ANTÔNIO BAPTISTA JÚNIOR

**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA MECÂNICA**

MAPEAMENTO 3-D PARA ROBÔS

São Carlos
2013

ANTONIO BAPTISTA JÚNIOR

Dissertação apresentada ao Departamento de Engenharia Mecânica da 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 das Máquinas e Sistemas

Orientador: Prof. Dr. Glauco Augusto de Paula Caurin

***ESTE EXEMPLAR TRATA-SE DA VERSÃO
CORRIGIDA.
A VERSÃO ORIGINAL ENCONTRA-SE DISPONÍVEL
JUNTO AO DEPARTAMENTO DE ENGENHARIA
MECANICA DA EESC-USP.***

São Carlos
2013

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

B222m Baptista Junior, Antônio
 Mapeamento 3D para robôs / Antônio Baptista Junior;
 orientador Glauco Augusto de Paula Caurin. São Carlos,
 2013.

 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, 2013.

 1. mapeamento robótico 3D. 2. tracking de
 movimentos. 3. extração de plano de fundo. 4.
 alinhamento de nuvens de ponto. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **ANTONIO BAPTISTA JUNIOR.**

Título da dissertação: "Mapeamento 3-D para robôs".

Data da defesa: 13/11/2013

Comissão Julgadora:

Resultado:

Prof. Associado **Glauco Augusto de Paula Caurin** (orientador)
(Escola de Engenharia de São Carlos/EESC)

Aprovado

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

Aprovado

Prof. Dr. Paulo Marcos Aguiar
(Instituto Federal de São Paulo/IFSP)

Aprova do

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

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

DEDICATÓRIA

À minha mãe, Olga Manzini - *in memoriam* -, que me acompanhou em todas as etapas de minha vida e sempre esteve ao meu lado para me ensinar a diferenciar o certo do errado para poder assim agir corretamente. Foi com o apoio e força que ela sempre me deu que aprendi a superar dificuldades e obstáculos. Agradeço por tudo que ela me proporcionou durante sua vida, por seu grande esforço para criar os filhos e netos e, muito especialmente por seus ensinamentos e história de vida.

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Glauco Augusto de Paula Caurin, pela confiança, em mim depositada, pela oportunidade que me foi concedida e por seu desvelo nas orientações para concretização deste trabalho.

Ao Prof. Dr. Marcelo Becker, pelo suporte e orientação no decorrer do desenvolvimento das aplicações práticas.

A todos os colegas do laboratório, que sempre estiveram dispostos a ajudar e colaborar nas etapas do trabalho, especialmente aos alunos de doutorado Leonardo Marquez Pedro e Valdinei Belini.

À equipe do Instituto de Ciências Matemáticas e de Computação (ICMC), Laboratório de Robótica Móvel (LRM), Prof. Dr. Denis Fernando Wolf, Doutorando Leandro Carlos Fernandes e Msc. Alberto Hata, pelo suporte e equipamentos fornecidos para realização dos experimentos desta monografia.

Aos autores e comunidades de software livres, pelo fornecimento de produtos de altíssimo nível científico e tecnológico, como Linux, MRPT, PCL, Player, ROS e GCC.

Aos professores Dr. José Pacheco de Almeida Prado e Eng. Msc. Marcello Cláudio de Gouvêa Duarte, por terem me ajudado e por acreditarem na concretização deste trabalho.

Ao amigo Dr. Glimaldo Marino pelas sugestões e correções do trabalho.

Ao meu pai, Antônio Baptista, pelos ensinamentos, seu carinho e compreensão. Pelo apoio e recepção para que fosse possível concluir este trabalho.

Ao meu tio Aparecido Benedito Manzini e minhas tias Itazil Correa Manzini e Arminda Baptista, pela recepção, carinho e momentos de inspiração durante a conclusão deste trabalho. Agradeço em especial a minha avó Conceição Silveira por toda sua atenção e dedicação. Ao amigo João Luís Guilherme Benassi pelo apoio e atenção no processo de conclusão deste trabalho e pelos vários momentos de reflexão durante todo processo de desenvolvimento deste trabalho.

"Nenhuma decisão sensata pode ser tomada sem que se leve em conta o mundo não apenas como ele é, mas como ele virá a ser."

Isaac Asimov (1920 - 1992)

Resumo

BAPTISTA JÚNIOR, A. (2013). **Mapeamento 3-D para robôs**. Dissertação (mestrado). Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

Na robótica, mapear o ambiente é tarefa importante, porque ela oferece informação para o planejamento e execução de movimentos do robô. Por este motivo, aqui são apresentados estudos que visam a construção de mapas 3-D e técnicas que auxiliam na tarefa de mapeamento. Quando são construídos mapas 3-D, é habilitado para outros pesquisadores e empresas de robótica a desenvolverem trabalhos de análise e planejamento de trajetórias em todos os seis graus de liberdade do corpo rígido que serve para modelar um robô movel, robô manipulador ou robô móvel manipulador. Com uma representação do ambiente em 3-D, é aumentada a precisão do posicionamento do robô em relação ao ambiente e também o posicionamento de objetos que estão inseridos no campo de atuação do robô. Para solucionar o problema de mapeamento são apresentadas técnicas teóricas e suas aplicações em cada caso estudado. Nos experimentos realizados neste trabalho foi adotada a criação de mapas com *grids* (malhas) de ocupação. Vale lembrar, no entanto, que a construção de mapas por malhas de ocupação pressupõe o conhecimento do posicionamento do robô no ambiente. Neste trabalho foram conduzidos três experimentos e seus objetivos são redução de dados provenientes de falhas e redundâncias de informação com utilização de técnicas probabilísticas, detecção de movimento através da técnica de extração de fundo e mapeamento 3-D utilizando a técnica de ponto mais próximo. No experimento cujo o objetivo é reduzir os dados, foi possível reduzir para 4,43% a quantidade de pontos necessários para gerar a representação do ambiente com a utilização do algoritmo deste trabalho. O algoritmo de mapeamento 3-D feito com uso de modelos probabilísticos bem estabelecidos e disponíveis na literatura tem como base a probabilidade de eventos independentes e a proposta do trabalho envolvendo probabilidade *a posteriori*. O experimento de detecção de movimento foi gerado com a utilização da openCV e a tecnologia CUDA e utilizam a técnica do modelo de mistura gaussiana (GMM), foi analisado o tempo de processamento desempenhado por cada implementação e a qualidade do resultado obtido. Para obter uma representação precisa do ambiente foi conduzido o experimento que utiliza técnica iterativa do ponto mais próximo (ICP), para realização foi utilizado o sensor de movimento Kinect e os resultados apresentados não foram satisfatórios devido ao volume de dados adquiridos e a ausência de um sistema de estimativa da localização.

Palavra chave: mapeamento robótico 3-D, *tracking* de movimentos, extração de plano de fundo, alinhamento de nuvens de pontos

Abstract

BAPTISTA JÚNIOR, A. (2013). **3-D mapping for robots**. Dissertação (mestrado). Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

In robotics, map the environment is an important task, because it provides information for planning and executing movements of the robot. For this reason, studies presented here are aimed to build 3-D maps and techniques that aid in the task of mapping. When we build 3-D maps, we enable other researchers and robotics companies to develop analyzes and path planning in all six degrees of freedom rigid body that serves to model a mobile robot, manipulator or mobile robot manipulator. With a representation of the environment in 3-D, we increase the accuracy of the robot positioning in relation to the environment and also the positioning of objects that are inserted into the field of action of the robot. To solve the problem of mapping we presented theoretical techniques and their applications in each case studied. In the experiments in this work we adopted the creation of maps with grids of occupation. However, building grids of occupation assumes knowledge of the position of the robot on the environment. In this work we conducted three experiments and their aims are the reduction of data from failures and redundant information using probabilistic techniques, motion detection by background extraction technique and 3-D mapping technique using the closest point. In the experiment whose goal is to reduce the data has been further reduced to 4.43% the number of points required to generate the representation of the environment with the use of our algorithm. The algorithm of 3-D mapping done with probabilistic models available and well established in the literature is based on the probability of independent events and the proposed work involving the posterior probability. The motion detection experiment was performed with the use of openCV and CUDA technique using the Gaussian mixture model (GMM), and we analyzed the processing time and the quality of each implementation result. For an accurate representation of the environment was conducted the experiment using the technique of iterative closest point (ICP) was used to perform the motion sensor Kinect and the results were not satisfactory due to the volume of data acquired and the absence of a system location estimate.

Keywords: 3-D robotic mapping, motion tracking, background extraction, alignment of point clouds

Lista de Figuras

Figura 1.Robôs em disputa por passagem em uma porta com espaço restrito.....	33
Figura 2.Rota de colisão detectada (destacada em vermelho) no trajeto dos robôs.....	33
Figura 3.Planejamento de rota recalculado observando restrições de espaço.....	34
Figura 4: Navegação sendo realizada em condições de espaço restrito.....	34
Figura 5.Exemplo sala com obstáculos.....	36
Figura 6.Exemplo de representação de uma sala em mapa grid.....	37
Figura 7.Erro de odometria, Fonte: Cortesia de Sebastian Thrun.....	40
Figura 8.Um mapa construído pelo robô em um ambiente cíclico, Fonte: Cortesia de Sebastian Thrun.....	41
Figura 9.Localização e rastreamento de pessoas simultânea, Fonte: cortesia de Montemerlo.	43
Figura 10.Robô que transita entre andares.....	44
Figura 11.Robô com falha na localização. a) situações parecidas na leitura; b) mapa plano 2-D, contendo localizações parecidas.....	45
Figura 12: Ângulo de inclinação do robô em relação ao eixo x.....	46
Figura 13.Robô transitando por corredor entre prateleiras.....	46
Figura 14.Robô realizando manobra no corredor entre prateleiras.....	47
Figura 15.Robô em rota de colisão com a carga alocada incorretamente.....	47
Figura 16.Robô em colisão com carga alocada incorretamente, no momento de recarregar uma carga alocada em um nível superior.....	48
Figura 17.A vantagem da abordagem de mapas MLS em comparação com mapas elevados padrão. Em contraste com mapa MLS (direita) a mapa elevado (à esquerda) não tem a capacidade para modelar estruturas verticais, porque ele faz a médias de todos os valores de altura medidos. Uma vez que a distância do ponto de extremidade de um feixe de laser para o próximo ponto no mapa elevado pode ter desvios substanciais da verdadeira distância, a localização torna-se difícil, Fonte:Cortesia de Rainer Kümmerle.....	51
Figura 18.variância de medições de uma altura dependendo da distância do feixe, Fonte: cortesia de Patrick Pfaff.....	55
Figura 19.Exemplo de diferentes células em um mapa MLS, Fonte: cortesia de Triebel.....	56

Figura 20.Estrutura suportada pelo Player, (GERKEY et. al., 2001).....	64
Figura 21.Indicação do trajeto percorrido pelo robô. Trajetória de A à B e B à A.....	70
Figura 22.Dimensão da célula e dimensão do mapa.....	76
Figura 23.Feixe do laser incidindo na mesma célula, a crença é aumentada.....	77
Figura 24.Configuração do sensor laser utilizado no experimento.....	79
Figura 25.Sistema de coordenadas utilizado no experimento.....	79
Figura 26.Estrutura cliente/servidor montada com o Player.....	80
Figura 27.Fluxo da informação entre os módulos individuais, baseado em (PFAFF, 2007)....	83
Figura 28.Possíveis falhas do sensor indicadas em vermelho, numeros de pontos elevados....	83
Figura 29.Imagem do laboratório gerada utilizando tamanho de célula igual a 0,1 m. São mantidos os ventiladores no teto e ângulos de estruturas.....	84
Figura 30.Imagem do laboratório gerada utilizando tamanho de célula igual a 0,1 m. Mantém detalhes importantes do ambiente.....	85
Figura 31.Imagem do laboratório gerada utilizando tamanho de célula igual a 0,5 m.....	85
Figura 32. Imagem do laboratório gerada utilizando tamanho de célula igual a 0,5 m.....	86
Figura 33.Imagem do laboratório gerada utilizando tamanho de célula igual a 1m.....	87
Figura 34.Imagem do laboratório gerada utilizando tamanho de célula igual a 1m.....	87
Figura 35.Representação do ambiente de realização do experimento.....	88
Figura 36.Ampliação do ambiente 2-D com mesas destacadas em branco.....	89
Figura 37: Imagem 1 adquirida pelo sensor de movimento Kinect.....	96
Figura 38: Imagem 1 processada pelo algoritmo da biblioteca openCV.....	97
Figura 39: Imagem 1 processada pelo algoritmo que utiliza CUDA.....	98
Figura 40: Imagem 2 adquirida pelo sensor de movimento Kinect.....	98
Figura 41: Imagem 2 processada pelo algoritmo da biblioteca openCV.....	99
Figura 42: Imagem 2 processada pelo algoritmo que utiliza CUDA.....	99
Figura 43: Imagem 100 adquirida pelo sensor de movimento Kinect.....	100
Figura 44: Imagem 100 processada pelo algoritmo da biblioteca openCV.....	102
Figura 45: Imagem 100 processada pelo algoritmo que utiliza CUDA.....	102
Figura 46: Imagem 120 adquirida pelo sensor de movimento Kinect.....	103
Figura 47: Imagem 120 processada pelo algoritmo da biblioteca openCV.....	103
Figura 48: Imagem 120 processada pelo algoritmo que utiliza CUDA.....	104

Figura 49: Leituras realizadas de pontos diferentes.....	108
Figura 50: Nuvem de pontos adquiridas pelo sensores posicionados de acordo com a figura 49.....	108
Figura 51: Nuvem de pontos após ser processado pelo algoritmo de ICP.....	109
Figura 52: Threshold ajustado para 1 m. Reconstrução insatisfatória.....	112
Figura 53: Threshold ajustado para 10 cm. Possibilidade de identificar a lata e falhas devido a falta de odometria.....	113
Figura 54: Reconstrução do ambiente 3-D, ainda apresenta falhas devido a ausência de um sistema de localização.....	113
Figura 55: Nuvem de pontos do ambiente captado pelo sensor Kinect. Elevado número de pontos desnecessários.....	114

Lista de abreviaturas

CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
FPS	Frame Per Second
GMM	Gaussian Model Mixture
GPU	Graphics processing unit
HAL	Hardware Abstract Layer
ICP	Interactive Closest Point
MLS	Multi Level Surface
MRPT	Mobile Robots Programming Toolkit
NARF	Normal Aligned Radial Feature
OpenCV	Open Computer Vision
OpenNI	Open Natural Interaction
PCL	Point Cloud Library
POSIX	Portable Operating System Interface
ROS	Robotic Operating System
SIFT	Scale Invariant Feature Transform
SIMD	Single Operations Multiple Data
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
VRML	Virtual Reality Modeling Language

Sumário

1. Introdução e justificativa.....	21
1.1 Hipótese-Tema, Proposta e Objetivos.....	21
2. Revisão bibliográfica.....	31
2.1 Mapeamento de ambiente com mapas de ocupação em grid.....	35
2.2 Simultaneous Localization and Mapping (SLAM) e técnicas para correção de erros.	37
2.3 Mapeamento 3-D utilizando Multi Level Surface (MLS).....	48
2.3.1 Construção de mapas MLS.....	52
2.3.2 Criação do mapa MLS.....	52
2.3.3 Representação do mapa.....	54
2.3.4 Atualização do mapa.....	56
3. Métodos teóricos do projeto.....	59
4. Experimentos de mapeamento: robô móvel e manipuladores robóticos.....	61
4.1 Softwares utilizados	61
4.1.1 Sistema Operacional.....	61
4.1.2 Player.....	63
4.1.3 VRMLVIEW.....	64
4.1.4 Mobile Robots Programming Toolkit.....	64
4.1.5 OpenCV.....	65
4.1.6 OpenNI.....	65
4.1.7 Tecnologia CUDA.....	66
4.2 Hardwares utilizados.....	67
4.2.1 ActiveMedia Pioneer, o robô.....	67
4.2.2 Laser SICK.....	67
4.2.3 Computadores pessoais.....	68
4.2.4 Kinect.....	68
4.3 Experimento de mapeamento para robótica móvel.....	69
4.3.1 Configuração do robô com o laser SICK.....	78
4.3.2 O funcionamento hardware e software.....	80

4.3.3 Resultados e comparativos do experimento.....	83
4.3.3.1 Células com tamanho de 0,1 m.....	84
4.3.3.2 Células com tamanho de 0,5 m.....	85
4.3.3.3 Células com tamanho de 1 m.....	86
4.3.3.4 Extração da representação 2-D do ambiente.....	88
4.4 Experimento de mapemanto para manipuladores robóticos.....	89
4.4.1 O método probabilístico do modelo de mistura gaussiana (GMM).....	90
4.4.1.1 Realização do experimento.....	94
4.4.1.2 Resultados e comparativos do experimento.....	95
4.4.1.3 Aplicação e discussão sobre o método GMM.....	105
4.4.2 O método Iterativo de ponto mais próximo (ICP).....	106
4.4.2.1 Realização do experimento.....	110
4.4.2.2 Resultados e comparativos do experimento.....	111
4.4.2.3 Discussão sobre o método ICP.....	114
5. Conclusões e trabalhos futuros.....	116
6. Referências Bibliográficas.....	121

2. Introdução e justificativa

Este capítulo apresenta a proposta, os objetivos e motivação para desenvolvimento de pesquisas de mapeamento 3-D na robótica móvel que podem ser extendidas para robôs manipuladores e robôs móveis manipuladores.

2.1 Hipótese-Tema, Proposta e Objetivos

Nas últimas duas décadas o Mapeamento Robótico tem sido o foco das pesquisas nas áreas de robótica e inteligência artificial e inúmeras pesquisas têm gerado os mais diversos produtos na área. Com estudos de mapeamento e localização foi possível desenvolver um robô que opera como um guia em museus, os trabalhos de Thrun et. al. (1999) e Thrun et. al. (2000a.) descrevem algumas técnicas utilizadas para viabilizar esta construção.

De acordo com Elfes (1987) um robô deve ser programado e ser possível reprogramá-lo. Em algumas aplicações os robôs móveis utilizam uma representação prévia do ambiente que possibilita sua operação. Em algumas situações o robô necessita percorrer os mesmos caminhos por longos períodos e a representação fornecida pode ser ineficiente devido a alterações do ambiente. Para Mcdermont (1992), ambientes que ocupam grandes áreas, a chance de informações semelhantes serem confrontadas aumenta.

Atualmente as aplicações que utilizam robôs móveis aumentaram, grande parte delas utilizam representações em 2-D. Os robôs móveis foram submetidos a novas situações, no entanto, as soluções encontradas para construção de mapas não armazenam toda informação obtida o que gera falhas na operação devido a incerteza de sua localização em relação ao ambiente. Em Thrun (2002) é dada atenção a toda informação obtida com objetivo de construir uma representação contendo mais informações e habilitar maior grau de autonomia para os robôs móveis. Uma representação contendo mais informações permite que os robôs móveis possam mapear ambientes desconhecidos, otimizar o planejamento de rotas e, de

forma coerente, promover a localização em lugares como passagens subterrâneas e rampas.

O trabalho de Nüchter et. al. (2005) apresenta o robô Kurt3D, este robô faz parte da categoria de robôs de regates e já foram utilizados no atentado de 11 de setembro de 2001 nos Estados Unidos. O robô teve como objetivo mapear os escombros na operação de resgate à vítimas.

Os avanços de NÜCHTER (2009) possibilitam a representação do ambiente com 6 graus de liberdade e trabalhos recentes apontam a aplicação dos métodos estudados na robótica móvel em diversas áreas da ciência robótica. Em aplicações que utilizam manipuladores robóticos, a citar: deslocamento de caixas e peças, processos de soldagem e pintura, corte, aperto de parafusos e outras que podem ser desempenhadas por esta categoria de manipuladores, o ambiente é controlado e limitado a área de atuação de um robô. Se inserido em um ambiente dinâmico, este tipo de robô não evita colisões que podem acarretar em acidentes que trazem prejuízos e riscos a vida humana.

Os robôs manipuladores, introduzidos na indústria na década de 60, foram, dentre outros sistemas, os precursores da robótica, e atualmente compõem a base disciplinar desta ciência. Por outro lado, os robôs móveis sofreram uma grande evolução nas últimas duas décadas e são temas de estudo de diversos centros de excelência.

Um robô manipulador inserido em ambientes dinâmicos necessita de informações sobre mudanças que acontecem frequentemente para planejar ou re-planejar suas rotas e cumprir com sua navegação evitando colisões que podem trazer qualquer tipo de risco.

Robôs manipuladores são eficientes em desempenhar suas tarefas, são capazes de realizar a tarefa de soldagem, apertar válvulas e recortar chapas de ferro ou plástico com alto grau de precisão. Estes robôs são fixados no ambiente e sua área de atuação é limitada. Em nosso cotidiano, nos deparamos com situações que robôs manipuladores podem colaborar com tarefas que trazem risco ao ser humano. Em caso de acidentes químicos ou nucleares, faz-se necessário isolar o ambiente. Os robôs móveis tem colaborado ao realizar a tarefa de identificação da área acidentada, no entanto, há necessidade de seres humanos realizarem o isolamento. As pessoas que trabalham nestas operações ficam expostas a altas radiações que trazem risco à vida. Os robôs manipuladores podem colaborar ao desempenhar tarefas que exijam alto grau de precisão, para isso faz-se necessário o deslocamento do manipulador

robótico.

