JORGE PABLO MORAGA GALDAMES

**SISTEMA DE LOCALIZAÇÃO PARA AGVS EM AMBIENTES SEMELHANTES A
ARMAZÉNS INTELIGENTES**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Mecânica.

Área de concentração:
Dinâmica de Máquinas e Sistemas.

Orientador:
Prof. Assoc. Marcelo Becker

São Carlos
2011

ESTE EXEMPLAR TRATA-SE DA VERSÃO CORRIGIDA DA DISSERTAÇÃO.
A VERSÃO ORIGINAL ENCONTRA-SE DISPONÍVEL NA EESC/USP.

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

G149s Galdames, Jorge Pablo Moraga.
Sistema de localização para AGVS em ambientes
semelhantes a armazéns inteligentes. / Jorge Pablo Moraga
Galdames ; orientador Marcelo Becker. São Carlos, 2011.

Dissertação (Mestrado - Programa de Pós-Graduação em
Engenharia Mecânica e Área de Concentração em Dinâmica de
Máquinas e Sistemas)-- Escola de Engenharia de São Carlos
da Universidade de São Paulo, 2011.

1. Robótica móvel. 2. Armazém inteligente. 3. AGV. 4.
Auto-localização. 5. Filtro de Kalman estendido. 6.
Feature-based map. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **JORGE PABLO MORAGA GALDAMES**.

Título da dissertação: "Sistema de localização para AGVS em ambientes semelhantes a armazéns inteligentes".

Data da defesa: 23/04/2012

Comissão Julgadora:

Resultado:

Prof. Associado **Marcelo Becker (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Daniel Varela Magalhães**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Denis Fernando Wolf**
(Instituto de Ciências Matemáticas e de Computação/ICMC)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Marcelo Areias Trindade**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo Cesar Lima Segantine**

9-102023

Protocolado em 23/04/2012

Protocolado em 23/04/2012

Protocolado em 23/04/2012

AGRADECIMENTOS

A Deus por me mostrar os verdadeiros valores da vida.

A minha família por seu amor e carinho incondicionais.

A meus padrinhos Lélío e Zezé por compartilharem sua alegria e entusiasmo.

Ao Prof. Glauco Augusto de Paula Caurin por me receber e convidar a fazer parte do grupo que na época se intitulava Laboratório de Mecatrônica.

Ao meu orientador Prof. Marcelo Becker pela sua compreensão, dedicação e pelo seu grande entusiasmo.

Aos meus amigos pelo apoio, motivação e amizade.

A minha amiga Kelen por compartilhar suas alegrias, angústias e conhecimentos, e pelo apoio durante toda esta jornada.

Aos professores, amigos e colegas do Grupo de Mecatrônica que dedicaram parte de seu tempo para dividir seus conhecimentos ou para tomar um café.

Aos professores e alunos do Laboratório de Robótica Móvel (LRM) do Instituto de Ciências Matemáticas e de Computação (ICMC-USP) pelo empréstimo dos equipamentos e pela ajuda concedida para a execução dos experimentos.

A todos os funcionários do Laboratório de Dinâmica e da Secretaria da Pós-Graduação que contribuíram para a realização deste trabalho.

Ao CNPq pela bolsa concedida.

RESUMO

GALDAMES, J. P. M. (2011). **Sistema de Localização para AGVs em Ambientes Semelhantes a Armazéns Inteligentes**. 100p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

A demanda por mais flexibilidade nas fábricas e serviços originou um aumento no volume de operações internas de carga e descarga, devido à maior diversidade dos elementos transportados. Logo, na busca por um fluxo de materiais mais eficiente, as empresas passaram a investir em soluções tecnológicas, entre elas, o uso de *Automated Guided Vehicles* (AGVs), por conta do custo mais atrativo e do avanço em relação aos primeiros AGVs, que até então dependiam de uma infraestrutura adicional para suportar a navegação. Muitos AGVs modernos possuem movimentação livre e são orientados por sistemas que utilizam sensores para interpretar o ambiente, sendo assim, tornar os AGVs autônomos despertou o interesse de pesquisadores na área de robótica móvel para o desenvolvimento de sistemas capazes de auxiliar e coordenar a navegação. Novas técnicas de localização, tal como a localização baseada em marcadores reflexivos, e a construção de armazéns com *layouts* estruturados para a navegação viabilizaram o uso de AGVs autônomos, entretanto sua utilização em armazéns existentes ainda é um desafio. Neste contexto, o Laboratório de Robótica Móvel (LabRom) do Grupo de Mecatrônica da EESC/USP, através do projeto do Armazém Inteligente, tem pesquisado os problemas de: roteamento, gerenciamento das baterias, navegação e auto-localização. Robôs autônomos precisam de um sistema de auto-localização eficiente e preciso para navegar com segurança, o qual depende de um mapa e da interpretação do ambiente utilizando sensores embarcados. Para alcançar esse objetivo este trabalho propõe um Sistema de Auto-localização baseado no *Extended Kalman Filter* (EKF) como solução. O sistema, desenvolvido em linguagem C, interage com outros dois sistemas: roteamento e navegação e foi implementado em um armazém simulado utilizando o *software* Player/Stage, mostrando ser confiável no fornecimento de uma estimativa de localização baseada em odometria e *landmarks* com localização conhecida. O sistema foi novamente testado utilizando a odometria de um robô móvel Pioneer P3-AT e os valores de um sensor de medição laser 2D SICK LMS200 extraídos de um ambiente *indoor* real. Para este teste foi construído um *feature-based map* a partir de um desenho de planta baixa no formato CAD e utilizou-se o algoritmo de segmentação *Iterative End-Point Fit* (IEPF) para interpretar o ambiente. Os resultados mostraram que as vantagens oferecidas pelas características padronizadas de um ambiente *indoor*, semelhante a um armazém, podem viabilizar o uso do Sistema de Auto-localização em armazéns existentes.

Palavras-chaves: Robótica Móvel. Armazém Inteligente. AGV. Auto-localização. Filtro de Kalman Estendido. *Feature-based map*.

ABSTRACT

GALDAMES, J. P. M. (2011). **Location System for AGVs in Environments Similar to Smart Warehouses**. 100p. M.Sc Thesis (Master) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

The demand for more flexibility in factories and services led to an increase in the volume of internal operations of loading and unloading, due to the greater diversity of elements transported. Hence, in the search for a more efficient materials flow, companies went to invest in technology solutions, among them, the use of Automated Guided Vehicles (AGVs), on account of the more attractive cost and improvement over the first AGVs, which hitherto depended of an additional infrastructure to support navigation. Many modern AGVs have free movement and are guided by systems that use sensors to interpret the environment, thus make AGVs autonomous aroused the interest of researchers in the mobile robotics field to development of systems able to assist and coordinate the navigation. New localization techniques, such as localization based on reflective markers, and the construction of warehouses with structured layouts for navigation did feasible the use of autonomous AGVs, however its use in existing warehouses is still a challenge. In this context, the Mobile Robotics Lab (LabRom) of the Mechatronics Group of EESC/USP, through the Intelligent Warehouse Project, has researched the problems: routing, battery management, navigation and self-localization. Autonomous robots need an efficient and accurate self-localization system to safely navigate, which depends on one map and of the interpretation of the environment using embedded sensors. To achieve this goal, this work proposes a Self-Localization System based on the Extended Kalman Filter (EKF) as a solution. The system, developed in C language, interacts with two other systems: routing and navigation and was implemented in a simulated warehouse using the Player/Stage software, showing to be reliable in providing an estimative of localization based on odometry and landmarks with known localization. The system was again tested using the odometry of mobile robot Pioneer P3-AT and the values of a 2D Laser Rangefinder SICK LMS200 extracted from a real indoor environment. For this test was built a feature-based map from a floor plan design in CAD format and was used the segmentation algorithm Iterative End-Point Fit (IEPF) to interpret the environment. The results showed that the advantages offered by the standard features of indoor environment, like a warehouse, can enable the use of the Self-Localization System on the existing warehouses.

Keywords: Mobile Robotic. Smart Warehouse. AGV. Self-localization. Extended Kalman Filter. Feature-based map.

LISTA DE FIGURAS

<i>Figura 1.1: AGV de empilhadeira adaptada.</i>	9
<i>Figura 1.2: AGV – Automated Guided Vehicle AGVRobots.com (2011).</i>	9
<i>Figura 1.3: KIVA Systems.</i>	10
<i>Figura 1.4: Fábrica Mercedes-Benz de Juiz de Fora, MG.</i>	10
<i>Figura 2.1: Ilustração das técnicas mais usualmente encontradas em soluções oferecidas por empresas que atuam na automação de empilhadeiras em armazéns.</i>	17
<i>Figura 2.2: Sensor SICK NAV300-2232, utilizado na navegação com marcadores refletivos.</i>	18
<i>Figura 2.3: Armazém de Bobinas de Papel.</i>	18
<i>Figura 2.4: Sensor de medição laser SICK NAV300 embarcado no AGV que navega no interior de uma carreta.</i>	19
<i>Figura 2.5: Player/Stage no Linux Ubuntu.</i>	24
<i>Figura 3.1: Hipótese de Markov.</i>	26
<i>Figura 3.2: Localização de Markov em um cenário de localização 1D.</i>	30
<i>Figura 3.3: O MCL em um cenário de localização 1D.</i>	33
<i>Figura 3.4: Diagrama Geral para a Localização de Robôs Móveis.</i>	34
<i>Figura 3.5: Modelo de Odometria - O movimento do robô no intervalo $(t-1, t]$.</i>	35
<i>Figura 3.6: O Filtro de Kalman em um cenário de localização 1D.</i>	38
<i>Figura 3.7: Ilustração da etapa de split do IEPF.</i>	40
<i>Figura 4.1: Arquitetura proposta para os sistemas envolvidos na aplicação de AGVs de empilhadeiras em armazéns inteligentes.</i>	45
<i>Figura 4.2: Imagem da planta 2D do armazém inteligente utilizado na simulação.</i>	48
<i>Figura 4.3: Landmarks inseridos no ambiente simulado.</i>	49

<i>Figura 4.4: Configuração geral do experimento para o teste do Algoritmo de Auto-localização 2D no ambiente indoor real.</i>	50
<i>Figura 4.5: Ambiente real – Sistema de coordenadas global escolhido.</i>	51
<i>Figura 4.6: Detalhe do desenho de projeto da planta baixa do primeiro pavimento do Prédio da Engenharia Mecatrônica da EESC – USP.</i>	51
<i>Figura 4.7: Esboço da planta baixa com as medidas reais obtidas.</i>	52
<i>Figura 4.8: Construção do Mapa 2D e do Feature-based Map.</i>	53
<i>Figura 4.9: Mapa 2D no formato CAD, em escala aproximada de 1:200, obtido a partir da planta baixa após ajuste das cotas de construção.</i>	54
<i>Figura 4.10: Classificação das áreas navegáveis no Mapa 2D.</i>	55
<i>Figura 4.11: Trajeto planejado – Experimento N° 01.</i>	56
<i>Figura 4.12: Trajeto planejado – Experimento N° 02.</i>	56
<i>Figura 4.13: Ponto de partida dos trajetos planejados para os experimentos 1 e 2.</i>	57
<i>Figura 4.14: Montagem do sensor de medição laser 2D SICK-LMS200 na plataforma do robô móvel Pioneer P3-AT.</i>	58
<i>Figura 4.15: Linhas do Mapa 2D selecionadas para serem incluídas no feature-based map</i>	60
<i>Figura 4.16: Linhas geométricas não recomendadas para a inclusão no feature-based map.</i>	63
<i>Figura 4.17: Robô móvel Pioneer 3-AT.</i>	65
<i>Figura 4.18: Posicionamento do roteador wireless para a coleta dos dados sensoriais.</i>	66
<i>Figura 4.19: Medidas coletadas em milímetros para o cálculo da localização inicial no ponto de partida do trajeto do experimento n° 01.</i>	67
<i>Figura 4.20: Medidas coletadas em milímetros para o cálculo da localização inicial no ponto de partida do trajeto do experimento n° 02.</i>	67
<i>Figura 5.1: Tarefa simulada de carga e descarga utilizando o Player/Stage</i>	70
<i>Figura 5.2: Valores de posição gerados pelo Player/Stage</i>	71
<i>Figura 5.3: Valores de posição gerados pelo Player/Stage com adição de erro aleatório</i>	72
<i>Figura 5.4: Valores de posição estimados pelo EKF</i>	73

<i>Figura 5.5: EXPERIMENTO 1 – Sequência dos valores de posição XY estimados pela odometria real a partir da junção de dois logs</i>	<i>76</i>
<i>Figura 5.6: EXPERIMENTO 1 – Sequência dos valores de posição XY estimados pelo EKF em parte do primeiro trecho</i>	<i>77</i>
<i>Figura 5.7: EXPERIMENTO 2 – Sequência dos valores de posição XY estimados pela odometria real.....</i>	<i>78</i>
<i>Figura 5.8: EXPERIMENTO 2 – Odometria do Trajeto de Ida</i>	<i>79</i>
<i>Figura 5.9: EXPERIMENTO 2 – Odometria do Trajeto de Retorno</i>	<i>80</i>
<i>Figura 5.10: EXPERIMENTO 2 – Sequência dos valores de posição XY estimados pelo EKF</i>	<i>80</i>
<i>Figura 5.11: EXPERIMENTO 2 – EKF do Trajeto de Ida.....</i>	<i>82</i>
<i>Figura 5.12: EXPERIMENTO 2 – EKF do Trajeto de Retorno</i>	<i>82</i>
<i>Figura 6.1: Mini-empilhadeira Robótica.</i>	<i>89</i>

LISTA DE QUADROS

<i>Quadro 4.1: Feature-based map composto por características do tipo linhas geométricas definidas em função dos parâmetros (r, α).</i>	<i>64</i>
<i>Quadro 5.1: Diferença entre a localização Real e a EKF (erro absoluto)</i>	<i>71</i>
<i>Quadro 5.2: Número de landmarks visualizados durante o trajeto simulado</i>	<i>74</i>

LISTA DE GRÁFICOS

<i>Gráfico 5.1: Visibilidade dos landmarks.</i>	<i>74</i>
<i>Gráfico 5.2: Taxa de proporção entre os landmarks detectados/identificados.</i>	<i>75</i>

LISTA DE SIGLAS

AGVs	<i>Automated Guided Vehicles</i>
AS/RS	<i>Automated Storage & Retrieval System</i>
EESC	Escola de Engenharia de São Carlos
EKF	<i>Extended Kalman Filter</i>
GPS	<i>Global Positioning System</i>
IEPF	<i>Iterative End-Point Fit</i>
JIT	<i>Just in Time</i>
KF	<i>Kalman Filter</i>
LGV	<i>Laser Guided Vehicle</i>
LT	<i>Line Tracking</i>
MCL	<i>Monte Carlo Localization</i>
MRS	<i>Multi-robot Systems</i>
POMDP	<i>Partially Observable Markov Decision Process</i>
RDP	Ramer-Douglas-Peucker
RFID	<i>Radio-Frequency IDentification</i>
SEF	<i>Successive Edge Following</i>
SKUs	<i>Stock Keeping Units</i>
UKF	<i>Unscented Kalman Filter</i>
USP	Universidade de São Paulo
WMS	<i>Warehouse Management Systems</i>

SUMÁRIO

1 INTRODUÇÃO	1
1.1 Motivação	1
1.2 O Uso de AGVs na Logística	4
1.3 A Navegação de um AGV Inteligente	7
1.4 Projetos de AGVs	8
1.5 O Projeto: Armazém Inteligente	10
1.6 Objetivo	11
1.7 Organização da Dissertação	12
2 A LOCALIZAÇÃO DE ROBÔS MÓVEIS EM AMBIENTES INDOOR	15
2.1 O problema da Localização na Robótica Móvel	15
2.2 Técnicas de Localização de AGVs	16
2.3 Classificação das Técnicas de Localização	20
2.4 O Mapeamento de Ambientes	23
2.5 O <i>Software</i> de Simulação Player/Stage	23
3 ALGORITMOS DO SISTEMA DE AUTO-LOCALIZAÇÃO	25
3.1 A Localização de Markov	25
3.2 A Localização de Monte Carlo	30
3.3 A Localização baseada no Filtro de Kalman Estendido	33
3.4 Algoritmos de Extração de Segmentos de Linhas	39
4 METODOLOGIA	43
4.1 O Ambiente Simulado	44
4.1.1 Representação do Armazém Inteligente	47
4.1.2 Inclusão dos <i>Landmarks</i> Artificiais	48
4.2 O Ambiente Real	49
4.2.1 Construção do <i>feature-based map</i>	50
4.2.2 Coleta dos dados sensoriais reais e pós-processamento	65
5 RESULTADOS	69
5.1 Resultados do Ambiente Simulado	69
5.2 Resultados do Ambiente Real	75

5.2.1 Experimento 1.....	75
5.2.2 Experimento 2.....	77
6 CONCLUSÃO E TRABALHOS FUTUROS	85
7 REFERÊNCIAS	91

1 INTRODUÇÃO

Neste capítulo é apresentada a motivação para o desenvolvimento deste trabalho e a sua relevância no contexto da logística atual. Também são apresentados os objetivos e a descrição da proposta, bem como as possíveis contribuições deste trabalho. Em seguida, é descrita a organização dos capítulos desta dissertação.

1.1 Motivação

Hoje, em um mundo cada vez mais competitivo, a pressão exercida pela concorrência força as empresas a reduzirem custos nas áreas de negócios que não fazem parte do seu *core business*, uma opção de redução é através do uso da logística moderna (WANG; ZHANG; HE, 2009). Neste contexto, as empresas reestruturam seus processos internos para adaptar-se às mudanças de mercado e ao surgimento de novas tecnologias. Assim, a automação do transporte passou a ser uma questão de alta prioridade para as empresas modernas. E nesse sentido, as empresas sentem a necessidade de automatizar seus armazéns com o objetivo de melhorar a produtividade, a flexibilidade e a qualidade (MORA et al., 2003).

A competitividade e a volatilidade do mercado são fatores que levam as empresas a buscarem maiores produtividade e eficiência e, ao mesmo tempo, reduzirem custos e impactos ambientais e ocupacionais. Nesse cenário, a logística se destaca entre as estratégias empresariais para a maximização do desempenho dos sistemas de distribuição de produtos, buscando menores custos e melhores níveis de serviço. Investimentos em melhorias operacionais, com o objetivo de aumentar a velocidade de resposta à demanda através de um fluxo eficiente de materiais, são cada vez mais frequentes (ODA et al., 2009) (VIVALDINI, 2010).

Sendo assim, nas últimas décadas, as mudanças na economia global tem redefinido significativamente a maneira como as empresas são operadas. Uma das principais mudanças é que a atividade principal de armazenagem na cadeia de

suprimentos não está mais confinada a manter uma grande quantidade de estoque. Em vez disso, pequenas quantidades de mercadorias são entregues imediatamente ao longo da cadeia de suprimentos, a partir de uma variedade muito ampla de SKUs (*Stock Keeping Units*) (LIU; YU; LIU, 2006). Este crescimento rápido no número de tipos de itens em estoque gerou um aumento no número de solicitações no processo de circulação (WANG; ZHANG; HE, 2009). Essas mudanças tornaram o planejamento e controle das instalações de armazenamento e dos sistemas de armazenagem ainda mais complexo (LIU; YU; LIU, 2006).

Impulsionados, em parte, pela adoção de uma série de filosofias de gestão, tal como o *Just in Time* (JIT) ou métodos da Manufatura Enxuta, o controle mais rigoroso do inventário, o menor tempo de resposta e a maior variedade de SKUs, hoje são os maiores desafios para a concepção de armazéns modernos (WANG; MCINTOSH; MILEHAM, 2010).

Além desses desafios, existe uma tendência por armazéns maiores e mais centralizados com fortes investimentos em automação. Isto se deve parcialmente às desvantagens da operação manual que resultam em armazenagem lenta, baixa velocidade de coleta, alto custo de mão-de-obra e alta frequência de erros humanos. Por outro lado, existe a crescente demanda por compras on-line, a cada dia mais e mais clientes estão usando este método para encontrar mercadorias a preços baixos, enquanto exigem uma entrega rápida. Deste modo, os varejistas necessitam de centros de distribuição mais eficientes nos quais melhoras no desempenho dos sistemas de armazenagem, através da automação, são cada vez mais almejados por essas empresas (WANG; MCINTOSH; MILEHAM, 2010). Neste contexto, o aumento da eficiência no manuseio de materiais é fundamental para o aumento da produtividade relacionada à logística e à distribuição de bens, já que uma quantidade significativa de horas de trabalho é dedicada ao manuseio de materiais e grande parte das atividades de manuseio de materiais envolve mão-de-obra (VIVALDINI, 2010).

Sendo assim, o funcionamento do armazém tornou-se uma atividade crítica na cadeia de suprimentos que requer atenção para superar a concorrência no atendimento ao cliente, cumprir prazos e reduzir custos. No entanto, os gerentes muitas vezes encontram dificuldade na formulação de um bom plano para a utilização de novos recursos tecnológicos. Isto se deve, em parte, à escassez de informações abrangentes sobre a utilização das soluções oferecidas por estes

recursos nas atividades do armazém (LIU; YU; LIU, 2006). Este fato, também, deve-se à falta de empresas ou profissionais especializados para analisar os processos produtivos e projetar soluções de automação com a certeza de que serão viáveis no contexto da aplicação.

