Smith Washington Arauco Canchumuni

Localização e Mapeamento Probabilístico Simultâneos de Robôs Móveis em Ambientes Internos com um Sensor de Varredura a Laser

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE ENGENHARIA MECÂNICA

Programa de Pós-Graduação em Engenharia Mecânica

Rio de Janeiro, abril de 2013

Pontifícia Universidade Católica
do Rio de Janeiro

Smith Washington Arauco Canchumuni

Localização e Mapeamento Probabilístico Simultâneos de Robôs Móveis em Ambientes Internos com um Sensor de Varredura a Laser

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica do Departamento de Engenharia Mecânica da PUC–Rio

Orientador: Prof. Marco Antonio Meggiolaro

Rio de Janeiro
Abril de 2013

Pontifícia Universidade Católica
do Rio de Janeiro

Smith Washington Arauco Canchumuni

Localização e Mapeamento Probabilístico Simultâneos de Robôs Móveis em Ambientes Internos com um Sensor de Varredura a Laser

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica do Centro Técnico Científico da PUC–Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marco Antonio Meggiolaro
Orientador
Departamento de Engenharia Mecânica — PUC–Rio

Prof. Mauro Speranza Neto
Departamento de Engenharia Mecânica – PUC-RIO

Prof. Karla Tereza Figueiredo Leite
Departamento de Engenharia Elétrica – PUC-RIO

Prof. Max Suell Dutra
Departamento de Engenharia Mecânica – Coppe

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 12 de abril de 2013

Smith Washington Arauco Canchumuni

Formado em Engenharia Mecatrônica pela Universidad Nacional de Ingeniería – UNI, Lima, Peru (2005-2009).

Ficha Catalográfica

Arauco Canchumuni, Smith Washington

Localização e Mapeamento Probabilístico Simultâneos de Robôs Móveis em Ambientes Internos com um Sensor de Varredura a Laser / Smith Washington Arauco Canchumuni; orientador: Marco Antonio Meggiolaro. — Rio de Janeiro PUC–Rio, Departamento de Engenharia Mecânica, 2013.

v., 99 f: il. (color) ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica.

Inclui referências bibliográficas.

1. Engenharia Mecânica – Tese. 2. SLAM;. 3. Sobreposição de Varreduras do Laser;. 4. Evolução Diferencial;. 5. Medidor de Varredura a Laser;. 6. Robô Móvel;. 7. Robótica Probabilística..
I. Meggiolaro, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. III. Título.

CDD:621

Dedicado aos meus pais, Atilio e Juana, a meus tios, Joseph e Maria pelo apoio todo este tempo.

Agradecimentos

A meu orientador Prof. Marco Antonio Meggiolaro, que sob sua orientação, confiança e suporte acadêmico foi possível a realização deste trabalho.

Aos Profs. Ricardo Rodríguez Bustinza e Nilton Cesar Anchayhua Arestegui, da Universidad Nacional de Ingeniería, pelo apoio para começar meus estudos de pós-graduação.

À PUC-Rio, através dos professores e da equipe técnica e administrativa do Departamento de Engenharia Mecânica.

Finalmente à CAPES, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Resumo

Arauco Canchumuni, Smith Washington; Meggiolaro, Marco Antonio. **Localização e Mapeamento Probabilístico Simultâneos de Robôs Móveis em Ambientes Internos com um Sensor de Varredura a Laser**. Rio de Janeiro, 2013. 99p. Dissertação de Mestrado — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Os Robôs Móveis são cada vez mais inteligentes, para que eles tenham a capacidade de se mover livremente no interior de um ambiente, evitando obstáculos e sem assistência de um ser humano, precisam possuir um conhecimento prévio do ambiente e de sua localização. Nessa situação, o robô precisa construir um mapa local de seu ambiente durante a execução de sua missão e, simultaneamente, determinar sua localização. Este problema é conhecido como Mapeamento e Localização Simultâneas (SLAM). As soluções típicas para o problema de SLAM utilizam principalmente dois tipos de sensores: (i) odômetros, que fornecem informações de movimento do robô móvel e (ii) sensores de distância, que proporcionam informação da percepção do ambiente. Neste trabalho, apresenta-se uma solução probabilistica para o problema SLAM usando o algoritmo DP-SLAM puramente baseado em medidas de um LRF (Laser Range Finder), com foco em ambientes internos estruturados. Considera-se que o robô móvel está equipado com um único sensor 2D-LRF, sem nenhuma informação de odometria, a qual é substituída pela informação obtida da máxima sobreposição de duas leituras consecutivas do sensor LRF, mediante algoritmos de Correspondência de Varreduras (Scan Matching). O algoritmo de Correspondência de Varreduras usado realiza uma Transformada de Distribuições Normais (NDT) para aproximar uma função de sobreposição. Para melhorar o desempenho deste algoritmo e lidar com o LRF de baixo custo, uma reamostragem dos pontos das leituras fornecidas pelo LRF é utilizada, a qual preserva uma maior densidade de pontos da varredura nos locais onde haja características importantes do ambiente. A sobreposição entre duas leituras é otimizada fazendo o uso do algoritmo de Evolução Diferencial (ED). Durante o desenvolvimento deste trabalho, o robô móvel "iRobot Create", equipado com o sensor LRF "Hokuyo URG-04lx", foi utilizado para coletar dados reais de ambientes internos, e diversos mapas 2D gerados são apresentados como resultados.

Palavras–chave

SLAM; Sobreposição de Varreduras do Laser; Evolução Diferencial; Medidor de Varredura a Laser; Robô Móvel; Robótica Probabilística.

Abstract

Arauco Canchumuni, Smith Washington; Meggiolaro, Marco Antonio (Advisor). **Probabilistic Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments with a Laser Range Finder**. Rio de Janeiro, 2013. 99p. MSc. Dissertation — Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

The robot to have the ability to move within an environment without the assistance of a human being, it is required to have a knowledge of the environment and its location within it at the same time. In many robotic applications, it is not possible to have an a priori map of the environment. In that situation, the robot needs to build a local map of its environment while executing its mission and, simultaneously, determine its location. A typical solution for the Simultaneous Localization and Mapping (SLAM) problem primarily uses two types of sensors: i) an odometer that provides information of the robot's movement and ii) a range measurement that provides perception of the environment. In this work, a solution for the SLAM problem is presented using a DP-SLAM algorithm purely based on laser readings, focused on structured indoor environments. It considers that the mobile robot only uses a single 2D Laser Range Finder (LRF), and the odometry sensor is replaced by the information obtained from the overlapping of two consecutive laser scans. The Normal Distributions Transform (NDT) algorithm of the scan matching is used to approximate a function of the map overlapping. To improve the performance of this algorithm and deal with low-quality range data from a compact LRF, a scan point resampling is used to preserve a higher point density of high information features from the scan. An evolution differential algorithm is presented to optimize the overlapping process of two scans. During the development of this work, the mobile robot "iRobot Create", assembled with one LRF "Hokuyo URG-04LX", is used to collect real data in several indoor environments, generating 2D maps presented as results.

Keywords

SLAM; Scan Matching; Evolution Differential; Laser Range Finder; Robot Mobile; Probabilistic Robotics.

Sumário

Lista de figuras

Lista de tabelas

O sábio não é o que sabe, o sábio é o que faz aquilo que sabe.

Nuno Cobra

Sumário das notações

Símbolos romanos

C_r	A Probabilidade de crossover	[–]
F	Fator de Escala de Evolução Diferencial	[–]
f_s	Frequência do Scan	[Hz]
g	Número de Generação em Evolução Diferencial	[–]
L_i	Saliência de un ponto dirigido	[–]
M	Tamanho do mapa	[–]
N_P	Número de População	[–]
$P_{x,g}$	Número de População na geração g	[–]
P_r	Posição de Referência do Robô móvel	$[(m, m, rad)]$
P_n	Posição nova do Robô móvel	$[(m, m, rad)]$
P	Número de Partículas	[–]
P_i	Ponto Dirigido	[–]
R_θ	Orientação do Robô móvel	$[rad]$
R	Localização do Robô móvel	$[(m, m, rad)]$
R_x	Localização no eixo x do Robô móvel	$[m]$
R_y	Localização no eixo y do Robô móvel	$[m]$
S_{ref}	Varredura de Referência	[–]
S_{atu}	Varredura Atual	[–]
t	Tempo	$[seg]$
u_t	Controle do Movimento do robô móvel no tempo t	$[m]$
v_{max}	Velocidade Translacional Máxima do Robô Móvel	$[\frac{m}{seg}]$
w_{max}	Velocidade Rotacional do Robô Móvel	$[\frac{m}{seg}]$

x_B^A	Rotação e Traslação no Plano entre A e B	[–]
z_t	Medição do sensor no tempo t	$[m]$

Símbolos gregos

Δx	Câmbio do deslocamento no eixo x do robô móvel	[-]
Δy	Câmbio do deslocamento no eixo y do robô móvel	[-]
$\Delta\theta$	Câmbio da orientação do robô móvel	[-]
$\oplus$	Composição de Transformações	[–]

Subscritos

$2D$	Bidimensional
$3D$	Tridimensional
DP	Distribuição de Partículas
$DP-SLAM$	Distribuição de particulas-Mapeamento e Localização Simultânea
ED	Evolução Diferencial
EKF	Filtro de Kalman Estendido
EIF	Filtro Estendido de Informação
$FastSLAM$	Fatorada Solução de Mapeamento e Localização Simultânea
FP	Filtro de Partículas
FDP	Função Densidade de Probabilidade
GPS	Global Positioning System
ICP	Iterative Closest Point
IDC	Iterative Dual Correspondence
ICL	Iterative Closest Line
LRF	Laser Ranger Finder

MS Robotics	Microsoft Robotics Developer Studio
NDT	Transformada de Distribuíções Normais
SLAM	Mapeamento e Localização Simultânea
VPL	Linguagem de Programação Visual

1
Introdução

Avanços significativos têm sido feitos no sentido de criar um robô capaz de realizar tarefas completamente autônomas. As tarefas básicas como planejamento de trajetórias, localização e navegação são bem compreendidas, e até um certo grau, foram resolvidas. Estes componentes são as ferramentas básicas para a resolução de tarefas de um nível superior, e permitir que o robô possa realizar alguma missão sem supervisão ou intervenção humana. Para alcançar qualquer nível útil de autonomia, mesmo meramente viajar sem ajuda de um ponto específico para outro, o robô precisa ter um retrato bastante completo do mundo, um bom mapa do meio ambiente. Este mapa, além de ser preciso, deve ser bastante completo englobando todos os lugares que o robô possa alcançar na sua exploração. Este problema de monitorar a posição de um robô e construir um mapa é conhecido como Localização e Mapeamento Simultâneo, ou SLAM.

O problema geral de SLAM tem sido objeto substancial de pesquisa para a comunidade robótica inclusive em áreas como sistemas de navegação de veículos tripulados e prospecção geofísica. Thrun [1] define a localização de robôs móveis como o problema de determinar a posição de um robô em relação a um determinado mapa do meio ambiente. No entanto, em muitas aplicações da robótica, não é possível dispor de um mapa, a priori, do meio ambiente. Em tais situações, o problema pode ser tratado através da construção de mapas locais do meio ambiente, enquanto o robô está executando uma missão e, posteriormente, determinar a posição do robô, combinando os mapas locais[3].

"SLAM tem sido um obstáculo para robôs autônomos. Os problemas de localização e mapeamento aparecem como dois desafios distintos mas, de fato, são intricadamente problemas entrelaçados."[4]. Para que um robô atualize o mapa corretamente, é necessário conhecer a localização do robô quando a observação do ambiente é feita. No entanto, para monitorar a localização do robô, é essencial ter um bom mapa com a qual comparar as observações. Resolver ambos problemas de forma incremental e simultânea significa que um pequeno erro em cada solução pode facilmente corromper todas as estimativas futuras.

Assim, a rápida acumulação de erros pequenos pode causar o fracasso de quase todos os métodos para resolver o problema de SLAM.

SLAM aborda o problema de um robô móvel autônomo que começa seu movimento em um local desconhecido, em um ambiente desconhecido e utilizando apenas observações relativas do meio ambiente, incrementalmente constroi um mapa deste ambiente, e simultaneamente usa este mapa para computar a localização absoluta do robô móvel[5][6]. Assim, a principal vantagem de SLAM, é que elimina a necessidade de infraestruturas artificiais ou um conhecimento topológico a priori do ambiente.

Para isso, o robô precisa interagir com o ambiente, que normalmente é um sistema dinâmico que apresenta inúmeras características. O robô móvel pode adquirir informações do ambiente usando seus sensores e também pode influenciar a percepção do seu ambiente através de seus atuadores.

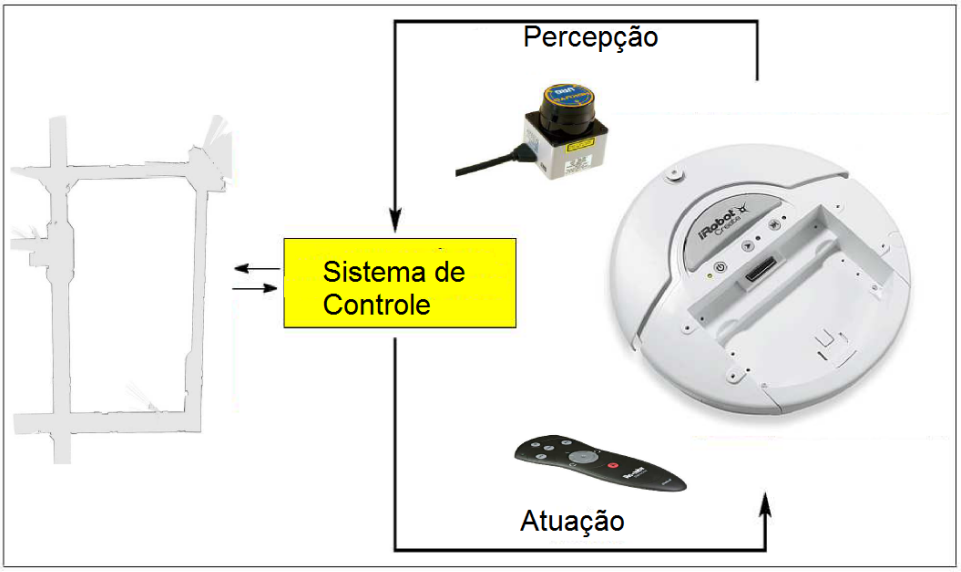


Figura 56: Interação do robô Móvel com o ambiente.

A maioria dos algoritmos utilizados para resolver o problema de SLAM precisam de sensores para capturar a percepção do ambiente, como sensores para capturar o deslocamento do robô móvel[7]. No entanto, os sensores são sensíveis ao ruído, e existem alguns objetos no ambiente que podem não ser detectados diretamente pelo sensor. Mas os sensores a laser, como Laser Range Finders (LRF) apresentam grandes vantagens, porque são capazes de localizar o robô móvel, mapear o ambiente e detectar obstáculos.

Para evitar a utilização de sensores que percebem o movimento das rodas do robô móvel (por exemplo os sensores de odometria), algoritmos de Correspondência de Varreduras têm sido amplamente utilizados nos últimos anos para realizar a localização do robô móvel[8]. Eles não assumem restrições geométricas nem representações do ambiente. Assim, é bem adaptado para a localização de um robô móvel com uma precisão elevada, tanto em ambientes estruturados e não estruturados [9].

Pelos motivos mencionados acima, este trabalho concentra-se no desenvolvimento de um método para resolver o problema de SLAM com um único sensor (LRF) de baixo custo que pode produzir uma aproximada representação do ambiente interno.

1.1 Motivação

- Uma solução para o problema de SLAM seria de valor inestimável para uma gama de aplicações em que a posição absoluta ou a informação exata de um mapa é inalcançável, incluindo, entre outros, exploração planetária autônoma, veículos autônomos submarinos, aéreos e de todo o terreno em tarefas como mineração e construção.
- Mapas poderiam ser feitos em, ambientes fechados, áreas que são perigosas ou inacessíveis aos seres humanos, tais como ambientes no fundo do mar ou estruturas instáveis.
- Uma solução para o problema de SLAM evitaria o uso de métodos alternativos de localização, como GPS. Seria assim possível a navegação de robôs em lugares onde o sinal GPS não esteja disponível, estações espaciais e outros planetas. Mesmo em locais onde esteja disponível um sistema GPS, a solução para o problema de SLAM seria inestimável devido à baixa precisão do GPS.
- O uso de sensores de odometria não é uma boa escolha devido ao fato das rodas do robô normalmente podem deslizar no chão. A localização por câmeras também não apresenta bons resultados em alguns ambientes naturais, devido à alta similaridade entre as imagens de vegetação, tornando difícil um confiável estabelecimento de pontos-chave.

1.2 Objetivos do Trabalho

Os objetivos principais do trabalho são:

- Realizar um Mapeamento e Localização Simultânea de Robôs Móveis em Ambientes Internos, utilizando um robô (iRobot Create) equipado com um único sensor 2D-LRF de baixo custo e de capacidade limitada ($URG-04LX-UG01$).
- Propor uma variante do algoritmo DP-SLAM, modificando o modelo de movimento do robô móvel por um modelo de varredura do sensor LRF, para utilizar o algoritmo sem nenhuma informação de odometria.
- Trabalhar o problema de Mapeamento e Localização Simultânea com um único sensor e sem conhecer a modelagem do robô móvel.
- Desenvolver um algoritmo para obter o deslocamento do robô móvel utilizando as medições do sensor 2D-LRF de baixo custo ($URG-04LX-UG01$), e otimizando o deslocamento mediante Algoritmos Genéticos .

1.3 Revisão Bibliográfica

1.3.1 SLAM

Uma forma de categorizar algoritmos de mapeamento é pelo tipo de mapa. Em geral, o mapa pode ser topológico ou métrico. Mapas métricos representam distâncias explícitas do meio ambiente. Estes mapas podem ser $2D$, geralmente uma projeção vertical, ou em $3D$, ou seja, um mapa volumétrico do ambiente. Além disso, as abordagens de SLAM podem ser classificadas pelo número de graus de liberdade da posição do robô móvel [10]. Uma representação estimada em $3D$ contém coordenadas (x,y) e a rotação θ, enquanto uma representação estimada em $6D$, considera todos os graus de liberdade que o robô móvel pode ter, ou seja coordenadas (x,y,z) e os ângulos rolagem, guinada e arfagem.

1.3.1.1 Mapeamento Planar $2D$

Uma das técnicas para a abordagem de mapeamento planar métrico é usando métodos probabilísticos, onde o robô tem modelos de movimento probabilísticos e modelos de percepção da incerteza. Ao integrar estas duas

distribuições, por exemplo, através do filtro de Kalman, ou de partículas, é possível localizar o robô móvel[2]. O mapeamento é muitas vezes uma extensão deste problema de estimação. Além da posição do robô, pontos de referência do mapa são estimados. Circuitos fechados, ou seja, um segundo encontro de uma área previamente visitada do ambiente, têm um papel especial aqui. Uma vez detectado, existem algoritmos que permitem limitar o erro deformando a área já mapeada de tal forma que um modelo topologicamente consistente seja criado. No entanto, não há garantia para um modelo preciso.

Um número de abordagens tem sido proposto para tratar tanto o problema de SLAM e como também problemas de navegação mais simplificada, onde um mapa adicional ou informações de localização do robô móvel são disponibilizadas. Thrun [11] fez uma revisão das técnicas existentes, como a estimativa de máxima probabilidade [12], expectativa de maximização [13], filtro de Kalman estendido (EKF) [14] ou filtros estendidos de informação (EIF) [15]. Além destes métodos, existe o FastSLAM [16], que aproxima as probabilidades posteriores por partículas e o método de Lu e Milios com base na Correspondência de Varreduras [9] . Finalmente, há o algoritmo baseado em LRF que não faz suposições de características do ambiente, DP-SLAM [4], trabalhado também por Ynoquio [2], que desempenha um papel importante no mapeamento $2D$.

SLAM baseado em Filtro de Kalman O mais popular, pois diretamente proporciona tanto uma solução recursiva para o problema de navegação quanto uma forma de calcular estimativas consistentes para a incerteza do robô móvel e localizações das características do mapa, com base em modelos estatísticos para o movimento do robô e observações das características relativas.

SLAM baseado em Probabilidades A segunda filosofia é evitar a necessidade de estimativas de posição absoluta e medidas precisas de incerteza e, em vez de empregar um conhecimento mais qualitativo da localização relativa das características do mapa e do robô móvel, constroi mapas e guia o movimento. Esta filosofia geral tem sido desenvolvida por vários grupos diferentes em um número de maneiras diversas [14]. Abordagem qualitativa para a navegação e o problema de SLAM tem muitas vantagens potenciais sobre a metodologia da teoria estimativa em termos de limitar a necessidade de modelos precisos e os requisitos computacionais.

Correspondência de Varreduras-SLAM Uma filosofia muito ampla, que fornece uma alternativa ao Filtro de Kalman ou o formalismo estatístico, man-

tendo uma abordagem essencialmente numérica ou computacional para resolver o problema de SLAM e navegação. Essas abordagens incluem a utilização de correspondência de duas Varreduras [17], registro global de mapas, regiões delimitadas e outras medidas para descrever a incerteza. As primeiras tentativas de realizar a localização de um robô móvel por correspondência de duas varreduras sucessivas foram inspiradas por Besl e McKay [18], que apresentaram o algoritmo Iterative Closest Point (ICP).

Lu e Milios [9] propuseram algumas alterações ao algoritmo ICP original para torná-lo mais adequado para aplicações robóticas. Além disso Alshawa [19] propor o algoritmo Iterative Closest Line (ICL).

FastSLAM A maioria das abordagens para resolver o problema de SLAM tem que lidar com um grande número de pontos de referência presentes em ambientes reais. Algoritmos baseados no filtro de Kalman, por exemplo, exigem um tempo quadrático em relação ao número de pontos de referência para incorporar cada observação do sensor [16]. O algoritmo FastSLAM, por outro lado recursivamente estima a completa distribuição posterior sobre a posição do robô e dos pontos de referência locais, a escalas que variam logaritmicamente com o número de pontos de referência do mapa. FastSLAM segue uma proposta feita por Murphy [20], usando um filtro de partículas Rao-Blackwellized para amostrar a posição do robô e acompanhar a posição de um número fixo de pontos de referência predeterminados utilizando um filtro de Kalman. *"As posições dos pontos de referência são condicionalmente independentes, dada a posição do robô"* [20].

Este método reduz alguns dos desafios em mapeamento, à custa de alguns desafios na seleção e identificação dos pontos de referência. Este último pode envolver um problema de associação de dados bastante complicado.

DP-SLAM Algoritmo puramente baseado em laser (LRF) e não faz suposições dos pontos de referência. DP-SLAM evita o problema de associação de dados, armazenando vários mapas detalhados em vez de pontos de referências esparsos, assim, assumindo associação com a localização. O algoritmo usa um filtro de partículas para representar tanto a posição de robô e as configurações possíveis do mapa. Usando uma nova representação do mapa, que chamaram de mapeamento por distribuição de partículas (PD-Mapping) [21] e [22], são capazes de manter e atualizar centenas ou mi-

lhares de candidatos de mapas e posições do robô em tempo real como o movimento do robô através do ambiente.

Em princípio, os métodos probabilísticos de mapeamento $2D$ planar são extensíveis para mapeamento $3D$ [23], No entanto, para nosso conhecimento nenhuma extração de características de confiança, nem uma estratégia para reduzir os custos computacionais, já foi publicado. O desvio qualitativo da complexidade é devido à necessidade de recolher amostras em cada dimensão.

O problema SLAM pode ser definido como um problema de otimização global, em que o objetivo é buscar o espaço de possíveis mapas do robô. Assim um algoritmo genético pode ser descrita para resolver este problema[24][25].

1.3.1.2 Mapeamento $3D$

Mapas tridimensionais podem ser gerados por três técnicas diferentes: em primeiro lugar, um método de localização planar combinado com um sensor $3D$; em segundo lugar, uma precisa estimação da posição $6D$ combinada com um sensor $2D$; e em terceiro lugar, um sensor $3D$ com um método de localização $6D$ [10]. As Tabelas 1.1 e 1.2 resume estas técnicas de mapeamento, em comparação com o mapeamento planar $3D$.

Tabela 1.1: Visão geral da dimensionalidade de abordagens SLAM-mapas 2D.

	Dimensionalidade da representação da pose	
Dados do Sensor	3D	6D
2D	Mapeamento planar 2D	Slice-wise 6D SLAM
3D	Mapeamento planar 3D	Full 6D SLAM

Mapeamento 3D planar. Em vez de usar um sistema de varredura $3D$, que produz consistentes varreduras $3D$ scans, alguns grupos têm tentado construir representações volumétricas de ambientes com LRF $2D$. Em trabalhos anteriores, são usados dois LRF $2D$ para a aquisição de dados em $3D$. Um LRF é montado horizontalmente e outra verticalmente. Este último captura uma linha de exploração vertical, que é transformada em pontos $3D$ com base na atual posição do robô móvel, porque a varredura vertical não é capaz de capturar as laterais dos objetos [26].

Tabela 1.2: Visão geral da dimensionalidade de abordagens SLAM-mapas 3D.

	Dimensionalidade da representação da pose	
Dados do Sensor	3D	6D
2D	Mapeamento 2D de sensores sonares e laser por varreduras[11]	Mapeamento 3D usando uma precisa localização, considerando a (x, y, z) posições e roll, yaw e pitch ângulos.
3D	Mapeamento 3D, utilizando un método de localização planar	Mapeamento 3D, utilizando um laser por varreduras 3D ou câmaras com posições estimadas a partir dos dados do sensor.

1.4 Roteiro da Dissertação

O capítulo 2 discute a modelagem matemática do problema, ou seja, as equações básicas de probabilidade, fundamentos de Correspondência de Varreduras, tipos de mapas e conceitos de otimização por evolução diferencial, à descrição dos métodos usados para resolver o problema de SLAM: DP-SLAM, Filtro de Partículas, e também é descrita a representação de mapas por DP-Mapping.

No capítulo 3 apresentam-se alguns detalhes da implementação, tais como Filtragem das Varreduras do sensor, Parâmetros e Considerações.

No capítulo 4 apresentam-se testes e Resultados, e é mostrado o mapeamento dos ambientes internos em 2D para ambientes simulados.

No capítulo 5 apresentam-se testes e Resultados, e é mostrado o mapeamento dos ambientes internos em 2D para ambientes reais.

As conclusões e os comentários finais são apresentados no capítulo final do trabalho.

2
Fundamentação Teórica

2.1
Conceitos Básicos em Probabilidade

Em robótica probabilística, quantidades, como as medições de um sensor, atuações dos controles, e os estados de um robô e seu ambiente são todos modelados como variáveis aleatórias. *Variáveis Aleatórias* podem assumir vários valores, de acordo com leis probabilísticas específicas. Inferência Probabilística estuda as leis que governam as variáveis aleatórias que normalmente são derivadas de outras variáveis aleatórias e dos dados observados [1], onde

$$p(x) = p(X = x)$$

denota a probabilidade de que a variável aleatória X assuma o valor de x. E, é claro, as probabilidades são sempre não negativas, ou seja $p(x) \geq 0$.