A revisão do estado da arte encontra-se uma nova categoria de robôs denominada robôs manipuladores móveis. Os avanços tecnológicos nas ciências de processamento de imagem e inteligência artificial alcançados nas últimas duas décadas proporcionam o estudo de novos sistemas robóticos, com altíssimo nível de autonomia, interação e eficiência.

Robôs manipuladores móveis são constituídos por braços robóticos montados sobre plataformas móveis e, portanto, constituem um sistema mecatrônico formado por dois dispositivos amplamente estudados na robótica: robôs manipuladores e robôs móveis.

Tanto os robôs manipuladores quanto os robô móveis apresentam teorias, clássicas e modernas respectivamente, amplamente aceitas na academia, contudo, um manipulador móvel é constituído pelo acoplamento dos dois sistemas descritos acima e apresenta característica híbrida que é representada por teorias distintas das convencionais.

O acoplamento mecânico entre plataforma móvel e manipulador modifica o comportamento dinâmico de ambos os sistemas no caso de navegação combinada com manipulação. O manipulador ao se movimentar aplica, sobre a plataforma, esforços que devem ser compensados por controladores de tração. Da mesma maneira a movimentação da plataforma produz efeitos de Coreolis e erros de posicionamento que devem ser devidamente considerados para posicionamento correto do efetuador do manipulador.

Ao interagir com objetos o robô modifica seu ambiente, desta maneira o mesmo deve ser capaz de reconhecer as modificações realizadas, ou seja, o robô ao modificar o ambiente deve reconhecer as modificações para evitar colisões. No caso de cooperação entre robôs, ou em ambientes onde há mais de um robô móvel trabalhando, as modificações podem ser realizadas por qualquer robô, portanto, todos os robôs devem ser capazes de identificar modificações no ambiente mesmo que não tenha sido por ele modificado.

Quanto à aplicação de robô manipuladores móveis pode-se citar: manipulação de produtos em estoques, peças usinadas, produtos em mercados; carregamento e descarregamento de produtos em caminhões, oficinas, transporte de material em fábricas; prover mobilidade aos manipuladores industriais fixos; em serviços domésticos onde é necessário fixar e mover objetos, entre outros.

Em algumas células robóticas vários robôs são necessários para ampliar o espaço de

trabalho. Os robôs podem ser substituídos por um único robô móvel capaz de se movimentar por toda a área de trabalho e manipular objetos, esta substituição só é válida quando temos a diminuição do tempo para realização de uma tarefa e não haja paradas na operação de outras células que realizam tarefas cooperativas. Exploração espacial, verificação e coleta de rochas e fragmentos. Robôs subaquáticos para manutenção de estações profundas de extração de petróleo. O sistema móvel também pode ser completamente integrado aos sistemas fixos existentes utilizando cooperação entre máquinas.

Os mapas podem ter sua integridade comprometida quando robôs são inseridos em ambientes altamente dinâmicos. O robô precisa monitorar e distinguir o que são objetos estáticos e objetos dinâmicos, desta forma conseguimos monitorar a trajetória e estimar posições futuras do objeto em movimento. Para realizar esta tarefa, faz-se necessário a utilização de técnicas que detectam movimentos.

Os robôs necessitam segmentar regiões e detectar ocorrências de movimentos. O trabalho de Koller et. al. (1994) aplica uma técnica baseada na utilização de filtro de Kalman para realizar o monitoramento de tráfego de rodovias. Para que robôs sejam capazes de interagir com o ambiente é necessário que as informações sejam atualizadas em tempo real, no trabalho de Ridder et. al. (1995) foi verificado que a utilização do filtro de Kalman é robusta mas seu desempenho é comprometido ao tratar objetos em movimento e sombras. A proposta STAUFFER e GRIMSOM (1999), apresenta uma técnica que pode auxiliar na tarefa de mapeamento. Com a utilização desta técnica é possível adquirir uma máscara e realizar a distinção de objetos móveis e estruturas fixas.

Os avanços tecnológicos que ocorreram nos últimos vinte anos tornaram possível a aquisição de imagens com maior qualidade. Ao aumentar a qualidade da imagem, é aumentado o número de informação a ser processada e os métodos apresentados tornam-se ineficientes para desempenhar a tarefa de segmentação e detecção de movimento.

A proposta de NVIDIA (2012) traz uma tecnologia que combina hardware e software para aumentar o desempenho na tarefa de processamento gráfico. Com a utilização desta tecnologia é possível paralelizar tarefas massivas e aumentar o desempenho do processamento.

Este trabalho partiu da hipótese que o estudo e desenvolvimento de um módulo de

mapeamento 3-D poderá ser utilizado em robôs móveis para ambientes indoor e capaz de se estender a robôs manipuladores e robôs manipuladores móveis. Por este motivo, esta dissertação tem por proposta produzir o referido módulo robô de mapeamento 3-D cujo sistema lhe permita ser integrado a um sistema de planejamento, navegação e localização. As tarefas de localização, planejamento de rota e navegação dependem das informações fornecidas pelo módulo de mapeamento para a realização de cada uma das tarefas a eles pertencentes.

Portanto, o objetivo precípua desta dissertação de mestrado é construir um módulo robótico capaz de produzir um mapa que lhe permita identificar pessoas, caixas, estruturas, mesas, cadeiras, portas, janelas, fios, cabos, etc. O referido módulo vai auxiliar as tarefas de localização, planejamento de rota e navegação; vale dizer – a criação deste módulo objetivou a extração de informações do ambiente no qual o robô esteja contido, para ser possível a ele planejar uma rota, navegar por ela e mapear todo o trajeto percorrido. A proposta objetivou, ainda, gerar um mapa 3-D que seja capaz de otimizar rotas de locomoção de robôs móveis em ambientes com severas restrições de espaço, assim como otimizar a navegação em ambientes dinâmicos e ocupados por pessoas e por outros robôs realizando tarefas semelhantes.

Com o desenvolvimento deste trabalho, colabora-se com métodos que podem ser empregados na tarefa de mapeamento e que servem de suporte a tarefas de localização, planejamento e navegação.

Esta monografia de mestrado, composta por 6 capítulos, está assim estruturada:

2. REVISÃO BIBLIOGRÁFICA: Este capítulo apresenta o referencial teórico adquirido com a leitura do estado da arte sobre mapeamento que utilizam robôs móveis, revisão essa que proporcionou base para esta pesquisa ser levada a efeito. Para melhor concatenar os pensamentos e o estudo, o material bibliográfico foi estudado e os textos foram agrupados nos seguintes sub-itens:
 - 2.1 Mapeamento de ambiente com mapas de ocupação em *grid* – em português, grade ou malha: Este capítulo apresenta a teoria para representação de mapas métricos que

utiliza ocupação em *grid*.

2.2 *Simultaneous Localization and Mapping* (SLAM) e técnicas para correção de erros: Este capítulo apresenta as dificuldades em desempenhar a tarefa de mapeamento, métodos utilizados em SLAM,

2.3 Mapeamento 3-D com a técnica *Multi Level Surface* (MLS), que discute a Técnica SLAM e a Construção de Mapas. Está subdividido nos seguintes itens:

2.3.1 Construção de mapas MLS: Este sub-item discute a metodologia MLS – *Multi Level Surface* – metodologia essa com largo uso em estruturas verticais e aborda, também, os conjuntos de dados necessários à essa construção, dados que são estatisticamente tratados por meio de média, desvio padrão e análise de variância aplicada à criação de mapas. Mostra, também, como esses conjuntos operam para representá-los.

2.3.2 Criação do mapa MLS: Este sub-item apresenta as informações necessárias para criar um mapa MLS.

2.3.3 Representação do mapa: Este sub-item apresenta a forma de representar mapas MLS.

2.3.4 Atualização do mapa: Este sub-item apresenta o método aplicado para novas informações adquiridas pelos sensores.

3. MÉTODOS TEÓRICOS DO PROJETO: Neste capítulo é discutida a metodologia; mais precisamente, ele discute os métodos existentes para criação de mapas de ocupação em *grid* e reporta, ainda, os métodos existentes para auxiliar a criação e atualização de mapas.

4. EXPERIMENTOS DE MAPEAMENTO: ROBÔ MÓVEL E MANIPULADORES ROBÓTICOS: Este capítulo apresenta os componentes de *software* e *hardware* necessários ao desenvolvimento do produto deste projeto – o robô móvel – além de reportar o funcionamento do *software* e do *hardware*. Trata dos experimentos que foram realizados e da aquisição de dados – que foram levados a efeito com uso do *Player Server*. Descreve e justifica a escolha do método utilizado para sua criação e

atualização; além de descrever o funcionamento do Robô móvel e de mencionar sua atualização. Além disso, trata da construção do algoritmo de mapeamento gerado para que o robô cumpra essa sua tarefa. Ele está subdividido em itens descritivos, a saber:

4.1. *Softwares* utilizados: A opção pelos sistemas Operacionais Linux e Player foi decidida em função de seu largo uso no mundo todo, além de estar disponível gratuitamente. Para esta pesquisa foi usada a plataforma cliente/servidor fornecida pelo *software Player*, com robôs de menores proporções, exatamente pelo fato de o objetivo ser construir robôs *indoor*.

4.1.1 Sistema Operacional: Como sistema operacional escolhido foi o Linux, não apenas por ser *software* gratuito e muito utilizado em pesquisas no mundo todo, mas pelo fato de ser possuidor de bons serviços de redes e recursos de compartilhamento de tempo. O Linux é um sistema operacional livre compatível com o padrão POSIX (*Portable Operating System Interface*) definido pela IEEE 1003, utilizado em diversos projetos de robótica.

4.1.2 *Software Player*: Este item descreve e justifica a opção pelos softwares da plataforma de simulação de robôs. Player é um servidor de rede para controlar robôs. Na terminologia de sistemas operacionais o *Player* é uma camada de abstração de *hardware* (HAL) para dispositivos robóticos micro-controlador. Uma camada de abstração de *hardware* (HAL) para dispositivos robóticos que oculta os detalhes do *hardware* do computador, dispositivo que é tratado por um driver . Informa, ainda, o papel do *Player* em fornecer mecanismo que permite a troca de dados entre o controlador e programas de controle, dos quais o de uso mais comum é baseado em *TCP-sockets*. Para este trabalho foi escolhido o uso da linguagem C.

4.1.3 *Software VRMLVIEW*: Este sub-item trata da opção pelo *software VRML* (*Virtual Reality Modeling Language*) o qual possui alta velocidade para realizar tarefa de visualização.

4.1.4 *Mobile Robots Programming Toolkit*: Este sub-item trata da opção pela biblioteca MRPT a qual possui acesso fácil a dispositivos utilizados no projeto

e um amplo conjunto de métodos utilizados na robótica.

4.1.5 OpenCV: Este sub-item trata da opção pela biblioteca OpenCV a qual possui um amplo conjunto de métodos utilizados para processamento de imagens.

4.1.6 OpenNI: Este sub-item trata da opção pelo framework OpenNI a qual possui um amplo conjunto de métodos utilizados para prover interação com usuários e reconhecimento de padrões..

4.1.7 Tecnologia CUDA: Este sub-item trata da opção a tecnologia CUDA que envolve questões de *hardware* e *software* e traz aumentos significativos ao desempenhar tarefas de processamento de imagens.

4.2 *Hardware*s: Este item mostra as opções com relação aos *hardware*s e suas interfaces com o *software*. Ele está subdividido em:

4.2.1. *Active Media Pioneer*, o robô: Explica a opção pelo *Active Media Pioneer*, que tem sido usado por pesquisadores em experimentos de localização e mapeamento. O *Pioneer* possui um computador pessoal em sua estrutura com interfaces de comunicações, interfaces normalmente utilizadas para conexão dos sensores que facilmente são acopláveis a ele.

4.2.2 *Laser SICK*: Escolhido para aquisição de dados de distância de objetos, obstáculos, pessoas e paredes.

4.2.3 Computadores pessoais: Apresenta a configuração dos computadores pessoais.

4.2.4 Kinect: Apresenta o sensor de movimento Kinect, seu potencial para pesquisas que envolvam robótica.

4.3 Experimento de mapeamento para robótica móvel: Neste sub-item está reportado o experimento realizado com robô móvel.

4.3.1 Configuração do Robô com *Laser Sick*: Neste sub-item está reportado como o sensor *SICK* foi montado na estrutura do robô

4.3.2 Funcionamento do *hardware* e *software*: Aqui está referido como foi montada a plataforma cliente/servidor, além de ser mostrada a configuração do servidor *Player*.

4.3.3 Resultados e comparativos do experimento: Mostra como o experimento foi realizado, e relata a configuração do simulador para ser possível repetir o experimento *off line*, - ou seja, apresenta o embasamento teórico do experimento. Também mostra a representação do ambiente 3-D do algoritmo, pseudo algoritmo que explica o funcionamento, bem como a configuração do arquivo base para visualização com VRMLView. Este sub-item, apresenta os resultados da extração realizada pelo algoritmo; os resultados com os 3 tamanhos de células adotados no projeto e a extração 2-D do ambiente. Este sub-item esta dividido em 4 sub-itens.

4.3.3.1 Células com tamanho de 0,1 m: Este sub-item apresenta figuras do ambiente extraído pelas leituras do sensor *laser* no ambiente percorrido pelo robô. Cada célula representa 0,1 m do espaço real.

4.3.3.2 Células com tamanho de 0,5 m: Este sub-item apresenta figuras do ambiente extraído pelas leituras do sensor *laser* no ambiente percorrido pelo robô. Cada célula representa 0,5 m do espaço real.

4.3.3.3 Células com tamanho de 1 m: Este sub-item apresenta figuras do ambiente extraído pelas leituras do sensor *laser* no ambiente percorrido pelo robô. Cada célula representa 1 m do espaço real.

4.3.3.4 Extração da representação 2-D do ambiente: Este sub-item apresenta a representação do ambiente percorrido pelo robô em 2-D.

4.4 Experimento de mapeamento para manipuladores robóticos: Apresenta os experimentos para robôs manipuladores. Apresenta a técnica de GMM implementada com a utilização da tecnologia CUDA e a encontrada na biblioteca OpenCV. Com a mesma fonte é descrito o experimento que utiliza o método ICP utilizado na tarefa de mapeamento. Este sub-item esta dividido em 2 sub-itens:

4.4.1 O método probabilístico do modelo de mistura gaussiana (GMM): Descreve o método GMM o experimento realizado, é sugerido aplicação deste método em tarefas que envolvam mapeamento ou reconhecimento de regiões de interesse.

4.4.1.1 Realização do experimento: Este sub-item descreve o

experimento que utiliza o método GMM para extração de fundo.

4.4.1.2 Resultados e comparativos do experimento: Este sub-item apresenta os resultados obtidos pelos métodos GMM encontrado na OpenCV e que utiliza tecnologia CUDA.

4.4.1.3 Aplicação e discussão sobre o método GMM: Este sub-item faz uma discussão em cima dos resultados obtidos e apresenta uma aplicação para o método.

4.4.2 O método Iterativo de ponto mais próximo (ICP):: Descreve o método ICP e o experimento realizado. São apresentados os resultados obtidos com o método ICP, é apontado as limitações do método com grandes volumes de dados e vantagens quando utilizando como sistema de apoio a localização.

4.4.2.1 Realização do experimento: Este sub-item descreve o experimento que utiliza o método ICP para realizar a tarefa de mapeamento.

4.4.2.2 Resultados e comparativos do experimento: Este sub-item apresenta os resultados obtidos pelos métodos ICP. Apresenta o volume de pontos que foram processados e o ajustes de *threshold*.

4.4.2.3 Discussão sobre o método ICP: Este sub-item faz uma discussão sobre a utilização do método ICP para aplicações de mapeamento. É apontado o potencial do método como auxílio a localização.

5. CONCLUSÕES E TRABALHOS FUTUROS: Neste capítulo consta a discussão sobre o estado da arte de mapeamento; aponta tendências para sua utilização e mostra os experimentos realizados que permitem verificar o que foi possível atingir com os experimentos realizados produzido para este trabalho.

6. REFERÊNCIAS BIBLIOGRÁFICAS: Este capítulo encerra a monografia com a listagem das obras consultadas no estudo

3. Revisão bibliográfica

Neste capítulo é feita a revisão bibliográfica, são apresentadas algumas aplicações que necessitam da tarefa de mapeamento e as dificuldades encontradas em alguns processos, é indicado o método utilizado para alcançar a solução em diversas aplicações do cotidiano. São apresentados três tipos de representação de mapas, a citar: mapas topológicos, mapas métricos e mapas híbridos que utilizam ambas as soluções. No sub-item 2.1 é dado foco na representação de mapas de ocupação em *grid*. Inseridos em ambientes dinâmicos, as representações dos mapas, podem variar bruscamente em questão de minutos ou até mesmo segundos. No sub-item 2.2 é abordada a técnica de que propõe adequação a este tipo de ambiente, a técnica de localização e mapeamento simultânea - SLAM (*Simultaneous localization and mapping*), uma técnica robusta e muito utilizada, é utilizada em aplicações do nosso cotidiano, são abordados métodos probabilísticos envolvidos na solução do problema. Nas últimas duas décadas, as pesquisas estavam direcionadas a aplicações que utilizam representações de mapas 2-D. No entanto, com os avanços das pesquisas novos problemas foram constatados e verificado a necessidade de um novo tipo de representação do ambiente. Esta nova abordagem deve ser capaz de representar um corpo rígido inserido no ambiente, e indicar sua posição relativa nas coordenadas x,y,z e ângulos para descrever sua orientação em um espaço euclidiano tridimensional. No sub-item 2.3 é apresentado um método utilizado para este tipo de abordagem, este método é conhecido como *Multi Level Surface* (MLS). Esta abordagem é capaz de representar o ambiente armazenando suas coordenadas e ângulos. Em seguida, apresentamos a base para construção de mapas MLS e um roteiro para criação, representação e atualização de mapas MLS nos sub-itens 2.3.1, 2.3.2 e 2.3.3 e 2.3.4.

A proposta do desenvolvimento do módulo atende a robôs móveis *indoor* que desempenham diferentes tarefas como transporte e armazenagem de materiais; guias de museus, robôs que exploram ambientes de minas acidentadas, robôs militares para reconhecimento de área; vale dizer ela é válida para todos os robôs que forem desempenhar tarefa de mapeamento e reconhecimento de ambientes. Um robô móvel – que desempenha a

tarefa de um guia de museu – foi apresentado por Thrun et. al. (1999) e Thrun et. al. (2000a.) Esse robô, apresentado na proposta dos pesquisadores, tem a capacidade de interagir com as pessoas que visitam o museu, além de haver a possibilidade de acesso remoto ao robô via internet. No caso de tarefas em locais de atentados terroristas, o robô desempenha naqueles locais de difícil acesso para seres humanos, obstruídos por escombros e poeira, que dificultam o acesso e a visibilidade, tal como mostra a proposta apresentada por Murphy (2004). Vale lembrar, no entanto, que o uso desses robôs não está limitado a essas atividades; eles estão aptos a desempenhar outras tarefas em minas acidentadas, em áreas de desastres naturais e, também, áreas com atividades radioativas ou com vazamentos de contaminantes que apresentam perigo ao ser humano (NÜCHTER et. al., 2005).

O objetivo precípua deste trabalho é construir um módulo que opere em um ambiente *indoor*, capaz de desempenhar tarefas de mapear pessoas, caixas, estruturas, mesas, cadeiras, portas, janelas, fios e cabos. A proposta de criar o referido módulo tem o objetivo específico gerar um mapa 3-D que seja possível otimizar rotas de locomoção de robôs móveis, bem como otimizar sua navegação em ambientes dinâmicos, habitados por pessoas e, ainda, conduzirem outros robôs a realizarem tarefas semelhantes.

A automatização de depósitos e armazéns, aumenta muito o valor deste módulo pelo fato de nesses ambientes ser restrito o espaço para navegação de um robô; o que torna necessário maior precisão de modelagem do ambiente para não ser preciso abrir mão de movimentos acelerados para evitar o risco de colisões. Em depósitos o tempo é sempre precioso, por isso, são indesejadas reduções de velocidades dos robôs. Para este módulo de robô também são interessantes aplicações de robótica cooperativa e, neste caso, podem haver a necessidade de operações em que múltiplos robôs naveguem muito próximos uns dos outros, como também pode haver limites impostos pelo ambiente.

O mapa criado com este módulo deve conter uma representação precisa do ambiente para que sejam habilitadas manobras em espaços restritos. Um robô trafegando no sentido de A para B, trabalha de forma cooperativa com um robô que segue o trajeto inverso, apresentado na Figura 1. Em determinado momento eles irão disputar o acesso a uma porta que está no trajeto de ambos conforme Figura 2 e, nesta situação, há necessidade de manter ambos em operação, mas para isso é necessário que haja precisão nas manobras de ambos para não

haver colisão. Para tanto as localizações de ambos têm que ser precisas o planejamento de rotas deve ser otimizado, e com possibilidade de replanejamento como apresentado na Figura 3 – é imprescindível evitar possíveis colisões com robôs ou obstáculos dinâmicos que possam surgir no ambiente.

Figura 1. Robôs em disputa por passagem em uma porta com espaço restrito

Figura 2. Rota de colisão detectada (destacada em vermelho) no trajeto dos robôs

As condições citadas envolvem distâncias inferiores a 5 cm para que a navegação seja realizada, para ser possível aumentar a utilização dos recursos disponíveis e diminuir o tempo de execução das tarefas desempenhadas pelos robôs. Na Figura 4 é mostrado dois robôs compartilhando um mesmo acesso e evitando colisões e pausa na operação.

Figura 3. Planejamento de rota recalculado observando restrições de espaço.

