Universidade Federal da Bahia
Escola Politécnica
Programa de Pós-Graduação em Engenharia Elétrica

Dissertação de Mestrado

Mapeamento 3D e Localização de Robôs Móveis utilizando Sensor LiDAR

MARCELO ESPINHEIRA CRAVO DE CARVALHO

Salvador
2024

Universidade Federal da Bahia
Escola Politécnica
Programa de Pós-Graduação em Engenharia Elétrica

Mapeamento 3D e Localização de Robôs Móveis utilizando Sensor LiDAR

Autor: Marcelo Espinheira Cravo de Carvalho
Orientador: Prof. Dr. André Gustavo Scolari Conceição - UFBA
Co-orientador: Prof. Dr. Tiago Trindade Ribeiro - UFBA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia como requisito parcial para conclusão do Mestrado em Engenharia Elétrica.

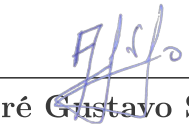
Salvador
2024

Mapeamento 3D e Localização de Robôs Móveis utilizando Sensor LiDAR

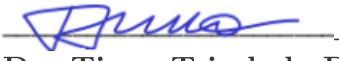
MARCELO ESPINHEIRA CRAVO DE CARVALHO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Escola Politécnica, Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Banca Examinadora da Dissertação



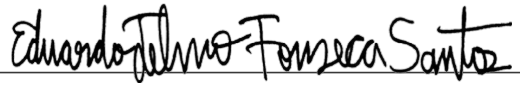
Prof. Dr. André Gustavo Scolari Conceição
Universidade Federal da Bahia - Orientador



Prof. Dr. Tiago Trindade Ribeiro
Universidade Federal da Bahia - Co-orientador

Prof. Dr. Paulo César Machado de Abreu Farias
Universidade Federal da Bahia - Membro Interno

Prof. Dr. Tito Luís Maia Santos
Universidade Federal da Bahia - Membro Interno



Prof. Dr. Eduardo Telmo Fonseca Santos
Instituto Federal da Bahia - Membro Externo

Salvador - BA, 17 de Dezembro de 2024

A Deus.
À minha amada esposa.
À minha querida filha.
Aos familiares e amigos que me ajudaram na realização deste trabalho.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, pela dádiva da vida.

Ao meu orientador, Professor Doutor André Gustavo Scolari Conceição, pela paciência, disponibilidade constante e incentivo prestado ao longo da elaboração da dissertação. Seu conhecimento e experiência foram fundamentais para o desenvolvimento deste trabalho.

Ao meu co-orientador, o Professor Doutor Tiago Trindade Ribeiro, por ter me acolhido, me motivado e me direcionado, especialmente com o uso das ferramentas computacionais necessárias para conclusão desta dissertação.

A minha amada esposa Karina, pelo amor, carinho, companheirismo, incentivo e dedicação de sempre.

A minha querida filha Marina, que é a minha fonte de motivação e inspiração, que esta conquista seja a primeira de muitas que compartilharemos juntos.

A toda a minha família, que mesmo distante, esteve presente em todos os momentos.

Muito obrigado a todos!

Salvador, Dezembro de 2024.

