



DIEGO PEREIRA RODRIGUES

**Estudo Comparativo de Métodos de Localização
para Robôs Móveis Baseados em Mapa**

Campinas, SP
2013



UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE
COMPUTAÇÃO

Estudo Comparativo de Métodos de Localização para Robôs Móveis Baseados em Mapa

Autor: Diego Pereira Rodrigues

Orientador: Prof. Dr. Eleri Cardozo

Dissertação de Mestrado apresentada ao programa de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Engenharia Elétrica na área de concentração: Engenharia de Computação.

Este exemplar corresponde à versão final da dissertação defendida por Diego Pereira Rodrigues, e orientada pelo Prof. Eleri Cardozo.

Campinas, SP
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

R618e Rodrigues, Diego Pereira, 1986-
Estudo comparativo de métodos de localização para robôs móveis baseados em mapa / Diego Pereira Rodrigues. – Campinas, SP : [s.n.], 2013.

Orientador: Eleri Cardozo.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Automação. 2. Robôs móveis. 3. Localização. I. Cardozo, Eleri, 1954-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Comparative study of localization methods for mobile robots based on map

Palavras-chave em inglês:

Automation

Mobile robotics

Localization

Área de concentração: Engenharia de Computação

Titulação: Mestre em Engenharia Elétrica

Banca examinadora:

Eleri Cardozo [Orientador]

Cairo Lucio Nascimento Junior

Ely Carneiro de Paiva

Data de defesa: 12-07-2013

Programa de Pós-Graduação: Engenharia Elétrica


COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Diego Pereira Rodrigues

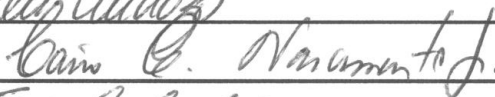
Data da Defesa: 12 de julho de 2013

Título da Tese: "Estudo Comparativo de Métodos de Localização para Robôs Móveis Baseados em Mapa"

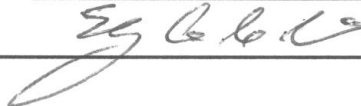
Prof. Dr. Eleri Cardozo (Presidente):



Prof. Dr. Cairo Lucio Nascimento Junior:



Prof. Dr. Ely Carneiro de Paiva:



Resumo

Esta dissertação trata de uma pesquisa sobre localização robótica baseada em mapas, analisando aspectos importantes do tema e discutindo alternativas para alguns dos problemas corriqueiramente encontrados. Todos os métodos trabalhados nesta dissertação utilizam uma abordagem probabilística, visto que isto permite ao sistema incorporar a incerteza existente em diversas partes do processo, tornando-o mais robusto e confiável. As técnicas trabalhadas são o Filtro de Bayes, Localização por Markov, Filtro de Kalman dinâmico, Filtro de Kalman estendido e Filtro de Partículas (Monte Carlo), realizando os experimentos em ambiente real, e as aplicando em mapas do tipo métrico e em grade. São abordados também alguns pontos cruciais para o funcionamento das técnicas, como o modelamento dos erros dos sensores e como a extração de características do ambiente foi implementada. Por fim, é feita uma comparação entre estas técnicas, explorando pontos-chaves com o propósito de contribuir para a literatura do tema. Esta comparação discute aspectos quantitativos, como erro de estimação originado de cada método e o tempo de execução de seus algoritmos; e aspectos qualitativos, como a dificuldade de implementação e deficiências de cada uma delas. Para a realização deste estudo, utilizou-se de uma plataforma robótica.

Palavras-chave: Localização, Robótica Móvel, Probabilística, Mapas.

Abstract

This dissertation deals with a research about robotics localization based on maps, analyzing important aspects of the subject and discussing alternatives to some of the problems routinely found. All methods employed in this paper use a probabilistic approach, since it allows the system to incorporate the uncertainty that exist in different parts of the process, making it more robust and reliable. The implemented techniques are the Bayesian filter, Localization by Markov, dynamic Kalman Filter, extended Kalman filter and Particle Filter (Monte Carlo), performing experiments in a real environment, using metric and grid maps. It also addressed some crucial points for the well functioning of the techniques such as the error modeling of sensors and the explanation on how the feature extraction from the environment was fulfilled. Finally, a comparison between these techniques is made, exploring key issues in order to contribute to the theme. This comparison discusses quantitative aspects such as the estimated error originated from each method and the runtime of their algorithm, and also qualitative aspects, such as the difficulty of implementation and deficiencies of each. For this study, a robotic platform was used.

Keywords: Localization, Mobile Robotics, Probabilistics, Maps.

Agradecimentos

Aos meus pais, Félix e Roseclê, pelo exemplo de vida, e à minha irmã Larissa, pela amizade verdadeira. Minhas vitórias só são possíveis graças a vocês. Obrigado pelo apoio incondicional.

Ao Prof. Dr. Eleri Cardozo, pela orientação sempre amigável e sincera, pelos conselhos, compreensão e motivação. Obrigado pela oportunidade de trabalhar com um grupo tão bom e diversificado.

À Emília, que me acompanhou de perto durante toda esta fase da minha vida, e de quem sempre recebi incentivos. Sua compreensão e paciência foram fundamentais.

Aos amigos que estiveram presentes, Johannes, Eduardo e Tiago, e que contribuíram para fazer destes, excelentes anos.

Aos companheiros de batalha do LCA, Guilherme, Fábio, Ricardo, Fernando, Leonardo, Eric, Lucio, Luisa, Victor, Mariana, Natasha, Leandro, Klaus e Prof^a Eliane, por tanto me ajudarem a tornar este trabalho realizável; além, é claro, dos momentos de descontração.

E à CAPES, pelo apoio financeiro.

*“Every day I watched closely for the sun or stars to appear,
to correct my chronometer, on the accuracy of which our
lives and the success of the journey would depend.”*

(F.A. Worsley, 1977 [1])

Sumário

Lista de Algoritmos	xix
Lista de Figuras	xxi
Lista de Tabelas	xxiii
Glossário	xxv
Trabalhos Publicados Pelo Autor	xxvii
1 Introdução	1
1.1 Projeto DesTINe	1
1.2 Motivações	2
1.3 Objetivos	4
1.3.1 Objetivo Geral	4
1.3.2 Objetivos Específicos	5
1.4 Contribuição	6
1.5 Cenário e Navegação	6
1.6 Apresentação da Dissertação	9
2 Localização para Robôs Móveis	13
2.1 Elementos da Localização Robótica	13
2.2 Taxonomias Utilizadas	15

2.2.1	Localização com Respeito à Informação Disponível	15
2.2.2	Ambientes Estáticos x Dinâmicos	16
2.2.3	Abordagem Passiva x Ativa	16
2.2.4	Único Robô x Vários Robôs	17
2.3	Métodos de Localização Robótica	17
2.4	Incertezas que Impactam na Localização Robótica	18
2.5	Extração de Características do Ambiente	20
2.6	Fusão de Sensores	20
2.7	Infraestrutura Experimental	22
3	Sensores: Extração de Características e Modelagem	27
3.1	Classificações e Características de Sensores	27
3.2	Modelagem dos Sensores	30
3.2.1	Calibração	31
3.2.2	Modelagem Estática	32
3.2.3	Modelagem do Erro Cinemático	33
3.3	Extração das Características	37
3.3.1	Variação do Número de Passos	39
3.3.2	Variação do <i>Threshold</i>	41
3.4	Conclusão	41
4	Filtro de Bayes	45
4.1	Introdução	45
4.2	Descrição e Implementação	46
4.3	Experimento e Análise dos Resultados	51
4.4	Reconhecimento de uma Topologia	53
4.5	Conclusão	54

5	Localização pelo método de Markov	57
5.1	Introdução	57
5.2	Descrição e Implementação	58
5.3	Experimento e Análise dos Resultados	61
5.4	Conclusão	62
6	Filtros Gaussianos	67
6.1	Introdução	67
6.2	Descrição e Implementação	68
6.2.1	Filtro de Kalman Dinâmico	69
6.2.2	Filtro de Kalman Estendido	70
6.2.3	Implementação	73
6.3	Experimento e Análise dos Resultados	74
6.4	Conclusão	75
7	Filtro de Partículas - Monte Carlo	77
7.1	Introdução	77
7.2	Descrição e Implementação	80
7.3	Experimento e Análise dos Resultados	83
7.4	Conclusão	90
8	Comparação entre Métodos	91
8.1	Comparação Quantitativa	91
8.1.1	Eficiência	92
8.1.2	Tempo de Execução	94
8.1.3	Consumo de CPU	96
8.1.4	Memória Alocada	97
8.1.5	Linhas de Código	98

8.2	Comparação Qualitativa	99
8.2.1	Dificuldade de Implementação	99
8.2.2	Deficiências	101
9	Conclusão	105
9.1	Considerações Finais	105
9.2	Contribuições	106
9.3	Trabalhos Futuros	106
	Referências	116

Lista de Algoritmos

1	Split	38
2	Merge	39
3	Filtro de Bayes ($bel(x_{t-1}), u_t, z_t$)	51
4	Localização pelo método de Markov ($bel(x_{t-1}), u_t, z_t, m$)	60
5	Filtro de Kalman Dinâmico ($\mu_{t-1}, \sum_{t-1}, u_t, z_t$)	70
6	Filtro de Kalman Estendido ($\mu_{t-1}, \sum_{t-1}, u_t, z_t$)	72
7	Monte Carlo (X_{t-1}, u_t, z_t, m)	83
8	Reamostragem	84

Lista de Figuras

1.1	Sistemas de coordenadas e postura do robô	3
1.2	Laboratório LCA	8
1.3	Planta baixa do corredor do laboratório LCA	9
2.1	Robô P3-DX	24
3.1	Cenário usado na modelagem estática	33
3.2	Leituras do Laser	33
3.3	Leituras do Sonar	34
3.4	Média do erro das leituras do laser com relação à distância ao obstáculo	34
3.5	Variância do erro das leituras do laser com relação à distância ao obstáculo	35
3.6	Média do erro das leituras do sonar com relação à distância ao obstáculo	35
3.7	Variância do erro das leituras do sonar com relação à distância ao obstáculo	36
3.8	Erro de odometria oriundo da translação do robô	37
3.9	Erro de odometria oriundo da rotação do robô	38
3.10	Split-and-Merge	39
3.11	Split-and-Merge de acordo com o quantidade de passos para o laser	40
3.12	Threshold (<i>mm</i>) x número de segmentos	41
3.13	Split-and-Merge de acordo com o valor de <i>threshold</i>	42
4.1	Exemplo clássico do uso de localização robótica por filtro de Bayes	47
4.2	Ângulos de Leitura do Laser para o Filtro de Bayes	48
4.3	Distribuição Multimodal das portas no corredor	50

4.4	Evolução da crença em ambiente real	53
4.5	Topologias usadas	54
5.1	Ângulos de Leitura do Laser para a Localização pelo método de Markov	59
5.2	Distribuição da característica específica (\parallel)	61
5.3	Evolução da crença pelo método de Markov	66
6.1	Ângulos de Leitura do Laser para os filtros Gaussianos	70
7.1	Ângulos de Leitura do Laser para o método de Monte Carlo	81
7.2	Curva Sigmoid que define a concentração ou dispersão das partículas	83
7.3	Evolução da crença por método de Monte Carlo	88
9.1	Possíveis Trabalhos Futuros	106

Lista de Tabelas

1.1	Dados sobre o corredor	8
1.2	Dados sobre a navegação	10
2.1	Exemplos de erros sistemáticos e não sistemáticos na odometria de um robô móvel. .	19
2.2	Informações físicas do robô.	25
3.1	Características de performance do laser e sonar	30
4.1	Características do Método de Filtro de Bayes Implementado	46
4.2	Resultados dos experimentos do Filtro de Bayes	52
4.3	Medições feitas com base no Filtro de Bayes	52
4.4	Estimação de topologias quando o robô alcança seu centro [2]	55
5.1	Características do Método de Markov Implementado	58
5.2	Resultados dos experimentos da Localização pelo método de Markov	62
5.3	Medições feitas com base na localização pelo método de Markov	62
6.1	Características dos Métodos de Filtro de Kalman Implementados	68
6.2	Resultados dos experimentos de Kalman dinâmico	75
6.3	Resultados dos experimentos de Kalman estendido	75
6.4	Resultados dos experimentos da Odometria	75
6.5	Medições feitas com base no filtro de Kalman dinâmico	75
6.6	Medições feitas com base no filtro de Kalman estendido	75

7.1	Características do Método de Monte Carlo Implementado	80
7.2	Parâmetros do método de Monte Carlo	83
7.3	Resultados dos experimentos do método de Monte Carlo	89
7.4	Medições feitas com base no método de Monte Carlo	90
8.1	Características dos Métodos de Localização Implementados	91
8.2	Tabela contendo os resultados obtidos em ambiente real para o erro de posição. . . .	93
8.3	Tabela contendo os resultados obtidos em ambiente real para o erro de orientação. . .	94
8.4	Tabela contendo os tempos de execução das técnicas.	96
8.5	Tabela contendo os valores de consumo de CPU das técnicas.	97
8.6	Tabela contendo a quantidade de memória alocada nas técnicas.	98
8.7	Tabela contendo a quantidade de linhas de código das técnicas.	99

Glossário

API - Application Programming Interfaces

ARCOS - Advanced Robotics Control and Operations

ARIA - Advanced Robotics Interface for Applications

BCI - Brain-Computer Interface

CPU - Central Processing Unit

DesTINe - Desenvolvimento de Tecnologias da Informação para Neurologia

FEEC - Faculdade de Engenharia Elétrica e de Computação

HTTP - Hypertext Transfer Protocol

IC - Intervalo de Confiança

IMM - Interacting Multiple Model

LCA - Laboratório de Engenharia da Computação e Automação Industrial

ME - Micro Edition

SLAM - Simultaneous Localization and Mapping

UNICAMP - Universidade Estadual de Campinas

XML - Extensible Markup Language

Trabalhos Publicados Pelo Autor

1. D. Rodrigues, L. Olivi, R. S. Souza, E. Guimarães e E. Cardozo, “Continuous Topology Learning And Early Recognition For Mobile Robots Navigation”. *XIX Congresso Brasileiro de Automática (CBA’12)*, Campina Grande, PB, Setembro, 2012.
2. L. Agostinho, L. Olivi, G. Feliciano, F. Paolieri, D. Rodrigues, E. Cardozo, e E. Guimarães, “A Cloud Computing Environment for Supporting Networked Robotics Applications”. *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC’11)*, Sysney, Australia, pg. 1110 - 1116, December, 2011.
3. R. S. Souza, L. Agostinho, F. Teixeira, D. Rodrigues, L. Olivi, E. Guimarães e E. Cardozo. “Control of Mobile Robots Through Wireless Sensor Networks”. *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC’11)*, Campo Grande, MS, pg. 805-818, Maio, 2011.
4. L. Rocha, L. Olivi, F. Paolieri, G. Feliciano, R. Souza, F. Pinho, F. Teixeira, D. Rodrigues, E. Guimarães e E. Cardozo, “Computer Communications and Networks - Advances in Educational Robotics in Cloud with Qualitative Scheduling in WorkFlows”, Springer, University of Derby, 2011.

Capítulo 1

Introdução

A robótica é um campo de estudo com aplicações em diversas áreas científicas, desde entretenimento até exploração espacial [3,4]. Dentre as subdivisões deste campo destaca-se a robótica móvel, que por ser peça fundamental em diversas aplicações, tornou-se tema deste estudo.

Robótica móvel refere-se à movimentação (navegação) de robôs, e consiste basicamente de quatro atividades: percepção, localização, cognição e controle de movimento [4]. Neste trabalho, será dado ênfase à análise de algumas técnicas utilizadas na localização de robôs móveis.

1.1 Projeto DesTINe

O projeto DesTINe (Desenvolvimento de Tecnologias da Informação para Neurologia) conduz atividades com foco em seis metas, conforme proposta aprovada pela FINEP [5, 6].

As metas do projeto DesTINe são:

- Concepção geral de um sistema de *Brain-Computer Interface* (BCI);
- Avaliação clínica de pacientes com deficiências motoras visando o uso de BCI;
- Aquisição de imagens multimodal;
- Desenvolvimento de técnicas de processamento de sinais cerebrais;

- Desenvolvimento de um sistema de software para robôs assistivos;
- Desenvolvimento de ambientes inteligentes para acessibilidade.

Dentro do escopo do projeto, este trabalho se encaixa na meta de desenvolvimento de um sistema de software para robôs assistivos. A ideia é propiciar a condução semiautônoma ou com autonomia compartilhada entre o condutor e o robô, visando minimizar o número de interações entre eles. Neste modo de operação do veículo (robô), o condutor supre um comando de alto nível (ex.: “vá para o quarto” ou “siga em frente”) e o robô autonomamente executará a ação, desviando de obstáculos, ajustando sua velocidade, etc. O condutor pode ainda suprir um comando que altera a ação corrente (ex.: “pare” ou “vire à direita”) [5]. Neste contexto, a localização assume papel importante para propiciar navegação autônoma ao robô.

1.2 Motivações

O problema de localização tem recebido grande atenção desde a década passada, e, como resultado, avanços significativos foram alcançados [4]. Nesta etapa, o robô deve determinar sua postura ou a postura específica de algum alvo no ambiente em que se encontra.

A postura do robô em um sistema métrico é geralmente denotado pela Expressão 1.1, que é suficiente para determinar esta transformação de coordenadas, assumindo que a postura P seja expressa no mesmo sistema de coordenadas do mapa (sistema de coordenadas global) [7]. Vale observar que a postura P do robô é dada por sua posição (x, y) e orientação (θ) no plano. Deve-se ainda fazer a distinção entre o sistema de coordenadas global e local (do robô), como demonstrado na Figura 1.1.

$$P = \begin{pmatrix} x & y & \theta \end{pmatrix}^T \quad (1.1)$$

Em diversas aplicações, a solução do problema de localização do robô é imprescindível, uma vez que é necessário que ele tenha um nível razoável de confiança sobre sua postura real a fim de

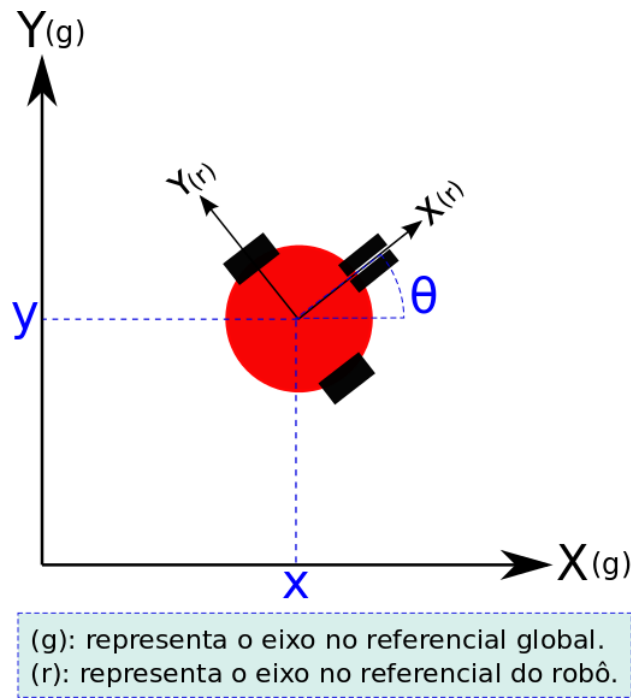


Fig. 1.1: Sistemas de coordenadas e postura do robô

realizar corretamente a tarefa proposta. Esta dissertação se propõe a analisar técnicas de localização e compará-las em situações reais (não simuladas).

O tema de localização é importante pois a partir dele pode-se responder três perguntas fundamentais na área de robótica móvel: “Onde está o robô?”, “Onde está seu alvo?” e “Que caminho o robô deve tomar para encontrar seu alvo?” [8]. Conhecer a sua própria postura é fundamental para que o robô possa tomar decisões com algum respaldo quanto à sua movimentação. Da mesma forma, saber onde o alvo se encontra é importante para permitir que o robô planeje a melhor forma de alcançar seu objetivo. A partir da representação do robô e do alvo no mapa do ambiente, pode-se então responder a terceira pergunta. Embora os métodos para responder esta última indagação não envolvam diretamente localização (no tocante ao caminho), só é possível solucioná-la quando as outras duas questões já foram solucionadas, o que de forma indireta acaba associando-a também à localização.

A escassez de textos que analisam o desempenho dos algoritmos de localização em ambientes reais, além da possibilidade de propiciar a avaliação de desempenho de algumas técnicas em cenários

específicos voltados aos interesses do estudo de caso (ver seção 1.1), tornaram a discussão deste tema bastante interessante, e estas são algumas das principais razões que motivaram a realização deste estudo.

1.3 Objetivos

Nesta dissertação, pretende-se analisar a performance de alguns métodos de localização nas situações em que geralmente são aplicados e obter resultados que possam descrever o desempenho dos mesmos, tanto quantitativamente quanto qualitativamente.

1.3.1 Objetivo Geral

A pesquisa conduzida nesta dissertação é constituída da análise dos seguintes métodos de localização: estimação recursiva de estado, filtros gaussianos, filtros não-paramétricos e localização pelo método de Markov; utilizando-se para este estudo os sensores: laser *rangefinder*, sonar, e a própria odometria do robô; em ambiente real; e fazendo uso de mapas métricos, em grade e topológicos.

O critério para a escolha dos métodos citados ilustra o interesse em abordar a maior variedade de características possíveis dentro da localização por mapas no tempo hábil.

Os pontos de análise deste estudo para cada um dos métodos de localização são listados a seguir. O capítulo 8 descreve a forma encontrada para mensurar cada um destes tópicos, além de apresentar os resultados obtidos.

- Análise Qualitativa
 - Dificuldade de implementação: esta seção discute o esforço empregado para implementação dos algoritmos das técnicas de localização utilizadas.
 - Deficiências: é importante conhecer as desvantagens e incapacidades das técnicas com que se pretender trabalhar.

- Análise Quantitativa
 - Eficiência (nível de erro da estimativa produzida): apresenta-se o erro médio produzido por cada umas das técnicas ao serem aplicadas no mesmo experimento.
 - Tempo de execução: esta seção explicita o tempo médio que cada método necessita para executar a rotina uma vez.
 - Consumo de CPU: este valor indica quanto de processamento cada método utiliza.
 - Memória alocada: análise da memória alocada durante a execução do experimento.
 - Linhas de código: ilustra o esforço de programação requerido para cada método.

1.3.2 Objetivos Específicos

- Implementar uma estratégia de decomposição do ambiente:
 - Split-and-Merge: método que, a partir das leituras do laser, gera linhas, tornando possível a criação de mapas métricos locais para comparação com o mapa métrico global (construído a priori) do ambiente. Mapa métrico é a representação do ambiente através de unidades métricas, como metros, milímetros, etc. Mapa local é o mapa criado a partir do obstáculos dentro do alcance e campo de visão dos sensores do robô, ao passo que o mapa global é o mapa completo do ambiente, como sua planta baixa.
- Implementar os métodos subsequentes [7]:
 - Localização com estimação recursiva de estado:
 - * Filtro de Bayes.
 - Localização com base em filtros gaussianos:
 - * Filtro de Kalman dinâmico;
 - * Filtro de Kalman estendido.

- Localização por meio de filtros não-paramétricos:
 - * Filtro de Partícula (Monte Carlo).
- Localização pelo método de Markov.

1.4 Contribuição

Com este trabalho deseja-se experimentar diversas técnicas de solução para o problema em questão, aplicando-as em cenários específicos, de forma a gerar um estudo comparativo entre os resultados visando contribuir com informações que venham a enriquecer a discussão sobre o tema. Além destas contribuições, pretende-se disponibilizar uma biblioteca de algoritmos e resultados utilizando o mesmo ferramental (robô, sensores, ambiente, plataforma, entre outros).

1.5 Cenário e Navegação

O estudo de caso apresentado consiste do robô emulando uma cadeira de rodas robotizada que deverá navegar por lugares compostos por salas, quartos, corredores, etc. Neste cenário, os corredores tem importância vital para a navegação entre cômodos, e foram escolhidos como caso de teste para os métodos empregados nesta dissertação.

Para a realização dos experimentos, foi escolhido o corredor central no Laboratório de Computação e Automação (LCA). A Figura 1.2 apresenta um diagrama esquemático do laboratório em questão. Esta escolha se baseia em dois pontos:

- A relevância dos corredores se deve ao fato de ligar os demais cômodos, tornando-se passagem obrigatória de pessoas e do próprio robô. Além disso, grande parte do problema de localização se dá neles, visto que são lugares mais longos e são os que mais contribuem para o aumento do erro de odometria.

- O corredor é um cenário propício para a implantação dos métodos de localização robótica escolhidos já que não favorece nenhum deles, o que é primordial para um estudo comparativo.

É possível observar que existem algumas salas ao longo do caminho, as quais são acessadas pelo corredor. Este corredor em específico não é formado por paredes estritamente paralelas, mas por paredes compostas de recuos nas portas e por avanços causados pelos pilares. Essa característica contribui com mais ruído para o sistema e sua presença fica evidente na planta-baixa do corredor apresentado na Figura 1.3. Os objetos alaranjados da Figura 1.2 são as portas das salas, utilizadas para simular um corredor genérico. Ademais, conhecer o estado das portas (se estão abertas ou fechadas) é bastante útil para alguns dos métodos experimentados, já que podem servir como marcos artificiais (*landmarks*) ou topologias. Na Figura 1.3 é ilustrado o estado de cada porta durante os experimentos, onde algumas portas estiveram abertas enquanto outras permaneceram fechadas. Para determinar o estado de cada porta (aberta ou fechada) procurou-se utilizar a maior quantidade possível de padrões entre as portas adjacentes (ex.: porta da esquerda fechada e a da direita aberta), de modo a criar uma variedade de topologias. O estado das portas foi mantido intacto durante todos os experimentos e a mesma composição foi empregada em todas as técnicas. A Tabela 1.1 traz alguns dados sobre o corredor.

A navegação do robô ao longo do corredor se dá em duas etapas. Na primeira, o robô inicia na postura representada em ambas as figuras (Figura 1.2 e Figura 1.3) pelo objeto/robô vermelho, o qual será denominado "Ponto Norte", e descende até o fim do corredor, parando a 1 metro da porta de entrada. O ponto final será denominado "Ponto Sul". Na segunda etapa, o robô deve girar 180° e reiniciar a navegação, agora em direção ao ponto Norte. Ao fim de navegação, será interessante observar o erro de postura acumulado após um razoável deslocamento, incluindo uma rotação. A nomenclatura escolhida para os pontos da Figura 1.3 são apenas para facilitar o entendimento das discussões que virão, esclarecendo que estes não guardam qualquer relação com os pontos de orientação propriamente ditos e que nenhuma bússola foi utilizada nos experimentos. Algumas informações sobre a navegação encontram-se na Tabela 1.2. Durante a navegação, o robô não para

	Comprimento (mm)	Largura (mm)
Corredor	23400	2240
Porta	810	-
Recuo da Porta	1620	140
Pilar	150	150

Tab. 1.1: Dados sobre o corredor

para coletar dados nem para processar os métodos de localização, realizando-a em movimento. Os resultados foram coletados por meio de uma trena e um transferidor.

Um outro aspecto que vale a pena ressaltar é o fato dos algoritmos, tanto de navegação quanto de localização, rodarem em uma máquina externa ao robô. Isso permite que o sistema tenha disponível um maior poder de processamento e memória.

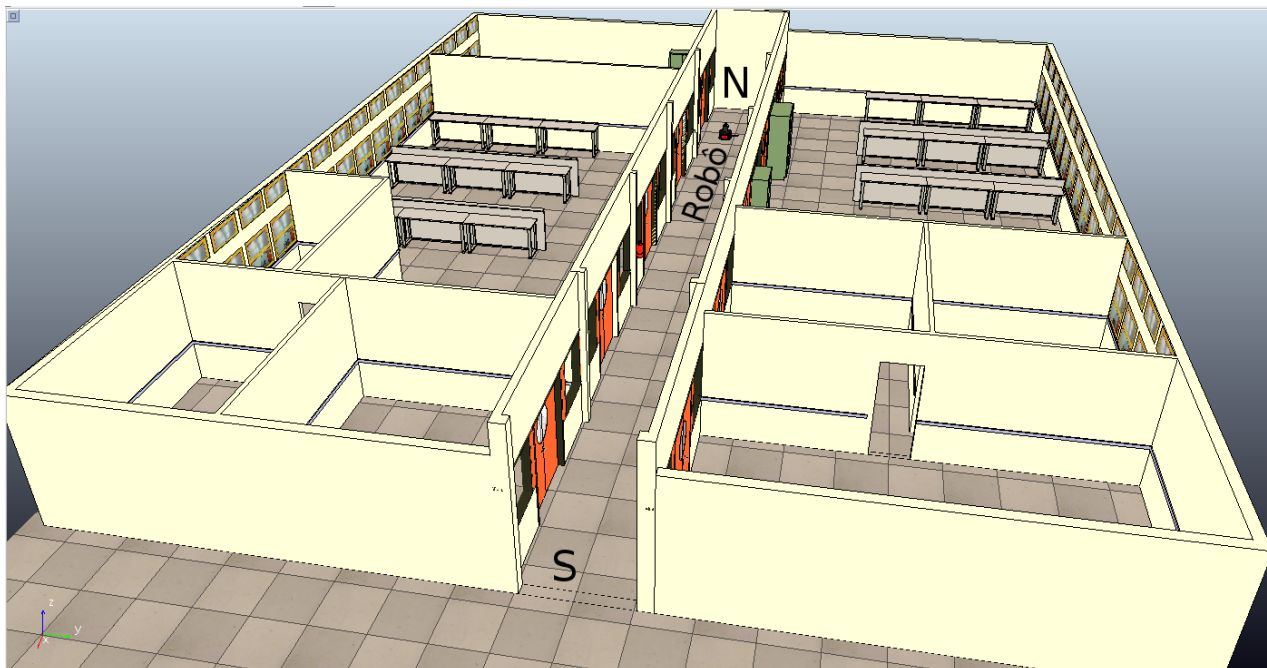


Fig. 1.2: Laboratório LCA

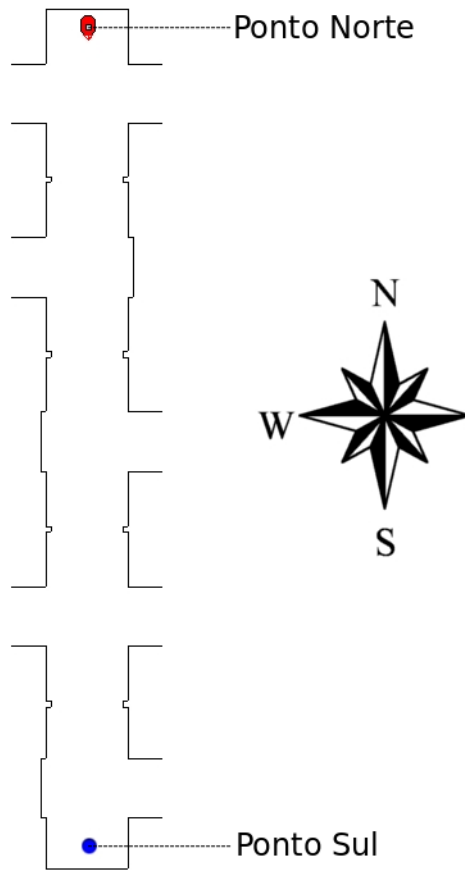


Fig. 1.3: Planta baixa do corredor do laboratório LCA

1.6 Apresentação da Dissertação

A dissertação encontra-se dividida em nove capítulos:

- Capítulo 2 - Localização para Robôs Móveis.