Figura 4: Navegação sendo realizada em condições de espaço restrito.

Na literatura são encontradas diferentes abordagens para construção de mapas: os mapas métricos utilizados por Elfes (1987); os mapas topológicos utilizados na abordagem de Kuipers e Byun (1990) e os mapas híbridos (mapas métricos e mapas topológicos) utilizados por Thrun e Bücken (1996), Tomatis et. al. (2003).

Mapas topológicos têm como fatores positivos: permitirem realizar planejamento e terem baixa complexidade, em função da complexidade do ambiente. Estes mapas não requerem precisão para determinar a localização do robô; também são capazes de criar uma representação conveniente para planejadores simbólicos; e podem ser utilizados para problemas que envolvam uma interface de linguagem natural. Por outro lado, são de difícil construção e também é difícil construir e mantê-los em ambientes grandes. O reconhecimento de lugares utiliza imagens, por meio de câmeras que podem apresentar ambiguidade, elas são sensíveis aos pontos de visão obtidos pelos sensores e construir caminhos sub-ótimo.

Em contraponto, no caso específico mapas de ocupação em *grid* apresentado por Elfes (1987), os mapas métricos são fáceis de construir, e bons para representar e manter sua integridade. No reconhecimento de lugares distinguem formas (baseado na geometria), não apresentam ambiguidade, seus pontos de visão são independentes e são simples os cálculos de caminho mais curto. Apresentam planejamento ineficiente devido ao fato de consumirem muito espaço por sua precisão não depender da complexidade do ambiente; de ser necessário

conhecer a posição precisa do robô; e por possuir uma interface com poucos recursos para soluções de problemas simbólicos.

Mapas híbridos têm a proposta de utilizar os pontos fortes de ambas as propostas, mas sua complexidade aumenta na proporção que o ambiente também aumenta.

A proposta deste trabalho explora os mapas de ocupação em *grid* pelo fato de o problema envolver precisão na navegação em situações que envolvem um espaço restrito para o robô se movimentar.

3.1 Mapeamento de ambiente com mapas de ocupação em grid

Segundo (ELFES, 1987),

[...] Para aumentar a faixa de aplicações de dispositivos robóticos, tanto nas aplicações de pesquisa como na indústria, é necessário desenvolver sistemas com alto nível de autonomia e capaz de operar em ambientes não estruturados e desconhecidos com pouca informação *a priori*. Para conquistar este grau de independência, um sistema robótico, através da aquisição de dados e manipulação de modelos, deve ser capaz de entender o mundo que o cerca, pela aquisição e manipulação de ricos modelos de ambientes de operação. Para isso, são necessários: uma variedade de sensores que são capazes de interagir com o mundo real obtendo informação deste e um conjunto de mecanismos capazes de extrair o significado destas informações. Sistemas com pouca ou sem capacidade de “sentir” o ambiente são geralmente limitados a sequências fixas de operações em áreas altamente estruturadas e não apresentam nenhum grau substancial de autonomia ou adaptabilidade. (p. 249)

Para preencher algumas lacunas dos problemas apontados, Elfes (1987) criou o modelo de representação de mapas de ocupação em *grid*, utilizando um robô com sensores ultra-sônicos; as áreas desconhecidas, ocupadas e vazias foram explicitamente representadas em um mapa. Esta abordagem viabilizou tarefas como planejar rotas, desviar de obstáculos, apresentar soluções de movimentos e reconhecer *landmark*. A sala observada na figura 5 – exemplo de um mapa de ocupação em *grid* – contém os obstáculos apresentados nas formas

retangulares em cor preta , permitiu ,a referida representação , como mostrado na figura 6, a qual demonstra perfeitamente e, portanto, sem dúvidas a presença ou não de obstáculos. As células do *grid* ocupadas com o valor 0, correspondem a lugares vazios; por sua vez as células ocupadas com os valores 1 representam os obstáculos contidos na sala. Não é possível produzir um mapa de uma sala no mundo real , devido a falhas dos sensores e movimentos do robô.

Figura 5.Exemplo sala com obstáculos

O propósito é produzir robôs mais eficientes e dotados de recursos que os habilitem a trabalhar por tempos maiores de operação. Nessas tarefas, o robô transita pelo mesmo trajeto durante horas de operação podendo desempenhar tarefas de carga de bateria e transporte de cargas. Nestas tarefas, a carga da bateria do robô tem um ponto fixo no mapa. Para cumprir a referida tarefa de carga de bateria o robô deve visitar o local no qual se encontra a base de alimentação e, em seguida, deve voltar para realizar a tarefa de transportar cargas.

As tarefas de visitar determinado local passaram a ter necessidade de novas técnicas para interpretar as novas situações vivenciadas pelo robô. Nesta nova abordagem, a tarefa de localização e a tarefa de mapeamento são realizadas simultaneamente; ao mesmo tempo em que o robô realiza sua localização ele atualiza seu mapa. No próximo subitem deste capítulo será abordada a técnica de localização e mapeamento simultâneo.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 6. Exemplo de representação de uma sala em mapa grid.

3.2 Simultaneous Localization and Mapping (SLAM) e técnicas para correção de erros

Há robôs que jamais necessitam revisitar um lugar; porém para Engelson e Mcdermont (1992), o foco são robôs que têm essa necessidade.

[..] “Se um robô precisa ir para um lugar que tem bananas, ele vai até o local mais próximo que sabe haver bananas, guiado por uma espécie de 'banana tropismo'. Enquanto tropismo baseados em informações locais podem ser úteis, eles podem ser ineficientes para navegação em grande escala.” [...]. (p. 2555)

Foi assumido que, ao caminhar pelo ambiente, o robô constrói uma representação desse ambiente e, por isso, pode encontrar posições exploradas, posições exploradas e modificadas e, ainda, posições não exploradas no ambiente.

A técnica de localização e mapeamento simultânea - SLAM (*Simultaneous localization and mapping*) torna possível a construção de mapas em ambientes desconhecidos e, ao mesmo tempo, mantém seu rastro da posição atual; esta tarefa não é tão simples como parece, devido às incertezas do movimento relativo do robô a partir de seus vários sensores. Com monitoramento confiável de movimento – odometria – mesmo com incertezas seria fácil a construção de mapas até mesmo na presença de incertezas dos sensores.

Um ponto observado por Engelson e Mcdermott (1992) no aprendizado de mapas foi a questão do acúmulo de erro, o que é devido à imprecisões dos sensores, ruídos e às fracas suposições do ambiente no qual o robô esteja inserido. Por esse motivo os autores indicaram: “[...] atenção deve ser dada para o problema da robustez em presença do aprendizado de erro[...]”.(p. 2555)

Thrun (2002) dirigiu um foco probabilístico ao mapeamento robótico: “[...] Eles são probabilísticos todos eles utilizam modelos probabilístico do robô e do ambiente, e todos eles contam com inferência para transformar as leituras dos sensores em mapas.”. (p. 4) Isto porque o que caracteriza o mapeamento robótico são a incerteza e o ruído do sensor.

Na estatística básica de Morettin (2010), um determinado evento A pode estar condicionado a um evento B , nesta situação temos um caso de probabilidade condicional. Para que um evento A aconteça, é necessário que B tenha acontecido e vice-versa. Sejam:

$$A \subset \Omega \text{ e } B \subset \Omega$$

Os eventos A e B , estão contidos no conjunto Ω . Intuitivamente, se A e B são eventos dependentes, temos que:

$$P\left(\frac{A}{B}\right) = \frac{P(A \cap B)}{P(B)}, \text{ se } P(B) \neq 0$$

também,

$$P\left(\frac{B}{A}\right) = \frac{P(B \cap A)}{P(A)}, \text{ se } P(A) \neq 0$$

definimos através do teorema do produto:

$$P(B \cap A) = P(B) \cdot P(A/B)$$

ou

$$P(A \cap B) = P(A) \cdot P(B|A)$$

Dois eventos são eventos independentes quando a realização ou não-realização de um dos eventos não afeta a probabilidade da realização do outro, ou vice-versa. Dados dois eventos A e B :

$$A \subset \Omega \text{ e } B \subset \Omega$$

Os eventos A e B estão contidos no conjunto Ω . Intuitivamente, se A e B são eventos independentes, temos que:

$$P\left(\frac{A}{B}\right) = P(A) \text{ e } P\left(\frac{B}{A}\right) = P(B)$$

Os eventos A e B são eventos independentes se:

$$P(A \cap B) = P(A) \cdot P(B)$$

Se A e B são mutuamente exclusivos, então A e B são dependentes e, dessa forma, a ocorrência de um dos eventos condiciona a não ocorrência do outro.

Como foi observado por (THRUN, 2002),

“[...] Os comandos de movimento (controles) emitidos durante a exploração do ambiente carrega uma importante informação na construção de mapas, uma vez que transmitem informações sobre os locais onde diferentes leituras dos sensores são tomadas, O movimento do robô também está sujeito a erros[...]”. (p. 2)

As medições dos ruídos representam grande desafio no mapeamento robótico. Se os ruídos em diferentes medições são estatisticamente independentes, a solução é fácil: basta o

robô adquirir inúmeras medições para anular os efeitos do ruído. No mapeamento robótico os erros de medição são estatisticamente dependentes. Assim, os erros de controle acumulam-se ao longo do tempo e passam a interferir na forma de interpretação das futuras medições.

Na Figura 7 podem ser observados os erros acumulativos de odometria; também está ilustrado o caminho do robô obtido por odometria em relação ao mapa. Pequenos erros de odometria podem ter grandes efeitos sobre uma estimativa de posições posteriores.

*Figura 7. Erro de odometria,
Fonte: Thrun (2002), cortesia de Sebastian Thrun.*

Os erros de odometria podem ser evitados de forma significativa. Para ser viável a construção de mapas coerentes, um robô deve estabelecer correspondência entre sua posição presente e suas posições passadas — tarefa que não é fácil ao encerrar um ciclo.

Para solução de SLAM foram adotadas diversas técnicas probabilísticas, filtros de Kalman e método de Monte Carlo. Alguns métodos são usados para validar os dados dos sensores, outros são utilizados para correções do posicionamento do robô.

A Figura 8 apresenta um exemplo no qual são aplicadas técnicas probabilísticas para realizar a correção nos dados da odometria e do laser. O robô percorre o ambiente realizando ciclos e, a cada etapa, o mapa é atualizado e corrigido. Uma das características do problema de SLAM são ambientes que fecham um laço, os ciclos; quando o robô atinge uma posição

já explorada, ele recalcula o mapa ajustando o posicionamento das células.

Figura 8. Um mapa construído pelo robô em um ambiente cíclico, Fonte: Thrun (2002), cortesia de Sebastian Thrun

As *landmarks* apresentada por Borestein et. al. (1997) são características distintas que o robô reconhece em um ambiente através de entradas de dados em seus sensores, os quais podem possuir formas como círculos, quadrados, retângulos e linhas. Há aplicações que incluem informações adicionais como códigos de barras.

O posicionamento de uma *landmark* deve ser fixo em uma posição conhecida. Através do posicionamento das *landmarks*, a principal tarefa desempenhada pelo robô é reconhecer a *landmark* e calcular sua própria localização no ambiente.

Outros problemas foram observados no decorrer da evolução dos algoritmos de SLAM; os algoritmos baseados em filtros de Kalman estendido (EKF), possuem bom desempenho em ambientes reduzidos e modelos planares *indoor*.. No entanto para realizar o processamento, a Σ_t requer cálculos na proporção de 4 vezes o número N de *landmark*, para ambientes grandes e, com um número elevado de *landmark*, a abordagem se torna inviável. , Montemerlo (2003) propôs o FastSLAM 1.0 e 2.0, e apresentou suas vantagens para o processamento de mapas com milhões de características. Neste trabalho, não foram utilizados filtros de Kalman ou filtros de partículas, mas em trabalhos futuros esses filtros podem ser

empregados; nesta proposta não foi necessária sua utilização porque o projeto atende a robôs móveis *indoor* e a distância percorrida não acumulou erros significativos para o experimento.

Como o robô tem sido inserido no cotidiano humano, a cada vez mais, ele deve desempenhar tarefas interativas; interação esta que pode ser observada de diversas formas, incluindo: diálogo falado, interação física, colaboração na execução de determinada tarefa e, também, compartilhamento de passagens (porta doméstica, ruas e avenidas urbanas). Para obterem sucesso, eles devem ter a capacidade de observar e modelar o comportamento dos seres humanos com os quais estão interagindo.

A localização de mapas baseados em rastreamento de pessoas é considerada como um problema comparado ao problema da galinha e o ovo. Se conhecermos a posição exata do robô no mapa podemos determinar quais leituras dos sensores são objetos estáticos, e, então, podemos simplesmente tirá-los do mapa. Inversamente, se as leituras dos sensores que correspondem a objetos dinâmicos podem ser retiradas (filtradas), então a posição do robô pode ser determinada com maior precisão. Quando a posição do robô e a posição próxima de todas as pessoas são desconhecidas, a localização e o rastreamento de pessoas se tornam problema de estimativa conjunta.

Como mostrado por Montemerlo (2003), pode acontecer situações parecidas que são difíceis para que um robô distinguir. No exemplo utilizado por Montemerlo (2003) – como mostra a figura 9(a) – uma pessoa está no corredor, em pé, na frente da porta #1, e o robô está em um dos dois lugares especificados no mapa. Se o robô também estiver na porta #1 virado para o corredor, ele vê a pessoa e adquire a varredura laser – como mostrado na figura 9(b). Se o robô estiver na porta #2, ele vê uma lixeira no corredor e adquire a varredura laser mostrada na figura 9(c). Embora as duas varreduras tenham uma visão muito semelhante, elas representam diferentes hipóteses. Se o robô não considerar a primeira hipótese ele assumirá que a pessoa é lixeira e, por consequência irá assumir a localização errada.. Um algoritmo de rastreamento de pessoas que não assumir a segunda hipótese irá considerar a lixeira como pessoa.

A técnica de SLAM tem o propósito do robô navegar por ambientes desconhecidos, construindo e mantendo o mapa da navegação e, também, de traçar a posição do robô (*tracking*). Nessa técnica podem ser implementadas funções para reconhecer pessoas e

objetos dinâmicos contidos no ambiente;; pessoas e objetos dinâmicos são interpretados por diferentes representações do mapa; esta técnica é possível em variações de implementação do algoritmo de SLAM. Estas representações podem adicionar inconsistências na informação, por exemplo, em uma primeira navegação por determinado local o robô pode encontrar uma caixa aguardando ser empilhada; em uma segunda passagem pelo mesmo local esta caixa já pode ter sido removida, fato que habilita passagens. Para realizar esta tarefa Wolf e Sukhatme (2005) apresentaram uma proposta que realiza a construção de 2 mapas, um que contém objetos estáticos como muros e paredes e o segundo mapa que armazena apenas objetos e pessoas que se movimentam pelo ambiente,. Nessa mesma proposta os referidos autores (*op.cit.*) criaram um terceiro mapa contendo as *landmarks* detectadas no ambiente.

Figura 9. Localização e rastreamento de pessoas simultânea, Fonte: Montemerlo (2003), cortesia de Montemerlo.

A vantagem de utilização da técnica de SLAM, além do rastreamento, é a correção aplicada com a visita de um lugar já visitado; esta correção é aplicada a cada ciclo, o que aumenta a precisão e consistência do mapa.

Baseado na técnica de produção de mapas MLS - *Multi Level Surface*, foi criada a representação do mapa deste trabalho sem utilizar técnicas para localização ou SLAM porque o propósito era criar uma representação 3-D do ambiente no qual o robô está inserido; ela é apresentada no próximo capítulo. A representação em pauta deve conter informações sobre

passagens (portas e janelas), pessoas (para rastreabilidade e evitar colisões) e objetos dinâmicos.

Mapas MLS são uma extensão de mapas elevados que criam uma representação em $2\frac{1}{2}$ - D. Nas representações $2\frac{1}{2}$ - D são perdidas muitas informações de forma das estruturas verticais; ocultar reentrâncias e formas côncavas que possibilitam a navegação. Na representação MLS robôs são habilitados a navegar por ambientes que possuem múltiplos andares, por exemplo, o estacionamento de um *shopping*., tal como o ambiente apresentado na figura 10, um ambiente pode conter passagens que ligam um andar ao outro.

Figura 10. Robô que transita entre andares.

Nestas condições, lugares distintos podem conter informações parecidas, fato este que pode gerar falha na localização do robô. A figura 11 a mostra que o robô que transita entre andares e pode obter leituras de andares distinto que contêm semelhanças para sua localização e esse fato pode gerar falhas em seu posicionamento; por exemplo o módulo de localização do robô pode informar que ele está no primeiro ou segundo andar, gerando redundância de posicionamento. Esta redundância fica mais evidente no caso de um mapa planar 2-D, como o apresentado na figura 11 b, nesta situação, o robô pode estar em qualquer um dos andares,

pois a coordenada x, y , coincide em ambos os andares.

Para que a localização seja correta é preciso armazenar informações relativas à inclinação em relação ao plano, e à altura em que o robô está se deslocando para que a localização funcione de forma correta. Se a informação reporta que o robô está inclinado em relação ao plano, é fácil deduzir se ele está subindo ou descendo uma rampa. A informação relativa à sua inclinação é importante para localizá-lo no ambiente. A cada andar pelo qual o robô passa, em sua componente k é adicionada a transição entre andares. Na figura 12 são apresentadas situações nas quais é possível extrair o ângulo β e adicioná-lo ao modelo de representação do mapa.

Figura 11. Robô com falha na localização. a) situações parecidas na leitura; b) mapa plano 2-D, contendo localizações parecidas.

A proposta deste trabalho objetiva atender robôs que transitem em lugares com pouco espaço para manobras. A Figura 13, mostra um robô que transita por um corredor, em cujas laterais estão prateleiras nas quais as cargas são alocadas; seu espaço para manobras é limitado, não há espaço suficiente para que toda sua estrutura física possa realizar uma manobra pelo corredor.

Figura 12: Ângulo de inclinação do robô em relação ao eixo x

Figura 13. Robô transitando por corredor entre prateleiras.

No entanto, existe um espaço vago à sua esquerda, o qual parte de sua estrutura pode ocupar e manobrar de forma a aumentar a eficiência da tarefa de transporte e controle do robô.

Na Figura 14 é mostrada, uma das possíveis manobras e aponta a necessidade de precisão na construção do mapa do ambiente.

Figura 14. Robô realizando manobra no corredor entre prateleiras.

Uma situação que pode apresentar falhas é se uma carga for alocada de forma incorreta; esta falha dificultará a navegação pelo corredor e poderão haver colisões entre a carga alocada de forma incorreta e a carga que está sendo transportada. A Figura 15, destaca esta situação.

Figura 15. Robô em rota de colisão com a carga alocada incorretamente.

Também existe a situação na qual o robô se depara com a caixa alocada de forma incorreta, quando o robô está a realizar a tarefa de coletar uma carga para transportá-la a outra área do depósito. Em sua tarefa, no momento da coleta, de acordo com a posição da caixa, que esta alocada em nível inferior ao da carga de interesse, pode haver colisão devida à falta desta informação que precisa estar contida na representação do mapa, para que sejam tomadas ações preventivas. Esta situação é apresentada na Figura 16.

Figura 16. Robô em colisão com carga alocada incorretamente, no momento de recarregar uma carga alocada em um nível superior.

Na abordagem MLS é possível representar o ambiente independente de suas estruturas verticais. No capítulo seguinte está descrito o método de mapeamento MLS.

3.3 Mapeamento 3-D utilizando Multi Level Surface (MLS)

Neste capítulo exploramos o mapeamento 3-D,

“O algoritmo de mapeamento de objetos 3-D explora um pedaço específico da informação[...] nós temos um enorme conhecimento sobre os objetos que nos

rodeiam no ambiente típico doméstico, da vegetação em ambientes abertos, de objetos encontrados no fundo do mar, e assim por diante. Em muitos casos, temos informações adicionais sobre um determinado ambiente[...]” (THRUN, 2002) (p. 24)

Com utilização de giroscópio, IMU, de odometria e de sensores laser é possível realizar a tarefa de mapeamento do ambiente adquirindo informações suficientes para a construção de mapas 3-D. Mapas em 3-D necessitam mais informações que mapas 2-D: informações essas que são relativas à inclinação do robô em relação ao solo ou a qualquer outro sistema de referência.

Um laboratório de uma faculdade é um ambiente dinâmico, no qual encontramos caixas (espalhadas, empilhadas e dos mais diversos tamanhos), corredores de máquinas, pessoas e pontos de interesse (bases de alimentação, pontos de carga e descarga, passagens possíveis e impossíveis, e sinalizações). Nesse ambiente foi incluída a tarefa de o robô transportar um objeto de um ponto a outro.

No ambiente proposto é indispensável abstrair o meio em que o robô se encontra e é ainda maior a necessidade de representar os seis graus de liberdade, devido a portas, maquinário parado e em operação. Trabalhos para tratar estes seis graus de liberdade foram realizados por Lamon et. al. (2006), Triebel et. al. (2006) e Kümmerle et. al. (2007) ; e uma nova abordagem de SLAM foi apresentada por Nüchter (2009) – a qual atualmente é denominada SLAM6D – realiza a tarefa de SLAM utilizando os seis graus de liberdade, a saber, os eixos x , y e z e os ângulos *roll*, *pitch* e *yaw*.