Hoje, a maioria das pequenas e médias empresas usam empilhadeiras industriais conduzidas por operadores para o transporte de paletes. As rotas diárias dessas empilhadeiras industriais são repetitivas e, portanto, operadores fazem um trabalho monótono e maçante. Além disso, estas rotas se tornam caóticas devido à falta de otimização dos *layouts* industriais. Todos esses fatores levam a uma perda de tempo no trabalho humano, que pode ser reduzida através da introdução de *Automated Guided Vehicles* (AGVs) (WANG; ZHANG; HE, 2009). Este tipo de veículo tem sido aplicado em tarefas tais como a movimentação de produtos em armazéns ou no transporte de componentes entre diferentes estações de montagem de uma linha de produção. Usando AGVs uma fábrica pode ficar ativa durante 24 horas por dia com um mínimo de interação humana (BARBERA et al., 2003).

Os avanços tecnológicos fizeram dos robôs uma parte insubstituível no processo de fabricação moderna. Braços robóticos e AGVs tornaram-se partes integrantes de armazéns e fábricas. Eles são usados em todos os segmentos da manufatura, desde a carga e descarga de bens até a montagem do produto acabado. As vantagens de automatizar o processo de fabricação incluem uma maior produtividade, custos operacionais reduzidos, menos erros de processamento e dispensa dos trabalhadores humanos de trabalhos difíceis, tediosos e perigosos. Apesar de fábricas modernas já estarem alcançando níveis significativos de operações autônomas, os sistemas que podem operar em ambientes desestruturados, sem qualquer intervenção humana, ainda representam uma área desafiadora e excitante nas pesquisas em andamento (MIKLIC; BOGDAN; KALINOVIC, 2011).

Entre os avanços tecnológicos nas áreas de armazenagem e distribuição, destaca-se a abordagem inovadora do KIVA Systems (2012) para o manuseio de materiais. Esta solução transforma cada prateleira de armazenamento em um robô móvel, ao invés de seres humanos circulando no armazém para atender aos pedidos, as prateleiras chegam até eles. Em muitas situações, aparentemente a redução de custo favorece esta abordagem. A KIVA abriu as suas portas em 2003. Em Abril de 2008, já havia expedido 1.000 unidades de robôs móveis. Isto são mais

unidades móveis lançadas por uma única empresa do que todo o mercado mundial de AGVs de dez anos atrás (KELLY, 2009).

1.2 O Uso de AGVs na Logística

AGVs (*Automated Guided Vehicles*) são transportadores robóticos projetados para ajudar a indústria a alcançar alta produtividade com um mínimo de custo. Exemplos típicos de uso são: fábricas de automóveis, armazéns, fábricas de papel, minas e manuseio de matéria-prima em sistemas de manufatura flexível (FAZLOLLAHTABAR; ZANDIEH, 2010). Em geral, existem dois modos de navegação para AGVs, posicionamento por guias no chão ou navegação utilizando marcadores visuais colocados no ambiente, marcadores reflexivos, por exemplo. AGVs que dependem de guias instalados no chão requerem a implantação de uma determinada infraestrutura adicional e seu caminho é restrito pelas guias, como um trem sobre trilhos. AGVs que utilizam marcadores reflexivos para navegar também têm o inconveniente de exigir uma infraestrutura adicional, entretanto podem modificar o seu trajeto para o desvio de obstáculos (BOUGUERRA et al., 2009).

Os AGVs reduzem os custos de produção e aumentam a eficiência de um sistema de produção (FAZLOLLAHTABAR; MAHDAVI-AMIRI, 2010) e também podem ser utilizados como alternativa para o manuseio de materiais em ambientes hostis para a atividade humana, tal como locais sujeitos a ruídos intensos ou ambientes com temperaturas extremas. A capacidade de incorporar autonomia, operação em alta velocidade, confiabilidade e precisão tornam a robótica móvel uma alternativa atraente se comparada aos processos manuais tradicionais com situações altamente repetitivas ou pouco amigáveis aos seres humanos (VIVALDINI, 2010) (WANG; MCINTOSH; MILEHAM, 2010).

Em sistemas produtivos AGVs são comumente utilizados para fins de transporte com o objetivo de reduzir custos, aumentar a capacidade produtiva, minimizar os danos e em procedimentos estruturados reduzir o número de erros (DAM et al., 2004). Estes erros de armazenagem podem gerar divergências de inventário e até gerar perdas materiais, pois o objeto armazenado, considerado como extraviado, pode tornar-se obsoleto ou perder seu prazo de validade,

resultando em prejuízos e em um aumento nos custos administrativos por conta da reposição de estoque dos produtos considerados extraviados.

Os primeiros AGVs eram dependentes de cabos elétricos enterrados ou linhas refletivas no piso para a sua navegação e tornavam cara e complicada a adaptação dos processos de produção ou armazenagem (DAM et al., 2004) (VAN DER MOLEN; GEERTS, 1991). Logo, para conseguir um aumento na flexibilidade da instalação e operação de um sistema de AGVs tornou-se necessária a utilização de outros meios de navegação. O uso de *landmarks* associado com a odometria trouxe uma melhor perspectiva em termos de custo de implantação (VAN DER MOLEN; GEERTS, 1991), entretanto a capacidade do computador embarcado limitava a complexidade dos programas de navegação. Como AGVs geralmente operam em ambientes fabris, o espaço para manobras é reduzido em função da otimização do *layout* (VAN DER MOLEN; GEERTS, 1991), isto resulta em uma condição crítica para a navegação, exigindo algoritmos de navegação mais sofisticados.

Como previsto por Barbera et al. (2003), os avanços tecnológicos e a redução dos custos na microeletrônica e na microinformática resultaram em uma nova geração com AGVs mais inteligentes, capazes de armazenar instruções sobre rotas e de tomar decisões. Por outro lado, o surgimento de sistemas de localização sem fio, utilizando sensores de medição laser, por exemplo, facilitaram a implantação dessas funcionalidades.

Entre todos os sistemas de manuseio de materiais utilizados em fábricas modernas, aqueles baseados em AGVs têm, por muitas razões, sido encontrados com grande sucesso nas últimas décadas. Sistemas que utilizam AGVs são muito mais flexíveis do que os sistemas tradicionais, tais como transportadores industriais fixos, permitindo uma reconfiguração dinâmica dos caminhos de orientação de acordo com as necessidades de mudança no transporte da fábrica. Além disso, os algoritmos de Inteligência Artificial e o desempenho computacional melhoraram dramaticamente nas duas últimas décadas, tornando-se possível construir veículos muito mais inteligentes em termos de desvio de obstáculos, detecção de colisão e com sistemas de visão a preços mais baixos. Por outro lado, o uso crescente de AGVs criou uma necessidade de resolver problemas complexos de gestão, principalmente relacionados à concepção de caminhos eficientes para os veículos e à gestão de tráfego de AGVs (CARICATO; GRIECO, 2005).

Tradicionalmente, o funcionamento confiável de AGVs depende da disponibilidade de uma infraestrutura adequada para apoiar a navegação (BOUGUERRA et al., 2009). Basicamente, as questões mais relevantes nas instalações que utilizam AGVs podem ser classificadas nas seguintes categorias principais: estratégias de navegação, estimativa do número de veículos, escalonamento das tarefas, posicionamento dos veículos em modo de espera, gerenciamento das baterias, roteamento e resolução de conflitos (LE-ANH; KOSTER, 2004). E no cenário industrial, a viabilidade da instalação e manutenção destes sistemas.

A automação da logística, bem como a gestão de armazéns, por exemplo, exige a localização de baixo custo de vários objetos com uma precisão de poucos centímetros, particularmente, em ambientes internos (SCHNEIDER; SALOMON, 2009). A localização destes objetos, tais como paletes, é controlada por sistemas informatizados. O reconhecimento desses volumes pode ser fornecido por ferramentas que utilizam códigos de barra ou RFID (WANG; MCINTOSH; MILEHAM, 2010) para identificar estes objetos. Códigos de barras têm sido amplamente utilizados em sistemas de gestão de armazém. No entanto, sua capacidade de leitura de curto alcance ainda limita a sua aplicação na identificação do item transportado e no posicionamento de equipamentos no armazém (WANG; MCINTOSH; MILEHAM, 2010).

Entretanto, estes recursos conhecidos nem sempre são aplicáveis como sistemas de referência na navegação de um robô móvel devido à necessidade de aproximação entre o objeto identificado e o sensor, que resulta em uma limitação de distância, conseqüentemente restringindo os movimentos do AGV, ou, ainda, na exigência de um número inviável de marcadores no ambiente, tornando o sistema custoso. Além disso, nem sempre as funcionalidades dos sensores de código de barras fornecem uma informação sobre a distância relativa entre o robô móvel e o objeto identificado com uma precisão conhecida, condição necessária para utilização do sensor em um sistema de auto-localização. No entanto, sensores de código de barras e de RFID ainda ocupam um posição de destaque na automação de armazéns, pois oferecem soluções de automatização economicamente atrativas, principalmente no campo de *Automated Storage & Retrieval System* (AS/RS).

O AS/RS é um sistema de armazenagem e coleta controlado por computador integrado em um processo de fabricação e distribuição, que se tornou

um importante e insubstituível recurso da tecnologia no desenvolvimento da logística moderna. O AS/RS resulta da convergência de equipamentos e controles que manuseia, armazena e coleta materiais com precisão, exatidão e velocidade através de técnicas de automação (WANG; ZHANG; HE, 2009).

Muitos sistemas de AGVs modernos não utilizam guias fixos. As rotas podem, por exemplo, ser programadas por um computador e enviadas aos controladores dos veículos. Estes veículos possuem uma movimentação livre e são orientados por sistemas baseados em técnicas que utilizam odometria, sensores de medição laser, visão computacional, sensores de ultrassom, giroscópios ou rádio frequência. Com a finalidade de fazer pleno uso da capacidade de flexibilidade de tais sistemas, são necessários AGVs dotados de sistemas inteligentes (LE-ANH; KOSTER, 2004). Para este tipo de aplicação, no entanto, uma série de requisitos no desenvolvimento de sistemas AGVs tem despertado o interesse de pesquisadores na área de robótica móvel, onde um fator essencial de um sistema robótico é torná-lo autônomo.

1.3 A Navegação de um AGV Inteligente

AGVs devem ter a capacidade de operar sem falhas, reagir às mudanças do ambiente por meio de sensores e ser flexíveis. Em termos de autonomia, destaca-se o problema da navegação, que é um dos principais sistemas para a sua locomoção segura no ambiente. O AGV parte de uma posição inicial e segue até uma posição desejada sem que ocorram colisões, para isso deve ser capaz de desviar de obstáculos fixos ou móveis no ambiente. Para que o sistema de navegação possa executar esta tarefa com êxito, o AGV precisa de um sistema de auto-localização eficiente e preciso.

Basicamente, o problema de sistemas de navegação autônomos ou semiautônomos envolve o reconhecimento do ambiente, auto-localização, planejamento de trajetória e controle dos movimentos do sistema no ambiente (MIRANDA Neto et al., 2009). Em geral, sistemas de navegação autônomos têm como necessidade o fornecimento da localização e a disponibilidade de um mapa para o planejamento de trajetória.

1.4 Projetos de AGVs

Ao nível da teoria do escalonamento de sistemas, procura-se migrar dos sistemas de controle habitualmente centralizados para sistemas de controle distribuídos mais robustos e eficientes, que assentam na exploração de mecanismos de cooperação e coordenação entre os vários robôs móveis da mesma frota, reforçando assim a autonomia e o nível de inteligência do controlador residente em cada AGV. O projeto mecânico dos AGVs e os métodos de orientação deles são normalmente estudados pelos engenheiros mecânicos e elétricos, e hoje em dia pelo surgimento de um novo ramo da engenharia chamada mecatrônica (BISHOP, 2002) (LENGERKE, 2010).

AGVs podem ser projetados a partir de uma empilhadeira comum adicionando um sistema composto de sensores, atuadores, controladores e sistemas embarcados adaptados para conseguir a navegação autônoma. Esta configuração tem como vantagem o aproveitamento do projeto mecânico em operação em diversos armazéns, que foi desenvolvido e testado no passado. Desse modo, as características e limitações do projeto mecânico são conhecidas, tal como a capacidade de carga. Isto oferece uma maior confiabilidade no quesito segurança e velocidade ao projeto, pois se restringe ao desenvolvimento do sistema mecatrônico adaptado. Outra vantagem é obtida com a redução do investimento inicial para a implantação do sistema de AGVs em ambientes que já dispõem de uma frota de empilhadeiras em operação. Um exemplo de projeto de AGV a partir de uma empilhadeira adaptada pode ser visto na Figura 1.1 de Bouguerra et al. (2009), este AGV utiliza no seu sistema de controle os drivers do Player/Stage nos módulos de percepção e navegação.

Figura 1.1: AGV de empilhadeira adaptada.

Por outro lado AGVs têm sido projetados por empresas especializadas e tornaram-se viáveis devido à popularização do uso em armazéns. Os projetos tornaram as empilhadeiras robóticas mais enxutas, pois o AGV não precisa dispor de uma cabine para acomodar um operador, reduzindo o espaço ocupado pelo AGV no armazém e diminuindo os custos de construção, que também foram reduzidos pela produção em escala. Um AGV provido de garfo para executar a função de uma empilhadeira pode ser visto na Figura 1.2.

Figura 1.2: AGV – Automated Guided Vehicle AGVRobots.com (2011).

Abordagens alternativas de AGVs para a movimentação de cargas dentro do armazém resultaram no projeto de robôs móveis com funções especializadas, tal

como a abordagem da KIVA Systems (2012), onde o projeto mecatrônico do robô móvel (Fig. 1.3) tem como finalidade transladar prateleiras contendo itens pequenos em vez de paletes. Ou ainda, AGVs projetados para transportar itens específicos em uma linha de produção, como o AGV mostrado na Figura 1.4 (AUTOMOTIVEBUSINESS, 2012) utilizado na linha de montagem da Mercedes-Benz de Juiz de Fora, MG.

Figura 1.3: KIVA Systems.

Figura 1.4: Fábrica Mercedes-Benz de Juiz de Fora, MG.

1.5 O Projeto: Armazém Inteligente

Embora existam diversas soluções de AGVs para uso em armazéns, a implementação destas soluções apresentam desafios comuns, tais como: o roteamento, o gerenciamento das baterias, a navegação suportada por um sistema de auto-localização, entre outros. Neste contexto, o Laboratório de Robótica Móvel (LabRom) do Grupo de Mecatrônica da EESC/USP, vem desenvolvendo o projeto do Armazém Inteligente, tendo como objetivo o estudo e pesquisa destes desafios.

O gerenciamento das baterias foi objeto de estudo no trabalhos de Oliveira et.al (2011) no qual a estimativa de carga da bateria permite gerenciar o número de tarefas que podem a ser atribuídas a uma empilhadeira robótica antes da necessidade de parada para recarga ou troca de baterias. O desenvolvimento de um sistema de roteamento automático baseado no algoritmo de Dijkstra, capaz de gerar rotas livres de conflitos e com trajeto otimizado, foi o tema da dissertação de mestrado de Vivaldini (2010). Nesse trabalho, a rota gerada para cada empilhadeira robótica é representada por uma sequência de números pertencentes a um grafo, onde cada número é associado a um local de um mapa preestabelecido.

Rotinas de navegação, que interpretam a rota gerada pelo sistema de Vivaldini (2010) e executam o planejamento de trajetória entre os locais representados pelos pontos de interligação de um grafo, foram desenvolvidas por Pasqual (2010). Duas rotinas foram implementadas: uma baseada no algoritmo A* e outra em campos potenciais. Como citado anteriormente, a navegação na robótica móvel é dependente do fornecimento da localização e inicialmente as rotinas foram implementadas tendo como referencia a localização fornecida pelo Player/Stage, software utilizado na simulação do armazém inteligente, a qual é baseada puramente em odometria e livre de erros. Visando uma implementação das empilhadeiras robóticas em ambiente real, no qual a localização está sujeita a erros e normalmente é estimada pela fusão sensorial de mais de um tipo sensor, foi proposto o desenvolvimento de um sistema de auto-localização e no contexto do projeto do armazém inteligente foi desenvolvido o sistema descrito no objetivo a seguir.

1.6 Objetivo

Esta dissertação tem como objetivo o estudo dos métodos de localização existentes na robótica móvel, especialmente aqueles utilizados em aplicações envolvendo AGVs. O foco do trabalho é o desenvolvimento de um Sistema de Auto-localização baseado no Filtro de Kalman Estendido, em inglês *Extended Kalman Filter* (EKF), um algoritmo probabilístico aplicado na auto-localização de um AGV em um ambiente *indoor* semelhante a um armazém inteligente a partir de dados

sensoriais. Neste trabalho, o AGV corresponde a uma empilhadeira robótica utilizada em armazéns inteligentes para a manipulação de cargas. O algoritmo utiliza os dados da odometria e de um sensor de medição laser embarcados na empilhadeira robótica para obter uma estimativa da localização do AGV. O algoritmo desenvolvido deve ser capaz de fornecer uma localização confiável, necessária para um planejamento de trajetória eficiente, tornando possível a execução de uma rota pré-estabelecida. Diversas variações e métodos de localização foram avaliados para atingir esse objetivo.

1.7 Organização da Dissertação

Neste capítulo foram apresentadas a motivação no contexto econômico, tecnológico e logístico e uma introdução sobre a relevância do uso de AGVs na logística, bem como a necessidade da auto-localização dos AGVs. O Capítulo 2 apresenta uma revisão bibliográfica sobre as principais técnicas de localização existentes para AGVs que navegam em armazéns e sobre os elementos envolvidos na auto-localização de robôs móveis. No Capítulo 3 são apresentados os algoritmos de localização de robôs móveis, sua descrição da modelagem matemática e suas características.

No Capítulo 4 são apresentadas as ferramentas de *software* utilizadas para a implementação dos algoritmos de auto-localização no ambiente simulado, o modelo do armazém utilizado nas simulações e as hipóteses adotadas. Também é apresentado o procedimento utilizado na construção do mapa do ambiente real *indoor* e a coleta de dados sensoriais reais para o teste do sistema de auto-localização.

No Capítulo 5 são apresentados os resultados obtidos durante a simulação do cenário de um armazém inteligente utilizando o *software* Player/Stage, no qual o trajeto da empilhadeira robótica dentro do cenário é o resultado de uma rota gerada por um sistema de roteamento e representa uma operação de carga e descarga. O objetivo é recriar situações nas quais poderia ocorrer a perda da localização e consequentemente desvio da rota planejada. Também são apresentados os resultados obtidos no teste com dados reais, capturados do ambiente utilizando um

robô móvel Pioneer 3-AT e o Player/Stage para a leitura e gravação dos dados sensoriais.

No Capítulo 6 são apresentadas as conclusões finais e as propostas para trabalhos de pesquisa futuros.

2 A LOCALIZAÇÃO DE ROBÔS MÓVEIS EM AMBIENTES *INDOOR*

Neste capítulo é apresentada uma revisão da bibliografia sobre as técnicas de localização utilizadas em robôs móveis em ambientes *indoor*, e como alguns dos conceitos dessas técnicas foram adaptados para o uso em armazéns inteligentes. Esta revisão tem como finalidade apresentar os pontos fortes e fracos de cada estratégia de localização, e como estes são refletidos na instalação e na operação de um sistema de AGVs dentro de um armazém.

2.1 O problema da Localização na Robótica Móvel

A tarefa de localização na robótica móvel é basicamente o problema de encontrar as coordenadas e a orientação do robô com relação ao ambiente assumindo que é fornecido um mapa do ambiente (THRUN; BURGARD; FOX, 2005). Tarefas de alto nível como planejamento de trajetória, navegação e exploração dependem de um sistema de auto-localização eficiente e preciso (EINSELE, 1997). Em geral, técnicas de fusão sensorial são empregadas para isso. E para obter uma estimativa atualizada da posição do robô móvel em movimento, o processo de fusão das informações e dos erros sensoriais deve ser executado em tempo real.

Por outro ponto de vista deve-se considerar que nem sempre se faz necessário saber o local geométrico exato do robô para conseguir uma localização eficiente. Por exemplo: quando se viaja ao longo de um corredor que não contenha locais distintos, a posição exata do robô dentro do corredor não é relevante. Nessa situação o importante é que o robô saiba para qual direção ele deve ir e reconheça o final do corredor, a fim de se relocalizar com exatidão novamente (EINSELE, 1997) (KUIPERS; BYUN, 1988) (MATARIC, 1990).

Outra questão importante na auto-localização de robôs móveis é o desenvolvimento de um método eficiente para a estimativa da localização sem precisar modificar o ambiente, bem como o fornecimento e a criação de um mapa do

ambiente. Para isso existem diversas soluções, que utilizam uma variedade de sensores (CASTELLANOS; TARDÓS 1996).

Embora modificar o ambiente não seja o ideal, pois torna o processo de auto-localização dependente e menos flexível, marcas que são reconhecidas por sensores embarcados são inseridas no ambiente para auxiliar na tarefa de auto-localização. Estas marcas de referência na robótica móvel são chamadas de *landmarks*. Tem-se que a auto-localização não ocorre sem o fornecimento de um mapa. Logo, a necessidade de construir um mapa é uma tarefa inevitável. A construção manual de um mapa é demorada e geralmente não é considerada uma solução prática para robôs que precisam operar em ambientes diferentes ou externos (DIOSI; KLEEMAN, 2004).

Já em ambientes internos, que é o caso dos armazéns, o método de construção do mapa do ambiente irá influenciar na flexibilidade e na escalabilidade do sistema de AGVs, ou seja, quanto menos complexo for o processo de construção ou atualização do mapa do ambiente, mais flexível será o sistema de AGVs implantado no armazém.