Este capítulo traz alguns elementos da localização robótica, as taxonomias normalmente empregadas na área, as fontes de erros, uma descrição sobre fusão de sensores, características dos ambientes usados nos experimentos e um resumo acerca da plataforma (*software* e *hardware*) usada.

- Capítulo 3 - Modelagem dos Sensores do Robô.

Distância entre Ponto Norte e Sul	21990 mm
Distância Total Percorrida	43980 mm
Ângulo Total Percorrido	180°
Velocidade Linear	150mm/s
Velocidade Angular	30°/s

Tab. 1.2: Dados sobre a navegação

A modelagem dos sensores é importante pois os métodos de localização utilizam estes modelos para localizar o robô. Neste capítulo, pode-se constatar como o erro dos sensores exteroceptivos (laser e sonar) se comportam na etapa de percepção do ambiente, assim como o comportamento do erro do sensor propioceptivo (odometria) quanto à movimentação do robô. A estimativa é baseada no erro proveniente das fontes de informação.

- Capítulo 4 - Filtro de Bayes.

Este capítulo traz a descrição da técnica, cenário utilizado, os resultados adquiridos e uma conclusão sobre a implementação do filtro de Bayes.

- Capítulo 5 - Localização pelo método de Markov.

Este capítulo apresenta a localização pelo método de Markov, incluindo a descrição da técnica, os resultados obtidos e uma conclusão sobre sua implementação.

- Capítulo 6 - Filtros Gaussianos.

Este capítulo traz a descrição das técnicas dos dois filtros de Kalman implementados, assim como o cenário utilizado, os resultados adquiridos e uma conclusão sobre a implementação dos dois métodos.

- Capítulo 7 - Filtro de Partículas.

Este capítulo trata do método baseado no filtro de partículas, conhecido como método de Monte Carlo, com a descrição da técnica, cenário utilizado, os resultados adquiridos e uma conclusão sobre sua implementação.

- Capítulo 8 - Comparação dos Métodos.

Neste capítulo, é feita a comparação de todos os métodos implementados com base nos aspectos discutidos na subseção 1.3.1, além da descrição dos motivos de escolha de cada um destes aspectos.

- Capítulo 9 - Conclusão, Considerações Finais e Trabalhos Futuros.

Neste capítulo, relata-se a visão geral da dissertação, assim como uma discussão final e o que se espera de trabalhos futuros que podem ser originados a partir deste.

Capítulo 2

Localização para Robôs Móveis

Este capítulo apresenta os conceitos, nomenclaturas e desafios básicos relacionados à localização de robôs móveis.

2.1 Elementos da Localização Robótica

Localizar um robô móvel consiste no desafio de determinar sua postura (*pose*) em relação a um referencial específico (por exemplo, o mapa do ambiente em que o robô se encontra). Devido à dificuldade de se obter resultados precisos, por consequência das fontes de incerteza, este problema é comumente chamado de estimação de postura (*pose estimation*) [7]. A estimação da postura é feita de forma estocástica por meio de um algoritmo probabilístico, o qual mantém hipóteses sobre possíveis locais do mapa em que o robô possa estar [3].

Localização é um dos problemas básicos na área da robótica móvel já que quase todas as tarefas em robótica requerem conhecimentos sobre a postura dos robôs e/ou de objetos que estão sendo manipulados [7]. O problema de localização pode ser visto também como um problema de transformação de coordenadas. Como os mapas estão descritos em um sistema global de coordenadas, totalmente independentes da postura do robô, o problema é resolvido ao se estabelecer a correspondência entre o mapa de coordenadas globais com o sistema local de coordenadas do robô.

Conhecer esta transformação de coordenadas permite ao robô expressar a localização dos objetos de interesse dentro do seu próprio sistema de coordenadas, o que é fundamental para a etapa de planejamento da navegação robótica.

Como dito anteriormente, definir a postura (P) precisa do robô não é tarefa fácil pois ela geralmente não pode ser determinada diretamente. O maior empecilho para essa determinação são os ruídos intrínsecos às características dos sensores. Como os sensores são a base para a percepção do ambiente pelo robô, a sua localização acaba sendo inferida através dos dados ruidosos oriundos destes sensores. A dificuldade chave disto decorre do fato de que um único tipo de sensor de medição é normalmente insuficiente para determinar P . Ao invés disso, o robô deve integrar os dados provenientes de dois ou mais tipos de sensores [7], preferencialmente com características e naturezas distintas, para evitar que estejam sujeitos ao mesmo tipo de ruído.

Ruído e Sobreposição como Desafio para a Localização

É evidente que os sensores do robô têm um papel fundamental em todas as formas de localização, e é devido à sua inexatidão que o problema de localização se torna um desafio [4]. Existem dois problemas intrínsecos aos sensores: ruído e sobreposição.

O ruído de um sensor induz uma limitação na consistência das leituras obtidas pelo mesmo, ainda que apuradas sobre o mesmo estado do ambiente. Em outras palavras, o ruído reduz a informação útil contida nas leituras supridas pelo sensor. Para contornar este problema, a solução comumente adotada é realizar diversas leituras com sensores exteroceptivos diferentes e empregar a fusão de sensores com os dados adquiridos. Esta solução tem como finalidade aumentar a qualidade (confiança) das informações adquiridas [4]. A subseção sobre Fusão de Sensores (2.6) trata especificamente deste tópico.

Uma segunda deficiência dos sensores faz com que eles produzam informações com pouco conteúdo, diminuindo a utilidade dos dados e consequentemente, dificultando sua utilização na localização [4]. Sobreposição (*aliasing*) é o fenômeno que ocorre quando a mesma informação é

suprida para leituras em diferentes posições do ambiente.

2.2 Taxonomias Utilizadas

Nesta seção são exibidas algumas classificações referentes à natureza do ambiente e ao conhecimento inicial que o robô tem do ambiente [7].

2.2.1 Localização com Respeito à Informação Disponível

Pode-se caracterizar o problema de localização quanto à informação disponível inicialmente e durante a execução de alguma tarefa. Deve-se fazer a distinção de três tipos de problemas de localização com grau crescente de dificuldade [7]:

- **Localização Local**

Denota a estimação de postura de um objeto móvel baseado em consecutivas medidas de sensores [9]. Nesta classificação, assume-se que a postura inicial do robô é conhecida. A incerteza proveniente desta postura é geralmente aproximada por uma distribuição de probabilidade (geralmente gaussiana). Trata-se de um problema local, já que a incerteza é local e confinada à região próxima à postura real do robô [7, 9].

- **Localização Global**

Neste caso, assume-se que a postura inicial do robô é desconhecida e que não há limitação no erro de postura, a incerteza é global. Para este caso, distribuição de probabilidade unimodal é geralmente inapropriada, podendo-se recorrer a uma distribuição multimodal, por exemplo. Este problema é mais complexo que o problema de localização local [3, 7].

- **Sequestro do Robô**

Este problema é uma variante do problema de localização global. Durante a operação, o robô é “sequestrado” e levado para uma postura desconhecida. Este problema é mais difícil pois o robô

acredita que está em uma postura, enquanto na verdade se encontra em outra. Assim, o robô deve identificar que a crença é falsa, diminuir a crença relativa à postura antes do “sequestro” e só então tentar se relocalizar. Este problema é interessante pelo fato de apontar para uma característica importantíssima dos algoritmos de localização, a habilidade de se recuperar de falhas ou falsas crenças [3, 7].

2.2.2 Ambientes Estáticos x Dinâmicos

O ambiente tem obviamente um impacto substancial na dificuldade de localização [7]. Ambientes podem ser classificados em:

- Ambientes Estáticos: São os ambientes onde o único estado variável é a postura do robô, ou seja, todos os outros objetos do ambiente permanecem sempre nas mesmas posições. Este tipo de ambiente apresenta algumas propriedades que os fazem passíveis de eficientes estimações probabilísticas [7].
- Ambientes Dinâmicos: Apresentam outros objetos além do robô cuja postura ou configuração muda com o tempo. Mudanças que não são perceptivas não são relevantes para a localização, e mudanças que afetam apenas um sensor são tratadas como ruído [7].

2.2.3 Abordagem Passiva x Ativa

O tipo de abordagem caracteriza o modo como o algoritmo de localização interfere no controle de movimentação do robô [7].

- Localização Passiva: Nesta abordagem, o módulo de localização apenas observa o robô operando.
- Localização Ativa: Nesta abordagem, o módulo de localização controla a movimentação do robô de modo a minimizar o erro de localização e os custos provenientes de sua movimentação em um ambiente.

2.2.4 Único Robô x Vários Robôs

A quantidade de robôs envolvidos também é importante no problema de localização [7].

- Localização de um Único Robô: Esta localização tem o conveniente de que toda a informação é coletada da mesma plataforma e não envolve comunicação entre plataformas robóticas.
- Localização de Vários Robôs: Naturalmente, a dificuldade do problema aumenta quando se aumenta o número de robôs. Embora a localização possa ser feita individualmente, quando se está em grupo é possível utilizar a crença de postura de um robô no cálculo da crença de outro robô, caso a distância relativa entre eles seja conhecida.

2.3 Métodos de Localização Robótica

Existem vários métodos para localização robótica, com características específicas que os tornam recomendáveis para um dado tipo de problema. Alguns dos métodos mais utilizados são:

- Posição estimada internamente (*Dead-reckoning*): Este método não utiliza nenhum ponto de referência externa. O exemplo mais usado deste tipo de método é o que utiliza a odometria, que adquire a informação necessária sobre o deslocamento a partir dos *encoders* alojados nas rodas do robô [4, 8, 9].
- Utilizando Marcas: A postura é estimada baseando-se a distância do robô à marcas específicas no ambiente. Estas marcas podem ser artificiais ou naturais do ambiente [4, 8, 9].
- Utilizando Sinais Luminosos (*Beacon*): A postura é estimada ao se calcular a distância percorrida por algum sinal, como o de luz. É possível obter uma estimação confiável quando há mais de uma fonte de sinal e quando se sabe a postura destas fontes [4, 8].
- Posicionamento Baseado em Visão: Utiliza sensores ópticos para a estimativa da postura do robô no ambiente [8].

- **Posicionamento Baseado em Mapa:** É o método em que o robô gera um mapa de sua redondeza e o compara com o mapa do ambiente armazenado em sua memória. Se o mapa gerado combina com alguma parte do mapa memorizado, o robô pode melhorar a estimativa de sua postura no ambiente [4, 8].

Devido à natureza do estudo de caso conduzido nesta dissertação, serão aplicados métodos apenas baseados em mapas. É razoável assumir que o mapa do ambiente sempre estará disponível, já que se estará considerando ambientes internos em locais projetados, como é o caso do laboratório LCA (ver seção 1.5).

2.4 Incertezas que Impactam na Localização Robótica

Robótica é a ciência de perceber e manipular o mundo físico por meio de dispositivos controlados, sensores e atuadores [7]. Infelizmente, todos estes dispositivos apresentam incertezas inerentes ao seu funcionamento, e cabe ao desenvolvedor de sistemas robóticos saber lidar com estas. As principais fontes de incerteza estão listadas abaixo [7].

- **Ambiente:** quanto mais dinâmico for o ambiente maior será sua contribuição com incertezas, já que é bastante difícil prever seu estado.
- **Sensores:** são bastante limitados no que podem captar. Essas limitações são frutos de restrições físicas do dispositivo, leis físicas e ruídos.
- **Robôs:** como o robô depende basicamente de movimentos de motores, a inexactidão desses faz com que o próprio robô atue como fonte de incerteza.
- **Modelos:** são inerentemente imprecisos, pois são abstrações do mundo real. Desta forma, modelam apenas parte dos processos físicos que estão em voga.

Erros sistemáticos	Erros não sistemáticos (randômicos)
Diâmetros das rodas desiguais;	Navegação em superfícies irregulares;
Diâmetros das rodas difere do nominal;	Objetos inesperados no solo;
Desalinhamento das rodas;	Escorregamento das rodas devido a:
Incerteza associada ao contato entre roda e solo;	- Solos escorregadios;
Resolução limitada dos <i>encoders</i> ;	- Sobre-aceleração;
Taxa de amostragem dos <i>encoders</i> .	- Viradas bruscas;
	- Forças internas e externas.

Tab. 2.1: Exemplos de erros sistemáticos e não sistemáticos na odometria de um robô móvel [16].

- Computação: como os robôs são sistemas em tempo real, há uma limitação na carga computacional que se pode executar. Assim, é comum o uso de aproximações que dão prioridade a respostas rápidas em detrimento da precisão.

Para o estudo empreendido nesta dissertação nos interessa especialmente o erro proveniente dos sensores, já que provêem grande parte das informações que serão empregadas nas técnicas de localização. Como mencionado anteriormente, sensores são dispositivos imperfeitos, com erros tanto sistemáticos quanto de natureza randômica. Estes erros randômicos não podem ser corrigidos, de maneira que representam fonte de incertezas ao sistema [4].

Para estudar os efeitos dessa incerteza, primeiramente deve-se apresentar uma representação estatística para o erro randômico associado a cada sensor. Uma ferramenta valiosa neste quesito é o uso do modelo Gaussiano. Os textos Bar-Shalom et al., 2001; Hall and Llinas, 2001; Raol, 2009; Thrun et al., 2005; e Siegwart and Nourbakhsh, 2004; [4, 7, 10–12] trazem mais detalhes sobre a representação estatística e distribuição de Gauss. O artigo Chong and Kleeman, 1997 [13], apresenta passos para a modelagem do erro a baixo custo computacional usando a odometria do robô e os artigos Smith and Cheeseman, 1986; Smith et al., 1990; [14, 15] trabalham com a estimação das incertezas no espaço.

Tão importante quanto as incertezas provenientes dos sensores exteroceptivos são os erros oriundos do sensor proprioceptivo, a odometria. Na Tabela 2.1 são apresentados alguns dos erros sistemáticos e não sistemáticos que influenciam diretamente na geração de erros de odometria.

2.5 Extração de Características do Ambiente

O robô móvel deve ser capaz de determinar características do ambiente a sua volta através de seus sensores. Essa habilidade é indispensável quando o sistema de localização se baseia em mapas [4]. A percepção do ambiente é fundamentalmente importante em dois aspectos da navegação: detecção dos limites de espaço vazio, navegável; e estimação da postura [17].

Existem vários métodos para extração de características do ambiente, assim como vários tipos de características a serem extraídas [4]. Nesta dissertação, é empregado o método *Split-and-Merge*, que forma linhas com os dados extraídos do ambiente pelos sensores. Este método tem como maior vantagem a pouca demanda de memória. Mapas de linhas requerem uma memória significativamente menor que a de mapas descritos ponto-a-ponto e portanto, escalam melhor com o tamanho do ambiente. Além disso, não sofrem com os problemas de discretização [4, 18, 19].

O trabalho apresentado em Sack and Burgard, 2004 [18], trata da comparação de alguns métodos de extração de linhas do ambiente e em Arras and Siegwart, 1997 [19], é detalhado a aplicação de um método de extração de características para uma navegação baseada em mapa. Um exemplo de como a localização pode ser feita com o método escolhido para a dissertação é tratado em Luo et al., 2008 [20], assim como a apresentação do algoritmo para sua execução.

2.6 Fusão de Sensores

Os robôs geralmente possuem mais de um tipo de sensor, capazes de sensoriar diversos tipos de informação (calor, distância por meio de som ou luz, cor, textura, fricção, etc), e todas estas podem ser integradas para a criação de uma estimativa sobre a postura. A maneira usada para integrar a informação destes sensores é bastante importante para a qualidade da estimação, principalmente quando os sensores possuem características diferentes [4, 7].

Como o mapa deve incluir detalhes suficientes para a localização do robô, a fusão dos sensores deve gerar uma representação do mundo que é suficientemente expressiva, bem além de um mapa

criado apenas por um tipo de sensor [4].

O termo fusão de sensores remete à combinação de dados sensoreados ou de informações derivadas destes dados de tal forma que a informação resultante tenha uma incerteza menor que a informação extraída separadamente de cada sensor [12].

Os três métodos mais usados para realizar a fusão dos sensores para aplicações que envolvem movimentação são [12]:

- Fusão de dados brutos de observação e medição, muitas vezes chamado de fusão centralizada;
- Fusão dos vetores dos estados estimados;
- Fusão híbrida, no qual dados brutos são fundidos com vetores de estado processados.

Dentre as maiores vantagens de se usar a fusão de sensores, é possível listar [11, 12]:

- Redundância: se vários sensores são utilizados, a combinação de suas observações tendem a resultar em uma estimativa melhorada. Uma vantagem estatística é conseguida por adicionar N observações independentes.
- Pode-se usar o local relativo ou movimento de sensores para melhorar o processo de observação (por exemplo, a triangulação).
- Melhorar o poder de observabilidade ao integrar sensores com características diferentes.
- Compensar dados incorretos, falhos ou perdidos por parte dos sensores.
- Aumentar a área sensoreada.

A fusão de sensores vêm sendo empregada extensivamente tanto em aplicações civis como militares, dentre as quais pode-se citar: rastreamento de alvos, identificação automática de alvos, guiar veículos autônomos, sensoriamento remoto, aplicações automatizadas com poder computacional limitado, localização robótica, etc [11]. É importante ressaltar que fusão de sensores é viável quando se tem dados de qualidade. A manipulação de dados incorretos não produzirá bons resultados, a não

ser com um alto custo computacional [12]. Existem alguns métodos básicos para fusão, como o uso do filtro de Bayes, fusão de mapas, rede neural classificadora, filtro de Kalman, etc [4, 7]. Algumas destas técnicas foram empregadas nesta dissertação.

Utilizando a fusão de mapas, será gerado um mapa com as informações obtidas de cada sensor em separado, e então estes mapas são integrados utilizando-se a estimativa mais conservadora. Este mapa resultante é o mais pessimista possível, visto que se em algum dos mapas gerados pelos sensores uma célula específica é considerada preenchida, a célula correspondente no mapa combinado também estará preenchida [7].

Na rede neural classificadora, qualquer quantidade e tipo de sensores podem ser combinados em uma rede que deverá usar os dados necessários para otimizar sua eficiência de classificação [4]. Bar-Shalom, 2001 [10], apresenta uma fusão de estimativas, enquanto Hall, 2001 [11], traz um diagrama representativo de aplicações e suas características.

2.7 Infraestrutura Experimental

O Ambiente dos Experimentos

Para os experimentos (realizados em ambiente real), foram utilizadas as instalações do laboratório LCA, localizado nas dependências da Faculdade de Engenharia Elétrica e de Computação - FEEC, na Universidade Estadual de Campinas - UNICAMP.

Plataforma de Software e Hardware Utilizada

Software

Foi utilizada uma plataforma de software denominada *Plataforma DesTINe*, desenvolvida para a navegação de robôs assistivos, como prevê o projeto ao qual este trabalho está ligado (ver seção 1.1). Esta plataforma permite [5]:

1. abstrair toda a diversidade de hardware do robô assistivo;
2. controlar o robô por meio de programas embarcados ou executando remotamente;
3. desenvolver os programas de controle utilizando diferentes linguagens/ambientes de programação;
4. interagir com ambientes inteligentes.

A plataforma consiste de duas partes cooperantes. A primeira é um servidor embarcado que interage com o hardware do robô. A segunda são interfaces de programação (APIs - Application Programming Interfaces) utilizadas pelas aplicações robóticas (denominadas APIs do cliente). Estas APIs são disponibilizadas em múltiplas linguagens de programação: C++, Java, Python, Matlab, Android, Java ME (Micro Edition), C# e JavaScript. A comunicação API-servidor se dá por meio do protocolo HTTP (Hypertext Transfer Protocol). Uma resposta do servidor tipicamente é um documento XML (Extensible Markup Language). Em certos casos o servidor retorna também uma imagem ou uma sequência de imagens (vídeo).

Hardware

Em todos os experimentos realizados foi utilizado o robô diferencial P3-DX (Figura 2.1). Esse robô conta com a relação de sensores listados a seguir. A falta de um giroscópio ou bússola eletrônica contribui para o aumento do erro de postura do robô, principalmente quando este executa rotações. Estes sensores poderiam ainda perceber pequenas mudanças de orientação devido ao deslizamento das rodas, o que evidencia ainda mais sua utilidade.

- *Encoders* instalados em ambas as rodas;
- 16 sonares, distribuídos igualmente na parte frontal e traseira do robô;
- Laser *rangefinder* com 180° de abertura;

- Sensores de colisão (*bumpers*);

Como o P3-DX é um robô diferencial, ambas as rodas podem ser controladas separadamente. Há ainda uma roda castor na parte de trás, que dá estabilidade ao robô. Além disso, o robô inclui o software ARIA (*Advanced Robotics Interface for Applications*), uma plataforma de desenvolvimento que provê uma interface cliente tanto para o microcontrolador interno (*ARCOS*) quanto para os acessórios do robô. A Tabela 2.2 contém informações físicas do robô relevantes para a implementação das técnicas de localização.



Fig. 2.1: Robô P3-DX

Tab. 2.2: Informações físicas do robô [21,22]

Informação	Medida (cm)
Distância axial entre as rodas motoras	32
Raio das rodas motoras	9,25

Todos os experimentos foram realizados utilizando-se um computador com o sistema operacional Linux (Ubuntu v12.10). Este computador apresenta processador *Intel Core 2 Duo T5550*, *2048MB PC5300 DDR2 SDRAM*.

Capítulo 3

Sensores: Extração de Características e Modelagem

Uma das tarefas mais importantes de um sistema autônomo de qualquer tipo é adquirir conhecimento sobre seu ambiente [4], o que é geralmente feito através de medições realizadas por intermédio de sensores. Sem eles, os robôs estariam limitados a uma automação rígida, executando a mesma tarefa repetitivamente em um ambiente cuidadosamente controlado. Em contrapartida, o robô que apresenta sensores adquire a capacidade de operar em ambientes não-estruturados e se adaptar enquanto o ambiente muda em torno dele, além de poderem interagir com o ambiente, com pessoas e até mesmo com outros robôs [23].

3.1 Classificações e Características de Sensores

Existe uma grande variedade de sensores usados para transformar informações do ambiente em dados inteligíveis [24]. Para um robô móvel, estas informações são cruciais, dado o risco de sofrer acidentes (ex.: cair em um buraco ou escada) ou causar acidentes (ex.: se chocar com obstáculos/pessoas). Além disso, este conhecimento pode contribuir para a conclusão de suas tarefas com sucesso, já que atualizam o conhecimento que o robô tem a respeito do local em que se encontra

e/ou age. O artigo White, 1987 [24], apresenta um diagrama esquemático contendo diversos tipos de sensores assim como vários tipos de grandezas possíveis de se medir. Os livros Borenstein et al., 1996 [8]; Siegwart and Nourbakhsh, 2004 [4]; e Everett, 1995 [23]; trazem descrições do funcionamento de vários tipos de sensores.

Devido à diversidade de características dos sensores, é comum identificá-los em subgrupos (classificações) de acordo com alguns parâmetros. As formas mais usuais de classificação de sensores são:

- Com respeito à visão [23]:
 - Sensores Visuais: adquirem os dados através de imagem/vídeo (ex.: câmeras).
 - Sensores Não-visuais: adquirem os dados através de métodos diferentes de imagem/vídeo.
- Com respeito ao local de extração da informação [4]:
 - Sensores Proprioceptivos: adquirem dados internos ao sistema (robô), como a velocidade das rodas, ângulo do robô, carga das baterias, etc;
 - Sensores Exteroceptivos: adquirem os dados do ambiente no qual o robô se encontra, como a distância a obstáculos, intensidade luminosa, som, temperatura, etc.
- Com respeito à energia [4]:
 - Sensores Passivos: captam a energia do ambiente. Exemplos de sensores deste tipo são câmeras, termômetros, microfones, etc;
 - Sensores Ativos: emitem energia para o ambiente e, posteriormente, medem como o ambiente reagiu a este estímulo.

A fim de obter o melhor resultado possível, é importante conhecer a característica de performance dos sensores disponíveis, principalmente quando se executa fusão de sensores. Algumas considerações gerais sobre estas características de performance são [4, 23]:

- **Campo de visão:** deve ser amplo o suficiente para se adequar à aplicação.
- **Alcance:** o alcance máximo e mínimo devem ser apropriados para o uso pretendido.
- **Erro:** é definido como a diferença entre o valor medido pelo sensor e o valor real. Dado um valor real v e um valor medido m , temos que o erro é dado pela Equação 3.1.

$$erro = m - v \quad (3.1)$$

- **Exatidão (*accuracy*):** é definido como o grau de conformidade entre a leitura do sensor e o valor real sendo medido, dado pela Equação 3.2.

$$exatidão = 1 - \frac{|erro|}{v} \quad (3.2)$$

- **Precisão:** um sensor é altamente preciso se ele produz a mesma leitura ao medir o mesmo valor real [4]. A precisão é dada pela Equação 3.3, onde σ denota o desvio padrão das leituras.

$$precisão = \frac{alcance}{\sigma} \quad (3.3)$$

- **Resolução:** é a menor diferença entre dois valores que podem ser observados pelo sensor.
- **Habilidade de detectar objetos no ambiente:** alguns objetos podem absorver a energia emitida ou ainda difratar a luz. É importante então conhecer que tipo de problemas o ambiente pode trazer ao tipo de energia com a qual o sensor trabalha.
- **Operação em tempo real:** a frequência de atualização precisa ser rápida o suficiente para estar de acordo com a velocidade de atuação do sistema.
- **Dados concisos e fáceis de interpretar:** o formato de saída deve ser realista do ponto de vista do processamento exigido.

Característica	Laser	Sonar
Campo de Visão	270°, passo de 0.5°	16 sonares cobrindo 360°
Alcance	25000 mm	5000 mm
Erro	24.0530 ± 0.5216 mm	121.8710 ± 0.0208 mm
Exatidão	0.9850 ± 0.0003 mm	0.9238 ± 0.0000 mm
Precisão	2970.7 mm	14909 mm
Resolução	1 mm	1 mm
Custo unitário (dólares)	3500	30

Tab. 3.1: Características de performance do laser e sonar

- **Redundância:** uma capacidade multimodal seria desejável para assegurar a detecção de todos os alvos, assim como para aumentar o grau de confiança da saída.

Os dois sensores exteroceptivos disponíveis para o robô utilizado nesta dissertação são o *laser* e o *sonar*. Para comparar suas características de performance, foram feitas 1000 leituras com cada sensor utilizando todo o seu espectro observável, e o resultado é apresentado na Tabela 3.1. Apesar do laser usado nos experimentos ter um campo de visão de 270° e resolução de 0.5°, optou-se por utilizar um campo de visão de 180° com resolução de 1°, dado que os demais robôs do laboratório LCA possuem laser com esta última configuração, tornando possível o uso dos algoritmos estudados em todos os robôs sem a necessidade de alteração dos parâmetros.

Foi incluído na tabela o custo dos dois tipos de sensores. O alto desempenho do laser (LMS-100) está atrelado a um alto custo (3500 dólares), enquanto a utilização de sonares é mais acessível (os 16 sonares do robô totalizam 480 dólares). Embora o custo não tenha sido usado como critério para a decisão de se utilizar apenas o laser para os experimentos desta dissertação, entende-se que este é um ponto relevante no caso em que o desenvolvedor necessite adquirir sensores para o robô.

3.2 Modelagem dos Sensores

A modelagem dos sensores é fundamental no processo de localização, visto que o sistema precisa lidar com o ruído intrínseco de sua operação e o conhecimento do mesmo pode ser o ponto chave

entre sucesso e fracasso. A seguir, são expostas a calibração e as modelagens para os sensores exteroceptivos (modelagem estática) e proprioceptivos (modelagem dinâmica).

3.2.1 Calibração

Com o passar dos dias, os robôs sofrem com alteração em suas estrutura, mudanças em suas carcaças, mudança de periféricos, deformações em pneus, mudança de seu centro de massa, variação da carga, etc. Com isso, o robô se torna descalibrado e o erro de odometria aumenta além dos limites que permitem uma operação adequada. Para resolver este problema, nos dias que antecederam a coleta dos dados que serão apresentados nos próximos capítulos, o robô teve que passar por um processo de calibragem, que remete ao ajuste dos parâmetros do robô para que sua localização possa condizer com o esperado pela odometria. A calibração consiste de uma rotina de experimentos pré-estabelecidos pela empresa fabricante do robô, a qual é descrita no manual do robô [21]. De acordo com estes experimentos, o robô executa uma operação definida pelo operador e o resultado desta serve de base para comparação com o resultado esperado, e assim, pode-se alterar alguns parâmetros do microcontrolador do robô (ARCOS) convenientemente. Seguindo o manual do robô P3-DX [21], os três parâmetros cruciais para manter o bom desempenho do robô são:

- Fator de curso. Este fator é responsável por fazer com que o robô possa andar em linha reta. Quando desregulado, uma roda pode girar mais rápido que a outra, acarretando um desvio na rota do robô para a direita ou esquerda, embora para a odometria o robô esteja navegando linearmente.
- Contagem de distância. Um outro aspecto que deve ser ajustado é a relação da contagem por meio dos *encoders* com a distância realmente navegada. A deformação das rodas é a principal causa deste erro.
- Contagem de revoluções. Da mesma forma que a distância percorrida, o ângulo do giro efetuado deve ser calibrado afim de diminuir o erro angular do robô.

Para a calibração do robô, cada um destes itens foi minuciosamente testado com diversos valores, até que os erros fossem minimizados. Apesar disso, não há como mitigar o erro totalmente. Além dos valores recebidos pelos parâmetros terem limitações (apenas valores inteiros), os erros provenientes do robô e do ambiente (por exemplo, desnível do piso) estão sempre presentes.

3.2.2 Modelagem Estática

Os sensores exteroceptivos (*laser* e *sonar*) são modelados a partir de leituras feitas em um mesmo cenário (Figura 3.1). As leituras aferidas pelo *laser* são mostradas na Figura 3.2, enquanto as leituras do sonar são apresentadas na Figura 3.3. É evidente que as leituras do *laser* são melhores que as do sonar, já que são mais próximas das leituras reais. Além disso, o sonar retornou leituras erradas, provavelmente devido a reverberação, denotadas pelos pontos distantes na Figura 3.3. Para a modelagem, foram realizadas 1000 leituras com ambos sensores, e os resultados podem ser observados nas figuras de 3.4 a 3.7.

A Figura 3.4 relaciona a média do erro das leituras com relação à distância para o objeto. Nota-se uma variação em torno de 16 mm em quase todo o espectro de distância aferida com exceção da região entre 2300 mm e 2500 mm aproximadamente, que coincidem com as leituras feitas da parede diretamente em frente ao robô. É possível constatar que estas são as leituras com maior erro por meio da Figura 3.2. Contudo, apesar da média do erro se manter praticamente estável, a variância correspondente, diretamente proporcional à incerteza dos dados, aumenta de maneira linear, como revelado na Figura 3.5.