RESUMO

Nos últimos anos é crescente o desenvolvimento de algoritmos de localização e mapeamento simultâneo (SLAM) utilizando dados de sensores LiDAR (*Light Detection Ranging*). Os sensores LiDAR possuem a capacidade de realizar varredura 3D em 360° de ambientes, com tempo de aquisição e consumo de energia relativamente baixos, o que possibilita a sua utilização em sistemas embarcados. O objetivo desta pesquisa é estudar e implementar técnicas SLAM utilizando dados de sensores LiDAR, para obtenção do mapeamento e localização em ambientes internos (indoor) para tarefas de navegação com robôs móveis. Neste contexto foram implementadas as técnicas LeGO-LOAM, A-LOAM e F-LOAM, objetivando analisar a influência do volume de referenciais estáticos *features* geométricas na qualidade do mapeamento e precisão da localização instantânea. As estratégias LOAM realizam a captura de *features* geométricas a partir da classificação dos pontos da nuvem capturada pelo sensor LiDAR, e posteriormente identifica correspondência entre varreduras consecutivas para obter a estimativa de localização. Para análise de desempenho das técnicas foram testadas em simulação e em ambiente real. O simulador utilizado foi o Gazebo, o robô móvel Husky, o sensor LiDAR Velodyne VLP-16, com varreduras realizadas nos seguintes ambientes: em um corredor reto fechado com janelas, corredor reto fechado sem janela, corredor quadrado fechado e em um modelo 3D do LaR - Laboratório de Robótica do DEEC/UFBA. Os experimentos reais foram conduzidos no LaR, utilizando o robô móvel Husky, o sensor LiDAR Velodyne VLP-16 e o sistema de captura de movimento 3D Optitrack. Os experimentos confirmaram a eficiência dos algoritmos LiDAR SLAM para localização e mapeamento de robôs móveis, com melhor desempenho em ambientes com muitos referenciais estáticos. Em cenários mais simples, os resultados também foram satisfatórios, exceto pelo algoritmo LeGO-LOAM, que apresentou maior erro no eixo x em corredores retos sem janelas devido à falta de referenciais. O algoritmo LeGO-LOAM se destacou nos mapas gerados, exibindo maior detalhamento das estruturas e objetos presentes no ambiente. Apesar de uma tendência a apresentar erros mais significativos em curvas, os algoritmos LiDAR SLAM avaliados demonstraram eficiência na correção das estimativas em trechos retos, evidenciando uma baixa suscetibilidade a erros acumulados. Esses resultados reforçam o potencial dessas técnicas como soluções confiáveis para mapeamento preciso e localização em ambientes complexos.

Palavras-chave: Robótica Móvel; LiDAR SLAM; Mapeamento; Localização.

ABSTRACT

In recent years, the development of simultaneous localization and mapping (SLAM) algorithms using data from LiDAR (Light Detection Ranging) sensors has been increasing. LiDAR sensors are capable of performing 3D scanning in 360° of environments, with relatively low acquisition time and energy consumption, which allows their use in embedded systems. The objective of this research is to study and implement SLAM techniques using LiDAR sensor data, to obtain mapping and localization in indoor environments for navigation tasks with mobile robots. In this context, the LeGO-LOAM, A-LOAM and F-LOAM techniques were implemented, aiming to analyze the influence of the volume of static geometric references *features* on the quality of the mapping and accuracy of the instantaneous localization. The LOAM strategies capture geometric *features* from the classification of the points of the cloud captured by the LiDAR sensor, and subsequently identify correspondence between consecutive scans to obtain the location estimate. To analyze the performance of the techniques, they were tested in simulation and in a real environment. The simulator used was the Gazebo, the Husky mobile robot, the Velodyne VLP-16 LiDAR sensor, with scans performed in the following environments: a closed straight corridor with windows, a closed straight corridor without windows, a closed square corridor and a 3D model of LaR - Robotics Laboratory of DEEC/UFBA. The real experiments were conducted in LaR, using the Husky mobile robot, the Velodyne VLP-16 LiDAR sensor and the Optitrack 3D motion capture system. The experiments confirmed the efficiency of the LiDAR SLAM algorithms for localization and mapping of mobile robots, with better performance in environments with many static references. In simpler scenarios, the results were also satisfactory, except for the LeGO-LOAM algorithm, which presented greater error on the x-axis in straight corridors without windows due to the lack of references. The LeGO-LOAM algorithm stood out in the generated maps, displaying greater detail of the structures and objects present in the environment. Despite a tendency to present more significant errors on curves, the evaluated LiDAR SLAM algorithms demonstrated efficiency in correcting estimates on straight stretches, evidencing a low susceptibility to accumulated errors. These results reinforce the potential of these techniques as reliable solutions for accurate mapping and localization in complex environments.