Sem uma capacidade de localização específica, um robô móvel baseia-se apenas na odometria, ou seja, em seu modelo cinemático para atualizar sua estimativa de localização, processo também conhecido como *dead reckoning*. No entanto, tal modo de operação resulta invariavelmente no acúmulo de grandes erros por conta da derrapagem, o desalinhamento das rodas e outras condições não ideais, e, eventualmente, o robô acaba ‘perdendo-se’ depois de um percurso suficientemente longo. Sendo assim, uma localização eficiente exige que o robô explore a informação proveniente de seus sensores exteroceptivos tal como medidores de distância e sistemas de visão. (FABRIZI et al., 1998).

2.2 Técnicas de Localização de AGVs

A Figura 2.1 abaixo, disponível em NCD8 (2012), ilustra seis das principais técnicas de auto-localização empregadas para a navegação de empilhadeiras robóticas oferecidas por empresas que atuam na área de automação de armazéns. A saber: o uso de sensor laser e fitas reflexivas que são posicionadas no ambiente;

o emprego de marcadores no chão do armazém; a combinação de câmera e marcadores posicionados nas paredes ou no teto do armazém; o uso de sensor magnético combinado com fitas magnéticas posicionados no solo do armazém; o emprego de sensores laser tipo *range finder*; e o emprego de sensor e cabo indutivo posicionado no solo do armazém.

Figura 2.1: Ilustração das técnicas mais usualmente encontradas em soluções oferecidas por empresas que atuam na automação de empilhadeiras em armazéns.

Na técnica de auto-localização baseada em sensor laser e fitas reflexivas, marcadores refletivos são inseridos no armazém a uma altura pré-fixada e são reconhecidos por um sensor de medição laser especial que retorna a posição relativa de cada marcador refletivo reconhecido pelo sensor em uma determinada área do armazém, dentro do seu perímetro de alcance. O sensor de medição laser NAV300 da Figura 2.2 (NAV300, 2012), por exemplo, pode medir um máximo de 32 refletores visíveis a cada instante e determinar as coordenadas relativas de cada refletor com relação à sua própria posição (SICK, 2008, 2012). A escalabilidade desta técnica é limitada pelo número máximo de marcadores mapeados pelo sistema de auto-localização, o qual reflete na autonomia do AGV, pois irá restringir o

espaço de movimentação do AGV em função do mapa de marcadores, que pode ser relevante na automatização de armazéns de grande porte.

Figura 2.2: Sensor SICK NAV300-2232, utilizado na navegação com marcadores refletivos.

Além do número máximo de marcadores artificiais mapeáveis, outro fator limitante desta técnica é a dependência do *layout* do armazém para o posicionamento dos marcadores refletivos. Armazéns que utilizam um arranjo não estruturado têm um número restrito de locais aptos a receber marcações. Um exemplo deste tipo de arranjo pode ser visualizado na Figura 2.3, extraída de Bouguerra et al. (2009), o armazém de bobinas de papel, pelo fato de ser composto por um galpão para facilitar a armazenagem e movimentação dos materiais, não possui um número suficiente de objetos de posição fixa, tais como: prateleiras, colunas ou pilares, nos quais poderiam ser fixados marcadores refletivos. Além disto, a altura das pilhas de papel obstrui a visão destes elementos no ambiente, nesta configuração a criação de um sistema de localização tradicional baseado em refletores torna-se impraticável. Em parte, este obstáculo é superado pelo uso de marcadores no teto.

Figura 2.3: Armazém de Bobinas de Papel.

A nova geração de sensores de medição laser para a navegação de AGVs, tal como o modelo NAV300 da SICK, incorporam algumas funcionalidades como a detecção de contornos espaciais extraídos do ambiente, com o retorno de suas coordenadas relativas com relação à posição do sensor. Este recurso serve como alternativa de localização em locais, onde os marcadores refletivos não são visíveis pelo sensor de medição laser. No contexto de armazéns, um exemplo de auto-localização de AGVs utilizando a detecção de contornos em locais com visibilidade limitada, é o interior de uma carreta, conforme mostra a Figura 2.4 de SICK (2012).

Figura 2.4: Sensor de medição laser SICK NAV300 embarcado no AGV que navega no interior de uma carreta.

Técnicas alternativas baseadas em sensores sem fio, tal como o LSB (*Localization by Superposing-Beats*) (SCHNEIDER; SALOMON, 2009), são objeto de estudo na atualidade. Este procedimento emprega transmissores de rádio frequência que emitem pulsos particularmente modulados. A superposição dos sinais especialmente configurados gera padrões de interferência que dependem da localização. Estes sinais modulados, traduzidos em formas de onda de baixa frequência, são analisados por um receptor para obter a posição relativa.

2.3 Classificação das Técnicas de Localização

A determinação confiável da localização é uma exigência fundamental se tratando de robôs móveis. Segundo Jetto, Longhi e Venturini (1999) existem dois tipos de localização: relativa e absoluta. A localização relativa é obtida através de sensores que medem as variáveis internas alteradas pelo movimento relativo do robô móvel no ambiente. Um exemplo típico de sensor interno é o *encoder* incremental fixado diretamente no eixo das rodas. Nesta montagem o número de incrementos, fornecidos pelo *encoder* a cada instante, é utilizado para estimar a localização em determinado intervalo de tempo. O grande problema deste método é o acúmulo de erro que degrada a estimativa de localização ao longo do tempo.

A localização absoluta é realizada utilizando sensores que fornecem medidas externas ao robô sobre o ambiente no qual se encontra. Um exemplo comum é a fixação de um conjunto de sensores para medir a distância que o robô móvel está em relação a certos objetos estáticos no ambiente. Se a localização desses objetos é conhecida torna-se possível, então, o cálculo da localização absoluta do robô móvel no ambiente. Esta necessidade de conhecimento a respeito da localização dos objetos representa uma deficiência, pois caso o robô seja inserido em um ambiente desconhecido, será necessário um novo levantamento sobre a localização dos objetos de referência. Isto não ocorre em sistemas onde a localização dos objetos de referência é obtida pelo próprio sistema, que é o caso dos satélites do sistema GPS.

Na robótica móvel objetos de referência encontrados no ambiente são denominados *landmarks*, os quais são divididos em artificiais ou naturais e sua utilização é amplamente estudada. Um *landmark* pode ser definido como uma característica ou marco distinto do meio em que um robô está inserido, cuja principal característica é poder ser reconhecido a partir de dados coletados pelos sensores (BETKE; GURVITS, 1994).

Landmarks artificiais são objetos ou marcos incorporados ao ambiente com o intuito de fornecer a captura de dados necessária referente à sua localização no ambiente. Alguns exemplos de *landmarks* artificiais são: figuras geométricas, texturas, códigos de barra, objetos tridimensionais, refletores, dentre uma infinidade de outros tipos (BORENSTEIN; EVERETT; FENG, 1996). Muitas das aplicações que

utilizam este tipo de *landmark* requerem o uso de um sistema de visão computacional.

Landmarks naturais também são conhecidos como *features*. Na definição de Siegwart e Nourbakhsh (2004, p. 151) *features* são estruturas reconhecíveis de elementos no ambiente, que geralmente podem ser extraídas a partir de medições sensoriais e ser descritas por um modelo matemático. São divididas em *low-level features* (primitivas geométricas) e *high-level features* (objetos).

As *features* exercem um papel importante na robótica móvel, especialmente na modelagem do ambiente, pois permitem uma descrição mais compacta e robusta do ambiente, auxiliando o robô móvel durante as tarefas de mapeamento e localização (SIEGWART; NOURBAKHS, 2004). Entre as diversas primitivas geométricas existentes, o segmento de linha é a mais simples, resultando ser fácil descrever a maioria dos ambientes internos utilizando esta primitiva geométrica (NGUYEN et al., 2005). Por outro lado, Diaz (2010, p. 20) complementa: “Em um ambiente interno, uma grande proporção das estruturas presentes pode ser representada a partir de segmentos de linhas, é por isto que esta primitiva geométrica é uma das mais utilizadas nos mapas baseados em *features*”.

Existem diversas características (*features*) que podem ser extraídas do ambiente, entre elas: linhas, *corners*, *virtual corners* (LARA; ROMERO, 2007). Segmentos de círculo são menos robustos do que outros tipos de características extraídas do ambiente, por isso são menos usados. Uma forma de avaliar o desempenho de um algoritmo de extração de *features* é o número de vezes que cada *feature* é detectado, sob o número de vezes que o *feature* é visível.

Sendo assim, em função da simplicidade proporcionada pelo segmento de linha e do ambiente escolhido para o teste do Sistema de Auto-localização corresponder a um ambiente interno, escolheu-se o segmento de linha como primitiva geométrica a ser utilizado neste trabalho. Em geral, *features* são extraídas do ambiente utilizando sensor de medição laser ou sonar e, em seguida, são comparadas com características previamente gravadas. Para este trabalho utilizou-se um sensor de medição laser devido ao seu alcance (*range*) maior em relação ao sonar.

A extração de *features* utilizando um sensor de medição laser é feita em duas fases: (1) detecção de pontos de quebra (*breakpoints*), onde ocorre uma divergência entre duas figuras geométricas e o descarte de pontos isolados e (2)

segmentação, que consiste em verificar quais pontos pertencem à mesma linha ou característica geométrica do ambiente (BORGES; ALDON, 2004).

Borges e Aldon (2004) propõem o uso de uma função de curvatura para a obtenção de *breakpoints*. A posição média é estimada com base em duas linhas que geram um *corner*, já que a varredura do laser não coincide exatamente com o ponto de cruzamento das linhas. Castellanos e Tardós (1996) também dividem a extração de linhas em duas fases, primeiramente encontram-se as regiões homogêneas formadas por polígonos e depois é feita a separação das linhas do polígono. Castellanos e Tardós (1996) também propõem o uso de um modelo para estimar a incerteza dos pontos lidos do sensor laser e dos segmentos extraídos.

Uma etapa comum às técnicas de extração é o processo de segmentação, que é necessário para obter as informações geométricas a partir do conjunto de pontos do sensor de medição laser. De acordo com o princípio como os dados são tratados, o processo de segmentação pode ser classificado em dois métodos principais: o processo sucessivo e o processo total. Durante o processo sucessivo os pontos medidos são tratados um a um, enquanto que no processo total o conjunto de pontos medidos é tratado como um todo (SIADAT et al., 1997).

Ainda, para a obtenção de segmentos de linha na execução do processo de segmentação utilizam-se algoritmos de extração de linhas que podem ser classificados também segundo a técnica de segmentação utilizada em: *Edge Following* ou *Line Fitting* (SIADAT et al., 1997). Como exemplo de algoritmo de extração de linhas que utiliza o processo de segmentação sucessivo tem-se o *Line Tracking* (LT), seu funcionamento em etapas resumidas é: (1) dada uma sequência de pontos, cria-se uma linha a partir dos dois primeiros pontos e seus parâmetros são calculados; (2) o 3º ponto é adicionado à linha se atende ao critério de distância máxima, caso não uma nova linha é criada a partir deste ponto; (4) o algoritmo termina quando o número total de pontos é alcançado. Tem como vantagem ser muito rápido e como desvantagem a tendência de adicionar pontos de superfícies curvilíneas como sendo pontos pertencentes a uma reta. Como exemplo do processo de segmentação total pode-se citar o algoritmo *Split-and-Merge*, que foi utilizado na parte experimental deste trabalho e será descrito em detalhes no Capítulo 3.

2.4 O Mapeamento de Ambientes

O mapeamento de ambientes em aplicações robóticas pode ser dividido em duas abordagens principais: o mapeamento métrico e o mapeamento topológico. Mapas métricos capturam as propriedades geométricas do meio ambiente, enquanto que os mapas topológicos descrevem a conectividade entre diferentes locais (THRUN, 2002a). Além desses, temos o *feature-based map*, que corresponde ao mapa de objetos ou *features* do ambiente com localização conhecida, neste caso o mapa corresponde ao registro da posição e orientação desses objetos com relação a um sistema de coordenadas global utilizado pelo robô móvel durante a navegação. A tarefa de mapeamento não é o foco deste trabalho, por esse motivo não serão apresentados os algoritmos utilizados neste processo. Durante a fase experimental descrita no capítulo 4 será utilizado o *feature-based map*, construído manualmente, tanto para a simulação como para o teste experimental em ambiente real.

As representações dos mapas baseados em *features* muitas vezes se limitam a descrever ambientes que possam ser expressos mediante formas geométricas simples, isto resulta em uma grande desvantagem, que pode dificultar seu uso em ambientes externos pouco estruturados (DIAZ, 2010).

2.5 O Software de Simulação Player/Stage

O projeto Player/Stage teve início em 1999 no USC *Robotics Research Lab* com o intuito de resolver uma necessidade interna na simulação de Sistemas Multi-Robôs (*Multi-Robot Systems* – MRS). Desde então, tem sido adotado, modificado e ampliado por pesquisadores ao redor do mundo (GERKEY; VAUGHAN; HOWARD, 2003). O projeto é composto pelo Player que funciona como um servidor de dados de dispositivos e o Stage, um simulador gráfico 2D de múltiplos robôs. Além destes, existe um conjunto de ferramentas auxiliares, diversas bibliotecas e *drivers* de sensores comerciais que acompanham o pacote de instalação.

Quando executado em um robô móvel, o Player apresenta uma interface de comunicação simples com os sensores e atuadores. O programa de controle de

clientes, através de uma interface de rede com protocolo TCP/IP, comunica-se com o Player, o qual configura os dispositivos, faz a leitura dos dados sensoriais e envia os comandos para os atuadores. O Player suporta uma grande variedade de *hardware* e comporta a programação de sofisticados sistemas de sensoriamento e de algoritmos de controle, tais como um detector de *landmarks* e a localização probabilística.

O Stage proporciona um ambiente simulado 2D contendo uma população de robôs e sensores simulados operando no mapa de um ambiente obtido a partir de um desenho *bitmap*. O Stage procura ser eficiente e configurável em vez de altamente preciso, dessa forma pode simular dezenas ou centenas de robôs num único computador. Os dispositivos simulados são acessados através do Player como se eles fossem um hardware real, fazendo com que os controladores operem de forma semelhante em robôs simulados ou reais. Lançado como *software* livre sob licença GNU, o Player/Stage funciona preferencialmente em Linux Ubuntu, mas pode ser configurado em outras plataformas Linux. A Figura 2.5 mostra o Player/Stage sendo executado em uma distribuição Linux Ubuntu.

Figura 2.5: Player/Stage no Linux Ubuntu.

3 ALGORITMOS DO SISTEMA DE AUTO-LOCALIZAÇÃO

Neste capítulo serão apresentados os algoritmos mais conhecidos e utilizados na robótica móvel para a implementação da função de auto-localização, com ênfase nos algoritmos escolhidos para o sistema de auto-localização desenvolvido, bem como a formulação matemática destes.

O algoritmo escolhido para realizar a fusão sensorial foi o Filtro de Kalman Estendido, em inglês *Extended Kalman Filter* (EKF), bastante utilizado devido à sua robustez e com diversas aplicações em outras áreas além da robótica móvel. Para o teste do sistema de auto-localização com dados reais do ambiente foi escolhido o *Iterative End Point Fit* (IEPF) na extração de *features* do ambiente. Também serão descritos os algoritmos da Localização de Markov e da Localização de Monte Carlo, devido à popularidade na tarefa de auto-localização da robótica móvel.

Os algoritmos de mapeamento utilizados na robótica móvel não serão descritos neste trabalho, pois o mapa do ambiente é fornecido para o sistema de auto-localização e foi construído a partir de um desenho de planta baixa dispensando o uso de algoritmos tanto na simulação como no teste com dados reais.

3.1 A Localização de Markov

A Localização de Markov é uma abordagem baseada em um modelo probabilístico, que é suficientemente expressivo para representar situações em que localização do robô móvel não pode ser determinada com unicidade. Sendo assim, a Localização de Markov permite que o robô móvel se localize, mesmo sob um contexto de incerteza global (FOX, 1998). A Localização de Markov aborda o problema de localização global, o problema de rastreamento da posição e o problema do robô sequestrado em ambientes estáticos (THRUN; BURGARD; FOX, 2005).

Segundo Fox (1998), o paradigma de localização, denominado Localização de Markov, foi proposto inicialmente por diversos pesquisadores, tais como: Nourbakhsh, Powers e Birchfield (1995), Simmons e Koenig (1995), Cassandra, Kaelbling e Kurien (1996) e Burgard et al. (1996). O paradigma da Localização de Markov é baseado em um modelo geral para a estimação de estado e aplica uma abordagem probabilística para representar a localização, o resultado das ações de movimento e as observações do robô (FOX, 1998).

A Localização de Markov recebe esse nome, pois adota a hipótese de Markov, na qual a localização do robô móvel depende apenas da informação de localização anterior e das observações atuais, portanto independente com relação às observações anteriores (ODAKURA, 2007). A hipótese de Markov é ilustrada na Figura 3.1, extraída de PROBABILISTIC ROBOTICS (2005), onde x_t representa a posição do robô móvel, u_t as ações de controle e z_t os dados sensoriais no instante t .

Figura 3.1: Hipótese de Markov.

O robô móvel DERVISH de Nourbakhsh, Powers e Birchfield (1995), utilizava um mapa topológico para a navegação em um modelo de escritório artificial. O sistema de navegação do DERVISH mantinha a informação de posição sob a forma de um conjunto de estados contendo todos os estados possíveis para a localização do robô móvel. Este conjunto de estados é, então, incrementado ou atualizado sempre que o robô recebia uma nova informação sensorial. Durante a navegação um sistema de hipóteses foi utilizado para o planejamento de trajetória, utilizando esse sistema o robô móvel assumia que estava localizado no estado mais provável do conjunto de estados.

Simmons e Koenig (1995) desenvolveram uma técnica de navegação que utiliza os modelos de Markov para rastrear de forma robusta a posição do robô móvel e direcionar o seu curso. Nesta técnica, um modelo baseado no *framework* POMDP (*Partially Observable Markov Decision Process*) foi construído a partir da

informação topológica sobre a conectividade do ambiente e da informação aproximada de distância, acrescida das características dos sensores e atuadores. Dessa maneira, o modelo de Markov estima a posição do robô sob a forma de distribuição de probabilidade. As probabilidades são então atualizadas quando o robô informa um deslocamento ou giro, e quando observa características tais como paredes e cruzamentos de corredores (SIMMONS; KOENIG, 1995). Segundo Fox (1998) a origem do termo Localização de Markov vem da aplicação direta do *framework* POMDP proposto por Lovejoy (1991) e White (1991) para a estimação de estados.

Os trabalhos de Nourbakhsh, Powers e Birchfield (1995), Simmons e Koenig (1995) e Cassandra, Kaelbling e Kurien (1996) utilizaram a Localização de Markov associada a um mapa topológico para a navegação em ambientes de escritórios. A utilização de grids para a distribuição da probabilidade das diversas posições no ambiente foi introduzida por Burgard et al. (1996), essa abordagem incorporou a representação métrica do ambiente para estimar a localização. Esta técnica foi utilizada por Fox (1998), Kröse e Bunschoten (1999) e Fox, Burgard e Thrun (1999).

A Localização de Markov é uma técnica probabilística que mantém uma distribuição de probabilidade sobre o espaço de todas as hipóteses de onde o robô poderia estar ao invés de manter uma única hipótese sobre a sua localização. Esta técnica utiliza dois modelos probabilísticos para estimar a localização, o modelo de movimento e o modelo de observação. O modelo de movimento estima a localização atual do robô em função da leitura dos sensores internos e da localização anterior e o modelo de observação descreve a relação entre a observação e a localização estimada (ODAKURA, 2007).

Na Localização de Markov a localização do robô é estimada com base em uma crença (em inglês, *belief*). Pela notação básica de Fox (1998), a crença $Bel(L_t)$ representa uma distribuição de probabilidade sobre todo o espaço de L (RIBEIRO; LIMA, 2002), onde L_t representa a variável aleatória que expressa o estado da localização do robô no instante t . A probabilidade do robô estar localizado no local l no instante t será dada por $Bel(L_t=l)$. Na localização 2D, l é um local que pode ser definido pelos parâmetros (x, y, θ) .

Aplicando a regra de Bayes a crença sobre o robô estar localizado no local l no instante t , dado um conjunto de medidas sensoriais, será fornecida pela equação

1, onde, a_i corresponde aos movimentos percebidos pelos sensores internos e s_i corresponde ao conjunto de observações obtidas dos sensores externos.

$$Bel(L_t = l | s_{1,...,t-1}, a_{1,...,t-1}) = \frac{P(s_t | L_t = l, s_{1,...,t-1}, a_{1,...,t-1})P(L_t = l | s_{1,...,t-1}, a_{1,...,t-1})}{P(s_t | s_{1,...,t-1}, a_{1,...,t-1})} \quad (3.1)$$

Aplicando as simplificações de notação e as deduções de Fox (1998), baseadas na hipótese de Markov, na Equação 3.1, são obtidos os dois modelos utilizados pelo algoritmo de Localização de Markov, o modelo de movimento e o modelo de observação. O termo $P(L_t = l | L_{t-1} = l', a_{t-1})$ do Algoritmo 1 corresponde ao modelo de movimento e $P(s_t | L_t = l)$, ao modelo de observação, onde l' é o local do robô antes de executar a ação a_{t-1} .