Relativo ao sonar, a Figura 3.6 mostra o erro médio com relação às distâncias aferidas. Como resultados das medidas falsas, é notável que o sensor gera grandes erros. A variância (Figura 3.7) destes erros não acompanham a disparidade do erro das médias pois o sonar tende a retornar valores de leitura muito próximos, mesmo quando errôneos, como evidenciado pela alta precisão do sonar exposta na Tabela 3.1.

A partir da análise da Tabela 3.1 e dos gráficos referentes à modelagem estática, conclui-se que

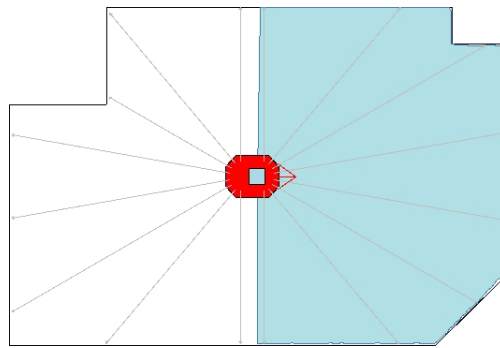


Fig. 3.1: Cenário usado na modelagem estática

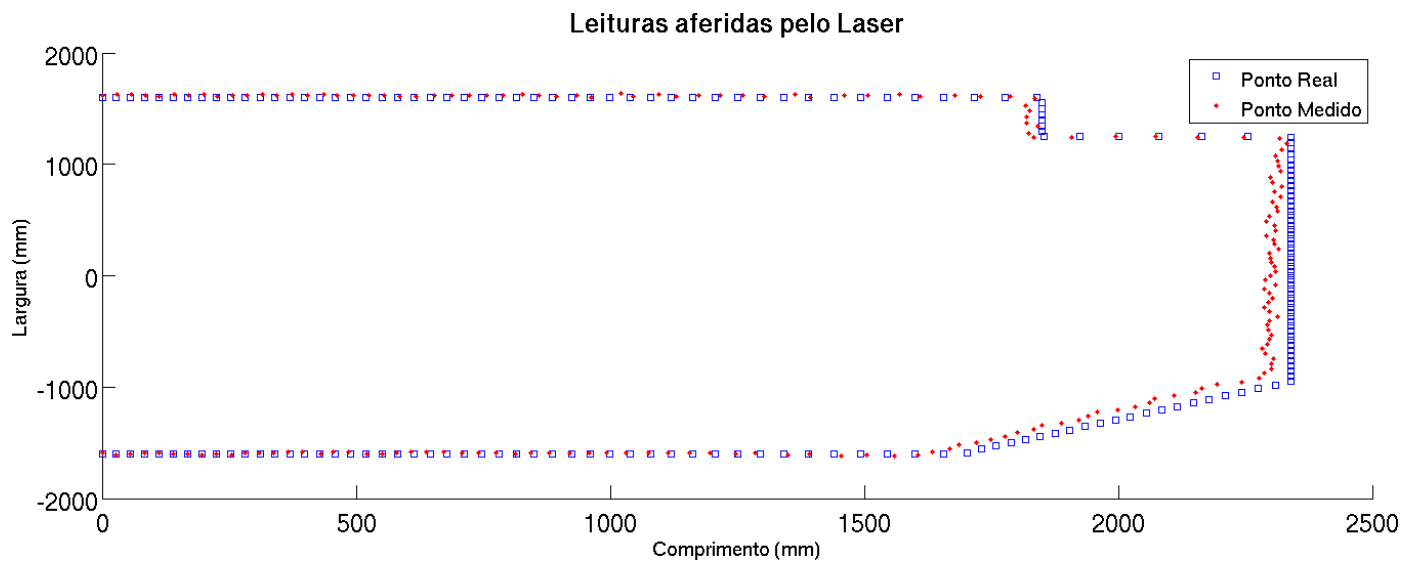


Fig. 3.2: Leituras do Laser

a qualidade de extração de informação do ambiente por intermédio do *laser* é muito superior à do sonar. Além do mais, para os objetivos futuros, principalmente os que envolvem fusão de sensores, a utilização do sonar poderia vir a prejudicar o sucesso do experimento. Com base nisto, estabeleceu-se que apenas o *laser* seria empregado nos métodos de localização abordados. Portanto, para realizar a fusão de sensores, serão usados o *laser* e a odometria do robô.

3.2.3 Modelagem do Erro Cinemático

O robô usado (vide seção 2.7) contém duas rodas motoras que atuam de forma independente, denominando-o robô diferencial [4]. Desta maneira, o modelo cinemático leva em consideração que

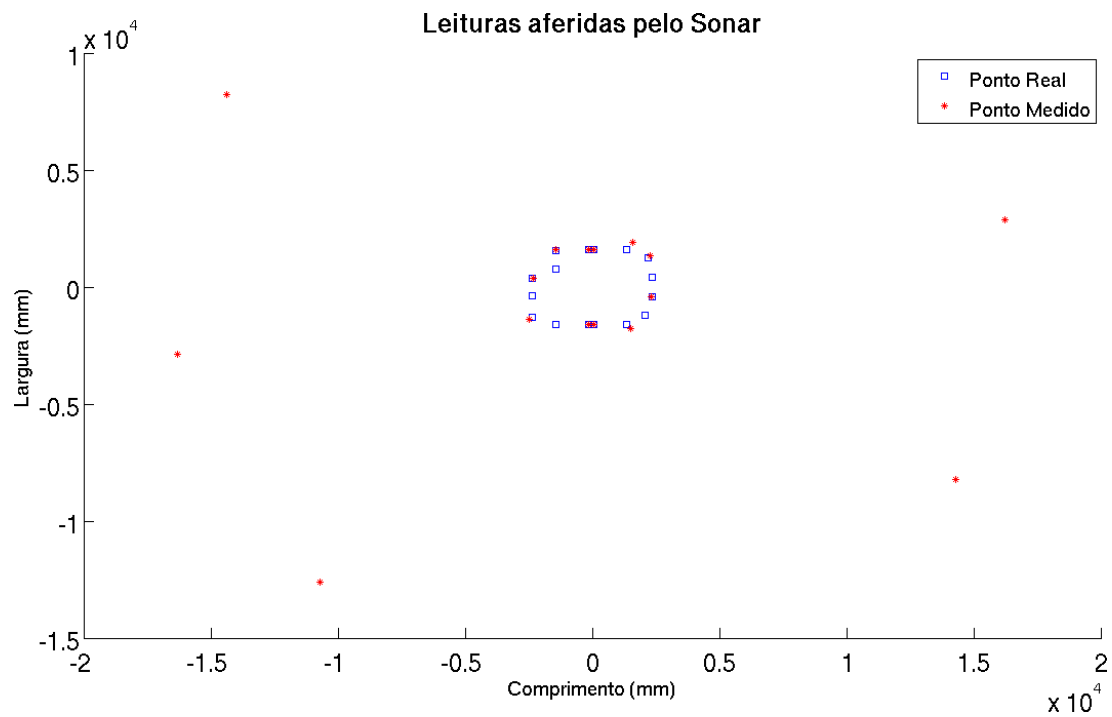


Fig. 3.3: Leituras do Sonar

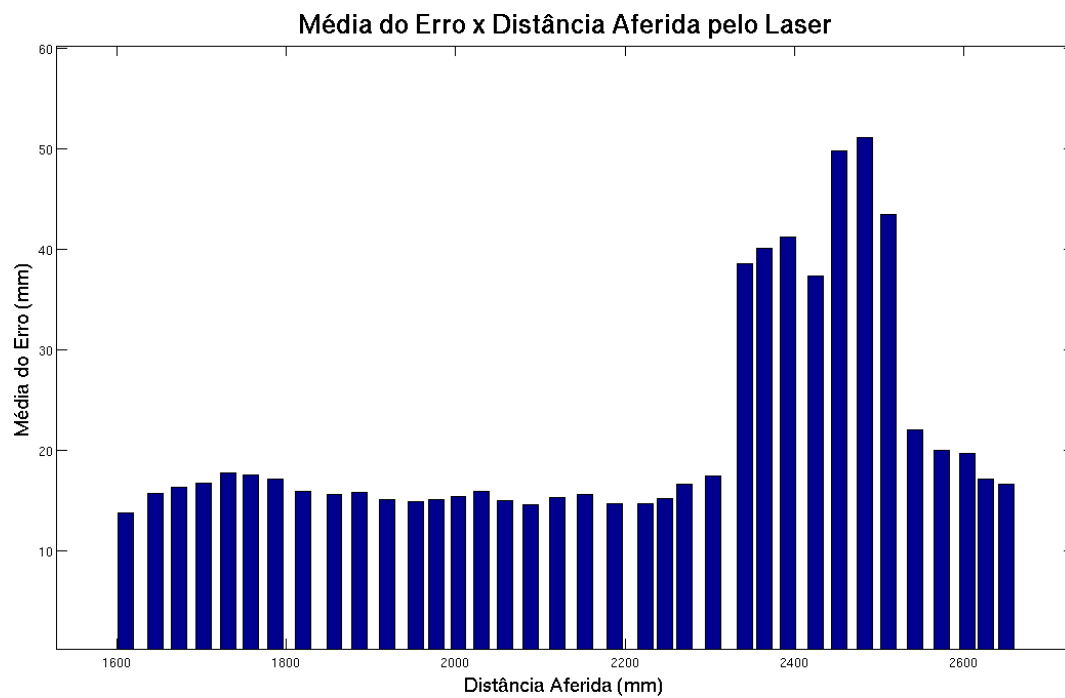


Fig. 3.4: Média do erro das leituras do laser com relação à distância ao obstáculo

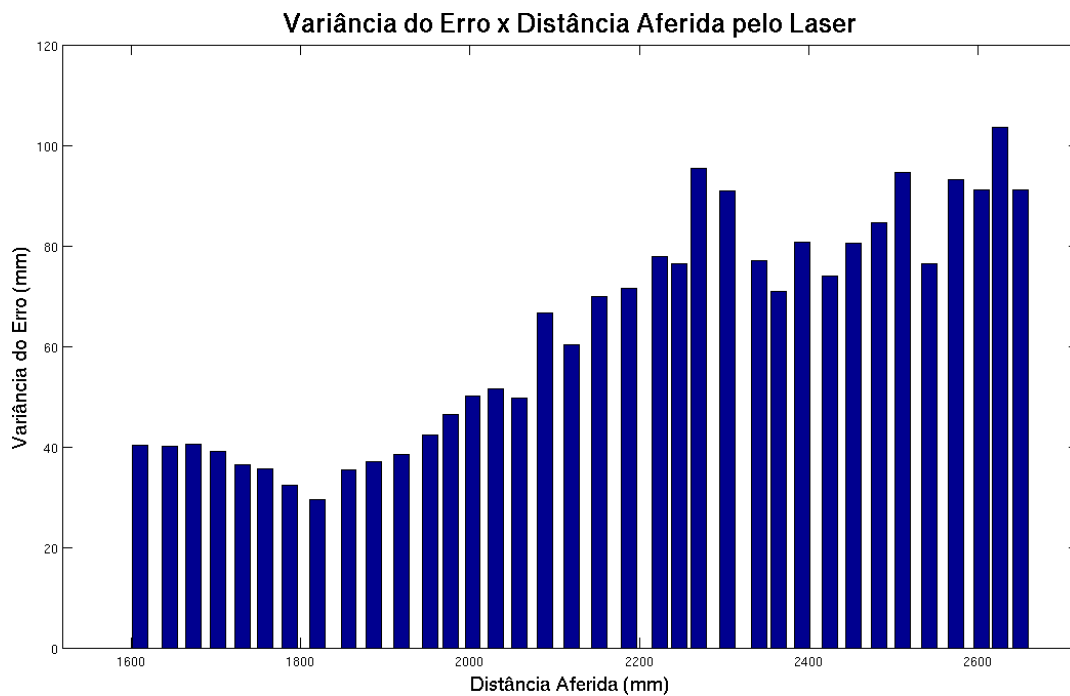


Fig. 3.5: Variância do erro das leituras do laser com relação à distância ao obstáculo

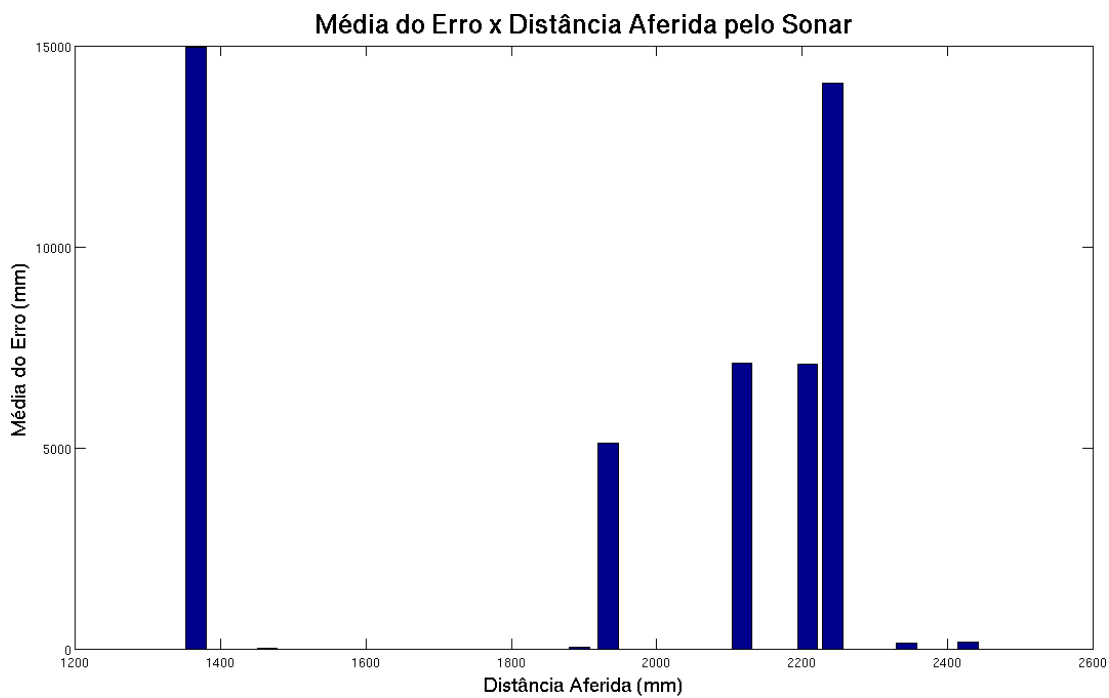


Fig. 3.6: Média do erro das leituras do sonar com relação à distância ao obstáculo



Fig. 3.7: Variância do erro das leituras do sonar com relação à distância ao obstáculo

as rodas podem agir de forma independente, e que a trajetória é definida pela diferença de velocidades entre as duas rodas. A modelagem dinâmica para robôs diferenciais é bastante conhecida e pode ser encontrada em textos como Siegwart and Nourbakhsh, 2004 [4], e Sanches, 2011 [16]. O objetivo desta seção é compreender como o ruído cinemático atua no robô. Na seção 2.4 foram apresentados alguns tipos de erros sistemáticos e não sistemáticos que afetam a odometria, e embora não seja possível ter controle dos erros não sistemáticos, é possível modelar as fontes de erro sistemático, e é este o propósito da modelagem dinâmica.

Para concretizar o modelo de erro cinemático do robô foram efetuados dois experimentos distintos. No primeiro, foram realizadas 10 navegações para cada um dos valores de deslocamento (400, 800, 1500 e 3000 mm), acarretando nos erros médios e variância apresentados na Figura 3.8. A variância particularmente alta do gráfico pode ser resultado do número relativamente baixo de dados (apenas 10), mas se mostraram bastante fiéis à realidade ao longo dos experimentos com os algoritmos de localização. No segundo experimento, executou-se 10 rotações para cada um dos ângulos descritos (90°, 180°, 270° e 360°), culminando nos erros médios e variâncias contidos na Figura 3.9. Ambas as

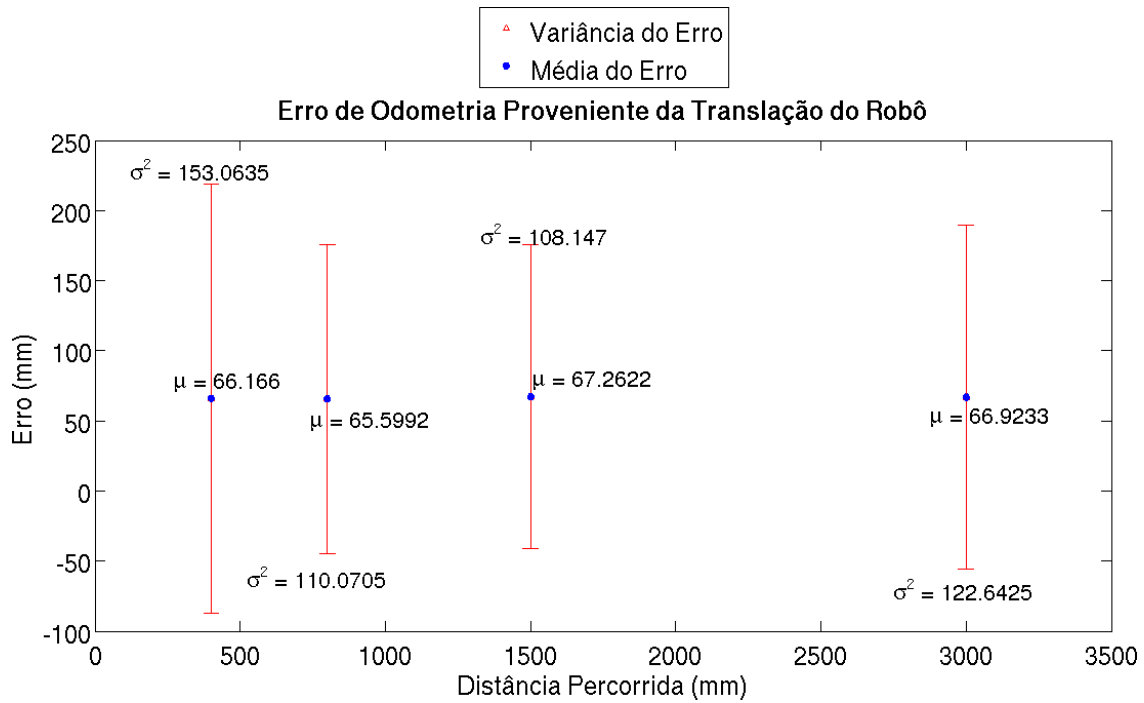


Fig. 3.8: Erro de odometria oriundo da translação do robô

modelagens dinâmicas representam a incerteza envolvida na odometria, e são usados pelas técnicas empregadas nesta dissertação.

3.3 Extração das Características

Os métodos de localização baseados em mapa precisam representar os dados coletados do ambiente por meio de um mapa métrico. Afim de realizar esta tarefa, surge a necessidade de extrair as características desejadas do meio ambiente. Entre os métodos para extração de características mais utilizados está o de segmentação, implementado nesta dissertação por meio da técnica Split-and-Merge. O objetivo deste método é caracterizar as leituras feitas de um ambiente para possibilitar a futura localização do robô ou correção de sua postura. Nesta abordagem, a característica extraída dos dados são retas.

O algoritmo Split-and-Merge (algo como Dividir-e-Fundir) pode ser dividido em duas partes distintas, *split* e *merge*. A descrição de cada uma destas partes são apresentadas no Algoritmo 1

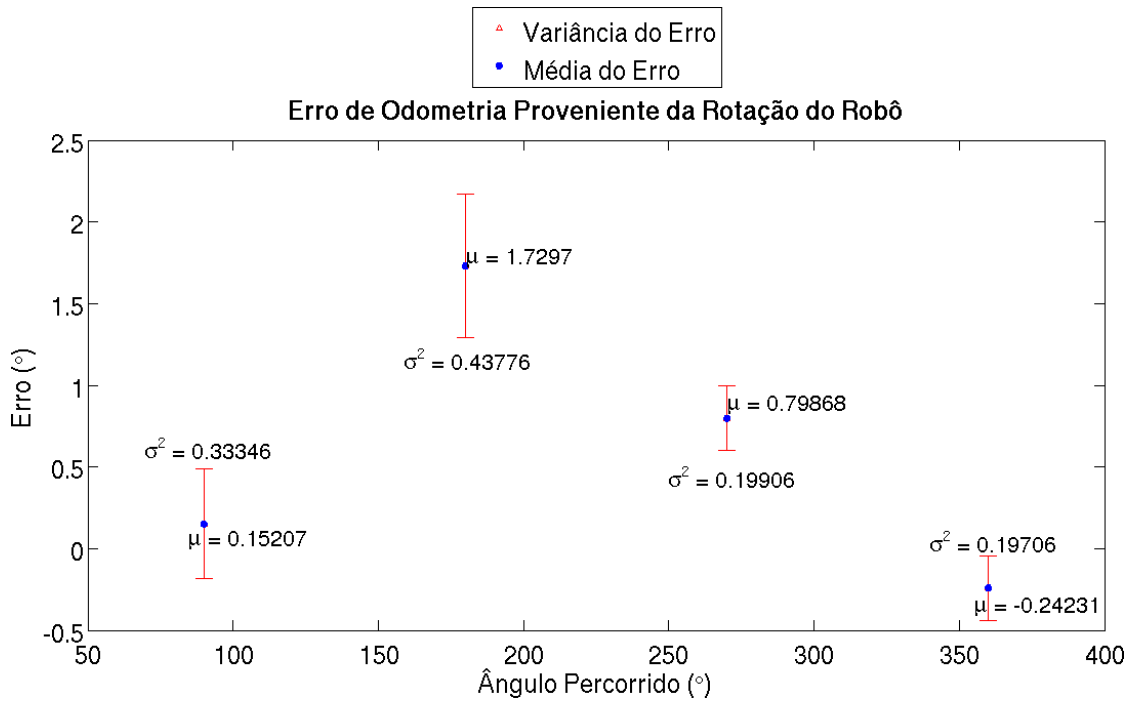


Fig. 3.9: Erro de odometria oriundo da rotação do robô

e Algoritmo 2 respectivamente. O funcionamento do Split-and-Merge pode ser melhor entendido pela visualização da Figura 3.10. O método deve gerar um mapa compatível com as necessidades do algoritmo e do sistema. A seguir, vários testes foram feitos em ambiente real e os resultados são apresentados. Esta análise é de grande importância pois o sucesso da localização depende bastante da qualidade do mapa utilizado. Para mais detalhes, ver Arras, [25].

Algoritmo 1 Split [25]

- Definir um valor limite de distância (*threshold*);
 - Traçar uma semi-reta entre os pontos da extremidade de sua caracterização do ambiente;
 - Achar o ponto mais distante a esta semi-reta;
 - Se a distância deste ponto até a semi-reta for maior que o *threshold*, divide-se a semi-reta no ponto que forneceu a maior distância.
-

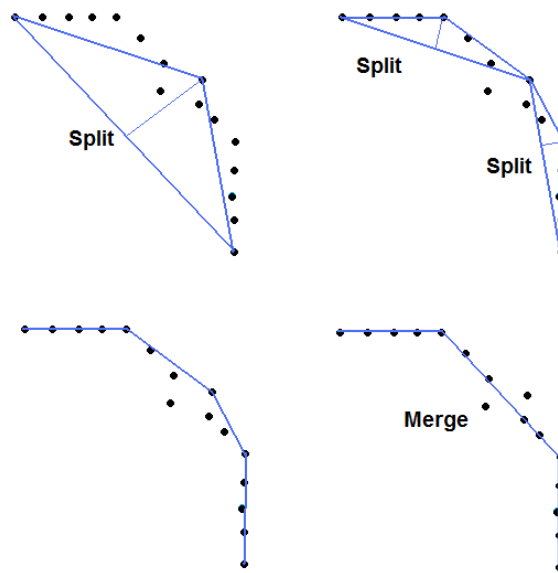


Fig. 3.10: Split-and-Merge

Algoritmo 2 Merge [25]

- Se dois segmentos de reta consecutivos são suficientemente colineares (coeficientes angulares de valor próximo), obter uma semi-reta ligando as extremidades destes dois segmentos e achar o ponto mais distante a ela;
 - Caso a distância do ponto mais distante a ela seja menor ou igual ao *threshold* estabelecido na rotina *split*, deve-se fundir (*merge*) os dois segmentos de reta em um só.
-

3.3.1 Variação do Número de Passos

Ao se variar o passo de captura de pontos do ambiente pelo laser, pode-se perder informações valiosas. Por outro lado, a carga computacional exigida é menor e o processo como um todo, mais rápido. A Figura 3.11 mostra a caracterização do ambiente após a rotina *Split-and-merge* para alguns valores de passos (ângulos entre as leituras). Esta análise permite ao desenvolvedor determinar o valor de passos que otimiza o sistema de extração em termos de velocidade de processamento sem que haja perdas significativas nas informações provenientes do ambiente. É possível notar que

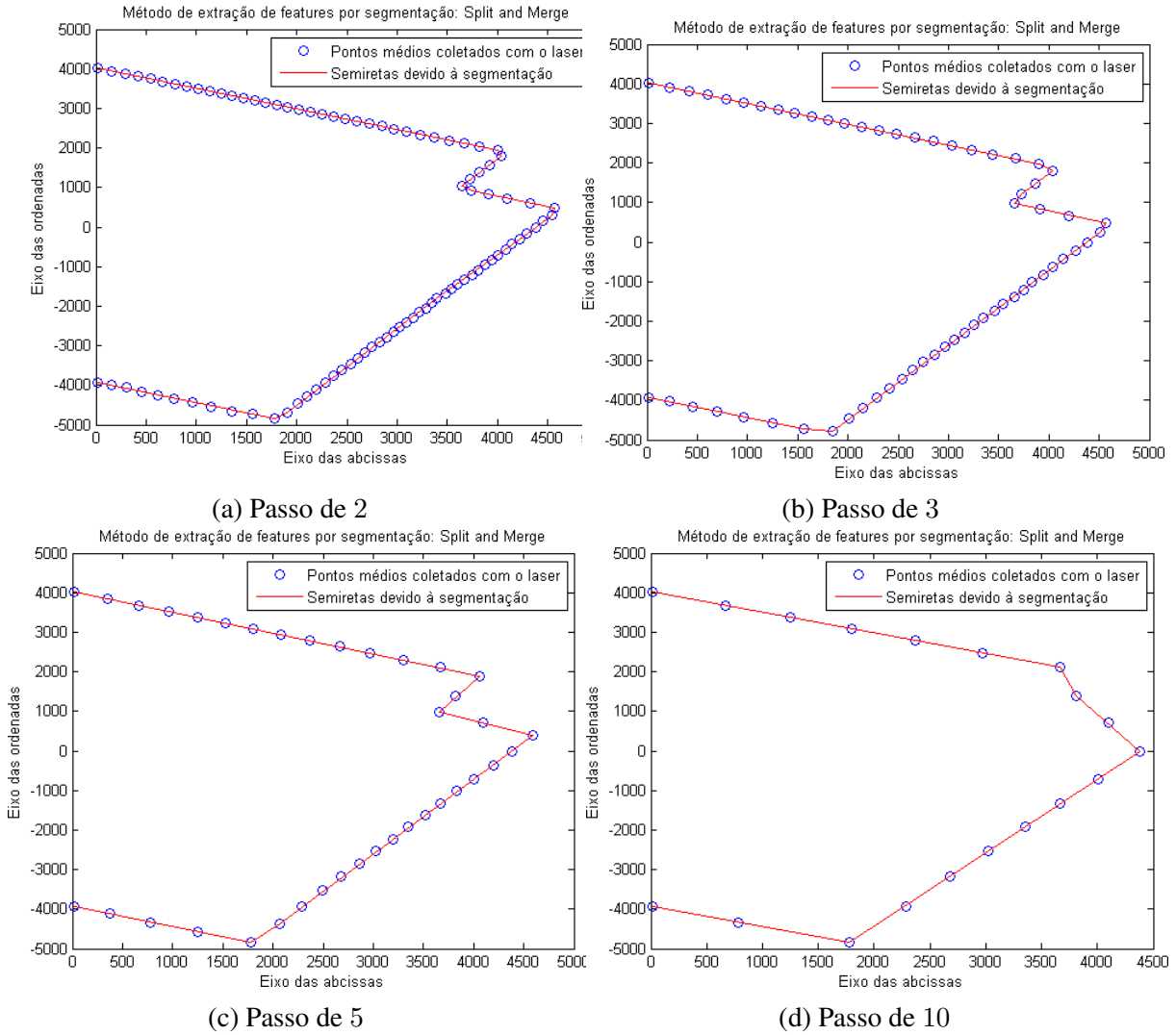


Fig. 3.11: Split-and-Merge de acordo com o quantidade de passos para o laser

quanto maior for o passo (Eq. 3.4), menor será a quantidade de pontos de laser usados, resultado em uma deformidade na estrutura gerada. Esta deformidade deve ser rejeitada, já que o mapa gerado deve ser o mais próximo possível do mapa real a fim de não comprometer a operação dentro dos algoritmos de localização.

$$Quantidade\ de\ Pontos\ Extraídos = -90^{\circ} : passo : 90^{\circ} \quad (3.4)$$

3.3.2 Variação do *Threshold*

O *threshold* é um parâmetro utilizado no método de segmentação para extração de características do ambiente. A variação deste parâmetro influencia nitidamente na fidelidade do mapa de mundo criado para consulta do robô. A Figura 3.13 guarda o resultado da variação deste parâmetro. É clara a variação no número de segmentos conforme o *threshold* aumenta. Pode ser visto na Figura 3.12 a relação entre a quantidade de segmentos de reta que o algoritmo deve armazenar conforme este parâmetro varia.

Pela Figura 3.12 é possível ver que o número de segmentos necessários para recriar o mapa de mundo varia inversamente proporcional ao valor do *threshold* imposto no método de extração de características do ambiente. Se o espaço para o armazenamento desses dados for um aspecto crítico na decisão de que método de extração o programador irá utilizar, ele deverá levantar um gráfico como o da Figura 3.12, a fim de determinar o *threshold* mais adequado.