Um trabalho na área de computação apresentado por Diebel et. al. (2006), mostra uma técnica Bayesiana seguida de um método utilizado em processamento de sinais – a dizimação – técnica com a qual é possível realizar a reconstrução de modelos de superfícies 3-D a partir de dados ruidosos dos sensores. A técnica desenvolvida foi aplicada no contexto de animação por computador. Porém, este método não pode ser utilizado em robôs devido ao tempo de processamento requerido. Na robótica existe a necessidade de realizar simultaneamente mapeamento e localização.

Uma abordagem para o mapeamento 3-D – conhecida como MLS foi proposta por

Lamon et. al. (2006). Um carro autônomo é apresentado e utiliza a técnica MLS para mapeamento; o objetivo do carro é navegar por ruas e desviar-se de obstáculos; ele é dotado de três sensores laser SICK LMS291, fixos ao carro, que são utilizados para navegação local e para desvio de obstáculos. Outros dois sensores laser SICK LMS291 são montados rotacionando no eixo correspondente à estruturas verticais e uma câmera omni-direcional é utilizada para aquisição de imagens panorâmicas e para extração de texturas do ambiente. Um mapa local MLS pode ser combinado com um mapa global MLS, sem necessidade de dados brutos. O processo de combinar os mapas normalmente é realizado com a utilização de algoritmos de ICP – *Iterative Closest Point*; de acordo com Pfaff e Burgard (2005), com poucas iterações é possível conseguir combinações precisas e produzir mapas com alta precisão a partir de mapas locais obtidos em diferentes pontos de visão.

Na proposta apresentada por Triebel et. al. (2006) foi apontada a necessidade da representação de estruturas verticais como pontes, passagens subterrâneas ou minas. Um robô *pioneer* dotado com odometria, sensor laser acoplado a um módulo AMTEC *tilt* realiza o movimento no eixo vertical para realizar a varredura de estruturas verticais. O robô percorreu uma área de aproximadamente 195 x 146 metros do campus da universidade e em seu trajeto realizou duas voltas fechadas; também trafegou por uma ponte e atravessou uma passagem subterrânea. Com a abordagem de mapas MLS, foi apresentada redução no uso da memória comparada com a representação que utiliza a nuvem de pontos; a precisão foi mantida em níveis satisfatórios, e também mostrou ser adequada para realizar a estimação da posição global e tarefas de circuitos fechados.

No trabalho apresentado por Kümmerle et. al. (2007), com uso de um filtro de partículas, os autores obtiveram o estado nas seis dimensões do robô. O robô utilizado foi o Pioneer dotado com odometria, sensor laser acoplado a um módulo AMTEC *tilt* que realiza o movimento no eixo vertical para realizar a varredura de estruturas verticais. O método de Monte Carlo foi utilizado para obter a localização do robô. Nos experimentos, o uso de mapas MLS convergiram mais rapidamente para a localização do robô, enquanto mapas elevados necessitam maior tempo para convergir e não obtiveram 100% de convergência de suas partículas. A Figura 17 mostra a eficiência apresentada pela técnica MLS em relação a mapas elevados.

Os mapas MLS propostos neste trabalho otimizam o cálculo de mapas consistentes. Ele resolve de forma eficiente o problema de associação que ocorre quando dois mapas com diferentes estimativas sobre níveis de superfície têm de ser parelhados ou combinados.

Figura 17. A vantagem da abordagem de mapas MLS em comparação com mapas elevados padrão. Em contraste com mapa MLS (direita) a mapa elevado (à esquerda) não tem a capacidade para modelar estruturas verticais, porque ele faz a médias de todos os valores de altura medidos. Uma vez que a distância do ponto de extremidade de um feixe de laser para o próximo ponto no mapa elevado pode ter desvios substanciais da verdadeira distância, a localização torna-se difícil, Fonte: Kümmerle et. al. (2007), cortesia de Rainer Kümmerle.

Na proposta apresentada por Pfaff e Burgard (2005), para realizar a tarefa de mapeamento com a técnica MLS, o robô é dotado de um sistema de odometria, sensor laser acoplado a um módulo AMTEC *tilt*. Esta configuração tem a finalidade de poder variar as leituras do sensor laser e ser possível reconstruir o ambiente, calculando a diferença entre as leituras obtidas em diferentes posições do sensor o que aumenta a precisão do mapa.

Os mapas MLS propostos neste trabalho otimizam o cálculo de mapas consistentes. Ele resolve de forma eficiente o problema de associação que ocorre quando dois mapas com diferentes estimativas sobre níveis de superfície têm de ser parelhados ou combinados.

No capítulo 2, item 2.3.1, inicia na página 54 e segue até a página 60, é mostrado o modo como a representação MLS realiza a tarefa de mapeamento e, também, ilustra a funcionalidade do sensor acoplado ao módulo AMTEC, processando os dados e armazenando informações suficientes para conseguir representar o ambiente.

Diante do escopo deste trabalho e da necessidade da precisão exigida, de início foi

apresentado a técnica de mapeamento em *grid* de ocupação que serve como base para representar ambientes adquiridos a partir dos sensores do robô. Em busca de soluções para atender às necessidades de precisão e visando trabalhos futuros, em considerações sobre a técnica de SLAM, pode ser concluído que para ambientes como armazens no quais o robô transita inúmeras vezes pelo mesmo local, a referida técnica apresenta vantagens como aumento na precisão sobre a localização do robô e aumento na precisão da representação de mapa obtido. A técnica de SLAM tem sido muito utilizada em trabalhos científicos, mas ainda não foram encontradas referências de uso em aplicações industriais o que sinaliza bom caminho para novos trabalhos e futuros desenvolvimentos.

Neste trabalho também foi apresentada uma introdução sobre técnica de construção de mapas MLS, a qual é utilizada para representar ambientes 3-D. Para ser obtida a representação do ambiente não são descartadas as aquisições de dados realizadas pelos sensores. Os dados são combinados através de algoritmos de ICP.

Em Triebel et. al. (2006) são apresentadas as etapas para criação de mapas MLS, nos sub-item de 2.3.1 a 2.3.4 deste trabalho são apresentadas estas etapas.

3.3.1 Construção de mapas MLS

De acordo com Triebel et. al. (2006), dado um conjunto de N pontos de digitalização 3-D, $C = \{ p_1, \dots, p_N \}$ com $p_i \in \mathbb{R}^3$, e um conjunto de variâncias $\{ \sigma_1^2, \dots, \sigma_N^2 \}$, a variância σ_i^2 expressa a incerteza no intervalo de medição em que o ponto p_i foi computado. Adotada incerteza igual em todas as três dimensões, particularmente a variação no tamanho é assumida sendo idêntica a σ_i^2 . Esta suposição é violada em ambientes reais, mas é viável para a proposta deste trabalho. Para isso, definimos a medição z como um par (p, σ^2) , com um ponto 3-D e a variância.

3.3.2 Criação do mapa MLS

Em Triebel et. al. (2006), a criação de mapas MLS pode ser executada de duas formas: a criação pode ser feita a partir de um conjunto de pontos 3-D; uma nuvem de pontos com variâncias ou por meio da união de dois outros mapas MLS em um. As duas formas são equivalentes, criando o mapa m_1 a partir de uma nuvem de pontos C_1 e o mapa m_2 a partir da nuvem de pontos C_2 , o mapa m_3 resultante da junção dos mapas m_1 e m_2 é idêntico ao mapa gerado pela união da nuvem de ponto $C_3 = C_1 \cup C_2$. Para dada nuvem de ponto C com variâncias $\sigma_1, \dots, \sigma_N$, podemos criar mapas MLS utilizando os seguintes procedimentos:

- Cada célula do mapa com índice (i, j) armazena todos os pontos $p = (x, y, z)$, para $si \leq x \leq s(i+1)$ e $sj \leq y \leq s(j+1)$ onde s representa o tamanho (comprimento da borda) de uma célula no mapa.
- Para cada célula é calculado um conjunto de intervalo de alturas dos valores de altura dos pontos armazenados. Quando dois valores consecutivos de alturas estão mais perto do que um determinado intervalo, ele passa a pertencer ao mesmo intervalo. Isso faz com que dois intervalos estejam metros de distância um do outro. O tamanho do intervalo pode ser escolhido de modo tal que um robô que navegue pelo mapa possa passar no intervalo, de forma que seja bem maior que o tamanho do robô. Neste trabalho vamos definir os seguintes valores para testes 0.1 m, 0.5 m e 1 m.
- Os intervalos são classificados como estruturas horizontais e verticais. Estas estruturas são diferenciadas de acordo com o tamanho do intervalo. Se o valor da espessura exceder $T = 10 \text{ cm}$, ele é considerado vertical, caso contrario é horizontal.
- Para cada intervalo classificado como vertical, são armazenadas a média e a variância da maior medição no intervalo. É intuitivo dizer que para a transversalidade as medições de valores mais altos são relevantes. Vale lembrar que também é importante armazenar o comprimento do intervalo, que é identificado pela profundidade d . Este valor é utilizado quando são combinados 2 mapas MLS em um só.
- Para cada objeto horizontal na célula, deve ser calculada a média μ e a variância σ

para todas as medidas no intervalo. Isto é feito aplicando a regra de atualização de Kalman para todas as medições. A profundidade d de um objeto horizontal é ajustado para 0.

Após serem calculadas a média, a variância e a profundidade, devem ser abandonados os dados da nuvem de pontos. Os demais cálculos são realizados usando apenas as informações do mapa. Esta técnica reduz substancialmente a memória necessária para um mapa MLS em comparação com uma nuvem de pontos e isso sem perder a precisão de representação.

3.3.3 Representação do mapa

Na proposta de Triebel et. al. (2006), um mapa MLS é representado por um grid 2-D de tamanho variável, no qual cada célula $c_{ij}, i, j \in \mathbb{Z}$ armazena uma lista de trechos de superfície $P_{ij}^1, \dots, P_{ij}^K$. O trecho de superfície é representado pela média μ_{ij}^k e pela variância σ_{ij}^k da altura medida na posição da célula c_{ij} no mapa. Cada trecho de superfície na célula reflete a possibilidade de cruzar o ambiente 3-D; a altura é dada pela média μ_{ij}^k e a incerteza da altura é dada através da variância σ_{ij}^k . Neste trabalho foi assumido que o erro na altura está de acordo com uma distribuição gaussiana e os termos *trecho de superfície* e *gaussiana* serão adotados em uma célula que seja passível de mudança.

Pfaff e Burgard (2005), propuseram aplicar o filtro de Kalman para estimar a variância e o desvio padrão da modificação em altura da célula. Na sequência são incorporadas as novas medidas z_t com desvio padrão σ_t no tempo t . Com a utilização do filtro de Kalman, é possível levar em consideração a incerteza da medição. Este modelo é utilizado quando a variação da altura de uma medição aumenta linearmente com a distância do feixe correspondente. Este processo pode ser observado na Figura 18.

Além de o mapa armazenar a média e a variância da modificação em altura da célula, ele também armazena a profundidade de valor d para cada trecho. Com este valor de profundidade é possível dizer se um trecho de superfície é uma estrutura vertical como portas, maquinário, rampas ou estruturas côncavas. Para casos como este, a profundidade é definida pela diferença da altura h_{ij}^k do trecho de superfície e a altura h'_{ij}^k da menor medida que é considerada pertencendo à vertical do objeto. Para objetos planos como chão, a profundidade é igual a 0. Alguns exemplos das células em um mapa MLS são mostradas na Figura 19.

Figura 18. variância de medições de uma altura dependendo da distância do feixe, Fonte: cortesia de Patrick Pfaff.

Cada célula possui comprimento, altura e profundidade ou largura e um conjunto dessas células pode determinar certo trecho de superfície, mais ainda, poderá determinar ou conter vários trechos de superfície. A Figura 19 mostra aglomerado de células em mapa MLS. As células estão agrupadas em coluna. μ

56

representa a medida da altura de uma célula - no caso de uma célula do conjunto A. A figura mostra, também, as médias da altura e da variância das células do conjunto A. Objetos no plano são representados com profundidade 0.

Figura 19.Exemplo de diferentes células em um mapa MLS, Fonte:Triebel et. al. (2006), cortesia de Triebel.

3.3.4 Atualização do mapa

Para Triebel et. al. (2006), sempre que uma nova medição $z=(p,\sigma)$ é inserida em um mapa MLS, primeiramente é necessário verificar se esta leitura é um objeto já representado no mapa ou se ela corresponde a um novo objeto. Para isso, é preciso determinar a célula c_{ij} em que o ponto medido falha. Então é encontrada a Gaussiana $(\mu_{ij}^k, \sigma_{ij}^k)$ em

c_{ij} cuja média μ_{ij}^k é a altura mais próxima de z . Se a gaussiana estiver próxima o suficiente, o mapa é atualizado com a nova medida de z , novamente utilizando a regra de atualização de Kalman. A Gaussiana é considerada próxima de z se o valor da altura z estiver dentro dos limites de 3σ a Gaussiana.

Se z estiver longe da próxima Gaussiana, ainda existe a possibilidade de corresponder a um objeto vertical. Esta informação é encontrada verificando se z está dentro de uma ocupação da Gaussiana na célula. Se isso ocorrer deve ser ignorada a medição z , devido ao fato de existir um objeto vertical sobre a Gaussiana. Caso não possua, z é introduzido como uma nova Gaussiana na célula.

4. Métodos teóricos do projeto

Os métodos utilizados para criação e atualização do mapa deste trabalho são descritos nas seções 4.3, 4.3.1 e 4.3.2. Para este trabalho foram usados os métodos de mapeamento em *grid* descrito em Thrun et. al. (2005) e o método teórico proposto em Triebel et. al. (2006), o qual pode ser implementado na linguagem de programação C para garantir portabilidade entre sistemas operacionais.

Este modelo de armazenamento de mapas, quando aplicado a tarefas de localização, apresentou aumento na taxa de sucesso de localização superiores aos mapas elevados, o erro em relação a translação apresentou valores inferiores apresentados, por exemplo, em Kümmerle et. al. (2007) utilizando o método de Monte Carlo.

O método GMM foi estudado para auxiliar a tarefa de mapeamento detectando movimentos, o resultado é uma imagem binária. O plano de fundo são representados pela região na cor preta e o primeiro plano que são objetos móveis na cor branca. O método utiliza como base teórica a proposta de STAUFFER e GRIMSOM (1999).

As implementações desenvolvidas para o experimento que utiliza o método ICP e a biblioteca MRPT (2013) como base de desenvolvimento, a representação do mapa é obtida através da ocupação em *grid*.

5. Experimentos de mapeamento: robô móvel e manipuladores robóticos

Neste capítulo são apresentados os componentes de software e hardware necessários aos experimentos realizados no decorrer deste trabalho; foram realizados experimentos, os dados foram adquiridos com o Player server. Para construção do algoritmo de mapeamento foram utilizados os dados do experimento no simulador do Player, mas é importante descrever o hardware embarcado no robô móvel.

5.1 Softwares utilizados

Nesta subsecção são mostradas as decisões tomadas no projeto com relação aos softwares. Também é apresentado um breve detalhamento na escolha do sistema operacional, plataforma cliente/servidor fornecida pelo Player e o VRMLVIEW.

5.1.1 Sistema Operacional

O sistema operacional escolhido foi o Linux, devido ao fato de ser um software gratuito e, bem por isso, ter largo uso em diversos centros de pesquisa no mundo.

Outros fatores que pesam na opção pelo sistema operacional Linux são os seus bons serviços de redes e seus recursos de compartilhamento de tempo. Thrun et. al. (2006) consideram que a frequência para aquisição de dados dos sensores de 100Hz; para controlar a direção, acelerador e freios é de 20Hz. Estas frequências podem ser garantidas com a utilização de serviços de tempo real fornecido pelo Linux.

Para este trabalho que utiliza a plataforma cliente/servidor fornecida pelo *software Player*, o robô tem proporção menor que a mencionado em Thrun et. al. (2006). Com esta

aplicação temos como objetivo robôs *indoor*, um estudo sobre os softwares e hardwares que podem ser adotados no projeto.

O Linux é um sistema operacional livre compatível com o padrão POSIX (*Portable Operating System Interface*) definido pela IEEE 1003, utilizado em diversos projetos apresentado por Thrun et. al. (2006), Nüchter et. al. (2007) e Aroca (2008). O *Real Time Linux* possui características que são importantes para tarefas em robótica móvel como bom serviço de rede; o módulo RTAI instalado junto ao *kernel*, apresentou bons resultados em testes de *real-time* à respostas de interrupções e gerenciamento de processos.

Sistemas de tempo real são utilizados em aplicações embarcadas que necessitam robustez e segurança. Segundo Labrosse (2002), um sistema de tempo real é caracterizado pelo fato de um resultado lógico ou propriedades de temporização correta do sistema não serem preenchidas e gerarem sérias consequências. Alguns exemplos de áreas de aplicações: controle de processos (processamento de alimentos e fábricas de produtos químicos), automotiva (engenharia de controle e sistemas de freios antitravamento), automação de escritórios (máquinas de FAX e copiadoras) periféricos de computadores (impressoras, terminais, *scanners*, *modems* e outros), comunicação (*switches* e roteadores), robótica (robôs industriais, robôs móveis, robôs domésticos e outros), aeroespacial (sistemas de gerenciamento de voos, sistemas de armamentos e sistema de motores a jato) e utilitários domésticos (fornos micro-ondas, máquinas de lava roupas, , máquinas de lavar louças e termostatos).

Na definição de Dozio e Mantegassa (2003), os sistemas de tempo real são divididos em 2 categorias principais, os *soft real time* (SRT) e *hard real time* (HRT). Os sistemas *soft real time* são caracterizados por sua capacidade de executar uma tarefa com um tempo predefinido, em médias. Uma exibição de vídeo pode ser utilizada como exemplo típico de SRT. Devido à dinâmica do olho humano, a perda ocasional de um *frame* não causa prejuízos ao sistema de percepção e seu desempenho, em média, continua aceitável. Técnicas como interpolação, por exemplo, podem ser utilizadas para haver essa compensação. Por sua vez, os sistemas *hard real time* são caracterizados por incorporar garantias – eles não podem perder os prazos; pois suas latências são extremamente limitadas e suas aplicações dependem do problema envolvido. Os sistemas HRT não podem se valer de desempenhos de casos

médios como compensação para casos com desempenhos piores. Eles são os HRT são compostos por um sistema de controle (computador) e um sistema controlado (planta). O estado corrente da planta, percebido pelo sistema de controle é consistente com o estado real da planta contendo uma margem de erro aceitável(nível de ruído). Podem acontecer requisitos de tempo correto em aplicações que utilizam controle para atuação, e, nestes casos devem ser usadas taxas de amostragem de acordo com a proposta do sistema de controle de tempo discreto projetado.

5.1.2 *Player*

Conforme reportado por Gerkey et. al. (2003), o *Player* é um servidor de rede para controlar robôs. Na terminologia de sistemas operacionais o *Player* é uma camada de abstração de *hardware* (HAL) para dispositivos robóticos. Para o sistema operacional ele oculta os detalhes do *hardware* do computador com definições genéricas e cada detalhe possui seu padrão de interface. Para inclusão de um dispositivo especial, o suporte para este dispositivo é tratado por um controlador (*driver*).

Um ponto forte na utilização do *Player* é apresentado na Figura 20, na qual as letras P em vermelho representam *Player servers*, e as letras C em azul representam *Player clients*. Os dispositivos como câmeras e *lasers* não precisam ser incluídos no robô, pois fazem parte do sistema como um todo – máquinas podem apenas trabalhar como *data logger* e servidores armazenam mapas para serem utilizados pelos robôs em operação no ambiente.

Outra característica do *Player* a ser ressaltada é fornecer mecanismos de transporte que permitem a troca de dados entre o controlador e programas de controle que são executados em diferentes máquinas. Atualmente, o mais comum transporte para plataformas cliente/servidor são baseados em *TCP-sockets*. O *Player* é executado com um arquivo de configuração que define quais controladores devem ser instanciados e como eles são ligados ao *hardware*. Os controladores são executados no servidor *Player* (muitas vezes em várias *threads*) e o programa do usuário executa como um cliente para este servidor.

Figura 20. Estrutura suportada pelo Player, (GERKEY et. al., 2001).

Para este trabalho foi escolhido o uso da linguagem C.

5.1.3 VRMLVIEW

O software VRMLVIEW foi utilizado como visualizador para os dados extraídos do experimento. Ele interpreta um código desenvolvido utilizando a linguagem VRML (Virtual Reality Modeling Language) definida pelo padrão (INTERNATIONAL STANDARD, 2010).

A vantagem no uso deste visualizador, é a velocidade para realizar tarefas como rotação nos pontos adquiridos para obter outros pontos de visão do mapa coletado durante o experimento.

5.1.4 Mobile Robots Programming Toolkit

A biblioteca Mobile Robot Programming Toolkit possui classes e métodos que nos permitem facilmente acessar dados armazenados em arquivos ou adquirir-los através de um dispositivo conectado ao computador (cameras, lasers, kinect), por este motivo foi utilizada

em nosso trabalho. Os códigos gerados neste trabalho foram desenvolvidos para leitura de arquivos que contém conjunto de dados no formato proposto pela biblioteca (arquivos .rawlog) ou por um dispositivo kinect conectado. A MRPT permite ao usuário geração de arquivos .rawlog quando possui um dispositivo suportado conectado ao computador.

A MRPT é uma biblioteca de altíssimo nível para demais soluções que envolvam robótica móvel ou manipuladores robóticos. Sua documentação favorece o aprendizado e seus exemplos facilitam a compreensão.