Uma função não negativa utilizada para representar a distribuição de probabilidade caso a variável aleatória seja contínua é a *Função Densidade de Probabilidade* (FDP). Uma função de densidade comum é a distribuição normal com média μ e variância σ^2. Esta distribuição é dada pela função *Gaussiana*

$$p(x) = \frac{1}{\sqrt{2\pi\mu^2}} e^{-\frac{(x-\mu)^2}{\sigma^2}} \tag{2-1}$$

A distribuição normal (2-1) assume que x é um valor escalar. Frequentemente, x é um vetor. Neste caso, são chamadas distribuições normais multivariáveis, dadas por

$$p(x) = \frac{1}{\sqrt{det(2\pi\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{2-2}$$

onde μ é o vetor média e Σ é a matriz de covariância, simétrica (semidefinida positiva). A *Distribuição Conjunta* de duas variáveis aleatórias X e Y é dada por:

$$p(x,y) = p(X = x \quad e \quad Y = y)$$

Esta expressão descreve a probabilidade do evento onde a variável aleatória X

assume o valor de x e Y assume o valor de y. Se X e Y são *Independentes*, temos:

$$p(x,y) = p(x) \cdot p(y)$$

Muitas vezes, as variáveis aleatórias incluem informações sobre outras variáveis aleatórias. Suponha que já sabemos que o valor Y é y, e gostaria de saber a probabilidade do valor de X ser x condicionando a esse fato. Tal probabilidade é chamada de *Probabilidade Condicional*.

$$p(x|y) = p(X = x|Y = y)$$

Se $p(y) > 0$, a probabilidade condicional é definida como:

$$p(x|y) = \frac{p(x,y)}{p(y)} \tag{2-3}$$

Um fato interessante, que resulta da definição de probabilidade condicional e dos axiomas de medidas de probabilidade, é muitas vezes referido como *Teorema da Probabilidade Total* [1]:

$$p(x) = \sum_y p(x|y) \cdot p(y), \quad \text{Discreto} \tag{2-4}$$

$$p(x) = \int p(x|y) \cdot p(y) dy, \quad \text{Contínuo} \tag{2-5}$$

2.1.1 Regra de Bayes

A regra de bayes desempenha um papel predominante na área de robótica probabilística (e inferência probabilística em geral). Ela relaciona condicionais do tipo $p(x|y)$ para seu "inverso", $p(y|x)$. A regra requer que $p(y) > 0$:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{2-6}$$

Se x for uma quantidade que gostaríamos de inferir a partir de y, a probabilidade $p(x)$ será referida como *Distribuição de Probabilidade Anterior*. A distribuição $p(x)$ resume o conhecimento que temos sobre X antes de incorporar informação de y. A probabilidade $p(x|y)$ é chamada de *Distribuição de Probabilidade Posterior* sobre X.

Se estamos interessados em inferir uma quantidade x a partir de dados Y, a regra de bayes nos permite fazê-lo através da probabilidade inversa, que especifica a probabilidade de dados y assumindo que x ocorreu. Em robótica, a probabilidade $p(y|x)$ é frequentemente denominada "modelo generativo",

porque descreve, em algum nível de abstração, como o estado da variável X causa a medição do sensor Y.

Como $p(y)$ na equação (2-6) não depende de x, o fator $p(y)^{-1}$ é escrito como um normalizador da Regra de Bayes e é denotado como η, resultando em

$$p(x|y) = \eta p(y|x)p(x) \tag{2-7}$$

2.1.2 Filtro de Bayes para SLAM

A regra de bayes é o arquétipo de inferência probabilística [11], o princípio essencial de praticamente todos os algoritmo de mapeamento.

Suponha que nós queremos aprender sobre uma quantidade x (o mapa), com base em dados de medição d (e.g. Varreduras ou odometria). Então a regra de bayes diz que, o problema pode ser resolvido pela multiplicação de dois termos: $p(x|d)$ e $p(x)$, ver Equação (2-7). O termo $p(d|x)$ especifica a probabilidade de observar a medição d sobre a hipótese de x. Assim, $p(d|x)$ é um modelo generativo, na medida em que descreve o processo de geração das medições do sensor sobre ambientes diferentes m. O termo $p(x)$ (prévio) especifica a probabilidade de assumir que x é o caso no ambiente antes da chegada de todos os dados.

$$p(x|d) = \eta \quad p(d|x)p(x) \tag{2-8}$$

No processo de SLAM há dois tipos diferentes de dados: sensores de medição z_t e controles u_t. Aqui, subscritos são usados como índice de tempo. Em particular, z_t é a medição do sensor tomada no tempo t, e u_t especifica o comando de movimento do robô no intervalo de tempo $[t-1, t]$.

No campo de mapeamento de robôs móveis, é um estimador recursivo para calcular uma sequência de distribuições de probabilidades posteriores sobre as quantidades que não podem ser observadas diretamente, como um mapa ou posição do robô. Este estado desconhecido é representado por x_t.

$$p(x_t|z^t, u^t) = \eta \quad p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) p(x_{t-1}|z^{t-1}, u^{t-1}) dx_{t-1} \tag{2-9}$$

onde z^t e u^t se refere a dados que conduzem até um tempo t, dados por

$$z^t = \{z_1, z_2, z_3, ..., z_t\}$$

$$u^t = \{u_1, u_2, u_3, ..., u_t\}$$

Notar que o filtro de Bayes é recursivo, isto é, a probabilidade posterior $p(x_t|z^t, u^t)$ é calculado a partir da mesma probabilidade de um tempo anterior. A probabilidade inicial no tempo $t = 0$ é $p(x_0|z^0, u^0) = p(x_0)$.

Se usarmos m para denotar o mapa e R para representar ao robô, da Equação (2-9) obtemos o filtro de bayes:

$$p(R_t, m_t|z^t, u^t) =$$

$$\eta \quad p(z_t|R_t, m_t) \int\int p(R_t, m_t|u_t, R_{t-1}, m_{t-1}) p(R_{t-1}, m_{t-1}|z^{t-1}, u^{t-1}) d\,R_{t-1} d\,m_{t-1} \tag{2-10}$$

Para simplificar a Equação (2-11):

- A maioria dos algoritmos de mapeamento assumem que o mundo é estático, o que implica que o índice de tempo pode ser omitido quando se refere ao mapa m.
- A maioria das abordagens assume que o movimento do robô é independente do mapa [11].
- Utilizando a hipótese de Markov, que postula que o estado atual x_t pode ser estimado usando apenas o estado da etapa anterior x_{t-1} [2].

Com essas simplificações, obtém-se uma forma conveniente do filtro de bayes:

$$p(R_t, m|z^t, u^t) = \eta \quad p(z_t|R_t, m) \int p(R_t|u_t, R_{t-1}) p(R_{t-1}, m|z^{t-1}, u^{t-1}) d\,R_{t-1} \tag{2-11}$$

Na equação (2-11) há duas importantes distribuições de probabilidades: $p(R_t|u_t, R_{t-1})$ e $p(z_t|R_t, m)$. Ambos são modelos generativos do robô e seu ambiente.

Modelo de Movimento A probabilidade $p(R_t|u_t, R_{t-1})$ é referida como modelo de movimento, e especifica o efeito de controle u_t no estado R_{t-1}. Descreve a probabilidade de que o controle u_t, se executado no estado R_{t-1}, possa levar ao estado R_t. Devido ao ruído ou efeitos exógenos não modelados. O resultado de um controle será descrito por uma probabilidade posterior. "Um modelo probabilístico adequado pode modelar com precisão os tipos específicos de incerteza que existem na atuação e percepção do robô"[1]. A nossa exposição se concentra totalmente na cinemática de robôs móveis que operam em ambientes planares, a posição de um robô móvel é resumida por três variáveis

$$R = \left(R_x, R_y, R_\theta\right)^T \tag{2-12}$$

onde $\begin{pmatrix} R_x \\ R_y \end{pmatrix}$, é a localização do robô e R_θ é a orientação do robô. Na Figura 57 mostra a posição de robô móvel em um sistema de coordenadas globais. Thrun [1] fornece em detalhes dois modelos específicos de movi-

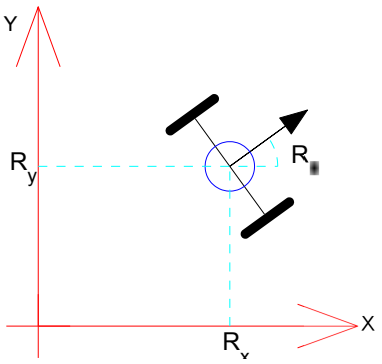


Figura 58: Posição e Orientação do Robô

mento probabilísticos, ambos para robôs móveis que operam no plano. O primeiro modelo assume que o controle de movimento u_t especifica um comando de velocidade, determinado pelos motores do robô. O segundo modelo assume que u_t contém informação de odometria (distância percorrida e ângulo de giro).

Modelo de Medição A probabilidade $p(z_t|R_t, m)$ é referida como modelo de medição, porque em termos probabilísticos descreve como as medições do sensor z_t são geradas para diferentes posições R_t e mapas m (um modelo generativo descrevendo o funcionamento dos sensores do robô). O Modelo de Medição conta a incerteza nos sensores do robô. Assim, há explicitamente modelos de ruído na medição do sensor. Na prática, segundo Thrun [1], é muitas vezes impossível modelar um sensor com precisão, principalmente por duas razões: o desenvolvimento de um modelo com precisão para o sensor pode ser extremamente demorado; e um modelo preciso pode exigir variáveis de estado que não são conhecidas, como o material da superfície.

Robótica probabilística acomoda as imprecisões dos modelos de sensores em aspectos estocásticos; modelando o processo de medição como uma densidade de probabilidade condicional, $p(z_t|R_t)$, em vez de uma função determinística $z_t = f(R_t)$, a incerteza no modelo de sensor pode ser

acomodada em aspectos não determinísticos do modelo. Robôs modernos usam uma variedade de tipos de sensores, tais como sensores tácteis, LRF, sensores de sonar ou câmeras. As especificações do modelo dependem do tipo de sensor.

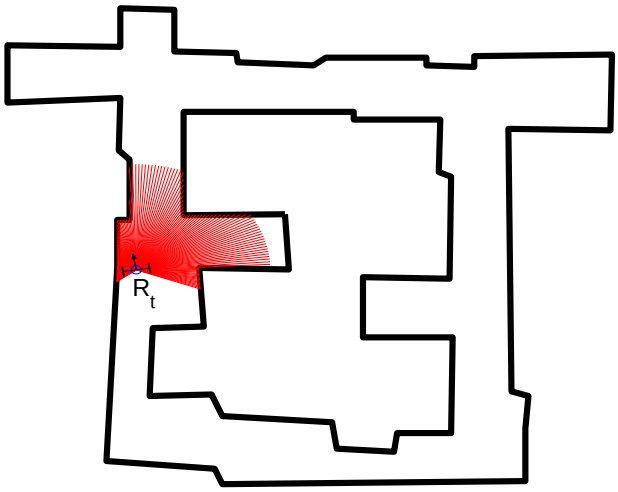


Figura 59: Robô móvel em um mapa obtendo as medições a partir do seu LRF.

Observamos também que a Equação de mapeamento (2-11) não pode ser implementada em um computador digital em sua forma global. Isso ocorre porque o posterior sobre o espaço de todos os mapas e posições do robô é uma distribuição de probabilidade sobre um espaço contínuo e, portanto, possui infinitas dimensões. Assim, qualquer algoritmo de mapeamento tem que recorrer a hipóteses adicionais. Estas hipóteses são as principais diferenças entre as diversas soluções para resolver o problema de SLAM.

2.1.3 Laser Range Finder

O desenvolvimento recente de LRFs a preços razoáveis rapidamente o transformou em um sensor dominante para utilização em mapeamento, como na localização de robôs móveis. Com os dados de varredura a laser 2D a aquisição em tempo real, um robô pode-se calcular a área de todo o espaço livre em uma sala, em seguida, ele pode selecionar o centro da sala como a sua posição para a construção do mapa[3], LRF pode dar medições precisas dentro de poucos centímetros, e pode proporcionar leituras muito mais precisas e exatas. Outra diferença importante do LRF é que o laser traça uma linha muito fina através do

ambiente, ver Figura 58 mostra o LIDAR $URG04LX-UG01$ [27], gerando leituras de alta resolução.

Figura 61: Laser Range Finder (LRF), Hokuyo URG 04LX-UG01.

2.2 Mapa

Um mapa do ambiente é uma lista de objetos no ambiente e suas localizações, e podem ser basicamente classificados em duas categorias: mapas baseados em características (*Feature-Based Maps*) e mapas baseados na Localização (*Ocupancy Grids*).

2.2.1 Representação de mapas por Grade de Ocupação

Oferecem um marcador para qualquer localização no ambiente, e geralmente não assumem restrições geométricas [28]. Visualize um ambiente plano dividido em uma grade regular de quadrados, todos de igual tamanho. Cada um destes quadrados corresponde a uma área física no ambiente, e, como tal, cada quadrado contém um conjunto diferente de objetos ou porções de um objeto. Grade de Ocupação é uma representação abstrata destas seções do mundo, contendo informações sobre se esse quadrado no ambiente real está ocupado ou não. Esta noção de ocupação pode mudar, dependendo da aplicação, mas em geral, quer indicar se a área poderia bloquear a passagem de um robô, ou se a área iria ser detectatda pelos sensores do robô móvel. Idealmente, estes dois conceitos devem os mesmos, mas podem não ser, devido ao ruído e limitações nos sensores. Grade de Ocupação pode ser representada de muitas formas diferentes. É importante assegurar que a representação é bem adaptada ao sensor a ser

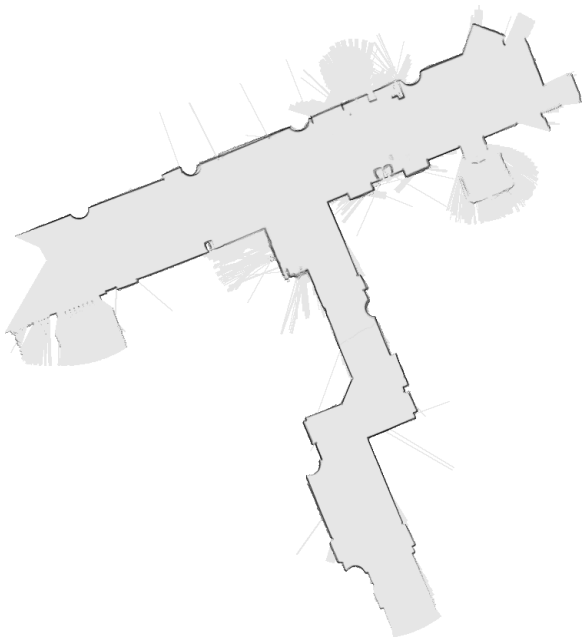


Figura 62: Representação de um ambiente mediante Grade de Ocupação

utilizado e à aplicação. A maioria das abordagens tradicionais pode ser generalizada em dois tipos: determinísticos e estocásticos [4].

Mapas Determinísticos São a representação mais simples, e têm um conjunto discreto de valores para as quadrículas. Estes geralmente são **VAZIOS** ou **OCUPADOS**, e às vezes podem incluir um valor para **DESCONHECIDO** ou **NÃO OBSERVADO**.

Mapas Estocásticos Têm uma noção de **OCUPADOS** e **VAZIOS**, assim como mapas deterministicos, mas em vez de ver estes valores como absolutos, eles têm uma escala de vários graus de ocupação. Qual a porcentagem do quadrado é acreditado para ser ocupado, ou o quanto um objeto é transparente para o sensor, são alguns dos fatores que afetam o valor de ocupação.

A representação estocástica e o modelo de observação correspondente precisam ser devidamente calibrados para o sensor utilizado.

Estes mapas têm dois grandes inconvenientes para a realização de localização. Primeiro, é computacionalmente caro combinar tais representações densas do espaço. Em segundo lugar, a própria granularidade de uma representação da Grade pode produzir estimativas de baixa resolução da posição do robô móvel.

2.2.2 Representação de mapas baseado nas Características

São constituídos por um conjunto de características, juntamente com a sua localização cartesiana, e são amplamente utilizados no contexto de localização. No entanto, estes mapas introduzem restrições geométricas, tais como a existência de linhas e cantos no ambiente [29].

A principal vantagem desse tipo de mapas é a sua compacta representação. Por outro lado, requer a existência de estruturas ou objetos que se distinguem uns dos outros o suficiente, e portanto não são adequados para modelar ambientes não estruturados.

Assim, é necessário um algoritmo extra para reconhecer e detectar características [2]. Na prática, os pontos de referência podem ter características semelhantes, o que frequentemente torna difícil distinguir um do outro. Quando isso acontece, o problema da associação de dados, também conhecido como o problema de correspondência, tem de ser tratado. *"O problema de correspondência seria determinar se as medições do sensor tomadas em diferentes pontos no tempo correspondem ao mesmo objeto físico no mundo"* [11]. É um problema difícil, porque o número de hipóteses possíveis pode crescer exponencialmente.

2.3 Interação do Robô com o Ambiente

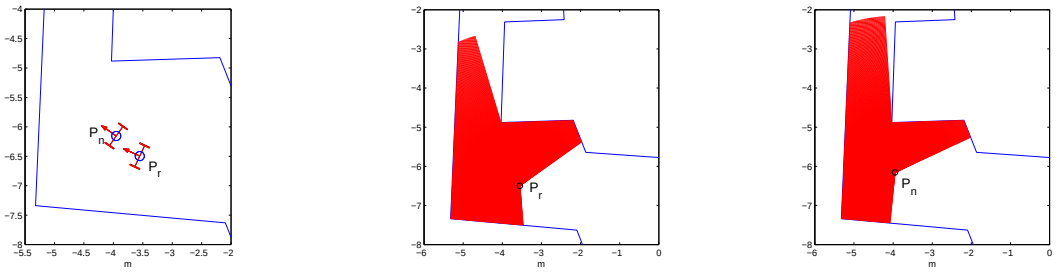
Existem dois tipos fundamentais de interações entre um robô móvel e seu ambiente: o robô pode influenciar a percepção do seu ambiente através de seus atuadores, e pode adquirir informações de seu ambiente através de seus sensores. A percepção é o processo pelo qual o robô utiliza seus sensores para obter informações sobre as características de seu ambiente. Por exemplo, um robô móvel pode levar uma câmera, um LRF , ou consultar seus sensores tácteis para receber informações sobre o ambiente. Os atuadores controlam as ações do robô móvel para alterar sua percepção do ambiente. Exemplos de ações incluem controlar o movimento do robô móvel e a manipulação de objetos, ver Figura 55.

2.4 Correspondência de Varreduras (Scan Matching)

Correspondência de Varreduras é o problema de encontrar a translação e rotação $(\triangle x, \triangle y, \triangle \theta)$ entre uma varredura de entrada e uma varredura de referência obtidos a partir de um dispositivo como o LRF, de modo que eles se

sobreponham maximamente [30].

Na Figura 64, se apresenta um robô móvel que começa na posição P_r (posição de referência), executa uma varredura S_{ref} (varredura de referência), se move através de um ambiente estático para uma nova posição P_n (posição nova) e executa outra varredura S_{atu} (varredura atual). Os algoritmos de Correspondência de Varreduras procuram então a diferença entre a posição P_n e a posição P_r. A maioria destes algoritmos precisam de um alinhamento inicial entre duas varreduras do sensor, que é fornecida pela leitura de odometria em muitos casos.



(a) (b) (c)

Figura 65: Posição Inicial e Final do robô móvel.

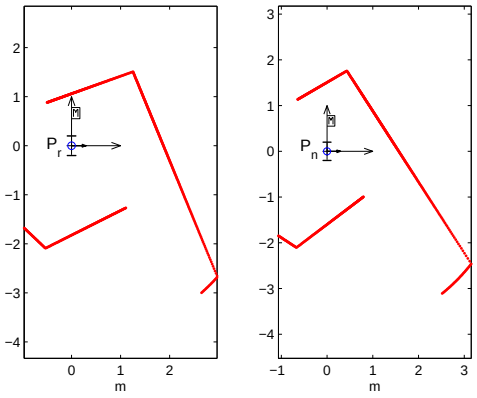


Figura 66: Varreduras geradas pelo sensor.

2.4.1 Algoritmo de Correspondência de Varreduras por NDT

As correspondências assumidas entre dois Scans de duas diferentes poses do robô móvel sempre apresentam erros. Biber [17] propõe uma representação alternativa para uma varredura, semelhante a uma grade de ocupação, subdividindo o plano $2D$ em grades. Para cada grade, se atribui uma distribuição normal, que localmente modela a probabilidade de medição de um ponto. *O re-*

sultado da transformação é uma parte contínua de uma densidade de probabilidade diferenciável [17]. Apresenta-se a seguir o algoritmo em detalhes e mostra-se uma aplicação para resolver o problema do SLAM.

2.4.1.1 Notação para o algoritmo

O algoritmo Correspondência de Varreduras requer dois conjuntos de leituras do LRF chamados Varreduras. $S_{ref} = \{q_1, q_2, ..., q_3\}$ representa um conjunto de n pontos que se reuniram no sistema de coordenada A, que é chamado **Varredura de referência**. $S_{atu} = \{p_1, p_2, ..., p_3\}$ representa um conjunto de m pontos que se reuniram no sistema de coordenada B, que é chamado **Varredura atual**.

O objetivo do algoritmo NDT é a de estimar o movimento do robô entre os sistemas de coordenadas A e B. x_B^A denota a estimação da posição relativa entre os sistemas de coordenadas A e B. Assim, x_B^A representa a Rotação e Translação no plano entre A e B. Uma suposição comum é que o erro na correspondência é Normal [17]. Assim, representamos a estimativa do algoritmo NDT como uma distribuição normal multivariável.

$$x_B^A = N(\hat{x}_B^A, P_B^A) \tag{2-13}$$

Aqui, $\hat{x}_B^A$ denota a média do vetor X que tem a forma $[\triangle x, \triangle y, \triangle \theta]$, onde $\triangle x$ e $\triangle y$ representam a translação e $\triangle \theta$ representa a rotação. Por conseguinte, a covariância P_B^A é uma matriz $3x3$.

Para lidar com os processos corrompidos por ruído Gaussiano, uma abordagem comum no mapeamento estocástico e SLAM é o uso do operador $\bigoplus$ (composição de transformações). Este operador é usado ao longo deste trabalho.

$S'_{atu} = \{p'_1, p'_2, ..., p'_3\}$ representa um conjunto de pontos projetados no sistema de coordenadas de S_{ref}. Isto é:

$$p'_i = x_B^A \oplus p_i, \qquad \forall p_i \in S_{atu}$$

O operador composição de transformações $\bigoplus$ representa a transformação espacial entre dois sistemas de coordenadas, que é dada por:

$$x_B^A \oplus p_i = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{2-14}$$

onde t_x e t_y descrevem a translação e ϕ a rotação entre as duas posições do robô móvel, (x,y) representa um ponto p_i do S_{atu}, (x',y') representa um ponto do S_{atu} projetado no sistema de coordenadas do S_{ref}, e

$$x_A^B = \begin{pmatrix} t_x \\ t_y \\ \phi \end{pmatrix} \tag{2-15}$$

A ideia do algoritmo NDT é encontrar o movimento do robô x_B^A que maximiza a sobreposição entre as partes do meio ambiente representadas por S_{ref} e S_{atu}.

No contexto do presente trabalho, a função de probabilidade pode ser definida como $f : \Re^2 \rightarrow \Re$, calculando a probabilidade de ter uma leitura de uma Varredura, no sistema de coordenadas $x-y$. Além disso, esta função é construída utilizando as leituras do LRF no S_{ref} , e não em mapas a priori.

É importante ressaltar que a função $f(x,y)$ não é uma FDP (Função Densidade de Probabilidade). No entanto, transformar $f(x,y)$ em uma FDP só envolve o uso de um fator de normalização η. Para entender completamente a utilidade de $f(x,y)$ no algoritmo de Correspondência de Varreduras, é definido a FDP $g(x,y)$ como:

$$g(x,y) = \eta f(x,y) \tag{2-16}$$

Mediante está função pode-se gerar o conjunto de pontos S'_{atu}, onde cada ponto $p'_i \in S'_{atu}$ foi gerado pelo desenho da FDP definida pela Equação (2-16). Sendo S'_{atu} o resultado da composição x_B^A com cada ponto de S_{atu}, a solução de Correspondência de Varreduras pode ser vista como o de encontrar x_B^A que torna a suposição do processo generativo verdade. De um ponto de vista probabilístico, isto pode ser expresso como o problema de maximizar a função de verossimilhança

$$\Psi(x) = \prod_{p_i \in S_{atu}} g(x \oplus p_i) \tag{2-17}$$

onde x representa a rototranslação entre os sistemas de coordenadas de S_{ref} e S_{atu}. A ideia por trás desta função (2-17) é projetar cada ponto da Varredura atual sobre a FDP g por meio da rototranslação x. Em seguida, a FDP é avaliada em cada um destes pontos projetados e os resultados são multiplicados. Uma boa rototranslação x vai projetar os pontos de S_{atu} sobre regiões de $G(x,y)$ com valores elevados (isto é, com elevada probabilidade de ter uma leitura do sensor). Assim, quanto melhor a rototranslação x, maiores os valores da função (2-17).

A rototranslação x que maximiza a Equação (2-17) constitui a estimativa de Correspondência de Varreduras x_B^A. Em consequência, a função representa a sobreposição entre os duas Varreduras.