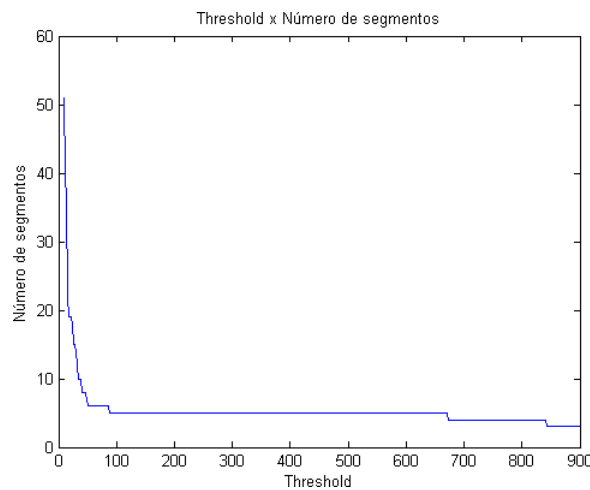


Fig. 3.12: Threshold (*mm*) x número de segmentos

3.4 Conclusão

Neste capítulo foram apresentados as classificações mais usuais para os sensores, uma descrição das características de performance dos mesmos, assim como a modelagem de erro tanto dos sensores

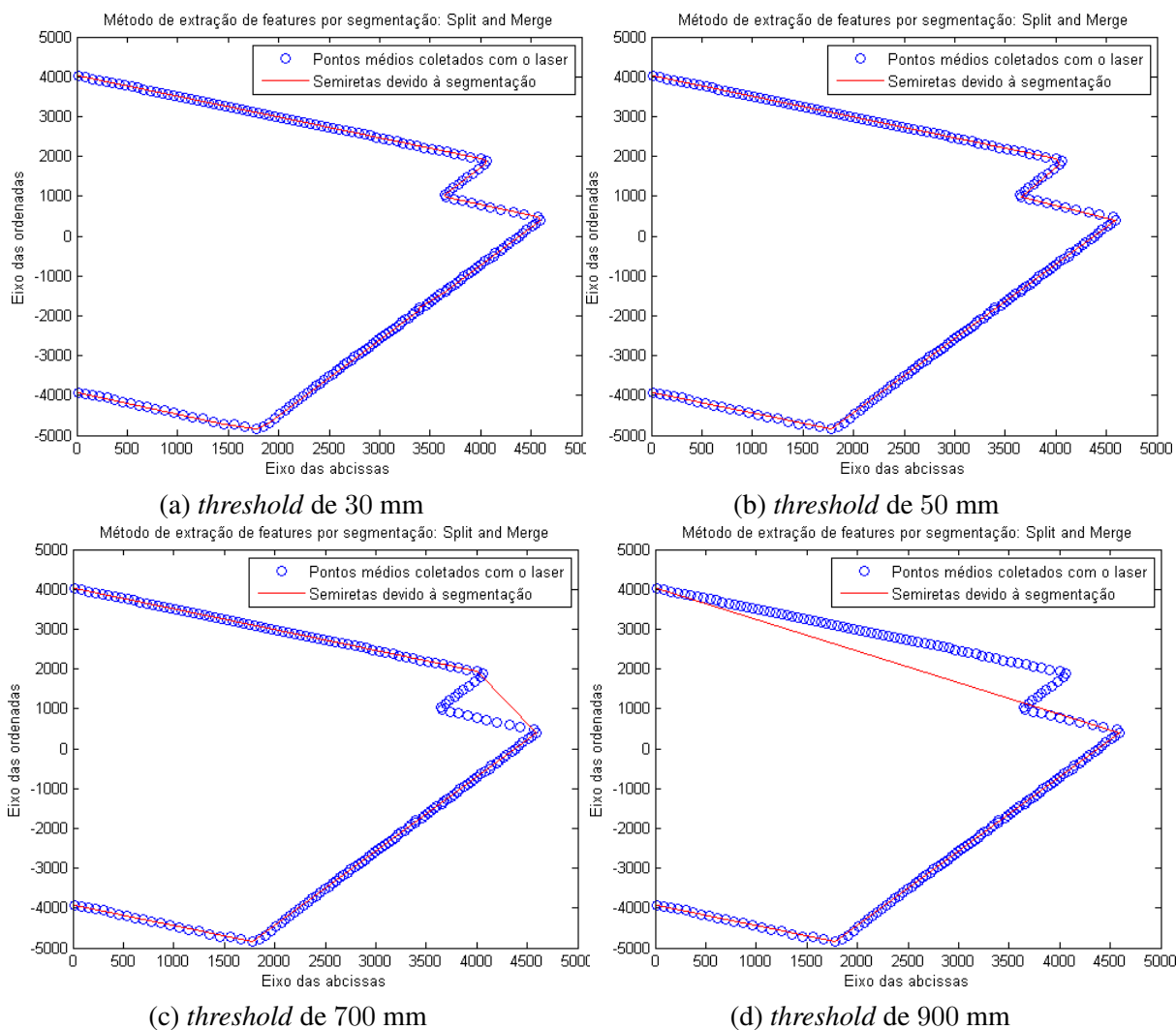


Fig. 3.13: Split-and-Merge de acordo com o valor de *threshold*

exteroceptivos como proprioceptivos. Com base nas informações coletadas, observou-se que o uso do *sonar* poderia comprometer o experimento ao contribuir para o aumento do erro, culminando na decisão de usar apenas o *laser* e odometria. Por fim, foi demonstrado o método de extração de características utilizado nos métodos de localização empregados nesta dissertação, analisando o efeito de seus parâmetros na qualidade do mapa gerado.

Capítulo 4

Filtro de Bayes

4.1 Introdução

Métodos Bayesianos fornecem uma ferramenta bastante genérica para trabalhar com problemas dinâmicos de estimação de estados. É a forma mais básica para o cálculo de uma crença *bel*, feita geralmente a partir das medidas e dos dados de controle. Além disso, é um filtro recursivo, o que significa que a crença em um determinado instante t é calculado com base na crença de um instante passado $t - 1$ [7, 26]. Em outras palavras, o filtro estima qual a probabilidade do robô estar na posição (x, y) , dado que as medidas feitas pelos sensores durante todo o rastreamento foram (z_1, z_2, \dots, z_t) [7, 27]. O filtro de Bayes é baseado na teoria matemática de filtragem probabilística descrita por Jazwinski [11].

Para um sistema de localização ser considerado Bayesiano, ele deve ter as seguintes características [11]:

- Distribuição à Priori: deve existir uma distribuição à priori sobre o estado do alvo. Se este alvo é capaz de movimentação, esta distribuição deve incluir uma descrição probabilística das características de movimentação do alvo. Geralmente, esta distribuição à priori é dada em termos de um processo estocástico.

- Funções de Verossimilhança: A informação contida nas medições devem ser caracterizadas por funções de verossimilhança.
- Distribuição à Posteriori: A saída básica de um sistema Bayesiano é a distribuição de probabilidade à posteriori sobre o estado. O estado à posteriori no tempo t é calculado combinando a atualização de movimentação antes do instante t com a função de verossimilhança das observações recebidas no instante t .

Muitas das técnicas de localização se baseiam em algum aspecto destas características, como o filtro de Kalman e o filtro de partículas, ou ainda no monitoramento multi-hipótese (*multihypothesis tracking*), como na localização baseada em grade, em abordagens topológicas, etc.

4.2 Descrição e Implementação

Como o filtro de Bayes utiliza todo o histórico de medidas feitas pelos sensores, a complexidade computacional aumenta exponencialmente ao longo do tempo devido ao aumento do número dessas medições. Os métodos que se baseiam no filtro de Bayes usam diferentes formas para o sistema dinâmico, sanando este problema. Ao se manter estritamente no algoritmo, geralmente se opta por escolher uma aplicação simples, de modo a não exigir um grande poder computacional, como o caso de estudo deste capítulo. A Tabela 4.1 reúne algumas das características da implementação baseada em filtro de Bayes desta dissertação. As informações coletada do ambiente são os estados das portas ao longo do corredor. Esta informação é usada para localizar o robô, e será detalhada na seção 4.3. A informação das portas abertas são coletadas a partir de duas leituras do laser para os ângulos mostrados na Figura 4.2.

Característica	Filtro de Bayes
Localização	Global
Mapa	Métrico
Informação do Ambiente	Portas abertas

Tab. 4.1: Características do Método de Filtro de Bayes Implementado

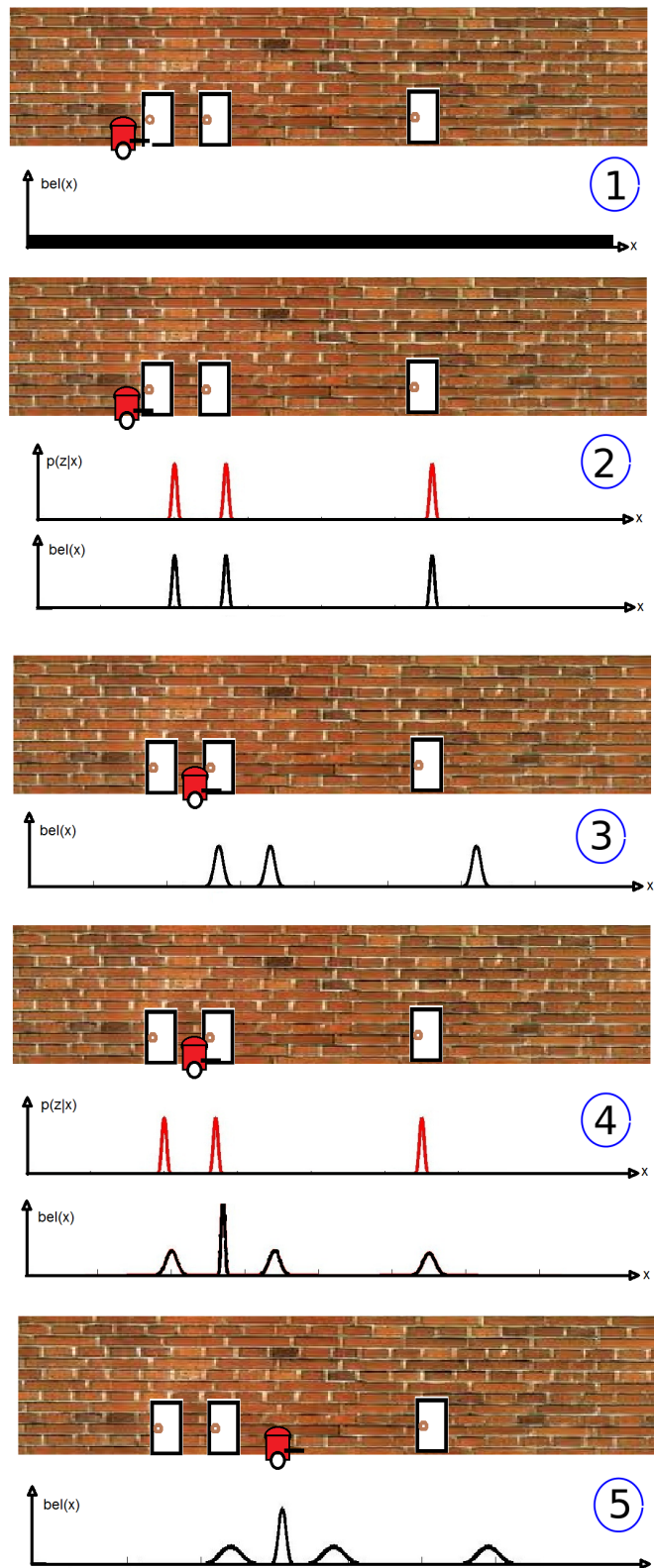


Fig. 4.1: Exemplo clássico do uso de localização robótica por filtro de Bayes

Ângulos de Leitura do Laser para o Filtro de Bayes

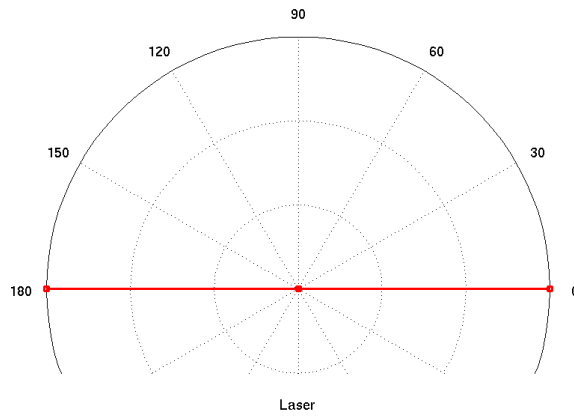


Fig. 4.2: Ângulos de Leitura do Laser para o Filtro de Bayes

O objetivo pretendido com a aplicação do filtro de Bayes nesta dissertação é reproduzir o exemplo clássico de localização robótica, em que um robô navega por um corredor observando o estado das portas, utilizando esta informação para se localizar, como ilustrado na Figura 4.1.

Os estados das portas ao longo do corredor são determinados através de leituras de *laser*. Caso retornem valores maiores que a de um valor fixo pré determinado (*threshold*), assume-se que o robô encontrou uma porta aberta, caso contrário, assume-se que a leitura foi feita em uma porta fechada ou parede. O *threshold* é determinado antes do início da navegação, ainda com o robô parado. São feitas duas leituras de laser nos ângulos 0° e 180° , perpendiculares à orientação do robô, a fim de medir a distância inicial entre o robô e as duas paredes que compõem o corredor. No decorrer da navegação, o método de Bayes utiliza estes dois valores para comparar com as leituras feitas ao longo do processo e distinguir leituras feitas em portas abertas.

O primeiro cenário da Fig. 4.1 mostra o robô iniciando a navegação. A crença ($bel(x)$) é constante pois, inicialmente, a estimativa de posição é igual para qualquer ponto ao longo do corredor, uma vez que nenhuma informação foi computada ainda. Na segunda etapa, o robô percebe a primeira porta do corredor. Utilizando a distribuição de portas do corredor ($p(z|x)$) à priori, o sistema atualiza a crença. As três portas do corredor são representadas pelos três lóbulos formados pela função gaussiana. Estes lóbulos têm suas médias e desvio padrões de acordo com as observações feitas das portas. Na terceira

fase, o robô se desloca até a segunda porta, inserindo erro de odometria em sua crença. Como resultado, os lóbulos são deslocados no mesmo sentido de movimento do robô e o desvio padrão destes se tornam maiores, acompanhando o aumento da incerteza que representam. Na quarta etapa, o robô observa que encontrou a segunda porta e adiciona esta informação ao sistema, utilizando para isso, a mesma distribuição das portas. Com esta segunda observação, a representação da crença de posição se modifica, guardando probabilidades distintas para os lóbulos presentes. Na quinta etapa, o robô volta a se deslocar e a incerteza de movimento é novamente incorporada à crença.

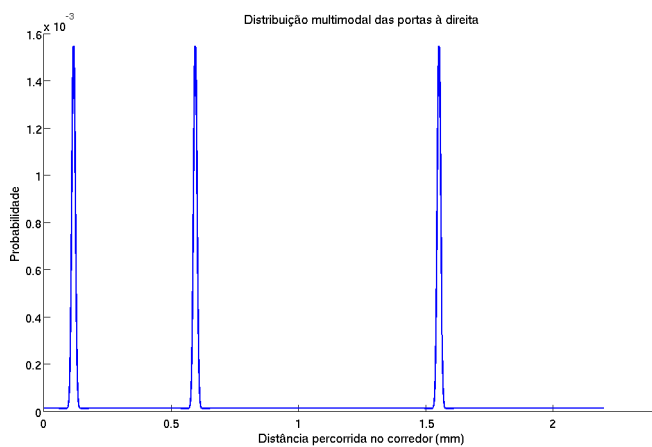
No cenário proposto para este método, o robô age conforme a explicação para a Figura 4.1, utilizando a distribuição multimodal das portas ao longo do corredor. O robô conhece em que sentido está navegando, neste caso, em direção ao ponto sul (vide Cap. 1), e assume-se que navega paralelamente às paredes que compõem o corredor com distâncias iguais a ela. Assim, a localização do robô é feita em apenas uma dimensão, x .

O filtro de Bayes trabalha diretamente com a crença do estado do robô, que neste experimento trata-se da posição (x) do robô ao longo de um corredor. O filtro de Bayes, exposto no Algoritmo 3 [7], é dividido em duas etapas:

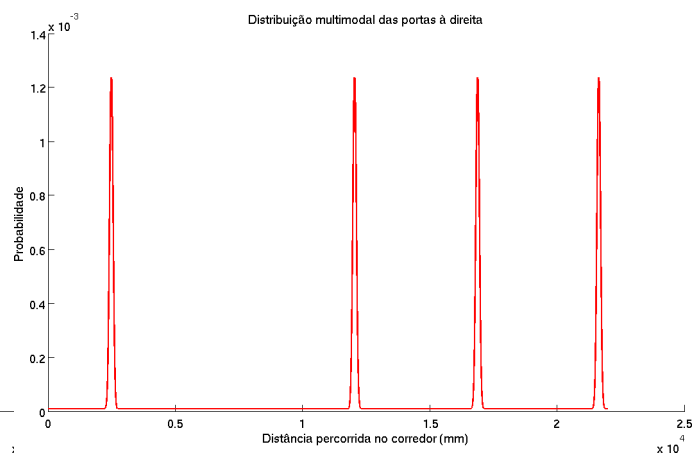
- Fase I: Estimação do estado atual (linha 1): a partir do estado anterior, do deslocamento captado pelo *encoder* (odometria) entre os instantes t_k e t_{k-1} e pelo modelo de erro dinâmico (subseção 3.2.3), uma nova estimativa de estado é feita.
- Fase II: Correção desta estimativa (linha 2): com base nas leituras feitas pelo laser e pelo modelo de erro estático (vide subseção 3.2.2), o sistema atualiza o estado estimado da fase I.

e para usá-lo, é necessário discutir como os termos destas equações são encontrados e definidos.

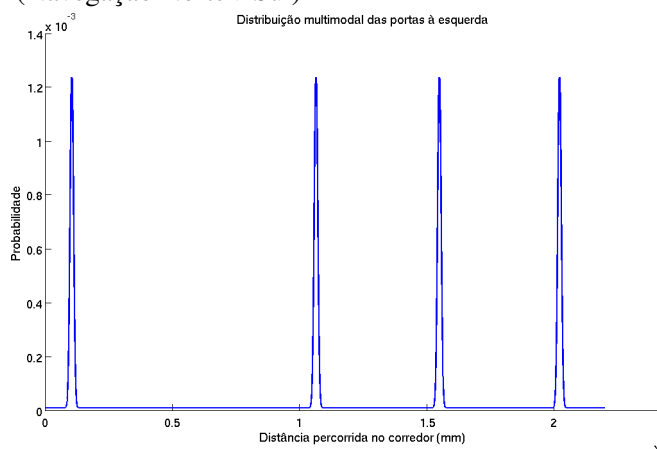
No Algoritmo 3, $p(x_t | u_t, x_{t-1})$ descreve a dinâmica do sistema, bel' é a crença estimada, η é uma constante de normalização e $p(z_t | x_t)$ descreve o modelo perceptual, que contém a possibilidade de uma dada característica z_t ser encontrada dado que o robô está na posição x_t . A especificação dos termos $p(x_t | u_t, x_{t-1})$ e $p(z_t | x_t)$ são imprescindíveis para a implementação, e suas definições serão descritas a seguir.



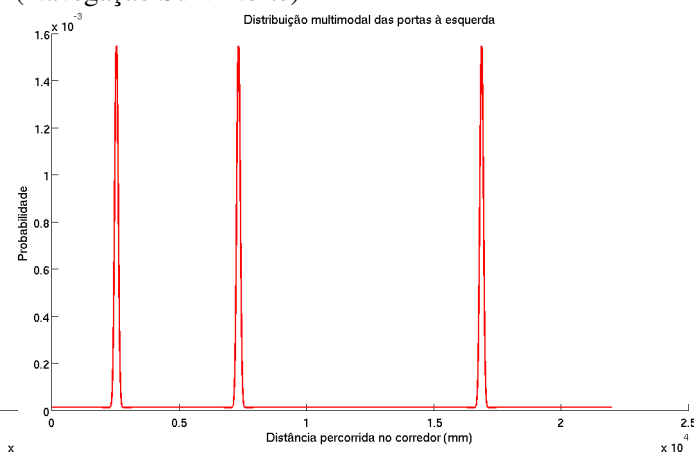
(a) Distribuição de Probabilidade das Portas à Direita (Navegação Norte->Sul)



(b) Distribuição de Probabilidade das Portas à Direita (Navegação Sul->Norte)



(c) Distribuição de Probabilidade das Portas à Esquerda (Navegação Norte->Sul)



(d) Distribuição de Probabilidade das Portas à Esquerda (Navegação Sul->Norte)

Fig. 4.3: Distribuição Multimodal das portas no corredor

- Dinâmica do Sistema

A dinâmica do sistema foi definida utilizando-se a odometria do robô, tornando possível obter uma estimativa de posição para cada estado. Ao navegar ao longo de um corredor sempre em linha reta, o deslocamento do robô se tornou unidimensional.

- Modelo Perceptual

Para a definição deste modelo, deve-se levar em conta quais características do ambiente irão ser extraídas pelo robô. Para isto, foram utilizadas portas abertas, cujas distribuições podem ser vistas na Figura 4.3.

Como a distribuição de probabilidade envolve todos os estados possíveis ao longo do corredor e como nele há mais de uma porta, esta distribuição é multimodal, representando cada porta aberta por um lóbulo.

4.3 Experimento e Análise dos Resultados

A distribuição das portas é representada por uma função densidade de probabilidade multimodal, que configura a probabilidade de se encontrar uma porta em uma dada altura do corredor. As distribuições das portas à direita e à esquerda são apresentadas na Figura 4.3. Como o robô navega primeiramente em direção ao ponto sul e posteriormente em direção ao ponto norte, as portas que ficam à direita e à esquerda são trocadas quando o sentido de navegação muda, e é por isso que a figura em questão discrimina também o sentido da navegação. Apesar de saber o sentido da navegação, o robô não conhece sua posição inicial.

Algoritmo 3 Filtro de Bayes ($bel(x_{t-1}), u_t, z_t$) [7]

```
1: for all  $x_t$  do  
2:    $\bar{bel}(x_t) = \int p(x_t|u_t, x_{t-1})bel(x_{t-1})$   
3:    $bel(x_t) = \eta p(z_t|x_t)\bar{bel}(x_t)$   
4: end for  
5: retorna  $bel(x_t)$ 
```

A Figura 4.4 contém os resultados do experimento (em ambiente real). Nela é possível acompanhar o progresso da crença ($bel(x)$) sobre o estado do robô conforme este percorre sua trajetória através do corredor. Esta figura descreve os estados da crença da posição desde o início (antes mesmo da primeira porta) até a última porta da primeira parte da navegação (até chegar ao ponto sul) ser observada. O estado da crença na segunda parte da navegação não foi representada na figura para manter a clareza da figura, uma vez que as curvas da segunda parte sobreporiam as curvas da primeira parte da navegação. É interessante notar como a crença da posição se torna melhor à medida em que mais estados são introduzidos no filtro, gerando uma estimativa unimodal e com desvio padrão menor que as funções gaussianas iniciais, o que é causado pelo aumento de certeza sobre o estado atual do robô. Na fase de estimação do estado o erro presente na odometria é inserido no sistema.

O experimento foi executado 10 vezes utilizando o cenário descrito na seção 1.5. A Tabela 4.2 mostra o resultado obtido no experimento. O erro foi considerado razoável, principalmente pelo fato de não haver qualquer conhecimento inicial sobre o estado do robô. É importante lembrar que a forma pela qual o experimento foi montado faz com que o filtro trabalhe com dados em apenas uma dimensão, justificando os valores baixos de erro.

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Postura (x) [mm]	31.4800 ± 14.2872	2.8517	59.7014
Orientação (θ) [°]	-	-	-

Tab. 4.2: Resultados dos experimentos do Filtro de Bayes

Medição	IC de 95% do erro	Melhor Resultado	Pior Resultado
Tempo de execução [s]	0.0078 ± 0.0007	0.0058	0.0141
Consumo de CPU [%]	21.66 ± 1.63	19	26
Quantidade de Bytes alocados [KB]	3113.046 ± 90.319	3003.461	3353.717
Linhas de código	87	-	-

Tab. 4.3: Medições feitas com base no Filtro de Bayes

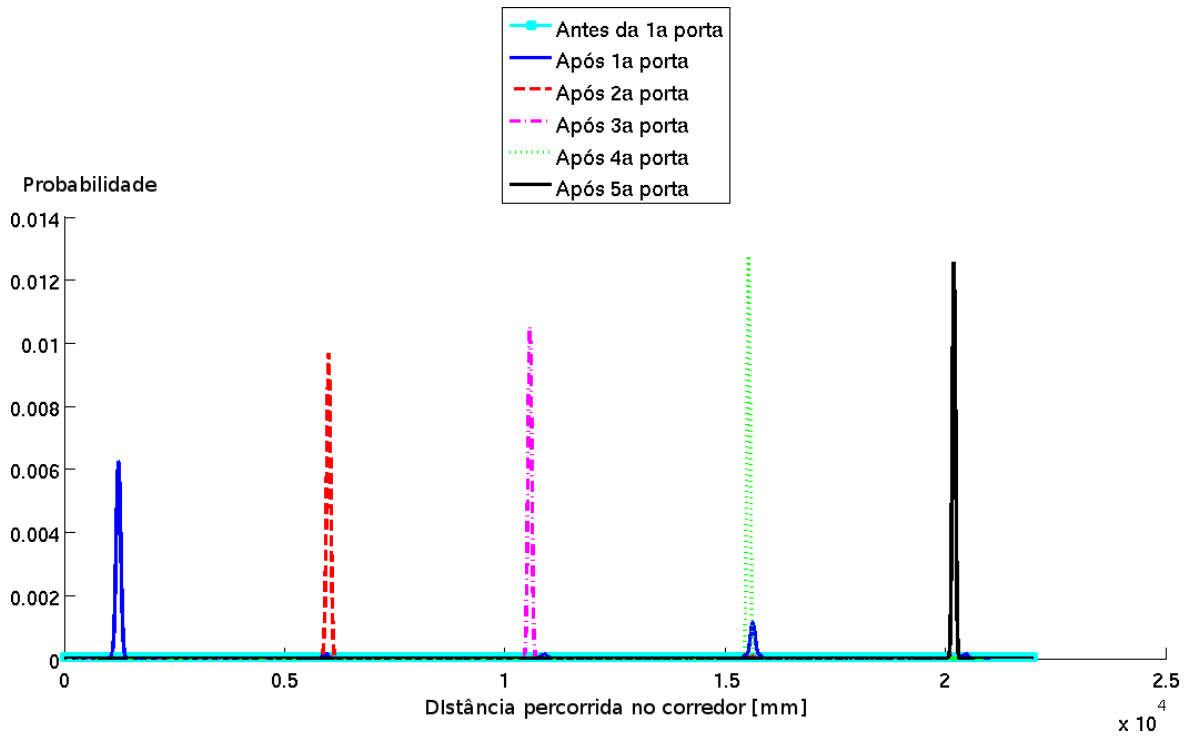


Fig. 4.4: Evolução da crença em ambiente real

4.4 Reconhecimento de uma Topologia

Uma das contribuições originadas a partir do estudo desta dissertação foi o método de reconhecimento de topologias descrito pelo artigo D. Rodrigues *et al.*, 2012 [2]. Embora não seja um método de localização, pode servir a este propósito e será descrito a seguir.

A localização de um robô em um mapa pode acontecer em níveis mais altos do que a estimação de sua postura, como no caso da localização em topologias. A localização topológica estima a topologia do mapa em que o robô se encontra, o que pode ser suficiente para diversas aplicações. Para a realização deste tipo de localização, é imprescindível que haja um método eficiente de classificação das observações em topologias. O artigo D. Rodrigues *et al.*, 2012 [2], trata do reconhecimento de topologias, utilizando para isso, o filtro de Bayes e Fuzzy.

O reconhecimento de topologias faz uso de características específicas (*landmarks*) topológicas para atualização da crença do sistema. Esse tipo de informação serve a diversos propósitos, inclusive o de localizar o robô à medida que este navega pelo mapa e passa pelas topologias. Nesta dissertação,

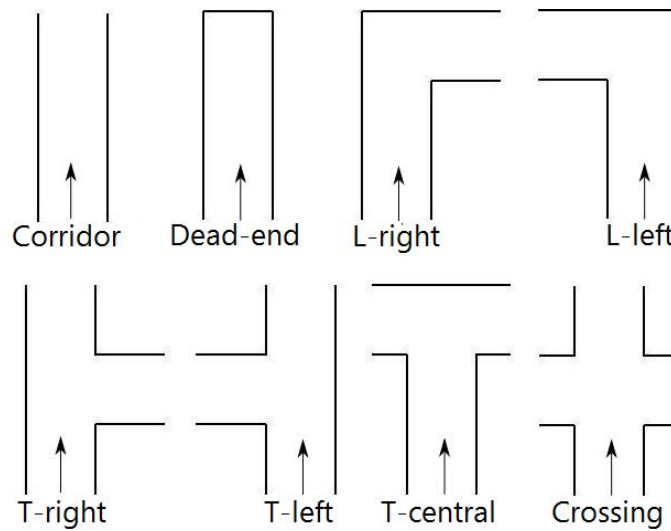


Fig. 4.5: Topologias usadas [2].

o termo topologia é empregado para definir cada tipo de cruzamento visto na Figura 4.5, presentes no mapa do estudo de caso. Em um corredor com portas, o estado destas (abertas ou fechadas) ao longo do corredor formam as topologias necessárias para a implementação do filtro na forma como foi proposta.

No experimento descrito no artigo, o robô navega pelo mapa, faz leituras do ambiente usando o *laser*, extrai informações destas leituras, classifica a informação como umas das topologias pré-definidas (Fig. 4.5), e utilizada esta classificação como entrada no algoritmo de localização. O artigo Rodrigues *et al.*, 2012 [2] fornece mais detalhes sobre reconhecimento de topologias para aplicações em robótica assistiva.

A Tabela 4.4 contém os valores de estimação após o robô chegar ao centro das topologias. Cada linha começa com o símbolo que representa a topologia real no qual o robô navegou, e cada célula na linha relaciona seu valor de estimação com o símbolo no topo da coluna.

4.5 Conclusão

Apesar das limitações de uso do filtro de Bayes, sua implementação é bastante válida e interessante, visto que inúmeros métodos de localização robótica são baseados nele. Além disso,

Tab. 4.4: Estimação de topologias quando o robô alcança seu centro [2]

	Estimação da topologia (%)							
	\parallel	Γ	$\bar{\Gamma}$	Π	$\bar{\Gamma}$	$\bar{\Gamma}$	$\bar{\Gamma}$	$\bar{\Gamma}$
\parallel	100	-	-	-	-	-	-	-
Γ	-	100	-	-	-	-	-	-
$\bar{\Gamma}$	-	-	100	-	-	-	-	-
Π	-	-	8	92	-	-	-	-
$\bar{\Gamma}$	-	-	-	-	100	-	-	-
$\bar{\Gamma}$	-	-	-	-	-	96	-	4
$\bar{\Gamma}$	-	-	-	-	-	-	100	-
$\bar{\Gamma}$	-	-	-	-	-	-	-	100

para aplicações específicas, o filtro pode vir a ser o mais recomendável, pela sua simplicidade e rapidez de execução, como é o caso do cenário estudado neste capítulo. Apresentamos ainda, como contribuição extra, uma forma de utilizar o filtro de Bayes para localizar o robô topologicamente em um mapa.

Capítulo 5

Localização pelo método de Markov

5.1 Introdução

Na localização pelo método de Markov, a crença do robô sobre seu estado é geralmente representada por uma função densidade de probabilidade multimodal, para representar cada possível postura do robô no mapa. O método é razoavelmente robusto devido ao fato do robô estar averiguando múltiplas posições possíveis ao mesmo tempo. Esta técnica apresenta três grandes vantagens: poder estimar a postura do robô globalmente; poder eficientemente rastrear a postura quando o robô apresentar baixa incerteza; e ainda detectar e se recuperar de falhas de localização [4, 28].

Entretanto, para atualizar a probabilidade de todas as posições possíveis é necessária uma representação do espaço, como uma grade por exemplo. Quando o mapa é representado dessa forma, ele precisa repartir o ambiente em um número finito e discreto de possíveis posições para o robô (incluindo sua orientação) [4]. Desta forma, os processos de estimação e de correção (descritos na seção 4.2) precisam atualizar a probabilidade de todas as células da grade a cada iteração. Com isso, a memória requerida e o poder computacional podem agir como limitadores tanto para a precisão do método quanto para o tamanho do mapa.