Keywords: Mobile Robotics; LiDAR SLAM; Mapping; Localization.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

2D	*2 dimensões*
3D	*3 dimensões*
A-LOAM	*Advanced implementation of LiDAR Odometry And Mapping*
CIR	*Centro Instantâneo de Rotação*
EKF	*Extended Kalman Filter*
F-LOAM	*Fast LiDAR Odometry And Mapping*
FoV	*Field of View*
GPS	*Global Positioning System*
ICP	*Iterative Closest Point*
IMU	*Inertial Measurement Unit*
LeGO-LOAM	*Lightweight and Ground-Optimized LiDAR Odometry And Mapping*
LiDAR	*Light Detection and Ranging*
LM	*Levemberg-Marquardt*
LOAM	*Lidar Odometry and Mapping*
ROS	*Robot Operating System*
SLAM	*Simultaneous Localization and Mapping*
ToF	*Time of Flight*
UFBA	*Universidade Federal da Bahia*

Lista de Símbolos

c	Velocidade média da luz
d	Distância do objeto detectado por um sensor *laser*
$det(.)$	Determinante de uma matriz
$\mathbf{D}$	Matriz formada pelos vetores dados pela diferença entre pontos de uma nuvem de pontos e sua respectiva média no instante de tempo k
$e(.)$	Função de custo para cálculo de uma pose relativa
$exp.$	Exponencial natural
E	Matriz diagonal retangular
EMA	Erro Médio Absoluto
EQM	Erro Quadrático Médio
f_b	*Features* de borda
f_p	*Features* de plano
$\mathbf{H}_o^i$	Deslocamento horizontal da origem da estrutura do sensor para o laser i
$\mathbf{I}$	Matriz identidade
j	Ponto de varredura k
k	Instante de tempo discreto
l_{max}	Distância máxima percorrida em um percurso reto
m	Quantidades de pontos detectados por um sensor LiDAR

M	Mapa
M_k	Mapa no instante k
M_L	Mapa local
n	Variável de iteração
N	Número total de poses do robô
O	Sistema de coordenada inercial
p_{CIR}	Coordenada do centro instantâneo de rotação
p_A^O	Vetor de posição do sistema de coordenadas A com respeito ao sistema inercial O
$p_{A,x}^O$	Componente x do vetor p_A^O
$p_{A,y}^O$	Componente y do vetor p_A^O
$p_{A,z}^O$	Componente z do vetor p_A^O
p_B^O	Vetor de posição do sistema de coordenadas B com respeito ao sistema inercial O
$p_{g,k}$	Pose do *ground truth* no instante de tempo k
p_L	Vetor da posição estimada pela odometria LiDAR
$q_{k-n,k+n}$	Vetor formado pelas posições do robô dadas nos instantes $k-n$ e $k+n$
$q_{k,k-n}$	Vetor formado pelas posições do robô dadas nos instantes k e $k-n$
Q	Matriz formada pelos vetores dados pela diferença entre pontos de uma nuvem de pontos e sua respectiva média no instante de tempo $k-1$
$\mathbf{R}$	Matriz de rotação
$\mathbb{R}^2$	Espaço de configuração $\mathbb{R}$ bidimensional
$\mathbb{R}^3$	Espaço de configuração $\mathbb{R}$ tridimensional

