com passo constante $\Delta d\ =\ 7,6\ cm$. O campo potencial é atualizado usando Gauss-Seidel em um taxa $r\ =\ 30$ iterações/passo.

A Figura 5.6 mostra um experimento onde o robô Nomad 200 explora um ambiente *interno* em nosso Instituto. Este ambiente consiste em uma sala, dividida por duas paredes de compensado, conectada a um corredor em forma de **L**. A sala possui mesas e cadeiras situadas ao longo de suas paredes. A fim de mapear o ambiente, utilizamos um mapa de $100\ \times\ 100$ células, onde cada célula corresponde a uma região de $15,2\ cm\ \times 15,2\ cm$. Para explorar todo o ambiente, o robô percorreu um caminho de comprimento igual a $L = 36,6\ m$ em aproximadamente $5\ min$, com uma velocidade média igual a $12,7\ cm/s$. Sua janela de atualização é mostrada na Figura 5.6 e tem raio igual a $2\ m$. Além disso, todas as células do espaço explorado foram atualizadas durante o processo exploratório.

No caso das Figuras 5.3, 5.5 e 5.6, como um alvo não estava presente, o campo potencial foi descartado no final da exploração. Neste caso, após um número suficiente de atualizações o gradiente é nulo em todos os pontos do espaço livre. Portanto, a saída do processo exploratório é unicamente o mapa geométrico que

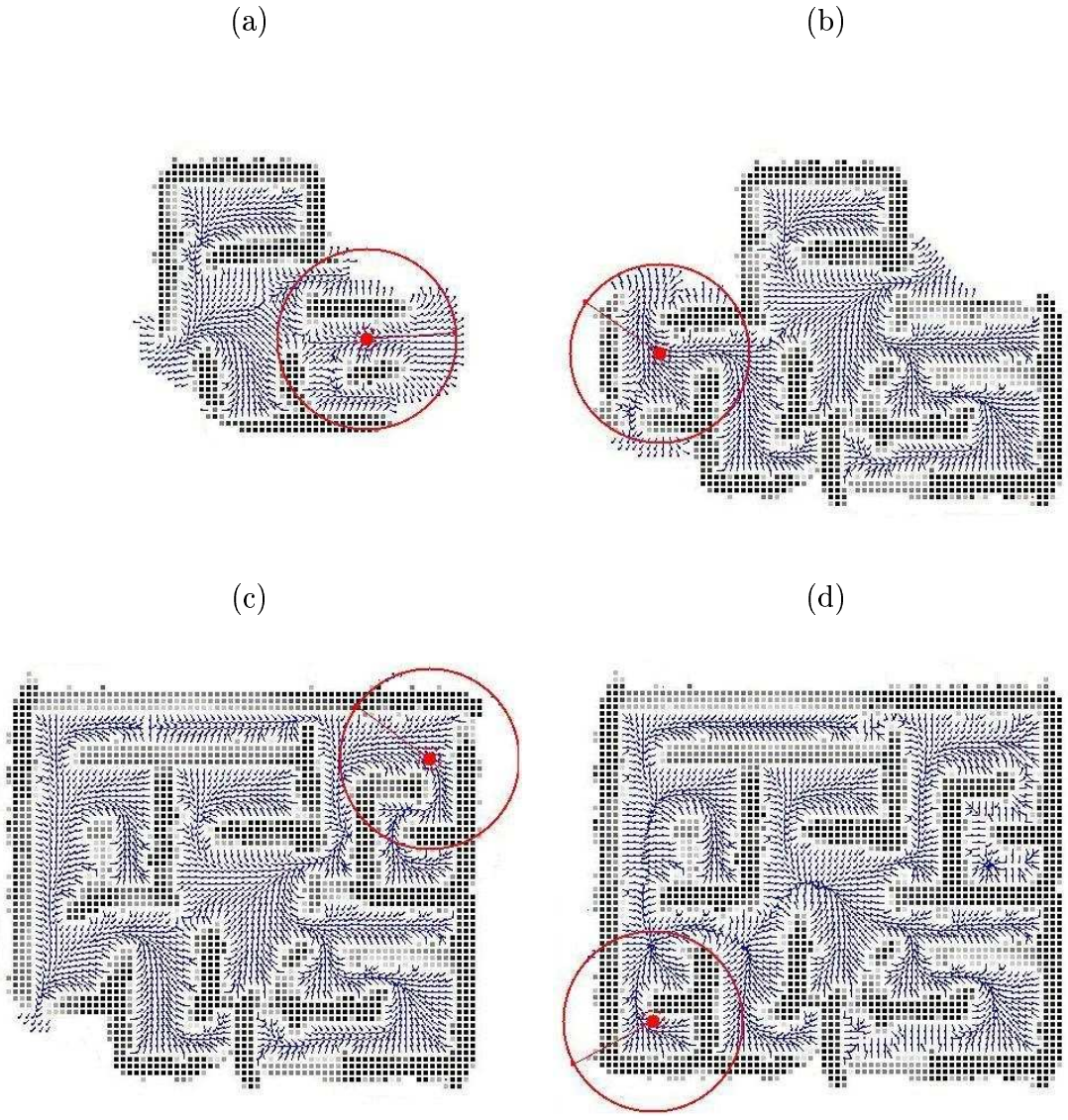


FIGURA 5.4 – Seqüência de exploração realizada pelo simulador do Nomad: Instâncias parciais do campo vetorial e do mapa durante a exploração. O campo vetorial representa o gradiente descendente em cada célula do espaço livre e é normalizado para facilitar sua visualização. No mapa, o valor de certeza das células é representado em níveis cinza, onde branco corresponde ao valor 0 o preto ao valor 15. O tamanho do mapa é 80×80 células, onde cada célula corresponde a $15,2\ cm \times 15,2\ cm$ da região do espaço real.

mostra a localização de todos os obstáculos através de seu valor de certeza na grade, $c(\mathbf{r})$ para $\mathbf{r} = (i, j)$, $i \in [1, N_x]$, e $j \in [1, N_y]$.

5.4.2 Comparando com outras estratégias

A fim de calcular o desempenho da nossa estratégia com os algoritmos de exploração padrão, nós consideramos um modelo muito simples onde o ambiente é um quadrado $L \times L$ dividido em $N \times N$ células que estão ocupadas ou não, e o robô é um ponto que se move no espaço contínuo. Durante seu movimento ele

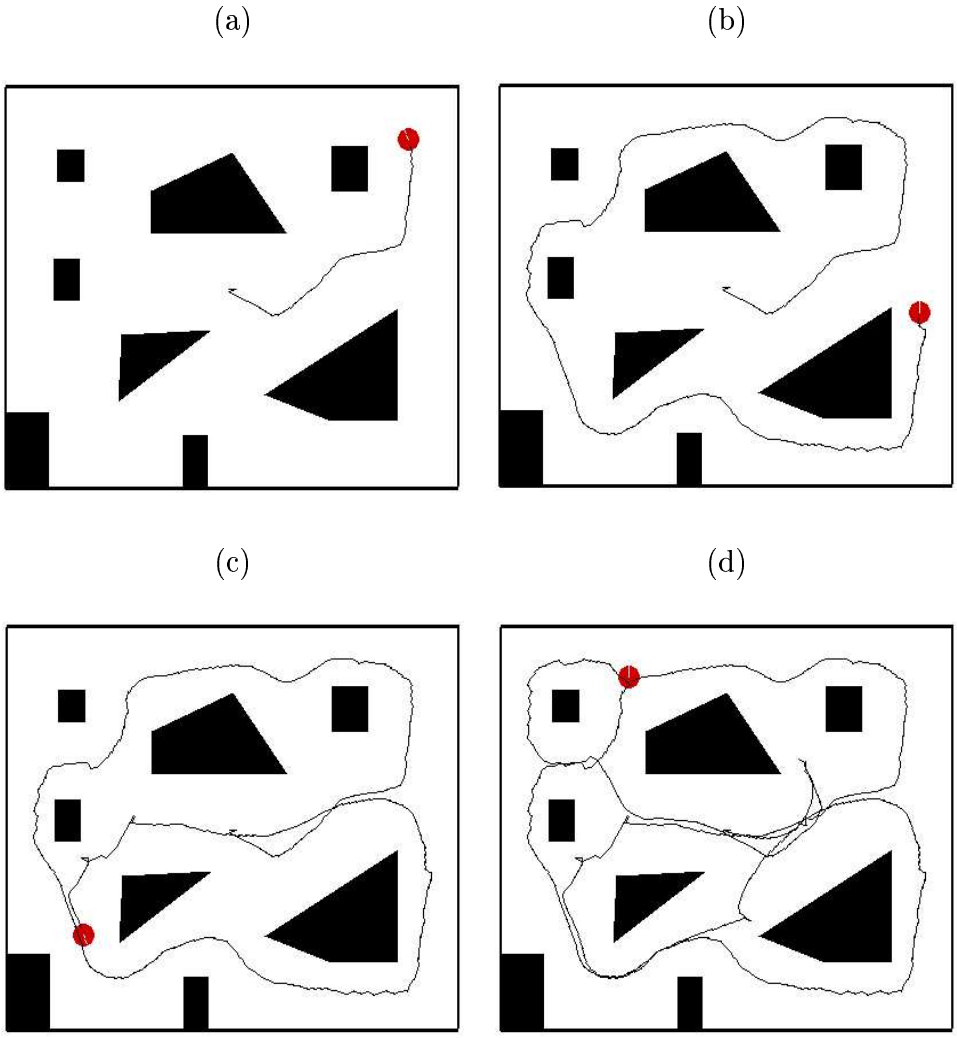


FIGURA 5.5 – Seqüência de exploração realizada pelo simulador do Nomad: Ambiente poligonal. O ambiente é $10,2\ m\ \times\ 8,4\ m$, e o deslocamento do robô corresponde ao passo $\Delta d = 7,6\ cm$. O campo potencial é atualizado usando Gauss-Seidel com uma taxa r = 30 iterações/passo.

explora uma região circular de diâmetro **A**. Neste modelo, não são considerados erros relacionados à detecção dos obstáculos e quando uma célula está dentro da janela de ativação e está ocupada por um obstáculo, ela tem seu valor de certeza atualizado com o valor máximo permitido.

Consideramos que o desempenho de um algoritmo é dado pelo comprimento total do caminho (ℓ) seguido pelo robô durante sua exploração. É importante destacar que esta medida mistura dois diferentes aspectos: A qualidade do caminho (*sua suavidade*) e a estratégia de cobertura do ambiente (*a seqüência de exploração seguida pelo robô*). Apesar disso, decidimos não fazer distinção entre elas.

Os algoritmos utilizados são:

- *exploração aleatória.* O robô se desloca aleatoriamente no ambiente de acordo com

$$x_{t+1} \quad = \quad x_t + \Delta d\ \cos(\theta)$$

(a) (b)

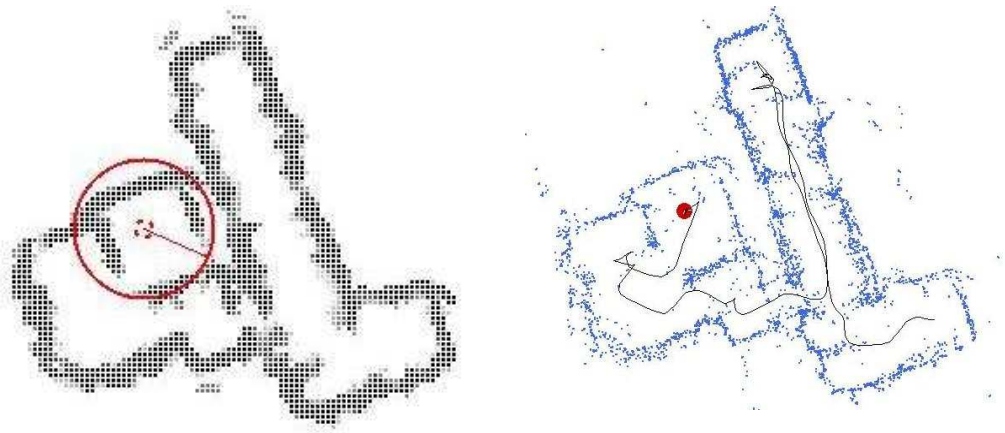


FIGURA 5.6 – Exploração de um ambiente *interno* pelo robô Nomad 200. a) Mapa obtido após a exploração. O tamanho do mapa é 100×100 células, onde cada célula corresponde a uma região quadrada de $15,2\ cm \times 15,2\ cm$. A janela de ativação possui raio igual a $2\ m$. b) A trajetória seguida durante a exploração. Seu comprimento total é $L = 36,6\ m$ e foi navegada com velocidade média igual a $12,7\ cm/s$. As leituras do sonar são também mostradas. Observe que elas são mais ruidosas que o mapa. Isto se deve ao fato de que as leituras fora da janela de ativação são descartadas e somente as leituras consistentes atualizam o atributo certeza das células.

$$y_{t+1} \quad = \quad y_t + \Delta d \ \sin(\theta) \tag{5.1}$$

onde (x, y) corresponde à posição do robô e θ sua direção, a qual é escolhida aleatoriamente quando sua distância em relação a um obstáculo é menor que uma distância mínima predefinida d_{min}. Isto prova ser mais eficiente que uma trajetória puramente aleatória onde o robô escolhe uma nova direção a cada passo.

- *wall following ideal.* O robô se desloca seguindo um caminho paralelo às paredes a uma certa distância. Aqui, simplesmente medimos o comprimento total da trajetória gerada pelo *wall following* necessária para explorar completamente o ambiente. Isto não é uma implementação realística do algoritmo *wall following*, entretanto é possível obter uma estimativa geométrica da ordem de magnitude do procedimento.

A Tabela 5.1 mostra os resultados médios $\bar{\ell}$ e o desvio padrão σ_ℓ do comprimento do caminho total necessário para a exploração completa do ambiente na Figura 5.7. Este ambiente foi discretizado em um mapa de 100×100 células. Para um melhor comparativo com os resultados do simulador do Nomad, nós apresentamos o resultado na escala definida pelo tamanho do ambiente (L). Para cada método foram realizados 30 execuções com o robô iniciando de posições aleatórias. No caso da exploração aleatória, o robô não pode chegar mais próximo que $d_{min} = 0,5A$ das paredes. Enquanto que para o *wall following* o robô segue as paredes a uma

distância $d_{min}/L = 0,05$ para $A/L = 0,1$ e $d_{min}/L = 0,08$ para $A/L = 0,4$. Nas funções harmônicas, foi usado Gauss-Seidel com taxa $r = 30$ iterações/passo.

TABELA 5.1 – Resultado da simulação para a exploração aleatória, *wall following* e as funções harmônicas no ambiente da Figura 5.7 sobre 30 execuções.

Método	A/L	$\bar{\ell}/L$	σ_ℓ/L
Exploração Aleatória	0,1	150,86	57,60
	0,4	48,41	27,82
Wall following	0,1	8,31	0,25
	0,4	6,69	0,69
Funções Harmônicas.	0.1	21,50	2,40
	0,4	6,15	0,42

Os resultados indicam que o método baseado em funções harmônicas tem desempenho comparável ao procedimento *wall following* ideal quando o alcance sensorial é grande o suficiente. Para $A/L = 0,1$ a comparação não é apropriada pois o ambiente não é completamente explorado seguindo as paredes a uma distância fixa. É importante observar que, mesmo que o procedimento *wall following* ideal seja competitivo com o nosso, o procedimento real poderia não ser. A razão é que em geral este procedimento tem que lidar com paredes não conectadas. Por isso, um método adicional de busca tem que ser introduzido o que pode causar um decaimento no desempenho global do algoritmo.

5.4.3 Analisando os métodos de relaxação e a taxa de atualização

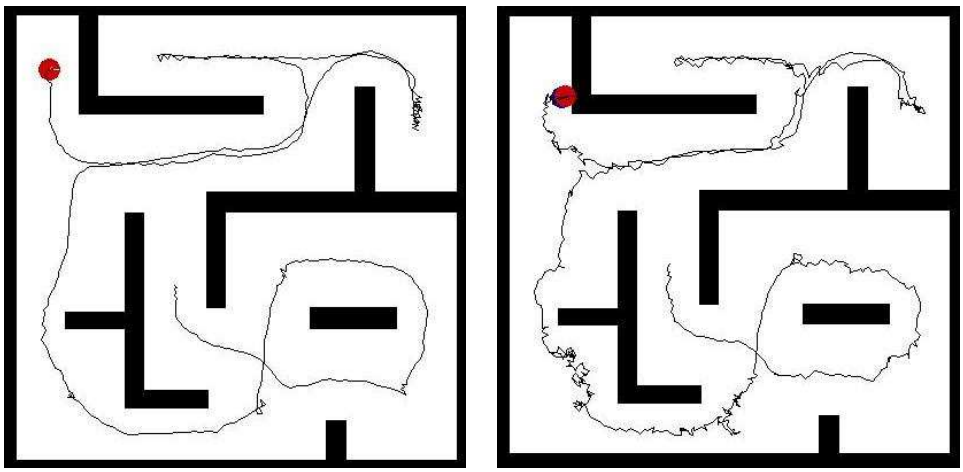
As Figuras 5.7(a) e (b) mostram exemplos da trajetória seguida pelo robô no ambiente de simulação quando o campo potencial é atualizado usando os métodos Gauss-Seidel e SOR, respectivamente, com uma taxa $r = 10$ iterações/passo. Com Gauss-Seidel, o robô finaliza a exploração do ambiente com $\ell = 59,8\ m$ enquanto que o SOR faz o robô colidir com uma parede.