Com base na sua concepção de postura, um mecanismo poderoso de atualização é usado para calcular a nova crença do estado, resultado da incorporação de uma nova informação junto à antiga

crença do estado, utilizando para isso a fórmula de Bayes [4, 7]. A descrição matemática do método pode ser encontrada em diversos textos ([4, 7, 10]) e o método implementado aqui se encontra no Algoritmo 4.

No livro Siegwart and Nourbakhsh, 2004 [4], são encontrados dois casos de estudo da localização robótica utilizando o método de Markov, um caso com mapa topológico e outro com mapa em grade. O artigo Burgard and Derr, 1998 [28], traz uma integração da estimação da postura global com o rastreamento de postura usando localização pelo método de Markov, o artigo Burgard et al., 1994 [29], traz uma fusão da localização pelo método de Markov e por filtro de Kalman, herdando a precisão do filtro de Kalman e a robustez e velocidade do método de Markov, e o artigo Gutmann et al., 1998 [30], compara o método de Markov com outros métodos.

5.2 Descrição e Implementação

A Tabela 5.1 traz algumas características de como o método foi implementado neste trabalho. Aqui, o método utiliza um mapa em grade, representando os estados com base no sistema métrico. Como fonte de informação do ambiente, o método utiliza os dados provenientes apenas de dois raios, conforme ilustrado na Figura 5.1. Esta escolha foi feita principalmente para diminuir o poder computacional requerido. Com esse tipo de informação, o robô consegue se localizar à medida que navega pelo mapa e passa por pontos específicos, que neste caso, são os estados das portas (abertas) ao longo do corredor. O sistema não diferencia portas fechadas e as paredes que formam o corredor, e assim, apenas observa portas abertas e parede. A detecção de portas para este método funciona exatamente como no método Filtro de Bayes.

Característica	Método de Markov
Localização	Global
Mapa	Grade
Informação do Ambiente Utilizada	Leituras brutas

Tab. 5.1: Características do Método de Markov Implementado

Ângulos de Leitura do Laser para a Localização por Markov

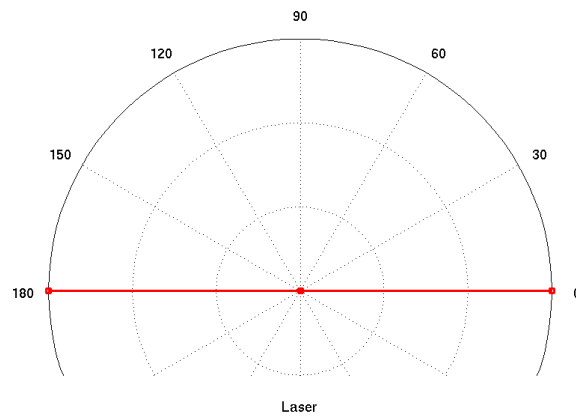


Fig. 5.1: Ângulos de Leitura do Laser para a Localização pelo método de Markov

Primeiramente, o mapa deve ser dividido em células, definindo o comprimento (discretização na dimensão x) e largura (discretização na dimensão y) destas. A partir de testes simulados, dois valores satisfatórios para estes parâmetros foram encontrados: comprimento de aproximadamente 204 mm e largura de 390 mm. Esses valores discretizam o mapa em 11 colunas (dimensão de x) e 60 linhas (dimensão de y). A escolha destes parâmetros influenciam no tempo de processamento e no erro desejável de localização. Ao fim, o sistema retorna a linha e coluna com maior valor de crença, e a posição do robô é definida pelo centro desta célula. Desta forma, quanto maior forem as dimensões escolhidas, maior será a área coberta pela célula e maior o erro de localização. O critério para escolha das dimensões das células levou em conta a eficiência do método e o tempo de execução requerido.

Neste método, o algoritmo percorre as seguintes fases:

- Fase I: Estimação do estado atual (linha 1): a partir do estado anterior, do deslocamento captado pelo *encoder* (odometria) entre os instantes t_k e t_{k-1} e pelo modelo de erro dinâmico (subseção 3.2.3), uma nova estimativa de estado é feita.
- Fase II: Correção desta estimativa (linha 2): com base nas leituras feitas pelo laser e pelo modelo de erro estático (vide subseção 3.2.2), o sistema atualiza o estado estimado da fase I.

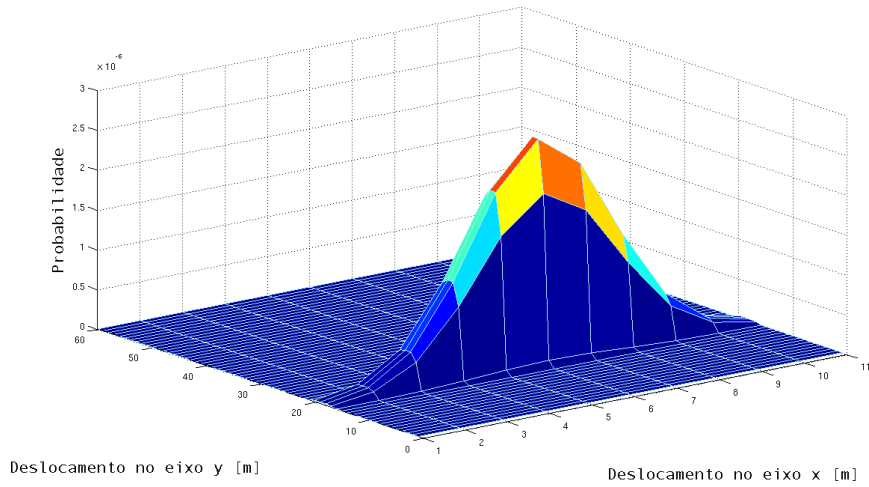
O algoritmo de localização pelo método de Markov tem grande semelhança com o filtro de Bayes,

Algoritmo 4 Localização pelo método de Markov ($bel(x_{t-1}), u_t, z_t, m$) [7]

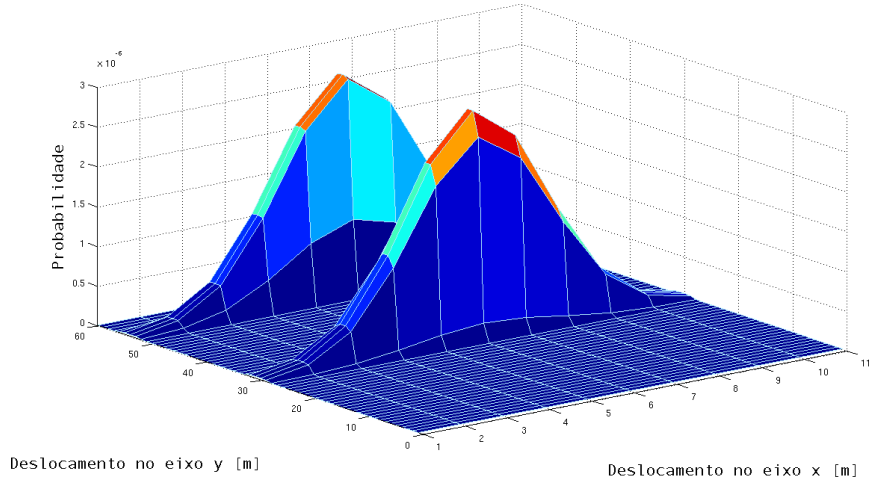
```
1: for all  $x_t$  do  
2:    $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1}, m) bel(x_{t-1})$   
3:    $bel(x_t) = \eta p(z_t|x_t, m) \overline{bel}(x_t)$   
4: end for  
5: retorna  $bel(x_t)$ 
```

já que usa uma distribuição multimodal para representar a crença específica. Ao utilizar o método de Markov, localizamos o robô não apenas com base em sua posição (x, y) , mas também quanto a sua orientação. A orientação do robô transforma a matriz em duas dimensões em uma matriz tridimensional, com o número de camadas iguais à quantidade de valores possíveis para a orientação. Novamente, o tempo de processamento é diretamente proporcional à quantidade de camadas. Para otimizar o tempo de processamento deste algoritmo, foi assumido que o robô estaria navegando sempre paralelo às paredes do corredor, e por tanto, assumiria apenas duas possíveis orientações, 270° (sentido Norte-Sul) e 90° (sentido Sul-Norte). Assim, a crença será trabalhada em duas camadas simultaneamente, descritas pelo sentido de navegação do robô.

No caso do mapa em grade, matrizes extras representam a distribuição de probabilidade dos tipos específicos de características encontradas, como exemplificado na Figura 5.2 para o caso em que as leituras apresentam obstáculo apenas à esquerda do robô. Esta figura representa a crença de se encontrar uma característica deste tipo em cada uma das células da grade, nas duas camadas possíveis, já que inicialmente o robô não conhece sua orientação. Quando o sistema identifica em que característica específica o robô está, a matriz correspondente é utilizada juntamente com a matriz da crença do estado anterior para gerar a matriz que guarda o estado da crença atual. A cada iteração do algoritmo, a distribuição da característica específica observada pelo robô é usada para atualizar a crença de acordo com o algoritmo de localização pelo método de Markov, apresentado no Algoritmo 4. Para atualizar a crença com relação ao eixo x não se pode utilizar as distribuições das portas, já que portas abertas não retornam valores de distâncias úteis. Para isto, os valores anteriores de distância do robô para ambas as paredes é utilizada.



(a) Distribuição no sentido Norte-Sul



(b) Distribuição no sentido Sul-Norte

Fig. 5.2: Distribuição da característica específica (\parallel)

5.3 Experimento e Análise dos Resultados

Apresenta-se a forma como o experimento foi realizado na Figura 5.3. A Figura 5.3a apresenta os pontos do mapa aonde as crenças das figuras subsequentes foram obtidas. As sub-figuras 5.3b até 5.3h apresentam três gráficos de crença, de modo que os gráficos superiores (crenças de direção) representem respectivamente as crenças do robô estar indo do ponto norte ao sul e vice-versa, enquanto o gráfico inferior (crença atual) ilustra a crença combinada dos dois gráficos superiores.

Assim que o sistema é acionado (Figura 5.3b), o algoritmo calcula a a crença atual sobre o estado (postura) do robô. Neste instante, o reconhecimento da característica específica ainda não aconteceu,

e portanto a crença inicial é igual para todas as células. Em seguida, o robô varre os obstáculos locais, compreende que se encontra em um corredor e atualiza a crença em ambas as camadas da crença de direção e da crença atual (Figura 5.3c). O robô então começa a navegar pelo corredor e reconhece outras características específicas ao longo do caminho. A evolução da crença do estado do robô desde o momento inicial até o final nos pontos específicos da Figura 5.3a são representados na Figura 5.3. A crença é originalmente constituída por funções gaussianas, que com o passar do tempo e devido à discretização que a crença sofre para ser representada pelas matrizes, acabam tomando um formato pontiagudo. O experimento foi repetido 10 vezes em ambiente real, obtendo-se o resultado discriminado na Tabela 5.2.

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Postura (x, y) [mm]	92.8633 ± 33.2107	31.5318	208.1224
Orientação (θ) [°]	0.5199 ± 0.1775	0.1735	0.8812

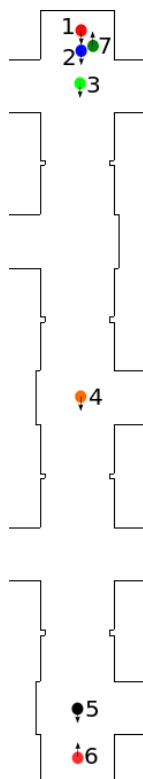
Tab. 5.2: Resultados dos experimentos da Localização pelo método de Markov

Medição	IC de 95% do erro	Melhor Resultado	Pior Resultado
Tempo de execução [s]	0.0109 ± 0.0009	0.0097	0.0154
Consumo de CPU [%]	13.33 ± 1.97	10	17
Quantidade de Bytes alocados [KB]	107.117 ± 0.381	106.527	107.635
Linhas de código	162	-	-

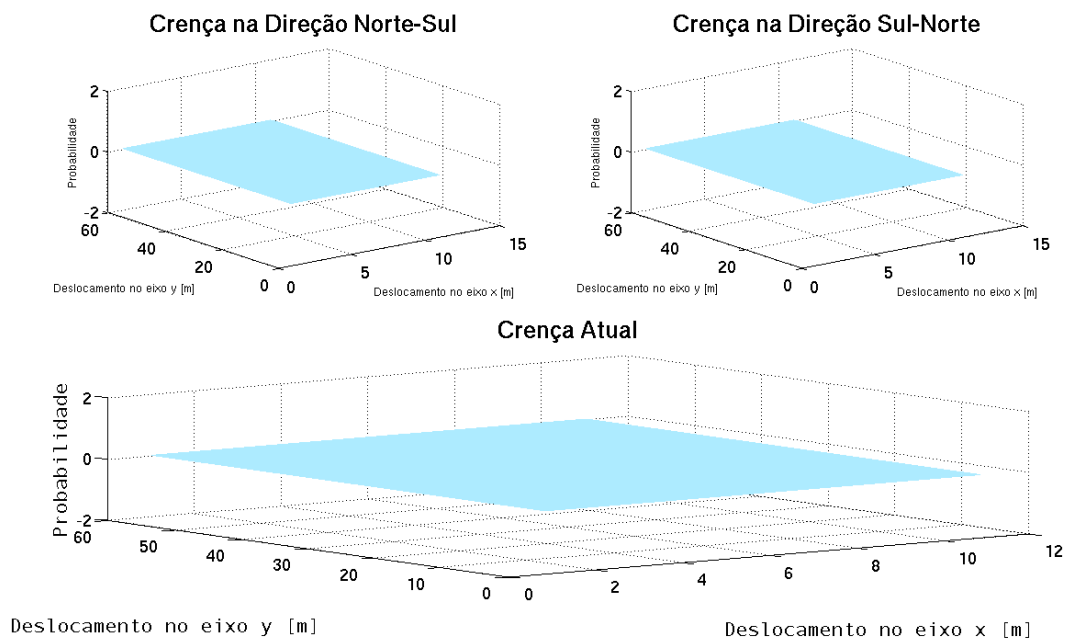
Tab. 5.3: Medições feitas com base na localização pelo método de Markov

5.4 Conclusão

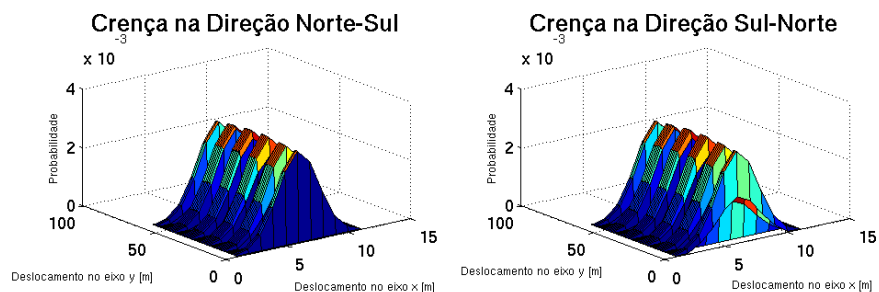
O algoritmo de localização pelo método de Markov se mostrou bastante eficaz, dado o nível de sucesso obtido. Um dos problemas usuais ao se aplicar o método de localização pelo método de Markov (alto esforço computacional) foi eliminado ao se trabalhar com dados topológicos ao invés do mapa métrico, e ao se diminuir consideravelmente o número possível de orientações do robô.



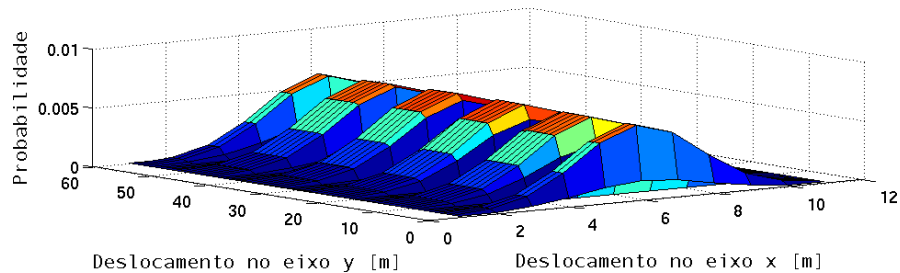
(a) Pontos nos quais a crença foi extraída



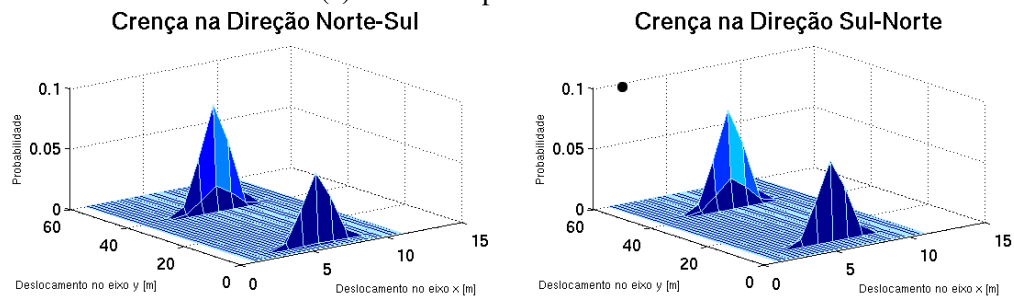
(b) Estado no ponto 1



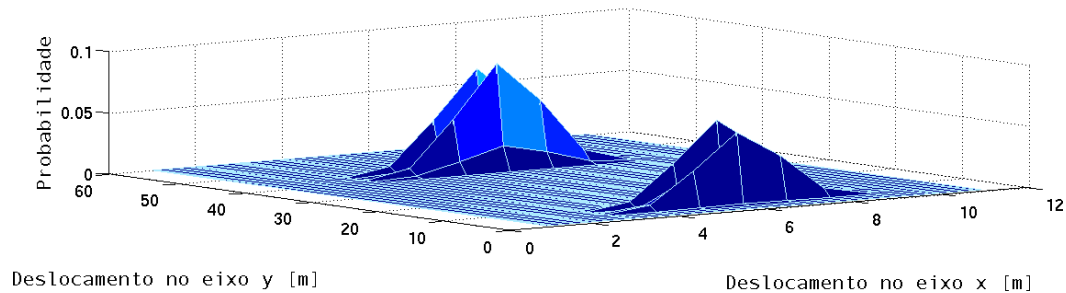
Crença Atual



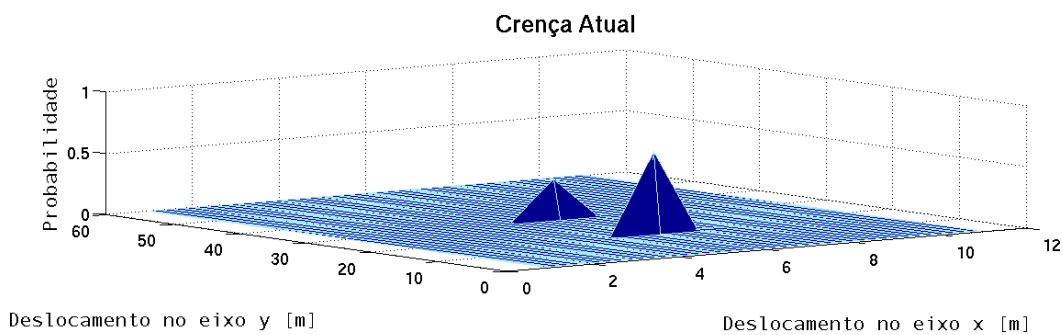
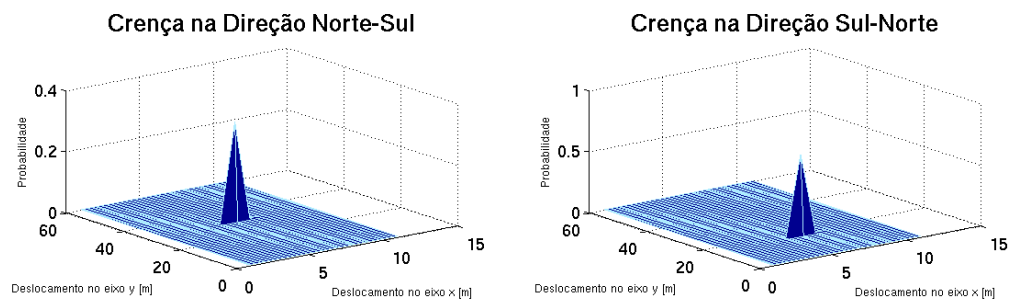
(c) Estado no ponto 2



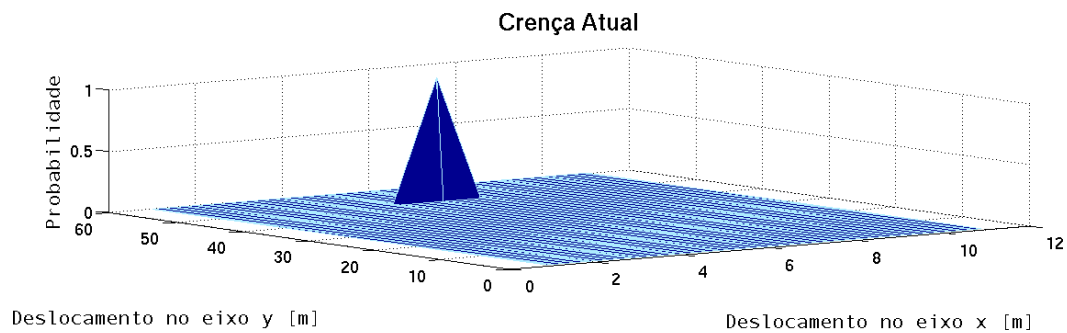
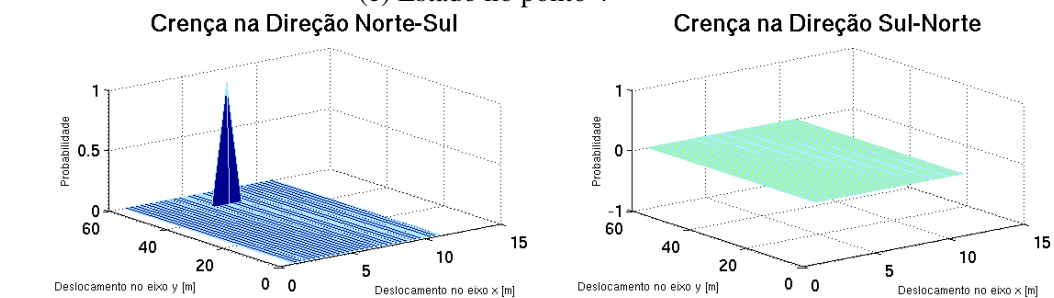
Crença Atual



(d) Estado no ponto 3



(e) Estado no ponto 4



(f) Estado no ponto 5

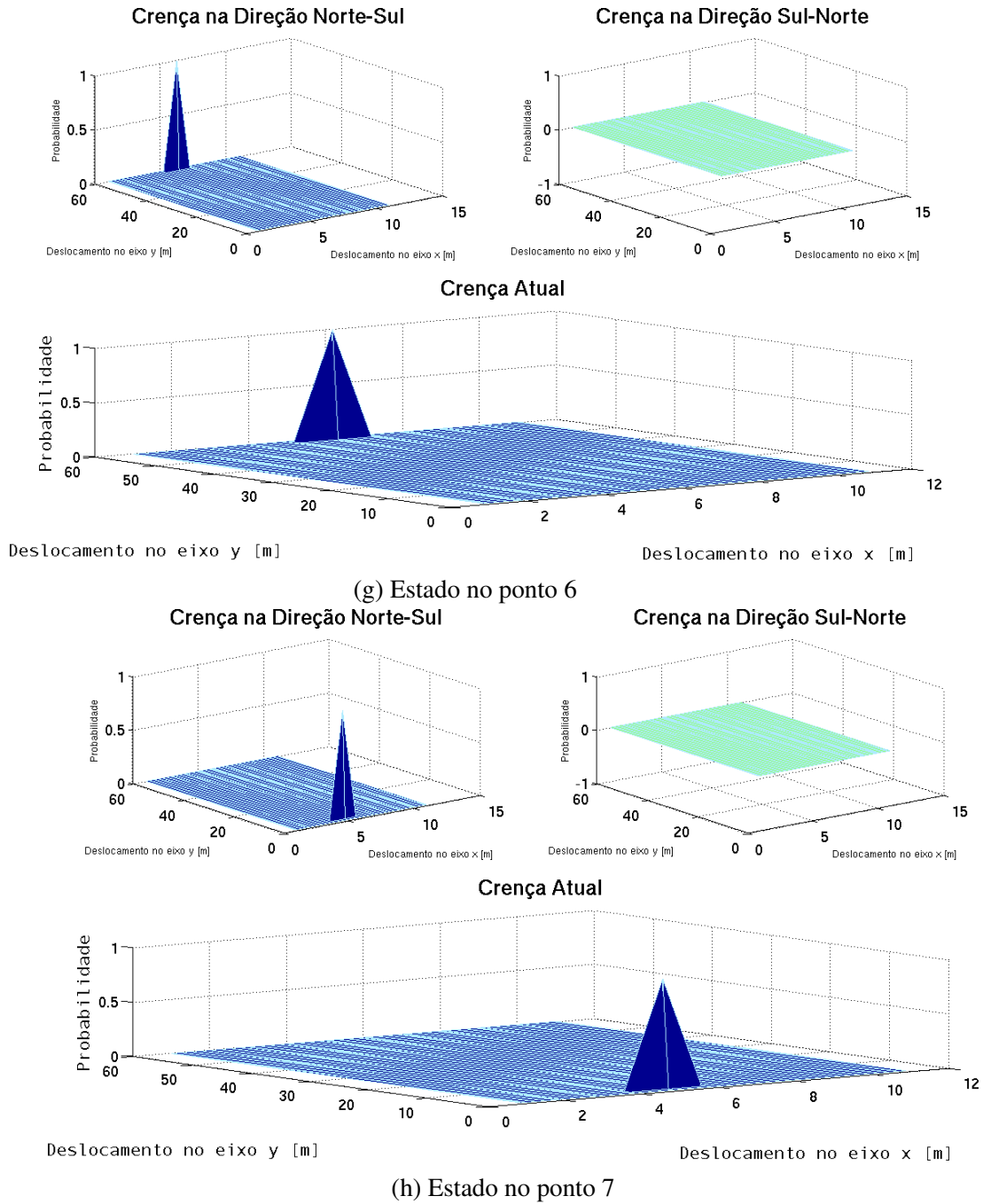


Fig. 5.3: Evolução da crença pelo método de Markov

Capítulo 6

Filtros Gaussianos

6.1 Introdução

Filtros Gaussianos são métodos que modelam a incerteza do sistema por meio de uma função de densidade de probabilidade Gaussiana. Dentro do campo de robótica, os filtros Gaussianos mais utilizados são o filtro de Informação e o filtro de Kalman. O filtro de Kalman propiciou o surgimento de outros filtros, como o filtro de Kalman estendido, *Unscented* [7, 31] e Cubature [32]. Para este estudo, abordou-se os dois tipos mais comuns, os filtros de Kalman dinâmico e estendido.

A função Gaussiana representa a relação entre uma grandeza e o erro associado a ela. Os modelos de erro usados nos métodos dos filtros de Kalman são os apresentados no capítulo 3. A maior vantagem em usar funções gaussianas para modelar o erro de uma grandeza é a facilidade de manipulação matemática, já que uma função gaussiana pode ser gerada a partir de apenas dois parâmetros, a média μ e o desvio padrão σ .

A localização por filtro de Kalman rastreia a postura do robô a partir de uma postura inicial conhecida, e é reputado por ser inerentemente preciso e eficiente. O filtro pode ser usado em representações contínuas, entretanto, se a incerteza se torna muito grande e deixar de ser verdadeiramente unimodal, o filtro pode falhar na captura da multiplicidade de posições e se tornar irremediavelmente perdido [4]. O filtro de Kalman implementa a crença para estados contínuos, não

sendo aplicável em sua forma original a espaço de estados discretos ou híbridos [7].

O filtro de Kalman é uma ferramenta matemática que produz uma estimação ótima baseada no conhecimento do sistema e nas medições feitas. Ele funde as informações provenientes dos sensores com o conhecimento do sistema de maneira ótima. Dentro da teoria do filtro de Kalman, assume-se que o sistema é linear e com ruído Gaussiano unimodal e com média nula [4]. Este filtro evoluiu e se tornou o método estado da arte para estimação de estados. As razões para isso são sua intuitiva formulação de espaço de estados para sistemas lineares, um preditor-corretor de estimação e sua estrutura de filtragem recursiva. Além disso, ainda pode ser implementado em computadores e unidades digitais de processamento de sinal [3, 12].

Muitas das aplicações modernas do filtro de Kalman empregam modificações nas equações originais. Essas modificações são realizadas a fim de adicionar ao filtro outras características, como a possibilidade de tratar modelos não-lineares dos sensores, modelos não-Gaussianos, ou reduzir complexidade computacional [3]. Uma introdução geral do filtro de Kalman pode ser encontrada em Kalman, 1960 [33] e Welch and Bishop, 1995 [34].

6.2 Descrição e Implementação

Característica	Filtros Gaussianos
Localização	Local
Mapa	Métrico
Informação do Ambiente	Linhas

Tab. 6.1: Características dos Métodos de Filtro de Kalman Implementados

O filtro de Kalman considera o robô em movimento, ou seja, o estado da postura do robô varia conforme as medições são feitas. Desta forma, o deslocamento do robô assim como a incerteza que a acompanha devem ser integradas à crença. A Tabela 6.1 contém algumas das características dos filtros Gaussianos. O sistema extrai as linhas que compõem o ambiente a partir das leituras feitas pelos sensores exteroceptivos. Estas leituras são feitas para os ângulos mostrados na Figura 6.1.

Ambos os métodos implementados neste capítulo são construídos seguindo as seguintes fases [4]:

- Fase I: Estimação Inicial da Postura. Aplica-se um modelo Gaussiano de erro de movimentação proporcional à medida de deslocamento captada pelo *encoder* (odometria). O modelo de erro dinâmico (subseção 3.2.3) é utilizado nesta fase para calcular o erro associado à essa estimação.
- Fase II: Observação. O robô coleta as informações atuais dos sensores exteroceptivos e extrai desses dados as características desejáveis (retas). A extração de características é feito pelo algoritmo Split-and-Merge, apresentado na seção 3.3. O modelo de erro estático (vide subseção 3.2.2) é utilizado nesta fase para calcular o erro associado ao mapa criado a partir das observações feitas pelo laser do robô.
- Fase III: Estimação das Medições. Baseado na estimação de postura feita na fase I, o robô estima quais leituras faria com o sensor exteroceptivo.
- Fase IV: Casamento das Características (*Matching*). Nesta etapa, o robô identifica a melhor combinação entre as características extraídas durante a etapa de observação (fase II) e as características estimadas na fase III. Como resultado, esta fase gera o erro entre o que o robô observou e o que estaria observando caso estivesse na postura estimada na fase I.
- Fase V: Estimação Final da Postura. Por fim, o filtro de Kalman funde as informações provenientes da odometria e da extração de características do ambiente para atualizar o estado da crença do robô.