$\mathbf{R}_A^O$	Matriz de rotação do sistema de coordenadas A para o sistema de coordenadas inercial O
$\mathbf{R}_B^O$	Matriz de rotação do sistema de coordenadas B para o sistema de coordenadas inercial O
$\mathbf{R}_B^A$	Matriz de rotação do sistema de coordenadas B para o sistema de coordenadas A
RM_{xy}	Ruído Médio de Localização no Plano xy
$\mathbf{R}_x(\phi)$	Matriz de rotação elementar ao redor do eixo x dada pelo ângulo ϕ
$\mathbf{R}_y(\theta)$	Matriz de rotação elementar ao redor do eixo y dada pelo ângulo θ
$\mathbf{R}_z(\psi)$	Matriz de rotação elementar ao redor do eixo z dada pelo ângulo ψ
s	Coeficiente de suavidade normalizada de um ponto em função dos pontos vizinhos
s_L	Vetor do ponto medido do ambiente pelo sensor LiDAR
$s_{L,x}$	Coordenada x do vetor s_L
$s_{L,y}$	Coordenada y do vetor s_L
s_L^R	Ponto medido do ambiente pelo sensor LiDAR com respeito ao sistema de coordenadas do robô R
s_L^O	Ponto medido do ambiente pelo sensor LiDAR com respeito ao sistema de coordenadas inercial O
$s_{L,z}$	Coordenada z do vetor s_L
$s_{L,z}$	Coordenada z do vetor s_L
$s_{L,z}$	Coordenada z do vetor s_L
S_L	Conjunto de pontos do sensor LiDAR
t	Instante de tempo
U	Matriz unitária

$v_{R,d}$	Velocidade linear das rodas do lado direito de um robô
$v_{R,e}$	Velocidade linear das rodas do lado esquerdo de um robô
v_R	Velocidade linear do robô na direção do eixo x
$\mathbf{V}_o^i$	Deslocamento vertical da origem da estrutura do sensor laser i
V^*	Matriz conjugada transposta da matriz unitária V
$\mathbf{x}_A^O$	Pose do sistema de coordenadas A com respeito ao sistema inercial O
x_R	Vetor da pose do robô
y_{CIR}	Coordenada y do centro instantâneo de rotação
α	Parâmetro que está associado a um ajuste que considera as particularidades mecânicas na transmissão dos movimentos para as rodas do robô
β_i	Correção horizontal do laser
Δt	Variação do tempo t
ϵ	Ângulo mensurado pelo encoder
ε_i	Correção vertical do laser i
θ	Ângulo de *roll*
θ	Ângulo *pitch*
θ_A^O	Componente de rotação θ do sistema A com respeito ao sistema O
θ_L	Ângulo de rotação do eixo y de um sensor LiDAR
μ_L	Vetor da média da pose do sensor LiDAR
μ_{f_b}	Média de *features* de borda registradas
μ_{f_p}	Média de *features* de plano registradas

σ_{f_b}	Desvio padrão features de borda
σ_{f_p}	Desvio padrão features de plano
ϕ_A^O	Componente de rotação ϕ do sistema A com respeito ao sistema O
φ_A^O	Representação mínima por ângulos de *roll*, *pitch* e *yaw* do sistema A com respeito ao sistema inercial O
ψ	Ângulo *yaw*
ψ_A^O	Componente de rotação ψ do sistema A com respeito ao sistema O
ψ_{max}	Ângulo rotacional total
ψ_L	Ângulo de rotação do eixo z de um sensor LiDAR
ω_R	Velocidade angular do robô em torno do eixo x

Índice

Capítulo 1

INTRODUÇÃO

Entre as principais habilidades dos robôs autônomos, o mapeamento e a estimativa de pose, estão entre as tarefas fundamentais na robótica móvel. Um grande esforço tem sido aplicado pela comunidade acadêmica para alcançar, em tempo real, uma estimativa de localização e mapeamento simultâneos (SLAM) com dados provenientes de sensores visuais ou de sensores LiDAR [1].

Sensores exteroceptivos possuem vantagens claras sobre sensores proprioceptivos em aplicações que demandam interação com o ambiente externo. Enquanto sensores proprioceptivos são limitados à medição de estados internos, como velocidade angular ou carga em rodas, os exteroceptivos coletam informações do ambiente, como distâncias, obstáculos e condições de luminosidade. Em sistemas autônomos, como veículos e robôs, isso é essencial para a navegação em ambientes dinâmicos e imprevisíveis. Tecnologias como câmeras, radares e sensores LiDAR, amplamente utilizadas, permitem uma percepção detalhada e em tempo real, indispensável para decisões precisas e seguras em tarefas de alta complexidade [2].