Se incrementarmos a taxa de atualização para $r = 30$ iterações/passo obteremos o resultado mostrado nas Figuras 5.7(c) e (d). Com esta taxa, a atualização SOR não falha e o robô consegue explorar completamente o ambiente. Entretanto, Gauss-Seidel é ainda melhor gerando um caminho com comprimento $\ell = 59,3\ m$ contra um caminho de comprimento $\ell = 61,4\ m$ gerado pelo SOR.

A Tabela 5.2 apresenta o desempenho médio do método usando as funções harmônicas para a exploração do ambiente da Figura 5.7 sobre 10 execuções. Em cada execução, o robô foi colocado em posição e orientação diferentes. O desvio padrão relativamente pequeno dá uma medida de confiabilidade do algoritmo. Compare-o, por exemplo, ao desvio padrão da exploração aleatória na Tabela 5.1.

A Figura 5.8 mostra uma instância de um caminho exploratório em um labirinto constituído de 3 salas realizados pelo robô Nomad usando Gauss-Seidel com taxa $r = 20$ iterações/s. A Tabela 5.3 mostra o resultado de 4 experimentos no mesmo ambiente com o robô iniciando de diferentes posições.

(a) (b)



(c) (d)

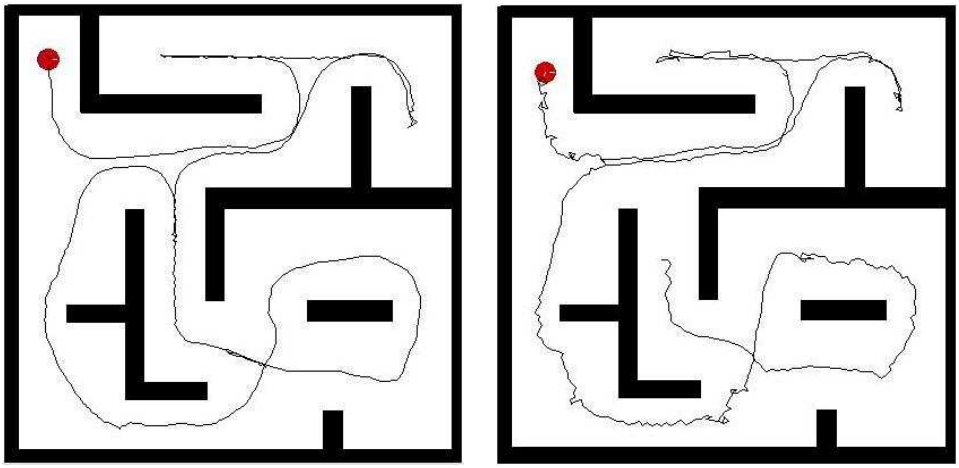


FIGURA 5.7 – Seqüência de Exploração.a) Trajetória seguida usando o método Gauss-Seidel para a taxa $r = 10$ iterações/passo. b) Trajetória seguida usando o método SOR para a taxa $r = 10$ iterações/passo. c) Trajetória seguida usando o método Gauss-Seidel para a taxa $r = 30$ iterações/passo. d) Trajetória seguida usando o método SOR para a taxa $r = 30$ iterações/passo. O ambiente é $10,2\ m$ por $10,2\ m$, e o robô se move em passos de $\Delta d = 7,6\ cm$. O tamanho do mapa é 80×80 células, onde cada célula corresponde a uma região quadrada de $15,2\ cm \times 15,2\ cm$ do espaço real. A janela de ativação possui raio igual a $2\ m$.

5.4.4 Analisando o tempo de exploração

A fim de complementar os dados obtidos nas fases anteriores, apresentamos alguns resultados relativos ao tempo médio de exploração. Todos os experimentos ilustrados nesta seção foram realizados diretamente no robô NOMAD 200. A Figura 5.9 ilustra uma exploração típica em nosso Instituto. A Figura 5.9 (a) mostra o mapa do ambiente e a trajetória seguida pelo robô. A Figura 5.9 (b) mostra o robô

TABELA 5.2 – Resultado da simulação utilizando o simulador do Nomad no ambiente da Figura 5.7. As dimensões do ambiente são $10,2\ m \times 10,2\ m$, a janela de ativação tem raio igual a $2\ m$, e o desempenho médio foi extraído a partir de 10 execuções.

Método Relax.	Taxa iter.	$\bar{\ell}/L$	σ_ℓ/L
Gauss-Seidel	10	6,35	0,50
	30	5,60	0,34
SOR	10	4 colisões em 10 exec.	-
	30	6,53	0,90

TABELA 5.3 – Resultado médio extraído de 4 experimentos no labirinto da Figura 5.8 com o robô Nomad. As dimensões do ambiente são $3,5\ m \times\ 7,8\ m$. A janela de ativação tem raio igual a $2m$, e a taxa $r\ =\ 20$ iterações/s.

Método Relax.	Taxa iter.	$\bar{\ell}$	σ_ℓ
Gauss-Seidel	20	13,7 m	62 cm

(a) (b)

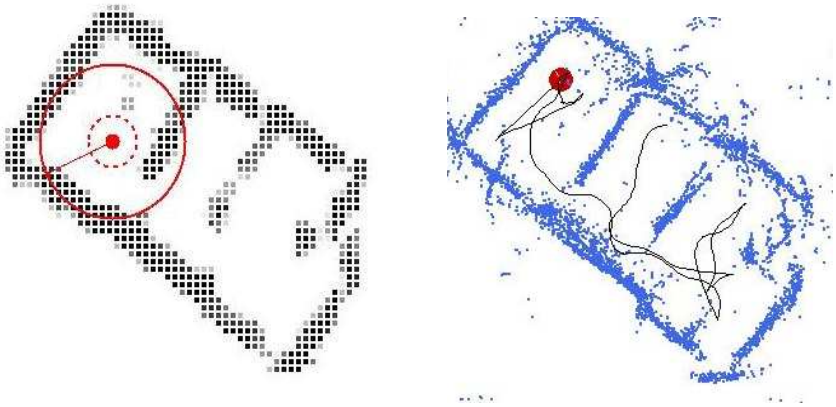


FIGURA 5.8 – Experimento com o robô Nomad 200 em um labirinto. As dimensões do ambiente são aproximadamente $3,5\ m \times\ 7,8\ m$. A janela de ativação tem raio igual a $2\ m$, e a taxa de iteração é igual a $r\ =\ 20$ iterações/s usando o método Gauss-Seidel.

retornando ao seu ponto de partida após o término da exploração.

O ambiente de teste possui dimensões aproximadas de $13\ m \times 7\ m$. Cada célula do mapa representa uma região quadrada de $10,1\ cm \times 10,1\ cm$ do espaço real. Neste ambiente, o robô se deslocou com velocidade máxima igual a $25,4\ cm/s$ com uma janela de ativação de raio igual a $2\ m$.

O robô partiu de posições diferentes com orientações aleatórias. O tempo de exploração foi medido utilizando o relógio interno do computador. A Tabela 5.4 mostra o tempo médio ($\bar{\tau}$) e o comprimento médio do caminho de exploração ($\bar{\ell}$) em

(a) (b)

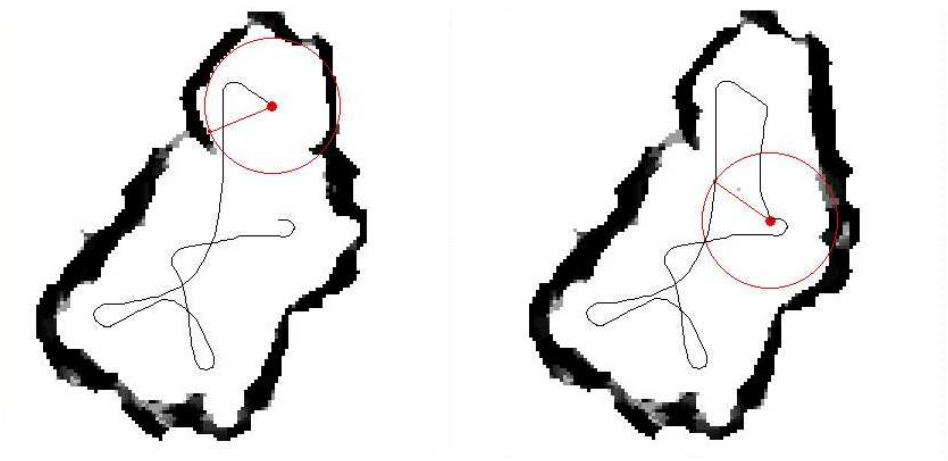


FIGURA 5.9 – Experimento de exploração com o robô NOMAD em nosso Instituto. a) Mapa do ambiente e a trajetória seguida pelo robô NOMAD. b) Retorno do robô NOMAD ao seu ponto de partida

um conjunto de 6 experimentos

TABELA 5.4 – Tempo médio extraído de 6 experimentos realizados com o robô Nomad em nosso Instituto.

$\bar{\tau}$	σ_τ	ℓ	σ_ℓ
2min e 29sec	25sec	$26,7m$	3,56m

É importante observar que os dados adquiridos acima são dependentes do comprimento do raio de ativação e da velocidade máxima do robô. Quando maior for o raio de ativação maior será a região explorada pelo robô a cada instante, conseqüentemente, mais rápido será o mapeamento do ambiente. Similarmente, quanto mais rápido o robô se deslocar mais rápido ele irá mapear completamente o ambiente, entretanto, é importante observar que a velocidade deve ser tratada com cautela, pois o rápido deslocamento exige uma atenção maior a eventos inesperados.

5.4.5 Analisando o erro de odometria

Além do tempo gasto pelo robô durante o processo de exploração, medimos o erro de odometria inserido em seus trajetos durante seu deslocamento pelo ambiente. Como discutimos inicialmente na Seção 4.2.2, os erros de odometria não são expressivos pois o robô se move com uma velocidade baixa em um ambiente pequeno.

Para comprovar isto, elaboramos um conjunto de 5 experimentos relacionados à exploração. Os experimentos foram realizados da seguinte maneira. Definimos uma marca no chão para indicar a posição inicial do robô. Esta marca era alinhada com o centro do robô e correspondia a posição $(0, 0)$ no mapa. Em seguida, faziamos o robô explorar um ambiente com a mesma estrutura daquele ilustrado na Figura

5.9. Ao final da exploração, fazíamos o robô retornar ao seu ponto de partida, ou seja, a posição $(0, 0)$.

Utilizamos a mesma configuração apresentada na seção anterior. O mapa é composto de células que representam regiões quadradas de $10,1\ cm \times 10,1\ cm$ do espaço real. Além disso, o robô se desloca com velocidade máxima igual a $25,4\ cm/s$ e com uma janela de ativação de raio igual a $2\ m$.

Devido a discretização do ambiente e ao atraso na comunicação do robô com a sua base, para determinar quando o robô atingiu sua posição inicial fizemos com que ao invés do robô atingir uma posição específica, ele devesse atingir uma determinada região circular de raio igual a $25\ cm$ centrada na posição de partida $(0, 0)$. Após ter alcançado a posição inicial, medimos a distância entre o centro do robô e a posição marcada de partida no chão. Chamaremos esta distância de discrepância.

A Tabela 5.5 mostra o comprimento médio $(\bar{\ell})$ do caminho seguido pelo robô durante sua exploração e a discrepância gerada $(\bar{\zeta})$.

TABELA 5.5 – Discrepância e comprimento médio do caminho seguido pelo robô sobre 5 experimentos.

$\bar{\ell}$	σ_ℓ	$\bar{\zeta}$	σ_ζ
31,7m	9,1m	22cm	2,7cm

Observe que a discrepância média se encontra dentro da faixa de $25\ cm$ estipulada previamente e representa aproximadamente $0,6\%$ do comprimento total percorrido pelo robô o que indica que o erro de odometria deve ser substancialmente menor que $22\ cm$. Um dos casos mais interessantes é ilustrado na Figura 5.11(f). Nesta situação o robô teve que lidar com obstáculos dinâmicos (pessoas) assim como realizar diversos movimentos que incluem translações e rotações. Nesta situação, o robô percorreu um caminho igual a $44,0\ m$ com apenas $20\ cm$ de discrepância. No experimento ilustrado na Figura 5.9, o robô percorreu uma trajetória de comprimento igual a $24,35\ m$ com um erro igual a aproximadamente $20\ cm$.

Isto vem comprovar o que discutimos na Seção4.2.2. Um dos grandes benefícios do cálculo do potencial através de métodos de relaxação é a geração de trajetórias suaves que evitam mudanças abruptas na orientação do robô, uma das principais fontes de erros de odometria.

5.4.6 Explorando ambientes internos com obstáculos dinâmicos

Esta seção apresenta alguns resultados extraídos com o robô NOMAD no nosso Instituto com obstáculos móveis. Consideramos que existiam pessoas passando pelo local onde o robô estava se deslocando durante o processo de exploração. O tratamento realizado pelo sistema para pessoas é o mesmo que aquele dado para obstáculos fixos como paredes. A diferença é que as pessoas são representadas no mapa por um período de tempo curto, pois, a próxima vez que o robô passar pelo local onde foi detectada uma pessoa, ele notará que aquele local não existe um objeto e atualizará seu mapa interno para representar sua observação.

Este experimento foi realizado em um ambiente com dimensões aproximadas de $7\ m\ \times\ 18\ m$. A velocidade máxima do robô é igual a $25,4\ cm/s$ e ele é capaz de identificar qualquer objeto a uma distância de $2\ m$. As Figuras 5.10 e 5.11

ilustram uma sequência de exploração em um ambiente com obstáculos dinâmicos. A presença de pessoas é ilustrada nas figuras com um círculo pontilhado. Isto pode ser visto nas Figuras 5.10(a),(d) e (e), e nas Figuras 5.11(b), (c) e (d).

Observe que as pessoas detectadas são gradualmente esquecidas pelo sistema. Isto pode ser visto através da cor das células da posição que representa a pessoa[5]. Observe que as células vão se tornando mais claras à medida que o robô se aproxima delas. Este esquecimento é proporcional ao tempo de exposição da região, que possui uma pessoa detectada, ao robô.

Tomemos como base a Figura 5.10(a). Neste caso, uma pessoa foi detectada e foi gradualmente esquecida pelo robô (ver Figuras 5.10 (b) e (c)), entretanto, ela apenas foi esquecida completamente quando o robô passou por ela pela segunda vez, observe a Figura 5.10 (e).

A Figura 5.11 (b) ilustra outra situação onde uma pessoa foi detectada pelo robô. É importante destacar que nesta situação, a região marcada pelo robô não foi esquecida pois existiam outras pessoas próximas a primeira, isto pode ser visto nas Figuras 5.11 (d) e (e), onde a região marcada cresce ao invés de diminuir.

Este experimento vem comprovar o que foi discutido na Seção 4.2.4. Naquela seção discutimos que o método HIMM e sua generalização permite um fácil tratamento tanto de obstáculos dinâmicos quanto estáticos. Este tratamento é feito apenas incrementando ou decrementando a propriedade certeza de cada célula sobre o cone de visão dos sensores do robô.

5.4.7 Exploração orientada a objetivos

Nesta fase, ilustramos uma exploração orientada a objetivo, onde o robô deve explorar o ambiente a fim de localizar um dado objeto, sem que exista conhecimento prévio sobre sua localização. Os parâmetros do ambiente são os mesmo usados na Seção 5.4.1. O robô está inserido em um ambiente de $10,2\ m \times 10,2\ m$ representado por um mapa de 80×80 células. A janela de ativação é circular com raio igual a $2\ m$ e o mapa é atualizado com Gauss-Seidel usando uma taxa $r = 30$ iterações/passo com atualização global de toda região explorada.

As Figuras 5.12 e 5.13 mostram o resultado da exploração no simulador do Nomad. Quando o objeto é encontrado[6] a exploração finaliza, e como resultado final é gerado um mapa parcial da região explorada pelo robô. Neste caso, o alvo corresponde ao **X** à direita da figura. Este experimento não difere fundamentalmente daquele mostrado na Seção 5.4. O principal ponto aqui é que o objetivo é um objeto atrativo estático que pára o processo de exploração quando é alcançado[7].

Entretanto, o campo potencial que foi descartado no fim da exploração não-direcionada (sem-objetivo), aqui carrega uma importante informação sobre como alcançar o objetivo. Assim, podemos dizer que o produto final do processo exploratório é o mapa parcial do ambiente $c(\mathbf{r})$ e o campo potencial $p(\mathbf{r})$ que pode ser

[5]Lembre-se que quanto menor a confiança na presença de um obstáculo mais clara a célula correspondente a ele será. Quando não existe obstáculo algum, a célula terá cor branca.