Em seu código podemos encontrar algoritmos para localização (Filtros de Kalman, Método de Monte Carlo, etc), extratores de características (SURF, NARF), algoritmos para busca de correspondências (ICP), representação de nuvens de pontos (Mapas simples ou coloridos). Na atual distribuição lançada no ano de 2013 foi implementado a representação de mapas utilizando o método de árvores octree.

5.1.5 OpenCV

O processamento dos quadro de imagens adquiridos foi realizado por métodos implementados na biblioteca de desenvolvimento OpenCV mantida atualmente pela equipe da Willow Garage, empresa que desenvolve *hardware* e *software* para aplicações de robôs pessoais. É importante ressaltar que esforços são realizados para definir padrões na área da robótica, projetos como *Robotic Operating System* (ROS) e *Point Clouds Library* (PCL) utilizam a OpenCV integrada em seu núcleo ou para auxiliar o processamento de imagem. Ambos os projetos fornecem bibliotecas e ferramentas de altíssimo nível para o desenvolvimento de aplicações robóticas.

5.1.6 OpenNI

No final de 2010, com o surgimento de novas tecnologias para monitorar ambientes, movimentos e gestos foi criado o OPENNIORG (2013), um consórcio formado por empresas

de cinco setores distintos, a citar as empresas e seus setores são: PrimeSense especializada em sistemas de interação natural e aquisição de dados 3-D, Willow Garage especializada em aplicações de robôs pessoais, Asus fabricante da hardware para aplicações de jogos, SideKick especializada em jogos em que o controle é realizado através de movimentos e Open Perseption sem perfil definido mas colaborou criando a Point Cloud Library (PCL).

Este consórcio foi criado com o objetivo de definir padrões e compatibilidade e interoperabilidade de interações naturais, dispositivos, aplicações e *middleware*.

Atualmente, o consórcio OPENNIORG (2013) é um grande framework de desenvolvimento e uma comunidade formada por especialistas de diversas áreas. É fornecido gratuitamente um SDK para desenvolvimento de aplicações que necessitam detectar movimentos das mãos, braços, pernas ou até mesmo combinações no esqueleto humano. Também são encontradas diversas publicações das mais diversas áreas que utilizam o SDK de desenvolvimento.

O SDK de desenvolvimento atualmente esta disponível para os seguintes sistemas operacionais: Windows, Linux, OSX, Android.

5.1.7 Tecnologia CUDA

As placas aceleradoras gráficas Nvidia Geforce utilizadas no trabalho possuem a *Compute Unified Device Architecture (CUDA)*, esta tecnologia desenvolvida pela Nvidia propõe computação paralela e um método de programação visando o aumento na performance computacional como apresentado em NVIDIA (2012).

Placas aceleradoras gráficas possuem uma unidade para processamento gráfico (GPU). Esta tecnologia utiliza a arquitetura *single operations multiple data* (SIMD) para realizar o processamento em paralelo de tarefas massivas.

Algumas precauções devem ser adotadas ao desenvolver uma solução para processamento em paralelo. Em alguns casos, o custo de tempo para deslocar os dados para a memória pode exceder o ganho com o processamento disponibilizado. Outro fator a ser observado é criar diversas ramificações, o ganho de tempo pode apresentar perdas durante o

processo de sincronização e tarefas de sincronização podem não ser triviais.

5.2 Hardwares utilizados

Nesta subsecção são mostradas as decisões tomadas no projeto com relação aos hardwares e suas interfaces com o software.

5.2.1 ActiveMedia Pioneer, o robô

A plataforma robótica escolhida para desenvolvimento deste trabalho foi o ActiveMedia Pioneer por estar presente em experimentos relacionados à localização e mapeamento Thrun et. al. (2000b), Wolf E Sukhatme (2003) e presente, também, em experimentos recentes Kümmerle et. al. (2007). Este robô vem equipado com odometria e sonares. O software Player possui interface de comunicação com os hardwares de odometria e sonares, convertendo sinais dos sensores em unidades do sistema de medida.

O Pioneer possui um computador pessoal em sua estrutura com interfaces de comunicações no padrão RS-232, USB e outras fornecidas pela arquitetura dos computadores pessoais. Estas interfaces normalmente são utilizadas para conexão dos sensores.

5.2.2 Laser SICK

Para aquisição de dados que correspondem à distância de objetos, obstáculos, pessoas e paredes foi utilizado o sensor SICK LMS-200; o software Player realiza o interfaceamento entre hardware e software, trazendo as leituras armazenadas na forma de ângulo e distância medida.

O laser fornece uma matriz com 360 leituras, variando $0,5^\circ$ entre cada uma e cobrindo 180° do campo de visão do *bearing*.

5.2.3 Computadores pessoais

Os experimentos para manipuladores robóticos foram realizados em 3 computadores pessoais e equipados com hardware distinto, para compreensão do trabalho vamos chama-los de A, B, C e abaixo seguem as configurações:

A - Um notebook com processador Core 2 Duo, 4 Gb RAM e sistema operacional openSuse Linux versão 11.2 plataforma x86 32 bits.

B - Um computador de mesa equipado com processador core i3, 4Gb RAM, sistema operacional openSuse Linux versão 11.4 plataforma x86 64 bits e placa aceleradora gráfica Nvidia Geforce GT 220 1Gb RAM.

C - Um notebook com processador core i5 2.5Ghz, 4 Gb RAM e sistema operacional openSuse Linux versão 12.2 plataforma x86 64 bits e placa aceleradora gráfica Nvidia Geforce 610 M com 2Gb RAM.

5.2.4 Kinect

No final de 2010, foi lançado o sensor de movimento Kinect que revolucionária o incrível mundo dos jogos eletrônicos, com uma nova forma de controlar os personagens através de gestos realizados com as mãos ou movimentos do corpo, uma nova forma de interagir com outros personagens através de linguagem falada e reconhecimento facial. A princípio foi lançado como dispositivo para conectar ao Xbox 360. Logo em seguida a seu lançamento, entusiastas das mais diversas áreas a citar, reconhecimento de imagens e especialistas em robótica, voltaram suas atenções. O Kinect vem equipado com uma camera RGB com resolução capaz de adquirir quadro de imagens com 640 x 480 *pixels*.

O Kinect também é capaz de adquirir nuvens de pontos e tem sido utilizado como *depth range sensor* por pesquisadores e entusiastas da robótica. O Kinect possui compatibilidade com computadores pessoais utilizando conexão USB.

Comparado com sensores lasers, o sensor de movimento Kinect possui baixo custo. Por outro lado, sua área de atuação é limitada a 6m² de acordo informações contidas no manual e recomendações do fabricante que podem ser verificadas em KINECT (2013) e MSDN (2013).

Outra vantagem apresentada pelo Kinect é o acelerômetro capaz de dizer a orientação do sensor em 2-D.

O baixo custo e compatibilidade com computadores pessoais, faz do Kinect uma opção capaz de auxiliar o reconhecimento e mapeamento de ambientes.

Conforme avaliações realizadas em Pedro (2013), nas aplicações que requerem precisão é recomendado fixar o Kinect a uma distância de 1 m à 1,5 m da área de atuação do robô manipulador. No processo de calibração adotado, o erro é menor nas proximidades do sensor e aumenta na medida que esta distância aumenta.

5.3 Experimento de mapeamento para robótica móvel

O objetivo do experimento foi fazer o robô percorrer o laboratório de dinâmica do Departamento de Engenharia Mecânica da Escola de Engenharia de São Carlos na Universidade de São Paulo realizando leituras de odometria e *laser* para a construção de um mapa 3-D. Para compreensão do experimento foram marcados os pontos A e B na Figura 21 e foi produzido um vídeo descrevendo o experimento, o qual pode ser encontrado no sítio (MAPEAMENTO3DEESCUSP, 2010).

O robô tinha por objetivo sair do ponto A e atingir o ponto B, com uso da interface de controle playerjoy, e percorreu o trajeto marcado na Figura 21;. Durante este trajeto seus sensores, laser e odometria realizaram leituras que foram armazenadas em log. A seguir foi percorrido o trajeto inverso – partir de B e chegar ao ponto A. Vale ressaltar que neste experimento o robô andou apenas em linha reta, o que permitiu a criação do mapa desconsiderando erros de odometria; outras tarefas devem ser implementadas antes de experimentos mais complexos como *loop closed* e SLAM . Para tais tarefas há necessidade de um módulo de localização preciso para os erros não serem computados no mapa.

Figura 21. Indicação do trajeto percorrido pelo robô. Trajetória de A à B e B à A.

Quatro trajetos nomeados `log_ida` e `log_volta` foram gerados. Os logs foram gerados configurando o servidor Player conforme a Listagem 1. Com os arquivos de log gerados, foi possível reproduzir o experimento inúmeras vezes simulando as leituras adquiridas com a utilização do Player. Para realizar esta tarefa foi utilizado o arquivo de configuração da listagem 2.

No arquivo apresentado na Listagem 1, foram carregados os drivers de odometria, laser e geração de log. É importante resaltar que os logs foram gerados no próprio robô e adquiridos após o experimento através de conexão remota. A Listagem 1 descreve o arquivo de configuração para utilização do *dataset* adquirido com o servidor Player durante o experimento apresentado neste

capítulo.

1. Listagem: Arquivo de configuração do software Player para repetir o experimento offline.

```

1. driver
2. (
3.   name "readlog"
4.   provides ["log:0" "laser:0" "position2d:0"]
5.   filename "writelog_ida1.log"
6.   speed 1.0
7. )

```

Com o ambiente de simulação do experimento programado, foi criado o algoritmo para mapeamento; a base do desenvolvimento desse algoritmo está descrita em Thrun et. al. (2005) e parte do princípio proposto por Elfes (1987) conhecido como mapas de ocupação em *grid*.

Os dados obtidos dos sensores, denominados x , as informações fornecidas pela odometria formadas pela tupla (x, y, Θ) , e as leituras do laser z variando $0,5^\circ$ entre elas possibilitaram a implementação do algoritmo. Utilizando o componente x fornecido pela tupla da odometria. Através dos ângulos das leituras e da distância do objeto fornecidas pelo laser foram calculadas as componentes y e z . Esta solução foi alcançada por meio do experimento realizado, no qual foi considerada a variação do movimento na componente x , conforme sistema de coordenadas apresentado na Figura 25. Para definir o conjunto de leituras, foi usado o cálculo *a posteriori* apresentado em Thrun et. al. (2005) que definiu a equação (4.3.1):

$$p(m|z_{1:t}, x_{1:t}) \quad (4.3.1)$$

O mapa denominado m contém todas as leituras coletadas pelo sensor laser durante o experimentos e são definidas $z_{1:t}$; e as leituras que definem o trajeto do robô realizadas pela

odometria são definidas $x_{1:t}$. Em mapas de ocupação em *grid*, os controles aplicados $u_{1:t}$, não são utilizados porque as informações de deslocamento do robô são conhecidas e contém erros que foram desprezados.

Para representação 3-D, os mapas de ocupação em *grid* possuem alto custo computacional, mas eles podem ser contornados através da otimização do algoritmo. Se for aumentada a variação angular entre as leituras utilizadas pelo algoritmo, será aumentada a velocidade de processamento das células. Aqui foi adotada m_i , em que i representa uma célula do grid de ocupação,. O mapa é representado pelo conjunto finito de algumas células e representado na equação (4.3.2):

$$m = \{ m_i \} \quad (4.3.2)$$

Na proposta apresentada por Thrun et. al. (2005), um valor binário foi atribuído à célula m_i , e a ele foi atribuído valor 1 quando a célula estiver ocupada e valor 0 quando vazia.

A dimensionalidade do mapa envolve a probabilidade *a posteriori* apresentada na equação (4.3.1); o número de células do mapa representado pela equação (4.4.2), pode atingir dezenas de milhares. Utilizando um mapa com 125 mil células, o número de mapas que pode ser representado por este mapa é de 2^{125000} , o cálculo da probabilidade *a posteriori* de um único mapa passa a ter difícil tratamento ou pode, até mesmo, ser intratável.

Este problema de estimação pode ser solucionado pela divisão dos mapas em uma coleção de problemas separados., definindo a estimativa para todas as células do mapa m_i como apresentado na equação (4.3.3).

$$p(m_i | z_{1:t}, x_{1:t}) \quad (4.3.3)$$

Cada estimativa é transformada em um problema binário com estado estocástico. Tal decomposição pode atender às necessidades, mas não está livre de falhas. Como a dependência entre as células vizinhas não são representadas por este modelo, foi utilizado o

produto das probabilidades das células marginais, conforme ilustrado na equação (4.3.4).

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \quad (4.3.4)$$

Através da fatoração, a estimativa da probabilidade de ocupação de cada célula do *grid* passou a ser representada como uma estimação binária com estado estocástico. Na robótica, alguns problemas são melhor formulados por estimação binária que não muda ao longo do tempo. Tal fato surge quando um robô estima uma quantidade binária fixa no ambiente proveniente de uma sequência de leituras do sensor. O filtro Bayes binário é aplicado na construção de mapas de ocupação em *grid*, para cumprir a tarefa de estimar a crença *a posteriori*. Quando existe um estado estocástico, a crença é somente em função das medições, conforme mostrado na equação (4.3.5).

$$bel_t(x) = p(x|z_{1:t}, x_{1:t}) = p(x|z_{1:t}) \quad (4.3.5)$$

O estado é escolhido entre dois valores possíveis, pode ser x ou $-x$. Na crença $bel_t(-x) = 1 - bel_t(x)$, o estado x não possui índice no tempo, o que mostra que este estado não varia. Normalmente, a crença é implementada pelo cálculo da taxa das probabilidades. A equação (4.3.6) considera que as probabilidades de um estado x é definida pela razão da probabilidade deste evento dividido por sua negação.

$$\frac{p(x)}{p(-x)} = \frac{p(x)}{1 - p(x)} \quad (4.3.6)$$

A partir da equação (4.3.6) podemos calcular a taxa logarítmica de probabilidade, ilustrada pela equação (4.3.7).

$$l(x) := \log \frac{p(x)}{1 - p(x)} \quad (4.3.7)$$

Os cálculos assumem valores de $-\infty$ a ∞ . Desta forma é possível evitar os truncamentos que surgem para probabilidade que esteja entre 0 e 1. O algoritmo é aditivo – qualquer algoritmo que aumenta ou diminui uma variável na resposta às medições pode ser interpretado como um filtro de Bayes na forma de cálculo das probabilidades. Este filtro binário de Bayes usa um modelo inverso de medições $p(x|z_t)$, as células que antecedem a leitura medida tem sua credibilidade reduzida. O modelo inverso das medições especifica uma distribuição sobre a variável de estado (binário) em função das medições z_t . Os modelos inversos geralmente são utilizados em situações nas quais as medições são mais complexas que o estado binário. Por exemplo, como é possível afirmar que uma porta esteja aberta ou fechada através de uma imagem adquirida com uma câmera? Os estados são muito simples, a porta pode estar aberta ou fechada, mas todas imagens adquiridas são extremamente grandes. A solução é mais simples se for calculada a probabilidade de a porta estar fechada a partir de uma imagem da câmera, que descrever todas a distribuição sobre todas imagens da câmera que mostram a porta fechada.

Com a equação (4.3.7) em mapas de ocupação em *grid* é obtida a posição local da célula indexada pelo tempo t e cada célula do *grid*, é pode ser obtida da equação (4.3.8):

$$l_{t,i} = \log \frac{p(m|z_{1:t}, x_{1:t})}{1 - p(m|z_{1:t}, x_{1:t})} \quad (4.3.8)$$

Como o sensor *laser* possui variação no ângulo de suas medições, o algoritmo percorre todas as células do *grid* e atualiza as que estão no campo de visão das leituras do sensor. Para realizar as atualizações do mapa de ocupação em *grid* a implementação do algoritmo está baseada na equação (4.3.8). Também pode ser utilizada uma constante l_0 que armazena um valor *a priori* da ocupação representada como uma taxa logaritmica de probabilidade obtida pela equação (4.3.9)

$$l_0 = \log \frac{p(m_i=1)}{p(m_i=0)} = \log \frac{p(m_i)}{1 - p(m_i)} \quad (4.3.9)$$

O valor da célula no mapa é representado pela crença de ocupação no estado anterior, somado ao valor do modelo inverso do sensor e subtraída a crença do estado inicial da célula. A posição $l_{t,i}$ no algoritmo pode ser definida pela equação (4.3.10):

$$l_{t,i} = l_{t-1,i} + l(x) - l_0 \quad (4.3.10)$$

Na proposta de Thrun et. al. (2005), o algoritmo elimina as redundâncias e erros nas leituras do sensor. O algoritmo utiliza variações na crença de cada célula e têm início com valor 0; quando uma leitura do laser incide na célula sua credibilidade aumenta e pode atingir o valor máximo 10, as leituras que antecedem a distância máxima da leitura incidente são decrementadas podendo atingir valores mínimos de -10.

Para mapear o ambiente foram realizadas 180 leituras do *laser* com variação angular igual a 1° , que traduz uma perda não significativa de informação e aumento no desempenho do algoritmo para processamento de leituras realizadas pelo *laser*.

Para o mapa deste trabalho foi determinado o tamanho máximo nas variáveis MAX_X, MAX_Y e MAX_Z. Este mapa contém células com dimensão fornecida por TAMANHO_CELULA. Por exemplo, foi definido o TAMANHO_CELULA com 0.1 m ou 10 cm, cada célula no mapa irá conter uma área de 10x10x10 cm. Definido o mapa com MAX_X igual a 500, MAX_Y igual a 500 e MAX_Z 150 – esse mapa cobre uma área de 50x50x15 m conforme apresentado na Figura 22.

Este mapa teve seu tamanho definido em função do experimento proposto no presente trabalho. Para o experimento foi utilizada uma pequena área do mapa definido, o que mostra a possibilidade de aplicação da técnica para ambientes na proporção indicada na Figura 22.

Figura 22. Dimensão da célula e dimensão do mapa

O código gerado realiza conexão com o servidor Player e inicia as leituras adquirindo informações do experimento realizado. Esta informação é processada aumentando as crenças das células nas quais os raios do *laser* incidem e diminuindo os valores de células que antecederam o dos locais de incidência, de acordo com implementação do modelo inverso dos sensores proposto em (THRUN et. al., 2005). As crenças das células do mapa são iniciadas com crença 0 e podem variar de -10 para células com pouca probabilidade de ocupação e 10 para células com alta probabilidade de ocupação. Conforme (BAPTISTA et. al., 2010) a função de atualização do mapa utilizado no algoritmo de mapeamento está descrita na Listagem 2:

2. Listagem: Algoritmo para atualização da crença das células do mapa.

```
1.  Algoritmo atualiza_mapa( mapa, posição_atual, leitura_laser)
2.
3.    para posição_atual
4.        Se crença_da_célula[leitura_posição_atual] < MAX_CRENÇA
5.            aumenta crença_da_célula
6.        Enquanto leitura_posição_atual > posição_atual
7.            leitura_posição_atual – TAMANHO_CELULA
8.            Se leitura_posição_anterior != leitura_posição_atual
9.                diminui crença_da_célula
```

As informações de odometria e do *laser* contam com a precisão de algumas casas decimais em suas leituras, estas casas na implementação do algoritmo são utilizadas nos cálculos e suprimidas na representação do mapa; este fato é apresentado na Figura 23, na qual podem ser visto que várias leituras do *laser* incidem na mesma célula, o que aumenta suas crenças, uma vez que várias leituras correspondem à mesma célula na representação do mapa.

Figura 23. Feixe do laser incidindo na mesma célula, a crença é aumentada.

As diferenças de distância entre as leituras não são significativas, porque uma única célula representa uma variação de 0,1 m, isto é, caso a variação entre as leituras não seja maior que 0,1 m, ela irá corresponder à mesma célula do mapa. A regra é mantida para outros valores de células e mapas em que as variações das células iguais ao seu tamanho.

Após a execução do algoritmo de mapeamento, é gerado um arquivo com os pontos que possuem sua elevada crença de ocupação. O formato do arquivo está definido em pontos x, y, z, e são adicionados em um arquivo .wrl para visualização no VRMLVIEW.

A listagem 3 traz um exemplo do arquivo fonte para visualização.

3. Listagem: Estrutura do arquivo para visualização com VRMLVIEW

1. #VRML V2.0 utf8
 2. Shape{ geometry PointSet { coord Coordinate {point [
 - 3.
-

```
4. 0.0000000 244.0000000 0.0000000
5. 0.0000000 244.0000000 1.0000000
6. ...
7. ...
8. ...
9. ]]]]
```

5.3.1 Configuração do robô com o *laser SICK*

Como o foco do trabalho foi criar uma representação do ambiente em 3-D, foi adotada a configuração do sensor *laser* mostrada na Figura 24. Com esta configuração foi suprida a necessidade de informações necessárias para obtenção do ambiente 3-D. Para obter variação nas posições das leituras, o robô navegou pelo laboratório cobrindo um arco vertical de leituras.

Figura 24. Configuração do sensor laser utilizado no experimento

As informações utilizadas foram: posição do robô fornecida pela odometria composta da tupla (x, y, Θ) e posição dos objetos que circundam o robô no campo de visão do *laser*

fornecendo o eixo z ; a partir de cálculos simples foram adquiridas as informações z e y . O sistema de coordenadas adotado está apresentado na Figura 25.

Figura 25. Sistema de coordenadas utilizado no experimento

O sensor *laser* fornece a informação de uma distância. Conhecido o ângulo em que foi realizada a leitura, formando um sistema polar de coordenadas, é possível converter o ângulo da leitura de graus para radianos; para obter z , basta multiplicar o valor da distância da leitura pelo seno do ângulo; para a componente y basta multiplicar o valor da distância pelo cosseno da leitura.