6.2.1 Filtro de Kalman Dinâmico

O algoritmo do filtro de Kalman dinâmico é apresentado no Algoritmo 5, enquanto sua descrição matemática pode ser encontrado em vários textos [4, 7, 10, 12, 34]. O artigo Olfati-saber, 2006 [35], aplica o algoritmo em redes de sensores, o artigo Roumeliotis and Bekey, 2000 [36], apresenta um aplicação que combina o filtro de Kalman com estimação Bayesiana, o artigo Gutmann, 2002 [37],

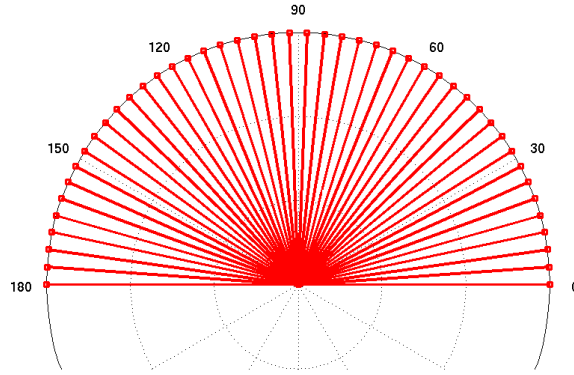


Fig. 6.1: Ângulos de Leitura do Laser para os filtros Gaussianos

traz uma fusão da localização pelo método de Markov e por filtro de Kalman, herdando a precisão do filtro de Kalman e a robustez e velocidade do método de Markov, o artigo Arras e et al., 2001 [38], usa o filtro de Kalman em uma aplicação de fusão entre laser e visão monocular, e em Gutmann and Fox, 2002 [39], é feita uma comparação deste filtro com outros métodos.

Algoritmo 5 Filtro de Kalman Dinâmico $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$ [7]

- 1: $\bar{\mu}_t = A_t \mu_{t-1} + B u_t$
 - 2: $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
 - 3: $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
 - 4: $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
 - 5: $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
 - 6: retorna μ_t, Σ_t
-

6.2.2 Filtro de Kalman Estendido

Transições lineares de estado e medições lineares com ruído Gaussiano integrado são raros de se encontrar na prática. Isso faz com que a forma original do filtro de Kalman não possa ser aplicável na maioria dos problemas de robótica, e foi para resolver um destes problemas que surgiu o filtro de Kalman estendido. Este filtro adiciona ao filtro original a habilidade de trabalhar com a não-linearidade. Isso ocorre devido às funções de probabilidade do estado futuro e das medições passarem

a ser não-lineares. Com funções não-lineares arbitrárias, a crença não pode mais ser descrita por uma função Gaussiana. O que o filtro de Kalman estendido faz então é calcular uma aproximação para a crença, e geralmente representa essa aproximação por uma função Gaussiana. O filtro de Kalman estendido herda do filtro de Kalman dinâmico a representação básica da crença, enquanto diferem no sentido de que a crença do filtro estendido é só aproximado enquanto o do filtro dinâmico é exato [7].

A ideia básica por trás do filtro de Kalman estendido é a linearização. Linearização aproxima uma função não-linear por uma função linear que é tangente a ela na média da função Gaussiana. Uma vez que a função não-linear é linearizada, a mecânica da propagação da crença é equivalente ao do filtro de Kalman dinâmico, podendo-se utilizar as equações do próprio filtro dinâmico [3,7]. Existem várias técnicas de linearização para funções não-lineares. A técnica usada no filtro de Kalman estendido é a expansão de Taylor de primeira ordem [7].

O filtro de Kalman estendido se tornou a ferramenta mais usada para a estimação de estados em robótica. Sua popularidade se deve à sua eficiência computacional. Em contrapartida, uma importante limitação de filtro vem do fato de que ele aproxima a transição do estado e das medições usando expansão de Taylor. A qualidade desta expansão depende basicamente de dois fatores, do grau de não-linearidade e do grau de incerteza da função a ser linearizada [7].

O algoritmo do filtro de Kalman estendido é apresentado no Algoritmo 6, enquanto a descrição matemática deste filtro pode ser encontrada em vários textos ([3, 7, 9, 10, 12]). O livro Choset et al., 2005 [3], apresenta o caso de estudo da aplicação do filtro no problema de localização usando sensores de distância. O artigo Kong et al., 2006 [40], faz uma aplicação deste filtro no problema de localização, o artigo Lee et al., 2008 [41], utiliza o filtro de Kalman estendido e mapa em grid para resolver o problema de localização, o artigo Chen et al., 2009 [42], trabalha o problema de auto-localização com base em visão, e o artigo Schiele and Crowley, 1994 [43], compara este método com outros usando um mapa em grid.

Algoritmo 6 Filtro de Kalman Estendido $(\mu_{t-1}, \sum_{t-1}, u_t, z_t)$ [7]

- 1: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
 - 2: $\bar{\sum}_t = G_t \sum_{t-1} G_t^T + R_t$
 - 3: $K_t = \bar{\sum}_t H_t^T (H_t \bar{\sum}_t H_t^T + Q_t)^{-1}$
 - 4: $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$
 - 5: $\sum_t = (I - K_t H_t) \bar{\sum}_t$
 - 6: retorna μ_t, \sum_t
-

Matrizes de Covariâncias

As matrizes de covariância R_t e Q_t para ambos os filtros são dadas por [4, 7, 10, 12, 34]:

$$R_t = \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T \quad (6.1)$$

$$Q_t = \begin{bmatrix} \sigma_A^2 & \sigma_{AR} \\ \sigma_{AR} & \sigma_R^2 \end{bmatrix} = F_{PQ} C_X F_{PQ}^T \quad (6.2)$$

onde:

$$\Sigma_u(k) = \text{covariância}(\Delta s_d, \Delta s_e) = \begin{bmatrix} k_d |\Delta s_d| & 0 \\ 0 & k_e |\Delta s_e| \end{bmatrix} \quad (6.3)$$

$$\nabla_u f = \begin{bmatrix} \frac{1}{2} \cos(\theta + \frac{\Delta\theta}{2}) - \frac{\Delta s}{2b} \sin(\theta + \frac{\Delta\theta}{2}) & \frac{1}{2} \cos(\theta + \frac{\Delta\theta}{2}) + \frac{\Delta s}{2b} \sin(\theta + \frac{\Delta\theta}{2}) \\ \frac{1}{2} \sin(\theta + \frac{\Delta\theta}{2}) + \frac{\Delta s}{2b} \cos(\theta + \frac{\Delta\theta}{2}) & \frac{1}{2} \sin(\theta + \frac{\Delta\theta}{2}) - \frac{\Delta s}{2b} \cos(\theta + \frac{\Delta\theta}{2}) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \quad (6.4)$$

$$\Delta s = \frac{\Delta s_d + \Delta s_e}{2} \quad (6.5)$$

$$\Delta\theta = \frac{\Delta s_d - \Delta s_e}{b} \quad (6.6)$$

$$C_X = \begin{bmatrix} \text{diag}(\sigma_{\rho_i}^2) & 0 \\ 0 & \text{diag}(\sigma_{\theta_i}^2) \end{bmatrix} \quad (6.7)$$

$$F_{PQ} = \begin{bmatrix} \frac{\partial \alpha}{\partial P_1} & \frac{\partial \alpha}{\partial P_2} & \cdots & \frac{\partial \alpha}{\partial P_n} & \frac{\partial \alpha}{\partial Q_1} & \frac{\partial \alpha}{\partial Q_2} & \cdots & \frac{\partial \alpha}{\partial Q_n} \\ \frac{\partial r}{\partial P_1} & \frac{\partial r}{\partial P_2} & \cdots & \frac{\partial r}{\partial P_n} & \frac{\partial r}{\partial Q_1} & \frac{\partial r}{\partial Q_2} & \cdots & \frac{\partial r}{\partial Q_n} \end{bmatrix} \quad (6.8)$$

O parâmetro b é a distância entre as rodas do robô; $\Delta s_d, \Delta s_e$ são as distâncias percorridas pelas rodas direita e esquerda respectivamente; e k_d, k_e são constantes que representam os erros provenientes dos parâmetros não-determinísticos dos motores das rodas do robô e da interação com o chão. Para o robô usado nos experimentos, os parâmetros foram $b = 32\text{cm}$ e $k_d = k_e = 1$.

Como visto na eq. 6.3, assumimos que os erros das duas rodas são independentes e que a variância dos erros são diretamente proporcionais ao valor absoluto das distâncias percorridas ($\Delta s_d, \Delta s_e$).

Para a matriz C_X , é necessário modelar a variância dos valores de distância ($\sigma_{\rho_i}^2$) e do ângulo em que a leitura foi feita ($\sigma_{\theta_i}^2$). A modelagem dessas variâncias foi realizada conforme descrito no Capítulo 3.

6.2.3 Implementação

A implementação do filtro de Kalman dinâmico se deu seguindo a descrição apresentada em Siegwart and Nourbakhsh, 2004 [4]. Já para o filtro de Kalman estendido, fez-se uso do algoritmo apresentado em Santana *et al.*, 2008 [44]. Por se tratar de algoritmos bastante conhecidos, preferiu-se deixar o equacionamento matemático das cinco fases que descrevem ambos os filtros a cargo destas duas referências e dar espaço às análises dos resultados obtidos.

6.3 Experimento e Análise dos Resultados

Assim como nos demais métodos, os dois filtros de Kalman foram colocados em prática por 10 vezes no mesmo cenário, o corredor do LCA. As tabelas 6.2 e 6.3 contém respectivamente o erro oriundo do filtro de Kalman dinâmico e estendido. Como este filtro tem como característica a localização local, é apresentado também o erro da odometria ao longo do experimento por questão de comparação, na Tabela 6.4.

Dentre os dois filtros gaussianos propostos, o filtro de Kalman dinâmico obteve melhor êxito. A razão para o filtro estendido não ter melhor resultado pode estar por trás do trajeto executado pelo robô. A linearização feita no filtro estendido fez com que sua primeira fase (Estimação da Postura) tenha tido um desempenho pior que o filtro de Kalman dinâmico. Como a trajetória é retilínea, a linearização por Taylor não é a mais adequada. Uma outra evidência para esta conclusão é a de que, em dois experimentos simulados anteriores, um com trajetória apenas circular e outro com o robô no estado de *wander*, o filtro estendido teve performance comprovadamente superior ao do dinâmico.

Apesar de ter sido o filtro gaussiano com melhor atuação, o filtro de Kalman dinâmico apresentou um resultado inferior aos demais métodos, embora a expectativa, principalmente pelo fato de empregar a localização local, apontasse para o contrário. A isto, deve-se o fato da fase de Observação trabalhar com linhas produzidas a partir das leituras, as quais são em seguida representadas em coordenadas polares. Assim, o erro das leituras dos pontos que geram essas linhas acarretam um erro mais elevado, e então, na fase IV, Casamento das Características (Matching), o erro resultante é maior. Provavelmente, um mapa mais próximo ao estado real do corredor, principalmente abordando as salas adjacentes (e seu conteúdo) ao corretor, renderiam um resultado melhor.

Apesar do erro médio da odometria ter superado os 300mm, é importante enfatizar que este erro só foi obtido graças à calibração feita dias antes da realização dos experimentos. Como a calibração demanda tempo e um trabalho minucioso sobre os resultados obtidos das alterações dos parâmetros dos robôs, é de se esperar que a maioria dos robôs não estejam calibrados e que os erros de odometria apresentados por estes sejam consideravelmente maiores.

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Posição (x, y) [mm]	132.7236 ± 52.2285	10.0700	303.8380
Orientação (θ) [°]	1.5498 ± 0.7493	0.0083	3.5871

Tab. 6.2: Resultados dos experimentos de Kalman dinâmico

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Posição (x, y) [mm]	157.6944 ± 65.5876	34.7776	288.0896
Orientação (θ) [°]	1.8047 ± 0.8045	0.4511	4.5515

Tab. 6.3: Resultados dos experimentos de Kalman estendido

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Posição (x, y) [mm]	308.5692 ± 83.7584	121.3641	531.4973
Orientação (θ) [°]	2.2205 ± 0.7432	0.6175	4.5583

Tab. 6.4: Resultados dos experimentos da Odometria

Medição	IC de 95% do erro	Melhor Resultado	Pior Resultado
Tempo de execução [s]	0.0154 ± 0.0001	0.0072	0.0435
Consumo de CPU [%]	8.18 ± 0.35	7	9
Quantidade de Bytes alocados [KB]	168.792 ± 0.1958	168.478	169.130
Linhas de código	441	-	-

Tab. 6.5: Medições feitas com base no filtro de Kalman dinâmico

Medição	IC de 95% do erro	Melhor Resultado	Pior Resultado
Tempo de execução [s]	0.0145 ± 0.0001	0.0064	0.0331
Consumo de CPU [%]	7.86 ± 0.37	6	9
Quantidade de Bytes alocados [KB]	167.007 ± 0.302	166.610	167.866
Linhas de código	464	-	-

Tab. 6.6: Medições feitas com base no filtro de Kalman estendido

6.4 Conclusão

Este capítulo apresentou dois métodos bastantes utilizados para a localização de robôs móveis, o filtro de Kalman dinâmico e estendido. Embora tenham rendido um erro maior que os demais filtros, os resultados obtidos destes filtros foram satisfatórios, especialmente em comparação com o erro obtido pela odometria.

Capítulo 7

Filtro de Partículas - Monte Carlo

7.1 Introdução

Quando não há linearidade nem no modelo especificado nem no processo observado, outros métodos de localização são necessários [45]. Os filtros não-paramétricos surgem como alternativa para filtros gaussianos. Estes filtros não se baseiam em uma forma de função fixa do posterior, como os filtros gaussianos. Ao invés disso, aproximam posteriores (a crença após a incorporação das observações feitas pelo robô) por um número finito de valores, cada um correspondendo aproximadamente a uma região no espaço de estados. Alguns filtros não-paramétricos Bayesianos se baseiam na decomposição do estado espacial, em que cada valor corresponde à probabilidade acumulativa da densidade posterior em uma sub-região compacta do espaço de estados. Outros aproximam o espaço de estados através de amostras aleatórias retiradas da distribuição posterior. Em todos esses casos, o número de parâmetros utilizados para aproximar o posterior pode ser variada. A qualidade da aproximação depende do número de parâmetros usados para representar o posterior. Quando o número de parâmetros tende para o infinito, as técnicas não-paramétricos tendem a convergir uniformemente para o posterior correto [7].

O filtro de partículas utiliza o método de Monte Carlo para estimar poses posteriores do robô e para representar as crenças, e pode ser utilizado tanto para localização global como local. O

algoritmo do método de Monte Carlo se tornou um dos mais populares para localização devido à sua fácil implementação e sua eficiência em uma grande gama de problemas de localização. O filtro de Partículas é adequado para representar crenças multi-modais, permitindo ao robô suportar fases de incertezas globais e problemas de associação de dados, que geram hipóteses distintas. Entretanto, a forma como essas técnicas representam a crença requer uma maior complexidade computacional comparada às outras técnicas apresentadas nesta dissertação [7].

Esta técnica representa o posterior por um grande número de amostras que evoluem e se adaptam recursivamente enquanto novas informações passam a ficar disponíveis. O ponto chave do filtro de partículas é a representação do posterior $bel(x_t)$ por um conjunto randômico de amostras do estado do posterior, chamadas de partículas. Partículas, no caso desta dissertação, são definidas como unidades que representam robôs adimensionais. Desta forma, cada partícula é estruturada como se fosse um robô, com postura (x, y, θ) definida.

A representação do posterior $bel(x_t)$ é aproximada mas é não-paramétrica, e portanto pode representar um espaço muito mais amplo de distribuição que, por exemplo, funções gaussianas. Além disso, outra característica do filtro é que as densidades de probabilidade multi-modais do estado podem ser representados de uma maneira mais favorável, de modo que a robustez da estimação do processo seja menos tendenciosa a falhas [7, 12, 45].

Estes algoritmos apresentam algumas diferenças com relação às técnicas que usam funções Gaussianas [7]:

- Podem processar medidas brutas dos sensores, ou seja, não precisam extrair nenhuma característica dos valores obtidos dos sensores.
- São não-paramétricos e não possuem distribuição unimodal.
- Podem resolver localização global e em alguns casos, problemas de sequestro do robô.

Apropriadamente implementados, essas técnicas são capazes de localizar o robô globalmente e se recuperar frente a falhas de localização (como no caso de sequestro do robô). Como resultado de sua

robustez, estes algoritmos são escolhidos principalmente quando se necessita assegurar a realização das operações pelo robô [7]. O método de Monte Carlo pode aproximar praticamente qualquer distribuição de importância prática. Não está presa a subconjuntos paramétricos de distribuições, como o caso do Filtro de Kalman Estendido. Este algoritmo pode representar distribuições probabilísticas multi-modais complexas e misturá-las de forma integrada no estilo de distribuições gaussianas. Isto provê ao método a habilidade de resolver localizações globais com alta eficiência. Esta é uma das maiores vantagens da técnica [7, 46].

A eficiência da aproximação é facilmente determinada pelo tamanho do conjunto de partículas M . Aumentando-se o número total de partículas, aumenta-se a eficiência da aproximação, mas em contrapartida, aumenta-se também a demanda de poder computacional. Desta forma, a implementação pode se adaptar aos recursos computacionais disponíveis. Quanto melhor forem os recursos, mais eficiente será o algoritmo. Esta adaptação é difícil de ser obtida usando localização em grade ou técnicas que usam distribuições gaussianas [7].

A maior desvantagem do método é a incapacidade de se recuperar de falhas globais de localização e do problema de sequestro do robô. Isto é evidenciado pelo método em si, já que as partículas dos lugares com baixa crença desaparecem. Para contornar esta deficiência, pode-se adicionar um conjunto de partículas randômicas, que podem apontar locais diferentes e fazer o algoritmo se recuperar nos casos citados. A tabela 8.3 do livro Thrun et al., 2005 [7], apresenta o algoritmo para realizar essa otimização.

Este filtro vem sendo usado em algumas aplicações para estimação de sistemas dinâmicos não-lineares e sistemas com funções não-gaussianas [12]. Os artigos Carpenter et al., 2004 [45], e Arulampalam et al., 2002 [47], detalham o método para sua aplicação, enquanto os artigos Cen and Matcasuhira, 2008 [48], Yin et al., 2010 [49] e Carpenter et al., 2004 [45], apresentam uma aplicação para o filtro de partículas voltado para localização robótica. O artigo Hsu et al., 2010 [50], apresenta um filtro de partícula amplificado, aplicado à localização em futebol de robôs. O livro Bar-Shalom et al., 2001 [10], apresenta uma forma de estimar a eficiência do método de Monte Carlo (página 79). Os artigos Fox et al., 1999 [46], Dellaert et al., 1999 [51] e Thrun et al., 2001 [52], abordam

as características do tema e, como o artigo Gutmann et al., 1998 [30], o comparam a várias outras formas de localização. O artigo Zhang et al., 2009 [53], traz uma aplicação para o método onde as amostras se auto adaptam, o artigo Seifzadeh et al., 2009 [54], mostra uma forma eficiente e a baixo custo de implementação.

7.2 Descrição e Implementação

Neste capítulo, descrevemos o experimento envolvendo o método Monte Carlo para localizar o robô em um mapa métrico global. Algumas das características da técnica estão expostas na Tabela 7.1. Como informação do ambiente, este método utiliza as leituras feitas pelo laser do robô. A cada iteração, estima-se quais leituras cada uma das partículas espalhadas pelo mapa fariam caso tivessem seus próprios lasers, coletando valores de distância. Posteriormente, estes valores são utilizados para calcular o nível de discrepância entre a observação feita pelo robô e as observações estimadas das partículas. As leituras feitas tanto pelo laser do robô quanto pelas partículas seguem os ângulos mostrados na Figura 7.1.

Característica	Método de Monte Carlo
Localização	Global
Mapa	Métrico
Informação do Ambiente	Leitura do laser

Tab. 7.1: Características do Método de Monte Carlo Implementado

O algoritmo do método de Monte Carlo segue as seguintes fases:

- Fase I: Estimação do novo estado de cada partícula. A partir do estado anterior das partículas, do deslocamento captado pelo *encoder* (odometria) do robô entre os instantes t_k e t_{k-1} e pelo modelo de erro dinâmico (subseção 3.2.3), uma nova estimativa de estado é feita.
- Fase II: Estimação de observações. Com base nas novas posturas de cada partícula (calculadas na fase I), é então estimado o que cada uma destas partículas observaria caso estivessem munidas de laser, e os valores de leitura que seriam retornados são guardados.

Ângulos de Leitura do Laser para Monte Carlo

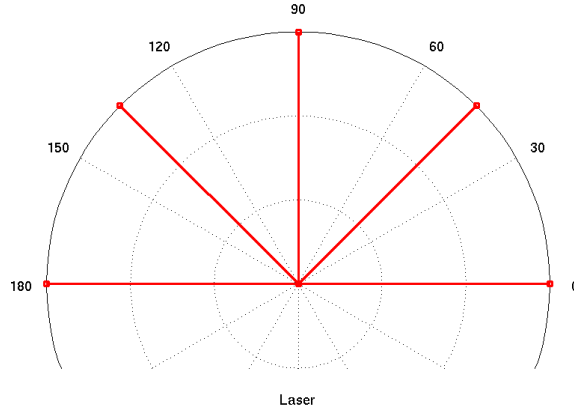


Fig. 7.1: Ângulos de Leitura do Laser para o método de Monte Carlo

- Fase III: Erro de observação e peso. A partir dos valores guardados na fase II para todas as partículas e das leituras feitas pelo robô, o erro de observação é calculado. Este erro serve de índice para quão distante está a postura de cada partícula em relação à postura do robô. Os erros são normalizados e a cada partícula é dado um peso correspondente ao erro produzido por suas leituras. Este peso é relativo ao erro de todas as partículas, assim, a partícula que gera o menor erro tem o maior peso, e a soma de todos os pesos retorna 1. Esta fase utiliza o modelo de erro estático (vide subseção 3.2.2) para o cálculo do erro.
- Fase IV: Reamostragem. As partículas de maior peso são usadas como partículas de referência e servem de ponto de partida para a reamostragem das demais partículas.

O código implementado para o filtro de partículas encontra-se no Algoritmo 7. A crença $bel(x_t)$ é representada por um conjunto de M partículas como $X_t = \{x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]}\}$. Tendo em mente que este é um algoritmo clássico, deixaremos a descrição de seu funcionamento ao texto [7] e entraremos apenas nos detalhes específicos da implementação feita para este experimento.

A linha 8 do Algoritmo 7 trata da amostragem das partículas. Esta etapa ocorre respeitando a probabilidade que cada partícula possui de estar na postura real do robô. Existem vários métodos para realizar esta etapa, dos quais alguns são abordados no artigo Doucet, 1998 [55]. Um destes

métodos serviu de base para o algoritmo utilizado no experimento, cujo código é apresentado no Algoritmo 8. Nesta fase, seleciona-se partículas com probabilidade maior (partículas de referência), e a partir das posições destas, distribui-se as demais partículas (partículas submissas) de modo que a nuvem de partículas tenda à posições que geram erros menores. O número de partículas de referência selecionadas depende do erro absoluto que elas apresentam. Apenas as partículas com erro menor que um limite estabelecido pelo desenvolvedor são consideradas partículas de referência.

Apesar da probabilidade p ser usada para identificar as partículas com menor erro, sua normalização faz com que se torne um parâmetro relativo. Uma das modificações feitas para o algoritmo de reamostragem foi a utilização da norma da diferença entre as leituras das partículas e do laser (Alg. 8: linha 7). Com a normalização, este valor D carrega o erro absoluto da partícula. Isto é muito importante pois evidencia se as partículas estão realmente próximas da postura real ou não, uma vez que nem mesmo a partícula com maior probabilidade é garantia de que sua postura gere um erro absoluto satisfatório.

Aplicando um fator de ajuste (α) em D , o resultado estará dentro do domínio da função Sigmoid da Figura 7.2 (entre 0 e 10). Esta curva é empregada pois discrimina facilmente as partículas de referência satisfatórias das não satisfatórias pelo uso do erro absoluto das partículas, uma vez que delimita as zonas satisfatória, intermediária e não satisfatória, como indicado na figura. Além disso, possibilita uma rápida transição de uma região para a outra, diminuindo a região intermediária entre as zonas. Ao ser multiplicada pelo valor máximo de dispersão (Λ) desejado, as partículas de referência com erros absolutos maiores têm suas partículas submissas espalhadas em um raio de até Λ mm, enquanto as partículas de referência com erro absoluto menor têm suas partículas submissas postas com uma dispersão bem menor. Do modo que esta operação é feita, privilegia as regiões com erro menor enquanto tenta divergir as partículas com referência em posições com erro insatisfatório, dificultando assim que o sistema convirja para pontos falso-positivos ou mínimo-locais. Os valores dos parâmetros empregados são fornecidos pela Tabela 7.2.

Parâmetro	Valor
Número de Partículas	1600
Quantidade de raios do laser	5 (−90 : 90 : 45)
α	10^{-3}
Ω	2000 mm

Tab. 7.2: Parâmetros do método de Monte Carlo

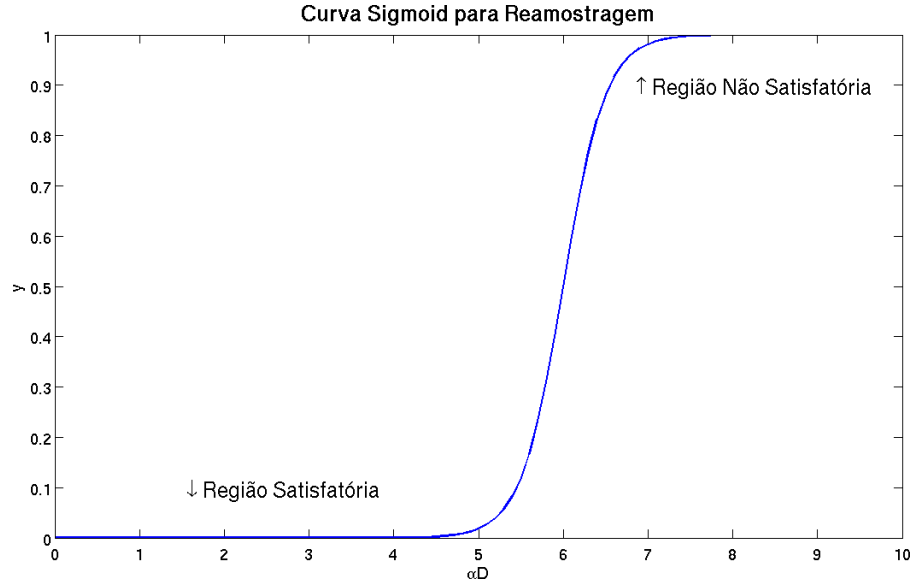


Fig. 7.2: Curva Sigmoid que define a concentração ou dispersão das partículas

Algoritmo 7 Monte Carlo (X_{t-1}, u_t, z_t, m) [7]

```

1:  $X_t = X_t = \emptyset$ 
2: for  $m = 1$  para  $M$  do
3:    $x_t^{[m]} = \text{modelo de movimentação}(u_t, x_{t-1}^{[m]})$ 
4:    $\omega_t^{[m]} = \text{modelo de percepção}(z_t, x_t^{[m]}, m)$ 
5:    $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, \omega_t^{[m]} \rangle$ 
6: end for
7: for  $m = 1$  para  $M$  do
8:   amostrar  $m_i$  com probabilidade  $\propto \omega_t^{[i]}$ 
9:   adicionar  $x_t^{[i]}$  a  $X_t$ 
10: end for
11: retorna  $X_t$ 

```

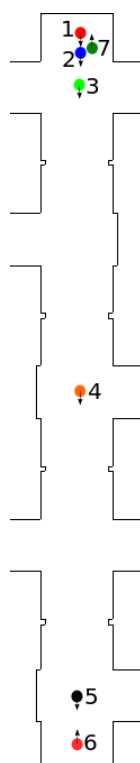
7.3 Experimento e Análise dos Resultados

O experimento de localização aplicando método de Monte Carlo foi realizado 10 vezes no cenário proposto. A Figura 7.3 contém a evolução do estado da crença do robô ao longo do experimento. A

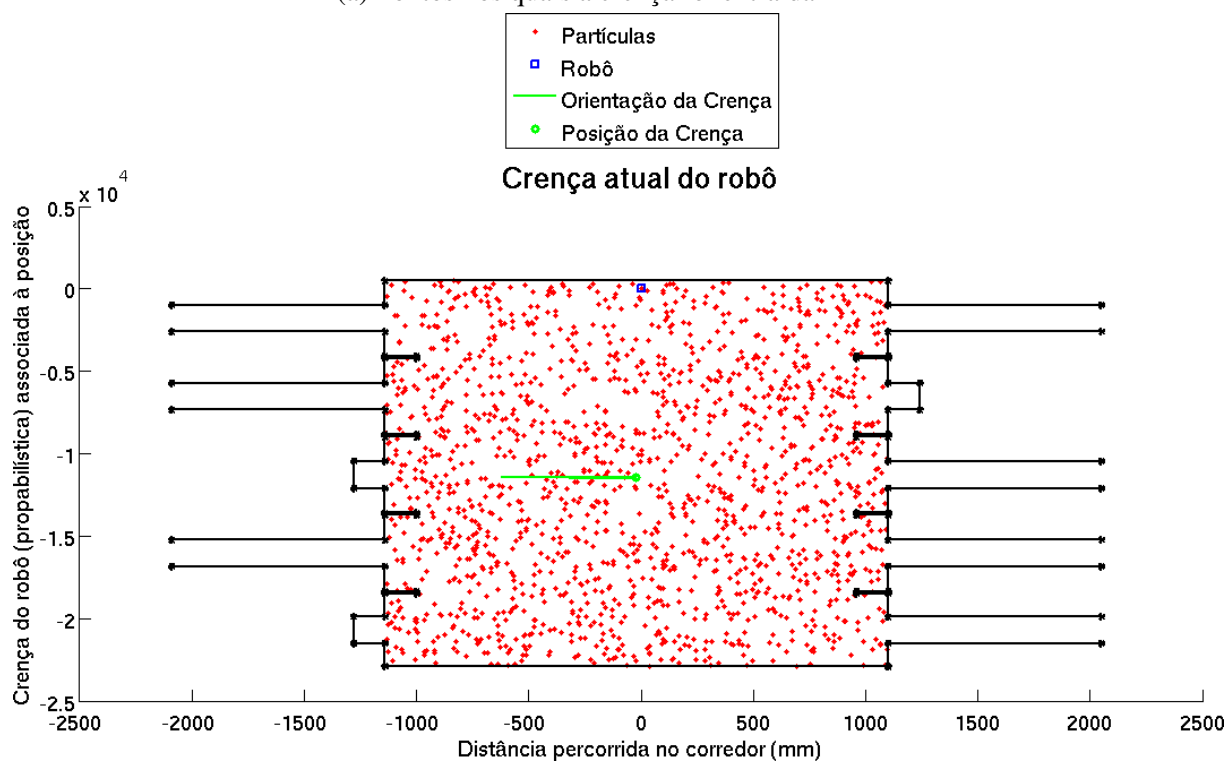
Algoritmo 8 Reamostragem

```
1: for all  $p_j > T$  do
2:    $N = \lfloor N \text{ floor}(p_{\%,j}) M \rfloor$ 
3:    $index = \lfloor index \ j \rfloor$ 
4: end for
5:  $N_{max(N)} = N_{max(N)} + (M - \sum N)$ 
6: for  $K = 1$  para tamanho ( $index$ ) do
7:    $D = \sqrt{(\ell_{1,real} - \ell_{1,m_{index(K)}}) + \dots + (\ell_{n,real} - \ell_{n,m_{index(K)}})}$ 
8:    $\Lambda = \Omega \text{ sigmf}(\alpha D)$ 
9:   for  $\sum_1^K(N)$  para  $\sum_1^K(N) + N_K$  do
10:     $m'_K = m_K + \Lambda [randn(x, y, \theta)]$ 
11:   end for
12: end for
13:  $m = m'$ 
```