[6]Como não dispomos de um sistema de reconhecimento de objetos, consideramos que um objeto é reconhecido quando sua posição está dentro do campo de visão dos sensores do robô.

[7]Na simulação o robô não é programado para parar, assim ele se move ao redor do objetivo. Além disso, para construir um campo potencial conectando todos os pontos no espaço explorado ao objetivo, as fronteiras têm que deixar de ser atrativas. Assim, quando o objetivo é reconhecido todas as células na fronteira têm seu potencial alterado para 1.

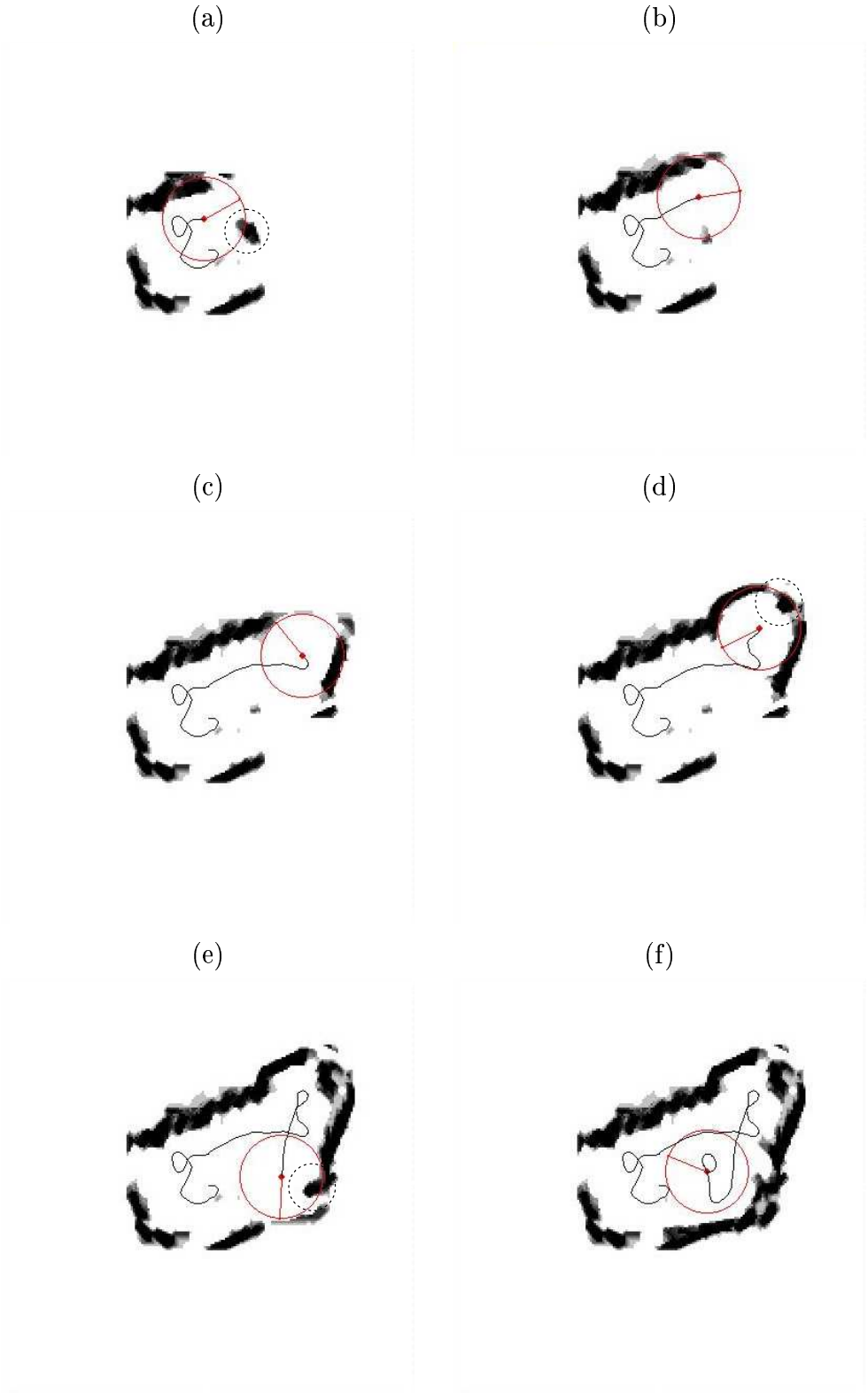


FIGURA 5.10 – Experimento em ambientes com obstáculos dinâmicos - parte I. A figura ilustra o processo de mapeamento de um ambiente realizado pelo robô NOMAD na presença de pessoas. As pessoas são ilustradas por um círculo pontilhado.

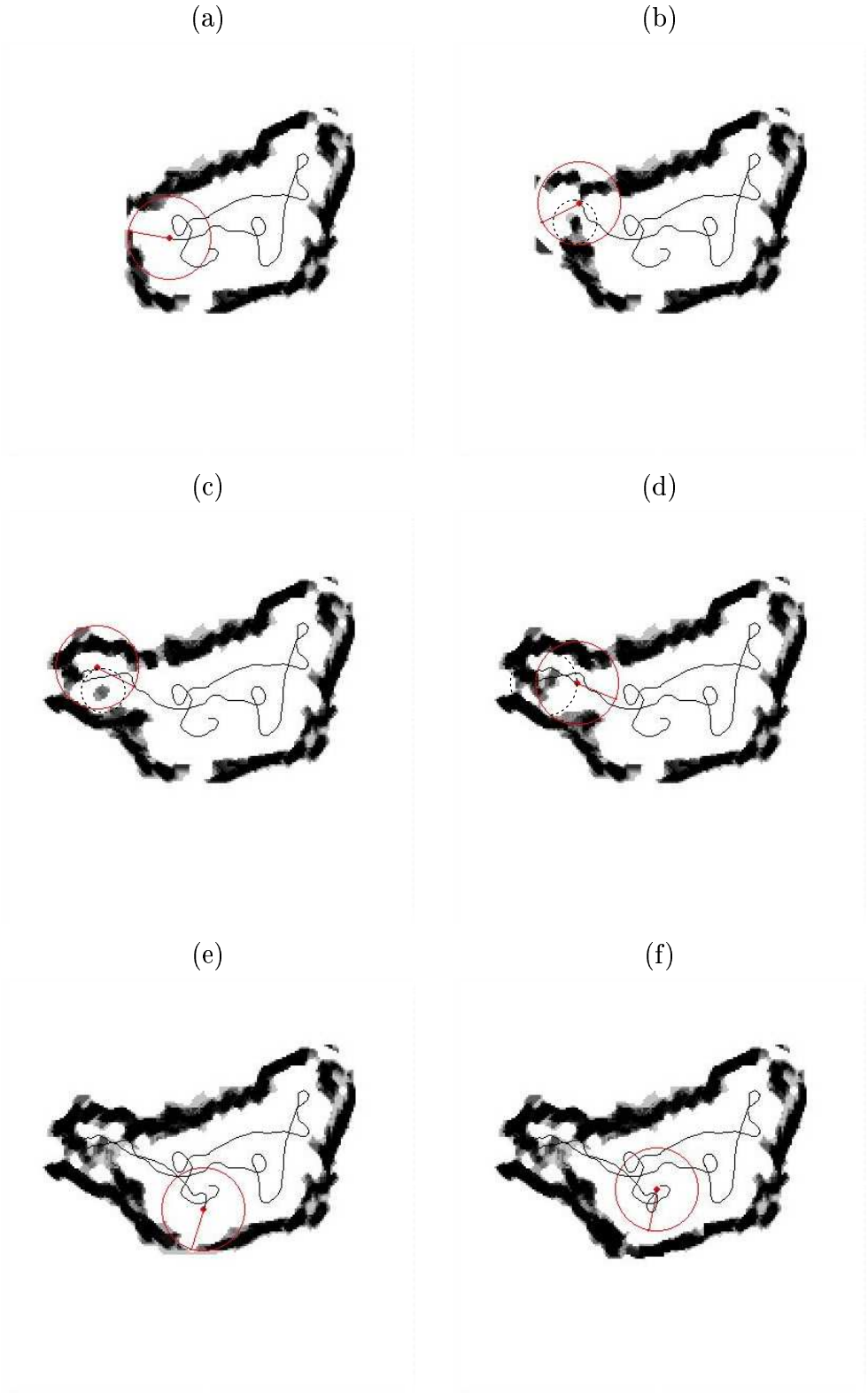


FIGURA 5.11 – Experimento em ambientes com obstáculos dinâmicos - parte II. A figura ilustra o processo de mapeamento de um ambiente realizado pelo robô NOMAD na presença de pessoas. As pessoas são ilustradas por um círculo pontilhado.

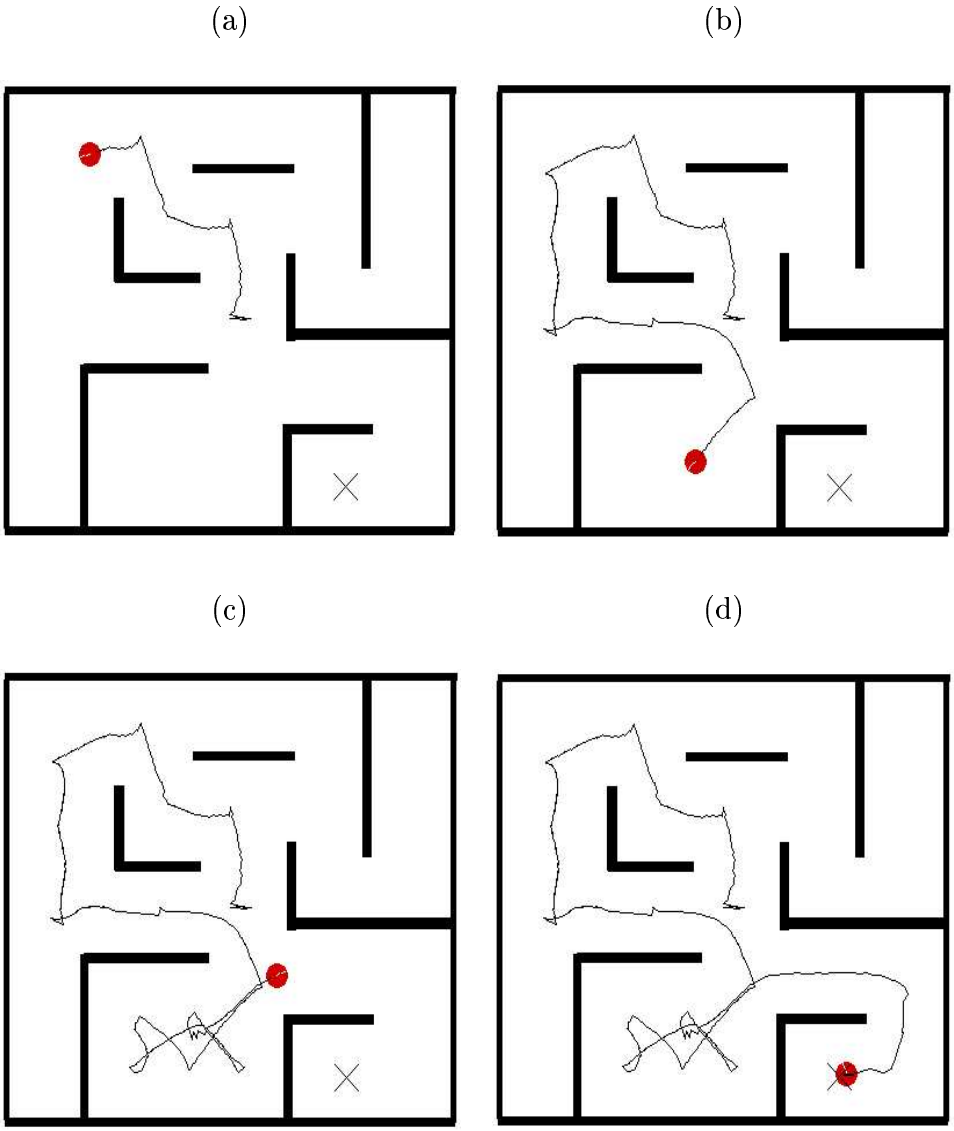


FIGURA 5.12 – Exploração orientada a objetivos: trajetória seguida pelo robô. O ambiente é $10,2\ m\ \times\ 10,2\ m$, e o robô se desloca a cada instante $\Delta d = 7,6\ cm$. O tamanho do mapa é 80×80 células, onde cada célula corresponde a uma região quadrada de $15,2\ cm \times 15,2\ cm$ do espaço real. A janela de ativação é um círculo com raio igual a $2\ m$. A taxa de iteração é igual a $r\ =\ 30$ iterações/passo usando Gauss-Seidel.

usado para planejar caminhos ao objetivo.

O mesmo experimento poderia ser realizado priorizando a exploração. Neste caso, o alvo é reconhecido, mas suas células correspondentes são ainda consideradas como *espaço livre*. O resultado seria muito mais interessante. O robô seria fortemente atraído para as fronteiras, mas com a possibilidade de retornar ao alvo a qualquer instante.

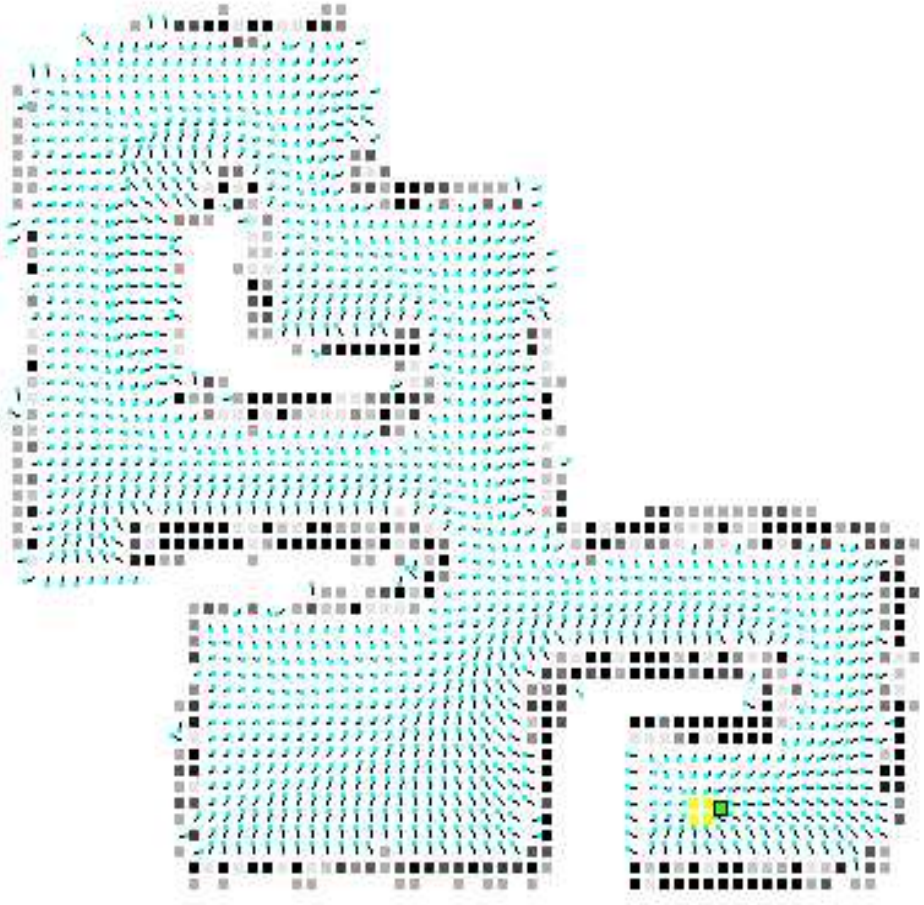


FIGURA 5.13 – Exploração orientada a objetivos: mapa e o campo vetorial. Como antes, o campo vetorial representa o gradiente descendente em cada célula do espaço livre. Ele é normalizado para facilitar a sua visualização. No mapa, o valor de certeza é representado em níveis de cinza, onde o valor 0 corresponde à célula de cor branca e o valor 15 à célula de cor preta.

5.4.8 Usando a janela de ativação adaptativa

Na Seção 4.3.1 discutimos alguns dos possíveis efeitos colaterais provocados pelo uso de uma janela de ativação fixa em um ambiente cujas distâncias entre obstáculos e paredes variam muito. Esta seção apresenta resultados comparativos preliminares entre o uso da janela adaptativa e o uso de uma janela fixa durante o processo de mapeamento de um ambiente cujo espaço livre é inferior ao alcance médio dos sensores do robô.

Os parâmetros dos experimentos são iguais àqueles utilizados nas Seções 5.4.5, 5.4.4 e 5.4.6. Naqueles experimentos, o robô consegue explorar completamente um ambiente semelhante àquele ilustrado na Figura 4.9 (b) com uma janela fixa igual a 2 m. Aqui inserimos o robô em um ambiente com corredores estreitos, semelhante ao apresentado na Figura 4.9 (a).