Como maximizar a Equação (2-17) pode ser computacionalmente caro, uma abordagem usual é a utilização das funções *log-likelihood*. Assim, o problema de Correspondência de Varreduras pode ser expressado como a maximização da função *log-likelihood*

$$F(x) = log(\Psi(x))$$

$$F(x) = \sum_{p_i \in S_{atu}} log(g(x \oplus p_i))$$

$$F(x) = m\,log(\eta) + \sum_{p_i \in S_{atu}} log(f(x \oplus p_i)) \tag{2-18}$$

onde m é o número de pontos em S_{atu}, e η representa um valor constante que não vai influenciar no processo de maximização. Assim, não é necessário considerar η, simplificando para obter a função a ser maximizada

$$F(x) = \sum_{p_i \in S_{atu}} log(f(x \oplus p_i)) \tag{2-19}$$

2.4.1.2 Função de Otimização

O processo de otimização consiste em maximizar a Equação (2-19). Como os problemas de otimização são normalmente descritos como problemas de minimização, adotaremos nossa notação esta convenção. Assim, a função a ser minimizada é $-F(x)$, que chamaremos **Função Objetivo**:

$$score(x) = -\sum_{p_i \in S_{atu}} log(f(x \oplus p_i)) \tag{2-20}$$

2.4.1.3 Transformada de Distribuíções Normais (NDT)

A NDT modela a distribuição de todos os pontos $2D$ reconstruídos de uma Varredura dado pelo LRF, por um conjunto de distribuições normais locais. A NDT é construída a partir de $q_i \in S_{ref}$. Em primeiro lugar, o espaço $2D$ em torno do robô subdivide-se regularmente em N quadrículas com tamanho constante LxL. Em seguida, procura-se o conjunto de pontos de q_i dentro de cada quadrícula. Biber [17] e Burguera [31] propõem um valor de $L = 1m$. Dependendo da posição de origem da quadrícula no intervalo $[0, L)x[0, L)$, os

pontos de S_{ref} serão divididos em grupos diferentes. Vamos denotar por α uma posição particular da origem da quadrícula. Depois, para cada quadrícula $j = 1,2,3,...,N$, que contém pelo menos três pontos, é feito o seguinte:

1. Reúna o conjunto de n pontos $q_i \in S_{ref}$ contida nesta quadrícula em $\Omega_{\alpha,j}$.

2. Calcule a média $\mu_{\alpha,j}$ e a matriz de covariância $P_{\alpha,j}$ dos pontos $q_i \in \Omega_{\alpha,j}$:

$$\mu_{\alpha,j} = \frac{1}{n}\sum_i q_i \tag{2-21}$$

$$P_{\alpha,j} = \frac{1}{n}\sum_i (q_i - \mu_{\alpha,j})(q_i - \mu_{\alpha,j})^T \tag{2-22}$$

   para $i = 1,2,...,n$

3. Para evitar que a matriz de covariância $P_{\alpha,j}$ seja singular ou esteja próximo de ser singular, o menor autovalor de $P_{\alpha,j}$ é testado para ser pelo menos $0,001$ vezes o maior autovalor. Caso contrário, ele é definido para possuir este valor. O parâmetro $0,001$ foi experimentalmente ajustado em [17].

4. Modelar a probabilidade de ter uma leitura no ponto x contida na quadrícula j pela distribuição normal $N(\mu_{\alpha,j}, P_{\alpha,j})$. Deixando o fator de normalização η fora da FDP (2-16), a função de Probabilidade correspondente à quadrícula j é

$$f_{\alpha,j}(x) = e^{-\frac{(x-\mu_{\alpha,j})^T P_{\alpha,j}^{-1}(x-\mu_{\alpha,j})}{2}} \tag{2-23}$$

Assim, a função de probabilidade $f_\alpha(x)$ associada ao S_{ref} e à origem da quadrícula α é construída usando a Equação (2-23), como segue:

$$f_\alpha(x) = \begin{cases} f_{\alpha,1}(x), & \text{se x} \in \Omega_{\alpha,1}\ , \\ f_{\alpha,2}(x), & \text{se x} \in \Omega_{\alpha,2}, \\ ... & \\ f_{\alpha,N}(x), & \text{se x} \in \Omega_{\alpha,N}, \end{cases} \tag{2-24}$$

Para minimizar os efeitos da discretização, a abordagem original do NDT [17] propõe a utilização de quatro quadrículas que se sobrepõem, em vez de usar 1 quadrícula. Agora, cada ponto cai em quatro quadrículas. Assim, quatro funções de probabilidade são construídas, cada uma delas considerando uma posição particular de origem da quadrícula. Dado um ponto x, quatro valores de probabilidade, $f_1(x)$, $f_2(x)$, $f_3(x)$ e $f_4(x)$, estão disponíveis. Usando essa abordagem, para avaliar a função de probabilidade de x, as contribuições das quatro quadrículas são somadas

$$f(x) = \sum_{1 \le \alpha \le 4} f_\alpha(x)$$

Quando $f(x)$ é definido, o processo de otimização tem que ser realizado. De acordo com à Equação (2-20), a função a ser otimizada na abordagem NDT aplicando sobreposição deve ser

$$score(x) = -\sum_{p_i \in S_{atu}} log(\sum_{1 \leq \alpha \leq 4} f_\alpha(x \oplus p_i)) \tag{2-25}$$

No entanto, em vez de minimizar $score(x)$, tal como definido na Equação (2-25), a NDT original [17] minimiza a seguinte função, $h(x)$:

$$h(x) = -\sum_{p_i \in S_{atu}} \sum_{1 \leq \alpha \leq 4} f_\alpha(x \oplus p_i) \tag{2-26}$$

A principal vantagem de definir a função $h(x)$ como a função a ser otimizada, em vez da função $score(x)$ (2-25), é porque $h(x)$ torna o processo de otimização mais fácil e mais rápido. Além disso, Biber [32] mostra que $h(x)$ é uma boa aproximação de $score(x)$.

2.4.1.4 Esboço do Algoritmo Correspondência de Varreduras por NDT

O algoritmo de Correspondência de Varreduras por NDT pode ser esboçado pelas etapas:

1. Construir o NDT do S_{ref} para determinar a função de probabilidade (2-24) associada ao S_{ref}.

2. Estimar um valor inicial para as variáveis (t_x, t_y, ϕ) (nulo ou usando dados de odometria).

3. Para cada ponto $p_i = (x_i, y_i)$ do S_{atu}, determinar o ponto $p'_i = (x'_i, y'_i)$ que representa a transformação espacial $2D$ no sistema de coordenadas do S_{ref} mediante a Equação (2-14), de acordo com os parâmetros (t_x, t_y, ϕ).

4. Avaliar cada ponto p'_i com a função de probabilidade associada ao S_{ref}, e determinar a função de otimização (2-26).

5. Calcular uma estimativa nova para os parâmetros (t_x, t_y, ϕ), tentando otimizar a função de otimização (2-26).

6. Retornar à etapa 3 até que o critério de convergência seja verdade.

2.5 Inteligência Computacional

È um ramo da Ciência da Computação empregada, principalmente, na solução de problemas para os quais não existem procedimentos efeitivos capazes de solucioná-los satisfatoriamente. Problemas assim, geralmente podem ser modelados como tarefas de aprendizagem, percepção, previsão, adaptação ou evolução, e estas são as principais características presentes nas técnicas de Inteligência Computacional. Atualmente a Inteligência Computacional compreende um grande conjunto de técnicas, mas as de maior destaque são as Redes Neurais Artificiais, a Lógica Nebulosa e os Algoritmos Genéticos.

Os Algoritmos Genéticos (AG) pertencem ao grupo dos métodos evolutivos que são baseados em po´pulações de potenciais soluções de um problema. Métodos dessa natureza se mostram interessantes na resolução de problemas complexos de otimização porque conseguem um equilíbrio entre capacidade de exploração do espaço de soluções e também de aproveitamento das melhores soluçõesn ao longo da evolução.

2.6 Evolução Diferencial

Evolução Diferencial (ED) é um algoritmo evolutivo, derivado de um AG com seleção por torneio ,desenvolvido para otimização numérica global. ED tem sido aplicado a uma grande variedade de tarefas de otimização, frequentemente com grande sucesso [33]. Em termos mais simples, otimização é a tentativa de maximizar as propriedades desejáveis de um sistema ao mesmo tempo, e minimizar suas características indesejáveis.

Em geral, para otimizar certas propriedades do sistema através de parâmetros do mesmo, esses parâmetros são geralmente representados por um vetor. O primeiro passo de um problema de otimização começa por elaborar uma função objetivo, que pode modelar os objetivos do problema, incorporando quaisquer restrições. Como quase todos os algoritmos evolutivos, ED é um otimizador baseado na população que, como ponto de partida, ataca o problema por amostragem da função objetivo em N_P vezes, escolhendo aleatoriamente pontos iniciais.

Cada vetor é indexado por um número de 0 a $N_P - 1$. ED gera novos pontos que são perturbações de pontos existentes. Essa perturbação dos vetores é feita por uma diferença de dois vetores de populações selecionadas aleatoriamente.

Para produzir um vetor intermediário u_i, ED adiciona o vetor diferença a um terceiro vetor da população selecionado aleatoriamente. Na etapa de seleção, o vetor intermediário u_i compete com o vetor população do mesmo índice, que neste caso é o número i ($i = 0,1,2,...,N_P - 1$). O vetor com o menor valor da função objetivo é marcado como um membro da próxima geração. Os N_P sobreviventes das competições se tornam pais para a próxima geração do ciclo evolutivo.

2.6.1 Estrutura da População

A implementação mais versátil de ED mantém um par de populações de vetores, ambos os quais contêm N_P vetores D-dimensionais de parâmetros de valor real.

$$P_{x,g} = [\mathbf{x}_{i,g}], \qquad i = 0,1,2,...,N_P - 1, \quad g = 0,1,2,...,g_{max} \tag{2-27}$$

$$\mathbf{x}_{i,g} = [x_{j,i,g}], \qquad j = 0,1,2,...,D-1$$

onde o índice g indica a geração a que vetor pertence, $P_{x,g}$ é a população na geração g, a cada vetor é atribuído um índice de população i, e os parâmetros dentro de cada vetor são indexados com j.

Uma vez inicializado, é realizada a etapa de mutação dos vetores escolhidos aleatoriamente para produzir uma população intermediária:

$$P_{v,g} = [\mathbf{v}_{i,g}] \tag{2-28}$$

$$\mathbf{v}_{i,g} = [v_{j,i,g}]$$

Cada vetor na população atual é recombinado com a população intermediária para produzir uma população

$$P_{u,g} = [\mathbf{u}_{i,g}] \tag{2-29}$$

$$\mathbf{u}_{i,g} = [u_{j,i,g}]$$

2.6.2 Etapas do Algoritmo de ED

2.6.2.1 Inicialização

Antes que a população seja inicializada, os limites superiores e inferiores para cada parâmetro devem ser especificados como o vetor (b_L, b_U), onde b_L e

b_U indicam os limites inferior e superior, respectivamente. Assim, um gerador de números aleatórios atribui a cada parâmetro de cada vetor um valor dentro da faixa prevista. Por exemplo, o valor inicial ($g = 0$) do parâmetro j do vetor população de ordem i é:

$$\mathbf{x}_{j,i,0} = rand_j(0,1) * (b_{j,U} - b_{j,L}) + b_{j,L} \tag{2-30}$$

2.6.2.2 Mutação

Uma vez inicializado, ED faz a mutação e recombina a população para produzir uma população de N_P vetores intermediários

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{2-31}$$

onde F é o fator de escala que controla a taxa em que a população evolui $F \in [0,1]$ [33]. Os índices $r1$ e $r2$ são também escolhidos aleatoriamente, uma vez por mutação, o índice $r0$, pode ser determinada numa variedade de formas (aleatoriamente, maior valor da função objetivo).

2.6.2.3 Crossover

ED também emprega crossover uniforme, às vezes referido como recombinação discreta. Em particular, ED cruza cada vetor com um vetor mutante:

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g}, & \text{Se } (rand_j(0,1) \leq Cr || j = j_{rand}) \\ x_{j,i,g}, & \text{outro caso} \end{cases} \tag{2-32}$$

A probabilidade de crossover, $Cr \in [0,1]$, é um valor definido pelo usuário que controla a fração dos valores atribuídos a cada parâmetro que são copiados a partir do mutante.

2.6.2.4 Seleção

Se o vetor intermediário $u_{i,g}$ tem um valor de função objetivo igual ou menor do que o objetivo de seu vetor do qual herda parâmetros, $x_{i,g}$, ele substitui o vetor na seguinte geração, caso contrário o vetor mantém o seu lugar na população pelo menos por mais uma geração.

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{Se } (f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g})) \\ \mathbf{x}_{i,g}, & \text{outro caso} \end{cases} \tag{2-33}$$

Uma vez que a nova população esteja calculada, o processo de recombinação, mutação e seleção é repetido até que um ótimo seja atingido, ou um critério de parada pré-especificado seja satisfeito, por exemplo, o número de gerações atingindo um máximo pré-definido g_{max} [33]. A Figura 67 mostra o processo do algoritmo de evolução diferencial.

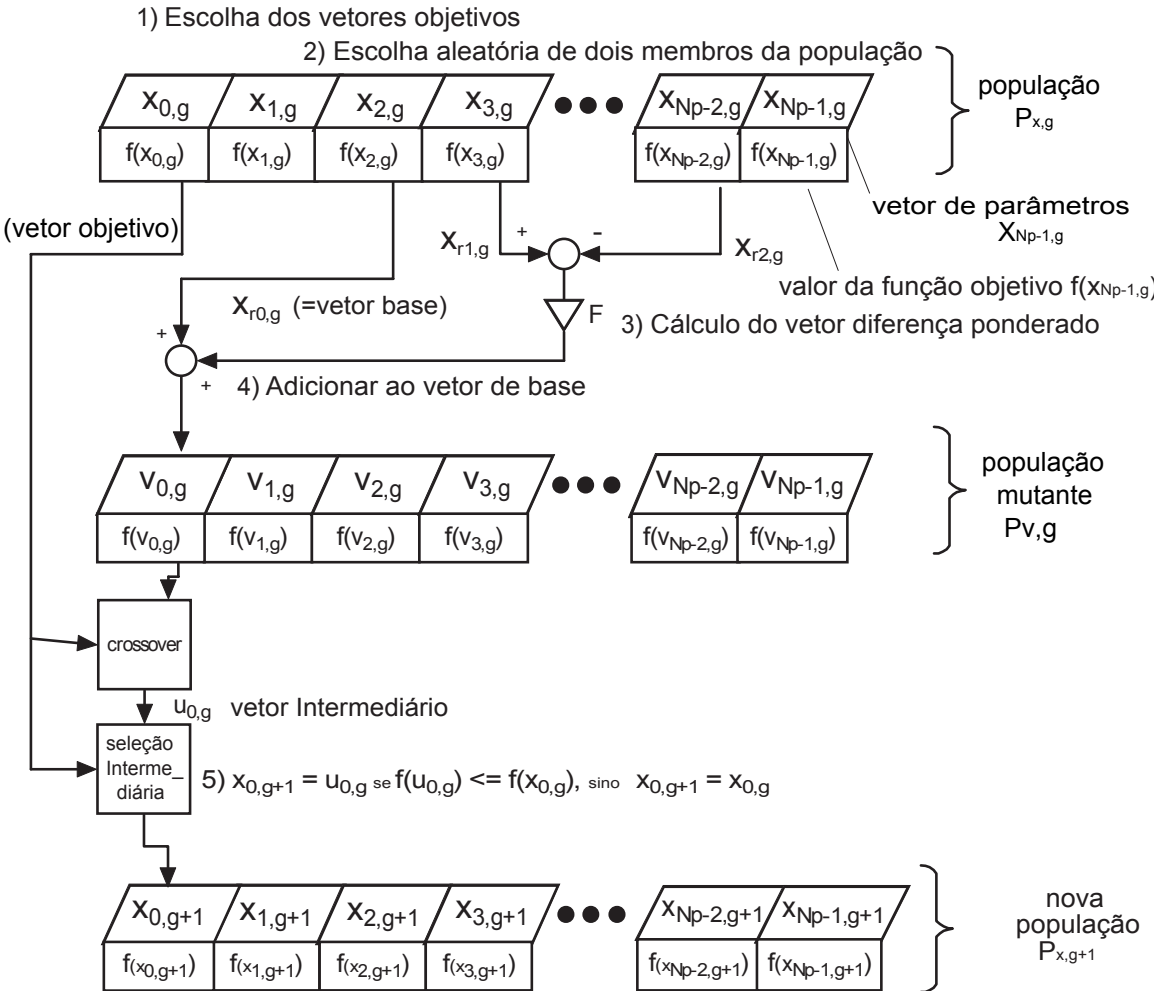


Figura 68: Processo de Evolução Diferencial

2.7 DP-SLAM

O algoritmo de DP-SLAM fornece uma solução precisa para resolver o problema de SLAM, mantendo de forma eficaz uma distribuição conjunta através de mapas e as posições do robô móvel usando filtro de partículas, fornece mapas precisos [21]. Quando se utiliza um filtro de partículas para resolver o problema de SLAM, cada partícula corresponde a uma trajetória específica através do ambiente, e possui um mapa específico associado a ele. Quando uma partícula é reamostrada, o mapa inteiro é tratado como parte do estado oculto que está

sendo monitorado.

Para que um filtro de partículas acompanhe corretamente a distribuição conjunta sobre a posição do robô móvel e o mapa, é necessário para cada partícula manter um mapa separado. Durante a fase de reamostragem do filtro de partículas, cada partícula pode ser reamostrada várias vezes. Por conseguinte, é necessário copiar o mapa inteiro para cada nova partícula, para manter cada hipótese do conjunto de atualizações do mapa. Esta distribuição conjunta, e a correspondente necessidade de múltiplas hipóteses de mapas, é ponto de partida crucial para o processo de SLAM [4].

DP-SLAM usa uma representação eficiente para fazer cópias de mapas, ao mesmo tempo reduzindo a memória total exigida para representar este grande número de grades de ocupação. Isto é conseguido através de um método de mapeamento chamado partículas distribuídas (PD-Mapping), o qual explora as redundâncias significativas entre os diferentes mapas. Além disso, é capaz de manter e atualizar centenas de mapas candidatos e posições do robô móvel de uma forma eficiente [22].

2.7.1 Filtro de Partículas

O filtro de partículas (FP) é uma implementação alternativa não paramétrica do filtro de bayes. Ele não depende de uma forma fixa funcional da distribuição de probabilidade, tal como as gaussianas. Filtros de partículas aproximam a distribuição por um número finito de valores.

A ideia principal do FP é representar uma função de distribuição posterior $p(x_t)$ por um conjunto aleatório de amostras retiradas a partir desta distribuição [1]. A Figura 69 mostra essa idéia para uma distribuição Gaussiana. Em vez de representar a distribuição de uma forma paramétrica, o FP representa uma distribuição de um conjunto de amostras desta Gaussiana. À medida que o número de amostras vai para o infinito, o FP tende a convergir uniformemente para a distribuição correta. Em FP, as amostras da distribuição $p(x_t)$ são chamadas de *partículas*. Assim, a distribuição de $p(x_t)$ é representada por M partículas ponderadas:

$$\Phi_t := \{\langle x_t^i, w_t^i \rangle / i = 1, 2, ..., M\} \tag{2-34}$$

A intuição por trás de filtros de partículas é aproximar a distribuição $p(x_t|z^t, u^t)$ pelo conjunto de partículas Φ_t. Assim a aproximação feita por FP é dada por:

$$p(x_t|z^t, u^t) \approx \Phi_t \tag{2-35}$$

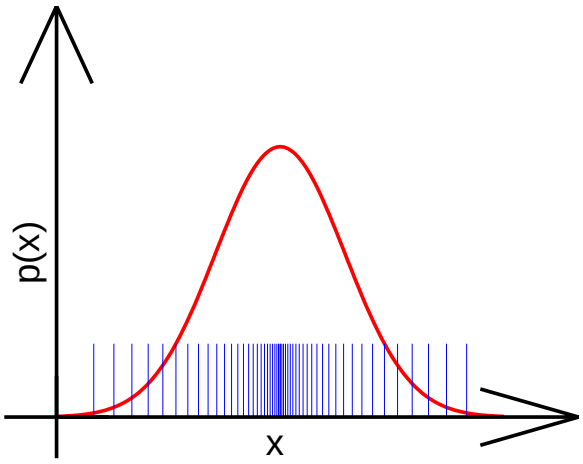


Figura 70: Representação da Gaussiana por um conjunto de partículas

O algoritmo de FP constrói a distribuição $p(x_t|z^t, u^t)$ recursivamente a partir da distribuição $p(x_{t-1}|z^{t-1}, u^{t-1})$ de um tempo anterior. Isto significa que os FP podem construir um conjunto de partículas Φ_t recursivamente a partir do conjunto de partículas Φ_{t-1} [1].

A variante mais básica do algoritmo FP é indicada na Tabela 2.1. A entrada deste algoritmo é o conjunto de partículas Φ_{t-1}, junto com o controle mais recente u_t com a mais recente medição z_t.

2.7.2 Representação do Mapa

DP-SLAM utiliza uma representação para o mapa do tipo Grade de Ocupação, mais especificamente por mapas estocásticos, onde cada quadrado da grade tem uma escala de vários graus de ocupação.

DP-SLAM concentra-se em um modelo de LRF. Com as características do LRF um método mais adequado para representar a incerteza do mapa, que leva em conta a distância que o laser viaja através de cada quadrado da grade. O pressuposto básico é que cada quadrado da grade responde à luz do laser de uma forma que é independente dos quadrados vizinhos. Além disso, o comportamento não depende do ângulo de incidência ou a parte do quadrado que é atingido. O comportamento em relação a um quadrado particular depende apenas da distância que o laser percorre até o quadrado.

Tabela 2.1: Algoritmo de Filtro de Partículas [1]

Algoritmo $Particle\_filter(\Phi_{t-1}, u_t, z_t)$	
1:	$\overline{\Phi}_t = \Phi_t = 0$
2:	for $i = 1$ to M do
3:	sorteie $x_t^i \sim p(x_t \mid u_t, x_{t-1}^i)$
4:	$w_t^i = p(z_t \mid x_t^i)$
5:	$\overline{\Phi}_t = \overline{\Phi}_t + \langle x_t^i, w_t^i \rangle$
6:	end for
7:	for $i = 1$ to M do
8:	sorteie i com probabilidade $\propto w_t^i$
9:	some x_t^i a Φ_t
10:	end for
11:	retornar Φ_t

O modelo corresponde à suposição de que, entre as varreduras, as obstruções em cada quadrado da grade são redistribuídas de maneira uniforme dentro do quadrado. Na representação por Grade de Ocupação com ocupação parcial, cada grade é um potencial obstáculo [2].

Seja $P_c(x,\rho)$ a probabilidade cumulativa do laser ter sido interrompido depois de viajar à distância x através de um meio de tipo ρ. Esta condição de consistência pode ser indicada de modo mais geral, em termos de k divisões como [21]:

$$P_c(x,\rho) = \sum_{i=1}^{k} P_c(\frac{x}{k},\rho)(1 - P_c(\frac{x}{k},\rho))^{i-1} \tag{2-36}$$

Esta soma representa que o laser pode ser interrompido durante todo o segmento de comprimento $\frac{x}{k}$, acumulando a probabilidade de parada em cada um. Dentro da soma, o primeiro termo é a probabilidade de que a varredura será obstruído no segmento dado. O segundo termo representa a probabilidade de cada segmento anterior não obstruir ao laser.

A distribuição exponencial $P_c(x,\rho) = 1 - e^{-\frac{x}{\rho}}$, para um escalar positivo ρ, satisfaz essa condição de consistência, onde ρ refere como a opacidade do quadrado da grade.

A probabilidade de um raio inteiro do laser ter sido parado em algum ponto

ao longo de sua trajetória é, portanto, a probabilidade cumulativa que o raio do laser seja interrompido por n grades.

$$P(parada = verdade) = P_c(X, \boldsymbol{\rho}) = \sum_{i=1}^{k} P_c(x_i, \rho_i) \prod_{j=1}^{i-1} (1 - P_c(x_j, \rho_j)) \quad (2\text{-}37)$$

onde $X = (x_1, x_2, ..., x_n)$ é o vetor de distâncias percorridas através das grades e $\boldsymbol{\rho} = (\rho_1, \rho_2, ..., \rho_n)$ são as opacidades correspondentes daquelas grades.

Assim, a probabilidade de que o raio do laser será interrompido em uma grade quadrada j é $P(parada = j)$, que é calculada como a probabilidade de que o laser atingiu o quadrado $j-1$ e, em seguida, parou em j, logo

$$P(parada = j) = P_c(x_j, \rho_j)(1 - P_c(X_{1:j-1}, \boldsymbol{\rho}_{1:j-1})) \quad (2\text{-}38)$$

onde $X_{1:j-1}$ e $\boldsymbol{\rho}_{1:j-1}$ têm a interpretação natural como fragmentos dos vetores X e $\boldsymbol{\rho}$.

Suponha que $\boldsymbol{\delta} = (\delta_1, \delta_2, ..., \delta_n)$ é um vetor, tais que δ_i é a diferença entre o comprimento total da medição do laser e o comprimento até a grade i . Eliazar [21] define a probabilidade condicional de medição, dado que o raio do laser foi interrompido no quadrado i, como: $P_L(\delta_i | parada = i)$, para o qual a literatura faz uma suposição típica, de medição do ruído normalmente distribuído [34]. Observe que os termos δ_i são definidos apenas se a medição do laser observa um ponto específico de parada.

O evento $(\delta_i, parada = i)$ forma uma partição de todo possível evento de parada do laser. Assim, a probabilidade de medição segundo o teorema da probabilidade total é a soma, sobre todas as quadrículas do alcance do laser, do produto da probabilidade condicional da medição dado que o raio tenha parado, e a probabilidade de que o raio parou em cada grade.

$$P(L, parado = verdade) = \sum_{i=1}^{n} P_L(\delta_i | parada = i) P(parada = i) \quad (2\text{-}39)$$

2.7.3 Atualização do Mapa

A média da distribuição exponencial com opacidade ρ é simplesmente ρ, o que faz a atualização do mapa particularmente simples. Para cada grade, basta manter a soma das distâncias totais d_t viajadas por um laser da varredura através da grade. Também é preciso manter registro do número de vezes que se presume que o laser tenha parado na grade h. Segundo [21], a estimativa de ρ é representada por:

$$\hat{\rho} = \frac{d_t}{h} \tag{2-40}$$

Grades que não tenham sido observadas por qualquer antepassado de uma partícula em consideração são tratadas de uma maneira especial, porque é difícil de estimar a probabilidade de parada do laser da Equaçaõ (2-38) sem experiência prévia. Eliazar propõe em [21] inicializar a probabilidade das grades desconhecidas como:

$$P_c(x,\rho) = 0 \tag{2-41}$$

O traço da linha do laser é continuado após o ponto observado até uma distância de 6σ, onde σ^2 é a variância do modelo de ruído do LRF. Isto permite ter a certeza de que qualquer ponto razoável na trajetória do laser é permitido ser uma possível fonte de observação [21]. Nos casos em que o laser viajar através quadrículas anteriormente não observadas, a probabilidade total calculada de que o laser será interrompido, dada a trajetória e o mapa, pode ser inferior a um. Neste caso, cada área do mapa desconhecida é dado o benefício da dúvida.

2.7.4 PD-Mapping

A maior contribuição da DP-SLAM é uma representação eficiente do mapa, ao mesmo tempo reduzindo a memória total exigida para representar um grande número de grades de ocupação. Isto é conseguido através de um método chamado de mapeamento de partículas distribuídas (PD-Mapping), o qual explora as redundâncias significativas entre os diferentes mapas.

Cada mapa possui muito em comum com a maioria dos outros mapas, e reproduzindo toda essa informação várias vezes é um uso ineficiente dos recursos de memória. Para tornar este conceito mais claro, vamos introduzir a noção de descendência de partículas.

Partícula Pai É uma partícula amostrada a partir da distribuição no tempo $t-1$, que produz partículas novas no tempo t.

Partículas Filhos São a geração de partículas no tempo t, que foram produzidas por uma partícula pai no tempo $t-1$

Partículas Irmãs São duas partículas com o mesmo pai.