As técnicas SLAM correspondem a tecnologia de localização ativa, operando independente de sinal externo, incluindo satélite e campos geomagnéticos. O SLAM apresenta desempenho satisfatório para navegação em ambientes internos e externos, sendo essencial, servindo de base e garantia para percepção inteligente de robôs não tripulados [3].

Quando navegando em um ambiente desconhecido, um sistema não tripulado adquire observações da cena através dos sensores embarcados, e o SLAM recupera a posição, altitude e orientação em tempo real através da observação da correlação

entre espaço e tempo, realizando a localização e o mapeamento. Os sistemas SLAM são divididos em duas técnicas principais, de acordo ao tipo de sensor, o Visual SLAM e o LiDAR SLAM. Comparado com o Visual SLAM, o LiDAR SLAM, também chamado de LiDAR 3D tem a vantagem de maior alcance, boa estabilidade, não é afetado por mudanças de iluminação e não apresenta desvio de escala, o que o torna mais popular em aplicações reais de grande escala [3][4].

As estratégias baseadas em fusão sensorial em LiDAR SLAM vêm demonstrando maior precisão da localização e estabilidade, que é o atual foco de pesquisa e tendência de desenvolvimento sobre o tema. O desenvolvimento da tecnologia LiDAR SLAM promoverá amplamente o progresso e os avanços das aplicações de percepção robótica inteligente, como o levantamento 3D e o mapeamento, direção não tripulada, navegação militar e exploração espacial [3].

A abordagem mais utilizada para transformação entre duas leituras de LiDAR sucessivas é a ICP (*Iterative Closest Point*) [5]. Ao encontrar correspondência em um nível pontual, o ICP alinha dois conjuntos de pontos iterativamente até que os critérios de parada sejam satisfeitos. Em uma estimativa de posição em movimento a distorção é removida em duas etapas: primeiramente é realizada uma estimativa de velocidade pelo ICP; e posteriormente a compensação de distorções usando a velocidade computada [6].

Quando o escaneamento inclui uma grande quantidade de pontos, o ICP aumenta o custo computacional de forma a inviabilizar a sua utilização. Algumas variantes do ICP foram criadas com o objetivo de melhorar a sua eficiência e acurácia. Os algoritmos precisam ser simples o suficiente para operar em tempo real, e possuir robustez para tratar os inevitáveis ruídos de medição e erros de modelagem.

Uma das metodologias de 3D LiDAR SLAM mais empregadas atualmente é a denominada LOAM (*Lidar Odometry and Mapping*), que é uma técnica em tempo real baseada no algoritmo ICP [7, 8]. A estratégia realiza a extração de características (*features*) geométricas a partir da classificação dos pontos da nuvem capturada pelo sensor LiDAR, e posteriormente identifica correspondência entre varreduras consecutivas para obter a estimativa de localização [7].

A eficiência do desempenho em tempo real das estratégias LOAM é alcançada por meio de uma abordagem inovadora que divide o problema de estimativa em dois

algoritmos que trabalham em conjunto. Um algoritmo de alta frequência fornece estimativas rápidas da velocidade do sensor, porém com precisão moderada. Paralelamente, um segundo algoritmo opera em baixa frequência, mas com alta precisão, para estimar o deslocamento. As saídas desses dois algoritmos são combinadas para gerar uma única estimativa de movimento precisa e em tempo real [9].

Na navegação autônoma, por exemplo, o SLAM é fundamental para garantir que os robôs possam se locomover com precisão e segurança em ambientes dinâmicos. Seja em ambientes internos ou externos, a capacidade de construir e atualizar mapas em tempo real permite que os robôs adaptem suas trajetórias conforme mudanças no ambiente, evitando colisões e otimizando rotas.

1.1 Motivação

O SLAM é crucial para que robôs móveis e veículos autônomos naveguem de forma precisa em ambientes desconhecidos ou dinâmicos, permitindo a construção de mapas e a localização simultânea dentro deles. A técnica LiDAR SLAM oferece precisão e robustez em ambientes desafiadores, como áreas sem GPS ou ambientes altamente dinâmicos, sendo crucial para o avanço de sistemas autônomos.