Os experimentos foram realizados utilizando o robô NOMAD em nosso laboratório (LRI) conforme ilustrado na Figura 5.14. Ao longo das paredes do laboratório existem cadeiras e mesas e em seu centro existe uma mesa que corresponde ao campo

pertencente ao futebol de robôs. Como nosso objetivo é apenas ver o desempenho do robô em um ambiente com corredores estreitos consideramos que a porta do laboratório está fechada. O laboratório tem dimensões de aproximadamente $6,8\ m \times$

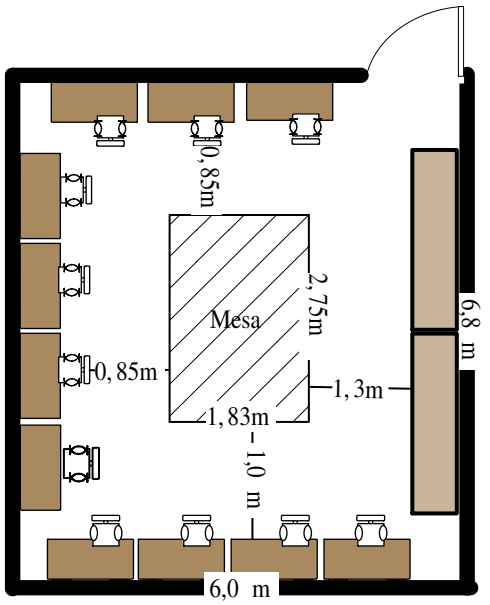


FIGURA 5.14 – Ambiente para teste da janela adaptativa. Suas dimensões externas são aproximadamente $6,8\ m \times\ 6,0\ m$.

$6,0\ m$. As células do mapa, que representa o ambiente, correspondem a uma região de $10\ cm \times\ 10\ cm$ do espaço real. A velocidade máxima do robô é igual a $15\ cm/s$ e ele é capaz de identificar qualquer objeto a uma distância de $2\ m$. É importante destacar que o diâmetro do robô é igual a $0,57\ m$ e o corredor de menor e maior largura medem $0,85\ m$ e $1,3\ m$, respectivamente.

As Figuras 5.15(a) e (b) ilustram o uso de uma janela de ativação adaptativa. Ambas utilizam o retorno dos sensores do tipo sonar do robô. A primeira utiliza o menor de todos os retornos, dada por 4.22, a segunda utiliza a média dos três menores retornos, dada por 4.23. Observe que neste caso, não houve uma significativa diferença entre os resultados produzidos. Ambas conseguiram produzir um mapa parecido com a estrutura do ambiente ilustrado na Figura 5.14. Entretanto é importante destacar que a segunda consegue lidar com maior facilidade com os efeitos colaterais produzidos pelo ruído inerentes na leitura dos sensores do robô, pois este é atenuado através da média dos três menores retornos. Uma sugestão para pesquisa é o uso de um fator de elegibilidade, semelhante ao η apresentado em 4.17, que mantenha uma transição suave entre o tamanho do raio de ativação entre instantes sucessivos no tempo, a fim de que este valor não seja maximizado ou minimizado, indiscriminadamente, devido à presença de ruídos.

As Figuras 5.15(c) e (d) ilustram o uso de uma janela de ativação fixa. No experimento ilustrado na Figura 5.15(c), o robô consegue explorar e mapear corretamente *quase* todo o ambiente. No final da exploração devido ao raio de ativação ser grande comparado às dimensões dos corredores, a quantidade de células classificadas erroneamente fez com que um corredor fosse fechado. O caso mais grave é ilustrado na Figura 5.15(d), pois, devido à classificação errônea o robô se aprisionou.

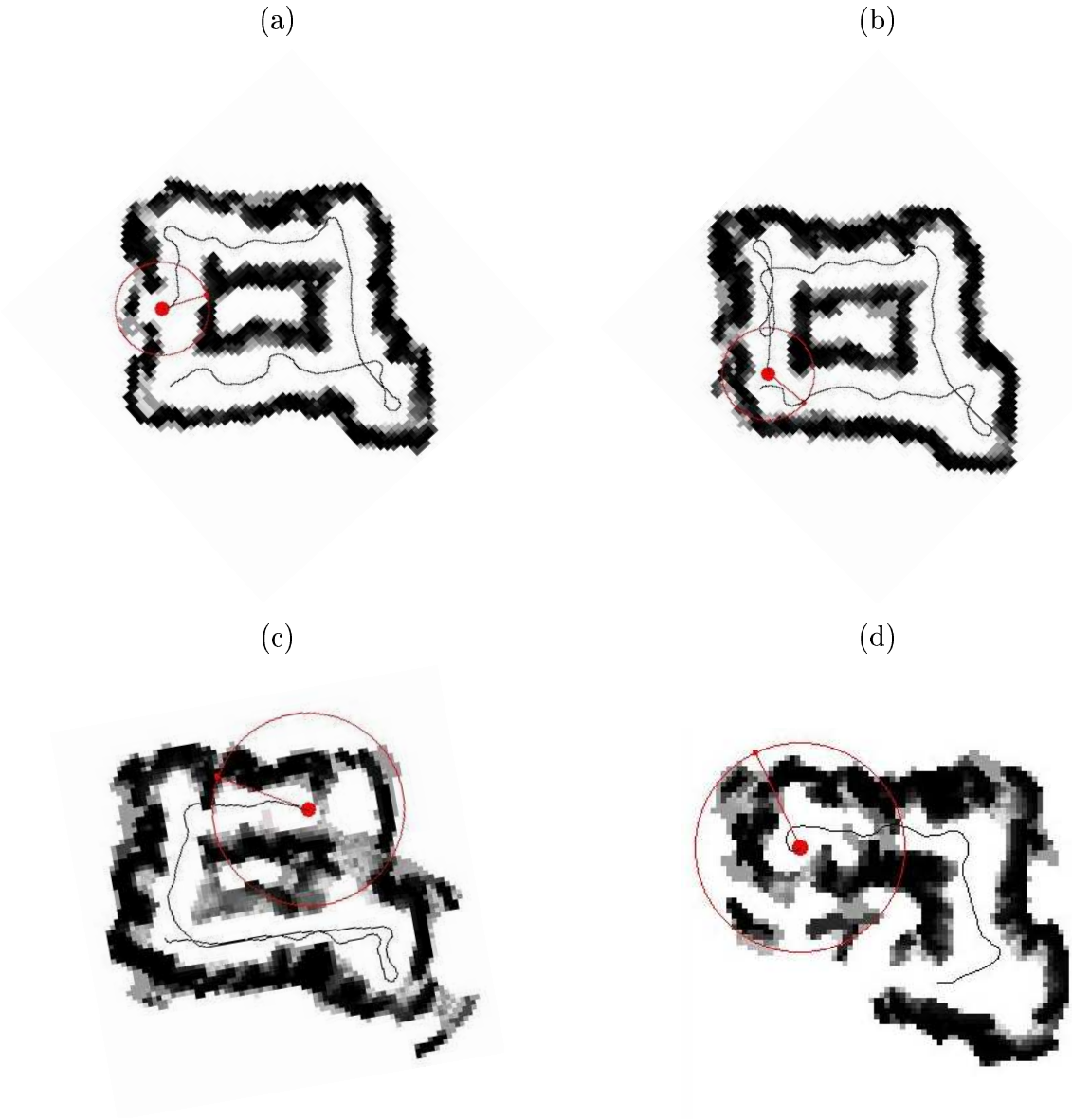


FIGURA 5.15 – Experimento de exploração usando as janelas de ativação adaptativa e fixa. a) e b) usando janela adaptativa. c) e d) usando janela fixa com raio de ativação igual a 2 m. As dimensões externas do ambiente são aproximadamente $6,8\ m \times\ 6,0\ m$.

Além disso, podemos ver uma sensível diferença (para pior) na qualidade do mapa e no processo de exploração produzidos quando comparados ao processo que usa a janela adaptativa. Baseado nestes experimentos, podemos concluir que é vantajosa a utilização da janela adaptativa.

5.4.9 Analisando o conhecimento global × conhecimento adaptativo

Esta seção apresenta alguns resultados preliminares relacionados ao refinamento da exploração baseada em PVC discutido na Seção 4.3.2. A Figura 5.16 (a) mostra o robô explorando um ambiente com diversas paredes formando um labirinto

utilizando seu conhecimento global definido pela sua janela global. A Figura 5.16 (b) mostra o robô explorando o ambiente ilustrado em a) utilizando o seu conhecimento adaptativo.

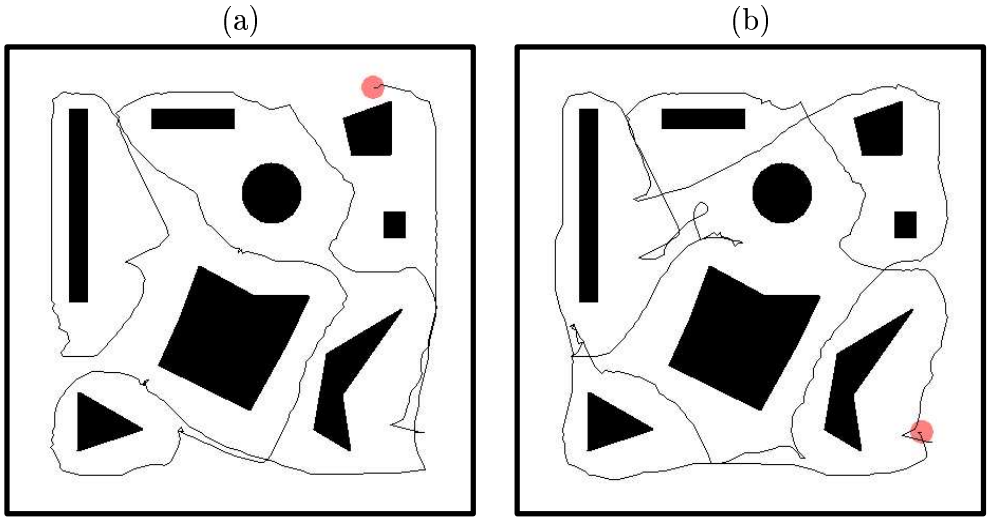


FIGURA 5.16 – Conhecimento global × conhecimento adaptativo. a) exploração orientada pelo conhecimento global. b) exploração orientada pelo conhecimento adaptativo.

A escolha de uma ou outra estratégia não afeta qualitativamente o caminho seguido pelo robô. Entretanto, existe uma diferença considerável relacionada à quantidade de células atualizadas por unidade de tempo como é mostrada na Figura 5.17.

Para comprovar a eficiência do conhecimento adaptativo, a Tabela 5.6 mostra o resultado de um conjunto de 10 experimentos em 2 ambientes compostos de obstáculos de diferentes formatos e tamanhos formando labirintos. Ambos ambientes possuem dimensões iguais a $1\ m \times 1\ m$. As células do mapa correspondem a uma região de $2\ cm \times 2\ cm$ do espaço real e o robô é capaz de identificar qualquer objeto a uma distância de $10\ cm$.

A Tabela 5.6 mostra a quantidade média ($\bar{q}_a$) de células atualizadas usando conhecimento adaptativo e conhecimento global, assim como a flutuação de seus resultados, durante todo o processo de exploração. Observe que utilizando o conhecimento adaptativo estamos atualizando apenas $39,1\%$ do número de células que seriam atualizadas de maneira tradicional através do conhecimento global ainda mantendo a qualidade do processo de exploração.

TABELA 5.6 – Quantidade média de células atualizadas durante todo o processo de exploração de um ambiente típico usando conhecimento global e adaptativo sobre 10 experimentos.

Conhecimento	$\bar{q}_a$	σ_{q_a}
global	$9,2 \times 10^6$	$2,1 \times 10^6$
adaptativo	$3,6 \times 10^6$	$1,5 \times 10^6$

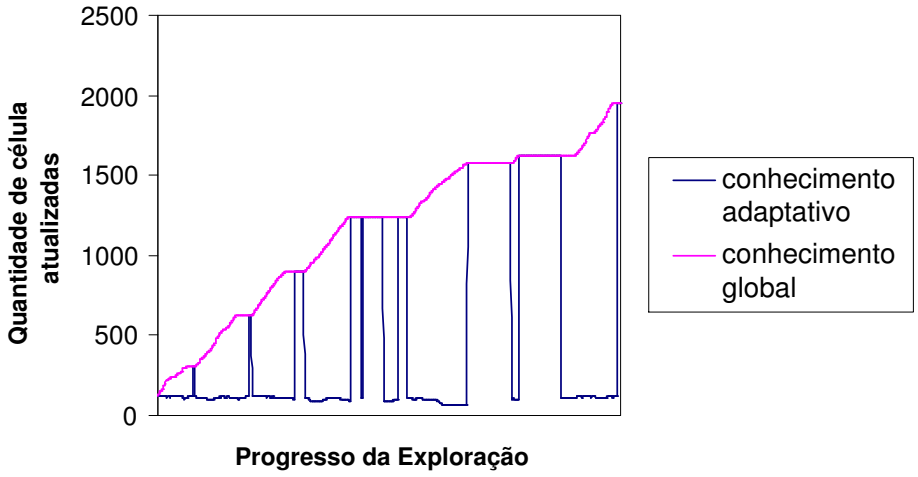


FIGURA 5.17 – Quantidade de células atualizadas durante o processo de exploração de um ambiente interno típico.

É importante ressaltar que os valores mostrados na Tabela 5.6 dizem respeito apenas as células que têm sua propriedade potencial atualizada durante o processo de exploração. Além dessa informação, convém considerarmos também a quantidade de células que são acessadas pelo sistema, mas não são atualizadas durante o processo de relaxação, por exemplo, células que correspondem a obstáculos e a regiões não exploradas[8].

A Tabela 5.7 mostra a quantidade média ($\bar{q}_v$) de células visitadas usando tanto o conhecimento adaptativo quanto o conhecimento global, assim como a flutuação de seus resultados, durante todo o processo de exploração. No caso do conhecimento global, estas células correspondem a todas as células que estão dentro da janela global do robô. No caso, do conhecimento adaptativo, estas células correspondem a todas as células que estão dentro da janela local, quando esta janela estiver sendo utilizada, ou todas as células que estão dentro da janela global, quando esta estiver sendo utilizada.

TABELA 5.7 – Quantidade média de células visitadas durante todo o processo de exploração de um ambiente típico usando conhecimento global e adaptativo sobre 10 experimentos.

Conhecimento	$\bar{q}_v$	σ_{q_v}
global	$14,4 \times 10^6$	$3,2 \times 10^6$
adaptativo	$5,3 \times 10^6$	$2,2 \times 10^6$

Na Tabela 5.8 mostramos o comprimento médio ($\bar{\ell}$) do caminho percorrido pelo robô usando tanto o conhecimento adaptativo quanto o conhecimento global,

[8] Tanto o conhecimento global quanto o conhecimento adaptativo são definidos por uma janela retangular que circunscreve a região potencial, ou seja, eles podem possuir células que ainda não foram exploradas pelo robô.

assim como a flutuação de seus resultados, durante todo o processo de exploração. Observe que o robô seguiu um caminho usando conhecimento adaptativo, em média, $4,9\%$ maior que o caminho produzido através do uso do conhecimento global. Entretanto, é importante destacar que este aumento é compensado com o ganho de aproximadamente 61% no cálculo do campo potencial. Em resumo, é vantajoso utilizar o conhecimento adaptativo, pois é possível manter na maior parte do tempo um baixo custo computacional relacionado à atualização do potencial das células do mapa.

TABELA 5.8 – O caminho médio seguido pelo robô durante todo o processo de exploração de dois ambientes típicos usando conhecimento global e adaptativo sobre 10 experimentos.

Conhecimento	ℓ	σ_ℓ
global	$15,5m$	$1,4m$
adaptativo	$16,3m$	$1,2m$

5.5 Experimentos de exploração em ambientes internos *esparsos*

As seções anteriores mostraram o funcionamento do nosso agente baseado em um caso particular dos problemas de valores de contorno, as funções harmônicas. Esta seção discute o uso desta particularidade em ambientes internos esparsos.

Nós consideramos um ambiente esparso, como sendo aquele onde o robô navega durante um longo tempo sem observar obstáculo algum. Tanto nas funções harmônicas quanto nos métodos similares que objetivam minimizar a função de custo relacionada à presença dos objetos no ambiente, a esparsidade resulta em um comportamento exploratório ineficiente, pois quando o robô detecta um obstáculo é automaticamente gerada uma força que o empurra em direção oposta o fazendo seguir um caminho oscilatório e visitar um mesmo lugar diversas vezes [THR 98].

Devido a isso, generalizamos a equação 3.2(ver Seção 3.2) adicionando um termo que age como um campo de força à função de atualização do potencial. Agora o cálculo da função potencial não se resume apenas ao cálculo da equação de Laplace, mas ao cálculo de uma equação mais geral que representa uma família de equações. A inserção deste termo resulta em uma exploração mais eficiente enquanto mantém sua robustez aos mínimos locais [PRE 2002a].