Algoritmo 1: Localização de Markov

para cada local l **faça**

 inicialize $Bel(L_t = l_0)$

fim para

enquanto verdadeiro

se uma ação a é executada **então**

para cada local l **faça**

$$P(L_t = l | L_{t-1} = l', a_{t-1}) \leftarrow \sum_{l'} P(L_t = l | L_{t-1} = l', a_{t-1}) Bel(L_{t-1} = l')$$

fim para

fim se

se uma observação s é percebida **então**

para cada local l **faça**

$$Bel(L_t = l) \leftarrow P(s | l) P(L_t = l | L_{t-1}, a_{t-1})$$

fim para

fim se

para cada local l **faça**

$$\text{Normalize } Bel(L_t = l) \leftarrow \frac{Bel(L_t = l)}{P(s | L_t)}$$

fim para

fim enquanto

No Algoritmo 1 a crença $Bel(L_t=l_0)$ ou $Bel(L_0)$ reflete o conhecimento a respeito da localização inicial do robô e é utilizada para inicializar o algoritmo de Localização de Markov. Se a localização do robô com relação ao mapa é totalmente desconhecida $Bel(L_0)$ é inicializada por uma distribuição uniforme. Se a posição inicial é conhecida com certeza absoluta, então $Bel(L_0)$ é uma distribuição de Dirac centralizada nesta posição (FOX, 1998). Segundo Odakura (2007) se a localização inicial do robô é conhecida aproximadamente, então $Bel(L_0)$ é uma distribuição gaussiana estreita centrada nesta posição.

Após a inicialização da crença para cada local l o Algoritmo 1 torna-se recursivo e passa a atualizar a crença $Bel(L_t=l)$ de cada local l sempre que o movimento do robô é percebido pelos sensores internos, geralmente através de odometria, ou uma nova observação é obtida da leitura dos sensores externos, tais como sonares ou um sensor de medição laser.

A função Normalize do Algoritmo 1 irá normalizar os valores de $Bel(L_t=l)$, após serem atualizados pela aplicação do modelo de movimento e/ou pela aplicação do modelo de observação, de modo que a distribuição de probabilidade de $Bel(L_t)$ seja igual a um, ou seja, os valores são normalizados para que a somatória de todas as crenças dentro do espaço L seja igual a um.

A Figura 3.2, extraída de PROBABILISTIC ROBOTICS (2005), ilustra o funcionamento da Localização de Markov assumindo que o cenário de localização do robô móvel seja de uma dimensão. A seguir, a descrição sobre as etapas ilustradas, complementando a explicação de Ribeiro e Lima (2002).

Na etapa da Figura 3.2a o robô é colocado em algum lugar do ambiente, mas a sua localização inicial não é fornecida, sendo assim, a crença $bel(x)$ (na notação do texto, $Bel(L_t)$) é inicializada com uma distribuição uniforme. Na Figura 3.2b o robô consulta seus sensores e descobre que está próximo a uma porta, a crença $bel(x)$ é atualizada aplicando o modelo de observação, como a crença atual $bel(x)$ é uniforme será apenas incrementada nos locais do mapa próximos a uma porta, neste caso três. Na Figura 3.2c o robô móvel se desloca para frente e a crença $bel(x)$ é atualizada aplicando o modelo de movimento, $bel(x)$ fica mais suave devido à compensação do erro sensorial, inerente ao movimento do robô. Na Figura 3.2d o robô móvel consulta seus sensores e descobre que novamente está próximo a uma porta, o modelo de observação é aplicado novamente, no entanto como a

crença atual $bel(x)$ não é mais uniforme a nova distribuição de $bel(x)$ se concentra no local próximo à localização real atual do robô. Na Figura 3.2e o robô móvel se desloca para frente e a crença $bel(x)$ é novamente atualizada aplicando o modelo de movimento, o modelo de observação será aplicado novamente assim que o robô perceber a terceira porta.

Figura 3.2: Localização de Markov em um cenário de localização 1D.

3.2 A Localização de Monte Carlo

Também conhecida na literatura como Filtro de Partículas, a Localização de Monte Carlo (em inglês, *Monte Carlo Localization* (MCL)) é uma versão da Localização de Markov (FOX et al., 1999). Neste método, a incerteza da localização é representada pela manutenção de um conjunto de amostras sorteadas aleatoriamente do espaço de uma função densidade de probabilidade. A atualização

desta representação é feita utilizando métodos de Monte Carlo inventados por Handschin (1970) e mais tarde aplicados no rastreamento de alvos por Gordon, Salmond e Smith (1993), estatística por Kitagawa (1996) e visão computacional por Isard e Blake (1996, 1998) (DELLAERT et al., 1999) (FOX et al., 1999).

Em outras palavras: “Este algoritmo aproxima o estado do robô valendo-se de um conjunto de partículas que representam amostras do espaço de estados distribuídas de acordo com uma função de probabilidade” (MAZZOTTI, 2009, p. 12). De modo que, “um conjunto de partículas constitui uma aproximação discreta de uma distribuição de probabilidade” (ODAKURA, 2007, p. 30).

Devido a esta representação baseada em amostragem, este método reduz drasticamente a quantidade de memória requerida em comparação a Localização de Markov baseada em grids, permitindo que medições sejam integradas em uma frequência consideravelmente maior (DELLAERT et al., 1999) (FOX et al., 1999). De acordo com Odakura (2007) a vantagem do Filtro de Partículas em relação à Localização de Markov é a redução na computação e memória necessárias.

Fox et al. (1999) utilizou um esquema de amostragem adaptativa de Koller e Fratkina (1998) que determina o número de amostras durante a execução. Como resultado, o MCL gera muitas amostras durante a localização global, que é quando estas são mais necessárias, enquanto que o tamanho do conjunto de amostras é pequeno durante o rastreamento, quando a posição do robô é aproximadamente conhecida.

Filtros de Partículas foram capazes de resolver dois importantes problemas, anteriormente não resolvidos, conhecidos como a localização global e o robô sequestrado, nos quais o robô tem que recuperar sua posição a partir de uma incerteza global (THRUN, 2002b). E com mais precisão do que a Localização de Markov com células de tamanho fixo, pois o estado representado nas amostras não é discretizado (DELLAERT et al., 1999) (FOX et al., 1999).

Na modelagem do MCL, partículas são pares ordenados, incluindo postura e peso, (x^m, w^m) , para $m=1, \dots, M$, onde $x=(x, y, \theta)$ e $w \geq 0$ é o fator de importância análogo a uma distribuição de probabilidade discreta com $\sum_{m=1}^M w^m = 1$. O conjunto de partículas é dado por $\chi = \{(x^1, w^1), (x^2, w^2), \dots, (x^M, w^M)\}$ (ODAKURA, 2007).

O Algoritmo do MCL segundo a descrição de Thrun, Burgard e Fox (2005) é apresentado no Algoritmo 2. O mecanismo de funcionamento deste algoritmo é ilustrado na Figura 10 extraída de PROBABILISTIC ROBOTICS (2005).

Algoritmo 2: MCL (χ_{t-1}, u_t, z_t, m)

$$\bar{\chi}_t = \chi_t = \emptyset$$

para $m = 1$ **até** M **faça**

$$x_t^{[m]} = \text{amostra do modelo de movimento}(u_t, x_{t-1}^{[m]})$$

$$w_t^{[m]} = \text{modelo de medição}(z_t, x_t^{[m]}, m)$$

$$\bar{\chi}_t = \bar{\chi}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$$

fim para

para $m = 1$ **até** M **faça**

amostre i com probabilidade $\propto w_t^{[i]}$

adicione $x_t^{[i]}$ a χ_t

fim para

retorna χ_t

A Figura 3.3 ilustra o MCL usando o exemplo do corredor unidimensional. Seguindo a explicação de Thrun, Burgard e Fox (2005), a incerteza global inicial é conseguida por meio de um conjunto de partículas de poses escolhidas ao acaso e de maneira uniforme ao longo de todo o espaço de posturas, como mostrado na Figura 3.3a. À medida que o robô detecta a porta, o MCL atribui fatores de importância para cada partícula. O conjunto de partículas resultante é mostrado na Figura 3.3b, onde a altura de cada partícula representa o peso do fator de importância. A Figura 3.3c mostra o conjunto de partículas após a reamostragem e depois de incorporar o movimento do robô. Isto leva a um novo conjunto de partículas com pesos de importância uniformes, mas com um aumento do número de partículas próximo dos três lugares prováveis. A nova medição atribui pesos de importância não uniformes para o conjunto de partículas, tal como mostrado na Figura 3.3d. Nesta etapa, a maior parte da massa da probabilidade acumulada está centrada sobre a segunda porta, que é também o local mais provável. Além disso, o movimento leva a outro passo de reamostragem e a um passo em que um novo

conjunto de partículas é gerado de acordo com o modelo de movimento (Figura 3.3e).

Figura 3.3: O MCL em um cenário de localização 1D.

3.3 A Localização baseada no Filtro de Kalman Estendido

Técnicas baseadas no Filtro de Kalman têm sido amplamente utilizadas na problemática da estimativa de localização, tal como no rastreamento de mísseis e na navegação de navios (LEONARD; DURRANT-WHITE, 1991, 1992). Houve muitas aplicações notáveis do EKF em sistemas de robôs móveis, tal como o sistema de visão de tempo real de Dickmanns (1987), a representação 3-D do ambiente de Ayache e Faugeras (1989) e a navegação utilizando visão estéreo em Matthies e Shafer (1987) e em Kriegman, Triendl e Binford (1989) (LEONARD; DURRANT-WHITE, 1992).

Técnicas de localização baseadas no Filtro de Kalman provaram ser robustas e precisas em manter uma estimativa da posição do robô nos trabalhos de: Leonard e Durrant-white (1992), Schiele e Crowley (1994) e Gutmann e Schlegel (1996). O Filtro de Kalman é, também, um algoritmo particularmente eficiente devido à sua representação concisa, pois as matrizes de média e covariância são suficientes para descrever toda a densidade de probabilidade do estado estimado (DELLAERT et al., 1999).

Segundo Fabrizi et al. (1998) a adoção do EKF é a abordagem mais usual para a localização de robôs. O uso do EKF, no lugar do Filtro de Kalman padrão, deriva da necessidade de lidar com um modelo cinemático não linear para determinar uma previsão da posição do robô e a incerteza associada (representada por uma matriz de covariância), através da odometria e a partir das saídas não lineares (medidas do sensor) relativas ao estado do robô (FABRIZI et al., 1998). Devido à sua robustez, o EKF ainda continua a ser utilizado na localização de robôs móveis em trabalhos mais recentes tal como a localização em ambientes estruturados baseada em EKF de Teslić, Škrjanc e Klančar (2011).

O EKF substitui as previsões lineares do Filtro de Kalman, desenvolvido por Kalman (1960), por generalizações não lineares. O algoritmo de localização baseado no EKF é formado por um conjunto de equações matemáticas utilizadas para lidar com as incertezas provenientes do ruído sensorial e do movimento do robô no ambiente. A metodologia matemática aplicada nas equações do algoritmo inclui a modelagem matemática do robô e envolve diversas operações matriciais, tais como: soma, subtração, multiplicação, cálculo da matriz transposta, cálculo da matriz inversa, bem como o cálculo de uma matriz Jacobiana.

Figura 3.4: Diagrama Geral para a Localização de Robôs Móveis.

O EKF é um algoritmo recursivo e segue o Diagrama Geral para a Localização de Robôs Móveis (Figura 3.4), proposto por Siegwart, Nourbakhsh e Scaramuzza (2011). É dividido em duas partes principais: Predição e Correção. Na etapa de predição é feita uma estimativa inicial da posição usando as equações que descrevem o movimento do robô. Existem dois modelos mais utilizados na modelagem cinemática do robô, são o modelo baseado na velocidade e o modelo baseado na odometria, o modelo a ser utilizado depende da configuração comercial do robô. Neste trabalho optou-se pelo modelo de odometria por ter um equacionamento mais simples em relação ao modelo baseado na velocidade. No modelo de odometria o deslocamento do robô é dado em função da leitura de odometria em um intervalo de tempo, no qual o movimento do robô é composto por uma rotação inicial δ_{ROT1} , seguida por uma translação δ_{TRANS} e por uma segunda rotação δ_{ROT2} (THRUN; BURGARD; FOX, 2005), o modelo de odometria é ilustrado na Figura 3.5 extraída de PROBABILISTIC ROBOTICS (2005).

Figura 3.5: Modelo de Odometria - O movimento do robô no intervalo $(t-1, t]$.

Já na etapa de correção, características extraídas do ambiente na forma de dados sensoriais são comparadas com medidas estimadas, calculadas em função de um mapa conhecido do ambiente e da posição estimada na etapa de predição. Esta comparação também é conhecida, em inglês, como *matching*. Na sequência, a posição do robô é atualizada com base no *matching* positivo. O Algoritmo 3, adaptado de Thrun, Burgard e Fox (2005), mostra em detalhes as equações da Localização baseada no EKF.

Como citado anteriormente, o EKF é um algoritmo recursivo, por isso a estimativa de posição resultante, composta pelo vetor de estado μ_t e pela matriz de covariância Σ_t , é utilizada como parâmetro de entrada para a próxima iteração do algoritmo no instante $t+1$.

Algoritmo 3: Localização baseada no EKF ($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m$)

Predição:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad (3.2)$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \quad (3.3)$$

Correção:

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \quad (3.4)$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t)) \quad (3.5)$$

$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \quad (3.6)$$

Retorna μ_t, Σ_t

Onde:

μ_t Vetor das variáveis de estado (x_t, y_t, θ_t)

$g(u_t, \mu_{t-1})$ Função de transição das variáveis de estado

u_t Vetor de controle ($\delta_{ROT1}, \delta_{TRANS}, \delta_{ROT2}$)

Σ_t Matriz de covariância das variáveis de estado

G_t Matriz Jacobiana de $g(u_t, \mu_{t-1})$

R_t Matriz de erro do modelo de movimento

K_t Ganho de Kalman

H_t Matriz Jacobiana de $h(\bar{\mu}_t)$

Q_t Matriz de erro da medição

z_t Vetor da medição (r_t^i, ϕ_t^i, s_t^i)

$h(\bar{\mu}_t)$ Medição estimada ($\hat{r}_t^i, \hat{\phi}_t^i, \hat{s}_t^i$)

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \quad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

Detalhamento das funções $g(u_t, \mu_{t-1})$ e $h(\bar{\mu}_t)$:

$$g(u_t, x_{t-1}) = x_{t-1} + \delta_{TRANS} \cdot \cos(\theta_{t-1} + \delta_{ROT1})$$

$$g(u_t, y_{t-1}) = y_{t-1} + \delta_{TRANS} \cdot \sin(\theta_{t-1} + \delta_{ROT1})$$

$$g(u_t, \theta_{t-1}) = \theta_{t-1} + \delta_{ROT1} + \delta_{ROT2}$$

$$\hat{r}_t^i = \sqrt{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2}$$

$$\hat{\phi}_t^i = a \tan 2(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta}$$

$$\hat{s}_t^i = m_{j,s}$$

Na Equação 3.2 é estimada uma posição utilizando os parâmetros δ_{ROT1} , δ_{TRANS} e δ_{ROT2} , calculados com base na posição (x, y, θ) fornecida pela função `position2d` do *software* Player/Stage. No ambiente simulado esta função retorna uma posição exata do robô, logo seria suficiente como referência de localização para o sistema de navegação. Entretanto, como um dos objetivos do trabalho é simular uma aplicação real, erros aleatórios, proporcionais ao deslocamento, são gerados e foram incluídos em cada uma das variáveis δ_{ROT1} , δ_{TRANS} , δ_{ROT2} . Os erros numéricos gerados aleatoriamente durante a simulação podem ser negativos, positivos ou nulos.

Na Equação 3.3 é feita a atualização da covariância utilizando a última covariância estimada e a covariância do ruído do modelo de movimento, no caso o modelo de odometria. Em aplicações reais o ruído da odometria é obtido experimentalmente. Na simulação o ruído incluído é coerente com os valores reais calculados em experimentos disponíveis na literatura. Este ruído serve para o cálculo do valor numérico do erro de odometria utilizado na simulação.

Na Equação 3.4 ocorre o ganho de Kalman que computa o ruído de medição. Neste trabalho o ruído da medição é incluído na distância (r) e no ângulo (ϕ), obtidos a partir da posição relativa entre o *landmark* e o robô. O ruído e o erro desta medição são tratados no simulador da mesma maneira que o ruído e os erros de odometria.

Uma característica importante do EKF é que a precisão sensorial da etapa de correção deve ser maior que a precisão da etapa de predição. Logo o ruído simulado para a odometria é maior que o ruído simulado para a medição da distância em relação aos *landmarks*.

Na Equação 3.5 é feito o cálculo da inovação que corresponde à diferença entre os valores lidos e os valores previstos com base no mapa, conforme descrição detalhada da função $h(\bar{\mu}_t)$. Na Equação 3.6 é feita a correção da covariância.

A etapa de correção é executada para cada *landmark* identificado pelo robô no instante que o algoritmo é solicitado, por esse motivo quanto maior o número de *landmarks* identificados pelo robô no instante t , mais precisa será a resposta do algoritmo. Se o robô não identifica qualquer *landmark* no instante t , a posição é estimada somente com base na odometria.

Figura 3.6: O Filtro de Kalman em um cenário de localização 1D.

A Figura 3.6, montada a partir das figuras de *PROBABILISTIC ROBOTICS* (2005), ilustra as etapas do funcionamento do Filtro de Kalman em um cenário de uma dimensão. Tendo como contexto a localização 1D, na etapa da Figura 3.6a tem-se uma estimativa inicial (Gaussiana vermelha) a respeito da localização do robô com uma determinada incerteza, que reflete o conhecimento sobre a localização inicial do robô, na Figura 3.6b o robô utilizando os dados sensoriais efetua uma medição do ambiente (Gaussiana verde) com uma incerteza proveniente do ruído sensorial. Na Figura 3.6c, combinando a estimativa anterior com a medição sensorial obtêm-se uma nova estimativa através do Filtro de Kalman, com uma distribuição Gaussiana mais estreita representada em azul. Após a movimentação do robô em direção à direita (Figura 3.6d) tem-se um aumento na incerteza e uma

nova estimativa de posição é calculada (Gaussiana magenta), uma nova medição é obtida do ambiente na Figura 3.6e (Gaussiana ciano). Novamente, o Filtro de Kalman combina a última estimativa com a última medição do ambiente gerando uma estimativa atualizada representada em amarelo na Figura 3.6f.

3.4 Algoritmos de Extração de Segmentos de Linhas

Nguyen et al. (2005) fez um estudo comparativo de seis algoritmos de extração de segmentos de linhas comumente utilizados na robótica móvel e na visão computacional. O estudo foi baseado em experimentos utilizando dados reais obtidos a partir de um sensor laser de medição 2D em um ambiente *indoor* e teve como critérios de comparação, velocidade, complexidade, exatidão e precisão.

Segundo Nguyen et al. (2005), o *Split-and-Merge* é provavelmente o algoritmo de extração de segmentos de linhas mais popular e teve origem na visão computacional através de Pavlidis e Horowitz (1974). O *Split-and-Merge* também é conhecido como algoritmo de Ramer-Douglas-Peucker (RDP), sugerido independentemente por Ramer (1972) e Douglas e Pecker (1973), e também como *Iterative End Point Fit* (IEPF) (RAMER–DOUGLAS–PEUCKER, 2012).

No contexto de aplicações de tempo-real o *Split-and-Merge* é considerada a melhor escolha devido à sua velocidade superior entre os algoritmos testados por Nguyen et al. (2005), sendo a primeira escolha para problemas de localização com um mapa conhecido, onde a ocorrência de falsas linhas não é tão importante. Essa afirmação deve-se ao fato que durante a execução do algoritmo de localização, as falsas linhas dificilmente são computadas na estimativa da localização, pois existe uma probabilidade menor das falsas linhas serem emparelhadas com as linhas que compõem o mapa conhecido durante o processo de *matching*.

O IEPF é um algoritmo recursivo inicialmente proposto por Duda e Hart (1973) no tratamento imagens, passou a ser aplicado na navegação de robôs móveis por Crowley (1985, 1989). Segundo Siadat et al. (1997) o IEPF apresentou-se lento quando comparado a outros algoritmos como o *Successive Edge Following* (SEF) ou o *Line Tracking* (LT), pois necessita de um pré-processamento e de um pós-processamento dos dados, entretanto possui um comportamento muito robusto

tanto em ambientes *indoor* como *outdoor*. No experimento comparativo de Siadat et al. (1997) foi o algoritmo que apresentou o melhor resultado para representar o ambiente, mas foi considerado lento na época para ser aplicado na estimativa de posição. Hoje, a evolução dos recursos computacionais torna viável o uso deste algoritmo na navegação de robôs móveis em tempo-real e por conta de sua popularidade na robótica móvel foi o algoritmo escolhido para a extração de linhas no sistema de localização deste trabalho. O funcionamento do IEPF é descrito no Algoritmo 4 apresentado brevemente a seguir.

O IEPF recebe um conjunto de N pontos obtidos da leitura do sensor. Este conjunto de pontos inicialmente é tratado como um único segmento s_1 e é armazenado em uma lista de segmentos L . Inicia-se, então, a etapa recursiva de *split* ilustrada na Figura 3.7, adaptada de Duda e Hart (1973). Uma reta ℓ é obtida conectando o primeiro e o último ponto do segmento s_i da lista L e procura-se o ponto p_j cuja distância e_{MAX} ponto-linha da reta ℓ seja máxima. Se e_{MAX} é maior que um limiar d_{MIN} , o segmento s_i é dividido no ponto p_j em s_{i1} e s_{i2} e a lista L é atualizada substituindo s_i por s_{i1} e s_{i2} . Caso e_{MAX} não seja maior, calcula-se os parâmetros (ρ, α) da reta obtida a partir dos pontos $(p_1, \dots, p_m)$ de s_i , e repete-se o processo para o próximo segmento s_{i+1} de L . Com a atualização da lista L , os pontos do segmento s_{i1} são analisados novamente e processados seguindo o mesmo critério, s_{i1} poderá ser dividido até que o limiar d_{MIN} seja alcançado. A etapa de *split* termina quando todos os elementos de L são processados. Finalmente, ocorre a etapa de *merge* que combina as linhas com parâmetros similares obedecendo a um limiar. O algoritmo então retorna o conjunto R de retas descritas pelos parâmetros (ρ_i, α_i) .