Para ver como isso pode ser útil, suponha que um raio laser do LRF varre uma área de tamanho $A << M$ (onde M é o tamanho do mapa), e considere dois irmãos S_1 e S_2. Cada irmão corresponderá a uma posição diferente do robô e cada um fará atualizações ao mapa que herda de seu pai, máximo uma área do tamanho A. Assim, os mapas para S_1 e S_2 podem diferir do seu pai no máximo

em uma área de tamanho A. Todo o restante do mapa será idêntico.

Para que um raio do laser procure um obstáculo no mapa, seria necessário trabalhar com toda a descendência da partícula atual, e consultar uma lista de diferenças armazenada para cada partícula. A complexidade desta operação seria linear ao número de iterações do filtro de partículas. O desafio é, por conseguinte, fornecer as estruturas de dados que permitem atualizações eficientes para o mapa de localização e consultas eficientes com complexidade de tempo que seja independente do número de iterações do filtro de partículas. O conceito básico da árvore de descendência de partículas é detalhado em [4] e [2].

2.7.5 SLAM usando PD-Map

Acessando uma grade particular em um PD-map é um pouco mais complicado do que em uma grade de ocupação padrão. Uma vez que cada grade quadrada contém todo um conjunto de observações, descobrir se uma partícula específica fez uma observação neste local requer uma busca através deste conjunto de observações. No entanto, cada partícula herda as atualizações do mapa inserido por seus antepassados. Por isso, precisamos dar um passo para trás através da descendência da partícula, comparando cada ancestral com o conjunto de observações a este quadrado de grade.

Fazer uma atualização no mapa é um processo mais simples do que acessá-lo. Quando uma nova observação é feita, o valor atualizado é adicionado ao conjunto de observações do quadrado de grade. Ao mesmo tempo, a partícula ancestral que causou a atualização mantém um ponteiro para a observação específica no mapa.

Exclusões de entradas para o mapa são realizadas apenas quando uma partícula antepassada é removida. Usando a lista de atualizações de mapas para esta partícula ancestral, que foi criada quando as atualizações de mapas foram inicialmente adicionadas, podemos facilmente remover todas as observações pertinentes.

No próximo capítulo, os detalhes da implementação dos algoritmos no sistema experimental são apresentados.

3
Implementação dos Algoritmos no Sistema Experimental

3.1
SLAM

3.1.1
Filtragem das Varreduras

Um problema crucial na *Correspondência de Varreduras* é como selecionar pontos das Varreduras que são úteis para a correspondência. Por que uma Varredura inclui centenas de pontos, muitos deles podem ser redundantes e afetar a eficiência e precisão dos algoritmos, em especial quando a forma do meio ambiente é simples. Além disso, o valor da Equação (2-26) é influenciado pela densidade das leituras. Regiões com maior densidade de leitura são produzidos quando o robô móvel está perto de uma parede, assim a Equação (2-26) resulta em valores mais elevados nestas regiões, veja a Figura 72. O algoritmo de Correspondência de Varreduras pode dar resultados inadequados de sobreposição, como na Figura 74. Para superar esta situação, este trabalho utiliza dois métodos, uma reamostragem uniforme utilizada por Burguera [31] e Ynoquio [2], e um reamostragem destacando as características mais relevantes (*Reamostragem baseada em Saliências* [35].

3.1.1.1
Reamostragem Uniforme

Este filtro é capaz de criar uma nova amostra da Varredura com uma perda mínima de informação e custo computacional baixo. A ideia por trás deste filtro é mover uma janela circular sobre a Varredura e substituir os pontos dentro da janela com seu centro de gravidade. A Figura 78 exemplifica o efeito do passo de reamostragem. O Filtro tem o efeito de alisamento da distribuição de pontos ao longo da Varredura e também reduz o número de pontos da Varredura, sem perder muita informação.

O raio da janela define a distância mínima entre os pontos na Varredura filtrada, ver Figura78 (b). Este raio tem de ser definido experimentalmente. Os

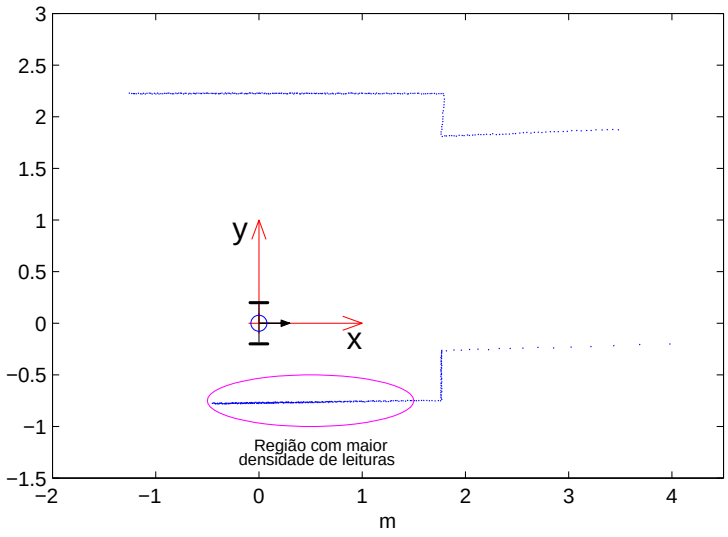


Figura 74: Regiões de alta densidade, produzida por um robô móvel situado perto da parede

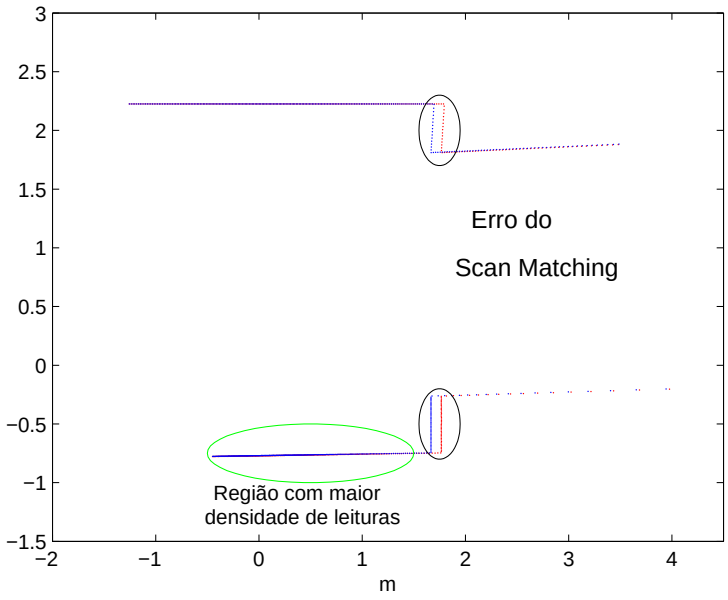
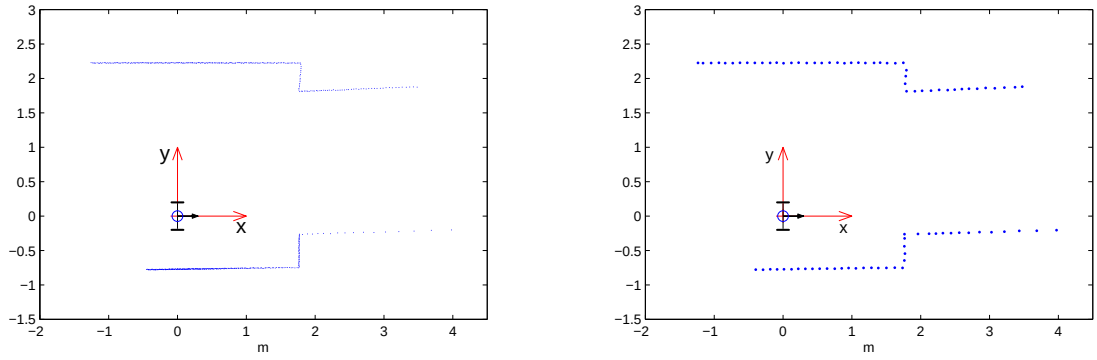


Figura 76: Erro na sobreposição das Varreduras (círculos pretos), a primeira Varredura (pontos vermelhos) e a segunda Varredura (pontos azuis) apresentam regiões de alta densidade (círculo verde) de pontos

valores baixos deste parâmetro não resolvem a influência da densidade das leituras, enquanto um valor elevado pode causar perda de informações da Varredura. Burguera[31] experimentalmente testou o parâmetro a um valor de $5cm$ para o sensor utilizado. Mas, neste trabalho, este parâmetro é definido a $10cm$, devido ao tipo de sensor adotado, menos preciso.



(a) (b)

Figura 80: Reamostragem Uniforme da Varredura. (a) Varredura original com 594 pontos (b) Varredura depois da reamostragem com 85 pontos.

3.1.1.2 Reamostragem Baseada em Saliências

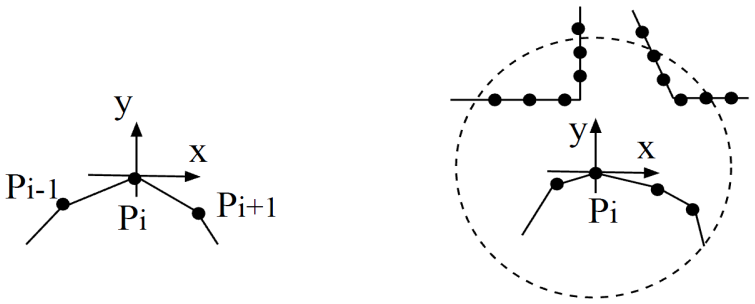
O método calcula a *Saliência* de cada ponto da Varredura de acordo com a quantidade de informação que tem o ponto, ou o comprimento do segmento de linha em que se encontra o ponto. Com base na *Saliência*, o método determina se cada ponto da Varredura deve ser reamostrado ou não. Pontos do Varredura redundantes são descartados e pontos que apresentam características relevantes são preservados, desse modo a eficiência e precisão do algoritmo de Correspondência de Varreduras é melhorada.

Reamostragem baseada em saliências não usa os recursos explícitos como linha e canto. Em vez disso, utiliza um ponto da Varredura com a sua direção tangente, que é referido como *Ponto Direcionado*, como é mostrado na Figura81(a). O sistema de coordenada local de um *Ponto Direcionado* é tal que a origem está localizada no ponto da Varredura e o eixo x é a tangente do ponto indicado [30]. Também define-se a *Assinatura* de um *Ponto Direcionado* P_i como um conjunto P_j de outros *Pontos Direcionados* em torno P_i. Transformando a coordenada de cada P_j para o sistema de coordenada local de P_i, é possível fazer a *Assinatura* invariante à translação e rotação de P_i.

Para comprimir a *Assinatura*, aplica-se a transformação de Hough para cada P_j da *Assinatura* e projeta-se no espaço de Hough (ρ, θ):

$$\rho(\theta) = x_o cos(\theta) + y_o sin(\theta) \tag{3-1}$$

Isto gera um padrão no espaço de Hough. Em seguida, uma partição do espaço



(a) (b)

Figura 82: Representação da *Saliência*

de Hough em quadrículas é obtida para gerar uma tabela, Se selecionam os pontos Hough que são máximos locais em termos do número de ocorrências. O tamanho da quadrícula aqui adotada é $5[cm]$ x $5[graus]$ [30]. Assim a *Assinatura* de P_i é definida como:

$$G_i = \{(q_{ij}, w_{ij}) / 1 \leq j \leq N_i\} \tag{3-2}$$

onde q_{ij} é um ponto Hough (ρ_{ij}, θ_{ij}), w_{ij} é o número de votos de q_{ij} e N_i é o número de valores máximos locais.

O método define a $Salincia\ L_i$ de um *Ponto Direcionado* P_i com base na quantidade de informação da sua *Assinatura* G_i.

$$L_i = -\sum_{j}^{N_i} log(P(q_{ij})) \tag{3-3}$$

onde $-log(P(q_{ij}))$ é a quantidade de informação q_{ij}, e $P(q_{ij})$ é a probabilidade que q_{ij} aparece em uma *Assinatura*.

A Equação (3-3) fornece uma boa medida para a *Saliência*, mas consome tempo de computação, porque as *Assinaturas* de todos os *Pontos Direcionados* devem ser calculadas antes da reamostragem deles. Para evitar isto, a transformação de Hough é aplicada a cada P_i da Varredura, está transformação cria uma tabela, através de uma aproximação para a *Saliência* definida como:

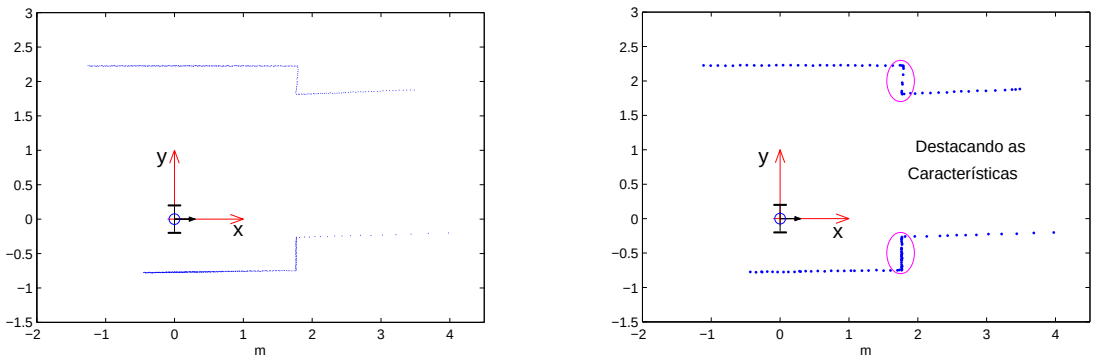
$$L_i = \{\frac{u_i}{N}\}^{-1} \tag{3-4}$$

onde u_i denota o número de votos da quadrícula da tabela de Hough que corresponde a um *Ponto Direcionado* P_i, e N é o número total de pontos P_i da Varredura.

A reamostragem baseada em Saliências consiste em:

- Reamostrar os pontos da Varredura uniformemente com um intervalo de $10cm$ (Reamostragem Uniforme).
- Reamostrar um ponto da Varredura que tem uma *Saliência* L_i elevada.

A Figura 83 mostra os resultados para uma Varredura adquirida em um corredor. O número de pontos da Varredura é 594. A reamostragem reduz o número de pontos para 134. Como pode ser observado, todos os pontos em torno das quinas são preservados, e os pontos nas paredes são redefinidos quase uniformemente.



(a) (b)

Figura 85: Reamostragem baseada em saliências. (a) Varredura original com 594 pontos (b) Varredura depois do reamostragem com 134 pontos.

3.2 Algoritmo de Correspondência de Varreduras

O algoritmo de Correspondência de Varreduras é importante para realizar um mapeamento e localização do robô móvel utilizando um LRF sem sensores de odometria. Como a informação fornecida pelo LRF é composta de Varreduras em diferentes posições do robô móvel dentro do ambiente, com esta informação podemos substituir o sensor de odometria.

Assim, para substituir um sensor de odometria, usam-se duas Varreduras consecutivas consecutivos para fazer uma sobreposição deles usando o algoritmo de Correspondência de Varreduras. O resultado desta sobreposição é o deslocamento entre duas Varreduras consecutivas. Se o LRF é montado no centro do robô móvel, o deslocamento é igual ao deslocamento do robô móvel.

O algoritmo de Correspondência de Varreduras utilizado neste trabalho foi descrito no capítulo 2. O algoritmo de Correspondência de Varreduras por NDT fornece uma boa função para representar uma sobreposição de duas leituras consecutivas do LRF pela Equação (2-26). O algoritmo NDT original utiliza como método de minimização o algoritmo de Newton [17]. Mas, neste trabalho, como método de otimização é usado evolução diferencial ED [2].

O algoritmo de evolução diferencial é utilizado no processo de otimização para tornar o algoritmo de Correspondência de Varreduras mais robusto. A optimização através de algoritmos genéticos foi detalhada no capítulo 2.

3.3 Parâmetros e Considerações

3.3.1 O Sensor

Talvez o mais conhecido LRF em robótica é o LRF da família SICK [2]. Mas o LRF utilizado neste trabalho é da família URG, um sensor com uma faixa de medição muito menor que os sensores da família do SICK.

O modelo utilizado é $URG-04LX-UG01$. A Tabela 3.1 mostra as suas características mais importantes. Uma característica importante a considerar

Tabela 3.1: Características do sensor LRF modelo $URG-04LX-UG01$.

Especificações	
Faixa de Detecção	$20mm$ a $4000mm$
Área de Varredura	240^o
Frequência de Varredura	$10Hz$ $(100msec/scan)$
Resolução	$1mm$
Precisão	$20mm-1000mm$: $\pm 30mm$ $20mm-4000mm$: $\pm 3\%$ da medição
Resolução Angular	0.36^o

é a velocidade máxima linear e angular do robô móvel, a fim de garantir que seu movimento não afeta a leitura do LRF. As Equações (3-5) e (3-6) calculam a velocidade máxima mediante uma função da frequência da varredura do sensor.

$$v_{max} = \xi_t \cdot f_s \tag{3-5}$$

$$w_{max} = \xi_r \cdot f_s \tag{3-6}$$

onde f_s é a Frequência da varredura, e ξ_r e ξ_t são o erro máximo introduzido pelo movimento do robô. Assim, por exemplo, para erros $\xi_r = 1^o$ e $\xi_t = 10mm$, as velocidades máximas do robô móvel precisam ser:

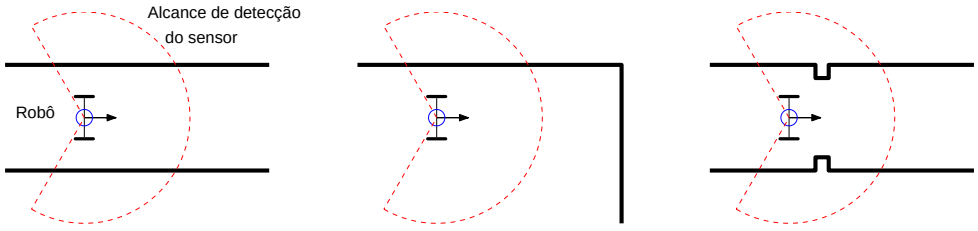
$$v_{max} = 0.1m/s$$

$$w_{max} = 0.18rad/s$$

3.3.2 O Ambiente

A faixa de medição do sensor utilizado é de até $4m$, implicando que o ambiente a ser mapeado precisa possuir variedade de características numa seção inferior a $4m$. Por exemplo, longos corredores com paredes paralelas, ou lugares espaçosos cujas paredes estão fora da faixa de detecção do LRF, produzem varreduras sucessivas que o algoritmo de Correspondência de Varreduras resolve erradamente, pois a correspondência consome muito tempo por causa de sua ambiguidade. Em particular, se o S_{ref} é constituído por uma ou duas linhas paralelas, infinitas possibilidades seriam obtidas. Essas varreduras são comuns em um ambiente real. A Figura 87 mostra exemplos de ambiguidade; nos casos (a) e (b) a exata correspondência é impossível na teoria.

Em alguns casos, varreduras tem partes salientes pequenas, como mostra a Figura 87 (c), fornecendo poucas informações para fazer a correspondência. Para lidar com este problema, é necessário utilizar filtros que destaquem pontos com características distintas e que descartem pontos redundantes.



(a) Paredes paralelas (b) Parede única (c) Paredes com pilares

Figura 88: Exemplos de degeneração.

3.3.3 Robô Móvel

Os experimentos utilizaram o robô *iRobot Create* [36] para os testes de SLAM. Esse modelo de robô (ver Figura 89) é didático e amplamente utilizado por estudantes e pesquisadores. O *iRobot Create* é um dispositivo que recebe comandos pela porta serial e os executa, ou também pode ser controlado remotamente. Ao robô é acoplado a LRF modelo $URG-04LX-UG01$ e um

Figura 90: O Robô *iRobot Create* acoplado a um LRF ($URG-04LX-UG01$)

microcomputador portátil da classe netbook, modelo Asus Eee PC 1000HEB, com processador Intel Aton N280 de $1,66Ghz$, responsável por enviar comandos de movimentos e capturar dados do LRF.

Aqui, listamos algumas características importantes que justificam o uso do *iRobot Create* nos testes:

- Possui amplo espaço para instalação de hardware adicional.
- Pode usar um cabo serial para enviar comandos a partir de um PC.
- Uma série de protocolos totalmente documentados, proporciona acesso total aos sensores, atuadores e funcionalidade do robô móvel.
- Autonomia de 90 a 120 minutos por carga, dependendo do piso. Carregamento da bateria independente.
- Movimento de translação e rotação independente.

- A velocidade que o robô pode atingir é entre -500 e 500 mm/s.

Uma das características do robô é ser capaz de receber comandos para realizar seu movimento a partir de um computador. Além disso, Microsoft Robotics Developer Studio tem bibliotecas especializadas para o *iRobot Create*. A assim, mediante a linguagem de programação VPL (Visual Program Lenguage), é possivel criar um painel de controle para seu movimento, ver Figura 91. O painel

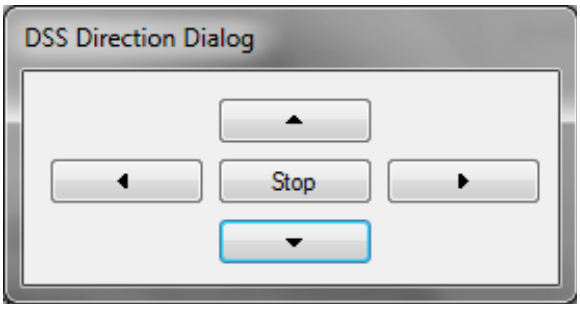


Figura 92: Controle Mediante Microsoft Robotics Developer Studio

de controle de movimento (Figura 91), envia o movimento a ser realizado pelo *iRobot Create*: seguir em frente, para trás, vire à esquerda ou à direita. A Figura 94 mostra o esquema de programação em VLP (Microsoft Robotics Developer Studio). Nos experimentos, os comandos são enviados por um teleoperador, ou

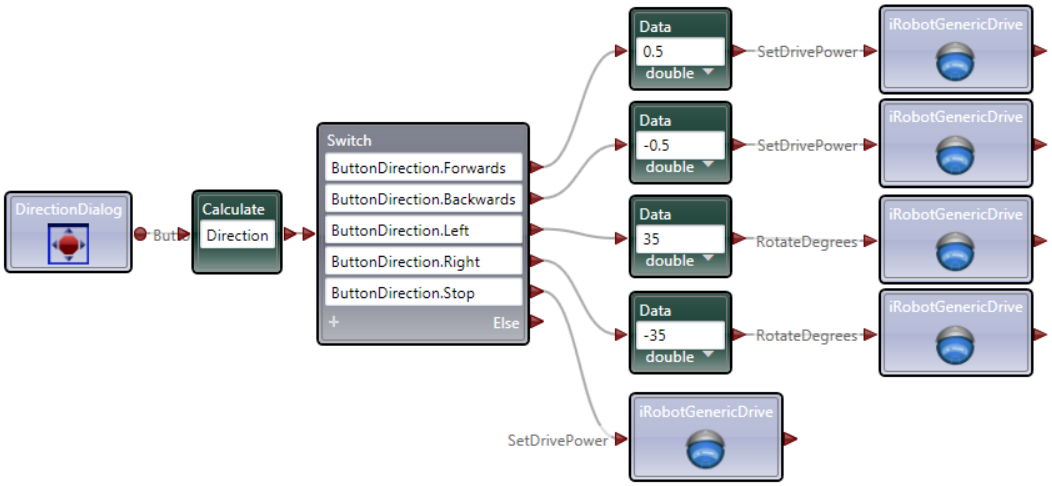


Figura 95: Esquema do Controle em Microsoft Robotics Developer Studio

pré-programados no microcomputador. Note, no entanto, que os algoritmos de SLAM implementados não fazem uso destes comandos, gerando os mapas e localizando o robô apenas a partir de varreduras sucessivas pelo LRF.

No próximo capítulo, algumas simulações são apresentadas.

4
Resultados da Simulação

Algumas simulações foram feitas para testar os algoritmos realizados neste trabalho, e são apresentados neste capítulo. Os resultados da simulação são divididos em : análise de "Correspondência de Varreduras"de dados obtidos mediante a simulação do movimento do robô móvel, do sensor LRF e do ambiente; a solução do problema de SLAM mediante Correspondência de Varreduras; e a solução do problema de SLAM usando o algoritmo DP-SLAM.

4.1
Correspondência de Varreduras

O ambiente de simulação, onde o robô móvel faz o seu movimento, foi idealizado para avaliar alguns parâmetros relevantes para o processo de otimização, tais como o tamanho de grade para o algoritmo NDT, o número de iterações, e o tamanho de população, importantes no processo de otimização mediante Evolução Diferencial (Algoritmos Genéticos).

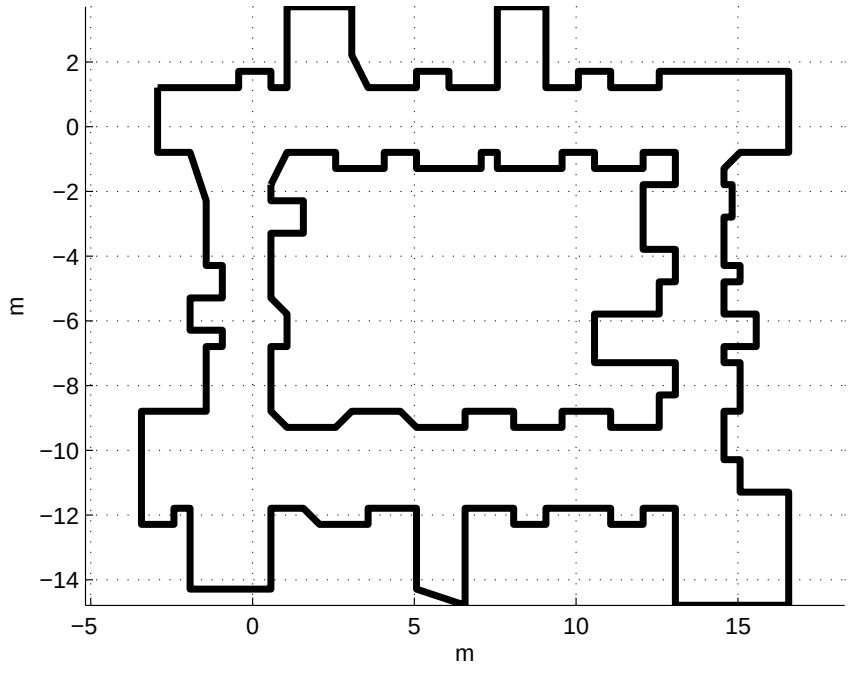
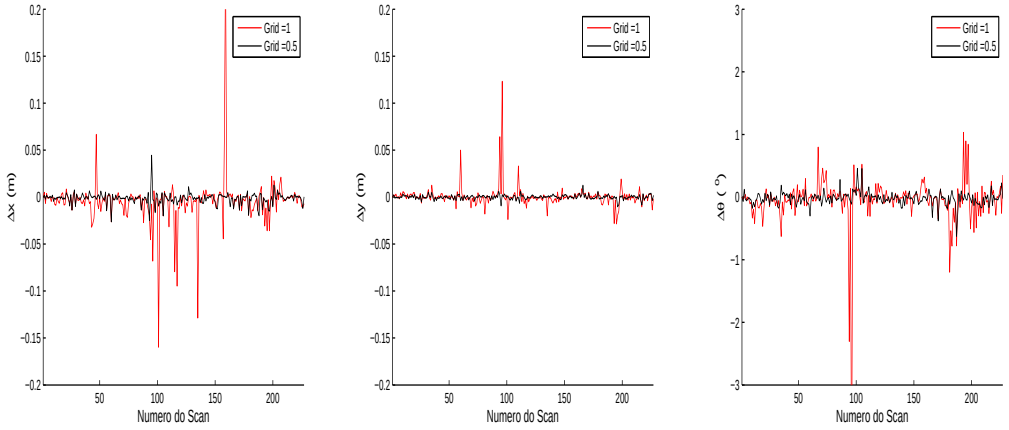


Figura 96: Ambiente Simulado

Para um ambiente simulado mostrado na Figura 95, durante o movimento do robô móvel dentro do ambiente foram tomadas 227 leituras com o LRF.