Esta seção apresenta inicialmente uma introdução ao problema de exploração de ambientes internos esparsos, seguida da codificação das regras de atualização do campo potencial apresentadas na Seção 3.2 e dois grupos de experimentos. É importante destacar, que a configuração de parâmetros utilizada nestes experimentos é a mesma configuração utilizada nas seções anteriores.

5.5.1 Explorando ambientes críticos : ambientes internos *esparsos*

Os ambientes ilustrados anteriormente possuem corredores cuja largura é da ordem do alcance dos sensores do robô, de modo que sempre existem obstáculos

sobre campo de visão do robô. Conseqüentemente o robô sempre está sob influência da força repulsiva gerada pelos obstáculos.

Esta força repulsiva mantém o robô centrado nos corredores o empurrando automaticamente para as regiões desconhecidas do ambiente, o que faz com que a decisão do robô se reduza a escolher a ordem de exploração dos corredores. Ao final do processo, geralmente se tem uma trajetória cuja qualidade é próxima à ótima.

Quando o robô é colocado em um ambiente esparso a situação muda. A esparsidade aumenta a complexidade do processo de exploração, pois agora os obstáculos e paredes estão separados por distâncias que em média são maiores que o alcance dos sensores. Nesta situação, o robô tem que decidir de maneira eficiente qual é a seqüência de passos que levam a exploração completa do ambiente sem a orientação de qualquer informação estrutural. Dessa forma, a questão é: na ausência de retorno das paredes, o agente ainda produzirá uma exploração ótima?

A qualidade da exploração produzida pelas funções harmônicas é pobre, pois como será visto adiante, elas produzem um campo potencial que direciona o robô sempre para a maior região não explorada, fazendo com que ele assuma um comportamento oscilatório e visite um mesmo lugar diversas vezes.

Para melhorar o desempenho do processo exploratório, consideramos $F(\mathbf{r}) \neq 0$ (ver equação 3.2) e testamos duas funções diferentes para $F(\mathbf{r})$. A primeira chamada **regra 1** igual a $F(\nabla p) = \nabla p \cdot \mathbf{v}$, e a segunda chamada **regra 2** igual a $F(\nabla p) = (\nabla p)^2$ (ver Seção 4.2.6).

Na regra 1, $\mathbf{v}$ é um vetor constante que introduz um campo de força homogêneo na dinâmica do sistema que cria uma direção preferencial de exploração. Consideramos inicialmente, o vetor $\mathbf{v} = (1, 1)$, em seguida mostramos que a direção de $\mathbf{v}$ não afeta o desempenho do processo. A segunda regra muda o relacionamento entre a repulsão e a atração no ambiente. Ambas as formas foram introduzidas para contrapor a tendência natural do robô de se afastar dos obstáculos detectados a fim de possibilitar uma exploração mais local.

Como mostrado na Seção 5.4.3 e em [PRE 2002], SOR não é um método de relaxação adequado quando é necessário um potencial parcialmente relaxado. Dessa forma, as equações acima são escritas utilizando o método Gauss-Seidel como mostrado na Seção 4.2.6.

5.5.2 Experimentos no robô simulado

Inserimos o robô em 2 ambientes quadrados: o primeiro, chamado ambiente **A** mede $1\ m \times 1\ m$; e o segundo chamado **B** mede $2\ m \times 2\ m$. Em ambos os casos, cada célula do mapa corresponde a uma região de $2,5\ cm \times 2,5\ cm$.

Os ambientes são compostos por poucos obstáculos e a única diferença entre eles é sua dimensão. Eles correspondem a uma grande sala com duas colunas em cantos opostos. Com esta representação, é possível realizar dois níveis de exploração: uma global, correspondendo ao grande espaço vazio central e outra mais refinada, correspondendo aos espaços menores atrás das colunas.

A simulação do ambiente esparso foi feita considerando que o robô apenas consegue detectar objetos a uma distância de 10 cm. Em todos os experimentos consideramos a taxa de atualização $r = 30$ iterações/passo. A medida de desempenho entre as regras é feita através do comprimento total do caminho (L) seguido pelo robô para cobrir todo o ambiente.

Desempenho das Funções Harmônicas

A Figura 5.18 mostra o desempenho do robô no ambiente **A** e **B** usando o potencial harmônico. No ambiente **A**, o robô realizou a exploração seguindo um caminho de comprimento igual a $L = 10,1\ m$. Enquanto que no **B**, a exploração foi realizada seguindo um caminho igual a $L = 40,1\ m$. Em ambos os casos nós podemos observar que o centro da sala foi visitado várias vezes durante a exploração.

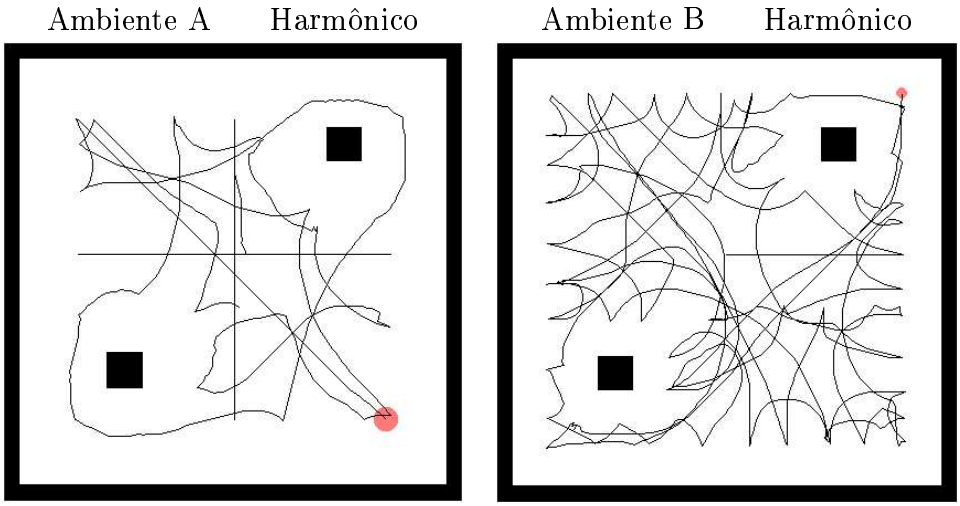


FIGURA 5.18 – Seqüência de exploração nos ambientes esparsos usando o potencial harmônico. O ambiente **A** foi explorado seguindo um caminho de comprimento $L = 10,1\ m$, enquanto que no ambiente **B**, este caminho foi igual a $L = 40,1\ m$.

Desempenho das Regras 1 **e** 2

A Figura 5.19 mostra o desempenho do robô usando a regra 1, no ambiente **A**, para dois valores diferentes para o parâmetro ϵ. Para $\epsilon = 0,5$, a exploração foi realizada com $L = 5\ m$, enquanto que com $\epsilon = 0,8$, ela resultou em um caminho razoavelmente melhor com $L = 4,6\ m$. Os caminhos seguidos pelo robô mostram um melhoramento evidente, pois ele não mais oscila no centro da sala como mostrado na seção anterior.

A Figura 5.20 mostra o desempenho do robô usando a regra 2 no ambiente **A**. Nesta situação, o robô explorou o ambiente completamente com $L = 7,3\ m$ para $\epsilon = 0,5$ e com $7,6\ m$ para $\epsilon = 0,8$. O desempenho do robô não foi tão notável quanto o apresentado pela regra 1. Observe que mesmo que o comprimento do caminho total tenha diminuído, o centro da sala ainda exerce uma grande atração sobre o robô.

A Figura 5.21 mostra o desempenho do robô usando as regras 1 e 2 com os melhores valores obtidos para o parâmetro ϵ. Este experimento tem o objetivo de analisar como o desempenho do robô é afetado com o escalonamento das dimensões do ambiente. Nesta situação, o robô explorou completamente o ambiente com a regra 1 com $L = 26,3\ m$ e com a regra 2 um com $L = 36,2\ m$.

Nos experimentos realizados acima, observe que o comportamento do robô gerado pelo potencial harmônico é muito ineficiente e oscilatório, pois ele é apenas influenciado pela força repulsiva gerada pelos obstáculos. Em compensação, os resultados apresentados pelas regras 1 e 2 mostram um favoritismo em realizar uma exploração mais local, o que resultou em um melhoramento considerável no desempenho geral do processo exploratório.

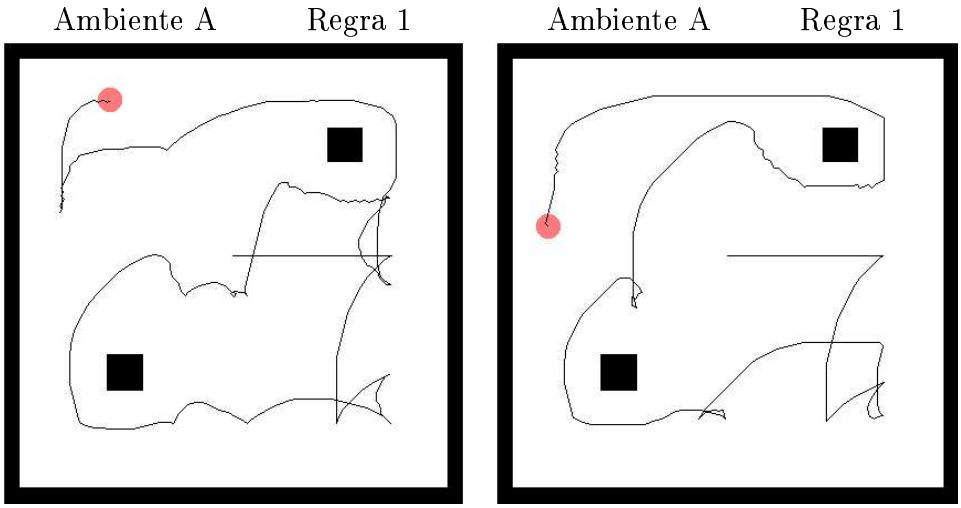


FIGURA 5.19 – Seqüência de Exploração usando a regra 1 no ambiente **A**. Para $\epsilon = 0,5$, $L = 5\ m$ e para $\epsilon = 0,8$, $L = 4,6\ m$.

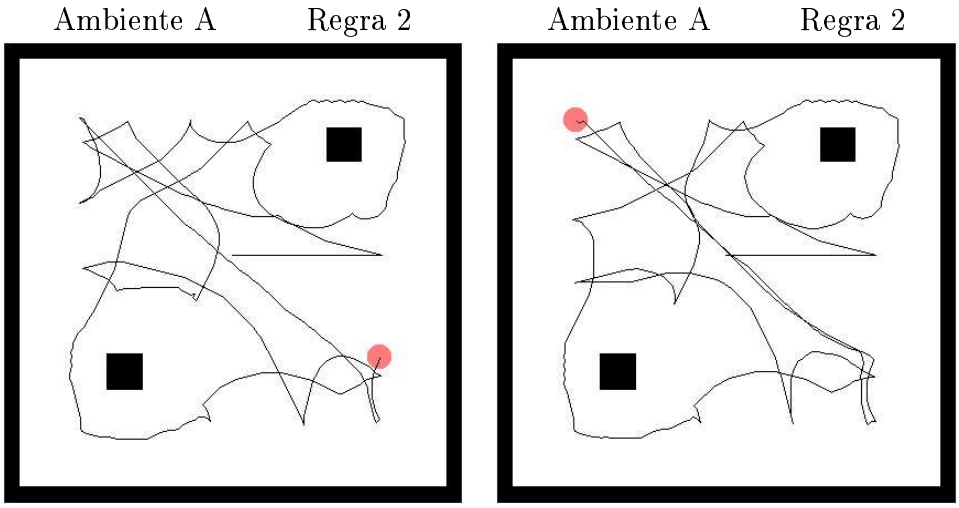


FIGURA 5.20 – Seqüência de Exploração usando a regra 2 no ambiente **A**. Para $\epsilon = 0,5$, $L = 7,3\ m$ e para $\epsilon = 0,8$, $L = 7,6\ m$.

5.5.3 Avaliação do desempenho

Os experimentos apresentados anteriormente apresentam desempenhos típicos. Entretanto, isto não é suficiente para assegurar a utilidade e a qualidade das novas regras. Esta seção apresenta resultados extraídos de diversas execuções com o robô iniciando de posições aleatórias no ambiente e parando apenas quando o ambiente está completamente explorado. Isto é feito a fim de observar como o desempenho flutua ao redor de seu valor médio.

A Tabela 5.9 mostra o desempenho médio e o desvio padrão correspondente a 6 diferentes julgamentos feitos com as duas principais regras. Neste caso, os experimentos foram realizados utilizando a regra básica definida pelas funções harmônicas e a melhor regra entre as duas novas, a regra 1.

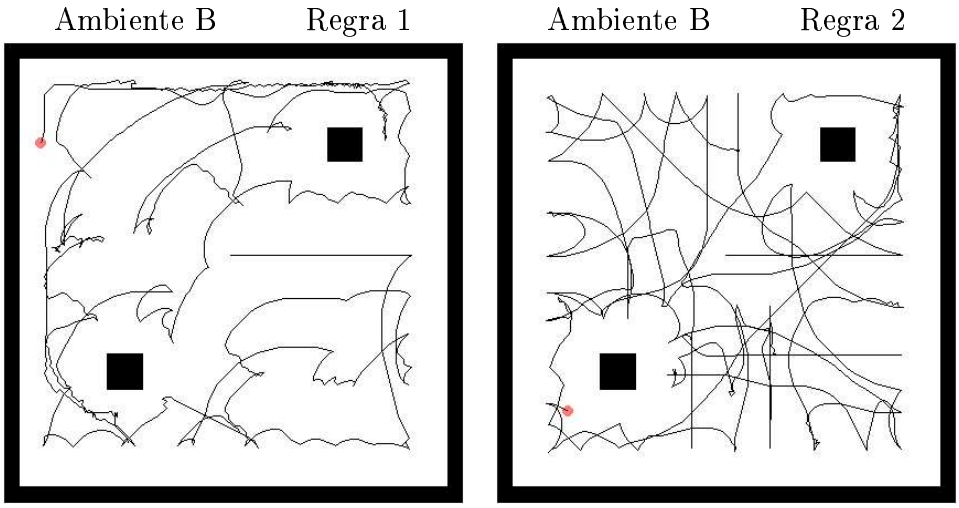


FIGURA 5.21 – Seqüência de Exploração no ambiente **B**. Para a regra 1 e $\epsilon = 0.8$, $L = 26,3\ m$ e para a regra 2 e $\epsilon = 0,8$, $L = 36,2\ m$.

Como o ambiente é simples e tem muitas simetrias, poucas execuções nos fornecem uma boa estimativa do desempenho médio das regras. Na prática, o tamanho do conjunto de exemplos é dependente da complexidade do ambiente.

TABELA 5.9 – Resultado do desempenho médio e o desvio padrão correspondente a 6 diferentes julgamentos feitos com as duas principais regras e o potencial harmônico.

Regra	ϵ	Tamanho	L(m)
Harmônico	-	1×1	$9,0(1 \pm 0.09)$
Regra 1	0,5	1×1	$6,3(1 \pm 0.08)$
Regra 1	0,8	1×1	$5,2(1 \pm 0.08)$
Harmônico	-	2×2	$41,4(1 \pm 0.06)$
Regra 1	0,8	2×2	$24,3(1 \pm 0.06)$

Os valores para o desvio padrão indicam que o desempenho é consistente com uma dispersão ao redor de 10%.

Uma questão diferente é como a direção do vetor **v** influencia no desempenho do processo exploratório. A fim de abordar este problema testamos o robô em um ambiente quadrado com 2 paredes internas paralelas, como ilustrado na Figura 5.22. Uma parede é ortogonal à parede sul, e a outra é ortogonal à parede norte. Elas dividem o ambiente em 3 partes, todas conectadas formando uma estrutura em forma de 'S' com uma direção preferencial direção norte-sul. Nosso objetivo é testar como o robô se comporta se **v** não está orientado com a direção preferencial.

A Tabela 5.10 apresenta os resultados obtidos de 10 execuções. Ela mostra que o desempenho do processo de exploração não é afetado pela escolha da direção de **v**. O que é um bom resultado, pois se a direção de **v** fosse crítica para o desempenho teríamos que determiná-la previamente.

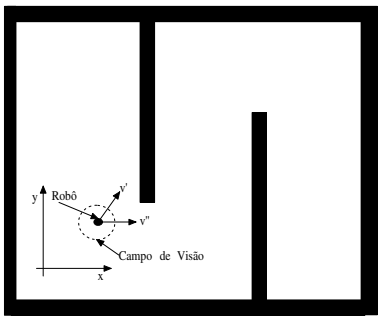


FIGURA 5.22 – Ambiente esparso para teste da direção do vetor **v** da regra 1.

TABELA 5.10 – Resultados médio obtidos sobre 10 execuções com diferentes direções de **v**.