Figura 3.7: Ilustração da etapa de *split* do IEPF.

Algoritmo 4: IEPF $(p_1, p_2, \dots, p_n)$

```

segmento  $s_1 \leftarrow [p_1, p_2, \dots, p_n]$ 
lista  $L \leftarrow \{s_1\}$ 
 $i \leftarrow 1$ 
lista de retas  $R \leftarrow \{ \}$ 
enquanto  $i \leq \text{tamanho}(L)$  faça
     $s_i \leftarrow L[i]$ 
     $[p_1, p_2, \dots, p_m] \leftarrow s_i$ 
    linha  $\ell \leftarrow \text{reta}(p_1, p_m)$ 
    busca o ponto  $p_j$  com  $e_{\perp MAX}$  de  $\ell$ 
    se  $e_{\perp MAX} > d_{MIN}$  então
        divida  $s_i$  em  $s_{i1}$  e  $s_{i2}$  no ponto  $p_j$ 
        substitua  $s_i$  em  $L$  por  $s_{i1}$  e  $s_{i2}$ 
    senão
        calcula  $(\rho_i, \alpha_i) \leftarrow \text{Mínimos Quadrados}(p_1, p_2, \dots, p_m)$ 
        adiciona  $(\rho_i, \alpha_i)$  a  $R$ 
         $i \leftarrow i + 1$ 
    fim se
fim enquanto
merge  $R$ 
retorna  $R$ 

```

4 METODOLOGIA

Como visto no Capítulo 1, o arranjo físico interno de um armazém é construído em função do tipo de produto que será armazenado. Pode depender, ainda, do espaço disponível, do fluxo interno e dos acessos para a entrada e saída dos produtos, entre outros fatores. Nessa situação, o *layout* de um armazém é um fator determinante na escolha de um modelo de localização apropriado. Este fator torna-se mais ameno quando o armazém é construído ou adaptado para comportar um sistema de localização ou navegação, como é caso dos primeiros AGVs ou mais recentemente do Kiva Systems. Porém, como o objetivo, em trabalhos futuros, é utilizar o Sistema de Auto-localização em AGVs inseridos nos processos de armazéns existentes, é importante que o sistema seja flexível ou adaptável ao arranjo físico do armazém que será automatizado, procurando minimizar as adaptações no armazém existente.

Visando obter um sistema de auto-localização para AGVs mais flexível e verificar as dificuldades de implementação de cada modelo de localização escolhido, foram desenvolvidas duas versões de algoritmo de auto-localização independentes. Cada versão utiliza um modelo de localização diferente e para testar cada um deles foram construídos dois cenários. O primeiro cenário corresponde a um armazém simulado, no qual foi testado o modelo de localização baseado em *landmarks* artificiais em conjunto com outros dois sistemas: navegação e roteamento. No segundo cenário é testada a localização baseada em *features* utilizando dados sensoriais reais obtidos a partir de um robô Pioneer 3AT e um *feature-based map* de um ambiente estruturado *indoor*. Ambos os algoritmos utilizaram os *drivers* do Player/Stage para obter os dados sensoriais e o EKF para realizar a fusão sensorial.

A parte do Sistema de Auto-localização correspondente à etapa de Predição do algoritmo de localização baseado em EKF é comum aos dois cenários, tendo como diferencial o erro aplicado no modelo de movimento. Em ambos cenários, o modelo de movimento escolhido foi o modelo baseado em odometria devido à facilidade de implementação tanto no ambiente simulado como no ambiente real.

O EKF comporta a utilização dos dois modelos de localização simultaneamente, porém com um aumento da complexidade do algoritmo de

localização. A utilização simultânea dos dois modelos de localização resulta em um ganho na flexibilidade do sistema de auto-localização, já que a localização poderia ser estimada com segurança em locais do armazém em que um dos dois modelos de localização seria falho ou inviável de ser implementado. Entretanto, devido às restrições de cada cenário e as hipóteses utilizadas, os modelos foram implementados separadamente, deixando o desenvolvimento e teste do algoritmo de localização composto por ambos os modelos para trabalhos futuros.

A descrição da modelagem e das etapas de construção de cada cenário testado foi dividida em dois tópicos principais: ambiente simulado e ambiente real, apresentados a seguir.

4.1 O Ambiente Simulado

Foi implementado no ambiente simulado 2D o Sistema de Auto-localização baseado em *landmarks* artificiais que interage com outros dois sistemas: Navegação Local e Roteamento, conforme mostra a Figura 4.1. O Sistema de Navegação e o Sistema de Roteamento foram desenvolvidos em parceria com outros dois alunos do Laboratório de Robótica Móvel do Grupo de Mecatrônica da EESC-USP. O Sistema de Navegação Local foi desenvolvido pelo aluno de Iniciação Científica & TCC Thales Bueno Pasqual, detalhes do funcionamento deste sistema podem ser vistos em Pasqual (2010). Já o Sistema de Roteamento foi desenvolvido pela aluna Kelen Cristiane Teixeira Vivaldini no trabalho de mestrado referenciado em Vivaldini (2010).

A Figura 4.1 apresenta alguns sistemas envolvidos em uma aplicação de AGVs de empilhadeiras em um armazém inteligente e mostra o fluxo das informações que ocorre entre estes sistemas para concluir uma tarefa de armazenamento ou retirada de materiais. A aplicação no modelo proposto da Figura 4.1 é composta por três sistemas: o Sistema de Roteamento, responsável por calcular a melhor rota, evitando obstáculos previsíveis; o Sistema de Navegação Local que executa a rota fornecida pelo Sistema de Roteamento; e o Sistema de Auto-localização que fornece a estimativa de posição e orientação durante a navegação.

Figura 4.1: Arquitetura proposta para os sistemas envolvidos na aplicação de AGVs de empilhadeiras em armazéns inteligentes.

Na arquitetura proposta da Figura 4.1 cada empilhadeira robótica possui um conjunto próprio de sensores e atuadores e dois sistemas embarcados: Auto-localização e Navegação Local. Tendo como base esta arquitetura, todas as empilhadeiras informam regularmente ao Sistema de Roteamento o status sobre as ações executadas. Dessa forma, todas as rotas podem ser verificadas globalmente pelo sistema de roteamento e computadas de forma centralizada de modo que as rotas possam ser recalculadas sempre que necessário para evitar conflitos ou otimizar trajetos, por exemplo. Entretanto, para o experimento simulado utilizando o Player/Stage o roteamento dinâmico não foi testado, devido à dificuldade de simular a execução dos três sistemas simultaneamente. Sendo assim, o Sistema de Roteamento foi executado *off-line*, ou seja, separadamente dos outros dois sistemas. Para simular a comunicação entre os sistemas proposta na Figura 4.1, uma tarefa é fornecida ao Sistema de Roteamento e após sua execução *off-line* obtém-se uma rota, a qual é gravada em um *log* e lida pelo Sistema de Navegação Local durante o experimento simulado.

O sistema de roteamento de Vivaldini (2010), baseado no algoritmo de Dijkstra (1959) e no Método de Roteamento com janela de tempo (DESROSIERS et al., 1986), recebe as informações sobre as tarefas que devem ser executadas

(quantidade de paletes, ponto de carregamento e descarregamento, tempo de início, etc.), baseado nestas informações determina a quantidade necessária de empilhadeiras robóticas para executar as tarefas recebidas. A rota de cada empilhadeira escalada para executar as tarefas é, então, calculada utilizando um mapa topológico do ambiente. Durante este processo são previstos possíveis bloqueios, congestionamentos e no final a rota é ajustada para otimizar o número de manobras. As rotas calculadas são, então, enviadas para cada empilhadeira robótica pelo Sistema de Roteamento que monitora regularmente o progresso da execução de todas as tarefas processadas em função dos estados de cada empilhadeira robótica (VIVALDINI, 2010).

A rota calculada pelo Sistema de Roteamento é composta por um conjunto de pontos que correspondem a posições (x, y) no mapa topológico, denominada rota global segundo Vivaldini (2010, p. 54). O Sistema de Navegação Local de cada empilhadeira robótica recebe este conjunto de pontos e calcula um caminho local para o trajeto entre os primeiros dois pontos da rota global, considerando obstáculos inesperados observados nesta posição através das informações dos sensores, utilizando como base o algoritmo A* (HART; NILSSON; RAPHAEL, 1968). Comandos são enviados aos atuadores para navegar no caminho local obtido com uma velocidade pré-estabelecida, ajustando a orientação do robô de modo a manter a direção dentro do caminho local. Na sequência, um novo caminho local é calculado quando o segundo ponto do trajeto local é alcançado, tendo como critério um limiar de proximidade para avaliar se o ponto foi alcançado ou não. A partir daí, novos caminhos locais vão sendo gerados e executados à medida que a rota global é processada ponto a ponto pelo Sistema de Navegação Local até atingir o último ponto da rota global.

Utilizando os dados dos sensores e o mapa do ambiente, que neste cenário corresponde a um *landmark-based map* de *landmarks* artificiais, o Sistema de Auto-localização de cada empilhadeira robótica atualiza os valores da localização estimada (x, y, θ) . Este sistema funciona em paralelo aos outros sistemas, dessa maneira a localização estimada é constantemente atualizada e registrada em variáveis globais acessadas pelo Sistema de Navegação Local, que busca estes valores em diversos instantes. Os valores da posição estimada (x, y) são utilizados pelo Sistema de Navegação Local para confirmar cada ponto da rota global percorrido durante a execução do trajeto, bem como para confirmar o destino final

da rota global. Os valores de localização também são utilizados para verificar se a posição do robô empilhadeira está dentro do caminho local traçado pelo Sistema de Navegação, através de um critério de comparação, no qual o robô é considerado fora do caminho local quando a distância entre a posição estimada com relação ao caminho local ultrapassa um limite pré-estabelecido. Caso a empilhadeira robótica seja considerada fora do trajeto previsto no caminho local, o Sistema de Navegação Local corrige o trajeto em andamento alterando a orientação da empilhadeira robótica em direção ao caminho local até que a posição estimada seja considerada dentro do caminho local. Para calcular a amplitude necessária nesta variação da orientação, o Sistema de Navegação Local utiliza como referência a orientação estimada θ da empilhadeira robótica, que também é fornecida pelo Sistema de Auto-localização.

A seguir, serão apresentadas as etapas de implementação deste cenário, tendo como foco os detalhes de implementação do Sistema de Auto-localização e as hipóteses utilizadas, que são: inserção do modelo do armazém inteligente no ambiente simulado, construção do *featured-based map* de *landmarks* artificiais e a localização utilizando os dados sensoriais fornecidos pelo Player/Stage.

4.1.1 Representação do Armazém Inteligente

O ambiente simulado representa um armazém inteligente de 30 x 50m. A simulação deste ambiente tem como objetivo buscar a solução mais adequada a esta aplicação, bem como analisar a viabilidade do sistema em armazéns já existentes, que podem ser reproduzidos no simulador. A simulação foi realizada no Linux Ubuntu usando o *software* de simulação Player/Stage. A partir de um modelo de armazém próximo do real é desenhada um planta baixa 2D, que é convertida em imagem (Figura 4.2) e interpretada em escala pelo simulador para gerar um modelo virtual do armazém. O modelo virtual fornece as informações sobre as fronteiras do armazém e as áreas inacessíveis pelas empilhadeiras robóticas.

Figura 4.2: Imagem da planta 2D do armazém inteligente utilizado na simulação.

4.1.2 Inclusão dos *Landmarks* Artificiais

Para simular o EKF, noventa e cinco *landmarks* foram incluídos no ambiente simulado de forma estruturada e com localização conhecida para serem detectados pelo robô móvel durante a simulação. A informação sobre o posicionamento de cada *landmark* em relação ao robô (r_t^i, ϕ_t^i, s_t^i) é calculada de acordo com a sua localização $(m_{j,x}, m_{j,y}, m_{j,s})$ no mapa e utilizada durante a execução do EKF. O conjunto formado pelas localizações de cada *landmark* $(m_{j,x}, m_{j,y}, m_{j,s})$ forma o *landmark-based map*.

Como o ambiente simulado corresponde a um armazém tem-se a vantagem do espaço estruturado, isto diminui o número de manobras de mudança de direção executadas pelo robô durante a execução de sua rota, consequentemente ocorre uma redução no acúmulo de erro de odometria. Porém, este tipo de ambiente também tem um espaço limitado nos corredores, logo, uma perda da localização, mesmo que temporária, pode resultar em colisão com as prateleiras ou as paredes.

Por essa razão, os *landmarks* foram distribuídos nas áreas navegáveis de acordo com o alcance do sensor de detecção de modo que durante a simulação espera-se que o robô possa identificar ao menos um a cada instante t . Foi utilizado o *driver fiducial* do *software* Player/Stage para simular os *landmarks*, este *driver* permite a configuração do alcance do sensor de detecção simulado e a identificação individual de cada *landmark*. A Figura 4.3 apresenta uma parte do ambiente simulado, nesta área é possível visualizar nove *landmarks*, sendo que, no instante

representado na figura, quatro são detectados, e destes quatro, três são reconhecidos (identificados) pelo seu número de id. (3, 4 e 9).

Figura 4.3: Landmarks inseridos no ambiente simulado.

Além do *driver fiducial*, também foram utilizados os *drivers* do Player/Stage: *stage*, *position2d* e *laser*. O *driver stage* é utilizado para a simulação, o *position2d* é responsável por fornecer os dados de odometria e o *driver laser* é requerido para o funcionamento do *fiducial*.

4.2 O Ambiente Real

O algoritmo do Sistema de Auto-localização proposto foi testado em um ambiente *indoor* real utilizando os elementos da Figura 4.4. Basicamente, esta fase experimental foi dividida em duas etapas principais: construção do *feature-based map* e coleta dos dados sensoriais para o teste do algoritmo de localização 2D utilizando o Player/Stage.

A etapa de construção do *feature-based map* foi realizada a partir de um desenho de planta baixa, ajustado conforme as cotas construtivas reais, que foram medidas e registradas em um esboço da planta baixa. A junção das informações dos dois desenhos de planta baixa resultou em um mapa 2D no formato CAD. Em seguida, o mapa 2D foi utilizado como referencial para o cálculo da posição e da orientação das características incluídas no *feature-based map*. Os detalhes deste processo são mostrados no item 4.2.1.

O funcionamento da plataforma de hardware formada pelos elementos da Figura 4.4 e os drivers do Player/Stage utilizados para a coleta de dados do ambiente real, bem como o pós-processamento destes, são apresentados no item 4.2.2.

Figura 4.4: Configuração geral do experimento para o teste do Algoritmo de Auto-localização 2D no ambiente indoor real.

4.2.1 Construção do *feature-based map*

Como frisado no Capítulo 2, o fornecimento do mapa do ambiente é um pré-requisito para o funcionamento do algoritmo de auto-localização. Neste cenário, o mapa fornecido é um mapa do tipo *feature-based map*. Diferentemente do mapa de *grid*, o *feature-based map* é composto por uma lista de características do ambiente

pré-selecionadas com o registro da posição e da orientação de cada uma em relação ao sistema de coordenadas global escolhido.

Figura 4.5: Ambiente real – Sistema de coordenadas global escolhido.

Em ambientes reais, a escolha da posição e da orientação do sistema de coordenadas global é livre. Desta forma, para o experimento com dados reais, a posição da origem e os eixos X e Y, do sistema de coordenadas, foi escolhida em função dos trajetos planejados para o experimento e, também, de modo a favorecer o cálculo da posição e da orientação das características a serem inseridas no *feature-based map*. A localização do sistema de coordenadas global escolhido para o experimento com dados reais é mostrado na Figura 4.5, a importância da posição deste sistema com relação ao Mapa 2D será mostrada mais adiante neste capítulo.

Figura 4.6: Detalhe do desenho de projeto da planta baixa do primeiro pavimento do Prédio da Engenharia Mecatrônica da EESC – USP.

Em um ambiente interno estruturado, ou seja, com paredes e divisórias internas, nem sempre é possível medir de forma direta os valores de posição e orientação de uma característica em relação ao sistema de coordenadas global. Estes valores são calculados em função de duas ou mais cotas e por esse motivo, durante a medição manual do ambiente, é importante considerar todas as cotas construtivas necessárias para o cálculo indireto da posição e da orientação dos *features* do ambiente relevantes para a navegação do robô móvel no ambiente real.

A altura do sensor de medição laser montado no robô móvel também foi considerada durante a medição manual, descartando os ressaltos das paredes que não seriam lidos pelo sensor. Objetos, tais como extintores de incêndio e lixeiras também não foram considerados na construção do Mapa 2D. O diagrama da Figura 4.8 ilustra o procedimento adotado para a obtenção do Mapa 2D, iniciando com o desenho de projeto até a obtenção do *feature-based map*.

Figura 4.8: Construção do Mapa 2D e do Feature-based Map.

As cotas relativas mencionadas na Figura 4.8 foram calculadas de forma direta utilizando as próprias ferramentas do *software* de desenho que fornecem a distância entre dois objetos ou pontos. Antes deste cálculo foi preciso definir a área do mapa 2D que será utilizada durante os experimentos com o robô móvel real, para então selecionar as características do ambiente que irão fazer parte do *feature-based map*. Este procedimento descarta características que não seriam utilizadas pelo algoritmo de auto-localização durante o processo de *Matching* e que sobrecarregam o algoritmo.

A Figura 4.9 mostra o Mapa 2D no formato CAD. As paredes de alvenaria são representadas na cor azul e as divisórias internas na cor vermelha, algumas

linhas em ambas as cores representam portas fechadas ou acessos restritos. Será possível visualizar melhor a marcação das áreas restritas, após a definição da área navegável delimitada para os experimentos.

Figura 4.9: Mapa 2D no formato CAD, em escala aproximada de 1:200, obtido a partir da planta baixa após ajuste das cotas de construção.

A área navegável delimitada para o experimento com o robô móvel real é ilustrada na Figura 4.10, as áreas externas não foram consideradas no escopo deste trabalho, por isso não foram classificadas. A cor verde representa as áreas livres utilizadas na navegação do robô durante o experimento de localização, esta área corresponde principalmente aos corredores. Os locais com acesso restrito são representados na cor amarela, tais como o poço do elevador e as escadas. A cor cinza indica os setores nos quais o robô móvel estaria apto a circular, porém foram excluídos dos trajetos planejados para o experimento de localização em função do grande número de objetos não mapeados.

Objetos não mapeados restringem a circulação do robô móvel e dificultam o processo de localização baseado em *landmarks* naturais mapeados a partir de uma planta baixa. Grande parte dos *landmarks* naturais mapeados de um ambiente interno corresponde às paredes, por esse motivo, os objetos não mapeados, tais como móveis de escritório, obstruem o campo de visão do sensor laser, reduzindo,

assim, a probabilidade de um *landmark* ser reconhecido e computado pelo algoritmo de auto-localização durante o processo de *Matching*.

Figura 4.10: Classificação das áreas navegáveis no Mapa 2D.

Ainda restam duas etapas no planejamento do experimento, relativas ao mapa 2D, a escolha das características que serão incluídas no *feature-based map* e a definição do trajeto do robô móvel. O trajeto do robô terá como ponto de partida um local próximo à origem do sistemas de coordenadas global escolhido e como ponto final o mesmo local de partida. O robô móvel iniciará e terminará o trajeto com uma posição inicial e uma posição final, respectivamente, próximas uma da outra, mas não necessariamente coincidentes.

Inicialmente foi previsto um único trajeto para a coleta de dados sensoriais do ambiente de modo que o robô móvel transitasse em toda a área navegável demarcada. Entretanto, a carga das baterias do robô não foi suficiente para completar o trajeto previsto e acabou sendo realizado em duas etapas com uma parada intermediária para a recarga das baterias. Para obter um registro de dados contínuo um segundo trajeto reduzido foi executado com sucesso em uma única etapa. O registro de dados, mais conhecido como *log*, precisa ser contínuo durante o teste do algoritmo implementado. Deve-se frisar que este não é um requisito, mas foi uma prática adotada para evitar que erros na junção dos *logs* de duas ou mais

etapas fossem confundidos com erros na implementação do Sistema de Auto-localização.

Após o teste do algoritmo, *logs* de ambos os trajetos foram utilizados na obtenção e análise dos resultados. O primeiro trajeto planejado é ilustrado na Figura 4.11 e o trajeto reduzido para a obtenção de um *log* contínuo é mostrado na Figura 4.12.

Figura 4.11: Trajeto planejado – Experimento N° 01.

Figura 4.12: Trajeto planejado – Experimento N° 02.

Na Figura 4.11 o trajeto planejado é indicado por setas de duas cores: azul escuro e preto, o objetivo é diferenciar o trajeto de cada etapa do primeiro experimento. A etapa um, representada na cor azul escuro, corresponde ao trajeto que inicia no ponto de partida e termina na parada intermediária para a recarga das baterias. Neste trabalho o trajeto, indicado através de setas, não representa uma rota ou caminho a ser seguido, e sim o sentido de fluxo da navegação, sendo assim, durante os experimentos o robô móvel foi conduzido preferencialmente através do espaço no meio dos corredores para tornar a navegação mais segura e confortável. Uma visão do ponto de partida no ambiente real é mostrada na Figura 4.13.