Sensores Velodyne LiDAR são populares devido à sua alta densidade de pontos e ampla adoção na indústria. O LOAM é uma técnica de referência no campo do LiDAR SLAM, reconhecida por sua capacidade de realizar odometria e mapeamento em tempo real com alta precisão [7].

Há uma carência de análises sistemáticas que avaliem o desempenho do LOAM em diferentes cenários e condições, o que é fundamental para entender suas limitações e potencialidades [4]. Um estudo comparativo sistemático pode fornecer informações importantes a respeito da robustez em diferentes terrenos e velocidades, eficiência computacional para aplicações em tempo real, impacto da densidade de pontos e ruído nos resultados.

Uma análise aprofundada do desempenho do LOAM com sensores Velodyne pode fornecer *insights* valiosos para pesquisadores e profissionais, auxiliando na escolha de soluções mais adequadas para aplicações específicas. Resultados práticos podem orientar a indústria, especialmente em setores que dependem de sistemas de navegação

precisos, como transporte autônomo e mapeamento 3D.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo desse trabalho é estudar, implementar e analisar o desempenho de estratégias LOAM utilizando sensor LiDAR 3D para estimação da localização e mapeamento simultâneos de ambientes internos aplicado ao problema de navegação de robôs móveis terrestres com rodas.

1.2.2 Objetivos Específicos

A dissertação de mestrado apresenta os seguintes objetivos específicos:

- Estudar as características do mapeamento utilizando o sensor LiDAR 3D Velodyne;
- Implementar as técnicas LeGO-LOAM, A-LOAM e F-LOAM, em ambientes simulados e em ambientes reais;
- Analisar os resultados obtidos para estimativa de pose e geração do mapa com os algoritmos analisados;
- Avaliar o desempenho das técnicas LiDAR SLAM com relação à estimativa de pose instantânea e comparar com o sistema de captura de movimento Optitrack;
- Disseminação dos resultados obtidos, via publicações em revistas e eventos científicos.

1.3 Organização do Documento

Este documento está dividido em mais cinco capítulos. No Capítulo 2 são abordados os aspectos teóricos relevantes para a realização da pesquisa. Dentre estes

aspectos teóricos destacam-se a exposição estado de arte do tema, com a apresentação dos principais trabalhos desenvolvidos com o uso de LiDAR para obtenção de odometria e mapeamento de ambientes, os principais desafios e as tendências de pesquisas sobre o tema.

No Capítulo 3 são apresentadas, de forma detalhada, as três técnicas LiDAR SLAM selecionadas para análise comparativa nese trabalho.

No Capítulo 4 é apresentada a metodologia da pesquisa, os recursos utilizados, as técnicas LiDAR SLAM avaliadas, os métodos de aquisição e análise dos dados.

No Capítulo 5, estão apresentados os resultados da pesquisa, com os achados das simulações realizadas em ambientes modelados no Gazebo e os resultados das simulações realizadas em ambientes reais.

Por fim, no Capítulo 6, são apresentadas as conclusões obtidas a partir dos resultados encontrados e sugestão de trabalhos futuros.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

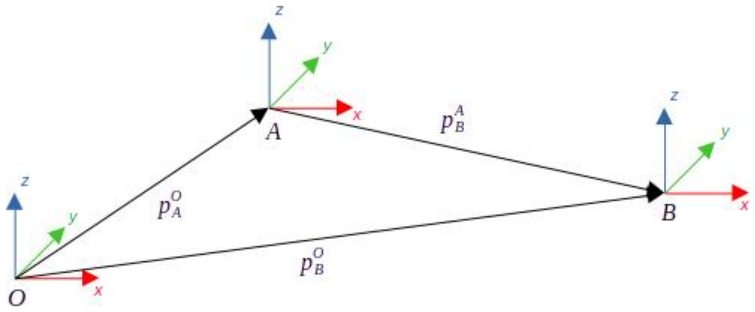
Este capítulo descreve os fundamentos da localização robótica utilizados para descrever a movimentação de robôs móveis, bem como para compreender as técnicas de localização e mapeamento simultâneo baseadas em sensores LiDAR. A Primeira Seção apresenta conceitos gerais da cinemática de corpos rígidos, e em seguida apresentando o modelo cinemático do robô com rodas da Classe *Skid-Steering*, modelo Husky UGV A200, plataforma robótica móvel que será utilizada nesta dissertação. Na segunda Seção são apresentados princípios de localização e mapeamento simultâneos, tipo de sensores, métodos de extração de *features* geométricas do ambiente, estimativa de pose e mapeamento. Por fim, na terceira Seção são apresentados os fundamentos das técnicas LOAM utilizadas neste trabalho.