Regra	ϵ	**v**	$L(m)$
Regra 1	0,8	$(1,1)$	$4,8(1 \pm 0.10)$
Regra 1	0,8	$\sqrt{2}(1,0)$	$4,9(1 \pm 0.08)$

5.5.4 Experimentos no robô Nomad

A dificuldade em encontrar um ambiente realmente esparso, fez com que simulássemos a esparsidade limitando por software o alcance dos sensores sonar do robô Nomad. Os sensores do Nomad permitem a detecção de um objeto a uma distância máxima de $6,5\ m$. Em nossos experimentos consideramos que um objeto para ser detectado deveria estar a uma distância máxima igual a $1,5\ m$, o que representa aproximadamente 24% de sua capacidade. Para acelerar o processo exploratório, aumentamos a velocidade máxima do robô de $10\ cm/s$ para $25,4\ cm/s$.

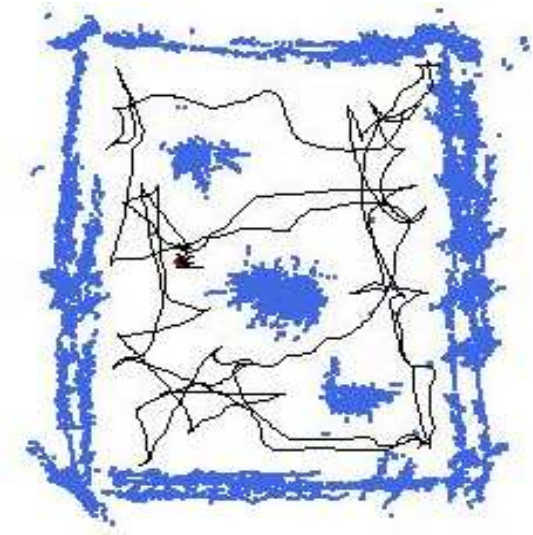
Os experimentos foram realizados em uma sala retangular de $14\ m \times 16\ m$ com uma coluna central no prédio do curso de botânica da UFRGS. Para que o experimento se aproximasse daqueles descritos anteriormente, colocamos 2 pequenos obstáculos próximos aos cantos opostos de maneira similar àqueles usados na simulação. Cada célula do mapa corresponde a uma região de $50\ cm \times 50\ cm$. As dimensões da célula foram escolhidas heuristicamente para agilizar o processo do sistema.

A Figura 5.23 mostra o resultado de dois experimentos realizados com o robô Nomad utilizando as funções harmônicas e a regra 1. Nesta figura é mostrado tanto a trajetória do robô quando o histórico da leitura do sonar durante todo o processo exploratório.

A Tabela 5.11 mostra o resultado médio de 4 execuções com o robô iniciando de diferentes posições no ambiente.

Estes resultados indicam uma melhoria considerável no desempenho do robô. Esta melhoria é menor que a observada nas simulações (ver Tabela 5.9) devido aos erros inerentes ao robô real.

Harmônico



Regra 1

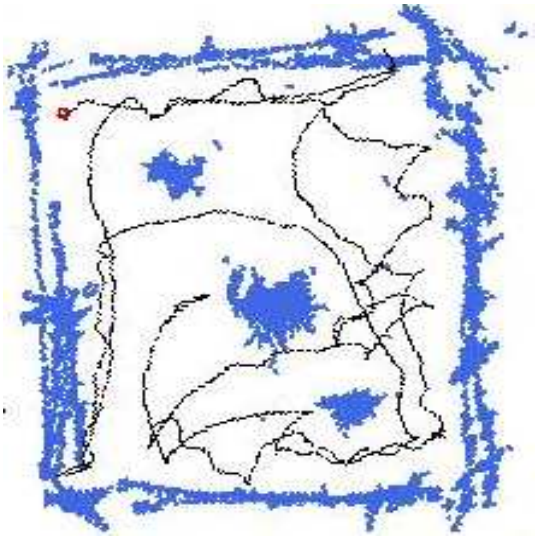


FIGURA 5.23 – Seqüência de exploração usando as funções harmônicas e a regra 1 com $\epsilon = 0,7$. Para as funções harmônicas, $L = 194,9\ m$ e para a regra 1, $L = 156,0\ m$.

TABELA 5.11 – Resultado médio de 4 execuções com o robô NOMAD 200 iniciando de diferentes posições no ambiente usando as duas principais regras.

Regra	L (m)
Harmônico	$185,5(1 \pm 0.08)$
Regra 1	$140,3(1 \pm 0.18)$

6 Conclusões

O objetivo desta tese foi investigar e desenvolver uma teoria unificada para o problema de planejamento e exploração de ambientes não estruturados e utilizá-la como núcleo de uma arquitetura de um agente exploratório. Esta teoria é baseada na solução de problemas de valores de contorno envolvendo a equação de Laplace e tem como objetivo fundamental unificar a compreensão e a solução de problemas relacionados a planejamento através de um princípio único, robusto e eficiente.

Neste trabalho damos ênfase ao processo exploratório, haja vista que o planejamento usando problemas de valores de contorno utilizando apenas a equação de Laplace foi anteriormente apresentado por Connolly [CON 93], tanto para o problema de posicionamento de braços mecânicos, como para o problema de estacionar robôs móveis utilizando mapas previamente definidos.

Antes de apresentarmos nossa teoria, destacamos alguns trabalhos relacionados ao nosso, situando sua importância e seus principais componentes (ver Capítulo 2). Observamos que, para que o processo exploratório seja bem sucedido, ele deve ser apoiado por diversos componentes individuais e complementares, entre as quais se destacam: um subsistema de mapeamento, um subsistema de auto-localização e um subsistema de planejamento de movimentos[1]. Cada um deles representa uma grande área dentro da robótica e compõem a base desejável para qualquer robô realmente autônomo. Utilizamos uma instância de cada um deles para a construção do nosso sistema, com exceção da auto-localização que está em fase de desenvolvimento.

Em seguida no Capítulo 3, apresentamos a nossa teoria baseada na solução de problemas de valores de contorno. Inicialmente introduzimos o planejador de caminhos que utiliza um mapa prévio, proposto por Connolly[CON 93], baseado na solução de um PVC e mostramos como estendê-lo gerando uma ferramenta robusta para exploração. Analisamos e constatamos que este método pode ser generalizado por qualquer PVC cuja equação diferencial parcial não apresente mínimos locais no potencial resultante. Assim, propomos uma família de equações que incluem a equação de Laplace que produz campos potenciais sem mínimos locais. Esta família gera campos potenciais que não são mais apenas definidos pelo contorno da região explorada, mas por uma direção preferencial, o que é de grande utilidade quando o robô está inserido em um ambiente interno esparso.

No Capítulo 4, abordamos nosso sistema do ponto de vista da IA, situando nossa estratégia como uma estratégia baseada em modelos que utiliza uma função heurística. Observamos que a principal característica das funções heurísticas em relação às estratégias sistemáticas, busca em largura e profundidade, é a capacidade de gerar uma política de movimento eficiente. Destacamos as principais características do sistema; apresentamos e descrevemos o funcionamento dos módulos, e a interação entre eles.

No Capítulo 5, realizamos uma série de experimentos de exploração utilizando dois tipos de métodos de relaxação: Gauss-Seidel e SOR. Implementamos e apresentamos outras técnicas padrão: exploração aleatória e *seguir paredes*; e comparamos

[1] É importante ressaltar que nossa teoria é a base para o subsistema de planejamento e corresponde a um módulo multicomportamental, o qual é responsável por realizar tanto a exploração de um ambiente desconhecido como o planejamento de caminhos utilizando o mapa aprendido durante a exploração.

com nossos resultados. Propusemos um mecanismo para aumentar a eficiência do processo de relaxação, através do chaveamento entre o conhecimento local e global do robô, e melhorar a qualidade do mapa produzido, através da janela adaptativa. Verificamos o tempo de exploração, os erros de odometria gerados e o comportamento do sistema em um ambiente dinâmico. Apresentamos resultados preliminares da utilização da janela adaptativa no processo de construção de mapas. Além disso, testamos nossa estratégia em dois tipos de ambientes, esparsos e densos, e analisamos a influência do termo linear utilizado no cálculo do problema de valor de contorno.

Podemos concluir a partir dos resultados apresentados que o algoritmo baseado em problemas de valores de contorno para guiar o processo exploratório é robusto e computacionalmente eficiente. Além disso, por ser baseado em um método global completo ele também herda a capacidade de planejar caminhos e a característica de ser completo. No caso da exploração, ser completo consiste em alcançar todas regiões não exploradas caso exista um caminho até elas. Este trabalho é um passo inicial para a extensão dos métodos globais de planejamento de caminhos baseados em campos potenciais.

A seguir analisamos alguns pontos do nosso sistema de exploração. Entre eles destacamos: a arquitetura utilizada e, particularmente, o módulo de planejamento, o qual corresponde ao núcleo da arquitetura e ao elemento responsável por gerar o comportamento exploratório do nosso agente. Além disso, analisamos os resultados obtidos e apresentamos sugestões para trabalhos futuros.

6.1 Arquitetura do agente

Nesta tese exploramos a utilidade da arquitetura *blackboard* para resolver o problema de exploração de ambientes desconhecidos. Para este problema, representamos o ambiente como um modelo icônico o qual é aprendido à medida que a exploração é realizada. Este modelo é discreto e armazena tanto a posição dos obstáculos como a posição dos objetivos candidatos. Ele pode ser pensado como um grafo de posições, onde cada nó deste grafo corresponde a uma posição no espaço real.

Neste caso em particular, este grafo é tratado como uma malha retangular, onde cada elemento é endereçado por índices ij e possui uma relação direta com uma região no espaço real. Além de fazer referência a uma região específica no ambiente, cada elemento ij pode armazenar propriedades, de sua região associada, chamadas *atributos*.

Este modelo assume o papel do grafo de espaço de estados do robô. Uma ação do robô *controla* a mudança de seu estado, entretanto tem a *intenção* de extrair o mapa (configuração) do ambiente. Além disso, este tipo de representação pode ser pensado como um modelo discreto onde ações contínuas podem ser calculadas. O robô se desloca no espaço real, mas suas ações são planejadas a partir de gradientes discretos extraídos de valores do potencial sobre o grafo.

6.2 Módulo de Planejamento

O módulo de planejamento tem um papel fundamental na arquitetura de nosso agente. Ele é responsável por atualizar o valor do potencial armazenado em cada nó do grafo que representa o ambiente. Este potencial é equivalente a uma função valorada que indica o sucesso ou a recompensa (do ponto de vista das técnicas de aprendizado por reforço) recebida pelo agente após a execução de uma determinada ação. Ele é calculado a partir unicamente do conhecimento da localização dos obstáculos.

Isto somente é possível devido ao uso da solução de um PVC. A solução da versão discreta da equação de Laplace e sua generalização para um conjunto de obstáculos gera automaticamente uma função valorada que corresponde à probabilidade de colisão do robô com os obstáculos [CON 94]. Através do cálculo do PVC, o módulo de planejamento guia o robô às regiões de baixo potencial que correspondem às áreas ainda não visitadas, ou seja, a exploração é o resultado da perseguição de células não exploradas. Isto corresponde a um modelo tele-reativo [NIL 98] onde o condicionamento do comportamento local resulta na realização de um objetivo mais global, no nosso caso, a exploração e o mapeamento do ambiente.

Entretanto, os PVC possuem alguns problemas que merecem destaque para futuras pesquisas. Primeiro, o tempo para o cálculo da função potencial é proporcional ao número de células do mapa. Isto pode gerar um grande ônus computacional quando a região a ser explorada é muito extensa. Para resolver este problema, uma solução é o uso de estruturas mais eficientes do tipo quad-trees [ZEL 98, FOL 90], ou uma abordagem que permita o gerenciamento de múltiplos mapas baseados em grade, onde o potencial apenas é calculado naquele mapa que contém a posição corrente do robô.

Outra limitação vem do fato de que as soluções de (3.2) são funções de rápido decaimento decorrente da precisão relacionada ao seu cálculo. Isto reduz sua utilidade para o planejamento de caminho em ambientes grandes, pois em regiões muito distantes de uma área não explorada ou de um objetivo o gradiente é nulo. Para abordar este problema estamos investigando o uso de uma família de funções com propriedades similares a elas [IDI 2002].

6.3 Discussão e sumário dos resultados

6.3.1 A equação base e suas variações

Em aplicações na robótica móvel, a solução da equação de Laplace (funções harmônicas) é utilizada como um simples planejador de caminhos. Nestes casos, tanto o modelo do ambiente quanto a localização da posição do objetivo devem ser precisos e conhecidos previamente. Neste capítulo mostramos que é possível estender sua aplicabilidade utilizando versões parcialmente relaxadas do seu campo potencial juntamente com um conjunto de sub-objetivos definidos implicitamente pelas regiões desconhecidas do ambiente[PRE 2001, PRE 2002]. Isto gera uma ferramenta robusta para exploração que elimina a obrigatoriedade de um mapa prévio do ambiente de atuação do robô, como pode ser observado nos experimentos ilustrados na Seção 5.4 e no robô real ilustrado na Figura 5.6.

Entretanto, mostramos que a equação de Laplace é capaz de lidar eficiente-

mente apenas com ambientes *internos* densos. Neste tipo de ambiente, a equação de Laplace conta com a largura dos corredores para indicar a região a ser explorada pelo robô.

No caso de ambientes esparsos a situação é mais problemática. Na maioria das vezes, o robô se encontra em uma situação onde existe um único obstáculo que o repele e o limite não explorado é maior que o alcance médio de seus sensores. Devido a isso, a repulsão gerada pelo obstáculo o empurra em direção oposta ao seu movimento o fazendo vagar em um espaço completamente vazio, ao invés de o empurrar ao limite não explorado mais próximo do obstáculo detectado. Isto faz com que o robô siga uma trajetória oscilatória, resultando em um comportamento ineficiente, pois um mesmo lugar é visitado diversas vezes (ver Figura 5.18).

Para solucionar este problema propomos a generalização da equação base (Laplace) do módulo de planejamento através da adição de funções mais gerais (ver Seção 3). Esta adição resultou em uma família de funções potenciais (Equação 3.2) para o desenvolvimento de algoritmos de exploração e planejamento de caminhos [PRE 2002a]. A principal característica destas funções é uma distorção provocada no potencial que guia o robô. Esta distorção produz um comportamento exploratório mais eficiente como mostrados nos experimentos realizados na Seção 5.5.2.

Em nossos experimentos, destacamos dois exemplos, a regra 1 e a regra 2 [2], que mostram uma substancial melhoria sem a perda da simplicidade e robustez do nosso método. A modificação mais bem sucedida, a regra 1, produz um caminho de comprimento aproximado 40% menor que o produzido pelo potencial harmônico. Isto ocorre devido esta regra introduzir um campo de força que quebra a simetria da exploração do ambiente fazendo com que o robô escolha, sempre que possível, uma direção preferencial de movimento determinada pelo vetor $\mathbf{v}$. Mostramos que a escolha interna da direção deste vetor parece não afetar o desempenho do robô.

Neste ponto temos de um lado a regra 1 para ambientes esparsos e a equação de Laplace para ambientes densos. Como poderemos tratar eficientemente de ambientes internos compostos por regiões esparsas e densas automaticamente ? Observe que a regra 1 corresponde à equação de Laplace quando $\epsilon = 0$. Conseqüentemente, é possível criar um ϵ adaptativo que permita a transição suave entre diferentes funções que regem o campo potencial. Isto mantém a idéia de um príncipio único funcionando durante todo o processo de atuação do robô com o ambiente. O que corresponde a uma vantagem da nossa teoria em relação às demais, pois o processo de decisão binária, o qual é não-linear em sua natureza e passível a erros, não existe.

Em resumo, mostramos que modificando a equação diferencial do potencial, que é o núcleo do agente, pode-se obter uma melhoria significativa no processo de exploração de ambiente esparsos. É importante enfatizar que o potencial calculado a partir de um PVC, com a equação de Laplace ou suas modificações, é vantajoso quando comparado àquele calculado a partir de funções de custo cumulativo [THR 98] que usam de probabilidades de ocupação. Isto ocorre devido a esta probabilidade ser igual a zero longe dos obstáculos, o que resulta em regiões planas cujo gradiente é nulo. Conseqüentemente, devido ao gradiente ser nulo, não existe nenhum tipo de orientação para guiar o robô.

[2] É importante destacar que é possível utilizar outras regras, haja vista, que nossa teoria é representada por uma família de funções.

6.3.2 Formas de atualização do potencial

Nos experimentos realizados variamos a taxa de atualização r e o método de atualização do campo potencial. O valor ótimo para a taxa de atualização é um problema importante para implementações práticas, pois este valor está diretamente relacionado com o tempo de execução do sistema. Nos experimentos descritos testamos valores para $r = 10$ iterações/passo e $r = 30$ iterações/passo. A solução ideal é o desenvolvimento de uma taxa de atualização adaptativa, que mude de acordo com o progresso da exploração.

Em relação ao método de atualização, investigamos a utilização do método Gauss-Seidel e do método SOR. O número de iterações necessárias à convergência da equação de Laplace com erro abaixo de 10^{-p} é $n_{GS} \sim p\ M$ para Gauss-Seidel e $n_{SOR} \sim p\ \sqrt{M}$ para SOR, onde M é o número de células a serem atualizadas [PRE 92]. Apesar da clara vantagem da utilização do método SOR, nossos resultados mostram que a versão parcialmente relaxada tem qualidade significantemente inferior à versão gerada pelo método Gauss-Seidel com a mesma taxa de relaxamento.