Uma vantagem de realizar simulações é que a trajetória do robô móvel dentro do ambiente é conhecida. Assim os valores dos parâmetros do processo de otimização $(\Delta x, \Delta y, \Delta\theta)$ são perfeitamente conhecidos e comparados com os obtidos mediante o algoritmo de Correspondência de Varreduras.

É importante a análise do tamanho de grade pois o algoritmo NDT, que aproxima uma função objetivo entre duas varreduras sucessivas do sensor, dela depende, como esta explicado no Capítulo 2. Assim, analisou-se o erro para uma grade $1,0m$ e de $0,5m$. A Figura 101 apresenta o erro dos parâmetros $(\Delta x, \Delta y, \Delta\theta)$, estimados pelo algoritmo de Correspondência de Varreduras.

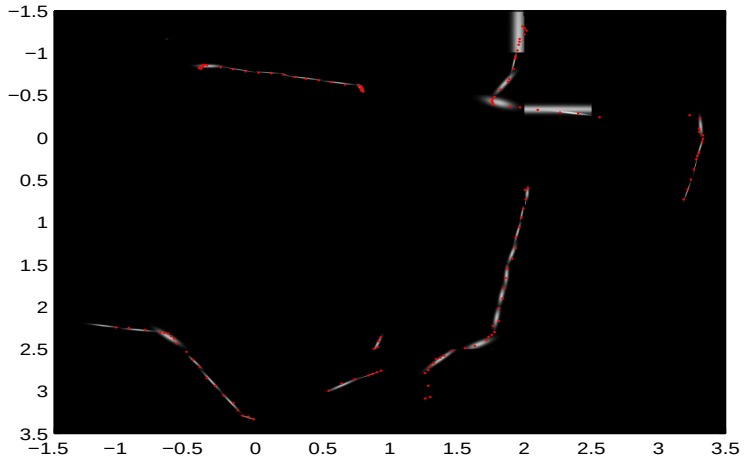


99(a): Erro x 100(b): Erro y 101(c): Erro θ

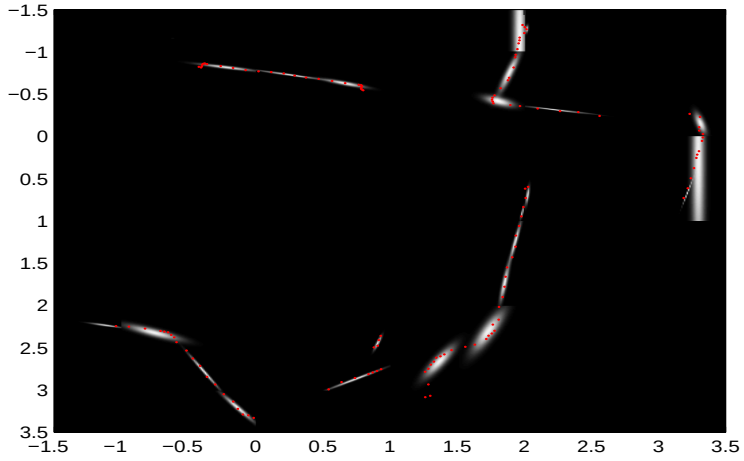
Figura 103: Erro da Correspondência de Varreduras para duas tamanhos de grade $(0,5m; 1,0m)$

Uma das características do sensor $URG-04LX-UG01$ é sua alta densidade de pontos (cada varredura possui 682 pontos), o que facilita considerar um tamanho de grade pequena, porque existe uma maior probabilidade de haver mais de 3 pontos da Varredura dentro da grade. Mas o custo computacional é maior, por otutro lado, um tamanho de grade alto piora a qualidade da função estimada. A Figure 106 mostra que em (a) a região branca é a melhor aproximação da Varredura, mas para um tamanho de grade maior $1,0m$ aparecem algumas descontinuidades, como se mostra em (b). Para os demais testes e simulações neste trabalho, o tamanho de grade é fixado em $0,5m$.

Numa otimização por Algoritmos Genéticos (Evolução Difetencial), descrito no Capítulo 2, a definição dos operadores genéticos é fundamental para a



105(a): Grade 0,5m



106(b): Grade 1,0m

Figura 107: Algoritmo NDT para duas resoluções de Grade

qualidade do processo evolutivo. Uma vez definida a representação cromossônica, é necessario estabelecer operadores que estejam de acordo tanto com a representação escolhida quanto con as características do problema de otimização.

O algoritmo de Evolução Diferencial depende dos operadores genéticos tais como o tamanho da população, número de iterações, taxa do crossover e taxa de mutação. A taxa de crossover utilizada neste é de $CR = 0,95$ e a taxa de mutação ultilizada $SF = 1,0$, de acordo com o trabalho prévio[2], estes valores foram testados aqui. Nas figuras a seguir 118-141 apresentamos o comportamento do erro dos parâmetros $(\Delta x, \Delta y, \Delta\theta)$ para diferentes números de populações e iterações.

Assim, mantendo o número de iterações em 50, para um tamanho da grade de $0,5m$, mas variando o tamanho da população, o erro dos parâmetros do algoritmo de Correspondência de Varreduras é analisado. A Figura 118 mostra o erro para uma população de 30 (vermelho) e uma população de 40 (azul). A Figura 122 mostra o erro para uma polulação de 50 (vermelho) e uma população de 60 (azul). E a Figura 126 mostra o erro para uma população de 70 (vermelho) e uma população de 100 (azul).

Na Figura 118 podemos observar que, para valores pequenos de população, o processo de otimização por evolução diferencial não converge totalmente, mas, à medida que aumenta o tamanho da população, melhora a convergência. A Figura 122 apresenta uma convergência em quase todos os pontos, e para valores altos de população a convergência é estabilizada, ver Figura 126.

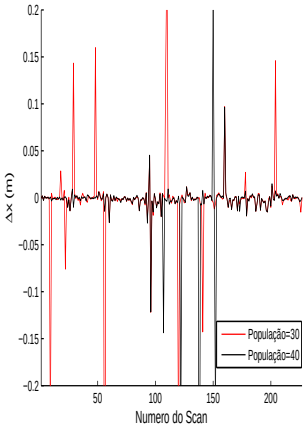
Outro parâmetro a considerar no processo de otimização por Evolução Diferencial é o número de iterações. Assim varia-se o número de iterações, mas para um tamanho da grade $0.5m$ e tamanho da população mantido constante igual a 100. As Figuras 133, 137 e 141 apresentam os erros dos parâmetros $(\Delta x, \Delta y, \Delta\theta)$, estimados pelo algoritmo de Correspondência de Varreduras.

A Figura 133 apresenta o erro para um número iterações baixas, no qual se pode observar que a convergência não é muito boa. Mas, para um número de iterações maior, a convergência melhora, ver Figuras 137 e 141. Observe que, para valores muito grandes de iterações, a convergência não varia muito. Mas consomem tempo de processamento, e portanto o número de iterações é fixado em 50 para os demais testes. A Tabela 4.1 apresenta os valores dos parâmetros a serem utilizados com dados reais.

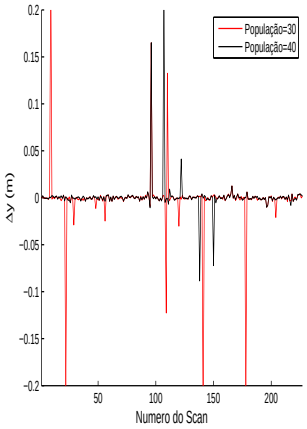
Tabela 4.1: Parâmetros a serem utilizados nos experimentos.

Tabela de Parâmetros

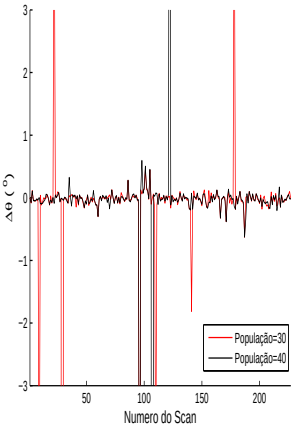
Algoritmo	Parâmetro	Valor
Evolução Diferencial	Mutação	1,0
	Crossover	0,95
	População	100
	Gerações	50
NDT	Grade	0,5



116(a): Erro x

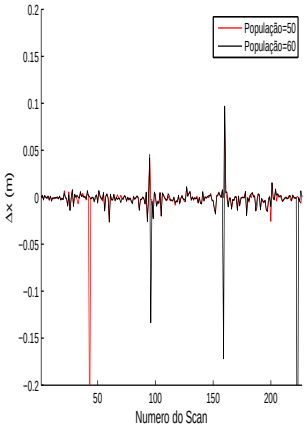


117(b): Erro y

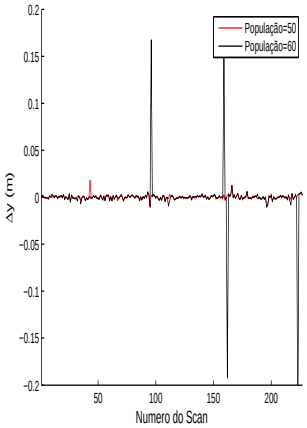


118(c): Erro θ

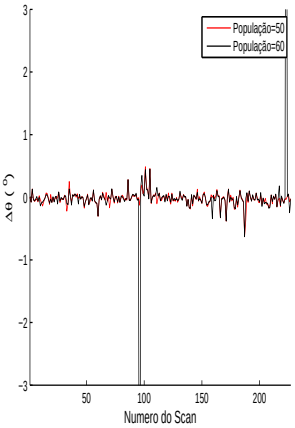
Figura 119: Erro do deslocamento, $(\Delta x, \Delta y, \Delta\theta)$. Pequenas Populações.



120(a): Erro x

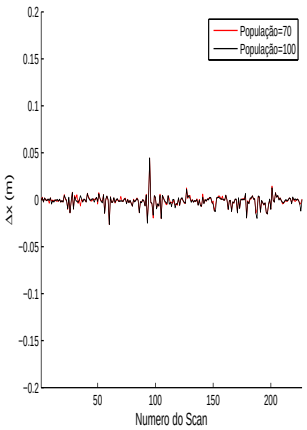


121(b): Erro y

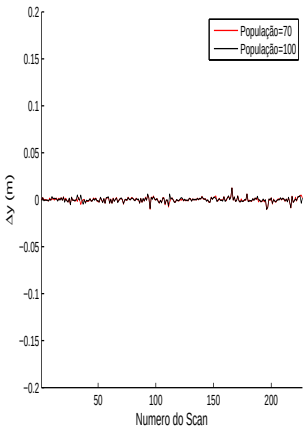


122(c): Erro θ

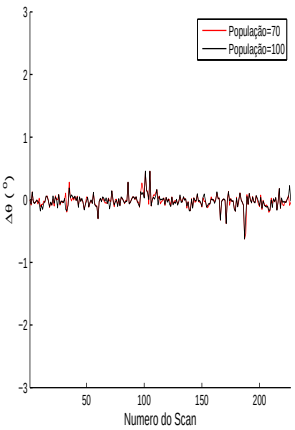
Figura 123: Erro do deslocamento, $(\Delta x, \Delta y, \Delta\theta)$. Populações Médias.



124(a): Erro x

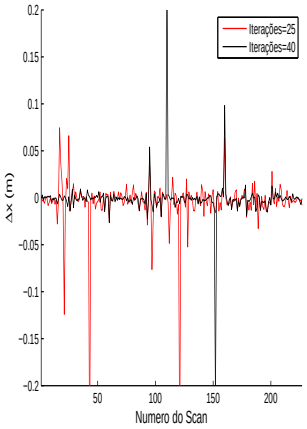


125(b): Erro y

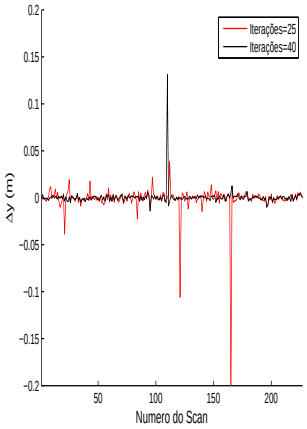


126(c): Erro θ

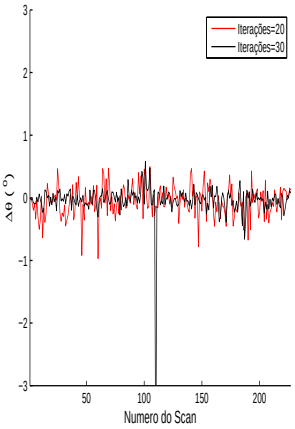
Figura 127: Erro do deslocamento, $(\Delta x, \Delta y, \Delta\theta)$. Grandes populações.



131(a): Erro x

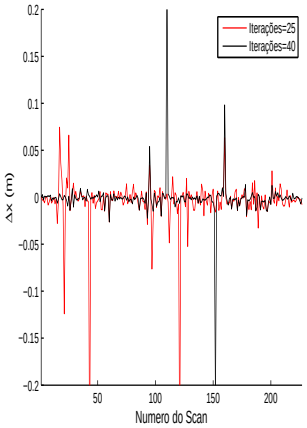


132(b): Erro y

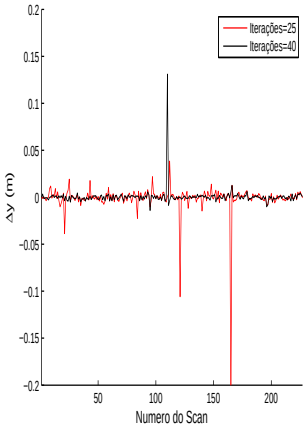


133(c): Erro θ

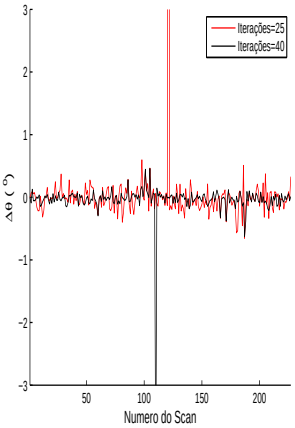
Figura 134: Erro do deslocamento, $(\Delta x, \Delta y, \Delta\theta)$. Iterações 20 e 30.



135(a): Erro x

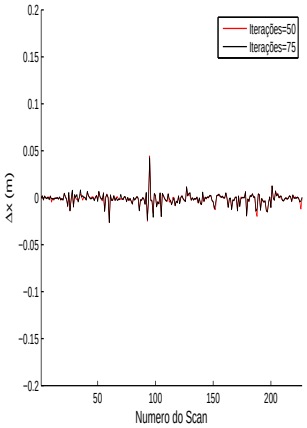


136(b): Erro y

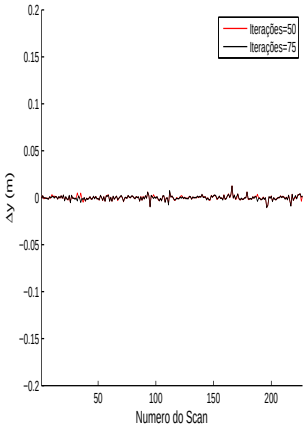


137(c): Erro θ

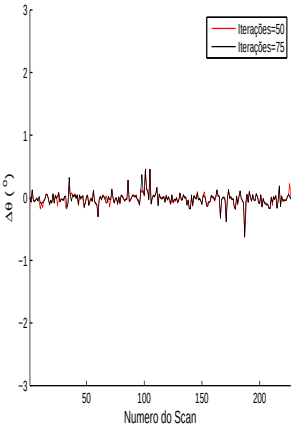
Figura 138: Erro do deslocamento, $(\Delta x, \Delta y, \Delta\theta)$. Iterações 25 e 40.



139(a): Erro x



140(b): Erro y



141(c): Erro θ

Figura 142: Erro do deslocamento, $(\Delta x, \Delta y, \Delta\theta)$. Iterações 50 e 75.

4.2 SLAM mediante Correspondência de Varreduras

O Algoritmo de Correspondência de Varreduras pode ser aplicado para resolver o problema de SLAM. Foi mostrado que é possível obter o deslocamento $(\Delta x, \Delta y, \Delta\theta)$ de cada leitura do sensor. Executando uma transformação de coordenadas para todas as leituras do sensor, de acordo com o deslocamento de duas Varreduras sucessivas, pode-se obter a trajetória e o mapa do robô móvel.

Assim, o mapa adquirido pelo algoritmo de Correspondência de Varreduras é apresentado na Figura 145. Note que o erro de cada Correspondência de Varreduras é cumulativo, portanto no momento de fechar o mapa emum circuito fechado pode ocorrer um desalinhamento. A Figura 145 mostra o mapa criado por Correspondência de Varreduras e o desalinhamento ao final de sua trajetória. Deve-se notar que a estrutura principal do ambiente é mantida, apesar do erro.

Da mesma maneira como o mapa foi criado por Correspondência de Varreduras, a trajetória do robô móvel é apresentada na Figura 151, Uma vez que o

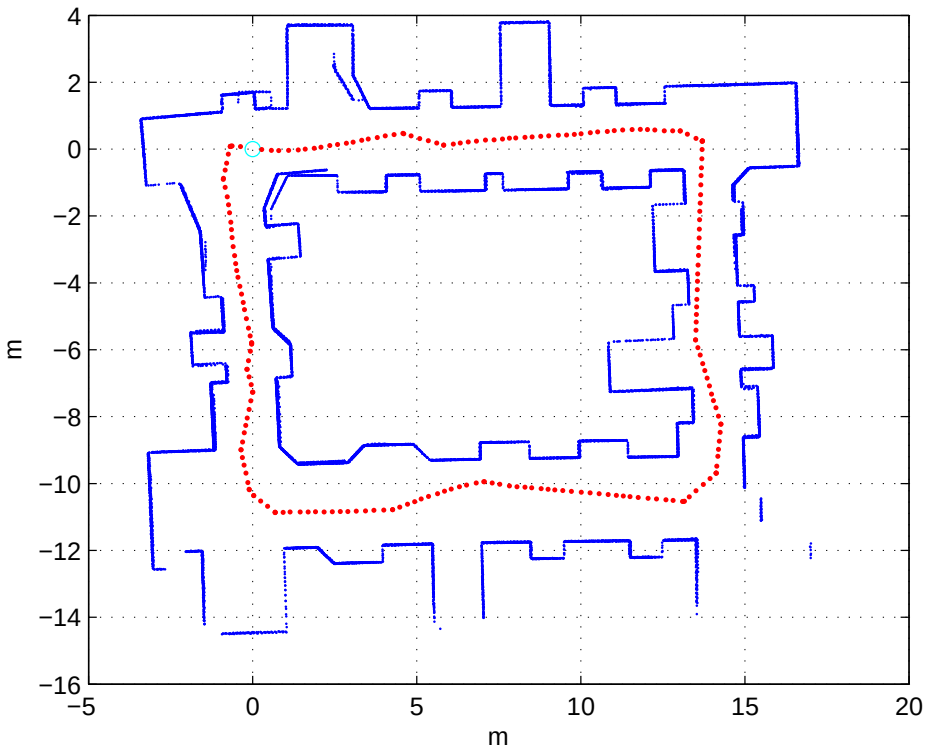


Figura 147: Mapeamento do Ambiente Simulado mediante Correspondência de Varreduras

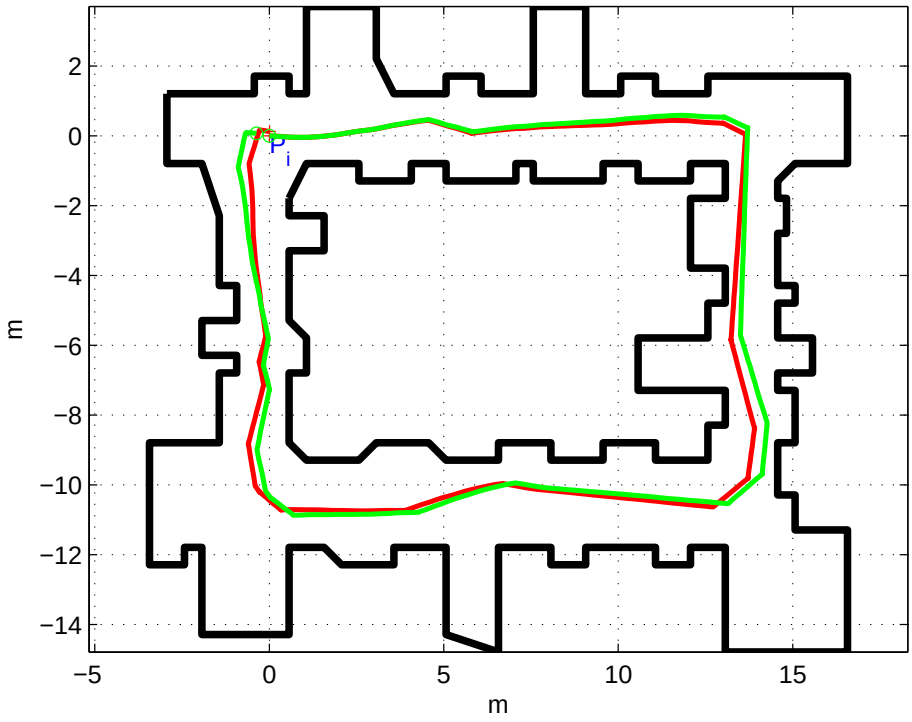


Figura 153: Trajetória do Ambiente Simulado. Trajetória real(vermelho) e Trajetória Estimada (verde)

mapa foi criado por software, a trajetória original é conhecida e pode ser comparada com a trajetória estimada pelo algoritmo de Correspondência de Varreduras. A linha vermelha representa a trajetória real do robô móvel, enquanto a trajetória estimada é mostrada pela linha verde. No início do movimento, como o erro acumulado é pequeno, as duas trajetórias são praticamente iguais, mas com o passar do tempo, o erro acumulado aumenta e a trajetória estimada difere da original, ver Figura 151.

4.3 DP-SLAM

SLAM mediante o algoritmo de Correspondência de Varreduras apresenta uma aproximação do mapa e da trajetória do ambiente. Para melhorar o mapa criado, neste trabalho se utiliza o algoritmo DP-SLAM, que fornece uma representação de grade de ocupação para o mapa. Além disso, DP-SLAM é usado para reduzir os erros produzidos pelo algoritmo de Correspondência de Varreduras.

Para implementar o algoritmo DP-SLAM é preciso conhecer o modelo matemático do movimento do robô móvel. Mas um dos objetivos deste trabalho é utilizar um único sensor (LRF) para obter o mapa e a localização do robô mó-

vel nele, sem uso de odometria ou outros sensores, muito menos as informações dos comandos enviados ao robô ou seu modelo. Então, para encontrar o deslocamento do robô móvel em cada instante de tempo, em vez de usar sensores como odômetros são utilizados os resultados do algoritmo de Correspondência de Varreduras para implementar um modelo de movimento.

4.3.1 Modelo de Movimento

Desenvolvendo um modelo matemático para o deslocamento do robô móvel, para uma posição (R_{t-1}) no instante $t-1$, dado um deslocamento

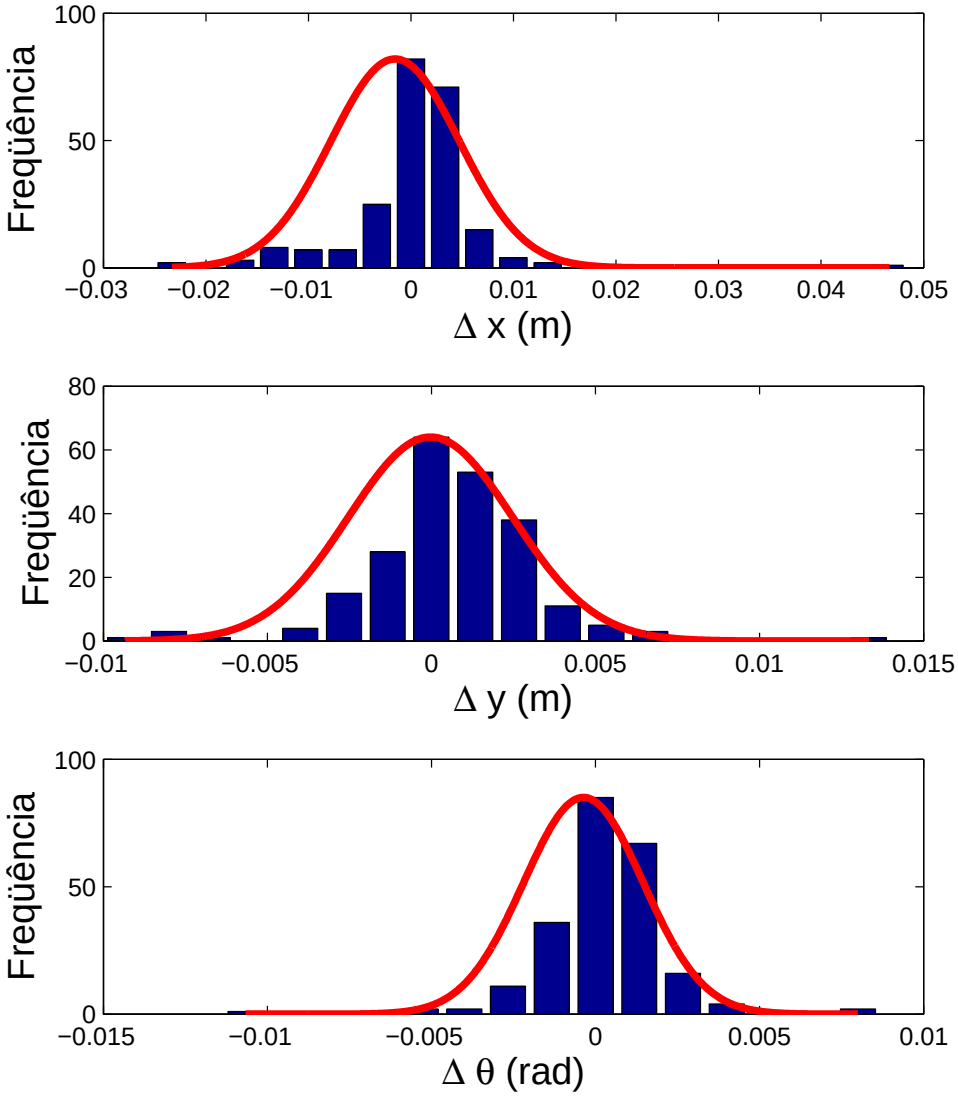


Figura 154: Distribuição Normal através do histograma do erro ($\Delta x, \Delta y, \Delta\theta$)

$(\Delta x, \Delta y, \Delta\theta)$ entre o instante $t-1$ e t, a posição (R_t), no instante t, é dada por:

$$R_t = R_{t-1} + \begin{pmatrix} d\,cos(\alpha) \\ d\,sin(\alpha) \\ \Delta\theta \end{pmatrix}$$

onde: $\alpha = R_{\theta_{t-1}} + tan^{-1}(\frac{\Delta x}{\Delta y})$ e $d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$.