Figura 4.13: Ponto de partida dos trajetos planejados para os experimentos 1 e 2.

A escolha do tipo de características do Mapa 2D que serão incluídas no *featured-based map* irá depender da programação da etapa de *Matching* do Sistema de Auto-localização utilizado. Nesta etapa as características extraídas do ambiente são comparadas com as características mapeadas, sendo assim, para que este processo ocorra com sucesso, as características extraídas têm que ser compatíveis

com as características disponíveis no *feature-based map*, ou seja, devem ser do mesmo tipo.

É necessário um algoritmo específico para interpretar cada tipo de característica a partir de dados sensoriais brutos coletados por um sensor. O sensor laser de medição utilizado nos experimentos deste trabalho (SICK-LMS200) é comercialmente classificado como um sensor de medição laser 2D por fornecer um conjunto de medidas de distância no mesmo plano 2D. Cada medida de distância corresponde à distância entre o obstáculo, reconhecido por reflexão do feixe laser, e o centro do espelho rotativo do sensor. Uma medida é fornecida para cada passo no ângulo de rotação do espelho rotativo, de acordo com a resolução angular e o ângulo de abertura, que dependem da configuração e do *datasheet* do sensor laser de medição utilizado.

O sensor de medição laser SICK-LMS200 tem um ângulo de abertura de 180° e foi configurado para uma resolução angular de $0,5^\circ$, portanto temos um conjunto de 361 medidas de distância a cada leitura dos valores sensoriais brutos. Estas medidas de distância são então convertidas em pontos no espaço, tendo como referência o sistema de coordenadas do robô móvel. Como as medidas são fornecidas por um sensor de medição laser 2D, os pontos resultantes pertenceram ao mesmo plano geométrico.

Figura 4.14: Montagem do sensor de medição laser 2D SICK-LMS200 na plataforma do robô móvel Pioneer P3-AT.

Durante a conversão das medidas de distância em pontos no espaço, deve-se considerar a posição do sensor de medição laser com relação ao sistema de

coordenadas do robô móvel. Sendo necessária uma compensação no cálculo dos pontos, quando a origem do sistema de coordenadas do robô móvel não coincide com a origem do sistema de coordenadas do sensor de medição laser em posição e/ou orientação.

A Figura 4.14 mostra a montagem do sensor de medição laser 2D na plataforma do robô móvel, nesta configuração, o plano geométrico, contendo os pontos lidos por este sensor no sistema de coordenadas do robô móvel, será paralelo ao plano horizontal do ambiente real.

Deste modo, o robô móvel irá interpretar o ambiente real a partir de um conjunto de pontos coplanares e o ambiente, então, será reconhecido através das formas geométricas 2D extraídas desse conjunto. As formas geométricas são obtidas pela segmentação dos pontos lidos pelo sensor de medição laser 2D, utilizando um critério de validação para conectar os pontos entre si. Sendo assim, qualquer forma geométrica resultante que reflita uma construção geométrica encontrada no ambiente real, pode caracterizar o ambiente e transformar-se em um *feature*.

Ambientes *indoor* distintos possuem padrões construtivos comuns, tais como pilares, paredes e portas, que são elementos padronizados na construção civil. No plano horizontal estes elementos construtivos podem ser caracterizados por linhas geométricas horizontais. Por esse motivo, neste trabalho, o conjunto ou tipo de características escolhidas, a serem extraídas do ambiente real e incluídas no *feature-based map*, serão as linhas geométricas horizontais. O algoritmo específico adotado para extrair este tipo de característica do ambiente é conhecido como algoritmo de extração de linhas.

O algoritmo de extração de linhas é um componente do algoritmo de localização utilizado neste trabalho, e tem como objetivo gerar os dados de entrada do processo de *Matching*. Sendo assim, para que o *feature-based map* seja compatível com este algoritmo durante o *loop* de comparação do processo de *Matching*, devem ser selecionadas apenas linhas geométricas do Mapa 2D como *landmarks* naturais.

A Figura 4.15 mostra as linhas geométricas inicialmente escolhidas para compor o *feature-based map*, que são diferenciadas das outras linhas do mapa pela cor verde. Foram selecionadas somente linhas localizadas dentro da área navegável com o objetivo de facilitar o *loop* de comparação do processo de *matching*. O

sistema de coordenadas global também é indicado na Figura 4.15, a importância de sua escolha na construção do *feature-based map* será explicada a seguir.

Figura 4.15: Linhas do Mapa 2D selecionadas para serem incluídas no *feature-based map*

Dependendo dos obstáculos entre o sensor de medição laser 2D e uma linha geométrica no ambiente, nem sempre é possível extrair toda a extensão desta linha durante a navegação. Ou seja, apenas uma parte da linha geométrica do ambiente é extraída e interpretada como uma linha de menor extensão. Este processo ainda é influenciado pelo alcance e pela resolução do sensor de medição laser 2D utilizado.

Devido à dificuldade de corresponder as linhas extraídas com as linhas mapeadas de tamanho e posicionamento diferentes, converte-se a linha extraída em dois parâmetros (r , α) que refletem a posição relativa entre o robô móvel e a linha extraída do ambiente. O parâmetro r corresponde à distância perpendicular entre a linha geométrica extraída e a origem do sistema de coordenadas do robô móvel, já o parâmetro α refere-se à rotação da reta perpendicular formada por r com relação ao eixo $Z_{\text{ROBÔ}}$, considerando como ponto zero o eixo $X_{\text{ROBÔ}}$.

A mesma transformação em parâmetros é aplicada às linhas geométricas escolhidas no Mapa 2D. Porém, neste caso o parâmetro r é tratado como a distância perpendicular entre a linha do mapa 2D e a origem do Sistema de Coordenadas Global, e α a rotação da reta perpendicular formada por r em relação ao eixo Z_{GLOBAL} , considerando como ponto zero o eixo X_{GLOBAL} .

Tanto na planta baixa original como no Mapa 2D, ajustado conforme as medidas reais, constatou-se que as paredes e divisórias são perpendiculares entre si. Esta peculiaridade facilitou o cálculo do parâmetro α , após a escolha estratégica dos eixos do sistema de coordenadas global.

Como se pode ver na Figura 4.15, as linhas geométricas escolhidas no Mapa 2D são paralelas ou são perpendiculares em relação ao eixo X_{GLOBAL} . A vantagem oferecida pela facilidade de construção do *feature-based map*, devido ao padrão na orientação das linhas, resulta em uma redução de precisão na correção da estimativa de localização, já que o EKF fará a correção da estimativa de localização em função dos parâmetros (r, α) . Em outras palavras, tomando como exemplo o sistema de coordenadas global escolhido, se no instante t o robô reconhece somente linhas paralelas ao eixo X_{GLOBAL} , será corrigida a posição do robô móvel apenas em relação ao eixo Y_{GLOBAL} .

Por essa razão, durante a seleção das linhas geométricas no mapa 2D devem ser escolhidos *landmarks* naturais paralelos ao eixo X_{GLOBAL} e *landmarks* naturais paralelos ao eixo Y_{GLOBAL} equilibradamente, pois a posição dos *landmarks* naturais escolhidos irá influenciar na precisão da posição estimada após a etapa de Correção do Sistema de Auto-localização.

Formas geométricas complexas podem corrigir a posição com relação aos dois eixos do sistema de coordenadas global, porém ocorrem em menor número no ambiente, podendo chegar a ser únicas. Logo, quando são reconhecidas pelo algoritmo de extração de características do ambiente, tem-se uma facilidade no processo de *Matching*. Durante este processo cada característica extraída do ambiente é comparada com todas as características mapeadas (*loop* de comparação), se correspondida com alguma característica mapeada é utilizada na próxima etapa do algoritmo de localização.

Já formas geométricas simples, tais como linhas, ocorrem em maior número, podendo haver múltiplas formas do mesmo tipo no ambiente e também múltiplas formas do mesmo tipo mapeadas. Desse modo quando uma característica extraída é comparada com as características mapeadas podem surgir diversas opções de casamento com probabilidades diferentes para a correspondência entre as características e que dependem da localização estimada do robô móvel.

Se uma forma geométrica é única no mapa o processo de *Matching* é direto, ou seja, independe da localização estimada atual do robô móvel. Entretanto, isto

pode demandar um algoritmo de extração com um grau de complexidade muito elevado, que pode comprometer o desempenho do algoritmo de auto-localização. Além disso, o ambiente deverá conter estas formas de identidade única para que o processo de Correção seja bem sucedido.

Por esse motivo, na prática são utilizadas as formas geométricas mais simples, as quais aparecem em abundância em ambientes *indoor*. Para corresponder uma característica lida com uma característica mapeada é aplicada uma tolerância para a diferença entre os valores dos parâmetros da forma geométrica lida e os valores dos parâmetros da forma geométrica virtual construída com base na posição estimada atual do robô móvel.

A tolerância de correspondência escolhida irá determinar a robustez do sistema de auto-localização de duas maneiras. Se a tolerância aplicada for muito alta tem-se um aumento na probabilidade de corresponder erradamente uma característica lida com uma característica mapeada. Por outro lado, se a tolerância escolhida for muito baixa, quando o erro acumulado na posição estimada ultrapassar a tolerância aplicada não ocorrerá correspondência de uma característica lida que está mapeada, conseqüentemente, a posição não será corrigida nesse instante. Uma tolerância maior também permite a retomada da estimativa de posição após um “salto” na estimativa de posição ocasionado por erros não previstos no mapa, nos dados sensoriais ou no processamento do algoritmo.

Seguindo este raciocínio, conclui-se que dependendo da posição relativa entre cada característica mapeada pode-se aplicar com segurança uma tolerância de correspondência individual para cada característica sem ter perdas significativas na robustez do algoritmo de localização. Esta tolerância individual poderia ser registrada no *feature-based map* junto com os parâmetros geométricos que definem a característica inserida no *feature-based map*. Resumindo, pode-se dizer que a escolha da tolerância de correspondência aplicada deve ser coerente com a construção do *feature-based map*, se possível definido de forma conjunta.

Considerando as questões envolvidas na definição de uma tolerância de correspondência, não é recomendado incluir no *feature-based map*, *landmarks* naturais posicionados, entre si, a uma distância menor ou igual ao valor atribuído para a tolerância de correspondência aplicada.

As linhas não recomendadas são indicadas na Figura 4.16 e diferenciadas pela cor roxa. Uma linha não recomendada foi inserida inicialmente propositalmente

no *feature-based map* com o objetivo de avaliar a influência da mesma na estimativa de localização durante a execução do algoritmo de auto-localização com dados reais. Esta linha é indicada como “teste” na Figura 4.16 e corresponde a um trecho da divisória interna das salas. Enquanto o robô móvel transita pela área navegável definida, ambos os lados da divisória deste trecho da divisória poderão ser reconhecidos pelo algoritmo de extração de linhas e correspondidos de forma invertida com as linhas cadastradas no *feature-based map*.

Figura 4.16: Linhas geométricas não recomendadas para a inclusão no *feature-based map*.

Essa inversão poderia ser evitada no processo de *Matching* se os *landmark* naturais cadastrados no *feature-based map* fossem classificados como aptos ou não de serem reconhecidos e, em seguida, filtrados antes do *loop* de comparação. Em outras palavras, seriam computados no *loop* de comparação somente os *landmarks* naturais cadastrados que poderiam ser reconhecidos pelo sensor de medição laser na posição estimada atual do robô móvel. Este mecanismo não foi testado devido à sua complexidade, portanto requer uma análise mais elaborada, ficando como sugestão para trabalhos futuros.

Após a seleção das linhas no Mapa 2D, são, então, calculados os parâmetros (r, α) . Um erro no cálculo das cotas relativas gerará um erro no *feature-based map*, logo a utilização de um mapa 2D no formato CAD diminui este risco, já que as distâncias relativas, com relação ao sistema de coordenadas global, são

fornecidas de forma direta sem a necessidade de cálculo, através de comandos disponíveis na própria ferramenta CAD.

Linhas Horizontais			Linhas Verticais		
id.	r (m)	α (rad)	id.	r (m)	α (rad)
01	-0,83	$\pi/2$	10	4,07	0
02	0,00	$\pi/2$	11	4,27	0
03	1,56	$\pi/2$	12	8,27	0
04	2,39	$\pi/2$	13	9,15	0
05	5,91	$\pi/2$	14	9,19	0
06	7,66	$\pi/2$	15	10,07	0
07	7,67	$\pi/2$	16	10,68	0
08	7,70	$\pi/2$	17	13,30	0
09	12,33	$\pi/2$	18	14,18	0
			19	14,21	0
			20	14,22	0
			21	14,25	0
			22	14,27	0
			23	15,72	0
			24	15,73	0
			25	16,54	0
			26	18,20	0
			27	18,97	0
			28	19,08	0
			29	19,12	0
			30	20,00	0
			31	21,63	0
			32	24,14	0

Quadro 4.1: Feature-based map composto por características do tipo linhas geométricas definidas em função dos parâmetros (r , α).

Os valores do conjunto de parâmetros (r , α) em função do sistema de coordenadas global são apresentados no Quadro 4.1. O conjunto é formado pelos segmentos de linhas pertencentes ao *feature-based map* após o teste do Sistema de

Auto-localização com o conjunto inicial. O teste inicial demandou ajustes no mapa, bem como a inclusão de novos *features*, inclusive alguns inicialmente não recomendados. No Quadro 4.1 foi atribuída uma identificação (id) a cada linha geométrica representada pelos parâmetros (r, α) , esta identificação é utilizada pelo algoritmo de localização durante o processo de *Matching* para validar uma correspondência entre uma característica lida e um *feature* mapeado. O *feature-based map* final é composto pela lista de *features*, cada um com uma identificação e seus respectivos valores de (r, α) .

4.2.2 Coleta dos dados sensoriais reais e pós-processamento

Figura 4.17: Robô móvel Pioneer 3-AT.

Para a coleta dos dados sensoriais reais foi utilizado o robô móvel Pioneer 3-AT da Figura 4.17 (ROBÔS, 2011). Sob a plataforma do robô móvel foram montados o sensor de medição laser 2D SICK LMS-200 e um notebook Sony Vaio VGN-FW31J. A comunicação, entre o robô móvel e o notebook, é feita através de um cabo USB com conversor Serial-USB. O sensor de medição laser 2D também é conectado ao notebook por meio de um segundo cabo USB com conversor Serial-USB. Este notebook com Linux Ubuntu instalado utiliza o Player/Stage para controlar os motores do robô móvel e para registrar os dados sensoriais em um *log*. O controle do robô móvel durante a navegação é feito remotamente utilizando um

segundo notebook HP G42-215BR que se comunica com o notebook montado no robô móvel através de uma rede sem fio utilizando a ferramenta PlayerViewer do pacote do Player/Stage.

A montagem da rede sem fio no Linux Ubuntu não será detalhada neste trabalho por tratar-se de uma rede sem fio comum, ou seja, foi configurada sem parâmetros especiais para o contexto da aplicação com robôs móveis. Entretanto, se esta aplicação não fosse apenas para teste seriam requeridos parâmetros especiais de segurança para evitar acidentes provenientes de acessos não autorizados. Entretanto, um fator importante durante o experimento foi o posicionamento do roteador *wireless* para a criação da rede sem fio. O roteador foi posicionado estrategicamente de modo que o alcance do sinal abrangesse toda a área navegável, a posição escolhida para o roteador que atendeu esta necessidade é mostrada na Figura 4.18.

Figura 4.18: Posicionamento do roteador wireless para a coleta dos dados sensoriais.

O algoritmo de auto-localização necessita que seja fornecida uma localização inicial para calcular a primeira estimativa de localização, esta dependência é uma desvantagem comum aos algoritmos baseados no Filtro de Kalman e suas variações. Além disso, o processo de *Matching* também utiliza a primeira localização estimada para corresponder os *landmarks* naturais mapeados,

por esses motivos, a obtenção da localização inicial do robô móvel no ponto de partida dos trajetos previstos para os experimentos no ambiente real foi necessária.

Figura 4.19: Medidas coletadas em milímetros para o cálculo da localização inicial no ponto de partida do trajeto do experimento n° 01.

Figura 4.20: Medidas coletadas em milímetros para o cálculo da localização inicial no ponto de partida do trajeto do experimento n° 02.

As Figuras 4.19 e 4.20 mostram as medidas coletadas em milímetros para o cálculo da posição inicial no ponto de partida dos experimentos n° 01 e n° 02, respectivamente. Como a localização deve ser fornecida na mesma base do sistema de coordenadas global, foi calculada medindo a distância da plataforma do robô móvel em relação a alguns elementos do ambiente próximos à origem do sistema de coordenadas global. Devido à medição ter sido feita somente com o auxílio de uma trena, tem-se um desvio com relação à localização real por conta da precisão das medidas, principalmente em relação ao cálculo da orientação inicial do robô móvel. Entretanto, sendo este desvio inferior à tolerância de correspondência aplicada, a posição é gradativamente corrigida pelo algoritmo de auto-localização a cada *update*.

Para a criação de um *log* durante a coleta de dados do ambiente utilizou-se o driver *writelog* e para a leitura dos dados sensoriais em tempo real a partir do *log*, o driver *readlog*, ambos do Player/Stage. Além destes, também foram necessários os drivers *position2d* e *laser* para a leitura dos valores da odometria e do conjunto de pontos do sensor de varredura laser, respectivamente, sendo necessários tanto para a leitura de dados do ambiente como para o pós-processamento dos dados a partir do *log*.

Com o *feature-based map* e o *log* contendo os dados sensoriais, adquiridos no ambiente *indoor* real, em mãos, o Sistema de Auto-localização desenvolvido em Linguagem C, utilizando as bibliotecas do Player/Stage, é finalmente testado e o *feature-based map* é ajustado. Os resultados obtidos no teste final do Sistema de Auto-localização, após ajustes no mapa e nas tolerâncias de correspondência, são apresentados e comentados no item 5.2 do capítulo a seguir.

5 RESULTADOS

Os algoritmos desenvolvidos foram testados nos dois cenários apresentados no capítulo 4 (Ambiente Simulado e Ambiente Real) com o objetivo de validar o Sistema de Auto-localização proposto. Os resultados obtidos em cada cenário serão apresentados neste capítulo em dois tópicos principais.

No Ambiente Simulado, o algoritmo de localização foi executado em paralelo aos algoritmos de navegação e roteamento, fornecendo valores de posição e orientação para o algoritmo de navegação checar os pontos intermediários da rota. Os valores de posição e orientação fornecidos pela odometria do Player/Stage, a odometria acrescida de erros e os valores estimados pelo EKF foram armazenados em um *log*. Além desses valores, também foram registrados no *log* os *landmarks* detectados e identificados a cada *update* do EKF, permitindo a obtenção de um perfil de localização e de visibilidade dos *landmarks* durante o trajeto executado. A análise deste perfil será apresentada no item 5.1.

No item 5.2 é apresentado um comparativo entre a localização fornecida somente com base na odometria e a localização fornecida pelo EKF para cada trajeto testado no Ambiente Real. Não é feito um comparativo em relação à localização real do robô móvel devido à falta de um sistema mais preciso durante o experimento que forneça esta informação.

5.1 Resultados do Ambiente Simulado

O algoritmo de localização baseado no EKF foi implementado em Linguagem C utilizando as bibliotecas disponíveis no *software* Player/Stage, que emulam a comunicação entre os dispositivos e sensores do robô configurado para o ambiente simulado (um armazém, onde ocorrem operações de carga e descarga). O algoritmo de roteamento calcula a melhor rota para a empilhadeira robótica, fornecendo uma série de pontos intermediários (conhecidos, em inglês, como *waypoints*) que são interpretados e validados pelo algoritmo de navegação, que

necessita do fornecimento dos dados de posição e orientação do EKF para checar se os pontos intermediários da rota designada foram alcançados. O destino final da rota designada é verificado quando o último *waypoint* é alcançado e confirmado pelo algoritmo de navegação através do EKF.

Para testar a robustez do algoritmo de localização foi simulada uma tarefa de carga e descarga, conforme mostra a Figura 5.1 em partes: (a) O robô sai da base; (b) o robô faz a retirada do produto na prateleira; (c) o robô descarrega o produto na área de expedição; (d) o robô retorna para a base. Durante a execução da rota calculada para esta tarefa o algoritmo do EKF foi solicitado 15.714 vezes gerando uma estimativa da posição (x , y , θ) a cada *update*.

Figura 5.1: Tarefa simulada de carga e descarga utilizando o Player/Stage.

Durante a simulação o EKF mostrou-se bastante robusto em relação aos erros de odometria, já que o robô não colidiu com as paredes ou prateleiras, manteve a rota pré-determinada e os valores calculados pelo filtro probabilístico não divergiram. O Quadro 5.1 apresenta o erro mínimo e máximo, a média e o desvio padrão entre a localização simulada real (sem erro) e a localização estimada pelo EKF. Pode-se ver no Quadro 5.1 que o erro máximo de posição no plano XY foi de 5,7cm no eixo X e 3,7cm no eixo Y, e o erro máximo de orientação foi de 0,285rad,

todos em relação ao sistema de coordenadas global. O erro mínimo dos três parâmetros foi zero. O ângulo de orientação foi o que apresentou a maior variabilidade já que o desvio padrão encontrado foi notavelmente superior à média.