Isto deriva do fato de que SOR faz atalhos para chegar na solução de equilíbrio ($\nabla^2 p(\mathbf{r}) = 0$) que implicam soluções intermediárias não confiáveis. De fato, durante os experimentos utilizando o método SOR observamos padrões semelhantes a ondas (ver Figura 6.1) que distorcem o campo vetorial produzindo caminhos que em alguns casos resultam em colisões (ver Figura 5.7(b)).

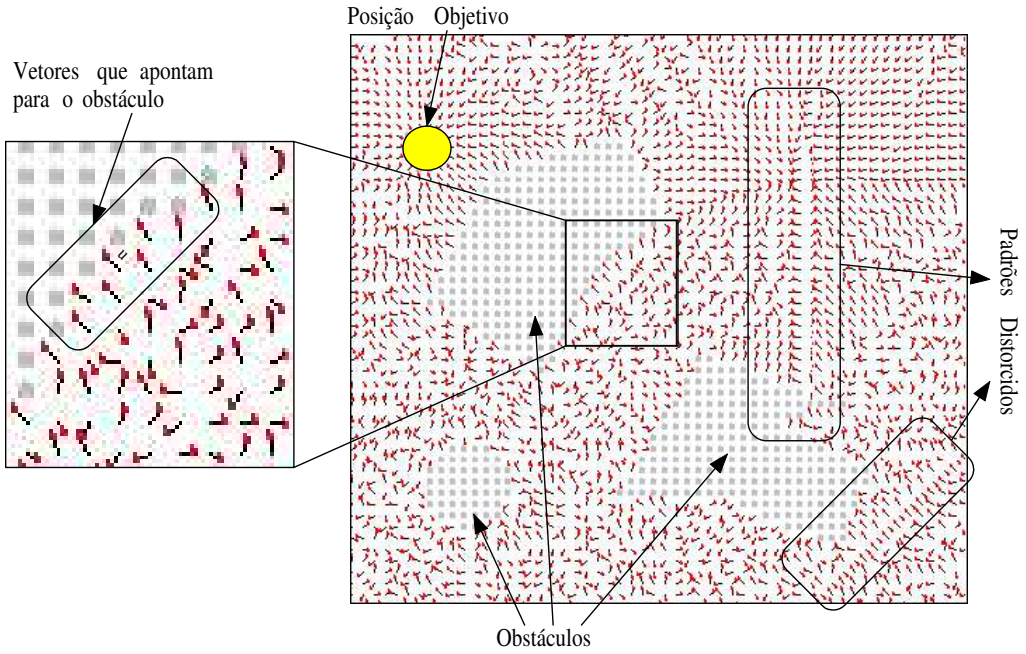


FIGURA 6.1 – Campo vetorial parcialmente relaxado utilizando o método SOR após 30 iterações de atualização. As setas indicam a posição dos obstáculos, objetivo, a presença de padrões distorcidos semelhantes à ondas, e de vetores que apontam em direção a um obstáculo.

Por outro lado, Gauss-Seidel resolve a equação de Laplace como uma versão discreta do problema da difusão $\partial p(\mathbf{r}, t)/\partial t = \nabla^2 p(\mathbf{r}, t)$. O que significa que soluções

intermediárias $p(\mathbf{r}, t)$ são consistentes com o problema físico de obstáculos repulsivos e objetivos atrativos. Conseqüentemente, isto nunca produzirá caminhos que terminam em colisões com paredes detectadas.

Um segundo problema é a qualidade dos caminhos produzidos. Pelas razões discutidas acima, Gauss-Seidel produz caminhos que são mais suaves que aqueles produzidos pelo SOR. Isto é refletido na Tabela 5.2, onde vemos que Gauss-Seidel com $r = 10$ iterações/passo é comparável ao SOR com $r = 30$ iterações/passo. A conclusão é que Gauss-Seidel é o método mais apropriado para atualizar o campo potencial quando versões parcialmente relaxadas são necessárias para controlar o robô em tempo real durante seu trajeto pelo ambiente.

Considerando o mecanismo de relaxação, observamos que é possível aumentar a eficiência deste processo reduzindo o número de células que são atualizadas por unidade de tempo através do chaveamento entre as janelas local e global do robô, chamado conhecimento adaptativo. Isto fez com que tivéssemos em média sobre 10 experimentos uma economia de aproximadamente 61% na quantidade de células relaxadas.

6.3.3 Comparação com outras técnicas

Além disso, comparamos nossa técnica com outras estratégias de exploração padrão. Devido a essa comparação ser dependente de elementos como implementação, recursos mecânicos, ambientes e robôs, optamos por fazer comparações com técnicas mais simples que são genéricas o suficiente para realizar uma implementação independente, neste caso a exploração aleatória e o método *wall following*.

A exploração aleatória é certamente a mais ineficiente de todas, entretanto ela serve como uma medida padrão para mensurar a complexidade do ambiente. Dessa forma, não estamos preocupados em superar os resultados gerados por ela.

O método *wall following* é uma técnica interessante para explorar obstáculos conectados. Contudo, falha ao explorar ambientes compostos por obstáculos não conectados como ilhas de paredes ou objetos. Conseqüentemente, este procedimento tem que ser associado com uma estratégia de busca alternativa, o que implica a adição de algum mecanismo que realize este chaveamento.

6.3.4 Tempo de exploração

Em relação ao tempo de exploração, observamos que o robô consegue explorar um ambiente pequeno em um tempo razoável, entretanto, convém ressaltar que os dados adquiridos são dependentes do comprimento do raio de ativação e da velocidade máxima do robô, pois quanto maior for o raio de ativação maior será a região explorada pelo robô a cada instante, conseqüentemente, mais rápido será o mapeamento do ambiente. Similarmente, quanto mais rápido o robô se deslocar mais rápido ele irá mapear completamente o ambiente, entretanto, é importante observar que a velocidade deve ser tratada com cautela, pois o rápido deslocamento exige uma atenção maior a eventos inesperados.

6.3.5 Erros de odometria

Ademais, observamos que a discrepância média se encontra dentro da faixa de 25 cm estipulada antes da exploração e representa aproximadamente $0,6\%$ do

comprimento total percorrido pelo robô. O que vem comprovar o que discutimos na Seção 4.2.2, onde afirmamos que um dos grandes benefícios do cálculo do potencial através de métodos de relaxação é a geração de trajetórias suaves que evitam mudanças abruptas na orientação do robô, uma das principais fontes de erros de odometria.

6.3.6 Ambientes dinâmicos

Testamos nosso sistema em ambientes dinâmicos e constatamos que a generalização do método HIMM permite um fácil tratamento tanto de obstáculos dinâmicos quanto estáticos através do incremento e do decremento da propriedade certeza de cada célula sobre o cone de visão dos sensores do robô. Um dos casos mais interessantes é ilustrado na Figura 5.11(f). Nesta situação o robô teve que lidar com obstáculos dinâmicos, pessoas, assim como realizar diversos movimentos que incluem translações e rotações.

6.3.7 Janela Adaptativa

Mostramos nas Figuras 5.15(a) e (b) que é possível através de uma janela de ativação de tamanho variável produzir um mapa mais confiável e preciso do ambiente que está sendo explorado. Isto é principalmente vantajoso em ambientes que possuem corredores estreitos, pois é possível realizar uma exploração mais refinada. Neste caso, em particular, uma janela de ativação de tamanho fixo pode produzir estados de aprisionamento como mostrado na Figura 5.15(d).

6.4 Trabalhos Futuros

Abaixo listamos alguns tópicos para trabalhos futuros utilizando as idéias propostas aqui:

- elaboração de uma taxa adaptativa para atualização do campo potencial que se altere de acordo com o progresso da exploração. A principal razão é que este valor é um problema importante para implementações práticas, pois ele está diretamente relacionado com o tempo de execução do sistema;

- elaboração de uma taxa adaptativa para o tempo de disparo dos sensores do robô. A principal razão é que esta taxa é dependente da velocidade do robô. Se o robô se move a uma velocidade alta é importante que os sensores sejam disparados em intervalos curtos de tempo, a fim de que ele consiga reagir rapidamente a eventos inesperados. No caso de velocidades baixas, o intervalo do disparo dos sensores deve ser aumentado, para que seja minimizado o efeito colateral produzido pela reflexão especular;

- averiguação de outras funções de atualização das células do mapa. Apesar do método HIMM ser limitado em relação à modelagem da incerteza gerada pelos sensores, sua utilização em conjunto com a atualização do campo potencial gera um modelo probabilístico poderoso. Entretanto, é importante testar outras funções de atualização a fim de verificar se é possível melhorar a precisão do mapa gerado;

- elaboração de uma estratégia de exploração multi-nível, para ambientes cuja dimensão é maior que a capacidade de mapeamento do robô. A idéia é permitir ao robô dividir o ambiente em diversas regiões, e em seguida fazê-lo explorar cada uma delas separadamente. Como o ambiente é dividido em diversas regiões, é possível elaborar uma estratégia utilizando um time de robôs para que a exploração destas regiões seja feita simultaneamente;

- modificação da representação baseada em grades para uma representação mais compacta a fim de aumentar a eficiência no processo de planejamento e exploração. Isto requer o estudo de técnicas de compactação da área de processamento de imagens;

- integração de sinais oriundos de diversos e diferentes sensores a fim de garantir maior precisão e confiança nos dados adquiridos pelo robô;

- elaboração de um módulo de auto-localização para ambientes internos e esparsos;

- exploração distribuída utilizando multi-robô;

- integração da representação baseada em grades e mapas probabilísticos durante o processo de exploração, a fim de garantir maior precisão tanto na localização do robô como na localização dos obstáculos;

- elaboração e implementação da nossa teoria em braços mecânicos com juntas de revolução;

- elaboração de uma estratégia para o problema de planificação decorrente do rápido decaímento das funções potenciais. Uma possível solução é o uso do método planejamento chamado, *propagação de distância*, em conjunto com o potencial gerado pelo PVC;

- investigar o uso de diferentes funções de atualização em diferentes partes do campo potencial durante o processo de relaxação;

- investigar a teoria da representação de forma baseada em curvatura [MOK 92] para o processo de auto-localização;

- investigar outras estratégias de ajuste automático do comprimento do raio da janela de ativação. Uma sugestão é o estudo mais aprofundado da utilização da média aritmética dos três menores retornos dos sensores do robô. Uma segunda sugestão é a utilização de um fator de elegibilidade para produzir uma transição suave e atenuar mudanças abruptas do comprimento do raio de ativação entre instantes sucessivos de tempo;

- permitir a utilização de um mapa prévio disponível da área de atuação do robô;

- determinação da complexidade da estrutura de ambientes através do conhecimento adaptativo, ou seja, através da quantidade de vezes que o robô realiza o chaveamento entre o conhecimento local e global durante sua exploração.

6.5 Contribuições da Tese

Este trabalho possui diversas contribuições tanto para nosso grupo de pesquisa quanto para a comunidade internacional de robótica. Entre elas podemos citar:

- é o primeiro trabalho desenvolvido no PGCC utilizando o robô NOMAD 200;
- durante o desenvolvimento desta tese foi construído um simulador em Delphi/windows para facilitar a manipulação com o primeiro robô testado, o robô khepera. Este simulador permite realizar testes e adquirir informações através dos sensores do robô. A comunicação entre o simulador e o robô é feita através de um cabo que conecta a porta serial do computador, que está executando o simulador, e o robô.
- a mudança do robô khepera para o robô NOMAD 200 permitiu o desenvolvimento de uma interface visual C++/linux que permite facilmente manipular os parâmetros do robô, adquirir e visualizar as informações sensoriais e o progresso da exploração;
- a estratégia para exploração autônoma baseada em problemas de valores de contorno é inédita;
- a estratégia corresponde a uma extensão do método de planejamentos de caminhos baseado em funções harmônicas. Devido a isso, seu princípio básico pode ser utilizado tanto com propósito navegacional exploratório quanto navegacional baseado em mapa;
- por ser baseada em um planejador de caminhos global, a estratégia fornece a base para estender outros métodos globais a fim de adicionarem em suas abordagens o comportamento exploratório;
- a utilização de métodos de relaxação para cálculo do campo potencial;
- a utilização do conhecimento adaptativo que garante um considerável aumento na eficiência do processo de atualização do campo potencial;
- a representação do ambiente é baseada em grades de ocupação e é atualizada com uma versão modificada do método HIMM. A versão inicial do HIMM considera apenas aquelas células sobre o eixo acústico dos sensores sonar do robô. Nossa versão considera todas as células sobre um cone de abertura α. Isto faz com que várias células sejam recrutadas e seja possível realizar o cálculo do campo potencial que guia o robô;
- o processo de parada do algoritmo de exploração é inspirado na estratégia de crescimento de regiões da área de processamento de imagens [GON 92]. Esta estratégia é executada concorrentemente com a exploração e é utilizada para identificar quando o ambiente foi completamente explorado pelo robô. Para fazer isso, ela segmenta o espaço livre conhecido e verifica se existe ainda alguma região não explorada fazendo fronteira. Caso esta verificação seja negativa, o processo de exploração é finalizado.

- os sinais sensoriais são refinados através da criação de uma janela de atualização de tamanho adaptativo. A janela adaptativa reduz os erros gerados pela versão modificada do HIMM.

- por ser um sistema exploratório e de planejamento de caminhos, ele possui mecanismos para realizar tarefas como: encontrar objetos e levá-los de uma posição a outra; adquirir a estrutura do ambiente onde está inserido; encontrar um caminho para um local já visitado, etc.

Bibliografia

[ALA 98] ALAMI, R. et al. An architecture for autonomy. **International Journal of Robotics Research**, [S.l.], 1998.

[ALB 89] ALBUS, J. S.; MCCAIN, H. G.; LUMIA, R. **Nasa/nbs standard reference model for tele-robot control system architecture (nasren)**. [S.l.]: National Institute of Standards and Technology, 1989.

[AMA 95] AMAT, J.; ESTEVA, F.; MáNTARAS, R. L. de. Autonomous navigation troup for cooperative modelling of unknown environments. In: INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS, 7., 1995, San Feliu de Guíxols, Spain. **Proceedings...** [S.l.: s.n.], 1995. v.1, p.383–389.

[BAI 2001] BAILEY, T.; NEBOT, E. Localisation in large-scale environments. **Robotics and Autonomous Systems**, [S.l.], v.37, p.261–281, Dec. 2001.

[BET 94] BETGé-BREZETZ, S. et al. **Adaptive localization of an autonomous mobile robot in natural environments**. Grenoble,France: Laboratoire d'informatique fondamentale et d'intelligence artificielle, 1994.

[BET 94a] BETKE, M.; GURVITS, L. Mobile robot localizaton using landmarks. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, IROS, 1994, Munich, Germany. **Proceedings...** [S.l.: s.n.], 1994. p.135–142.

[BET 97] BETKE, M.; GURVITS, L. Mobile robot localization using landmarks. **IEEE Transactions on Robotics and Automation**, [S.l.], v.13, n.2, p.251–263, 1997.

[BET 95] BETKE, M.; RIVEST, R.; SINGH, M. Piecemeal learning of an unknown environment. **Machine Learning**, [S.l.], v.18, n.2-3, p.1–24, Feb. 1995.

[BOR 89] BORENSTEIN, J.; KOREN, Y. Real-time obstacles avoidance for fast mobile robots. **IEEE Transactions on Systems, Man and Cybernetics**, [S.l.], v.19, n.5, p.1179–1187, Sept./Oct. 1989.

[BOR 91] BORENSTEIN, J.; KOREN, Y. Histogramic in-motion mapping for mobile robot obstacle avoidance. **IEEE Journal of Robotics and Automation**, [S.l.], v.7, n.4, p.535–539, 1991.

[BOR 91a] BORENSTEIN, J.; KOREN, Y. The vector field histogram-fast obstacle avoidance for mobile robots. **IEEE Journal of Robotics and Automation**, [S.l.], v.7, n.3, p.278–288, 1991.

[BRO 86] BROOKS, R. A. A robust layered control system for a mobile robot. **IEEE Journal of Robotics and Automation**, [S.l.], 1986.

[BUG 99] BUGARD, W. et al. Experiences with an interactive museum tour-guide robot. **Artificial Intelligence**, [S.l.], v.114, n.1-2, p.3–55, 1999.

[CAR 92] CARVER, N.; LESSER, V. **The evolution of blackboard control architectures**. [S.l.]: University of Massachusetts, 1992.

[CHA 96] CHATILA, R. et al. **Autonomous mobile robot navigation for planet exploration the eden project**. [S.l.]: Laboratoire d'analyse et d'Architecture des Systèmes, 1996.

[CHA 89] CHATILA, R.; LAUMOND, J.-P. Position referencing and consistent world modeling. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA, 1989. **Proceedings...** [S.l.: s.n.], 1989.