Os deslocamentos $(\Delta x, \Delta y, \Delta\theta)$ apresentam um erro, o qual deve ser introduzido ao modelo de movimento, assim com a análise de dados simulados feita, e calculando o histograma do erro; a partir deste, pode-ser criar uma distribuição normal para o erro de cada variável do deslocamento Δx, Δy e $\Delta\theta$. A Figura 153 mostra un exemplo a distribuição normal aproximada do erro.

Para os dados da Figura 153, a distribuição do erro em Δx pode ser aproximada por uma distribuição gaussiana com média $-0,0016m$ e um desvio padrão de $0,02m$, da mesma forma o erro em Δy pode ser aproximado por uma distribuição gaussiana com média zero e um desvio padrão de $0,01m$; e em $\Delta\theta$ por uma distribuição gaussiana com média zero e um desvio padrão de $0,008rad$.

Tabela 4.2: Modelo do Movimento de Correspondência de Varreduras [2].

$Algoritmo\_Sample\_Motion(R_{t-1}, u_t)$		
1:	$\Delta\hat{x} =$	$\Delta x + Sortear(0.02 + abs(\frac{\Delta x}{10}))$
2:	$\Delta\hat{y} =$	$\Delta y + Sortear(0.01 + abs(\frac{\Delta y}{10}))$
3:	$\Delta\hat{\theta} =$	$\Delta\theta + Sortear(0.008 + abs(\frac{\Delta\theta}{10}))$
4:	$d =$	$\sqrt{(\Delta\hat{x})^2 + (\Delta\hat{y})^2}$
5:	$\alpha =$	$R_{\theta_{t-1}} + atan2(\Delta\hat{y}, \Delta\hat{x})$
6:	$R_{x_t} =$	$R_{x_{t-1}} + d\,cos(\alpha)$
7:	$R_{y_t} =$	$R_{y_{t-1}} + d\,sin(\alpha)$
8:	$R_{\theta_t} =$	$R_{\theta_{t-1}} + \Delta\hat{\theta}$
9:	Retornar	$R_t = (R_{x_t}, R_{y_t}, R_{\theta_t})$

As médias e os desvios pradão destes três movimentos são usados no modelo de movimento. Embora o modelo de movimento tenha sido em informações de dados simulados, este modelo será estendido para trabalhar com

Tabela 4.3: Algoritmo Aproximado de uma Distribuição Normal

Sortear: $Normal\_Distribution(v)$
1: Retornar $\frac{1}{6}\sum_{i=1}^{12} rand(-v,v)$

dados reais.

O modelo de movimento será capaz de amostrar o deslocamento do robô móvel. O modelo de movimento é mostrado na Tabela 4.2, onde a função $Sortear$ retorna um valor aleatório aproximadamente normal, segundo a função descrita na Tabela 4.1, e $atan2()$ é a função arco-tangente tal que $a = atan2(b,c) \Rightarrow sin(a) = b$ e $cos(a) = c$.

4.4 Mapeamento e Localização usando DP-SLAM

O algoritmo DP-SLAM fornece um mapeamento de maior qualidade para um ambiente. Assim com a adição do modelo de movimento, descrita acima e as leituras do sensor LRF, nesta seção apresentamos 2D mapas, para o ambiente simulado.

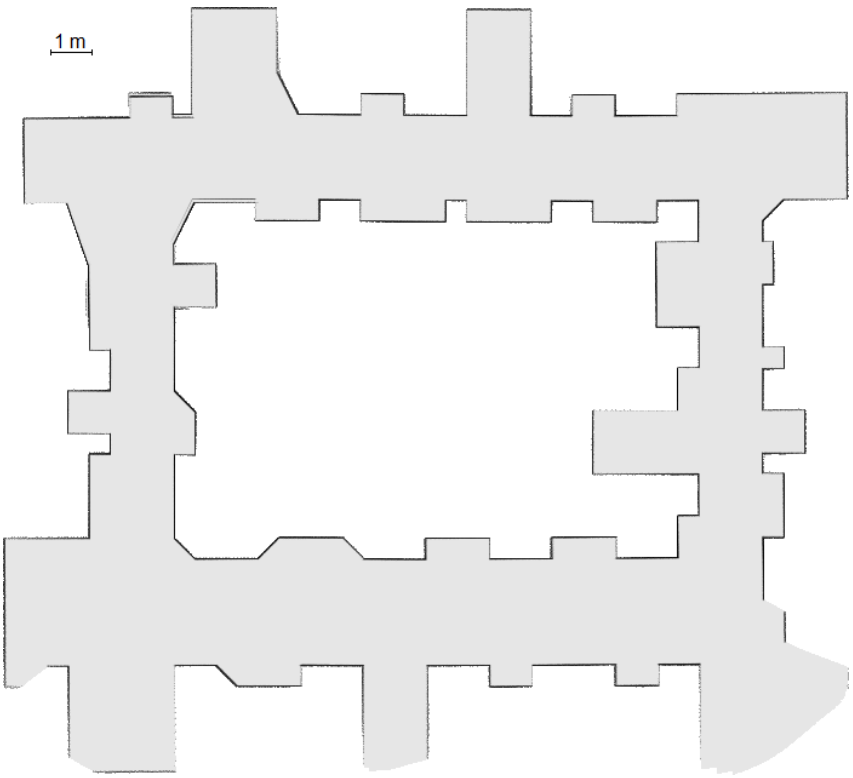


Figura 158: Mapa $2D$ do ambiente simulado obtido usando o algoritmo DP-SLAM

A Figura 156 mostra o mapa $2D$ do ambiente simulado. A grade de ocupação utilizada para representar o mapa tem uma resolução equivalente a $1m = 35grades$, onde os espaços pretos indicam que está ocupado e os espaços cinza indicam que está livre, e os espaços em branco são lugares que se encontram fora do alcance do sensor durante a experiência.

Como pode ser visto na Figura 156, o mapa obtido por DP-SLAM mostra um mapa de maior qualidade, em comparação com aquele criado pelo algoritmo de Correspondência de Varreduras. O mapa apresenta um pequeno desalinhamento, muito menor do que aquele mapa da Figura 145.

Os parâmetros do algoritmo DP-SLAM usados para todos os experímentos realizados estão na Tabela 4.4, estes parâmetros são recomendadas em [4].

Tabela 4.4: Parâmetros DP-SLAM dos experimentos.

Parâmetro	Valor
Resolução	$1m = 35grids$
Número de Partículas LSLAM	500
Número de Partículas HSLAM	500

No próximo capítulo, os experimentos realizados e seus resultados são apresentados.

5
Resultados Experimentais

Os resultados obtidos neste trabalho são apresentados neste capítulo. Para o desenvolvimento deste, foi utilizado um robô móvel (*"iRobot Create"*) e um único sensor LRF ($URG-04LX-UG01$), para obter informação da localização do robô móvel e do mapa por onde este realiza seu movimento, o robô adquire dados do sensor a cada $0,2s$ em ambientes reais. Os resultados são divididos em : análise de "Correspondência de Varreduras"de dados reais obtidos com un LRF; a solução do problema de SLAM mediante Correspondência de Varreduras; e a solução do problema de SLAM usando o algoritmo DP-SLAM.

5.1
Correspondência de Varreduras

Um dos objetivos deste trabalho é a utilização de apenas um sensor (LRF) para determinar a localização do robô móvel e o mapa do ambiente. O algoritmo de Correspondência de Varreduras calcula as estimativas dos parâmetros $(\Delta x, \Delta y, \Delta\theta)$, mas o resultado precisa de um sensor de movimento para ser comparado numericamente.

Nas Figuras 164-176 mostra duas varreduras consecutivas do sensor nas duas primeiras colunas e a terceira coluna a sobreposição delas. Para leituras do sensor em que partes onde os contornos do ambiente apresenta alguma variedade, tais como colunas, portas, corredores perpendiculares ou outras características, o erro da sobreposição é baixo, mas para ambientes com paredes paralelas sem muitas partes diferenciadas, o erro é maior, como pode ser visto na Figura 174.

A convergência do processo de otimização através de Evolução Diferencial foi definida como a minimização da função objetivo, ver Equação (2-26), mas para uma boa correspondência de duas leituras, o valor desta função deve ser próxima de zero. Para essa avalação o desempenho do algoritmo de Correspondência de Varreduras é analisado para dados reais em 7 experimentos.

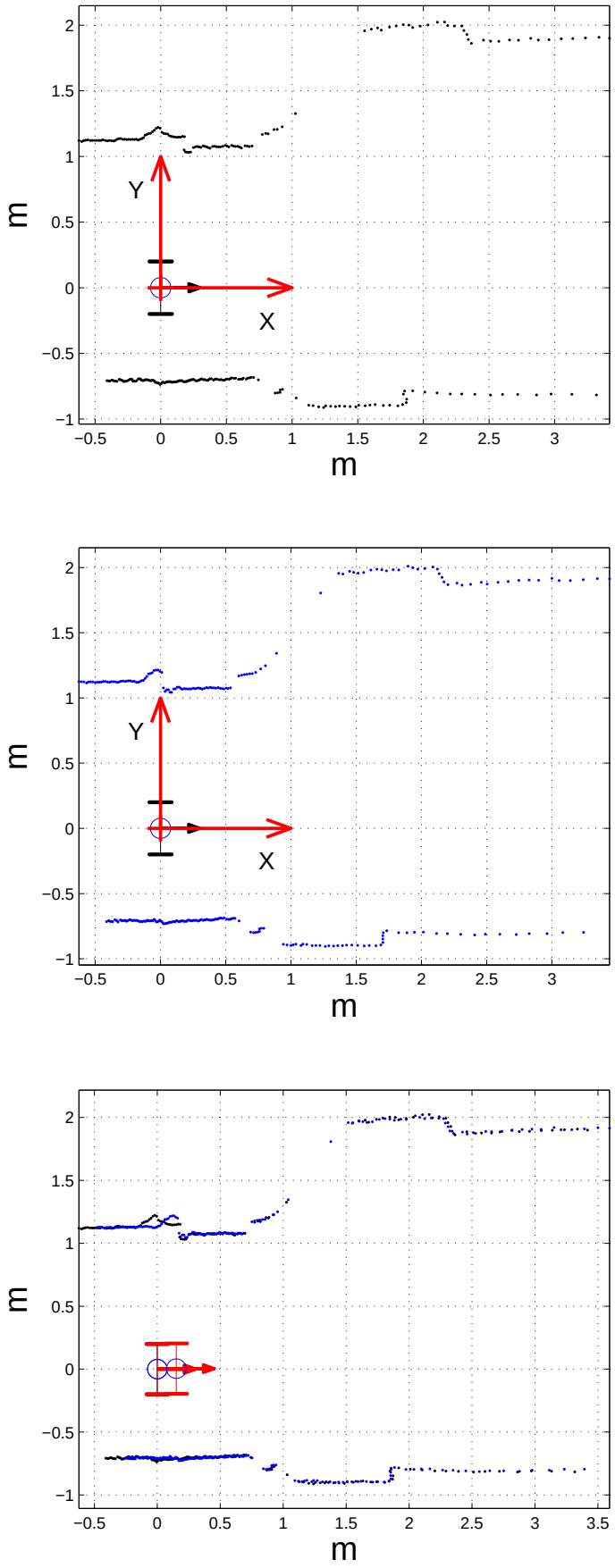


Figura 166: Correspondência de Varreduras com dados Reais para Diferentes Situações

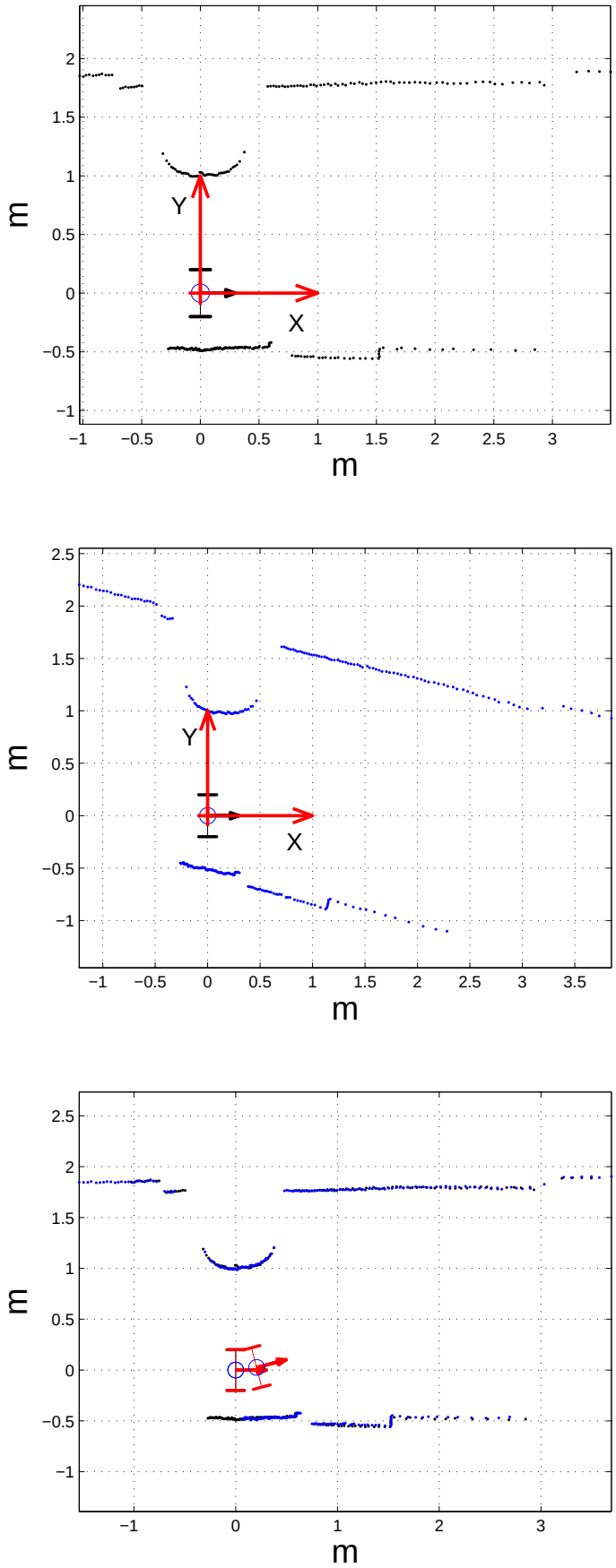


Figura 168: Correspondência de Varreduras com dados Reais para Diferentes Situações

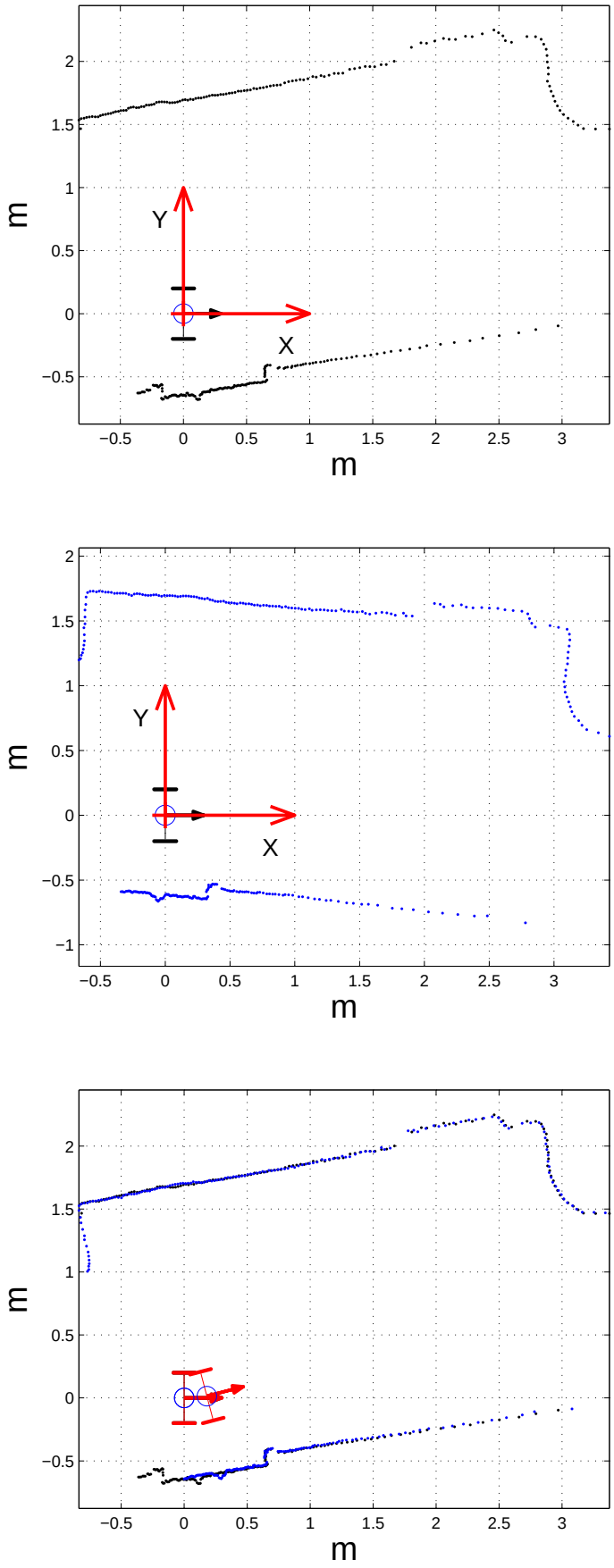


Figura 170: Correspondência de Varreduras com dados Reais para Diferentes Situações

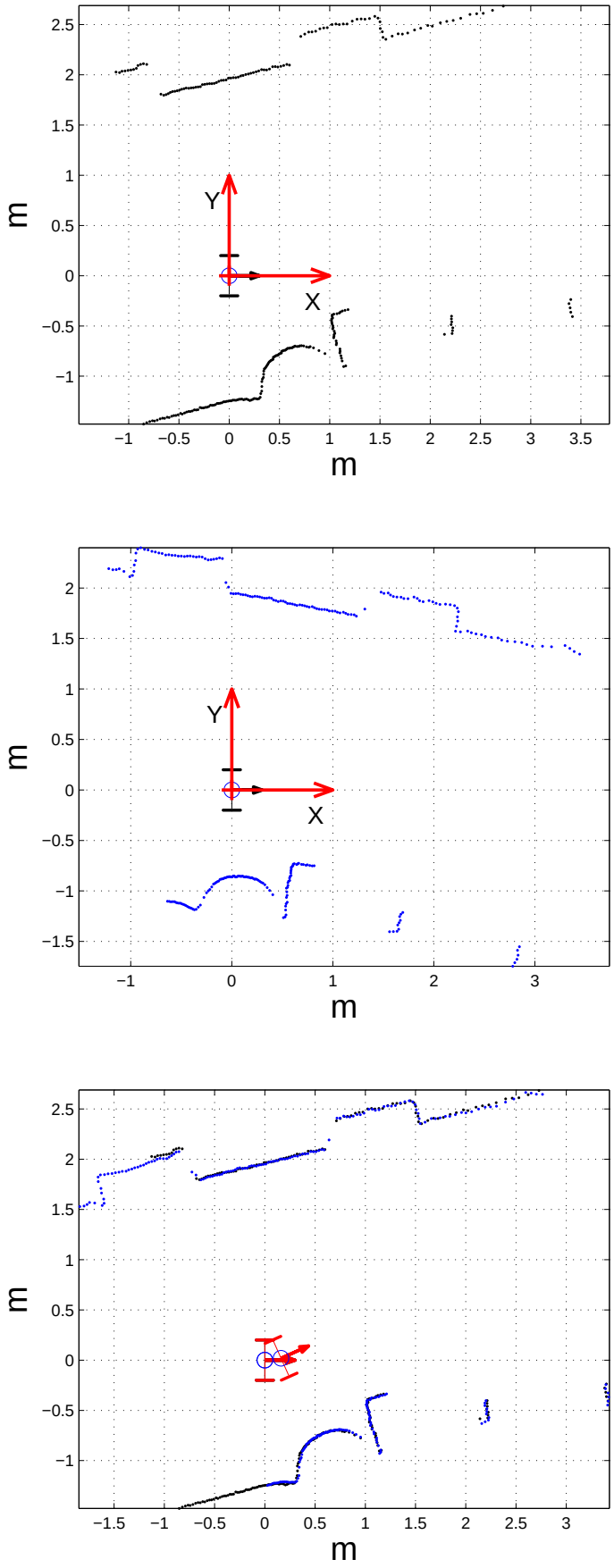


Figura 172: Correspondência de Varreduras com dados Reais para Diferentes Situações

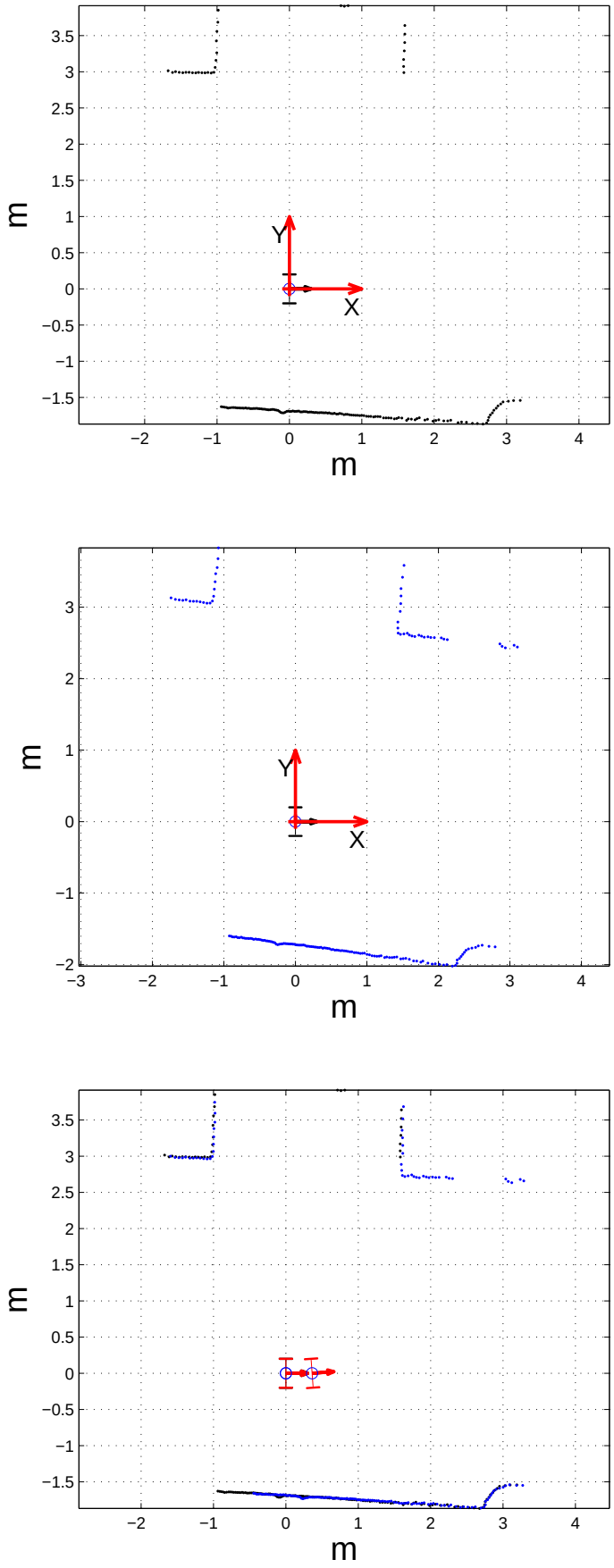


Figura 174: Correspondência de Varreduras com dados Reais para Diferentes Situações

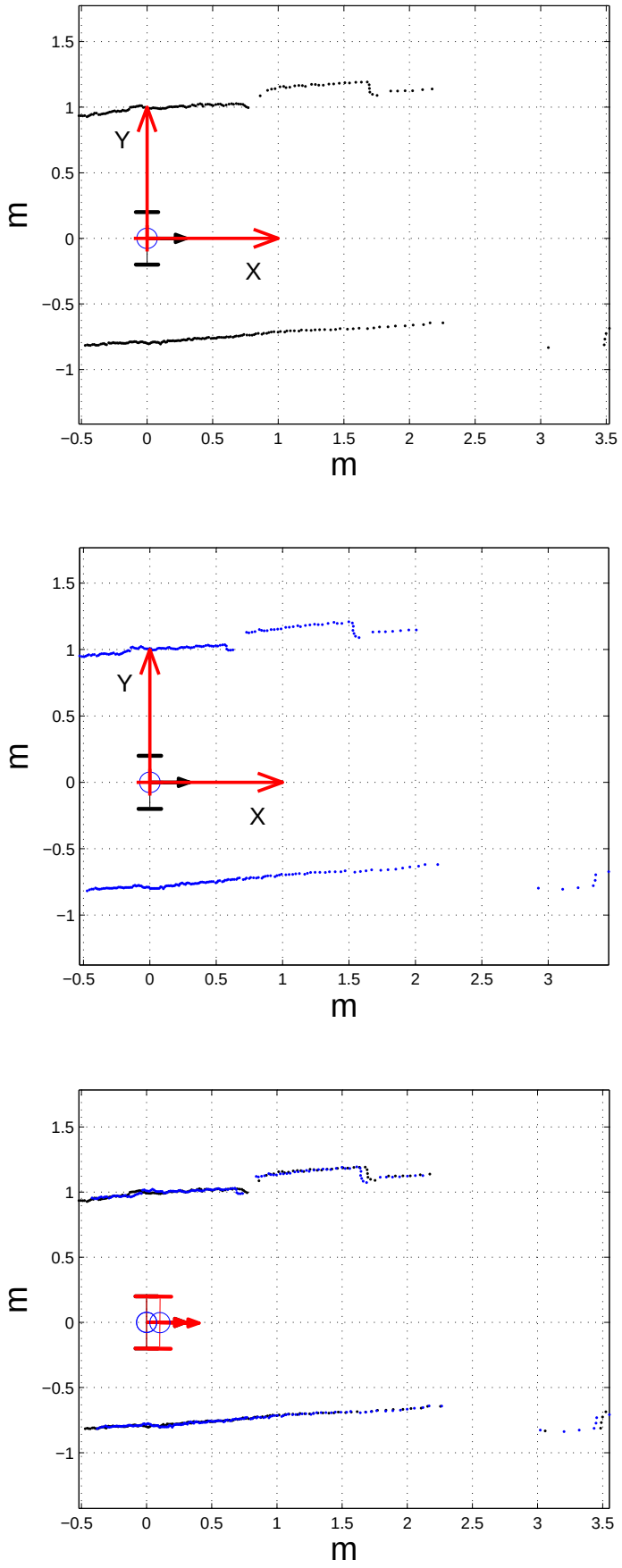


Figura 176: Correspondência de Varreduras com dados Reais para Diferentes Situações

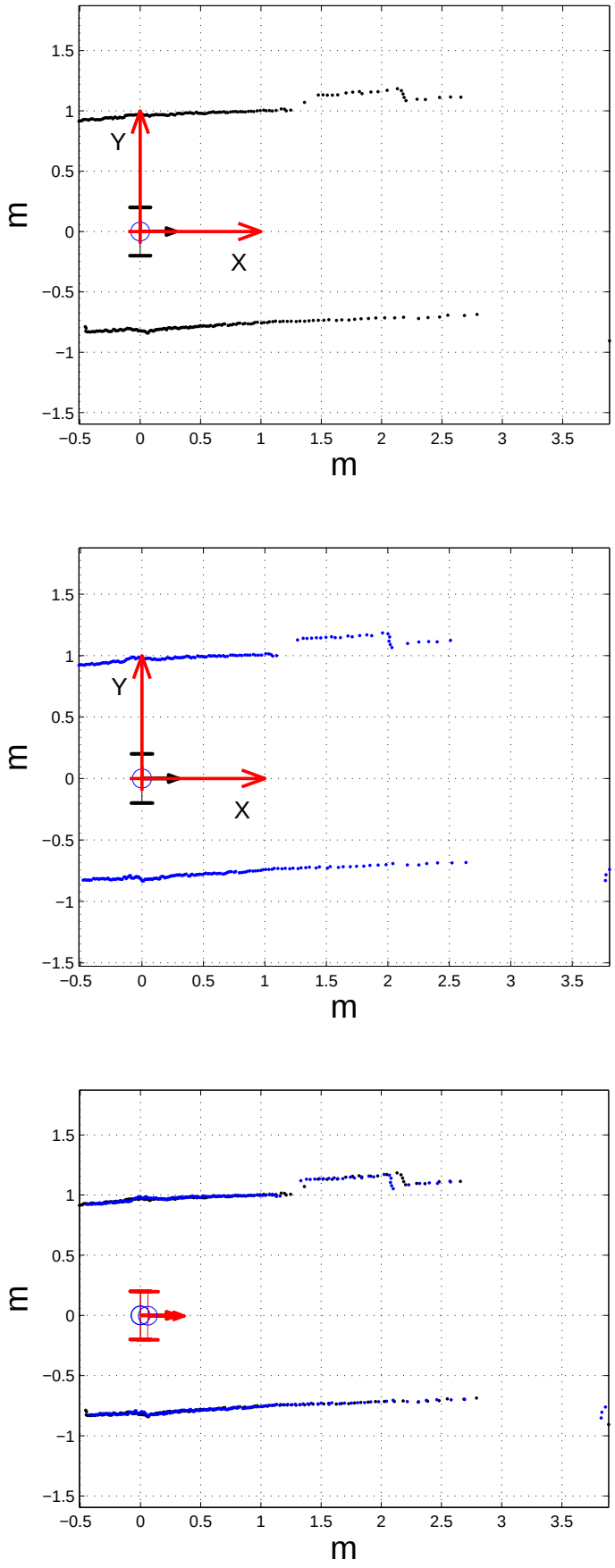


Figura 178: Correspondência de Varreduras com dados Reais para Diferentes Situações

O primeiro experimento foi realizado no corredor do quarto andar do Edifício Cardeal Leme da PUC-Rio, onde o robô móvel com o LRF realiza seu movimento e assim captura dados. Nesta experiência, o número de leituras feitas pelo sensor foi 376, em diferentes posições do robô móvel.