A seguir essas leituras são processadas de acordo com a posição obtida pela odometria e células pelos raios incidentes nas células.

5.3.2 O funcionamento *hardware* e *software*

Com a estrutura cliente/servidor fornecida pelo *Player*, o experimento foi montado utilizando o robô que possui um computador com sistema Linux; e um *notebook* com a interface cliente do *Player*, ambos conectados via rede sem fio através de um roteador

wireless. Esta configuração é mostrada na Figura 26.

Figura 26. Estrutura cliente/servidor montada com o Player

O notebook foi utilizado para controlar o robô, conduzindo-o de um ponto a outro. A conexão foi realizada via serviço SSH, que fornece acesso remoto ao sistema do robô para inicialização do servidor *Player*; após a inicialização o *notebook* foi conectado, e utilizado o *software* cliente *playerjoy*.

O *software playerjoy* – que é adquirido com o pacote do *software Player*, – fornece uma interface de controle e leituras do estado corrente dos sensores e atuadores, o que possibilita controlar o robô remotamente e, nestas condições, é possível realizar o teste que trabalho propõe.

A configuração de inicialização para gerar os arquivos de log durante o experimento está apresentada na Listagem 4. Nesta configuração foram utilizados : o módulo *p2os* que fornece a interface com o *hardware* de odometria do robô; o módulo *sicklms200* que fornece a interface com o sensor *laser* e o módulo *writelogs* que gera um arquivo com log contendo o tempo e as leituras realizadas pelos sensores (odometria e *laser*).

4. *Listagem: Arquivo de configuração do software Player para utilização com o robô Pioneer durante o experimento.*

```
1. driver
2. (
3.   name "p2os"
4.   provides ["odometry::position2d:0"
5.     "sonar:0"
6.     "aio:0"
7.     "dio:0"
8.     "power:0"
9.   ]
10.  port "/dev/ttyS0"
11. )
12. driver
13. (
14.   name "sicklms200"
15.   provides ["laser:0"]
16.   port "/dev/ttyUSB0"
17.   alwayson 1
18.   delay 35
19.   range_res 10
20.   resolution 50
21. )
22. driver
23. (
24.   name "writelog"
25.   requires ["laser:0" "laser:1" "position2d:0"]
26.   provides ["log:0"]
27.   log_directory "/home/lrm"
```

-
- 28. autorecord 1
 - 29. alwayson 1
 - 30.)
-

A interface entre *software* e *hardware* foi realizada de forma transparente com a utilização do *software Player*. No trabalho não foi utilizado o módulo de localização que realiza a tarefa de correção dos dados fornecidos pela odometria e pelo sensor *laser*. Portanto, o experimento conta com erros de odometria, os quais foram desconsiderados dadas as características do experimento abordadas no item 4.3.

O primeiro experimento realizado neste trabalho, teve por base o modelo de Pfaff et. al., (2007), o modelo do fluxo da informação é apresentado na Figura 27.

Os dados obtidos pela odometria foram adicionados ao modelo e eles forneceram a localização do robô na componente x, que corresponde ao avanço do robô pelo laboratório, e o trajeto por ele realizado está apresentado na Figura 21; os dados y e z foram calculados com os dados fornecidos pelo sensor *laser*, foi construído uma nuvem de pontos 3-D.

Figura 27. Fluxo da informação entre os módulos individuais, baseado em (PFAFF, 2007)

5.3.3 Resultados e comparativos do experimento

Este item apresenta os resultados obtidos durante as simulações dos dados obtidos com

o experimento. Os referidos dados foram extraídos utilizando 3 definições de tamanhos de células, são eles 0,1m para representar estruturas pequenas como mesas e ventiladores, 0,5m para ser possível identificar passagens e portas, e 1m para possibilitar identificação do ambiente. É apresentando a quantidade de pontos armazenados em cada proposta, e também, informações contidas nas imagens.

O primeiro arquivo gerado utiliza dados brutos e contém 540.000 pontos, nos quais estão contidos falhas do sensor e redundâncias nos dados. O tamanho do arquivo fica próximo a 16,3 MB. Na Figura 28 é apresentando uma imagem gerada pelo VRMLVIEW, composta por um elevado número de pontos e nessa imagem estão destacadas as possíveis falhas do sensor.

Figura 28. Possíveis falhas do sensor indicadas em vermelho, numeros de pontos elevados. Gerado com VRMLVIEW

5.3.3.1 Células com tamanho de 0,1 m

Com o tamanho da célula ajustado para 0,1 m, o arquivo gerado possui 23.958 pontos para descrever o ambiente. Com este tamanho de célula as informações importantes são mantidas e há grande redução na quantidade de pontos com relação aos dados brutos. O tamanho do arquivo gerado fica próximo a 229,2 KB. As figuras 29 e 30 apresentam esses resultados.

Figura 29. Imagem do laboratório gerada utilizando tamanho de célula igual a 0,1 m. São mantidos os ventiladores no teto e ângulos de estruturas.

Figura 30. Imagem do laboratório gerada utilizando tamanho de célula igual a 0,1 m. Mantém detalhes importantes do ambiente.

5.3.3.2 Células com tamanho de 0,5 m

Com o tamanho da célula ajustado 0,5 m, o arquivo gerado possui 1.703 pontos para descrever o ambiente. Com este tamanho de célula algumas das informações importantes são perdidas, estas informações são referentes a pequenos objetos como latas, fios e pequenas caixas. No entanto, há possibilidade de saber onde estão as possíveis passagens, os obstáculos e *landmarks*, por exemplo, na Figura 32 é destacado os ventiladores que podem ser utilizados como *landmarks*. O tamanho do arquivo gerado fica próximo a 16,2 KB. As Figuras 31 e 32 mostram os resultados.

É válido lembrar que em mapas de ocupação em *grid*, as formas não apresentam ambiguidade, portanto é possível utilizar as informações contidas nestes resultados em algoritmos que realizam a tarefa de localização para verificar o desempenho.

Figura 31. Imagem do laboratório gerada utilizando tamanho de célula igual a 0,5 m.

Figura 32. Imagem do laboratório gerada utilizando tamanho de célula igual a 0,5 m.

Comparadas às células de 0,1 m, as células de 0,5 m as formas são representadas com maior fidelidade, em ambas as soluções é possível extrair *landmarks* do ambiente, por exemplo; nas células de 0,5m, os ventiladores do teto são representados utilizando poucas células. Outro objeto fixado no teto que contenha uma forma semelhante pode apresentar ambiguidade, enquanto que nas células de 0,1m, os ventiladores são representados com maior fidelidade.

Com células de 0,5 m é possível realizar tarefas de navegação que incluem desvios de obstáculos e encontrar possíveis passagens, isso torna possível um robô se deslocar de forma autônoma pelo ambiente. Também é possível extrair características do ambiente e realizar a tarefa de localização, as características são bordas das paredes, cantos, e linhas que são representadas sem ambiguidade.

5.3.3.3 Células com tamanho de 1 m

Com o tamanho de célula ajustado para 1 m, o arquivo gerado possui 445 pontos para descrever o ambiente. Com este comprimento de célula é perdida muita informação que pode ser valiosa durante navegação e localização. Na figura 33 é possível observar a representação do ambiente, é possível observar que os ventiladores não há possibilidade de identifica-los, outras estruturas contidas no ambiente não possuem suas formas distinguíveis. Com esta representação o robô terá dificuldade para saber qual a distância que esta de um determinado objeto ou estrutura, lembrando que cada célula ocupada pode variar até 1 m. Não é possível reconstruir ou identificar estruturas com espessura inferior ao tamanho definido pela célula; mas para este trabalho foi possível uma abstração do ambiente no qual o robô esta inserido e condições de estruturas grandes como paredes. As figuras 33 e 34 mostram os resultados.

Experimentos de localização podem ser realizados para verificar a eficiência desta representação.

Figura 33. Imagem do laboratório gerada utilizando tamanho de célula igual a 1m.

Pouca informação foi obtida com este tamanho de célula, mas pode ser feita uma reconstrução do ambiente estrutural que envolve o robô.

Figura 34. Imagem do laboratório gerada utilizando tamanho de célula igual a 1m.

Na figura 34, fica claro que perdemos ângulos de inclinação das estruturas e objetos que poderiam ser utilizados como landmark, exemplo, ventiladores, mesas.

5.3.3.4 Extração da representação 2-D do ambiente

Com os dados do ambiente armazenados, foi assumida uma altura com valor igual a 1 metro em relação a altura que o *laser* esta fixado no robô. Para assumir este valor foi suposto que objetos com uma altura inferior a 1 metro, são obstaculos que o robô pode colidir no ambiente. Após executar a simulação, um código adicional que extrai a informação destes objetos é executado criando uma representação plana do ambiente utilizando a biblioteca de programação gráfica *openCV*. A Figura 35 apresenta os resultados obtidos extraindo objetos com altura igual a 1 metro ou inferior, no referido mapa com área de 50m². O trajeto percorrido pelo robô esta representado em vermelho.

É válido ressaltar que objetos com concavidades podem estar adicionados a esta representação. Como exemplo, podem ser citadas mesas, as quais que estão destacadas na cor branca da Figura 36 que apontam a região de interesse do mapa.

Figura 35. Representação do ambiente de realização do experimento.

Cada pixel da imagem corresponde a uma célula do mapa e representa 10 cm do Laboratório onde foi realizado o experimento, a capacidade do mapa é de armazenar uma área de 50mx50m (2500m²). De acordo com o experimento realizado, o robô percorreu aproximadamente 20 m. Para melhor observar a região percorrida, a figura 35 foi ampliada e foi obtido como resultado a figura 36.

Figura 36. Ampliação do ambiente 2-D com mesas destacadas em branco.

Nas regiões apontadas, o robô pode navegar pelas concavidades, mas a representação 2-D, aponta que elas são áreas ocupadas por objetos, impedindo essa navegação.

5.4 Experimento de mapeamento para manipuladores robóticos

Neste capítulo é apresentado no item 4.4.1 uma técnica para extração do plano de fundo e um comparativo entre implementações que estão disponíveis em bibliotecas para desenvolvimento comumente utilizadas na robótica. Com a utilização deste método temos como objetivo obter uma máscara para auxiliar a tarefa de mapeamento.

No item 4.4.2 é apresentado o método iterativo conhecido por *Interactive Closest Point* (ICP). O método ICP é capaz de encontrar a transformação entre duas nuvens de pontos visando diminuir o erro da distância entre os pontos obtidos por sensores de distância ou movimento.

5.4.1 O método probabilístico do modelo de mistura gaussiana (GMM)

Como formas de auxílio na tarefa de reconhecimento e mapeamento de ambientes em 3 dimensões faz-se necessário a utilização de técnicas para processamento de imagem. Estes quadro de imagens podem ser adquiridos através de conjunto de dados pre-existent em repositórios, webcams, câmeras IP, sensores de movimento (no caso específico o sensor Kinect da empresa Microsoft) e câmeras com resolução de alta qualidade (HD). O tamanho dos quadro de imagens variam sua resolução de acordo com o *hardware* utilizado. É considerado quadro de imagem de baixa resolução quando apresenta o número de linhas inferior a 720 pixels, o trabalho apresentado por STAUFFER e GRIMSOM (1999) utilizou quadro de imagens com 160x120 pixels, sendo considerados portanto, quadro de imagens de baixa resolução. Os quadro de imagens com tamanhos superiores 640x480 pixels são tratados como quadro de imagens de alta resolução neste texto.

Com o avanço tecnológico que ocorreu nos últimos 5 anos na área de aquisição e processamento de imagens e os esforços em trabalhos voltados para a área da robótica e mecatrônica, contamos com soluções de hardware de baixo custo e soluções em softwares de alta qualidade que agregam valor a projetos que visam robustez, confiabilidade e segurança.

Este experimento propõe duas soluções viáveis para a extração do cenário de fundo em imagens utilizando a técnica probabilística dos modelos de mistura gaussiana (GMM), esta técnica foi apresentada por STAUFFER e GRIMSOM (1999). Os dados que são eliminados dos quadros não trazem informações relevantes para a aplicação alvo de reconhecimento e mapeamento automático de objetos.

Os computadores com alto poder de processamento possibilitam a criação de sistemas mais robustos para o uso em ambientes comuns do cotidiano, como residências, escritórios, ambientes hospitalares, oficinas, fábricas, etc. Serviços de monitoramento de segurança instalados em prédios e empresas podem identificar alterações no ambiente, distingui-las e por exemplo detectar um acesso de uma área externa para uma área interna. No modelo proposto, o sistema é fixado na área externa e esta sujeito a variações do ambiente, as variações podem

ser decorrentes de luminosidade (luz do dia, sombras, nuvens), efeitos climáticos (vento a balaçar árvores) que causam um falso positivo.

O foco deste trabalho são aplicações de automação que necessitam de monitoramento para auxiliar manipuladores. Para que uma linha de produção seja altamente eficiente, devemos agregar a capacidade de distinguir objetos de diferentes formas e a capacidade de manipula-los de acordo com suas características.

Segmentação em tempo real de regiões que representam a ocorrência de movimento é uma necessidade básica para sistemas que utilizam visão computacional para monitoramento. Para esta tarefa utilizamos métodos adaptativos para extração de fundo. Em RIDDER et. al. (1995) é apresentada uma solução baseada na utilização de um filtro de Kalman. Esta abordagem se apresentou robusta ao tratar mudanças de iluminação, mas o seu processamento é lento e por isso ela se mostra pouco eficiente ao tratar objetos em movimento e sombras. No entanto, é possível tratar objetos em movimentos utilizando filtros de Kalman, uma aplicação desenvolvida por KOLLER et. al. (1994) apresenta um sistema para monitoramento de trânsito, o sistema possui um módulo de baixo nível que realiza o processamento das imagens adquiridas e um módulo de alto nível de descrições para retirar ambiguidades causadas por falhas devido a movimentos e fatores climáticos (chuva, nuvens, congestionamentos), com esta abordagem solucionaram falhas provenientes do sistema de baixo nível.

A abordagem de STAUFFER e GRIMSOM (1999) propõe a solução desta tarefa através da utilização do método probabilístico dos modelos de misturas gaussianas. Podemos citar como fator relevante para utilização desta técnica, a sua eficiência em aplicações que estão sujeitas a alterações de iluminação. Esta técnica apresentou-se robusta ao tratar objetos em movimento, objetos lentos, movimentos repetitivos e situações nas quais os objetos são introduzidos e retirados de uma cena. Os objetos lentos demoram mais tempo para serem incorporados ao fundo, este comportamento pode ser entendido pela variância da cor do objeto ser superior a variância do fundo da imagem.

No processamento dos quadros de imagens é considerado o valor individual de cada pixel no período de tempo. Os valores de cada pixel geram uma série temporal, onde temos um escalar para quadro de imagens em tons de cinza ou um vetor para quadro de imagens coloridas.

De acordo com as classes e métodos implementados na biblioteca OpenCV que seguem o modelo proposto por KAEWTRAKULPONG e BOWDEN (2001), o estudo dos métodos de atualização da Gaussiana possibilitou o desenvolvimento de um novo algoritmo. As vantagens deste novo algoritmo em comparação com a proposta de STAUFFER e GRIMSOM (1999), estão associadas com a redução do tempo de aprendizagem e com a maior precisão das informações extraídas. Este algoritmo apresentou melhores adaptações a mudanças no ambiente.

No código que acompanha a biblioteca OpenCV o autor avisa que na implementação utiliza a técnica de janelamento e não é capaz de reconhecer sombras. Esta limitação se deve a erros nas equações propostas e esta observação está contida no código fonte.

De acordo com as equações apresentadas no trabalho de STAUFFER e GRIMSOM (1999), ao trabalhar com quadro de imagens coloridas onde temos um vetor, a medida que aumentar o tamanho do quadro de imagem, aumenta também o custo computacional.

De acordo com a equação (4.4.1.1):

$$\{X_1, \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \quad (4.4.1.1)$$

A informação armazenada de cada pixel processado (x_0, y_0) é seu histórico e I representa um quadro de imagem. O valor de cada pixel representa uma medida da radiação na direção do sensor do primeiro objeto intersectado pelo raio ótico do pixel. Assumindo um plano de fundo estático e uma iluminação estática, o valor tende a ser constante. Quando algum ruído é introduzido no processo de aquisição da informação, é possível descrever por uma distribuição Gaussiana centrada no valor médio do pixel.

Aplicações do cotidiano humano, estão inseridas em ambientes com alto grau de dificuldade para abstração, onde o sistema está sujeito a variações de iluminação e variações inesperadas no cenário. Quando um novo objeto é introduzido no cenário, pode-se utilizar a Gaussiana para acompanhar (tracking) variação correspondente da imagem.

O modelo GMM é capaz de lidar com estas variações devido ao modelo de atualização. O histórico recente de cada pixel $X_1, \dots, X_t$ é modelado pela mistura de K distribuições gaussianas. A probabilidade de observar o pixel corrente é dada pela equação

$$P(X_t) = \sum_{i=1}^K w_{(i,t)} * \eta(X_t, u_{(i,t)}, \Sigma_{i,t}) \quad (4.4.1.2)$$

Nesta equação o componente:

K representa o número de distribuições,

$w_{i,t}$ é o peso estimado da i -ésima mistura Gaussiana no tempo t .

$u_{i,t}$ é a média dos valores da i -ésima mistura Gaussiana no tempo t e

$\Sigma_{i,t}$ é a matriz de covariância da i -ésima mistura Gaussiana no tempo t .

A distribuição normal η é obtida pela equação (4.4.1.3):

$$\eta \left(X_t, u, \Sigma \right) = \frac{1}{(2\pi)^{\frac{n}{2}} \left| \Sigma \right|^{\frac{1}{2}}} e^{-\frac{1}{2} (x_t - u_t)^T \Sigma^{-1} (x_t - u_t)} \quad (4.4.1.3)$$

A constante K é definida de acordo com a memória e o poder computacional. STAUFFER e GRIMSOM (1999) recomenda valores entre 3 e 5. O código implementado na biblioteca OpenCV utiliza o valor 5. Os trabalhos apresentados por ZIVKOVIC (2004), ZIVKOVIC e HEIJDEN (2006) e PHAM et. al., (2010) utilizam o valor 4 para o número de gaussianas.

Outras propostas foram apresentadas, alguns modelos utilizam lógica fuzzy para otimizar a eficiência no reconhecimento de sombras e contornar falhas devido a intensidade de iluminação, estas soluções podem ser encontradas em BAF et. al. (2008).

As propostas apresentadas trabalham com quadro de imagens em baixa resolução, exceto PHAM et. al., (2010) que utiliza quadro de imagens em alta qualidade. Com os avanços tecnológicos que ocorreram em dispositivos de aquisição dos quadro de imagens (cameras HD), foi possível adquirir quadro de imagens em alta qualidade. Ao trabalhar com

quadro de imagens de alta qualidade, as propostas de GMM podem apresentar perdas significantes para obter a informação desejada.

Quadro de imagens adquiridos por um sensor de movimento (kinect) são adquiridos com dimensões de até 640x480. Conforme apresentado em PHAM et. al., (2010), ao aumentar o quadro de imagem o algoritmo de ZIVKOVIC e HEIJDEN (2006) apresenta perdas no processamento e se torna lento.

De acordo com os resultados apresentados por CARR (2008), foi obtido um ganho em tempo de processamento de 5x para quadro de imagens com dimensão de 352x288 e apresentou uma taxa de 58 quadros por segundo, nesta proposta foi utilizado os recursos de hardware fornecido pela GPU. A proposta de PHAM et. al., (2010) mostra uma abordagem que utiliza o alto processamento oferecido por placas aceleradoras gráficas equipadas com arquitetura CUDA para obter ganhos no tempo de processamento dos quadro de imagens.

Comparado com a proposta de ZIVKOVIC e HEIJDEN (2006) o algoritmo obteve um ganho em tempo de processamento de 13x para quadro de imagens com dimensão 400x300 com uma taxa de 980 quadros por segundo. As propostas que utilizam a CPU e utilizam quadro de imagens em alta resolução, apresentaram um desempenho inferior a 30 quadros por segundo, enquanto que soluções que utilizam a GPU sempre obtiveram resultados acima de 30 quadros.

5.4.1.1 Realização do experimento

Foram utilizados como conjunto de dados alternativo arquivos de animações computadorizadas e conjunto de dados fornecido no site do projeto MRPT, que foram adquiridos através do sensor de movimento Kinect e as imagens estão no formato RGBD.

O arquivo de conjunto de dados utilizado neste trabalho estão disponíveis para download em MRPT (2013), o nome do arquivo é *rawlog_rgb_d_dataset_freiburg2_coke.tgz*.

As propostas estudadas utilizam uma camera fixa, apenas objetos que cruzam o campo de observação são detectados. Os arquivos de conjunto de dados que utilizamos, o equipamento é movido pelo ambiente o que dificulta a tarefa de extração de fundo. O uso do

método GMM favorece a solução deste problema por incorporar objetos fixos ao modelo e rastrear objetos móveis através da variação na gaussiana.

Os experimentos para manipuladores robóticos foram realizados nos computadores A, B, C. Suas configurações de hardware estão detalhadas no sub-item 4.2.3.

5.4.1.2 Resultados e comparativos do experimento

Os fatores relevantes observados neste trabalho foram o tempo de processamento para obter a extração e a qualidade dos quadros obtidos. O tempo de processamento que calculamos é referente ao cálculo da gaussiana, iniciado antes de executar o cálculo e encerrado logo após o término do cálculo. A qualidade dos quadros é um comparativo de resultados obtidos entre os quadros de imagens da proposta de PHAM et. al., (2010) e a proposta implementada no núcleo da biblioteca OpenCV que segue a proposta de ZIVKOVIC e HEIJDEN (2006).