	X (m)	Y (m)	Theta (rad)
Min.	0	0	0
Máx.	0,057	0,037	0,285
Média	0,020	0,012	0,0015
Desvio Padrão	0,0138	0,0136	0,01569

Quadro 5.1: Diferença entre a localização Real e a EKF (erro absoluto).

Os valores de posição armazenados no *log* durante a execução do EKF foram plotados em escala e são mostrados nas Figuras 5.2, 5.3 e 5.4. A Figura 5.2 mostra aos valores de posição gerados pelo Player/Stage através de sua função de odometria. Já na Figura 5.3 são apresentados os valores de posição gerados pelo Player/Stage acrescidos de erro aleatório e na Figura 5.4 os valores de posição retornados pelo EKF. Nas três figuras os valores formam um traçado que se aproxima do trajeto do robô empilhadeira durante o teste.

Figura 5.2: Valores de posição gerados pelo Player/Stage.

Em função do erro aleatório adicionado à odometria ser brando e previsível, não é possível visualizar grande diferença entre os trajetos azul (Figura 5.2) e vermelho (Figura 5.3). Por outro lado, visando tornar o erro gerado mais próximo do real, o erro foi calculado tendo como referência o deslocamento, ou seja, nos trajetos retilíneos onde houve apenas deslocamento no eixo X, por exemplo, o erro calculado gerou diferença somente nesta variação, desse modo os valores plotados no trajeto retilíneo acabam se sobrepondo. Além disso, a escala das três figuras é desfavorável à visualização das diferenças nos trajetos, mesmo quando sobrepostas, já que as figuras representam um ambiente de 50 x 30m e o erro máximo encontrado na estimativa do EKF foi de 5,7cm no eixo X, por exemplo, como visto no Quadro 5.1.

Figura 5.3: Valores de posição gerados pelo Player/Stage com adição de erro aleatório.

Os mesmos motivos se aplicam ao traçado verde (Figura 5.4) que corresponde aos valores estimados pelo EKF. Além disso, o EKF não deveria gerar uma estimativa pior do que a gerada pela odometria, já que a sua função é obter uma estimativa melhor utilizando os *landmarks*. Adicionalmente, a ausência de pontos verdes na Figura 5.4 que imita um gráfico, em locais longes ao trajeto, indica que o EKF não gerou valores de posição absurdos, tal fato significaria uma falha nos cálculos do código em linguagem C ou na modelagem matemática das equações do algoritmo do EKF, ou simplesmente um *bug* no Sistema de Auto-localização

proposto. Sendo assim, a ausência de “pontos verdes isolados” mostra que o EKF foi robusto durante o teste.

Figura 5.4: Valores de posição estimados pelo EKF.

O Quadro 5.2 apresenta uma análise do número de *landmarks* detectados e identificados durante o trajeto simulado e tem como objetivo avaliar de modo geral a distribuição (quantidade e posicionamento) dos *landmarks* no ambiente. Em um sistema de AGVs real um número baixo de *landmarks* pode resultar na perda da localização e um número alto custos na implantação e manutenção. Durante o trajeto pelo menos um *landmark* foi detectado pelo robô, sendo que em alguns pontos foram detectados dez simultaneamente, considerando todo o percurso em média 5,4 foram detectados.

A coluna Identificados no Quadro 5.2 mostra os *landmarks* identificados durante a simulação. Um *landmark* pode ser detectado, mas não necessariamente identificado, isto reflete a situação em que o robô “sabe” que existe um *landmark* dentro do alcance de seu sensor, mas não tem informações suficientes para identificar qual é o *landmark* mapeado que está sendo “visto”. Desse modo, o EKF somente poderá utilizar os *landmarks* identificados para melhorar a estimativa de localização. Na teoria quanto maior o número de *landmarks* identificados a cada instante, melhor seria a estimativa resultante gerada pelo EKF, entretanto seria necessário um número exaustivo de testes, preferencialmente, com dados reais para obter o número ideal de *landmarks* detectados, ficando esta tarefa como proposta

para trabalhos futuros. A coluna Identificados/Detectados fornece uma estimativa da proporção de *landmarks* identificados dado um número de *landmarks* detectados, em algumas situações todos os *landmarks* detectados são identificados (100%) e em outras nenhum (0%), em média 44,09% foram identificados considerando todo o trajeto.

	Detectados	Identificados	Identificados/ Detectados (x100%)
Min.	1	0	0%
Máx.	10	5	100%
Média	5,40	2,30	44,09%
Desvio Padrão	1,647	1,014	18,63%

Quadro 5.2: Número de *landmarks* visualizados durante o trajeto simulado.

No Quadro 5.2 o número mínimo de identificação foi zero, o máximo cinco e a média 2,3. O número zero significa que em alguns pontos do trajeto o robô não pôde melhorar sua estimativa de localização. Nesta situação, o robô ficou cego dependendo apenas da odometria para computar a sua localização. Uma proporção de quantas vezes essa situação ocorreu pode ser vista no Gráfico 5.1.

Gráfico 5.1: Visibilidade dos *landmarks*.

O eixo X do Gráfico 5.1 corresponde ao número de *landmarks* detectados ou identificados. As linhas Detectados e Identificados no quadro abaixo do gráfico mostram os valores das colunas no eixo Y, que corresponde ao número de *updates* do EKF durante o trajeto testado. Duas informações relevantes podem ser extraídas do Gráfico 5.1. Primeira, a maior concentração de *landmarks* identificados varia entre um e três por EKF *update*, totalizando 13.472 ou 85,72% do total. Segunda, zero *landmarks* foram identificados em 31 *updates*, que corresponde a somente 0,20% do trajeto testado, assumindo que todos os *updates* foram processados na mesma velocidade.

O Gráfico 5.2 mostra a proporção de ocorrência da taxa de *landmarks* Identificados/Detectados, a taxa com maior ocorrência é a taxa de 0,5 ou 50% de identificação. As taxas calculadas para os 15.714 EKF *updates* foram ordenadas em ordem decrescente para obter o gráfico apresentado.

Gráfico 5.2: Taxa de proporção entre os landmarks detectados/identificados.

5.2 Resultados do Ambiente Real

5.2.1 Experimento 1

Os valores de posição, fornecidos pelo Player/Stage a partir da odometria do robô Pioneer P3-AT e coletados do ambiente real de acordo com o trajeto planejado para o Experimento 1, ilustrado na Figura 4.11, foram sobrepostos no mapa do ambiente de 32 x 18 m de modo a visualizar o comportamento dos valores de odometria. A sequência destes valores, obtida a partir da junção de dois *logs* devido à parada realizada para a recarga das baterias, foi sobreposta ao longo do trajeto no mapa em escala reduzida mostrado na Figura 5.5.

A maior parte dos valores de odometria obtidos após o reinício do robô estende-se para posições que vão além dos limites do mapa e não são visíveis na Figura 5.5, pois na junção dos *logs* foi considerado o erro de odometria já acumulado no primeiro trecho do trajeto. O reinício da estimativa de odometria ocorre a partir da posição indicada na Figura 5.5. Uma análise mais minuciosa sobre o comportamento da odometria do robô Pioneer é realizada com os valores obtidos no trajeto do Experimento 2.

Figura 5.5: EXPERIMENTO 1 – Sequência dos valores de posição XY estimados pela odometria real a partir da junção de dois logs.

Os valores de posição estimados pelo EKF durante o início do trajeto do Experimento 1 são apresentados na Figura 5.6. Durante este trecho o EKF foi capaz de completar ao menos um *match* bem sucedido com *features* na posição vertical

e/ou horizontal. Nas condições do primeiro experimento no qual o robô foi conduzido por um trecho com bateria baixa (aviso sonoro do sensor SICK LMS-200), antes do ponto escolhido para a parada, pode representar uma fonte de erro nos dados sensoriais. Além disto, uma manobra de giro no próprio eixo de rotação do robô que resulta em pequenas derrapagens na posição indicada na Figura 5.5. Juntos, resultaram em um acúmulo de erro que o EKF não foi capaz de compensar utilizando os mecanismos desenvolvidos para o Sistema de Auto-localização. A funcionalidade destes mecanismos, bem como a sensibilidade do EKF em manter a estimativa de localização com os dados reais, são explicados na análise dos valores processados no Experimento 2.

Figura 5.6: EXPERIMENTO 1 – Sequência dos valores de posição XY estimados pelo EKF em parte do primeiro trecho.

5.2.2 Experimento 2

Diferentemente do Experimento 1, os dados do Experimento 2 foram coletados em um intervalo de tempo menor e sem paradas longas. E para conseguir uma navegação mais continua antes do esgotamento das baterias, o robô foi

conduzido por um trajeto menos extenso com uma velocidade ligeiramente superior à velocidade do Experimento 1. Os valores de posição, fornecidos pelo Player/Stage a partir da odometria do robô Pioneer P3-AT e coletados do ambiente real de acordo com o trajeto planejado para o Experimento 2 ilustrado na Figura 4.12, foram sobrepostos no mapa do ambiente de 32 x 18 m de modo a visualizar o comportamento dos valores de odometria isoladamente. A sequência destes valores sobrepostos ao longo do trajeto no mapa em escala reduzida é mostrada na Figura 5.7. Para uma melhor visualização da sequência de pontos (posições), o trajeto do Experimento 2 foi dividido em dois trechos: ida e retorno, a sequência parcial de cada trecho é mostrada nas Figuras 5.8 e 5.9, respectivamente.

Figura 5.7: EXPERIMENTO 2 – Sequência dos valores de posição XY estimados pela odometria real.

De acordo com a interpretação da Figura 5.7, entende-se que a odometria do experimento apresenta um desvio mais acentuado na leitura dos movimentos de rotação ou giro, sendo ainda maior no sentido horário conforme mostra a última mudança de direção no trecho de retorno da Figura 5.9. Outra característica observada é que a odometria não reflete pequenas mudanças de direção ou giros tendendo a mostrá-los como trechos aproximadamente retos. Pode-se observar que também ocorre um desvio no movimento de translação para a frente, pois a Figura

5.8 que indica uma colisão com a parede após a primeira mudança de direção, na verdade refere-se a um trecho no qual o robô foi conduzido através do centro do corredor. Estes desvios ficam mais evidentes quando comparados com o trajeto real, que é aproximado pelas estimativas de posição do EKF descritas a seguir.

Figura 5.8: EXPERIMENTO 2 – Odometria do Trajeto de Ida.

Os valores de posição estimados pelo EKF durante o trajeto completo do Experimento 2 são apresentados na Figura 5.10. Novamente, para uma melhor visualização da sequência de posições fornecidas pelo EKF, os valores foram separados adotando a mesma divisão de dois trechos aplicada à odometria. Os resultados fornecidos pelo EKF nos trajetos de ida e retorno são apresentados nas Figuras 5.11 e 5.12, respectivamente.

Durante a execução do EKF, o processo de *Matching* foi programado para fazer a busca do melhor emparelhamento entre aqueles possíveis, ou seja, a escolha da melhor combinação em pares de *features* lidos com *features* mapeados, cuja diferença entre a posição lida e a posição estimada a partir do mapa seja menor ou igual à tolerância calculada naquela condição. Como os *features* horizontais contribuem somente na correção da estimativa de posição no eixo Y global e os verticais no eixo X global, duas buscas independentes foram aplicadas para a escolha do melhor *match*, uma para os segmentos de linhas horizontais do mapa e

outra para os verticais. Deste critério obtêm-se quatro resultados possíveis a cada *update* do EKF que são: nenhum *match*, um *match* (horizontal ou vertical) e dois *match* (horizontal e vertical).

Figura 5.9: EXPERIMENTO 2 – Odometria do Trajeto de Retorno.

Figura 5.10: EXPERIMENTO 2 – Sequência dos valores de posição XY estimados pelo EKF.

Estas buscas têm como função evitar uma redução no desempenho do EKF por dois motivos, o primeiro é que o *update* do EKF torna-se mais lento conforme aumenta o número de casamentos aplicados na etapa de Correção, sem uma pré-seleção todas as combinações aprovadas seriam processadas durante o *update* do EKF. O segundo é que a matriz dos erros de medição, utilizada também na Correção, é a mesma para todos os *features* extraídos, logo, a estimativa de localização corrigida a partir de um *match* com alta probabilidade de sucesso pode ser empobrecida com uma nova correção a partir de outro *match* com menor probabilidade de sucesso.

Conforme os dados coletados no trajeto planejado vão sendo processados pelo Sistema de Auto-localização, o EKF atualiza a estimativa de localização. A cada *update*, uma nova combinação entre *features* lidos e mapeados é calculada, desse modo o *feature* de referência no mapa, utilizado no melhor emparelhamento da etapa de Correção, pode oscilar. Por exemplo, no primeiro momento do trajeto (1º corredor), os *features* extraídos das duas divisórias laterais (paredes) atendem os critérios de tolerância, logo, ambos seriam válidos para o *Matching*. Entretanto, como a diferença de posicionamento de um deles é menor, apenas um é melhor candidato para a Correção. Desse modo, em um instante a localização é corrigida em função do segmento de linha extraído de uma das paredes e em outro, de acordo com a parede oposta.

Essa variação de referências faz com que o traçado sequencial das posições estimadas (Figuras 5.11 e 5.12) tenha um aspecto descontínuo, tal como pequenos saltos ou trepidações. A amplitude destas discontinuidades é mais evidente em três situações, uma delas ocorre quando se tem uma imprecisão no mapa, que é absorvida pelo EKF, porém resulta em uma oscilação da estimativa de localização. A segunda ocorre quando dois *features* paralelos muito próximos conduzem a um emparelhamento falso positivo caso a distância entre os dois se enquadre na tolerância calculada e o EKF estime o *feature* errado como sendo o melhor candidato ao *Matching*. A terceira situação ocorre quando o *Matching* não obtém resultados horizontais, verticais ou ambos devido à inexistência de *features* mapeados visíveis em determinada área do trajeto ou porque a tolerância calculada não permite a aprovação de qualquer combinação de *features* lidos e mapeados, fazendo com que a localização seja estimada somente com base na odometria por

um determinado N° de ciclos do EKF. Assim que o *Matching* obtém um resultado, após alguns ciclos sem sucesso, a diferença entre a localização corrigida em relação à localização estimada anteriormente provoca um salto na posição estimada.

Figura 5.11: EXPERIMENTO 2 – EKF do Trajeto de Ida.

Figura 5.12: EXPERIMENTO 2 – EKF do Trajeto de Retorno.

Alguns exemplos de saltos na estimativa de localização, devido à falta de *matching* nos eixos X ou Y global, são indicados nas Figuras 5.11 e 5.12. A sigla sX corresponde a um salto na estimativa de localização em relação ao eixo X e a sigla sY a um salto em relação ao eixo Y.

6 CONCLUSÃO E TRABALHOS FUTUROS

Nesta dissertação o Sistema de Auto-localização baseado no EKF foi aplicado na localização de robôs móveis utilizando *landmarks* artificiais simulados e *features* naturais extraídos de um ambiente *indoor*. O algoritmo teve como função fornecer uma localização confiável, a qual possibilitou a execução de um planejamento de trajetória eficiente, evitando bloqueios e colisões com obstáculos mapeados no ambiente simulado e uma localização confiável no ambiente real *indoor* mapeado a partir de *features* naturais.

Embora a utilização de sistemas de localização baseados no EKF não seja uma aplicação recente na robótica móvel, o EKF ainda pode ser considerado um algoritmo com implementação viável, dado o seu comportamento robusto no fornecimento da estimativa de localização de um robô móvel. Do experimento com dados reais obtidos de um ambiente *indoor*, concluiu-se que o maior *drawback* do algoritmo é a inexistência da capacidade de localização global. Esta deficiência exige que no início da execução do algoritmo seja fornecida uma estimativa inicial de posição e orientação. Considerando um armazém como ambiente de operação, isto implica que o robô móvel parta sempre de uma posição com localização conhecida.

Por outro lado, a utilização de segmentos de linha como *features* do ambiente, devido à sua simplicidade e disponibilidade em ambientes *indoor*, também gera ambiguidade, durante a fase de correção do EKF, no processo de comparação dos *features* lidos com os *features* mapeados, que pode resultar em falsos positivos na execução do *Matching*. Com o objetivo de contornar o problema da ambiguidade no Sistema de Auto-localização implementado no teste com dados reais, utilizou-se a última posição estimada como referência no processo comparativo, entretanto, esta abordagem demanda que a estimativa de posição seja constante e, preferencialmente, corrigida em cada ciclo do EKF, pois conforme o erro na estimativa de posição se propaga, a probabilidade de obter um verdadeiro positivo no *Matching* diminui, já que o uso da última posição estimada como referência restringe a correspondência dos *features*. Logo, uma vez que o erro na estimativa de posição atinge um patamar que impede o *Matching* de efetuar uma correspondência positiva verdadeira, o erro da localização estimada se propaga indefinidamente até

que ocorra uma correspondência verdadeira ou falsa. Durante esse instante, pode-se dizer que o *tracking* da estimativa de posição foi perdido.

A existência de um mecanismo de localização global permitiria a retomada do *tracking* da estimativa de posição, já que a posição pode ser corrigida constantemente utilizando os valores fornecidos pelo mecanismo de localização global. Desse modo, nos trechos do mapa onde não ocorra a correspondência dos *features* naturais, a posição seria estimada com base na odometria e corrigida de acordo com os valores de localização global, ou seja, mesmo que a precisão do mecanismo de localização global seja inferior à precisão obtida com a correspondência dos *features* naturais, este, impede que o erro da estimativa de localização se propague além do erro tolerável pelo *Matching* para a obtenção de uma correspondência verdadeira positiva, pois o EKF retornaria um erro máximo próximo do erro de medição do mecanismo de localização global. Sendo assim, fica reforçada a hipótese de que a inexistência da capacidade de localização global no Sistema de Auto-localização baseado em EKF é sua maior deficiência.

A precisão do mapa é uma variável crítica comum entre os algoritmos de localização, entretanto, para o Sistema de Auto-localização, baseado no EKF e na extração de linhas, concluiu-se que ocorre a necessidade de um mapa mais preciso devido à dependência do *tracking* constante da estimativa de localização. Cada *feature* pode não ser visto pelo robô em todos os trechos navegáveis do ambiente mapeado, logo o Sistema de Auto-localização corrige a estimativa de localização correspondendo *features* diferentes em cada trecho do ambiente mapeado. Desse modo, quando ocorrem um ou mais erros no mapeamento de um *feature*, seja de posição ou orientação, a transição da referência utilizada pelo *Matching* de um *feature* para outro pode ser mal sucedida, em outras palavras, a localização estimada em um trecho do ambiente pode não servir para corresponder os novos *features* lidos do ambiente, assim, após os *features* anteriores não serem mais visíveis o erro da estimativa de localização se propaga indefinidamente, como exposto anteriormente.

A definição de uma tolerância para as diferenças de posição e de orientação entre os *features* mapeados e os *features* extraídos do ambiente pode ser aplicada para absorver possíveis erros no mapa. Não existindo um padrão para estes valores, concluiu-se que estes valores devem ser obtidos experimentalmente, pois dependem da complexidade e da extensão do ambiente mapeado, neste sentido a

simulação utilizando dados reais mostrou-se ser uma ferramenta bastante assistiva na obtenção destes parâmetros, pois reduz o número de experimentos necessários para o teste do algoritmo no ambiente real. O cálculo automático destes valores pode tornar-se objeto de estudo em trabalhos futuros, já que a implementação em armazéns inteligentes exigiria uma nova estimativa de parâmetros para cada novo ambiente mapeado.

Adicionalmente, conclui-se que quanto melhor é a aproximação dos erros de movimento e de medição, melhor é o desempenho do Sistema de Auto-localização baseado no EKF. Embora o funcionamento robusto do EKF permita a seleção de valores arbitrários para os erros de movimento e de medição, dentro de uma escala na qual os erros de medição sejam proporcionalmente inferiores aos erros de movimento, permitindo, assim, que a covariância estimada seja reduzida na etapa de correção, uma melhor seleção de valores irá incrementar o desempenho do sistema, já que este depende dos valores computados na matriz de covariância do EKF.

Em outras palavras, os valores de erro utilizados no algoritmo interferem diretamente no cálculo da matriz de covariância, desse modo, os erros de movimento escolhidos determinarão como o erro da estimativa de localização se propaga. Como o Sistema de Auto-localização utiliza os valores da matriz de covariância do EKF no cálculo das tolerâncias do *Matching*, estes influenciaram diretamente a ocorrência de correspondências verdadeiras positivas, consequentemente, o desempenho do Sistema, como explicado anteriormente. Por outro lado, o erro de medição dos *features* extraídos do ambiente pode ser diferente, porque pode depender do número de pontos que compõem o segmento de linha, da distância relativa entre o robô e o *feature* ou, ainda, da última covariância estimada. Dessa maneira, uma melhor aproximação do erro de medição de cada *feature* extraído irá modificar a matriz de covariância e a estimativa de localização com mais coerência, resultando em um desempenho superior.

Finalmente, tanto os resultados produzidos na simulação como os resultados obtidos a partir de dados de um ambiente real mostraram que os métodos de localização empregados no Sistema de Auto-localização são válidos para uma aplicação de AGVs em armazéns inteligentes. Os resultados demonstram que a precisão obtida na estimativa de localização atende aos requisitos do planejamento de trajetória de uma empilhadeira robótica para um armazém inteligente simulado e

para um ambiente real *indoor*, ficando a confirmar esta informação, em trabalhos futuros, para um armazém inteligente real.