O deslocamento do robô móvel é conseguido tomando duas leituras consecutivas e aplicando algoritmo de Correspondência de Varreduras. Assim, para este experimento foram aplicadas 375 sobreposições das leituras, e cada uma delas apresenta um valor numérico de sobreposição dado pela função objetivo do processo de otimização por Evolução Diferencial. A Figura 179 mostra o valor da função objetivo para cada deslocamento do robô móvel. Podemos ver que a maioria dos deslocamentos apresentam um valor entre 0 e 15, assim esses deslocamentos têm uma grande probabilidade de gerar um erro pequeno "Boa sobreposição entre duas leituras", assim um valor alto da função objetivo pode ser o sinal de uma sobreposição ruim.

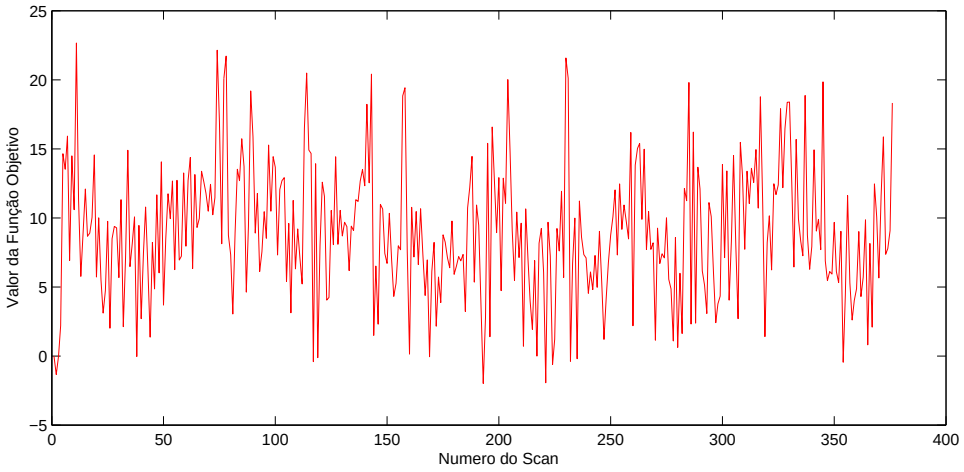


Figura 180: Valor numérico da Função Objetivo do Primeiro Experimento

Do mesmo modo, foram realizadas outras seis experiências, todas em ambientes internos (corredores) da PUC-Rio. A Tabela 5.1 mostra o ambiente testado e a quantidade de leituras que o robô móvel com o LRF capturou durante o seu movimento.

As Figuras 179, 186, 184 e 187 mostram o valor da função objetivo para cada deslocamento do robô móvel em cada experiência. Aqui também podemos ver que a maioria dos deslocamentos apresenta um valor entre 0 e 15, o qual indica que esses deslocamentos têm uma grande probabilidade obter um erro pequeno.

Tabela 5.1: Tabela das Experiências Realizadas.

Experiência	Local	Número de Leituras
1	4^o Andar CARDEAL LEME PUC-Rio	376
2	4^o Andar CARDEAL LEME PUC-Rio	557
3	1^o Andar CARDEAL LEME PUC-Rio	290
4	1^o Andar CARDEAL LEME PUC-Rio	964
5	1^o Andar KENNEDY PUC-Rio	262
6	1^o Andar CARDEAL LEME PUC-Rio	324
7	1^o Andar KENNEDY PUC-Rio	772

Tabela 5.2: Percentual de leituras com função Objetivo maior que 20.

Experiência	Número de Leituras	$F_{Obj} > 20$	%
1	376	9	2.40%
2	557	9	1.62%
3	290	5	1.72%
4	964	24	2.49%
5	262	14	5.34%
6	324	14	4.32%
7	772	25	3.24%

Para melhor compreensão dos resultados, podemos definir um valor de referência experimentalmente, que pode indicar se a sobreposição foi boa ou ruim. De acordo com as experiências, este valor é estimado por 20. A Tabela 5.2 mostra o número de deslocamentos onde a função objetivo (ED) de sobreposição é maior que 20 para cada experiência.

Nem todos os valores elevados da função objetivo pelo processo de otimização indicam uma sobreposição errada, em termos gerais, um valor próximo de zero indica que a maior parte das duas leituras foram correlacionadas, mas como o robô móvel está sempre em movimento, as leituras dos sensores contêm diferentes partes do meio ambiente. Por outro lado, como as leituras são consecutivas, elas deveriam conter uma grande parte comum do ambiente, que deve ser sobreposta.

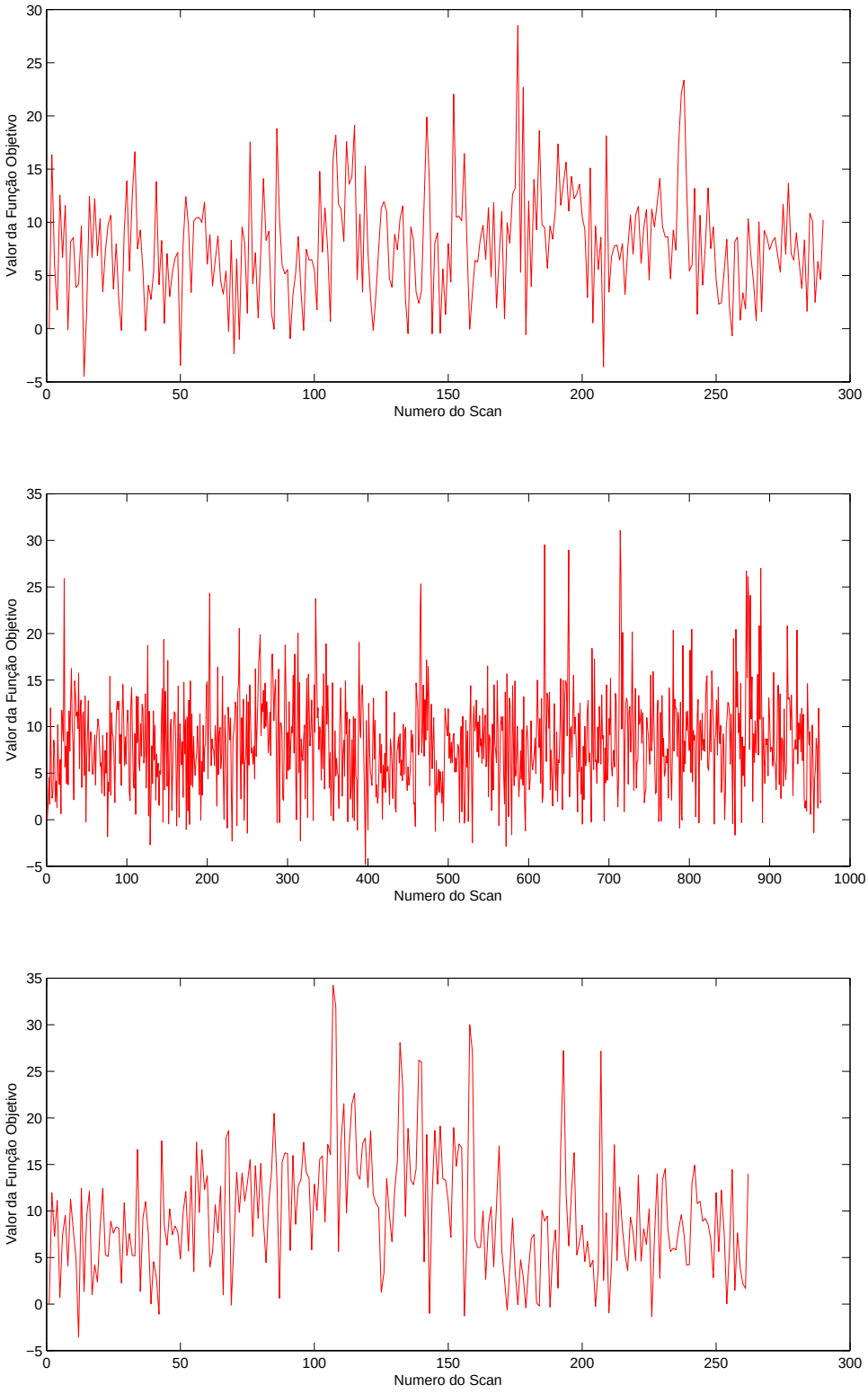


Figura 186: Correspondência de Varreduras com dados Reais para as Experiências 3,4 e 5

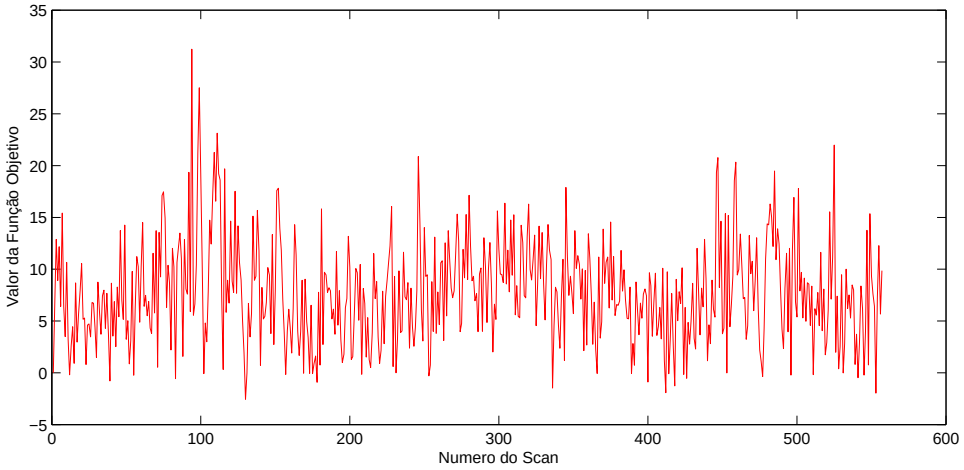


Figura 187: Valor numérico da Função Objetivo do Segundo Experimento

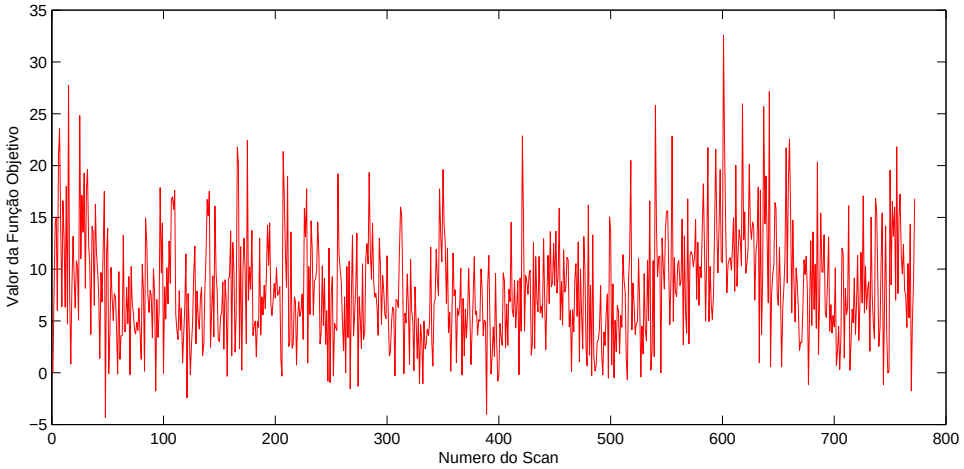


Figura 188: Correspondência de Varreduras com dados Reais (Experiência 7)

5.2 SLAM mediante Correspondência de Varreduras

Similarmente, para obter o mapa e a trajetória do robô móvel usando o algoritmo de Correspondência de Varreduras, é necessário fazer uma transformação de coordenadas de cada deslocamento do robô móvel. Assumindo assim que a posição inicial do robô móvel é $(0,0)$, a Figura 189 mostra a sua localização mediante os pontos vermelhos e o mapa mediante pontos azuis para parte do corredor no quarto andar do Prédio Cardeal Leme da PUC-Rio. É importante ressaltar que este ambiente apresenta duas paredes paralelas, uma delas com poucas características, o que faz o algoritmo de Correspondência de Varreduras ter dificuldade no momento de fazer a sobreposição de duas leituras consecutivas do sensor. Apesar disso, o mapa gerado fornece um esboço do mapa real.

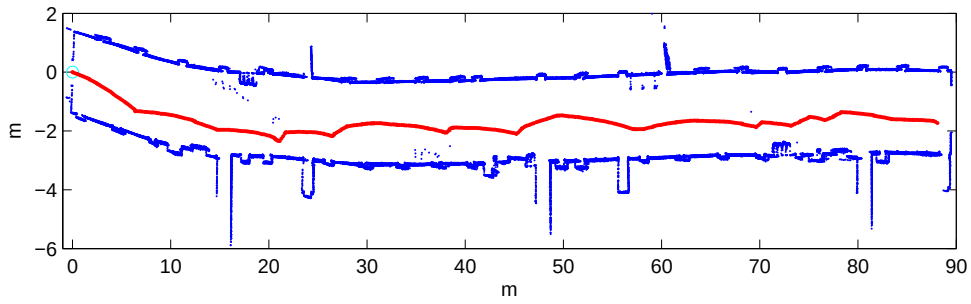


Figura 191: SLAM para dados reais usando Correspondência de Varreduras: Edifício Cardeal Leme 4º Andar PUC-Rio

A Figura 193 mostra o mapeamento do corredor no primeiro andar do Prédio Cardeal Leme da PUC-Rio, e também a trajetória do robô móvel. Ao contráRio corredor do quarto andar mapeado na Figura 189, neste corredor as duas paredes paralelas tem características (colunas e armários) que facilitam a sobreposição.

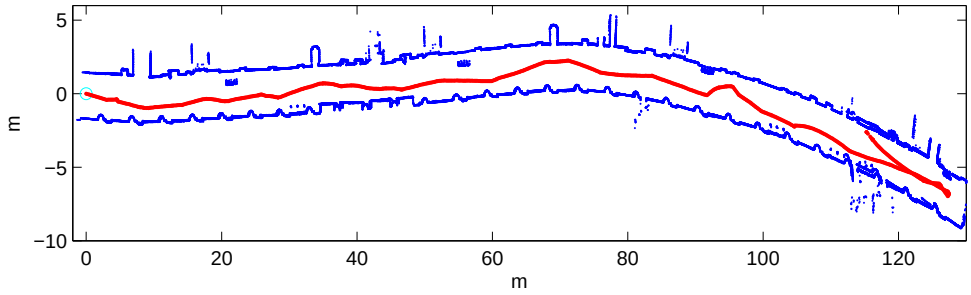


Figura 195: SLAM para dados reais usando Correspondência de Varreduras: Edifício Cardeal Leme 1º Andar PUC-Rio

Em ambos os experimentos, os corredores eram basicamente constituidos de paredes quase paralelas. As únicas características foram algumas colunas, portas pequenas e escadas, mas sem quinas ou corredores perpendiculares. A Figura 196 mostra outro ambiente com estas características. O ambiente onde o teste foi feito é o corredor do primeiro andar no Edifício Kennedy da PUC-Rio. Como se pode ver este corredor apresenta caminhos perpendiculares, colunas grandes e também paredes paralelas sem muitas características. Durante o movimento do robô móvel neste ambiente, o robô passa duas vezes por uma parte do mapa, mas é capaz de distinguir ambas as excursões

Assim, para os três ambientes testeados, o algoritmo de Correspondência de Varreduras gera um mapa aproximado e estima bem a localização do robô móvel dentro do mapa. Estes mapas podem ser melhoradas usando uma repre-

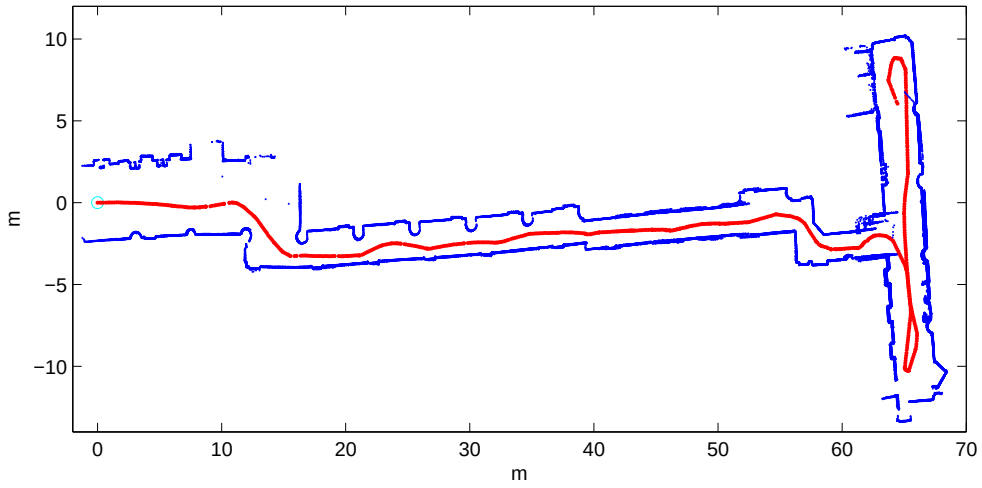


Figura 198: SLAM para dados reais usando Correspondência de Varreduras: Edifício Kennedy 1º Andar PUC-Rio

sentação de grade de ocupação.

5.3 Mapeamento e Localização usando DP-SLAM

O algoritmo DP-SLAM fornece um mapeamento de maior qualidade para um ambiente. Assim com a adição do modelo de movimento, descrita no capitulo 4 e as leituras do sensor LRF, nesta seção apresentamos 2D mapas, para ambientes reais onde o teste foi realizado.

Os parâmetros do algoritmo DP-SLAM usados para todos os experímentos realizados estão na Tabela 4.4, estes parâmetros são recomendadas em [4].

Figura 200: Mapa $2D$ do ambiente da primeira experiência obtido usando o algoritmo DP-SLAM

Para obter mapas dos ambientes reais com o algoritmo DP-SLAM, usamos os dados dos experimentos descritos na Tabela 5.1. A Figura 198 mostra o mapa

gerado com uma resolução de $1m = 35grades$ para o experimento 1 do Prédio Cardeal Leme no 4^o andar na PUC-Rio.

A segunda experiência em um ambiente real é mostrada na Figura 203, semelhante à experiência 1, referenteao corredor do 4^o andar do Prédio Cardeal Leme da PUC-Rio, mas com trajeto do robô móvel bem mais longo. Para uma melhor visualização, o mapa completo é particionado em 4, ver Figura 209.

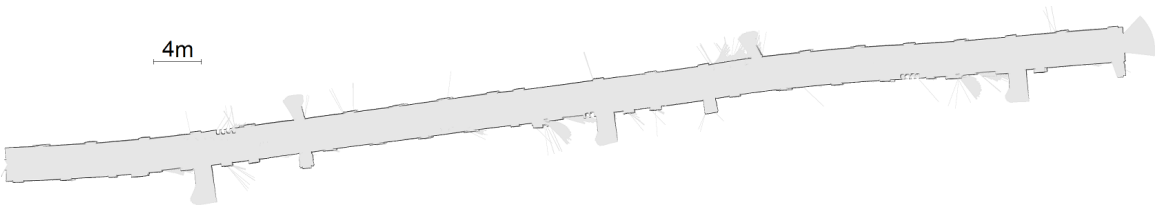


Figura 205: Mapa $2D$ do ambiente da segunda experiência obtido usando o algoritmo DP-SLAM

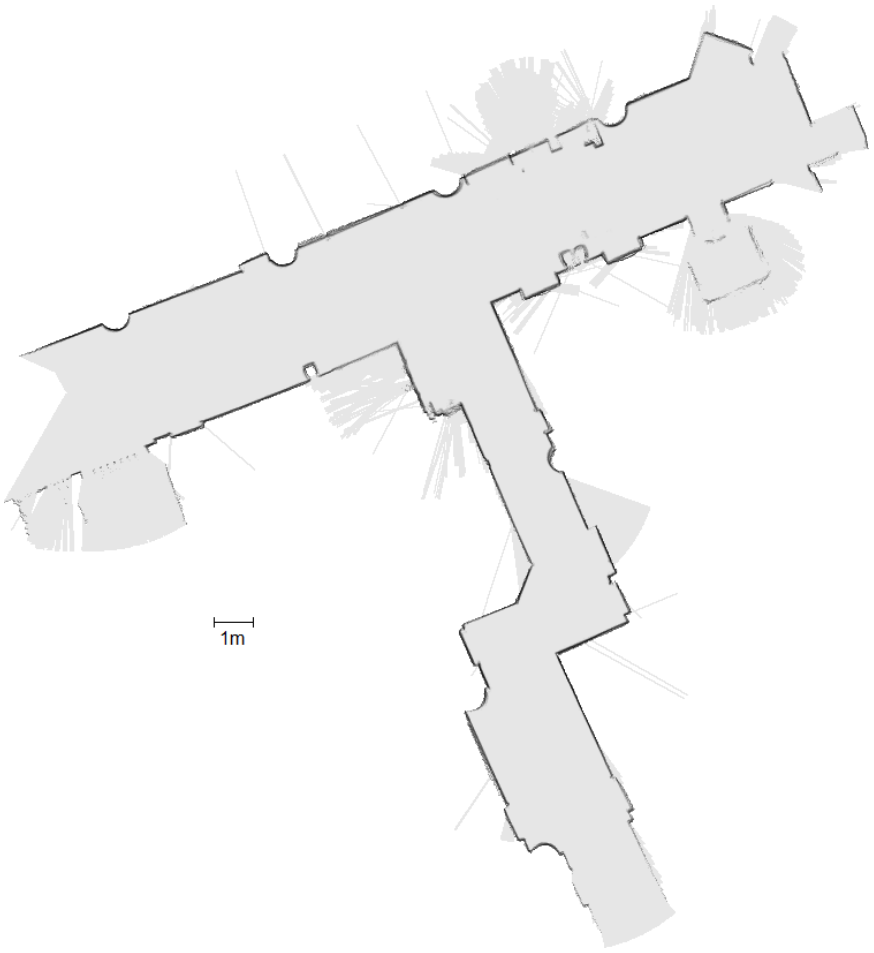


Figura 207: Mapa $2D$ do ambiente da quinta experiência obtido usando o algoritmo DP-SLAM

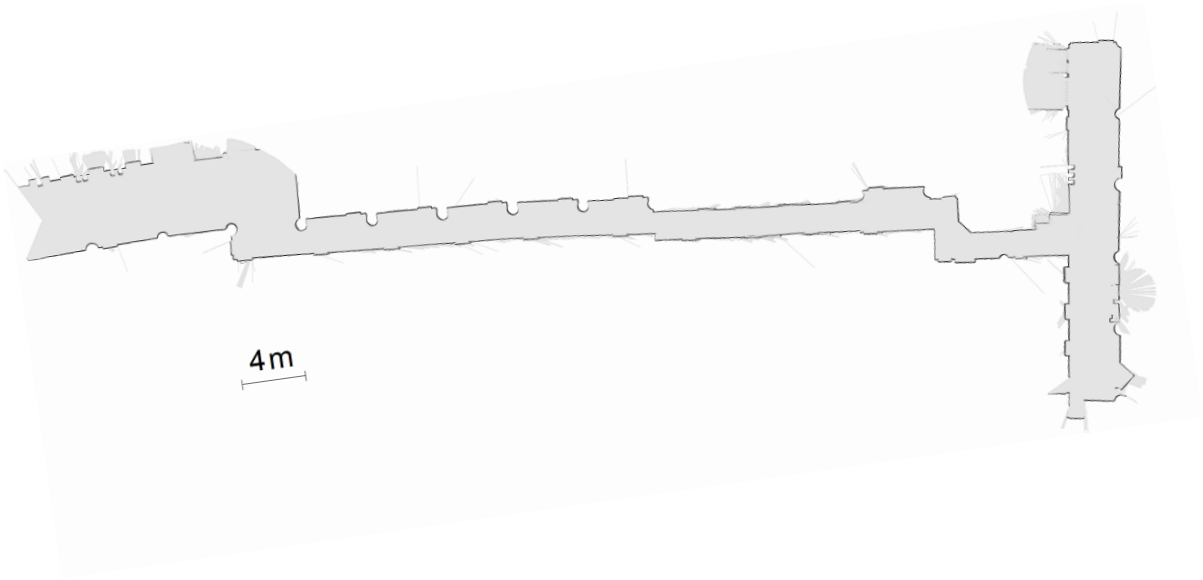


Figura 209: Mapa 2D do 1^o andar do Prédio Kennedy usando o algoritmo DP-SLAM

O mapa gerado usando os dados da quarta experiência feita no 1^o Andar do Prédio Cardeal Leme da PUC-Rio é apresentado na Figura 213 (a), o qual mostra o corredor deste prédio, que para uma melhor visualização é dividido em 3 partes. Neste mapa, Observam-se mais claramente as colunas, as portas, elevadores, as lixeiras, entre outras características, permitindo a criação de um melhor mapa.