O computador A não possui a arquitetura CUDA, portanto não foi possível executar o algoritmo proposto. O projeto do framework Ocelot oferece uma infraestrutura para compilar e executar códigos CUDA em computadores que não possuem GPU CUDA, no entanto para nossa aplicação o desempenho foi reduzido drasticamente.

A tabela 1 apresenta o tempo necessário por cada um dos computadores para realizar a tarefa do cálculo da GMM. A OpenCV de acordo com a arquitetura de hardware utilizada tem um ganho altíssimo em seu desempenho. Ao executar o código no computador A, foi consumido o tempo de 61 ms para realizar o processamento com uma taxa de 6,25 quadros processados por segundo, estes valores tornam a solução inviável para aplicação em nosso projeto. Quando executado no computador C, foi consumido o tempo de 13 ms para realizar o processamento com uma taxa de 22,94 quadros processados por segundo, estes valores viabilizam a proposta para atender nossas necessidades.

A primeira análise de nosso trabalho é a quantidade de quadro de imagens processado por segundo (fps) por cada um dos códigos, e ambos executados juntos sequencialmente. Os computadores apresentaram os resultados conforme tabela 1 para o conjunto de dados coke:

As configurações de interesse para o experimento são repetidas abaixo nos itens A, B e C.

A - Notebook / Core 2 Duo, 4 Gb RAM, SO 32 bits.

B - Computador de mesa / Core i3, 4Gb RAM, Geforce 1Gb RAM SO 64 bits.

C - Notebook / Core i5 2.5Ghz, 4 Gb RAM, Geforce 2Gb RAM SO 64 bits.

Tabela 1 – Tempo de processamento necessário para processar cada quadro nos computadores A, B, e C; e quantidade de quadros processados por segundo (fps). Quadros processados pela OpenCV, CUDA e OpenCV+CUDA.

	A	B	C
OpenCV	61ms / 6,25 fps	22ms / 19,99 fps	13ms / 22,94 fps
CUDA	não executa	11ms / 25,65 fps	5ms / 28,44 fps
OpenCV + CUDA	não executa	17ms 14ms / 13,37 fps	15ms 5ms / 18,96 fps

A proposta que utiliza arquitetura CUDA obteve valores muito significativos ao realizar a tarefa de extração de fundo, estes valores são referentes ao tempo de processamento necessário e estão apresentados na Tabela 1, qualidade do quadro de imagem são comparados nas figuras de número 37 até a figura de número 48 e o tempo de resposta analisado nas figuras de número 37 até a figura de número 42 em casos de reinicilização. Ao executar no computador C foi obtido o resultado em 5ms com uma taxa de 28,44 quadros processados por segundo, este valores também viabilizam a proposta deste projeto. Ao mudar a arquitetura do hardware para o computador B, o tempo necessário para realização teve um aumento significativo chegando próximo aos valores obtidos no computador C utilizando a OpenCV.

Figura 37: Imagem 1 adquirida pelo sensor de movimento Kinect

Com os valores da tabela 1, é difícil dizer qual solução adotar tendo em vista que o hardware envolvido pode interferir no resultado aumentando o tempo necessário para obter a resposta e até mesmo não executar o programa. Para melhor análise outros fatores foram levados em consideração.

Figura 38: Imagem 1 processada pelo algoritmo da biblioteca openCV

Na segunda análise, apresentamos o quadro de imagem obtido pela primeira leitura realizada, o quadro de imagem resultado que utiliza OpenCV e o quadro de imagem resultado que utiliza tecnologia CUDA, ambas realizadas no computador C. A imagem adquirida pelo sensor kinect pode ser observada na Figura 37 e os resultados obtidos pela OpenCV e CUDA podem ser observadas nas Figuras 38 e 39 respectivamente.

Analisando o resultado obtido pela OpenCV apresentado na Figura 38 e pela CUDA apresentado na Figura 39, foi observado que ao iniciar o algoritmo a solução que utiliza OpenCV retorna as regiões de cor preta que representam o segundo plano (fundo) e as regiões de cor branca que representam o primeiro plano (frente). A solução que utiliza CUDA retorna uma imagem próxima a distribuição normal no primeiro quadro de imagem processado, as regiões de cor cinza representam as sombras e as regiões de cor brancas representam o primeiro plano.

Figura 39: Imagem 1 processada pelo algoritmo que utiliza CUDA

Na Figura 40 é apresentado o quadro de imagem obtido pela segunda leitura realizada.

Figura 40: Imagem 2 adquirida pelo sensor de movimento Kinect

Figura 41: Imagem 2 processada pelo algoritmo da biblioteca openCV

Figura 42: Imagem 2 processada pelo algoritmo que utiliza CUDA

Na Figura 41 que representa segunda imagem processada podemos observar que o resultado apresentado pela OpenCV apresenta regiões na cor branca com maior nível de detalhamento.

O resultado apresentado na Figura 42 pela CUDA em comparação com o resultado apresentado na Figura 41 pela OpenCV, observa-se um nível maior de detalhamento do ambiente, sua adaptação ao movimento realizado no kinect incorporou regiões de sombras que podem ser interpretadas como segundo plano em alguns casos, e as formas foram preservadas.

É válido ressaltar que esta técnica é utilizada para auxiliar o reconhecimento de objetos e landmarks. Outros sistemas que podem empregar esta técnica são sistemas de localização e mapeamento com a finalidade de compor um sistema autônomo com alto grau de eficiência. A citar carros autônomos utilizam esta técnica para realizar o reconhecimento da pista.

Figura 43: Imagem 100 adquirida pelo sensor de movimento Kinect

Nas propostas estudadas as imagens foram adquiridas por cameras em posições fixas.

Em um ambiente totalmente estático e com iluminação homogênea o sistema pode retornar uma imagem preenchida por pixels na cor preta. Quando introduzido um objeto no campo de visão a gaussiana do sistema é alterada gerando pontos brancos na imagem onde o objeto está localizado na cena. Caso o objeto seja fixado ao ambiente ele é incorporado no modelo. Os casos estudados neste trabalho, a camera encontra-se em movimentação pelo ambiente, com a utilização do conjunto de dados adquirido em MRPT (2013) e abordado no sub-item 4.4.1.1, ambos os sistemas apresentaram resultados satisfatórios para realizar a tarefa de reconhecimento, no entanto, quando os algoritmos são utilizados em conjuntos de dados em que a movimentação da câmera desloca-se com movimentos bruscos, as soluções apresentaram resultados insatisfatórios e de difícil compreensão. De acordo com a Figura 43 que foi obtida a partir da leitura de número 100, é possível verificar na Figura 44 que o algoritmo OpenCV gera informações com ruídos sobre o primeiro plano (pontos na cor branca) devido as leituras anteriores e apresenta o fundo bem definido (pontos na cor preta). A Figura 45 gerada pelo algoritmo CUDA tem seu detalhamento enriquecido pelas regiões que representam sombras (pontos na cor cinza), é possível identificar a lata, ambos algoritmos não definem a mesa de forma coerente.

Nestas condições destacamos o algoritmo CUDA, a leitura de número cento e vinte apresentado na Figura 46, é possível observar na Figura 47 que o algoritmo OpenCV é possível identificar parte da lata, mas ainda encontra-se indefinido e contém ruídos, enquanto que na Figura 48, o algoritmo CUDA é capaz de identificar o objeto lata de forma coerente e o objeto mesa que podemos considerar como plano de fundo, foi incorporado ao modelo e considerado como objeto imóvel.

Figura 44: Imagem 100 processada pelo algoritmo da biblioteca openCV

Figura 45: Imagem 100 processada pelo algoritmo que utiliza CUDA

Figura 46: Imagem 120 adquirida pelo sensor de movimento Kinect

Figura 47: Imagem 120 processada pelo algoritmo da biblioteca openCV

Figura 48: Imagem 120 processada pelo algoritmo que utiliza CUDA

Estes fatores contam como vantagens para solução que utiliza CUDA. O tempo de processamento individual de cada quadro de imagem também favorece a esta solução e pode ser obtido em até metade do tempo apresentado pela OpenCV. Estas informações foram levadas nos computadores B e C descritos no sub-item 4.2.3.

De acordo com os testes realizados, foi observado que ao final do experimento quando utilizamos a solução CUDA o sistema apresenta lentidão, causando um aumento no tempo de processamento e traz perdas significativas na taxa de quadros por segundo (fps). Para identificar a causa foi utilizado um utilitário de monitoramento de sistema, de acordo com a solução que acompanha as distribuição OpenSuse, o algoritmo CUDA apresenta um consumo de memória linear de acordo com o tempo de execução e quantidade de quadro de imagens processados.

Para sistemas que realizam o monitoramento por um periodo de tempo indefinido, que pode atingir dias ou meses, isso é uma falha grave. O sistema pode se deparar com uma falta de memória para realizar o processamento. No algoritmo OpenCV, a memória necessária para realizar o processamento é fixa, a quantidade de memória necessária não varia na proporção

que aumentamos a quantidade de quadro de imagens processados.

5.4.1.3 Aplicação e discussão sobre o método GMM

O algoritmo CUDA apresentou resultados satisfatórios para realizar a tarefa de extração de fundo, isso nos possibilita realizar o reconhecimento de objetos em ambientes que manipuladores robóticos atuam. Também foi satisfatório o resultado do tempo necessário para realizar o processamento de um quadro de imagem individual, no entanto, o consumo de memória pode trazer falhas graves ao sistema. Uma sugestão deste trabalho é realizar estudos aprofundados na tecnologia CUDA visando otimizar o consumo e utilização de memória, desta forma o algoritmo pode ser utilizado em um sistema robusto. De acordo com o método GMM, os quadros de imagens já processados não são necessários para realizar o processamento do quadro de imagem da leitura atual do sensor. O algoritmo OpenCV apresentou resultados surpreendentes ao realizar o processamento de quadro de alta qualidade (640x480), ao contrário dos artigos estudados, ele desempenhou a tarefa de extração de fundo com uma taxa de quadro por segundo (fps) acima do esperado e atende nossas especificações. No entanto as falhas que ele apresentou podem ser significativas na tarefa de reconhecimento de objetos. O núcleo da OpenCV faz utilização da GPU padrão, caso exista. A sugestão deste trabalho de acordo com os resultados apresentados pelo algoritmo CUDA é implementar esta solução no núcleo da OpenCV, as vantagens citadas neste artigo, podem trazer ganhos expressivos a biblioteca. No código do algoritmo contido no núcleo da OpenCV há uma mensagem que “a técnica de janelamento é utilizada, mas não detecta sombras, Eu fiz minhas próprias modificações que fazem mais sentido. Contém alguns erros em algumas equações”.

Com o experimento que utiliza o método GMM visamos extrair uma máscara para identificar objetos que estão no campo de atuação do manipulador robótico.

No processo de reconhecimento e mapeamento de objetos faz-se necessário a aplicação de técnicas auxiliares para aumentar a crença sobre a informação ruidosas do sensor. Esta máscara será aplicada sobre uma nuvem de pontos obtida através do sensor kinect e visa monitorar apenas a área do campo de atuação do manipulador robótico e determinar a forma e

5.4.2 O método iterativo de ponto mais próximo (ICP)

Ao habilitar tarefas complexas para manipuladores robóticos, há necessidade de extrair informações do ambiente que inserimos o manipulador. Estas informações são obtidas através de sensores *lasers*, cameras e cameras com sensores de movimento (Kinect). Os sensores disponíveis possuem falhas que podem comprometer a precisão da representação obtida para o manipulador robótico e causar falhas em tarefas de planejamento de rota e navegação. Para que estas tarefas sejam realizadas com êxito, faz-se necessário uma representação fiel do objeto extraído pelos sensores..

O método iterativo de ponto mais próximo, definido como ICP, foi apresentado por BESL e MCKAY (1992), e tem como objetivo encontrar a transformação entre uma nuvem de pontos e uma superfície de referência (ou outra nuvem de pontos), minimizando os erros quadrado entre pontos correspondentes.

Neste trabalho apresentamos uma aplicação do método ICP para representar objetos no campo de atuação do manipulador robótico. A aquisição de dados foi realizada com a utilização sensores de movimento (kinect).

O método ICP foi descrito como uma forma importante para registrar dados de sensoramento de objetos rígidos não fixos com um modelo geométrico ideal antes de moldar a inspeção. A solução apresentada é capaz de manipular seis graus de liberdade ($x, y, z, \alpha, \beta, \gamma$), fornece soluções para conjuntos de pontos que não possuem correspondência e para o problema de correspondência em formas livres curvas.

O algoritmo proposto por BESL e MCKAY (1992) suporta variadas formas de representação de dados geométricos, a citar: Conjuntos de pontos, conjuntos de segmento de linha, curvas implícitas, curvas paramétricas, conjuntos de triângulos, superfícies implícitas e superfícies paramétricas.

A base do funcionamento do método ICP é registrar 2 conjuntos de dados em um sistema de coordenadas comum. A proposta visa minimizar o erro e calcular a transformação

de rotação e translação (R, t) para minimizar a equação (1).

$$E(R, t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \|m_i - (Rd_j + t)\|^2 \quad (1)$$

Na equação (1), N_m e N_d são o número de pontos do modelo m e do conjunto de pontos d respectivamente, $w_{i,j}$ são os pesos para os pontos correspondentes. O peso de $w_{i,j}$ é ajustado para 1 se m_i é um ponto próximo de d_j caso contrário o valor de $w_{i,j}$ é ajustado para 0. Para cada iteração é realizado o cálculo da transformação.

O método ICP proposto por BESL e MCKAY (1992) é dividido em 7 etapas, são elas:

- 1 - Pré-processamento: nesta etapa são retirados dados redundantes.
- 2- Correspondência: nesta etapa os pontos do mapa são associados com os dados obtidos pelo sensor. É realizada utilizando a busca por vizinho mais próximo.
- 3- Ajuste de peso (ponderação): nesta etapa são ajustados a importância de alguns pares.
- 4- Descarte: nesta etapa alguns pares são descartados.
- 5- Ajuste do erro: nesta etapa são ajustado o erro de cada par.
- 6- Minimização: nesta etapa é encontrada a melhor transformação que satisfaça o sistema.

As sete etapas são repetidas enquanto não há convergência, e o número de iterações (repetições) pode ser definido o valor máximo de iterações a serem realizadas no processamento. O método ICP é capaz de encontrar para cada ponto sua referência nos dados obtidos pelo sensor. Este método desempenha esta tarefa baseado na localização do sensor. O funcionamento do algoritmo é exemplificado nas figuras 49, 50 e 51. Na figura 49, é observado o cenário contendo um objeto, as leituras serão realizadas de dois pontos distintos. Na figura 50 é apresentada a leitura realizada em cada ponto que o sensor esta fixado.

Figura 49: Leituras realizadas de pontos diferentes.

Figura 50: Nuvem de pontos adquiridas pelo sensores posicionados de acordo com a figura 49.

Figura 51: Nuvem de pontos após ser processado pelo algoritmo de ICP.

Na figura 51, é apresentando as nuvens de pontos após o processamento pelo algoritmo de ICP, o alinhamento é realizado de forma a minimizar a distância entre os pontos e retorna a translação e rotação aplicada.

Ao observar a importância do método, aplicações na área de robótica surgiram criando variações do modelo para tratar a remoção de pontos estatisticamente desprezíveis e casos de desigualdade da incerteza entre os pontos.

Na proposta apresentada por LU e MILIOS (1994), o método ICP foi utilizado para estimar a localização do robô em um ambiente desconhecido, a auto-localização faz-se necessária para desempenhar a tarefa de execução de rotas previamente planejadas. Para solucionar o problema de auto-localização, é encontrada a correspondência dos dados adquiridos através de sensores contra os dados do mapa, desta forma é possível estimar a localização do sensor. O método ICP mostra-se eficiente como auxílio a odometria que apresenta erros acumulativos de posicionamento. Esta aplicação utiliza uma representação do mapa 2-D, seus dados são adquiridos por sensores lasers 2-D. Para reconstrução de modelos 3-D o custo computacional necessário para encontrar correspondência é elevado ao usar

nuvem de pontos.

De acordo com RUSINKIEWICZ e LEVOY (2001) as variações propostas alteram uma das etapas do método e podem causar diminuição na velocidade de processamento o que compromete aplicações que necessitam ser desempenhadas em tempo real. No mesmo trabalho ele apresenta uma implementação que executa as etapas com ganhos significativos para aplicações de tempo real. Outras variações foram apresentadas, o modelo proposto por CENSI (2008), foi realizado uma alteração na etapa de busca por correspondência, o cálculo da distância é realizado do ponto a linha (point-to-line), e aumenta a eficiência do algoritmo, no entanto, não é robusto quando submetido a grandes erros iniciais.

Em SEGAL et. al. (2009) é proposto uma generalização do método ICP junto a uma única estrutura probabilística, o cálculo da distância é realizado do ponto ao plano. Este cálculo traz como vantagem a informação sobre a normal. O trabalho teve como objetivo aumentar a fidelidade da representação obtida após o processamento. Comparado com o método ICP padrão e o método ponto ao plano (point-to-plane), a média de erro é inferior. O conceito utilizado reduz para duas etapas chaves para solução do problema, são elas: calcular as correspondências entre o modelo e a leitura obtida pelo sensor; e computar uma transformação que minimiza a distância entre os pontos correspondentes.

Nas implementações encontradas usa-se a definição da distância máxima dos pontos próximos para manipular os conjuntos de pontos que estão em uma mesma área, o ajuste deste valor limita a área de atuação e facilita o descarte de pontos que excedam o limite definido. Este limite é definido como *threshold*.

Com o embasamento teórico sobre o método ICP foi realizado nosso experimento com manipuladores robóticos e descrito no item 4.4.2.1

5.4.2.1 Realização do experimento

Para o experimento foi utilizado *logs* de conjunto de dados adquiridos no projeto livre MRPT. O processamento foi realizado com o método ICP implementado na biblioteca MRPT.

O conjunto de dados utilizado neste trabalho estão disponíveis para download em

(MRPT, 2013). Na aplicação realizada neste trabalho foi utilizado a representação por conjunto de pontos, conhecido também por nuvem de pontos.

Os dados contidos são dados coletados por um experimento que contém uma lata de refrigerante posicionada em cima de uma mesa. Uma pessoa movimenta-se com o sensor kinect ao redor da mesa e foca na lata de refrigerante. Estes dados foram úteis para o experimento porque existem desafios de localização, grande volume de dados adquiridos por sensores e processamento limitado.

Neste experimento não foi utilizado um sistema para realizar a autolocalização do robô manipulador ou robô móvel o que traz alguns erros no processo de reconstrução. O grande volume de dados foram processados sem utilizar métodos de filtragem o que traz perdas significativas no tempo necessário para o processamento.

5.4.2.2 Resultados e comparativos do experimento

O algoritmo criado neste trabalho inicia a aquisição e processamento dos dados com aproximadamente 55000 pontos relativos ao espaço onde o sensor esta inserido e de acordo com o conjunto de dados utilizado, este número pode variar conforme o ambiente monitorado. Aplicações que utilizam métodos ICP para encontrar correspondências entre as leituras adquiridas pelos sensores normalmente utilizam sensores Laser SICK ou equivalentes, estes sensores adquirem no máximo 360 leituras correspondentes ao plano (2-D), o custo computacional da tarefa iterativa não é alto. Comparado com o volume da dados utilizado neste trabalho a diferença entre os sensores chega ser superior 100 vezes.

Nos primeiros experimentos verificamos a necessidade de um ajuste no valor máximo de configuração do *threshold*, este valor refere-se a maior distância entre 2 pontos contidos em nuvens de pontos distintas. O valor utilizado foi 1 metro devido a movimentação realizada com o sensor pelo ambiente. A figura 52 apresenta o resultado, foi insatisfatório e incompreensível além de demandar muito tempo para processamento.

Figura 52: Threshold ajustado para 1 m. Reconstrução insatisfatória.

Ajustando o valor de threshold para 10 cm e 1 cm foi possível obter resultados mais expressivos e possibilitou a compreensão da utilização do método e suas limitações, vantagens e desvantagens. O resultado é apresentado na figura 53.

O algoritmo deste trabalho não foi capaz de realizar a tarefa de encontrar as correspondências em tempo real. Conforme resultado obtido não foi necessário analisar o tempo pois o sistema é visivelmente lento. Uma razão para este fato foi a quantidade elevada de pontos.

No decorrer da execução do algoritmo foi notado um aumento no número de pontos mapeados e em alguns casos redundantes. Esta falha é gerada pela ausência de um método para estimar a localização do sensor no ambiente.

Figura 53: Threshold ajustado para 10 cm. Possibilidade de identificar a lata e falhas devido a falta de odometria

Em um momento aleatório da execução foi extraída a figura 54. É possível observar a lata e redundâncias, estas falhas são devido a ausência de sistemas de localização. O algoritmo ICP realiza o processamento, mas não consegue minimizar o suficiente pela incerteza de sua localização e demandaria mais tempo para convergir e em alguns casos não converge.

Figura 54: Reconstrução do ambiente 3-D, ainda apresenta falhas devido a ausência de um sistema de localização.

Na figura 55 é possível observar o fundo do ambiente com uma enorme concentração de pontos desnecessários para o foco do experimento que é a lata.

Figura 55: Nuvem de pontos do ambiente captado pelo sensor Kinect. Elevado número de pontos desnecessários.