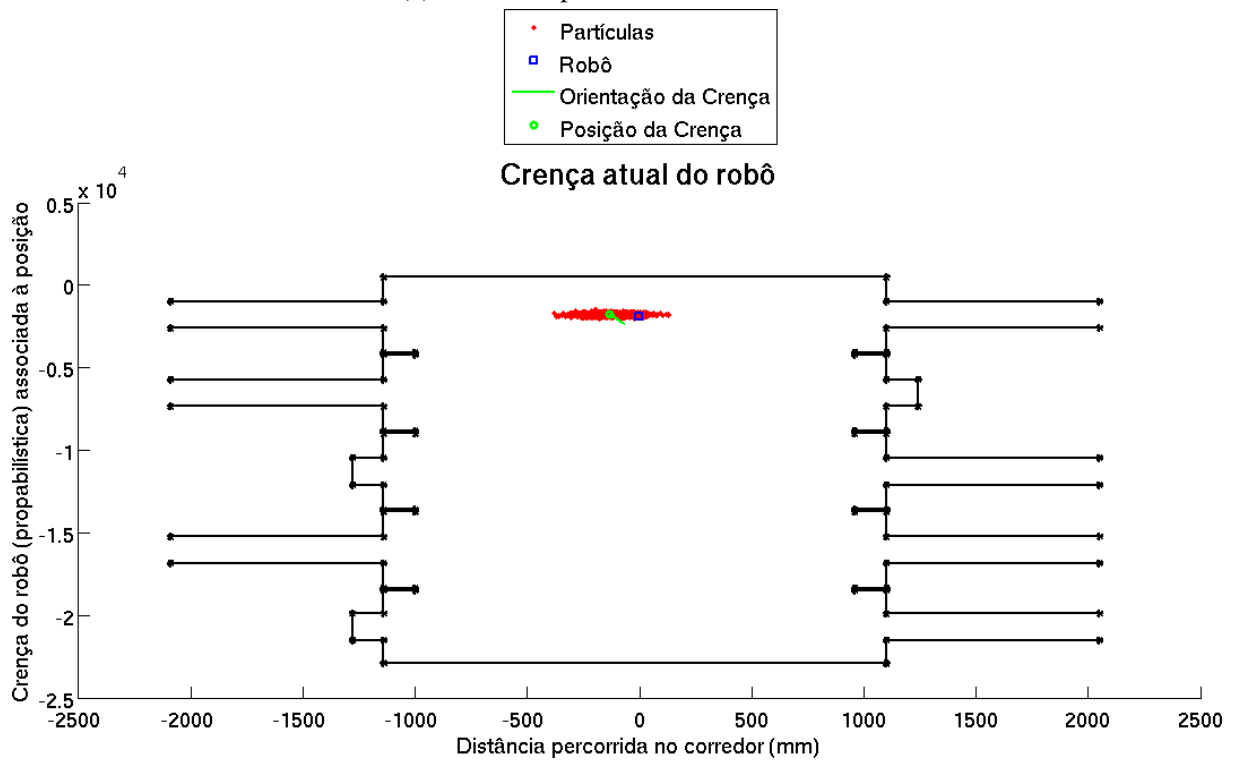
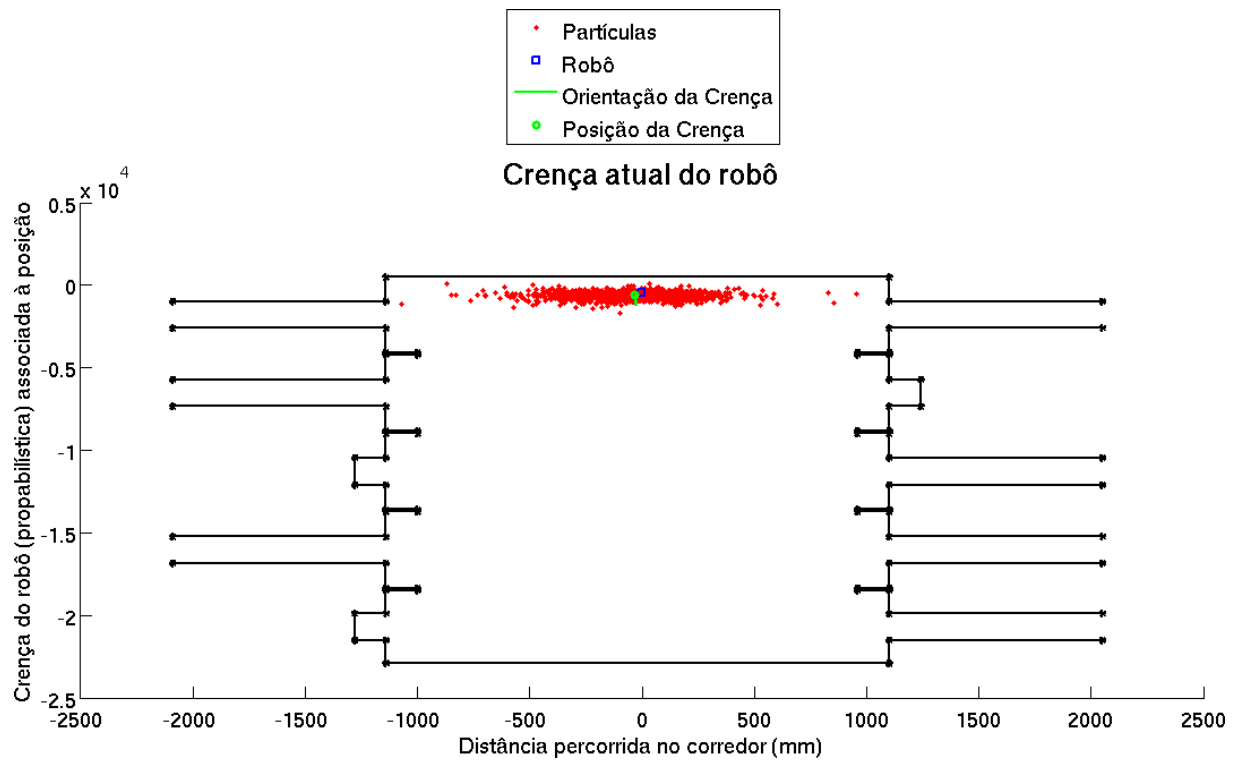
Figura 7.3a apresenta os pontos nos quais os estados da crença foram extraídos. Na Figura 7.3b é observado que a postura (posição e orientação) das partículas é determinada de forma randômica por todo o mapa. As demais imagens ilustram como as partículas se agrupam em uma nuvem desde as primeiras iterações.

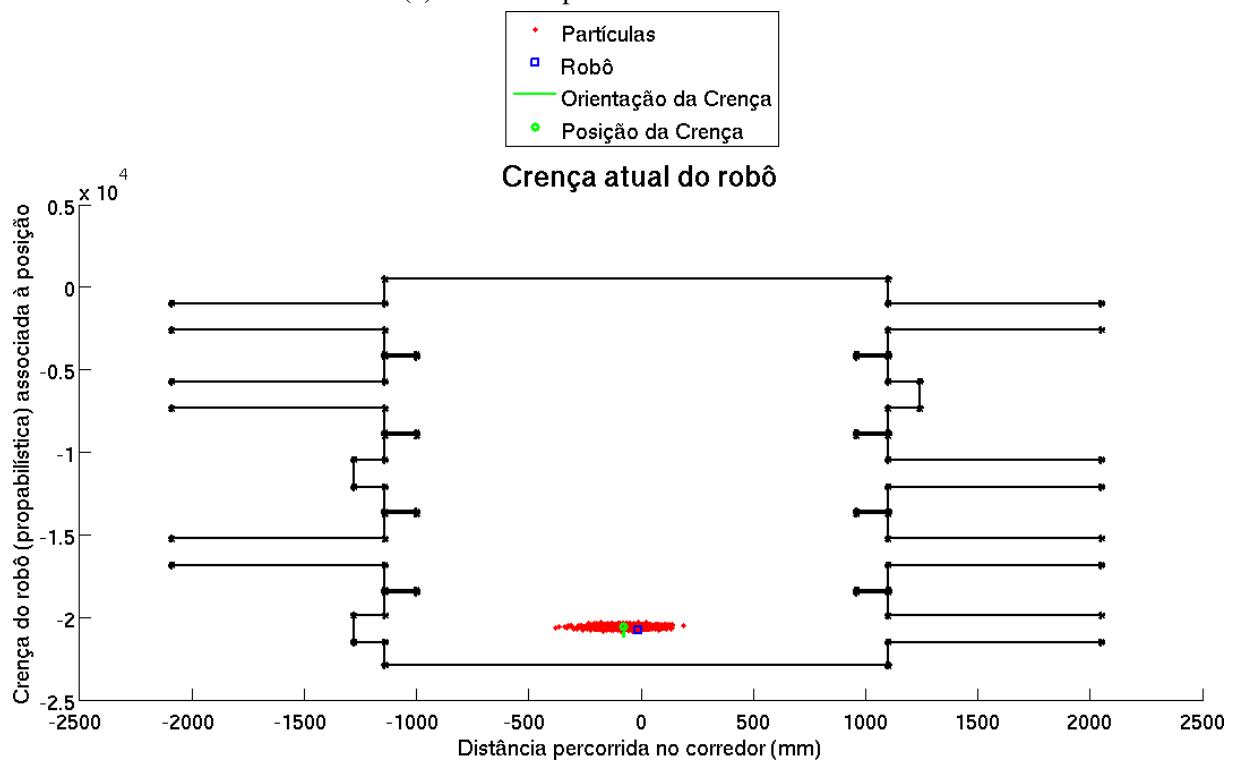
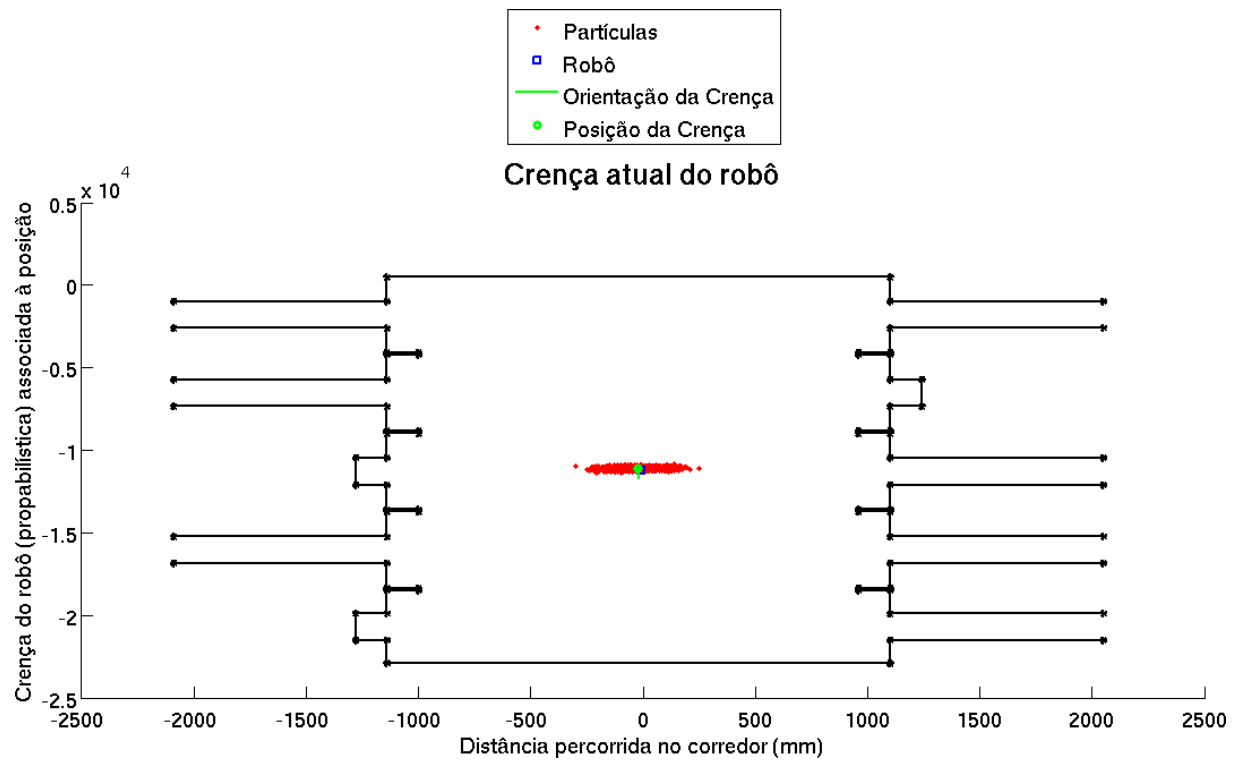


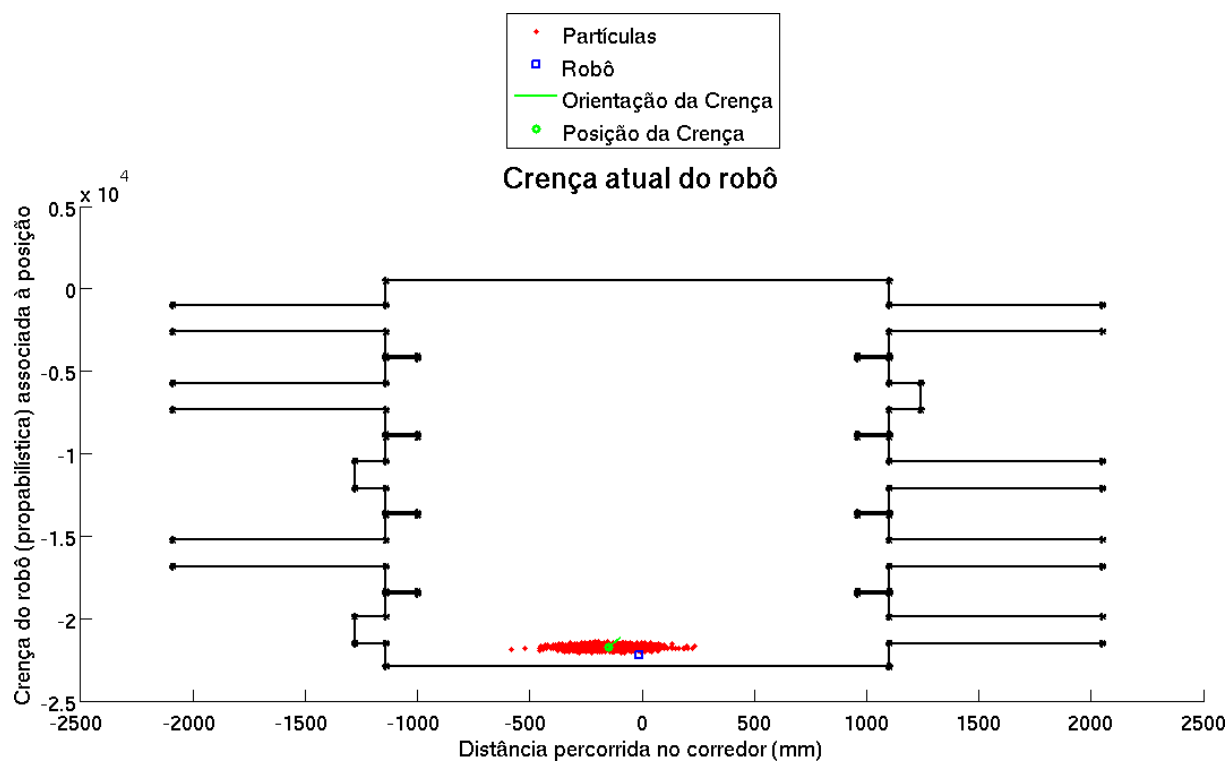
(a) Pontos nos quais a crença foi extraída



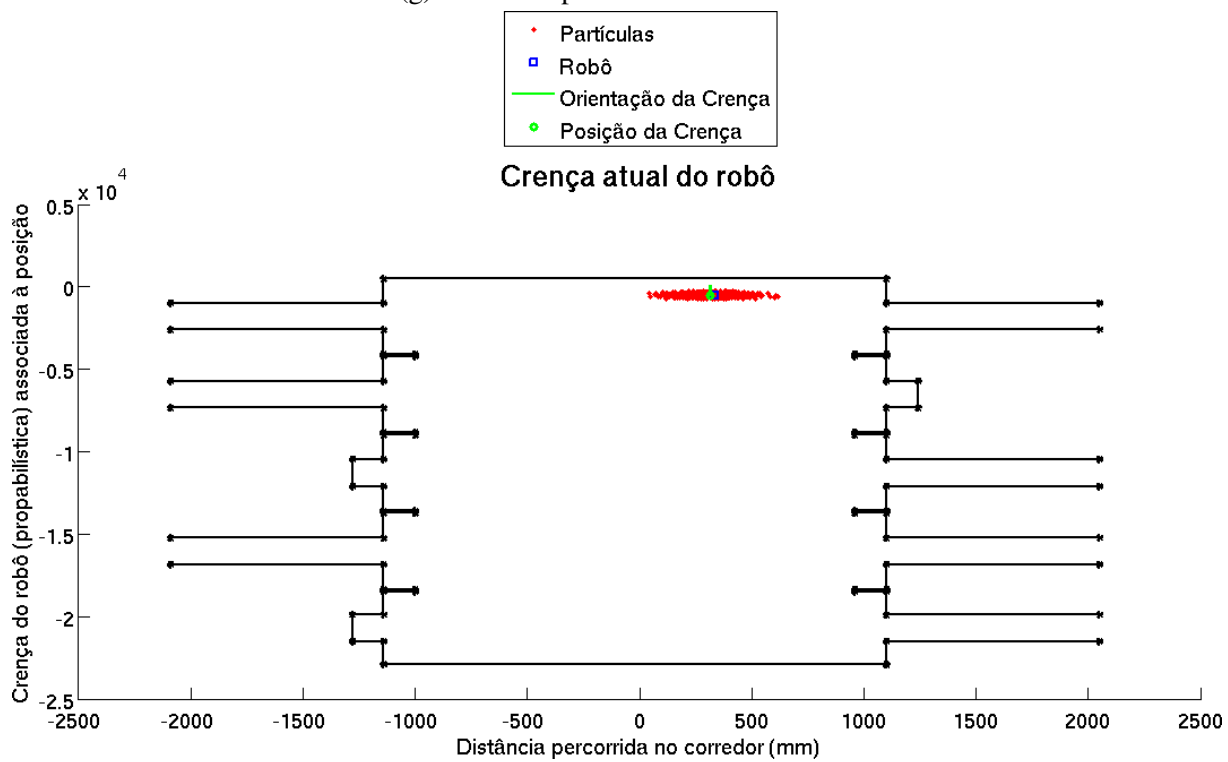
(b) Estado na postura 1







(g) Estado na postura 6



(h) Estado na postura 7

Fig. 7.3: Evolução da crença por método de Monte Carlo

Embora a Figura 7.3 apresente uma estimação realizada com sucesso desde o início, em alguns casos o algoritmo demora mais algumas iterações para encontrar a postura do robô, fazendo com que as partículas se acumulem em pontos de mínimos locais ou falsos positivos. Isto ocorre principalmente devido ao ruído não modelado do ambiente e pela simetria do corredor. Nem todas as características do ambiente foram modeladas no mapa, principalmente no que se refere aos obstáculos presentes dentro das salas (observadas na Figura 1.2). Este ruído é diminuído ao se reduzir o alcance das leituras feitas pelo *laser*. Quanto à simetria do corredor, pode-se notar que além de haver topologias iguais no corredor, um robô que sai da origem (ponto Norte) observa a mesma sequência de topologias de um robô que sai da 2ª topologia em forma de cruzamento (\parallel) na direção Sul-Norte. Isto acarreta ainda mais problemas de falsos-positivos e justifica ainda mais o uso de um sistema mais robusto de reamostragem.

O resultado do experimento é apresentado na Tabela 7.3, e apresenta valores bastante satisfatórios. O erro médio de menos de 2 cm prova a robustez do algoritmo e o desvio-padrão de menos de 1 cm, sua precisão. O fato de utilizar várias partículas faz com que o sistema averigue diversas hipóteses e aja sobre algumas delas. Assim, a probabilidade de sucesso aumenta. Os parâmetros usados no algoritmo estão contidos na Tabela 7.2. A quantidade de raios do laser se refere tanto ao robô quanto às partículas, isto porque a partir da postura (x, y, θ) das partículas e do mapa do ambiente, o método estima quais leituras cada partícula faria do ambiente. A quantidade de raios influencia tanto na qualidade da estimação (diminui o erro) quanto do tempo necessário para execução do algoritmo, e foi escolhida de forma a otimizar esta relação.

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Posição (x, y) [mm]	18.9354 ± 9.8588	3.4921	53.3109
Orientação (θ) [°]	1.1226 ± 0.5849	0.0249	2.9373

Tab. 7.3: Resultados dos experimentos do método de Monte Carlo

Erro	IC de 95% do erro	Melhor Resultado	Pior Resultado
Tempo de execução [s]	0.6725 ± 0.0026	0.6520	0.7142
Consumo de CPU [%]	$50.41 \pm$	38	63
Quantidade de Bytes alocados [KB]	130.213 ± 0.109	129.933	130.381
Linhas de código	183	-	-

Tab. 7.4: Medições feitas com base no método de Monte Carlo

7.4 Conclusão

O método de Monte Carlo é bastante interessante e mostrou-se uma ferramenta fortíssima para a localização global. Os resultados obtidos foram bastante razoáveis, além da contribuição feita com respeito à robustez da reamostragem, dada pelo algoritmo 8, para solucionar eventuais problemas de falso-positivos e mínimos-locais.

Capítulo 8

Comparação entre Métodos

Investir no estudo dos métodos de localização resulta em um apanhado de informações sobre como resolver o problema, entretanto, analisar comparativamente estas técnicas pode abrillantar ainda mais o estudo na medida em que se levantam questões dificilmente encontradas na literatura da área ou que necessitem da experiência de implementação para respondê-las. Neste capítulo, serão abordados tanto aspectos qualitativos quanto quantitativos sobre os métodos descritos na Tab. 8.1, de forma a facilitar o entendimento como um todo sobre as técnicas.

Característica	Filtro de Bayes	Markov	Gaussianos	Monte Carlo
Localização	Global	Global	Local	Global
Ambiente	Estático	Estático	Estático	Estático
Abordagem	Passiva	Passiva	Passiva	Passiva
Quantidade de Robôs	Um	Um	Um	Um
Mapa	Métrico	Grade	Métrico	Métrico
Informação do Ambiente	Portas abertas	Leituras brutas	Linhas	Leitura do laser

Tab. 8.1: Características dos Métodos de Localização Implementados

8.1 Comparação Quantitativa

A análise quantitativa nesta dissertação utiliza dados medidos durante a execução dos algoritmos e do desempenho de localização robótica obtido através dos mesmos. Após a análise individual, a

comparação dos métodos pode ser feita e conclusões podem ser geradas.

8.1.1 Eficiência

A Tabela 8.2 reúne os resultados que evidenciam a eficiência das técnicas sugeridas em ambiente real, podendo-se contrastar com a eficiência da odometria do robô. Dado que cada experimento foi executado com uma variedade de valores para os parâmetros, os valores da tabela foram obtidos pela forma descrita nos capítulos específicos de cada uma das técnicas. Nesta dissertação, entende-se por eficiência o erro de estimação gerado ao final da navegação do robô. Como o estado do robô é composto tanto da posição (x, y) quanto da orientação (θ) , decidiu-se apresentar estes erros separadamente.

Na Tabela 8.2, verifica-se que o melhor resultado foi obtido pelo método de Monte Carlo, seguido do filtro de Bayes, método de Markov, filtro de Kalman dinâmico e estático. O método de Monte Carlo rende erros bastante pequenos devido à sua capacidade de averiguar diversos pontos a cada iteração. Embora a crença do estado seja determinada pela média das partículas, quando as partículas são reamostradas a partir do peso destas o que se faz na verdade é averiguar qual das partículas tem menor erro e espalhar as demais partículas em torno desta. Assim, ao longo do deslocamento do robô, verifica-se milhares de posições e pode-se minimizar o erro deslocando a nuvem de partículas livremente, por exemplo, além do deslocamento apontado pela odometria para aquela iteração. Esta é uma das razões pelas quais o método de Monte Carlo é aplicado em problemas de sequestro do robô, já que a nuvem de partículas pode varrer o cenário livremente. Entretanto, esta liberdade não é compartilhada pelas demais técnicas, mesmo aquelas que também trabalham com a localização global. A localização pelo método de Markov limita a mudança de crença ao utilizar regiões (células) de tamanhos fixos e grandes (comparadas à área ocupada por uma partícula).

O filtro de Bayes também apresenta um bom desempenho, com erro médio de 31.5000, apesar de localizar o robô apenas na direção x . Este método ainda é valioso pois ao se deslocar ao longo do corredor, o robô na maioria das vezes executará uma trajetória linear, justificando assim a localização

Erro de Posição (x, y) [mm]			
Técnica	Erro (IC 95%)	Melhor Resultado	Pior Resultado
Filtro de Bayes	31.4800 ± 14.2872	2.8517	59.7014
Markov	92.8633 ± 33.2107	31.5318	208.1224
Filtro de K. Dinâmico	132.7236 ± 52.2285	10.0700	303.8380
Filtro de K. Estendido	157.6944 ± 65.5876	34.7776	288.0896
Monte Carlo	18.9354 ± 9.8588	3.4921	53.3109
Odometria	308.5692 ± 83.7584	121.3641	531.4973

Tab. 8.2: Tabela contendo os resultados obtidos em ambiente real para o erro de posição.

unidirecional. A localização pelo método de Markov por sua vez tem sua eficiência limitada pela área de suas células, pela quantidade de matrizes que compõem a crença (relacionadas à orientação do robô) e da fidelidade das densidades de probabilidade relacionadas a cada uma das características específicas (*landmarks*) no mapa. Reduzindo-se a área das células é possível diminuir o erro de posição, mas aumenta-se o tempo de execução e a quantidade de memória requerida.

Para os filtros gaussianos (filtro de Kalman dinâmico e estendido), a performance foi comprometida principalmente pela quantidade de ruído proveniente das salas adjacentes ao corredor. Trabalhando com 46 raios do laser, a quantidade de ruído inserido no algoritmo fez com que o erro fosse o maior dentre os métodos. Ainda assim, o erro originado é apenas metade do erro concebido pela odometria. E vale lembrar que o robô sofreu calibração anterior à coleta dos dados afim de tornar o funcionamento da odometria o melhor possível.

Quanto à orientação, novamente, todos os métodos obtiveram um resultado superior ao da odometria. Dessa vez, o método de Markov obteve a melhor resposta. Entretanto, deve-se fazer a distinção dentre as demais técnicas uma vez que o ângulo de estimação para o método de Markov foi limitado a 90° ou 270° , dessa forma, o erro baixo só foi obtido devido à orientação final do robô no decorrer do experimento ter sido em média próximo a 90° . Em outras palavras, caso o robô se desviasse ao longo do caminho chegasse ao destino com orientação de 85° , por exemplo, o método de Markov continuaria a estimar a orientação a 90° , e o erro seria o maior dentre os métodos. O sucesso então deve-se ao conhecimento prévio assumido no experimento de que os únicos ângulos possíveis para um robô navegando paralelamente às paredes seriam 90° e 270° . Para casos mais

Erro de Orientação (θ) [°]			
Técnica	Erro (IC 95%)	Melhor Resultado	Pior Resultado
Filtro de Bayes	-	-	-
Markov	0.5199 ± 0.1775	0.1735	0.8812
Filtro de K. Dinâmico	1.5498 ± 0.7493	0.0083	3.5871
Filtro de K. Estendido	1.8047 ± 0.8045	0.4511	4.5515
Monte Carlo	1.1226 ± 0.5849	0.0249	2.9373
Odometria	2.2205 ± 0.7432	0.6175	4.5583

Tab. 8.3: Tabela contendo os resultados obtidos em ambiente real para o erro de orientação.

genéricos, o sistema pode comportar mais ângulos, ao custo de se adicionar mais matrizes e perder rapidez na execução do algoritmo. As demais técnicas apresentaram erros semelhantes, encabeçados pelo método de Monte Carlo.

De modo geral, a Tabela 8.3 sugere que, para o cenário proposto nesta dissertação, quanto mais simples for o método de extração de características do ambiente e inserção desta informação no sistema, melhor será a eficiência. Enquanto no método de Monte Carlo esta informação é conseguida diretamente pela diferença entre as leituras das partículas e do robô, os filtros gaussianos requerem um processamento com complexidade maior.

8.1.2 Tempo de Execução

As técnicas de localização abordadas aqui são empregadas durante a navegação do robô, implicando que os algoritmos precisam funcionar em tempo real. Os tempos de execução devem ser adequados para a realidade da aplicação. Isto é importante principalmente pelo fato da localização robótica estar geralmente associada a alguma outra função do robô, e para isso, é necessário que o método de localização permita que o robô execute quaisquer demais funções a que se propõe, sem ocupar todo o tempo de processamento. Além disso, os algoritmos descritos neste texto não estão rodando paralelamente às outras funcionalidades, o que pode acarretar consequências desastrosas, já que durante o período de execução dos algoritmos o robô fica suscetível a colisões, quedas, tombamentos, etc.

A Tabela 8.4 traz os tempos necessários para se executar cada um dos algoritmos. O filtro de Bayes além de ser o mais simples, é também o mais rápido, seguido pelo método de Markov, os filtros Gaussianos e por fim, o método de Monte Carlo. Assumindo-se apenas dois possíveis ângulos gerou um resultado bastante razoável para o tempo de execução do método de Markov, já que ao invés de manipular inúmeras grandes matrizes, trabalha-se com um número reduzido de matrizes. Além disso, as dimensões das células têm papel fundamental aqui. Ao se estabelecer estas dimensões, foi levado em conta o custo benefício disso. Diminuindo um pouco que seja a dimensão das células, acarreta um tempo maior em cada manipulação de matrizes feita no algoritmo, implicando em um tempo maior. Caso o desenvolvedor tenha em mente um valor máximo de tempo de execução para o método de localização, pode-se avaliar se dimensões menores ocasionarão um ganho considerável no resultado.