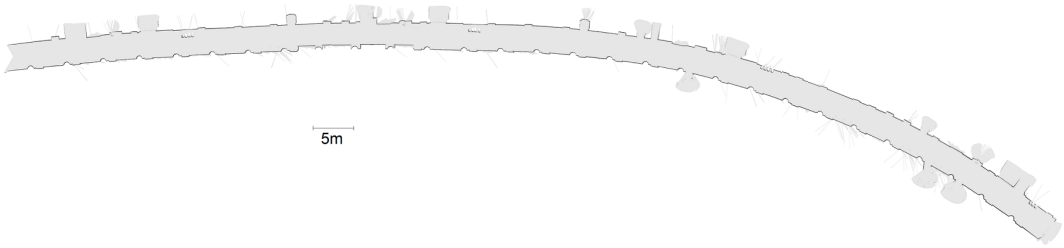
Os primeiros mapas apresentados até agora são longos corredores que têm paredes quase paralelas, com algumas características menores. Mas as experiências 5 e 7, por outro lado, inclui corredores que apresentam trechos perpendiculares.

A Figura 205 mostra o mapa 2D de uma parte do corredor do 1^o andar do Prédio Kennedy da PUC-Rio. Neste figura se pode ver que o algoritmo é capaz de gerar mapas mesmo na presença de corredores perpendiculares.

As Figuras 205 e 207 mostram o mapa do corredor do Prédio Kennedy (1^o andar). Este corredor, além de possuir colunas, portas e algumas características, apresenta caminhos perpendiculares e curvas.

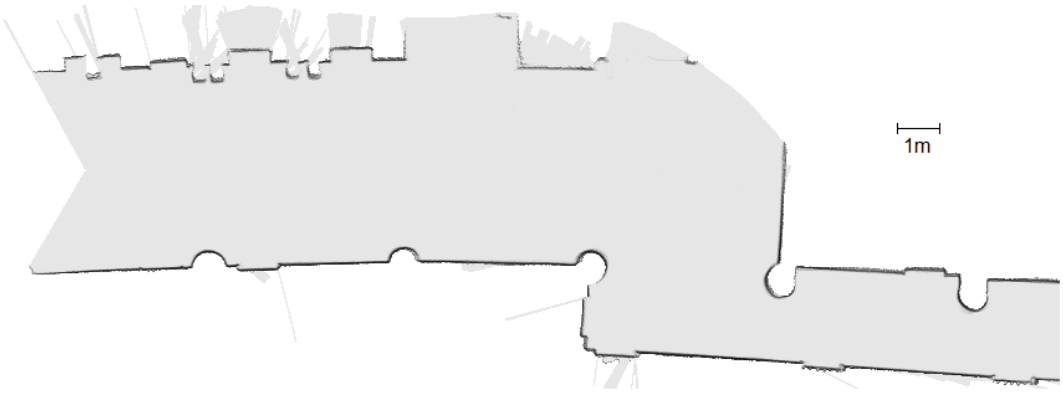
Por causa da longa viagem do robô móvel para executar este teste, o mapa gerado é dividido em três, para uma melhor visualização, ver Figura 219. Nesta figura na primeira partição do mapa, (a), podemos ver um mapeamento sem muita ambiguidade. Ao contráRio da segunda partição, (b), onde no lado direito,

Figura 211: Detalhes do Mapa 2D do ambiente da segunda experiência obtido usando o algoritmo DP-SLAM

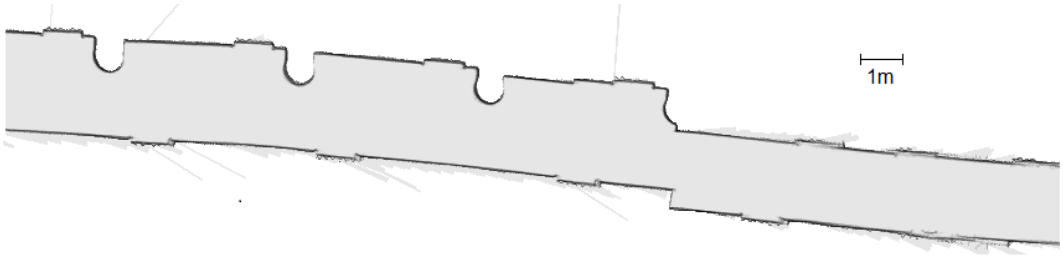


213(a): Mapa 2D do ambiente Completo

Figura 215: Detalhes do Mapa 2D do ambiente da quarta Experiência obtido usando o algoritmo DP-SLAM:



217(a):



218(b):

219(c):

Figura 221: Detalhes do Mapa $2D$ do ambiente da sétima experiência obtido usando o algoritmo DP-SLAM

há paredes paralelas que causam erro no processo de sobreposição, o que causou ambiguidade nesta seção do mapa. O mesmo efeito ocorre na parte inicial da terceira partição do mapa, (c).

5.4 Comparação com Plantas Baixas

Os mapas obtidos no presente trabalho podem ser comparados com as plantas baixas originais das instalações onde cada experiência teve lugar, disponibilizadas pelo Departamento de Engenharia Civil da PUC-Rio. As 7 experiências da Tabela 5.1 foram realizadas em três corredores da PUC-Rio, 2 no Prédio Cardeal Leme e 1 no Prédio Kennedy.

Assim a Figura 225 mostra a sobreposição do mapa criado por DP-SLAM com a planta baixa do quarto pavimento do Prédio Cardeal Leme da PUC-Rio. Um desalinhamento entre estes mapas pode ser observado, devido ao movimento longo do robô móvel. Este mapa corresponde à experiência número 1 da Tabela 5.1.

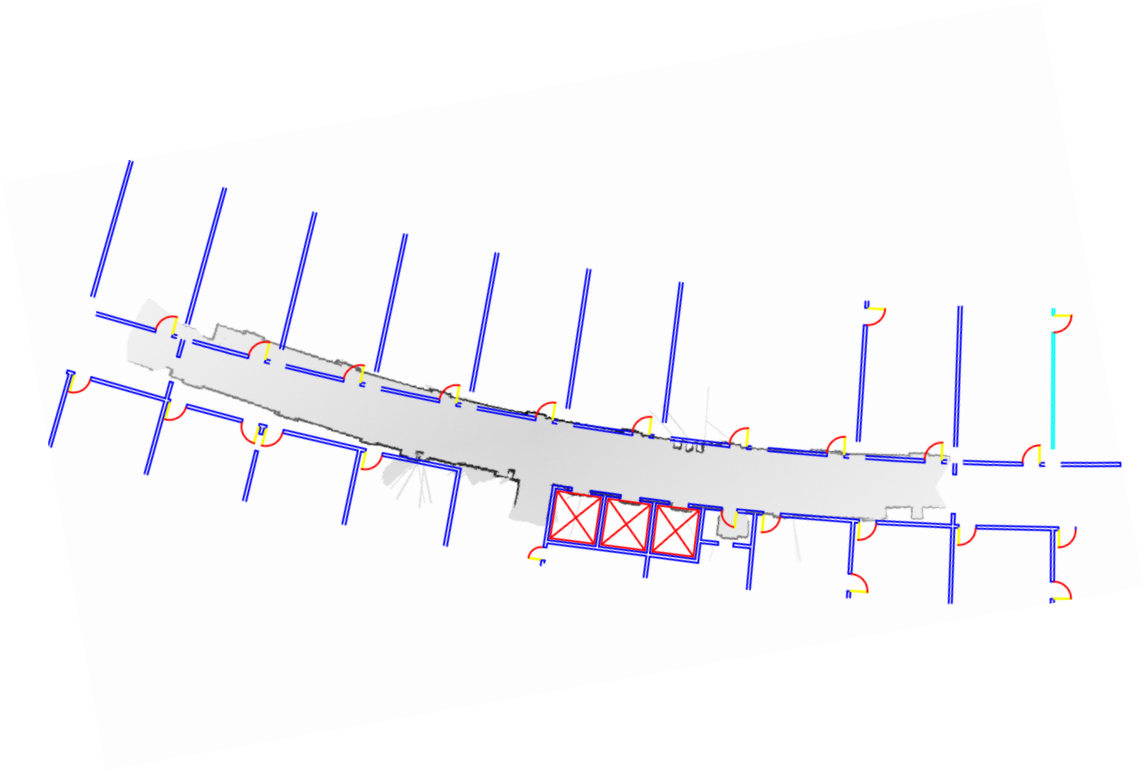
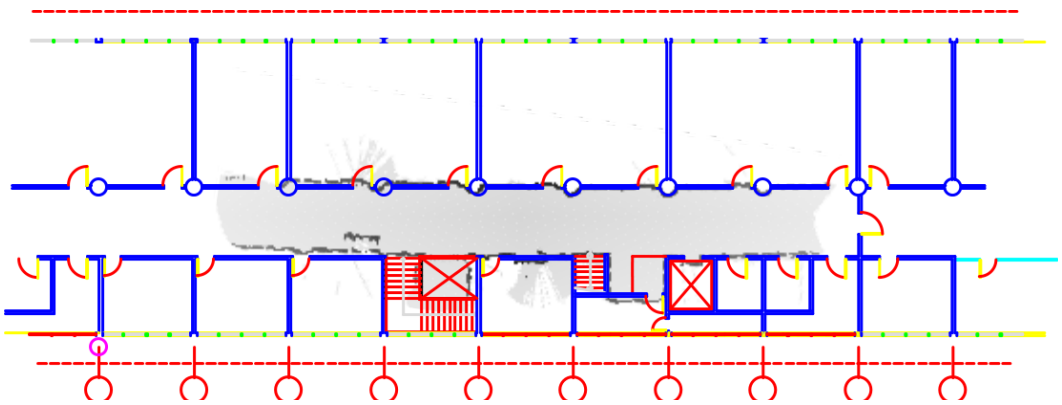
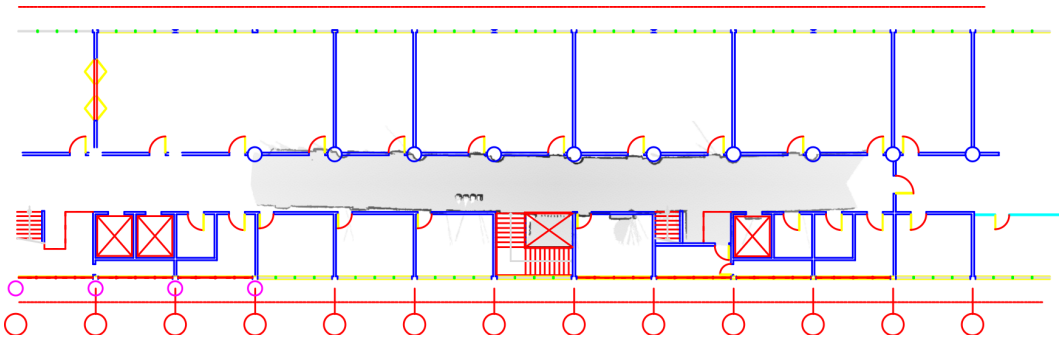


Figura 227: Sobreposição de mapas do Prédio Cardeal Leme 4º andar (Experimento 1)

A Figura 229 mostra a comparação dos mapas criados neste trabalho com a planta baixa do primeiro pavimento do Prédio Cardeal Leme da PUC-Rio. A



228(a): Experiência 3



229(b): Experiência 4

Figura 230: Sobreposição de mapas do Prédio Cardeal Leme 1^o andar

Figura 229 (a) corresponde à experiência número 3 da Tabela 5.1, e a Figura 229 (b) a uma parte da experiência número 4, semelhante à Figura 225, associada a corredores longos. Note-se que, nestas experiências, as paredes são quase paralelas e a trajetória do robô móvel é praticamente em uma só direção, o que prejudica a qualidade do mapa criado.

A Figura 235 mostra a sobreposição do mapa criado por DP-SLAM com a planta baixa do primeiro pavimento do Prédio Kennedy da PUC-Rio. Este ambiente apresenta melhores resultados que os anteriores, devido à variedade de características que possui.

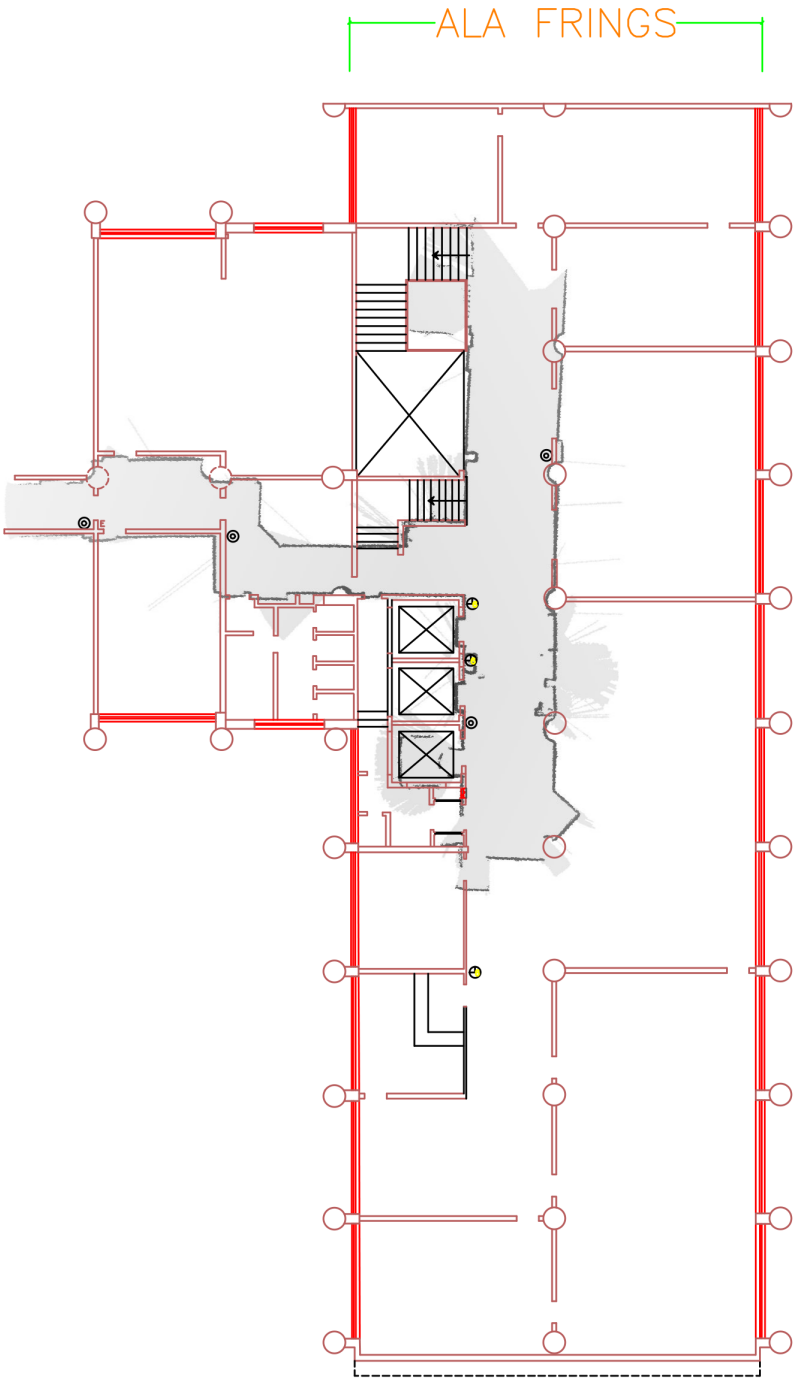


Figura 236: Sobreposição de mapas do Prédio Kennedy 1º andar

6
Comentários finais e sugestões

O presente trabalho buscou uma solução ao problema de SLAM para ambientes internos. Considera-se que o robô móvel (iRobot Create) está equipado com um único sensor 2D-LRF ($URG-04LX-UG01$), sem nenhuma informação de odometria. Baseado no trabalho prévio de Luis Ynoquio [2], foi proposta aqui uma variante do algoritmo DP-SLAM, modificando o modelo de movimento do robô móvel, como uma solução ótima. Embora, recomenda o uso de LRF com alta faixa de medição. Assim trabalhamos o algoritmo DP-SLAM com um LRF de baixo custo, na Tabela 3.1 mostra as características do sensor.

As simulações foram feitas usando dados experimentais desenvolvidos em MATLAB e dados reais obtidos com o robô móvel e o LRF. Os resultados obtidos, tanto no algoritmo de Correspondência de Varreduras quanto na solução do problema de SLAM, permitem estabelecer as seguintes conclusões:

1. No processo de Correspondência de Varreduras, um parâmetro importante para a obtenção de um ótimo desempenho do algoritmo é o tamanho da grade na NDT. Trabalhos anteriores recomendam um valor de grade de $1,0m$ para ambientes internos típicos, mas nas simulações com o LRF de faixa de medição de $4m$ os resultados do algoritmo de Correspondência de Varreduras apresentam um melhor desempenho para um valor de $0,5m$, ver Figura 101. Apesar da baixa qualidade de dados do sensor, ele apresenta uma grande densidade de pontos (682), o que permite trabalhar com valores Grade menores que $1,0m$. Mas, quanto menor o valor da grade, maior o tempo de processamento. Consideramos um valor adequado de $0,5m$ para a grade, que aproximadamente resulta em um tempo de $30s$ para cada Correspondência de Varreduras no microcomputador utilizado (Asus Eee PC 1000HEB, com processador Intel Atom N280 de $1,66GHz$).

2. Conclui-se, a partir desses experimentos, que a otimização do algoritmo de Correspondência de Varreduras usando Evolução Diferencial apresenta bons resultados no cálculo de deslocamento de robô móveis, para os parâmetros da Tabela 4.1 sobre o número de população e iterações. Estes valores são semelhantes aos calculados em [2].

3. O algoritmo de DP-SLAM é uma ferramenta muito poderosa para resolver o problema de SLAM, que também permite criar mapas de alta qualidade em ambientes reais com um sensor de baixo custo.

4. O uso do filtragem de varreduras baseada em saliências: onde pontos da varredura redundantes são descartados e pontos que apresentam características relevantes são preservados, melhora a eficiência e precisão do algoritmo de Correspondência de Varreduras em comparação com o trabalho prévio de Luis Ynoquio [2]. Além disso os algoritmos são testados, para obter mapas em ambientes reais.

5. Evolução Diferencial fornece ótimos resultados no processo de otimização, na análise de dados reais, como podemos ver na Figura 174, que mostra a sobreposição de algumas leituras do sensor.

Enfim, a ferramenta desenvolvida nesta dissertação permite que um robô móvel de baixo custo com um único LRF 2D obtenha mapas precisos de ambientes internos, que podem ser úteis em missões de busca e reconhecimento, por exemplo, em armazéns, instalações de produção e outras áreas industriais. As técnicas aplicadas aqui podem ser usadas para criar mapas tridimensionais. Esta aplicação fica como sugestão de trabalho futuro. Além disso, sugere-se como trabalhos futuros melhorar o tempo de procesamiento de $30s$, que é o tempo para realizar cada Correspondência de Varreduras, esse tempo não é adequado para aplicações em tempo real, pode-se usar a programação paralela ou computadores com maior eficiência. Também realizar o mapeamento de ambientes externos ou ambientes dinâmicos, utlilizando sensores de maior qualidade.

Referências Bibliográficas

[1] SEBASTIAN THRUN, WOLFRAM BURGARD, D. F.. **Probabilistic Robotics**. The MIT Press Cambridge, Massachusetts, London, England, 2005.

[2] HERRERA, L. Y.. **Mobile robot simultaneous localization and mapping using dp-slam with a single laser range finder**. Dissertação de Mestrado, Mechanical Engineering Department at PUC-Rio, 2011.

[3] YISHA LIU, Y. S.. **Mobile robot instant indoor map building and localization using 2d laser scanning data**. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCE AND ENGINEERING (ICSSE), 2012.

[4] ELIAZAR, A.. **DP-SLAM**. Tese de Doutorado, Department of Computer Science, Duke University, 2005.

[5] TEMELTAS HAKAN, K. D.. **Slam for robot navigation**. Aerospace and Electronic Systems Magazine, IEEE, 2008.

[6] GAMINI DISSANAYAKE, SHOUDONG HUANG, Z. W.. **A review of recent developments in simultaneous localization and mapping**. In: IEEE INTERNATIONAL CONFERENCE ON INDUSTRIAL AND INFORMATION SYSTEMS (ICIIS), 2011.

[7] YAP T.N., S. C.. **Slam in large indoor environments with low-cost, noisy, and sparse sonars**. In: IEEE-ICRA INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2009.

[8] MALLIOS A., RIDAO, P. R. D. H. E.. **Probabilistic sonar scan matching slam for underwater environment**. In: OCEANS 2010 IEEE - SYDNEY, 2010.

[9] LU FENG, E. M.. **Robo pose estimation in unknown environments by matching 2d range scans**. Nerth York, Canada M3J 1P3: Department of Computer Science, York University, 1994.

[10] ANDREAS NUCHTER, KAI LINGEMANN, J. H.. **6d slam 3d mapping outdoor environments**, 2007.

[11] THRUN, S.. **Robotic mapping: A survey**. In: EXPLORING ARTIFICIAL INTELLIGENCE IN THE NEW MILLENIUM. Morgan Kaufmann, 2002.

[12] JOHN FOLKESSON, H. C.. **Outdoor exploration and slam using a compressed filter**. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2003.

[13] SEBASTIAN THRUN, WOLFRAM BURGARD, D. F. H. H. M. M.. **A probabilistic approach to concurrent mapping and localization for mobile robots**. In: MACHINE LEARNING AND AUTONOMOUS ROBOTS, 1998.

[14] GAMINI DISSANAYAKE, PAUL NEWMAN, S. C.. **A solution to the simultaneous localization and map building (slam) problem**. IEEE Transactions on Robotics and Automation, 2001.

[15] SEBASTIAN THRUN, DAPHNE KOLLER, Z. G. H. D. W.. **Simultaneous localization and mapping with sparse extended information filters**. The International Journal of Robotics Research, 2004.

[16] MICHAEL MONTEMERLO, SEBASTIAN THRUN, D. K. B. W.. **Fastslam: A factored solution to the simultaneous localization and mapping problem**. In: IN PROCEEDINGS OF THE AAAI NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2002.

[17] BIBER, P.. **The normal distributions transform: A new approach to laser scan matching**. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2003.

[18] PAUL BESL, N. M.. **A method for registration of 3-d shapes**. IEEE Transactionson Pattern Analysis and Machine Intelligence, 1992.

[19] ALSHAWA, M.. **Iterative closest line a novel point cloud registration algorithm based on linear features**. Strasbourg, France: Potogrammetry and Geomatics Group MAP-PAGE UMR 694, 2007.

[20] MURPHY, K.. **Bayesian map learning in dynamic environments**. In: IN NEURAL INFORMATION PROCESSING SYSTEMS 11, 1999.

[21] AUSTIN ELIAZAR, R. P.. **Dp-slam 2.0**. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2004.

[22] AUSTIN ELIAZAR, R. P.. **Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks**. In: 18TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 2003.

[23] JAN WEINGARTEN, R. S.. **Ekf-based 3d slam for structured environment reconstruction**. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2005.

[24] DUCKETT, T.. **A genetic algorithm for simultaneous localization and mapping**. In: IEEE-ICRA INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2003.

[25] HAN WANG, YI YAN, D. W. W.. **A ga based slam with range sensors only**. In: INTERNATIONAL CONFERENCE ON CONTROL AUTOMATION ROBOTICS VISION (ICARCV), 2010.

[26] HUIJING ZHAO, R. S.. **Reconstructing textured cad model of urban environment using vehicle-borne laser range scanners and line cameras**. In: IN SECOND INTERNATIONAL WORKSHOP ON COMPUTER VISION SYSTEM (ICVS), 2001.

[27] Scanning range finder. **SOKUIKI sensor**, disponível em: <http://www.hokuyo-aut.jp>. Acessado: 04 Abr. 2013.

[28] ELFES, A.. **Occupancy grids: A probabilistic framework for mobile robot perception and navigation**. Tese de Doutorado, Department Electrical and Computer Engineering, Robotics Institute, Carnegie Mellon University, 1989.

[29] MICHAEL BOSSE, PAUL NEWMAN, J. L. S. T.. **Slam in large-scale cyclic environments using the atlas framework**. Internatioonal Journal of Robotics Research, 2004.

[30] TOMONO, M.. **A scan matching method using euclidean invariant signature for global localization and map building**. Proceedings of the IEEE, International Conference on Robotics and Automation, 2004.

[31] ANTONI BURGUERA, YOLANDA GONZALEZ, G. O.. **On the use of likelihood fields to perform sonar scan matching localization**. IEEE-RSJ International Conference on Intelligent Robots and Systems, 2008.

[32] PETER BIBER, SVEN FLECK, W. S.. **A probabilistic framework for robust and accurate matching of point clouds**. In: 26TH PATTERN RECOGNITION SYMPOSIUM, 2004.

[33] KENNETH V.PRICE, RAINER M. STORN, J. A. L.. **Differential Evolution: A Practical Approach to Global Optimization**. Springer, Berlin Heidelberg New York, 2005.

[34] FOX, D.. **Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation**. Tese de Doutorado, Institute of Computer Science III, University of Bonn, Germany, 1998.

[35] TOMONO, M.. **Eficient global scan matching using saliency based scan point resampling**. IEEE-RSJ International Conference on Intelligent Robots and Systems, 2005.

[36] Programmable Robot. **iRobot Create**, disponível em: <http://www.irobot.com>. Acessado: 04 Abr. 2013.