Ao observar o resultado do método ICP, a região onde a lata de refrigerante esta inserida, corresponde a uma proporção pequena relativa ao total da nuvem. Ao desconsiderar os pontos relativos a mesa, é possível reduzir este número para valores que viabilizam aplicações de manipuladores robóticos que necessitam de uma representação do ambiente.

5.4.2.3 Discussão sobre o método ICP

O método ICP é um método robusto para encontrar correspondências entre duas nuvens de pontos, no entanto, a ausência de um estimador de localização, o método pode inserir medições falhas ao modelo, aumentando drasticamente o número de pontos contidos no mapa, conseqüentemente uma nova iteração demandará maior custo computacional.

De acordo com BESL e MCKAY (1992) e Nüchter et. al. (2007), o método ICP

também pode ser utilizado como método de auxílio a estimadores de localização. As atenções estão voltadas para esta capacidade do método ICP, o algoritmo tem sido utilizado para localizar a posicionamento de objetos em relação ao ambiente, assim como auxiliar a localização do robô em relação ao ambiente. Desta forma faz-se necessários estudos para utilizar este potencial do método para auxiliar na localização, ao utilizar o método para encontrar as correspondências é obtido como resposta a transformação e rotação relativa entre as nuvens de pontos.

É válido ressaltar que a proposta de Nüchter et. al. (2007) utiliza o robô KURT equipado com sensores de distância a *laser* e o processo para encontrar correspondências não foi executado em tempo-real, os resultados são mapas com alto grau de fidelidade ao ambiente no qual o robô é inserido e tem contribuído com soluções de mapeamento.

6. Conclusões e trabalhos futuros

Os experimentos realizados proporcionaram o estudo dos assuntos relacionados à mapeamento robótico. Nas últimas duas décadas este tema foi muito estudado, houveram muitos avanços e viabilizaram a construção de robôs com alto grau de autonomia. Este trabalho contribui com robôs móveis, robôs manipuladores e robôs manipuladores móveis. Este robôs necessitam de mapas com alto grau de precisão e envolvem a representação de estruturas verticais. Este grau de liberdade é alcançado ao utilizar mapas métricos em 3-D.

As representações de mapas métricos utilizam células que representam regiões do ambiente real. Estas regiões podem estar vazias, habilitando a navegação do robô, ou ocupadas que obrigam refazer o planejamento de rota. A representação de mapas 3-D trabalham com elevadas quantidades de dados e faz-se necessário métodos para filtrar estes dados. Para realizar a esta filtragem são utilizados métodos probabilísticos. Ao desempenhar a tarefa de mapeamento, sabendo a localização do robô é fácil dizer o que está sendo observado, mesmo que medições dos sensores possuam falhas, com a utilização de métodos probabilísticos é possível minimizar estes erros. Com a ausência da localização do robô a tarefa de mapeamento é dificultada, se não é possível saber onde estamos e não é possível saber onde estamos observado, como é possível realizar a tarefa de mapeamento? Neste cenário, ao obter 2 leituras de uma mesma cena mas de pontos distintos, os dados não estão alinhados, o sistema de odometria possui falhas acumulativas e o sistema de mapeamento irá gerar mapas incompreensíveis. Faz-se necessário um módulo de autolocalização do robô em relação ao ambiente inserido.

Foi realizado um experimento voltado a robótica móvel, foi instrumentado um robô que torna possível o mapeamento em 3-D; o algoritmo de mapeamento gerou e trabalhou com um conjunto de dados bruto (*rawdata*) contendo 540.000 pontos referentes as coordenadas x,y,z,; O processamento foi realizado pelo nosso algoritmo que tem base em técnicas probabilísticas e no conceito teórico do trabalho e foi possível gerar a representação do ambiente em 3-D. A representação do ambiente gerada, obteve uma redução para 23.958 pontos que representa 4,43% dos pontos iniciais. Com esta quantidade de pontos foi possível

criar uma representação do ambiente.

De acordo com a pesquisa realizada as técnicas probabilísticas agregam valores na eliminação de dados redundantes, o módulo proposto neste trabalho necessita operar com volumes de dados reduzidos, mas suficientes para manter o robô em operação a realizar suas tarefas. A representação obtida torna possível robôs móveis planejar e navegar no ambiente proposto. Permite estudos que envolvam manipuladores robóticos móveis e experimentos que envolvam robôs móveis aéreos.

Os estudos realizados com o método GMM, mostra que a técnica é muito útil no auxílio da tarefa de mapeamento e reconhecimento de ambientes, objetos e pessoas. O resultado obtido com o método GMM pode ser utilizado como máscara para identificar informações de regiões específicas onde robôs podem navegar, por exemplo, faixas de rodovias, outra aplicação comum que utiliza este método são sistemas de monitoramento de tráfego em rodovias. Este método agrega valor ao módulo de mapeamento quando utilizado como técnica de auxílio ao mapeamento.

Nas décadas passadas, este método foi estudado e trabalhava com recursos limitados, os quadros de imagens não possuíam dimensões muito alta devido a limitações das cameras existentes. Ocorreram avanços tecnológicos nas cameras, os quadros de imagens aumentaram as dimensões e surgiram limitações no processamento. Em nosso trabalho estudamos as soluções implementadas na OpenCV e a versão que utiliza a tecnologia CUDA.

As imagens foram adquiridas pelo sensor de movimento Kinect e os quadros de imagens possuem tamanhos de 640x480 pixels. Em processadores Core Duo, o desempenho foi insuficiente e sem o *hardware* com arquitetura CUDA, ele não realiza a tarefa de extração de fundo. Ao migrar para processadores Core i3 e Core i5, os algoritmos realizaram a tarefa de segmentação com desempenho satisfatório para proposta deste trabalho. A solução encontrada na biblioteca OpenCV não implementa solução para processar mudanças na iluminação e regiões de sombras enquanto que a solução que utiliza arquitetura CUDA processa mudanças na iluminação e regiões de sombras e seu desempenho é superior.

A solução que utiliza arquitetura CUDA apresentou um aumento excessivo no consumo de memória, se o sistema for exposto a longos tempos de operação, ele pode parar por excasez de memória ou realizar o processamento com desempenho insatisfatório. Por

outro lado, a solução que implementada pela OpenCV suporta o sistema submetido a longos tempos de operação e não tem perdas no desempenho devido a consumo de memória. As soluções apresentaram uma taxa média de 21 quadros por segundo (fps) para solução que utilizam OpenCV e 27 quadros por segundo (fps) para solução que utiliza a arquitetura CUDA.

Foi realizado o teste de ambas soluções operando sequencialmente, neste caso o algoritmo estava submetido a consumo de memória excessivo e ao processamento inferior da solução que utiliza OpenCV. O algoritmo apresentou perdas significativas e sua taxa média de quadros foi 16 (fps).

O estudo é focado na tarefa de mapeamento, conforme verificado faz-se necessárias outras técnicas complementares para que a tarefa de mapeamento seja realizado de forma precisa e útil em aplicações do cotidiano. Com uma representação 2-D, partindo de um ponto A, o robô terrestre consegue deslocar-se por ruas e avenidas, e atingir um ponto B. Robôs aéreos podem realizar esta tarefa de forma otimizada se cruzar por entre as estruturas dos prédios e pontes. Os manipuladores robóticos estão expostos a situações que a representação do objeto deve ser fiel ao objeto real, desta forma tarefas como de *grasping* podem ser realizadas com alto grau de precisão e eficiência. Os avanços tecnológico nos proporcionam sensores com maior resolução e precisão, mas ainda podem acumular falhas e limitações. Neste contexto, o nosso trabalho estudou o método de ICP, a proposta deste método atende as necessidades deste trabalho.

Neste trabalho, foi verificado com o experimento a eficiência do método ICP ao trabalhar com representação em 3-D. O método de ICP possui ajustes que otimizam o desempenho e fazem o resultado convergir. Na proposta deste trabalho, ajustamos o valor do *threshold* que faz referência a distância máxima entre pontos de nuvens distintas, se este valor não estiver ajustado corretamente pode fazer o método não convergir ao resultado.

Após ajustado o valor do *threshold*, foi verificado que o método é ineficiente ao trabalhar com nuvens de pontos devido ao volume de dados contido. Nosso experimento tem como objetivo obter uma representação da lata de refrigerante. Ao realizar as leituras com o Kinect, é captado o objeto de interesse e o plano de fundo, os dados relativos ao plano de fundo representam aproximadamente 80% do volume de dados.

A representação do objeto obtida conta com falhas de posicionamento, nosso experimento não utilizou nenhum módulo para localização, isso dificulta a tarefa de mapeamento e compromete o desempenho do método de ICP.

Diante dos resultados apresentados, novos experimentos devem ser realizados. Os manipuladores robóticos operam em ambientes controlados ou quase controlados. Diante desta característica podemos implementar um sistema que definimos nossa região de operação com a utilização do método GMM, assim é possível eliminar os pontos desnecessários para representação. Com o volume de dados reduzido aplicamos o método de ICP e verificamos possíveis ganhos em seu desempenho. Este experimento visa obter uma representação fiel do objeto.

Este trabalho contribui com a proposta de de Pedro (2013), no sistema proposto é utilizado algoritmos para extração de fundo. A proposta utiliza os algoritmos *Running gaussian average* e *Temporal median filter*, estes métodos não apresentam bom desempenho a mudanças rápidas de iluminação. O método utilizado neste trabalho, possui bom desempenho quando ocorrem mudanças rápidas de iluminação e sua representação possui 3 estados: preto para regiões de fundo, branco para objetos móveis e cinza para possíveis sombras.

Para novos experimentos com robôs móveis e robôs manipuladores móveis faz-se necessário direcionar as pesquisas para um módulo de localização ou aprofundar os estudos em localização ou localização e mapeamento simultâneo.

No processo de mapeamento, a localização é de extrema importância. Um sensor está fixado em uma posição conhecida, é fácil obter a representação do ambiente, adicionar ou retirar novas leituras, no entanto, em plataformas móveis é necessário saber a localização para adicionar novas leituras, definir o que são objetos fixos e objetos móveis. O método de GMM estudado neste trabalho pode ser empregado no auxílio das tarefas que envolvam localização, mas não é capaz de resolver todas as dificuldades apresentadas.

A princípio é possível utilizar representação de mapas topológicos e estimar a localização com filtros de Kalman, esta solução pode atender aplicações limitadas. Para atender aplicações mais robustas, métodos para extração de características devem ser estudados. Podemos citar os métodos *Scale Invariant Feature Transform* (SIFT), *Speeded Up Robust Features* (SURF) e *Normal Aligned Radial Feature* (NARF). Estes métodos são

empregados em sistemas de localização e localização e mapeamento simultâneo, esta técnica é conhecida como SLAM. No estado da arte encontramos métodos avançados para desempenhar tarefas de SLAM, a citar: SLAM6D e VisualSLAM.

As técnicas citadas são utilizadas na robótica móvel e já apresentaram resultados satisfatórios para desempenhar as tarefas necessárias pelo módulo de mapeamento. São encontradas implementações destes métodos nas bibliotecas OpenCV e MRPT.

A plataforma de desenvolvimento pode explorar o Robotic Operating System (ROS) e a biblioteca Point Clouds Library (PCL). São softwares robustos e utilizados em aplicações de robótica.

Com novas pesquisas será possível colaborar para construção de um módulo robusto para mapeamento robótico e ser utilizado por robôs manipuladores, robôs móveis e robôs manipuladores móveis.

7. Referências Bibliográficas

AROCA, R. V. (2008). Análise de sistemas operacionais de tempo real para aplicações de robótica e automação. Escola de Engenharia de São Carlos, Dinâmica das Máquinas e Sistemas, 31/10/2008, São Carlos, Brasil.

BAF, F. E.; BOUWMANS, T.; VACHON, B. (2008). A fuzzy approach for background subtraction. In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference*, 2008.

BAPTISTA JÚNIOR, A.; GALDAMES, J. P. M.; BECKER, M.; CAURIN, G. A. P.; WOLF, D. F. (2010). Mapeamento de ambientes indoor utilizando mapas de multi-camadas de superfície baseado em odometria e sensor laser . In: *VI CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA*, 2010, Campina Grande PB. CONEM.

BESL, P; MCKAY, H. (1992) “A method for registration of 3-d shapes”, In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume:14 , Issue: 2)*.

BORENSTEIN, J.; EVERETT, H.R.; FENG, L.; WEHE, D. (1997). Mobile Robot Positioning - Sensors and Techniques. *Invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots*, VOL. 14, No. 4, p. 231-249, 1997.

CENSI, A. (2008). An ICP variant using a point-to-line metric, In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

DIEBEL, J. R.; THRUN S.; BRÜNIG, M. (2006). A Bayesian Method for Probable Surface Reconstruction and Decimation - *ACM Transactions on Graphics (TOG)*, 2006, VOL. 25 Edição 1: p. 39-59, Nova York, EUA.

DOZIO L.; MAMTEGAZZA P. (2003) Linux Real Time Application Interface (RTAI) in low cost high performance motion control - *Motion Control 2003, a conference of ANIPLA, Associazione Nazionale Italiana per l'Automazione (National Italian Association for Automation)*, Milano, Italy, 27-28 March 2003.

ELFES, A. (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, VOL. RA-3 N° 3:p. 249–265, Junho 1987.

ENGELSON, S.; MCDERMOTT, D. (1992). Error correction in mobile robot map learning. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, p. 2555–2560, Nice, France, Maio 1992.

GERKEY, B.; VAUGHAN, R.T.; HOWARD, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International*

Conference on Advanced Robotics (ICAR 2003), p. 317-323, Coimbra, Portugal, June 2003.

INTERNATIONAL STANDARD. ISO/IEC 14772-1, 1997. Disponível em: <<http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97>>. Acesso em: 25 Jun. 2010.

LU, F.; MILIOS, E. (1994) Robot Pose Estimation in Unknown Environments by Matching 2D Scans, In *IEEE Computer Vision and Pattern Recognition Conference (CVPR)*, p. 935-938.

KAWTRAKULLPONG, P.; BOWDEN, R. An improved adaptive background mixture model for real-time tracking with shadow detection. Video Based Surveillance Systems: In *Computer Vision and Distributed Processing*, p. 135–144, 2001.

KINECT. Kinect – Xbox.com. Microsoft, Jul 2013. Disponível em: <<http://www.xbox.com/en-us/kinect/>>. Acesso em: 10 Julho 2013.

KOLLER, D.; WEBER, J.; HUANG, T.; MALIK, J.; OGASAWARA, G.; B. Rao, and S. Russel. “Towards robust automatic traffic scene analysis in real-time.” In *Proc. of the International Conference on Pattern Recognition*, Israel, 1994.

KUIPERS, B.; BYUN, Y. T. (1990). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. TR, University of Texas at Austin.

KÜMMERLE, R.; TRIEBEL, R.; PFAFF, P.; BURGARD, W. (2007). Monte Carlo Localization in Outdoor Terrains using Multi-Level Surface Maps In *Proc. of the 6th International Conference on Field and Service Robotics (FSR 2007)*, Chamonix, France, 2007.

LABROSSE, J. J. (2002). MicroC/OS-II: The Real-Time Kernel. 2nd Edition, CMP Books.

LAMON, P.; STACHNISS, C.; TRIEBEL, R.; PFAFF, P.; PLAGEMANN, C.; GRISETTI, G.; KOLSKI, S.; BURGARD, W.; SIEGWART, R. (2006). Mapping with an Autonomous Car. In *Proc of the Workshop on Safe Navigation in Open and Dynamic Environments, IEEE International Conference on Intelligent Robots and Systems (IROS 2006)*, Beijing, China, 2006.

MAPEAMENTO3DEESCUSP, Disponível em: <http://www.mecatronica.eesc.usp.br/wiki/index.php/Mapeamento_3D_de_ambientes_indoor_e_outdoor>. Acesso em: 22 jun. 2010.

MRPT, Disponível em: <<http://www.mrpt.org>>. Acesso em: 04 julho 2013.

KINECT. Xbox 360 + kinect. Microsoft, Jul 2013. Disponível em: <<http://msdn.microsoft.com/en-us/library/jj131033.aspx>>. Acesso em: 10 Julho 2013.

MURPHY, R. R. (2004) Activities of the rescue robots at the world trade center from 11-21 September 2001. *IEEE Robotics & Automation Magazine*, 11(3), 2004.

MONTEMERLO, M. (2003). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association. 2003. 123 f. (Doutorado em computação) - School of Computer Science, Carnegie Mellon University, Pittsburgh

MORETIN, L.G. (2010). Estatística básica: probabilidade e inferência, Person Prentice Hall, São Paulo, 2010.

NÜCHTER, A.; LINGEMANN, K.; HERTZBERG, J.; SURMANN, H.; PERVÖLZ, K.; HENNING, M.; TIRUCHINAPALLI, K. R.; WORST, R.; CHRISTALLER, T. (2005). Mapping of Rescue Environments with Kurt3D, In *Proceedings of the International Workshop on Safty, Security and Rescue Robotics (SSRR '05)*, ISBN 0-7803-8946-8, pages 158 - 163, Kobe, Japan, June 2005.

NÜCHTER, A.; LINGEMANN, K.; HERTZBERG, J. (2007). 6D SLAM – 3D Mapping Outdoor Environments. *Journal of Field Robotics*, Vol. 24, No. 8-9, Junho, 2007, p. 699-722

NÜCHTER, A. (2009). 3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom. Springer, 2009.

NVIDIA, Disponível em: <http://www.nvidia.com.br/object/cuda_home_new_br.html>. Acesso em: 04 julho 2013.

PEDRO, L. M. Uma proposta de sistema robótico para manipulação e interação física segura em ambientes não estruturados. Tese (Doutorado) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

PFAFF, P.; BURGARD, W. (2005). An Efficient Extension of Elevation Maps for Outdoor Terrain Mapping, In *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, p. 165-176, 2005.

PFAFF, P.; TRIEBEL, R.; STACHNISS, C.; LAMON P., BURGARD W., SIEGWART R. (2007). Towards Mapping of Cities. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

PHAM, V.; VO, P; THANH, H.V.; HOAI, V.L. (2010) GPU Implementation of Extended Gaussian Mixture Model for Background Subtraction, *IEEE-RIVF 2010 International*

Conference on Computing and Telecommunication Technologies, Vietnam National University, Nov. 01-04, 2010. DOI: 10.1109/RIVF.2010.5634007.

RIDDER, C.; MUNKELT, O.; KIRCHNER, H. Adaptive Background Estimation and Foreground Detection using Kalman-Filtering, In *Proceedings of International Conference on recent Advances in Mechatronics, ICRAM'95, UNESCO Chair on Mechatronics, 193-199, 1995.*

RUSINKIEWICZ, S.; LEVOY, M. (2001) Efficient Variants of the ICP Algorithm, In *International Conference on 3D Digital Imaging and Modeling (3DIM 2001)*

SEGAL, A.; HAEHNEL, D.; THRUN, S. (2009) Generalized-ICP, In *Proceedings of Robotics: Science and Systems, Seattle, USA, Junho 2009.*

STAUFFER, C.; GRIMSON, W.E.L. (1999). Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. (Volume:2)*

THRUN, S.; BENNEWITZ, M.; BURGARD, W.; CREMERS, A.B.; DELLAERT, F.; FOX, D.; HÄHNEL, D.; ROSENBERG, C.; ROY, N.; SCHULTE, J.; SCHULZ, D. (1999). MINERVA: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).*

THRUN, S.; BEETZ, M.; BENNEWITZ, M.; BURGARD, W.; CREMERS, A.B.; DELLAERT, F. FOX, D.; HÄHNEL, D.; ROSENBERG, C.; ROY, N.; SCHULTE, J.; SCHULZ, D. (2000a) Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotics Research*, 19(11):972-999, 2000a.

THRUN, S.; FOX, D.; BURGARD, W. (2000b). A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping In *IEEE International Conference on Robotics and Automation*, 2000b, San Francisco.

THRUN, S. (2002). Robotc mapping: A survey. In *G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium*, Morgan Kaufmann.

THRUN, S.; BURGARD, W.; FOX, D. (2005) Probabilistic Robotics. MIT Press, Cambridge, MA.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; NIEKERK, J.; JENSEN, E.; ALESSANDRINI, P.; BRADSKI G., DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. (2006). Stanley: The Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics*, Vol. 23, No. 9, Jun, 2006, p. 661-692

TOMATIS, N., NOURBAKHSH, I. SIEGWART, R. (2003). Hybrid simultaneous localization and map building: a natural integration of topological and metric, *Robotics and Autonomous Systems*, Vol. 44, Issue 1, 31 Jul, 2003, p. 3-14

TRIEBEL, R.; PFAFF, P.; BURGARD, W. (2006). Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing In *Proc. of the International Conference on Intelligent Robots and Systems(IROS 2006)*, Beijing, China, 2006.

WOLF, D. F.; SUKHATME, G. S. (2003). Towards Mapping Dynamic Environments, ICAR.

WOLF, D. F.; SUKHATME, G. S. Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments, *Journal of Computer Science, Autonomous Robots*, Springer Netherlands, Vol. 19, No. 1, 2005, p. 53-65

ZIVKOVIC, Z. Improved adaptive Gaussian mixture model for background subtraction, *International Conference Pattern Recognition*, Vol.2, pages: 28-31, 2004.

ZIVKOVIC, Z.; HEIJDEN, F.V.D., Efficient adaptive density estimation per image pixel for the task of background subtraction, *Pattern Recognition Letters*, vol. 27, no. 7, pages 773-780, 2006.