Os filtros Gaussianos passaram por algumas otimizações que o tornaram razoavelmente mais rápidos, o que aproximaram seu tempo ao do método de Markov. Entretanto, o algoritmo que mais demandou otimização foi o método de Monte Carlo. Toda a eficiência comprovada na subseção 8.1.1 tem como contrapartida um consumo de tempo muito maior, dado que um grande número de partículas aumenta bastante o tempo de execução. Como um número menor de partículas pode fazer com que o algoritmo demore a convergir ou até diverja, resta à técnica conviver com esta demanda. Com base na experiência desenvolvida ao se implementar a técnica, a parte do algoritmo que consome mais tempo é a execução do modelo de percepção, na linha 4 do Algoritmo 7. Este modelo calcula a partir da postura das partículas e do mapa do ambiente qual seria a leitura que cada uma faria do ambiente. O tempo para execução desta fase aumenta com o aumento do número de linhas no mapa, com o aumento do número de partículas e com o aumento do número de raios do laser de cada partícula. Interessantemente, estes são os mesmos pilares nos quais a eficiência do método se apoia, portanto, ao invés de diminuí-los, opta-se pelo trabalho de otimização dos códigos. As outras fases do método também demoram um tempo considerável e também foram otimizadas, fazendo-se uso da rapidez com que o MatLab executa operações matriciais. Ainda assim, após toda a otimização e a rápida manipulação de matrizes possível através do MatLab, o tempo total de execução ainda é muito

Tempo de Execução [s]			
Técnica	Tempo (IC 95%)	Melhor Resultado	Pior Resultado
Filtro de Bayes	0.0078 ± 0.0007	0.0058	0.0141
Markov	0.0109 ± 0.0009	0.0097	0.0154
Filtro de K. Dinâmico	0.0154 ± 0.0001	0.0072	0.0435
Filtro de K. Estendido	0.0145 ± 0.0001	0.0064	0.0331
Monte Carlo	0.6725 ± 0.0026	0.6520	0.7142

Tab. 8.4: Tabela contendo os tempos de execução das técnicas.

superior aos demais, por volta de 40 vezes o tempo do filtro de Kalman dinâmico, a técnica com o segundo maior tempo de execução.

8.1.3 Consumo de CPU

O consumo de CPU (Central Processing Unit) é um aspecto importante a ser analisado já que pode ser o fator determinante na escolha da técnica a ser aplicada. É comum encontrar trabalhos de robótica na literatura cujo processamento dos algoritmos ocorre localmente, ou seja, dentro robô. Assim, é possível que haja um limite de processamento determinado pelo próprio hardware do robô. A Tabela 8.5 apresenta os valores em porcentagem do consumo de CPU correspondentes aos métodos empregados nesta dissertação. Os valores da tabela foram obtidos pelo comando *top*, no sistema descrito ao final da subseção 2.7.

Os filtros de Kalman dinâmico e estendido foram os que obtiveram o menor consumo de CPU dentre às técnicas empregadas. Ambos os métodos não demandam muito em termos de processamento devido ao uso de funções Gaussianas. As operações e manipulações realizadas na crença destes algoritmos são feitas sobre os parâmetros (média e covariância) da função Gaussiana, que por serem matrizes pequenas (3×1 e 3×3), tornam o processamento exigido relativamente baixo.

O método de Markov e o filtro de Bayes demandam um processamento um pouco mais elevado do que o apresentado pelos filtros gaussianos. O método de Markov apresenta operações em matrizes maiores que as usadas nos filtros Gaussianos, que é o caso da matriz que guarda a crença no método. Aliás, o valor de discretização desta matriz assim como a quantidade de matrizes para representar os

Consumo de CPU [%]			
Técnica	Consumo (IC 95%)	Menor Resultado	Maior Resultado
Filtro de Bayes	21.66 ± 1.63	19	26
Markov	13.33 ± 1.97	10	17
Filtro de K. Dinâmico	8.18 ± 0.35	7	9
Filtro de K. Estendido	7.86 ± 0.37	6	9
Monte Carlo	50.41 ± 4.72	38	63

Tab. 8.5: Tabela contendo os valores de consumo de CPU das técnicas.

ângulos de orientação permitidos são diretamente proporcionais ao consumo de CPU do método. Por sua vez, o filtro de Bayes exibe o segundo maior consumo da Tabela 8.5. Apesar de ser um método mais simples, a crença utilizada no método é descrita por meio de uma função contínua, fazendo uso de matrizes enormes ($1x22000$). O tamanho da matriz se justifica pelo comprimento do corredor e pela abertura das funções gaussianas determinadas pelas observações de cada uma das portas abertas. Como crítica ao trabalho fica a ideia de que a utilização de uma taxa menor de discretização para a crença do filtro de Bayes não impactaria na eficiência do método e reduziria seu consumo de CPU.

O método de Monte Carlo traz o maior consumo de CPU dentre as técnicas aplicadas. Mais uma vez, atribui-se este valor às matrizes utilizadas no algoritmo, em específico no processamento realizado na fase II, estimação de observações, descrito na subseção 7.2. Nesta fase, vasculha-se a matriz que guarda os parâmetros das retas que compõem o mapa do ambiente afim de encontrar os pontos que cada partícula observaria caso estivessem munidas de laser. Para isso, são feitas multiplicações e operações envolvendo matrizes relativamente grandes, consumindo uma parte considerável da CPU.

8.1.4 Memória Alocada

Outra característica relevante é a memória alocada pelo sistema. Assim como o consumo de CPU, este fator pode ser limitado pelo hardware do robô. A Tabela 8.6 apresenta a quantidade de *Bytes* alocados pelas matrizes e variáveis criadas pelas técnicas de localização. Estes dados foram coletados através do comando *whos* do MatLab.

Quantidade de Bytes Alocados [KB]			
Técnica	Bytes Alocados (IC 95%)	Menor Resultado	Maior Resultado
Filtro de Bayes	3113.046 \pm 90.319	3003.461	3353.717
Markov	107.117 \pm 0.381	106.527	107.635
Filtro de K. Dinâmico	168.792 \pm 0.1958	168.478	169.130
Filtro de K. Estendido	167.007 \pm 0.302	166.610	167.866
Monte Carlo	130.213 \pm 0.109	129.933	130.381

Tab. 8.6: Tabela contendo a quantidade de memória alocada nas técnicas.

O filtro de Bayes foi a técnica que mais demandou de memória. Isto decorre do mesmo fator descrito na subseção 8.1.3, utilizar matrizes enormes para representar sua crença. Na ordem decrescente do uso de memória alocada, vêm em seguida os filtros gaussianos. Apesar de apresentarem um consumo de CPU relativamente pequeno frente ao consumo das demais técnicas, ambos os métodos usam 46 leituras de raio laser, como ilustrado na Figura 6.1, e para guardar todas as informações proveniente destes valores estes métodos demandam mais memória.

O método de Monte Carlo apresentou em média 130.213 KB de memória alocada. Apesar de ser o método com maior consumo de CPU e de guardar uma matriz contendo a postura de cada partícula, este não utiliza outras grandes matrizes, mantendo a quantidade de memória alocada em um nível mais baixo que as técnicas já citadas nesta subseção. Por último está o método de Markov. O valor correspondente a esse método na Tabela 8.6 mostra que as limitações implementadas na discretização do ambiente que guardam a crença e na quantidade de valores de orientação possíveis, explicadas no capítulo específico da técnica, fizeram efeito. Caso a quantidade de valores de orientação possíveis aumentasse, provavelmente o método despontaria como um dos que mais precisam de memória.

8.1.5 Linhas de Código

Um dos parâmetros na escolha de uma técnica é o esforço de implementação. Atualmente, o número de linhas de código é um dos índices usados para medir este esforço. Assim, a Tabela 8.7 reúne a quantidade de linhas de código de cada uma das técnicas utilizadas.

A partir da Tabela 8.7 é possível constatar que o Filtro de Bayes foi o método que exigiu menor

Quantidade de Linhas de Código	
Técnica	Quantidade de linhas
Filtro de Bayes	87
Markov	162
Filtro de K. Dinâmico	441
Filtro de K. Estendido	464
Monte Carlo	183

Tab. 8.7: Tabela contendo a quantidade de linhas de código das técnicas.

esforço para implementação, seguido dos métodos de Markov e Monte Carlo. Enquanto isso, os filtros Gaussianos demandaram mais de cinco vezes o esforço do filtro de Bayes, com 441 (filtro de Kalman dinâmico) e 464 (filtro de Kalman estendido) linhas de código. Observa-se que o resultado do uso do número de linhas para medir o esforço computacional está de acordo com a análise sobre a dificuldade de implementação, feita na subseção 8.2.1.

Embora que para a execução dos experimentos alguns outros algoritmos foram criados, como o sistema de controle de navegação do robô ou o algoritmo de criação das distribuições para o Filtro de Bayes, a Tabela 8.7 apresenta apenas o número de linhas do algoritmo específico da localização.

8.2 Comparação Qualitativa

A comparação qualitativa leva em conta a experiência adquirida ao longo deste estudo, lidando com a implementação, execução dos experimentos, simulações e testes, modificações e otimizações dos métodos mencionados.

8.2.1 Dificuldade de Implementação

A dificuldade de implementação é algo bastante subjetivo, principalmente tendo em vista a experiência do desenvolvedor com estimadores, processos estocásticos e robótica. Nesta análise, foi levada em conta a experiência apenas do autor desta dissertação.

O filtro de Bayes é, dentre os filtros empregados aqui, o que apresenta menor complexidade.

Seu algoritmo contém apenas dois passos, de fácil entendimento e implementação. Além disso, o cenário de aplicação é o mais simples, já que trabalha com o estado composto de apenas uma variável, x . O maior trabalho para implementação deste filtro foi a formação das funções de densidade de probabilidade para as portas do corredor. Para isso, foi necessário executar a navegação do robô algumas vezes, coletar informações a respeito das posições das portas, analisar os dados estatisticamente e a partir disto, gerar as curvas. Após isto, já com o algoritmo funcionando, foi obrigatório ainda um ajuste fino dos parâmetros (média e variância) destas curvas para o bom funcionamento da técnica.

A localização pelo método de Markov por sua vez, traz um pouco mais de complexidade, embora guarde bastante semelhança com o filtro de Bayes. Adicionalmente à dificuldade imposta pelo filtro de Bayes, para o caso realizado nesta dissertação, foi necessário trabalhar com um mapa em duas dimensões e com representação em grade. Com isso, as funções de densidade de probabilidade se tornaram funções bidimensionais. Além disso, foi necessário ajustar as dimensões das células de acordo com o nível de eficiência x tempo de execução oportuno.

O terceiro algoritmo na escala crescente de complexidade é o método de Monte Carlo. Um dos fatores que contribuem para isto é a mudança de pensamento quanto à manipulação da crença. Neste algoritmo, a crença é representada pelo conjunto de partículas ao invés da crença multimodal das técnicas anteriores. Além disso, esta técnica tem um algoritmo mais longo, com algumas peculiaridades, como a forma de se executar a reamostragem das partículas (Algoritmo 8). Nesta etapa deve-se estabelecer os parâmetros da função Sigmoid, da função gaussiana que gera o peso das partículas e principalmente da abertura que a nuvem terá com base no erro das partículas de referência.

Por fim, temos os filtros Gaussianos, considerados pelo autor desta dissertação, os mais complexos para implementar. Esta opinião se embasa no fato da técnica lidar com uma matemática mais complexa e com um algoritmo mais denso em comparação com as demais técnicas. Para a implementação deste método foi necessário modelar o erro tanto dos sensores exteroceptivos quanto dos proprioceptivos, como descrito no Capítulo 3. Na fase de Observação, empregou-se o algoritmo

Split-and-Merge para, a partir das leituras do laser, formar retas, que por sua vez, compõem um mapa local. Além disso, vários de seus detalhes merecem atenção, como descritos passo-a-passo nos textos de referência apontados no Capítulo 6.

8.2.2 Deficiências

Tão importante quanto discutir as vantagens ou analisar a eficiência de cada técnica, é a abordagem de suas deficiências. Ao longo do contato que o autor teve com os algoritmos, alguns pontos chamaram atenção e valem ser apontados.

Filtro de Bayes

Uma das limitações do filtro de Bayes é de, por tratar a crença como uma função contínua, exigir mais memória que as demais técnicas e por isso, acaba sendo aplicada em sistemas de complexidade menor, como o cenário exposto no Capítulo 4. Além disso, a quantidade de características específicas (*landmarks*), assim como seus estados, devem ser cuidadosamente analisados. No caso apresentado no capítulo do filtro em questão, as portas estavam a uma distância praticamente constante entre si, o que pode dificultar a estimação da postura do robô, uma vez que os lóbulos da crença coincidiam com a distribuição das portas, mantendo o número de lóbulos constante. Uma forma de contornar este problema é definir características específicas diferentes. No experimento, o estado das portas na parede da direita e da esquerda foram definidas de tal configuram que foram uma topologia quase que única no corredor, e assim, ao inserir a distribuição de ambas as paredes no algoritmo, a crença coincide com um número bem menor de lóbulos da distribuição, e a estimação de postura acontece mais rapidamente. Entretanto, esta solução traz uma deficiência ao sistema, pois fica refém do estado das portas. Ao se considerar um ambiente dinâmico, não se teria um controle sobre o estado das portas das salas e estas não seriam mais apropriadas para uso no filtro de Bayes.

Localização pelo método de Markov

A localização pelo método de Markov apresenta o mesmo problema que o filtro de Bayes no que tange a simetria das portas. Esta dificuldade de estimação pode ser visualizada na Figura 5.3. A crença mantém mais de um lóbulo até o ponto 5, praticamente ao final do corredor. Este problema é enfatizado ainda mais pela simetria entre a sequência de topologias (formadas pelo estado combinado das portas opostas) em uma direção e em outra. Outra limitação surge da discretização do mapa e da quantidade de ângulos possíveis para a orientação do robô, visto que geram uma grande demanda computacional.

Filtros Gaussianos

Os filtros gaussianos exigem uma fidelidade grande para com o modelo do mapa do ambiente. Nem sempre é possível filtrar os dados coletados pelos sensores exteroceptivos, e o erro causado por estar acima dos valores desejados. Mas a maior deficiência do método é não tratar multi-hipóteses. O algoritmo só abrange a localização local, ou seja, é necessário o conhecimento prévio da postura inicial do robô. Decorrente disto, uma outra desvantagem é que o algoritmo pode divergir caso o robô seja sequestrado, ou mais simples, caso algum obstáculo inesperado atrapalhe a formação do mapa local e a fase de atualização da crença aponte para uma postura distante da real. Embora os demais métodos também estejam suscetíveis ao mesmo problema, por esses trabalharem com multi-hipóteses, a crença pode ser restaurada com algumas iterações.

Filtro de Partículas - Monte Carlo

Embora tenha apresentado os melhores resultados em termos quantitativos, o filtro de partículas também apresenta algumas deficiências. Como mencionado anteriormente, o tempo de execução pode ser um limitador para algumas aplicações. Entretanto, com base na experiência do autor, a maior limitação remete ao número de partículas. Para um ambiente com área de aproximadamente 54.4 m^2 , uma certa quantidade de partículas é exigida, ou o algoritmo pode não convergir em tempo

hábil. Mas esta desvantagem é ainda mais profunda. Mesmo com um número suficiente de partículas, o algoritmo ainda corre o risco de não convergir. Isto decorre do fato de se depender de como elas serão amostradas pelo ambiente. Como são amostradas randomicamente, é possível que, mesmo que suas posturas estejam cobrindo todo o mapa, suas orientações sejam opostas à real orientação do robô. Este problema é acentuado quando se trabalha com um mapa simétrico, como é o caso. A crença pode convergir para o ponto errado e não encontrar mais a postura do robô.

Capítulo 9

Conclusão

Este capítulo conclui a dissertação, e encontra-se dividido em três seções. A seção 9.1 trata do trabalho como um todo, enquanto a seção 9.2 apresenta uma retrospectiva dos principais pontos e contribuições de cada capítulo. Finalmente, a seção 9.3 propõe ideias para os trabalhos futuros com relação ao estudo encontrado neste texto.

9.1 Considerações Finais

Os métodos de localização implementados nesta dissertação tiveram desempenhos bastante satisfatórios, dentro dos limites propostos por cada uma das técnicas. De modo geral, alcançou-se o objetivo de expor o funcionamento de cada um deles, suas características principais, seus algoritmos, os cenários em que geralmente atuam, suas eficiências, limites, etc. Especialmente no capítulo 8, atingiu-se o real alvo do trabalho, analisar e discutir alguns fatores comuns aos métodos avaliados. Apesar das técnicas proporem soluções diferentes e não serem normalmente aplicados nos mesmos cenários, tentou-se analisar suas características de modo abrangente, produzindo informações ricas ao leitor, inclusive com carácter qualitativo. A elaboração deste capítulo visou auxiliar os desenvolvedores de sistemas robóticos na escolha do método ideal para se trabalhar.

9.2 Contribuições

Ao longo da dissertação, discutimos algumas formas de se realizar localização robótica, cada uma com um cenário bem definido. As contribuições estabelecidas na seção 1.4, quais sejam, realizar um estudo comparativo de métodos de localização robótica utilizando o mesmo arranjo experimental e a disponibilidade destes algoritmos na forma de uma biblioteca, foram alcançados.

No arranjo experimental, utilizou-se o mesmo equipamento, ambiente, sistema de desenvolvimento e sistema de acesso ao robô.

9.3 Trabalhos Futuros

Este trabalho gera diversas possibilidades de trabalhos futuros em diversas frentes, os quais são citados a seguir e ilustrados na Figura 9.1.

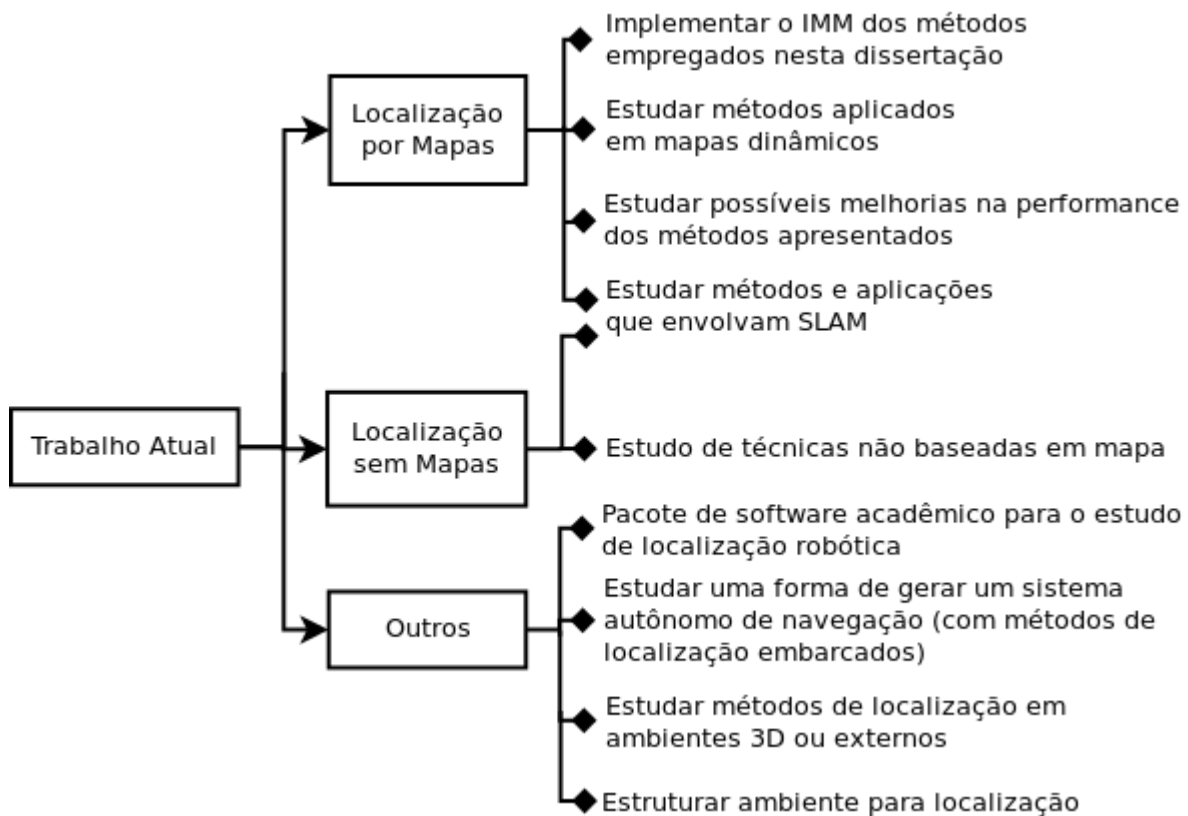


Fig. 9.1: Possíveis Trabalhos Futuros

- Implementar o IMM (Interacting Multiple Model) dos métodos empregados nesta dissertação;

O IMM consiste em um esquema para executar vários algoritmos paralelamente, alimentados com as respectivas entradas e determinando pesos para suas saídas, de forma a obter uma estimação com menor incerteza. A implementação de um IMM com os métodos empregados nesta dissertação faria com que a estimação de postura não ficasse refém das limitações de um algoritmo específico, tornando o sistema mais robusto.

- Estudar métodos aplicados em mapas dinâmicos;

Todos os métodos aplicados neste texto foram empregados em mapas estáticos, onde os obstáculos são considerados estáticos durante o processo de localização. Para ampliar a quantidade de âmbitos em que o robô pudesse interagir, um estudo em meios dinâmicos seria bastante cativante e proveitoso.

- Estudar possíveis melhorias na performance dos métodos apresentados

É legítimo que os métodos podem ser melhorados em diversas de suas características, e há espaço para isso. Pode-se trabalhar com fusão de alguns métodos ou até desenvolvimentos de uma nova técnica que resolva um problema específica com rendimento maior que as demais.

- Estudar métodos e aplicações que envolvam SLAM;

O conhecimento adquirido acerca das técnicas de localização baseadas em mapas propicia o avanço da pesquisa para o próximo patamar, o de localiza o robô enquanto se mapeia o ambiente. SLAM (Simultaneous Localization and Mapping) viabiliza a utilização do sistema em muito mais cenários, uma vez que eliminaríamos um dos requisitos de funcionamento, o mapa.

- Estudar métodos de localização não baseados em mapas;

A seção 2.3 apresenta diversas formas de se realizar localização não baseadas em mapas. Este estudo seria interessante pois esses métodos poderiam resolver alguns dos desafios encontrados

aqui de forma mais simples e/ou rápida. A ideia de poder fazer uso de mais tipos de localização é algo que torna o sistema muito mais poderoso.

- Pacote de software acadêmico para o estudo de localização robótica

Por fim, utilizar os métodos trabalhados no mundo acadêmico, a fim de poder testá-los em diversos tipos de cenários com o intuito de gerar discussões e desenvolver pesquisadores na área.

- Estudar uma forma de gerar um sistema autônomo de navegação (com métodos de localização embarcados);

Iniciar um estudo que tornasse factível um sistema de navegação autônoma. Neste sistema, o robô pode se dar conta de que está perdido ou de que já acumulou erro de odometria suficiente para dificultar a execução de próxima tarefa, e decidir por si só qual a melhor técnica a ser utilizada em cada caso. Apesar de requer uma base maior em inteligência artificial, este trabalho abrilhantaria o sistema autônomo com uma ferramenta deveras robusta.

- Estudar métodos de localização em ambientes 3D ou externos;

Uma outra característica deste estudo é que todos os procedimentos foram executados em ambientes de duas dimensões, já que se tratam de lugares internos. Trabalhar com localização também em ambientes externos é bastante interessante pois põe fim a mais uma das limitações do nosso sistema, no que diz respeito aos meios de aplicação. Com este conhecimento adicional, o robô poderia navegar tanto em ambientes internos como externos, aumentando a quantidade de aplicações a qual ele poderia se destinar.

- Estudar formas de estruturar um ambiente para ajudar na localização;

Já que nesta dissertação o robô navega por ambientes costumeiramente frequentados por humanos, é de se esperar que haja controle sobre sua infraestrutura. Uma forma relevante de apoiar todo o sistema robótico é propiciar um ambiente que facilite sua operação. Quais

as melhores formas de se estruturar um recinto e quais vantagens cada uma destas formas proporcionam?

- Sensores de baixo custo;

Implantar um sistema de localização que utilize sensores de baixo custo, como sonar, infravermelho, etc.

Referências

- [1] FA Worsley. *Shackleton's boat journey*. Jove Publications, Inc., New York, N.Y., 1977.
- [2] D. Rodrigues, L. Olivi, R. Souza, E. Guimarães, and E. Cardozo. Continuous Topology Learning and Early Recognition for Mobile Robots Navigation. *CBA, 2012*, 2012.
- [3] H. Choset, K. M. Lynch, and S. Hutchinson. *PRINCIPLES OF ROBOT MOTION, Theory, Algorithms and Implementations*, volume 24. The MIT Press, March 2005.
- [4] Roland Siegwart and Illah R Nourbakhsh. *Introduction to Autonomous Mobile Robots*, volume 23. The MIT Press, 2004.
- [5] J. R. Oliveira, E. Cardozo, E. G. Guimarães, G. Castellano, G. P. Coelho, L. L. Min, P. T. B. Fernandes, R. J. M. Covolán, and R. R. F. Attux. Desenvolvimento de Tecnologias da Informação para Neurologia (DesTINE). Technical report, Unicamp, 2011.
- [6] Destine Project. <http://destine.dca.fee.unicamp.br:9090/>, 4 February 2013.
- [7] Sebastian Thrun. *Probabilistic robotics*, volume 45. The MIT Press, March 2002.
- [8] J. Borenstein, H. R. Everett, and L. Feng. *Where am I? Sensors and methods for mobile robot positioning*. 1996.
- [9] K. Kozłowski. *Robot Motion and Control*. Springer, 2004.
- [10] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation. 2001*. John Wiley and Sons Ltd., 2001.

- [11] D. L. Hall and J. Llinas. Handbook of multisensor data fusion. *Handbook of Multisensor Data Fusion*, 2001.
- [12] J. R. Rao. *Multi-sensor data fusion with MATLAB*. 2009.
- [13] K. S. Chong and L. Kleeman. Accurate odometry and error modelling for a mobile robot. *IEEE International Conference on Robotics and Automation, 1997. Proceedings.*, (April):2783–2788, 1997.
- [14] R. C. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, December 1986.
- [15] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, pages 267–288, 1986.
- [16] V. L. M. Sanches. *Um sistema de localização robótica para ambientes internos baseado em redes neurais*. PhD thesis, Universidade de São Paulo, 2011.
- [17] J. L. Crowley. World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging. 3(February):1574–1579, 1989.
- [18] D. Sack and W. Burgard. A comparison of methods for line extraction from range data. *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV), 2004*, (1973), 2004.
- [19] K. O. Arras and R. Siegwart. Feature extraction and scene interpretation for map-based navigation and map building. *Proc. of SPIE, Mobile Robotics XII*, 3210, 1997.
- [20] R. C. Luo and Cheng-T. Chen. Indoor localization using line based map for autonomous mobile robot. In *2008 IEEE Workshop on Advanced robotics and Its Social Impacts*, pages 1–6. IEEE, August 2008.
- [21] AM Robotics. Pioneer 3 operations manual. *ActivMedia robotics, Amherst, USA*, 2004.

- [22] Robosoft SA & Activ Media Robotics. World's most popular intelligent wheeled robot. pages 1–4.
- [23] H. R. Everett. *Sensors for mobile robots*. 1995.
- [24] R M White. A sensor classification scheme. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 34(2):124–6, January 1987.
- [25] K. O. Arras. Line Extraction First-Order Error Propagation. pages 1–7.
- [26] N. J. Gordon, Salmond D. J., and Smith A. F. M. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107 – 113, 1993.
- [27] V. Fox, J. Hightower, and L. Liao. Bayesian filters for location estimation. *Pervasive Computing, IEEE*, 2(3):24 – 33, 2003.
- [28] W. Burgard and A. Derr. Integrating global position estimation and position tracking for mobile robots: the dynamic Markov localization approach. *Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems Innovations in Theory Practice and Applications*, 2(October):730–735, 1998.
- [29] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids. 1994.
- [30] J. S. Gutmann, W. Burgard, F. Dieter, and K. Konolige. An experimental comparison of localization methods. *Intelligent Robots and . . .*, (October):736–743, 1998.
- [31] M. L. Anjum, J. Park, and W. Hwang. Sensor data fusion using Unscented Kalman Filter for accurate localization of mobile robots. *Control Automation and Systems (ICCAS), 2010 International Conference on*, pages 947–952, 2010.

- [32] I. Arasaratnam and S. Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009.
- [33] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems 1. 82(Series D):35–45, 1960.
- [34] G. Welch and G. Bishop. An introduction to the Kalman filter. pages 1–16, 1995.
- [35] R. Olfati-saber. Distributed Kalman Filtering and Sensor Fusion in Sensor Networks. pages 1–13, 2006.
- [36] S. I. Roumeliotis and G. A. Bekey. Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 3:2985 – 2992, 2000.
- [37] J. S. Gutmann. Markov-Kalman localization for mobile robots. *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2:601–604, 2002.
- [38] K. O. Arras, N. Tomatis, and R. Siegwart. Multisensor on-the-fly localization: Precision and reliability for applications. *Robotics and Autonomous Systems*, 34:131–143, 2001.
- [39] J. S. Gutmann and D. Fox. An experimental comparison of localization methods continued. *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 1(October):454 – 459, 2002.
- [40] F. Kong, Y. Chen, and J. Xie. Mobile robot localization based on extended kalman filter. *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, 2(60474021):9242–9246, 2006.
- [41] Y.-C. Lee, W. Yu, J.-H. Lim, W.-K. Chung, and D.-W. Cho. Sonar Grid Map Based Localization for Autonomous Mobile Robots. *2008 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, pages 558–563, October 2008.

- [42] R. Chen, H. Zhao, and B. Xiao. Self-localization of mobile robot based on monocular and extended kalman filter. *2009 9th International Conference on Electronic Measurement & Instruments*, pages 2–450–2–454, August 2009.
- [43] B. Schiele and J. L. Crowley. A comparison of position estimation techniques using occupancy grids. *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 1628–1634, 1994.
- [44] Andre M Santana, Anderson A S Sousa, Ricardo S Britto, Pablo J Alsina, and Adelardo A D Medeiros. Localization of a mobile robot based in odometry and natural landmarks using extended kalman filter. (1).
- [45] J. Carpenter, P. Clifford, and P. Fearnhead. An Improved Particle Filter for Non-linear Problems. pages 1–14.
- [46] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *IN PROC. OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI, (Handschin 1970):343–349*, 1999.
- [47] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.
- [48] G. Cen and N. Matsuhira. Service robot localization using improved particle filter. *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, (September):2454–2459, 2008.
- [49] B. Yin, Z. Wei, Y. Cong, and T. Xu. A Novel Particle Filter Method for Mobile Robot Localization. *2010 International Conference on Measuring Technology and Mechatronics Automation*, pages 269–272, March 2010.

- [50] C.-C. Hsu, C.-C. Wong, H.-C. Teng, N.-J. Li, and C.-Y. Ho. Localization of mobile robots via an enhanced particle filter. *2010 IEEE Instrumentation & Measurement Technology Conference Proceedings*, pages 323–327, 2010.
- [51] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 2:1322–1328.
- [52] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, (February), 2001.
- [53] L. Zhang, R. Zapata, and P. Lépinay. Self-adaptive monte carlo for single-robot and multi-robot localization. *Automation and Logistics, 2009. ICAL '09. IEEE International Conference on*, (August):1927–1933, 2009.
- [54] S. Seifzadeh, D. Wu, and Y. Wang. Cost-effective active localization technique for mobile robots. *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 539–543, December 2009.
- [55] A. Doucet. On sequential simulation-based methods for Bayesian filtering. pages 1–26, 1998.