[CHO 2000] CHOSET, H.; BURDICK, J. Sensor based motion exploration: the hierarchical generalized voronoi graph. **International Journal of Robotics Research**, [S.l.], v.19, n.2, p.119–148, Feb. 2000.

[CHO 2000a] CHOSET, H. et al. Sensor-based exploration: incremental construction of the hierarchical generalized voronoi graph. **International Journal of Robotics Research**, [S.l.], v.19, n.2, p.126–148, Feb. 2000.

[CHO 2001] CHOSET, H.; NAGATANI, K. Topological simultaneous localization anf mapping (slam): toward exact localization without explicit localization. **IEEE Transactions on Robotics and Automation**, [S.l.], v.17, n.2, p.125–137, Apr. 2001.

[COH 96] COHEN, W. W. Adaptive mapping and navigation by teams of simple robots. **Robotics and Autonomous Systems**, [S.l.], v.18, p.411–434, 1996.

[CON 94] CONNOLLY, C. Harmonic functions and collision probabilities. **International Journal of Robotics Research**, [S.l.], p.497–507, 1994.

[CON 93] CONNOLLY, C.; GRUPEN, R. On the application of harmonic functions to robotics. **Journal of Robotic Systems**, [S.l.], v.10, p.931–946, 1993.

[COU 62] COURANT, R.; HILBERT, D. **Methods of mathematical physics**. [S.l.]: New York: John Wiley and Sons, 1962. v.2.

[CRO 85] CROWLEY, J. L. World modeling and position estimation for a mobile robot using ultrasonic ranging. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA,

1985, Scottsdale, Arizona. **Proceedings...** [S.l.: s.n.], 1985. p.674–680.

[CRO 85a] CROWLEY, J. L.; COUTAZ, J. **Navigation et modélisation pour un robot mobile**. Grenoble,France: Laboratoire d'informatique fondamentale et d'intelligence artificielle, 1985.

[CSO 97] CSORBA, M. **Simultaneos localisation and map building**. 1997. Tese (Doutorado em Ciência da Computação) — Departament of Engineering Science, University of Oxford.

[DAM 98] DAM, J. V. **Environment modelling for mobile robots**: neural learning for sensor fusion. 1998. Tese (Doutorado em Ciência da Computação) — Universiteit van Amsterdam, Faculteit WINS, Amsterdam, The Netherlans.

[DEV 95] DEVY, M. et al. On autonomous navigation in natural environment. **Robotics and Autonomous Systems**, [S.l.], v.16, p.5–16, 1995.

[DIS 2001] DISSANAYAKE, M. W. M. G. et al. A solution to the simultaneous localization and map building (slam) problem. **IEEE Transactions on Robotics and Automation**, [S.l.], v.17, n.3, p.229–241, June 2001.

[DUD 93] DUDEK, G.; FREEDMAN, P.; HADJRES, S. Using local information in a non-local way for mapping graph-like worlds. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1993, Los Altos, CA. **Proceedings...** [S.l.]:Morgan Kaufmann, 1993. p.1639–1647.

[ELF 87] ELFES, A. Sonar-based real world mapping and navigation. **IEEE Journal of Robotics and Automation**, [S.l.], v.RA-3, n.3, p.249–265, 1987.

[ELF 89] ELFES, A. Using occupancy grids for mobile robot perception and navigation. **Computer Magazine**, [S.l.], p.46–57, June 1989.

[ENG 94] ENGELSON, S. **Passive map learning and visual place recognition**. 1994. Tese (Doutorado em Ciência da Computação) — Departament of Computer Science, Yale University, New Haven, CT.

[FED 99] FEDER, H. J. S.; LEONARD, J. J.; SMITH, C. M. Adaptive mobile robot navigation and mapping. **International Journal of Robotics Research**, [S.l.], v.18, n.7, p.650–668, July 1999.

[FEY 72] FEYNMAN, R. P.; LEIGHTON, R. B.; SANDS, M. **The feynman lectures on physics**. [S.l.]: Addison-Wesley, 1972. v.2.

[FOL 90] FOLEY, J. et al. **Computer graphics - principes and practice**. [S.l.]: Addison-Wesley, 1990.

[FOX 99] FOX, D.; BURGARD, W.; THRUN, S. Probabilistic methods for mobile robot mapping. In: IJCAI WORKSHOP ON ADAPTIVE SPATIAL REPRESENTATIONS OF DYNAMIC ENVIRONMENTS, 1999. **Proceedings...** [S.l.: s.n.], 1999.

[GOL 2001] GOLFARELLI, M.; MAIO, D.; RIZZI, S. Correction of dead-reckoning errors in map building for mobile robots. **IEEE Transactions on Robotics and Automation**, [S.l.], v.17, n.1, p.37–47, Feb. 2001.

[GON 92] GONZALEZ, R.; WOODS, R. **Digital image processing**. [S.l.]: Addison-Wesley, 1992.

[GUI 2001] GUIVANT, J. E.; NEBOT, E. M. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. **IEEE Transactions on Robotics and Automation**, [S.l.], v.17, n.3, p.242–257, June 2001.

[HEB 95] HEBERT, P.; BETGé-BREZETS, S.; CHATILA, R. **Probabilistic map learning**: necessity and difficulties. Toulouse Cedex, France: Laboratoire d'Analyse et d'Architecture des Systèmes, 1995.

[HOW 96] HOWARD, A.; KITCHEN, L. Generating sonar maps in highly specular environments. In: INTERNATIONAL CONFERENCE ON CONTROL,AUTOMATION, ROBOTICS AND VISION,4., 1996. **Proceedings...** [S.l.: s.n.], 1996.

[IDI 2002] IDIART, M. A. P.; TREVISAN, M. Directing a random walker with optimal potentials. **Physica A : Statistical Mechanics and its Application**, [S.l.], v.307, n.1-2, p.52–62, 2002.

[KHA 96] KHATIB, M. **Sensor based motion control for mobile robots**. 1996. Tese (Doutorado em Ciência da Computação) — Laboratoire d'Automatique et d'Analise des Systèmes, Toulouse, France.

[KRE 66] KREIDER, D. L. et al. **An introduction to linear analysis**. [S.l.]: Addison-Wesley, 1966. v.3.

[KUI 91] KUIPERS, B.; BYUN, Y.-T. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. **Robotics and Autonomous Systems**, [S.l.], v.8, p.47–63, 1991.

[KUI 88] KUIPERS, B.; LEVITT, T. Navigation and mapping in large-scale space. **AI Magazine**, [S.l.], v.9, p.25–43, 1988.

[LAT 93] LATOMBE, J.-C. **Robot motion planning**. Assinippi, Norwell, Massachusetts: Kluwer Academic Publishers, 1993.

[LEE 96] LEE, W. **Spatial semantic hierarchy for a physical robot**. 1996. Tese (Doutorado em Ciência da Computação) — Departament of Computer Sciences, The University of Texas, Austin.

[LEO 92] LEONARD, J. J.; DURRANT-WHYTE, H. F.; COX, I. J. Dynamic map building for an autonomous robot. **International Journal of Robotics Research**, [S.l.], v.11, n.4, p.286–298, 1992.

[LU 97] LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. **Autonomous Robots**, [S.l.], v.4, n.4, p.333–349, 1997.

[MAJ 2000] MAJUMDER, S. et al. Map building and localization for underwater navigation. In: INTERNATIONAL SYMPOSIUM ON EXPERIMENTAL ROBOTICS, 2000. **Proceedings...** [S.l.: s.n.], 2000.

[MAT 90] MATARIC, M. J. **A model for distributed mobile robot environment learning and navigation**. 1990. Dissertação (Mestrado em Ciência da Computação) — MIT Artificial Intelligence Laboratory, Cambridge, MA.

[MCK 95] MCKERROW, P. J. **Introduction to robotics**. [S.l.]: Addison-Wesley, 1995.

[MED 98] MEDEIROS, A. A. D. A survey of control architectures for autonomous mobile robots. **Journal of the Brazilian Computer Society**, [S.l.], v.4, n.3, 1998.

[MEK 97] MEKREZ, I. **Application de reseaux de neurones recurrents au controle d'un robot autonome**. 1997. Dissertação (Mestrado em Ciência da Computação) — Institut National Polytechnique de Grenoble, Grenoble.

[MOK 92] MOKHTARIAN, F.; MACKWORTH, A. K. A theory of multiscale, curvature-based shape representation for planar curves. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Los Alamitos, v.14, n.8, p.789–805, Aug. 1992.

[MOR 85] MORAVEC, H. P.; ELFES, A. High resolution maps from wide angle sonar. In: IEEE INTERNATIONAL CONFERENCE OF ROBOTICS AND AUTOMATION, ICRA, 1985. **Proceedings...** [S.l.: s.n.], 1985.

[MUR 96] MURPHY, R. R.; HUGHES, K.; NOLL, E. An explicit path planner to facilitate reactive control and terrain preferences. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.2067–2072.

[NIL 98] NILSSON, N. J. **Artificial intelligence**: a new synthesis. [S.l.]: Morgan Kaufmann, 1998.

[O'KE 96] O'KEEFE, J.; BURGESS, N. Geometric determinants of the place fields of hippocampal neurons. **Nature**, [S.l.], v.381, p.425–428, 1996.

[OLI 2000] OLIVEIRA FORTUNA, A. de. **Técnicas computacionais para dinâmica de fluidos** : conceitos básicos e aplicações. [S.l.]: Ed. da Universidade de São Paulo, 2000.

[OLS 97] OLSON, C. F. Mobile robot self-localization by iconic matching of range maps. In: INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.447–452.

[OLS 99] OLSON, C. F. Subpixel localization and uncertainty estimation using occupancy grids. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.447–452.

[OLS 98] OLSON, C. F.; MATTHIES, L. H. Maximum likelihood rover localization by matching range maps. In: INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.272–277.

[ORI 97] ORIOLO, G.; ULIVI, G.; VANDITTELLI, M. Fuzzy maps : a new tool for mobile robot perception and planning. **Journal of Robotic Systems**, [S.l.], v.14, n.3, p.179–197, 1997.

[PET 97] PETTERSSON, L. **Control system architecture for autonomous agents**. [S.l.]: Department of Machine Design, KTH, 1997.

[POM 89] POMERLEAU, D. A. **Alvinn**: an autonomous land vehicle in a neural network. [S.l.]: Carnegie Melon University, 1989. (CMU-CS-89-107).

[PRE 92] PRESS, W. et al. **Numerical recipes in c**: the art of scientific computing. [S.l.]: Cambridge University Press, 1992.

[PRE 2001] PRESTES, E. et al. Exploration technique using potential fields calculated from relaxation methods. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, IROS, 2001, Havaí, EUA. **Proceedings...** [S.l.: s.n.], 2001. p.2012–2017.

[PRE 2002] PRESTES, E. et al. Exploration method using harmonic functions. **Robotics and Autonomous Systems**, [S.l.], v.40, n.1, p.25–42, July 2002.

[PRE 2002a] PRESTES, E. et al. Oriented exploration in non-oriented sparse environment. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, IROS, 2002, Lausanne, Suíça. **Proceedings...** [S.l.: s.n.], 2002. p.2353–2358.

[RAO 93] RAO, N. S. V. et al. **Robot navigation in unknown terrains**: introductory survey of non-heuristic algorithms. Oak Ridge, Tennessee: Oak Ridge National Laboratory, 1993. (ORNL/TM-12410).

[RAS 90] RASCHKE, U.; BORENSTEIN, J. A comparation of grid-type map-building techniques by index of performance. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA, 1990, Cincinnati, Ohio. **Proceedings...** [S.l.: s.n.], 1990.

[RAT 95] RATERING, S.; GINI, M. Robot navigation in a known environment with unknown moving obstacles. **Autonomous Robot**, [S.l.], v.1, p.149–165, 1995.

[REK 97] REKLEITIS, I. M.; DUDEK, G.; MILIOS, E. E. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1997, Nagoya, Japão. **Proceedings...** [S.l.: s.n.], 1997. v.2, p.1340–1346.

[ROM 2001] ROMERO, L. **Construcción de mapas y localización de robots móviles**: un enfoque híbrido. 2001. Tese (Doutorado em Ciência da Computação) — TEC de Monterrey, Cuernavaca, Morelos.

[ROM 2000] ROMERO, L.; MORALES, E.; SUCAR, E. A robust exploration and navigation approach for indoor mobile robots merging local and global strategies. In: IBERAMIA-SBIA:ADVANCES IN ARTIFICIAL INTELLIGENCE, 2000, [S.l.]. **Proceedings...** Berlin: Springer-Verlag, 2000. p.389–398.

[ROM 2001a] ROMERO, L.; MORALES, E.; SUCAR, E. Building maps for indoor mobile robots using ultrasonic and laser range sensors. **Computación y Sistemas**, [S.l.], Sept. 2001.

[ROY 97] ROY, N.; DUDEK, G. Learning to rendezvous during multi-agent exploration. In: EUROPEAN WORKSHOP ON LEARNING ROBOTS, 6., 1997, Brighton, UK. **Proceedings...** [S.l.: s.n.], 1997.

[ROY 2001] ROY, N.; DUDEK, G. Collaborative robot exploration and rendezvous:algorithms, performance bounds and observations. **Autonomous Robot**, [S.l.], v.11, n.2, p.117–136, 2001.

[ROY 99] ROY, N. et al. Coastal navigation - mobile robot navigation with uncertainty in dynamic environments. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA, 1999, Detroit, MI. **Proceedings...** [S.l.: s.n.], 1999.

[RUS 95] RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. [S.l.]: Prentice Hall, 1995.

[SCH 99] SCHULTZ, A. C. et al. Unifying exploration, localization, navigation and planning through a common representation. **Autonomous Robots**, [S.l.], v.6, n.3, p.293–308, May 1999.

[SHA 97] SHATKAY, H.; KAELBLING, L. P. Learning topological maps with weak local odometric information. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1997. **Proceedings...** [S.l.: s.n.], 1997. v.2, p.920–929.

[SMI 97] SMITH, C. et al. Feature-based concurrent mapping and localization for autonomous underwater vehicles. In: IEEE OCEANS, 1997. **Proceedings...** [S.l.: s.n.], 1997.

[SMI 87] SMITH, R.; SELF, M.; CHEESEMAN, P. A stochastic map for uncertain spatial relationships. In: IEEE INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH, 1987, Santa Cruz,CA. **Proceedings...** [S.l.: s.n.], 1987. p.467–474.

[STE 94] STENTZ, A. Optimal and efficient path planning for partially-known environments. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA, 1994. **Proceedings...** [S.l.: s.n.], 1994. v.4, p.3310–3317.

[STE 95] STENTZ, A. The focussed d* algorithm for real-time replanning. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 1995. **Proceedings...** [S.l.: s.n.], 1995. p.1652–1659.

[SUT 98] SUTTON, R.; BARTO, A. G. **Reinforcement learning**: an introduction. Cambridge,MA: MIT Press, 1998.

[SUT 88] SUTTON, R. S. Learning to predict by the method of temporal differences. **Machine Learning**, [S.l.], v.3, p.9–44, 1988.

[THR 98] THRUN, S. Learning maps for indoor mobile robot navigation. **Artificial Intelligence**, [S.l.], v.99, n.1, p.21–71, 1998.

[THR 96] THRUN, S.; BÜCKEN, A. Integrating grid-based and topological maps for mobile robot. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 13., 1996. **Proceedings...** [S.l.: s.n.], 1996. p.944–950.

[ULL 87] ULLRICH, R. A. **Robótica**: uma introdução (o porquê dos robôs e seu papel no trabalho). [S.l.]: Campus, 1987.

[VEL 84] VELOSO, P. et al. **Estruturas de dados**. [S.l.]: Campus, 1984.

[WEL 2001] WELCH, G.; BISHOP, G. **An introduction to the kalman filter**. Chapel Hill, NC: University of North Carolina, 2001. (TR 95-041).

[YAM 97] YAMAUCHI, B. A frontier based exploration for autonomous exploration. In: IEEE INTERNATIONAL SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE IN ROBOTICS AND AUTOMATION, 1997, Monterey,CA. **Proceedings...** [S.l.: s.n.], 1997. p.146–151.

[YAM 98] YAMAUCHI, B. Frontier-based exploration using multiple robots. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 2., 1998, Minneapolis. **Proceedings...** [S.l.: s.n.], 1998. p.47–53.

[YAM 98a] YAMAUCHI, B. et al. Integrating map learning, localization and planning in a mobile robot. In: IEEE INTERNATIONAL SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE IN ROBOTICS AND AUTOMATION, 1998, Gaithersburg,MD. **Proceedings...** [S.l.: s.n.], 1998. p.331–336.

[ZEL 98] ZELEK, J. S. A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments. In: AIPS-98 WORKSHOP ON INTEGRATING PLANNING, SCHEDULING & EXECUTION IN DYNAMIC & UNCERTAIN ENVIRONMENTS, 1998, Pittsburgh, PA. **Proceedings...** [S.l.: s.n.], 1998.

[ZEL 92] ZELINSKY, A. A mobile robot exploration algorithm. **IEEE Transactions on Robotics and Automation**, [S.l.], v.8, n.6, p.707–717, Dec. 1992.