Conforme exposto nos capítulos 1 e 2, sabe-se que a navegação dos AGVs dentro do armazém é limitada pela precisão do sistema de auto-localização e muitas vezes para obter a precisão requerida é necessário um investimento financeiro alto e, dependendo da aplicação, o investimento inicial aplicado na implantação de um sistema de AGVs é inviável. Parte deste investimento advém do custo dos sensores utilizados para a execução da tarefa de auto-localização ou, ainda, do custo de instalação de *landmarks* artificiais no ambiente de operação. Deste modo, o Sistema de Localização proposto neste trabalho para um sistema de AGVs baseado somente em odometria e no reconhecimento de *features* naturais do ambiente, pode ser considerado economicamente mais viável do que outros sistemas de localização envolvendo um número maior de recursos tecnológicos.

Como propostas para trabalhos futuros temos a utilização de outros sensores e técnicas de localização, tais como a extração de *landmarks* naturais utilizando visão ou a utilização de um sensor de marcadores refletivos para o reconhecimento de *landmarks* artificiais dispostos de forma estruturada no ambiente. Bem como, o desenvolvimento de sistemas de localização baseados em outros algoritmos probabilísticos. Como alternativa para a melhora do desempenho do Sistema de Localização desenvolvido, fica proposta a utilização de ferramentas de programação e hardware específicos, com o objetivo de obter um sistema com as características de um sistema embarcado de tempo real.

Além das propostas acima, tem-se como trabalho futuro a implantação do Sistema de Localização em mini-empilhadeiras robóticas operando em um ambiente real. As mini-empilhadeiras, apresentadas na Figura 6.1, foram desenvolvidas pelo Grupo de Robótica Móvel do Laboratório de Mecatrônica da EESC – USP para a representação de um armazém inteligente em escala reduzida. As mesmas encontram-se montadas e estão sendo utilizadas em outras linhas de pesquisa e frentes de trabalho do grupo do projeto.

Figura 6.1: Mini-empilhadeira Robótica.

7 REFERÊNCIAS

AGVROBOTS.COM. (2011). Forklift & Pallet Handling Automatic Guided Vehicles. Disponível em: <http://www.agvrobots.com/forklift_agv.html>. Acesso em: 08 Ago. 2011.

AUTOMOTIVEBUSINESS (2012). Disponível em: <<http://www.automotivebusiness.com.br>>. Acesso em: 22 Abr. 2012.

AYACHE, N.; FAUGERAS, O. D. (1989). Maintaining representations of the environment of a mobile robot. **IEEE Transactions on Robotics and Automation**, New York, v.5, n.6, p.804-819.

BARBERA, H.M. et al. (2003). i-Fork: a flexible AGV system using topological and grid maps. In: 2003 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 20., 2003, Taipei. **Proceedings...** New York: IEEE, v.2, p.2147-2152.

BETKE, M.; GURVITS, L. (1994). Mobile Robot Localization using Landmarks. In: IEEE/RSJ/GI INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 7., 1994, Munich. **Proceedings...** New York: IEEE, p.135-142.

BISHOP, R. H. (2002). **The Mechatronics Handbook**. New York: CRC Press.

BORENSTEIN, J.; EVERETT, B.; FENG, L. (1996). **Navigating Mobile Robots: Systems and Techniques**. Wellesley, MA: A. K. Peters, Ltd.

BORGES, G. A.; ALDON, M. (2004). Line extraction in 2D range images for mobile robotics. **Journal of Intelligent & Robotic Systems**, Dordrecht, v.40, n.3, p.267-297.

BOUGUERRA, A et al. (2009). An autonomous robotic system for load transportation. In: IEEE CONFERENCE ON EMERGING TECHNOLOGIES FACTORY AUTOMATION, 14., 2009, Mallorca. **Proceedings...** New York: IEEE, p.1-4.

BURGARD, W. et. al (1996). Estimating the absolute position of a mobile robot using position probability grids. In: THIRTEENTH NATIONAL CONFERENCE ON

ARTIFICIAL INTELLIGENCE, 13., 1996, Portland. **Proceedings...** Menlo Park: AAAI Press, p.896-901.

CARICATO, P.; GRIECO, A. (2005). Using simulated annealing to design a material-handling system. **IEEE Intelligent Systems**, New York, v.20, n.4, p.26-30.

CASSANDRA, A. R.; KAEHLING, L. P.; KURIEN, J. A. (1996). Acting under uncertainty: discrete Bayesian models for mobile-robot navigation. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 9., 1996, Osaka. **Proceedings...** New York: IEEE, v.2, p.963-972.

CASTELLANOS, J. A.; TARDÓS, J. D. (1996). Laser-based segmentation and localization for a mobile robot. In: M. JAMSHIDI, M.; PIN, F.; DAUCHEZ, P. **Robotics and Manufacturing: Recent Trends in Research and Applications**. New York: ASME Press, v.6, p.101-108.

CROWLEY, J. L. (1985). Navigation for an intelligent mobile robot. **IEEE Journal of Robotics and Automation**, New York, v.1, n.1, p.31-41, Mar.

_____. (1989). World modeling and position estimation for a mobile robot using ultrasonic ranging. In: 1989 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 6., 1989, Scottsdale. **Proceedings...** New York: IEEE, v.2, p.674-680.

DAM, K. H. VAN et al. (2004). Intelligent infrastructures: distributed intelligence in transport system control - an illustrative example. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2004, The Hague. **Proceedings...** New York: IEEE, v.5, p.4650-4654.

DELLAERT, F. et al. (1999). Monte Carlo localization for mobile robots. In: 1999 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 16., 1999, Detroit. **Proceedings...** New York: IEEE, v.2, p.1322-1328.

DESROSIERS, J. et al. (1986). Methods for routing with time windows. **European Journal of Operational Research**, Amsterdam, v.23, n.2, p.236-245, Feb.

DIAZ, C. P. O. (2010). **Estratégia de mapeamento em 2D de ambientes internos baseada na extração de segmentos de linhas usando sensores de ultrassom**. 75p. Dissertação (Mestrado) – Universidade de Brasília, Brasília, 2010.

DICKMANNS, E. D. (1987). 4D-dynamic scene analysis with integral spatio-temporal models. In: 4th INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH, 4., 1987, Santa Cruz, **Proceedings...** Cambridge: MIT Press, p.311-318.

DIJSKTRA, E. W. (1959). A Note on two problems in connexion with graphs. **Numerische Mathematik**, Berlin, v.1, p.269-271, Dec.

DIOSI, A.; KLEEMAN, L. (2004). Advanced Sonar and Laser Range Finder Fusion for Simultaneous Localization and Mapping. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 17., 2004, **Proceedings...** New York: IEEE, v.2, p.1854-1859.

DOUGLAS, D. H.; PEUCKER, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. **Cartographica: The International Journal for Geographic Information and Geovisualization**, Toronto, v.10, n.2, p.112-122, December.

DUDA, R. O.; HART, P. E. (1973). **Pattern Recognition and Scene Analysis**. New York: John Wiley & Sons. ISBN 0-471-22361-1.

EINSELE, T. (1997). Real-time self-localization in unknown indoor environment using a panorama laser range finder. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 10., 1997, Grenoble. **Proceedings...** New York: IEEE, p.697-702.

FABRIZI, E et al. (1998). A KF-based localization algorithm for nonholonomic mobile robots. In: 6th IEEE MEDITERRANEAN CONFERENCE ON CONTROL AND SYSTEMS, 6., 1998, Alghero. **Proceedings...** New York: IEEE, p.1-6.

FAZLOLLAHTABAR, H.; MAHDAVI-AMIRI, N. (2010). A bi-criteria AGV-based flexible jobshop manufacturing network having uncertain parameters. In: SECOND INTERNATIONAL CONFERENCE ON ENGINEERING SYSTEMS MANAGEMENT AND ITS APPLICATIONS, 2., 2010, Sharjah. **Proceedings...** New York: IEEE, p.1-6.

_____.; ZANDIEH, M. (2010). Cost optimization in an AGV-based automated manufacturing system associated with product inspection and machine reliability. In: SECOND INTERNATIONAL CONFERENCE ON ENGINEERING SYSTEMS MANAGEMENT AND ITS APPLICATIONS, 2., 2010, Sharjah. **Proceedings...** New York: IEEE, p.1-6.

FOX, D. (1998). **Markov localization: a probabilistic framework for mobile robot localization and navigation**. 146p. Tese (Doutorado) – University of Bonn, Germany, 1998.

_____.et al. (1999). Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In: THE SIXTEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 16., 1999, Orlando. **Proceedings...** Menlo Park: AAAI Press, p.343-349.

_____.; BURGARD, W.; THRUN, S. (1999) Markov localization for mobile robots in dynamic environments. **Journal of Artificial Intelligence Research**, El Segundo, v.11, p.391-427.

_____. (2002). KLD-Sampling: Adaptive Particle Filters. **Advances in Neural Information Processing Systems, Cambridge**, v.14.

GERKEY B.; VAUGHAN, R. T.; HOWARD, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: INTERNATIONAL CONFERENCE ON ADVANCED ROBOTICS, 11., Coimbra, 2003. **Proceedings...** Portugal: University of Coimbra, p.317-323.

GORDON, N. J.; SALMOND, D. J.; SMITH, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. **IEE Proceedings F**, New York, v.140, n.2, p.107-113, April.

GUTMANN, J. S.; SCHLEGEL, C. (1996). AMOS: comparison of scan matching approaches for self-localization in indoor environments. In: THE FIRST EUROMICRO WORKSHOP ON ADVANCED MOBILE ROBOT, 1., Kaiserslautern, 1996. **Proceedings...** New York: IEEE, p.61-67.

HANDSCHIN, J. E. (1970). Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. **Automatica**, Amsterdam, v.6, n.4, p.555-563, July.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. (1968). A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions of Systems Science and Cybernetics**, New York, v.4, n.2, p.100-107, July.

ISARD, M.; BLAKE A. (1996). Contour tracking by stochastic propagation of conditional density. In: EUROPEAN CONFERENCE ON COMPUTER VISION, 4., Cambridge, 1996. **Proceedings...** Heidelberg: Springer, p.343-356.

_____.; _____. (1998). CONDENSATION - Conditional Density Propagation for Visual Tracking. **International Journal of Computer Vision**, Heidelberg, v.29, n.1, p.5-28, August.

JETTO, L.; LONGHI, S.; VENTURINI, G. (1999). Development and experimental validation of an adaptive Extended Kalman Filter for the localization of mobile robot. **IEEE Transactions on Robotics and Automation**, New York, v.15, n.2, p.219–229.

KALMAN, R. E. (1960). A new approach to linear filtering and prediction problems. **Transactions of the ASME Journal of Basic Engineering**, New York, n.82, p.35-45. Series D.

KELLY, A. (2009). On The Upside: A Decade of Growth and Innovation [Industrial Activities]. **IEEE Robotics Automation Magazine**, New York, v.16, n.2, p.116-118.

KITAGAWA, G. (1996) Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. **Journal of Computational and Graphical Statistics**, Alexandria, v.5, n.1, p.1-25.

KIVA SYSTEMS. (2012). Disponível em: <<http://www.kivasystems.com>>. Acesso em: 14 Fev. 2012.

KOLLER, D.; FRATKINA, R. (1998). Using learning for approximation in stochastic processes. In: FIFTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 15., 1998, Madison. **Proceedings...** Burlington: Morgan Kaufmann, p. 287-295.

KRIEGMAN, D. J.; TRIENDL, E.; BINFORD, T. O. (1989). Stereo vision and navigation in buildings for mobile robots. **IEEE Transactions on Robotics and Automation**, New York, v.5, n.6, p.792-803, December.

KRÖSE, B.; BUNSCHOTEN, R. (1999). Probabilistic localization by appearance models and active vision. In: 1999 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 16., 1999, Detroit. **Proceedings...** New York: IEEE, v.3, p.2255-2260.

KUIPERS, B. J.; BYUN, Y. (1988). A robust, qualitative method for robot spatial learning. In: SEVENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 7., 1988, Saint Paul. **Proceedings...** Cambridge: AAAI Press, p.774-779.

LARA, C.; ROMERO, L. (2007). Robust Local Localization of a Mobile Robot in Indoor Environments Using Virtual Corners. RUEDA, L.; MERY, D.; KITTLER, J. **Progress in Pattern Recognition, Image Analysis and Applications**. Heidelberg: Springer. v.4756, p.901-910. (Lecture Notes in Computer Science)

LE-ANH, T.; KOSTER, M.B.M DE (2004). **A review of design of automated guided vehicle systems**. Rotterdam: ERIM. Technical Report ERS-2004-030-LIS.

LENGERKE, O. (2010). **Uma arquitetura mecatrônica de navegação para veículos com reboques guiados automaticamente em ambientes de sistemas flexíveis de manufatura**. 203p. Tese (Doutorado) – Instituto Alberto Luiz Coimbra de Pós-graduação e Pesquisa de Engenharia (COPPE), Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2010.

LEONARD, J. J.; DURRANT-WHITE, H.F. (1991). Mobile robot localization by tracking geometric beacons. **IEEE Transactions on Robotics and Automation**, New York, v.7, n.3, p.376-382.

_____.; _____. (1992). **Directed Sonar Sensing for Mobile Robot Navigation**. New York: Springer (The Springer International Series in Engineering and Computer Science, 175).

LIU, G.; YU, W.; LIU, Y. (2006). Resource management with RFID technology in automatic warehouse system. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 19., 2006, Beijing. **Proceedings...** New York: IEEE, p.3706-3711.

LOVEJOY, W. S. (1991). A survey of algorithmic methods for partially observed Markov decision processes. **Annals of Operations Research**, New York, v.28, n.1, p.47-65.

MATARIC, M. J. (1990). Navigating with a rat brain: a neurobiologically-inspired model for robot spatial representation. In: FIRST INTERNATIONAL CONFERENCE ON SIMULATION OF ADAPTIVE BEHAVIOR: FROM ANIMALS TO ANIMATS, 1., 1990, Paris. **Proceedings...** Cambridge: MIT Press, p.169-175.

MATTHIES, L.; SHAFER, S. A. (1987). Error modeling in stereo navigation. **IEEE Journal of Robotics and Automation**, New York, v.RA-3, n.3, p.239-248, June.

MAZZOTTI, B. F. (2009). **Co-projeto de hardware/software do filtro de partículas para localização em tempo real de robôs móveis**. 67p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2009.

MIKLIC, D.; BOGDAN, S.; KALINOVIC, L. (2011). A control architecture for warehouse automation - Performance evaluation in USARSim. In: 2011 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 28., 2011, Shanghai. **Proceedings...** New York: IEEE, p.109-114.

MIRANDA Neto, A. et al. (2009). Self-organizing maps for environments and states mapping of an autonomous navigation system based on monocular vision. In: COBEM 2009: 20TH INTERNATIONAL CONGRESS OF MECHANICAL ENGINEERING, 20., 2009, Gramado. **Proceedings ...** Rio de Janeiro: ABCM, p.1-10.

MORA, M. C. et al. (2003). Factory management and transport automation. In: IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION, 9., 2003, Lisboa. **Proceedings...** New York: IEEE, v.2, p.508-515.

NAV300. (2012). Laser Positioning Sensor. Disponível em: <<http://www.mysick.com/saqara/wrapper.aspx?id=im0006989>>. Acesso em: 17 Abr. 2012.

NDC8 (2012). NDC8 Platform: Navigation technologies. Disponível em: <<http://http://ndc8.com/>>. Acesso em: 10 Jul. 2012.

NGUYEN, V. et al. (2005). A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 18., 2005, Edmonton. **Proceedings...** New York: IEEE, p.1929-1934.

NOURBAKHS, I.; POWERS, R.; BIRCHFIELD, S. (1995). DERSH an office-navigating robot. **AI Magazine**, Menlo Park, v.16, n.2, p.53-60, Summer.

ODA, M. et al. (2009). Logística sustentável: contribuição a processos de gestão. **INTERFACEHS** - Revista de Gestão Integrada em Saúde do Trabalho e Meio Ambiente, São Paulo, v.4, n.1, p.1-6, abr./ ago.

ODAKURA, V. V. V. A. (2007). **Localização de Markov para multirrobôs cooperativos**. 127p. Tese (Doutorado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2007.

OLIVEIRA, M. M. et al. (2011). Battery state estimation for applications in intelligent warehouses. In: 2011 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 28., 2011, Shanghai. **Proceedings...** New York: IEEE, p.5511-5516.

PASQUAL, T. B. (2010). **Rotinas de navegação de robôs móveis - aplicação em armazéns inteligentes**. 42p. Monografia (TCC) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

PAVLIDIS, T.; HOROWITZ, S. L. (1974). Segmentation of plane curves. **IEEE Transactions on Computers**, New York, v.C-23, n.8, p.860-870. August.

PROBABILISTIC ROBOTICS. (2010). PowerPoint slides. Disponível em: <<http://www.probablistic-robotics.org>>. Acesso em: 17 out. 2010.

RAMER, U. (1972). An iterative procedure for the polygonal approximation of plane curves. **Computer Graphics and Image Processing**, Amsterdam, v.2, n.2, p.244-256, November.

RAMER–DOUGLAS–PEUCKER algorithm (2012). In: Wikipedia: the free encyclopedia. Disponível em: <http://en.wikipedia.org/wiki/Ramer–Douglas–Peucker_algorithm>. Acesso em: 12 jul. 2012.

RIBEIRO, M. I.; LIMA, P. (2002). **Markov Localization**. Instituto de Sistemas e Robótica (ISR), Instituto Superior Técnico (IST), Lisboa.

ROBÔS Móveis do LRM. (2011). São Carlos: ICMC-USP. Disponível em: <<http://www.icmc.usp.br/~lrm/robots.html>>. Acesso em: 01 dez. 2011.

SCHIELE, B.; CROWLEY, J. L. (1994). A comparison of position estimation techniques using occupancy grids. In: 1994 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 11., 1994, San Diego. **Proceedings...** New York: IEEE, v.2, p.1628-1634.

SCHNEIDER, M; SALOMON, R. (2009). Localization by superposing beats: laboratory experiments and detection algorithm. In: IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT SIGNAL PROCESSING, 6., 2009, Budapest. **Proceedings...** New York: IEEE, p.31-35.

SIADAT, A.; KASKE, A.; KLAUSMANN, S.; DUFAUT, M.; HUSSON, R. (1997). An optimized segmentation method for a 2D laser-scanner applied to mobile robot navigation. In: 3rd IFAC SYMPOSIUM ON INTELLIGENT COMPONENTS AND INSTRUMENTS FOR CONTROL APPLICATIONS, 3., 1997, Annecy. **Proceedings...** Kidlington: IFAC-Elsevier, p.153-158.

SICK. (2008). **NAV300 Laser Positioning Sensor**. Operating Instructions. Reute, GE: SICK AG. 72 p. Disponível em: <<http://www.mysick.com/saqqara/pdf.aspx?id=im0032036>>. Acesso em: 17 Abr. 2012.

SICK. (2012). **NAV3xx Laser Measurement Technology**. Product Information. Reute, GE: SICK AG. 12 p. Disponível em: <<http://www.mysick.com/saqqara/pdf.aspx?id=im0032036>>. Acesso em: 17 Abr. 2012.

SIEGWART, R.; NOURBAKHSI, I. R.; SCARAMUZZA, D. (2011). **Introduction to Autonomous Mobile Robots**. 2nd ed. Cambridge: The MIT Press.

SIMMONS, R.; KOEING, S. (1995). Probabilistic robot navigation in partially observable environments. In: FOURTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 14., 1995, Montreal. **Proceedings...** San Francisco: Morgan Kaufmann, v.2, p.1080-1087.

TESLIĆ, L.; ŠKRJANC, I.; KLANČAR, G. (2011). EKF-Based Localization of a Wheeled Mobile Robot in Structured Environments. **Journal of Intelligent and Robotic Systems**, Dordrecht, v.62, n.2, p.187-203.

THRUN, S. (2002a). **Robotic Mapping: A Survey**. "Exploring Artificial Intelligence in the New Millennium, Morgan Kaufmann. Editors: Lakemeyer, G., Nebel, B.

_____. (2002b). Particle filters in robotics. In: EIGHTEENTH ANNUAL CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, 18., 2002, Edmonton. **Proceedings...** San Francisco: Morgan Kaufmann, p.511-518.

_____.; BURGARD, W.; FOX, D. (2005). **Probabilistic Robotics**. Cambridge: MIT Press. ISBN 978-0-262-20162-9.

VAN DER MOLEN, G. M.; GEERTS, J. (1991). AGV path planning in docking situations. In: IEE COLLOQUIUM ON AUTONOMOUS GUIDED VEHICLES, 1., 1991, Salford. **Proceedings...** New York: IEEE, p.4/1-4/3.

VIVALDINI, K. C. T. (2010). **Roteamento automático de empilhadeiras robóticas em armazém inteligente**. 98p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

WANG, Q.; MCINTOSH, R.; MILEHAM, A. (2010). A RFID-based automated warehouse design. In: SECOND INTERNATIONAL CONFERENCE ON COMPUTER ENGINEERING AND TECHNOLOGY, 2., 2010, Jodhpur. **Proceedings...** New York: IEEE, v.6, p.359-363.

WANG, J.; ZHANG, N.; HE, Q. (2009). Application of automated warehouse logistics in manufacturing industry. In: ISECS INTERNATIONAL COLLOQUIUM ON COMPUTING, COMMUNICATION, CONTROL, AND MANAGEMENT, 2., 2009, Sanya. **Proceedings...** New York: IEEE, v.4, p.217-220.

WHITE, C. C. (1991). A survey of solution techniques for the partially observed Markov decision process. **Annals of Operations Research**, New York, v.32, n.1, p.215-230.