2.1 Cinemática de Corpos Rígidos

A cinemática é a área da física que se dedica ao estudo do movimento de corpos rígidos, sem levar em conta as forças e torques envolvidos. Ela descreve a posição e a orientação desses corpos, bem como suas derivadas de ordem superior. Um corpo rígido é composto por um conjunto de partículas que mantêm distâncias fixas entre si. Dessa forma, seu movimento pode ser analisado considerando apenas um ponto, como o centro de massa, o que simplifica a interpretação dos modelos matemáticos. Na robótica móvel, a cinemática foca em determinar a posição e a orientação do robô com base no deslocamento de suas rodas.

Para representar a posição e a orientação de um corpo rígido em relação a um

Figura 2.1: Representação gráfica da posição de um corpo rígido acoplado a um sistema de coordenadas A com respeito ao sistema de coordenadas inercial O, e da posição de um corpo rígido em B com respeito ao sistema de coordenadas A e com respeito ao sistema de coordenadas inercial O.



Fonte: Elaborada pelo autor.

sistema de coordenadas inercial O, é necessário atribuir a ele um sistema de coordenadas local. Como as dimensões de um corpo rígido não mudam, é possível simplificar o problema considerando o corpo como um ponto localizado na origem de seu sistema de coordenadas. Dessa forma, a posição do corpo rígido A em relação ao sistema O é expressa por meio de um vetor $p_A^O \in \mathbb{R}$, conforme ilustrado na Figura 2.1 e em (2.1),

$$p_A^O = \begin{bmatrix} p_{A,x}^O \\ p_{A,y}^O \\ p_{A,z}^O \end{bmatrix}, \tag{2.1}$$

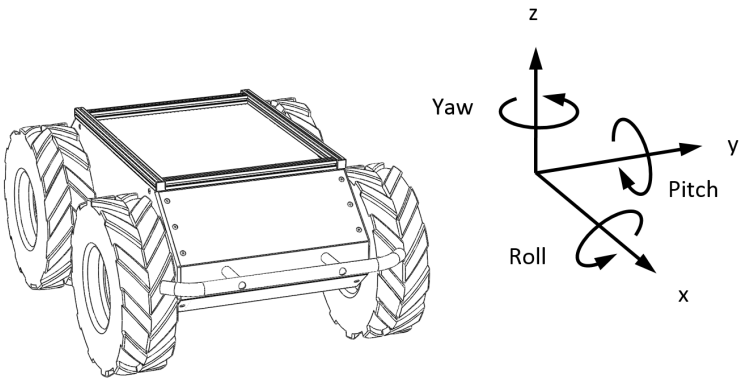
onde as componentes x, y e z do sistema cartesiano são representadas por $p_{A,x}^O$, $p_{A,y}^O$ e $p_{A,z}^O$, respectivamente.

A orientação de um corpo em um sistema de coordenadas A é definida pela rotação do seu sistema de coordenadas com respeito a um sistema de coordenadas de referência O, dada por uma matriz de rotação que pertence ao grupo especial ortonormal de dimensão 3, de forma $\mathbf{R}_A^O \in SO(3)$, conforme apresentado em (2.2),

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} : \mathbf{R}\mathbf{R}^T = \mathbf{I} \; e \; det(\mathbf{R}) = 1\}, \tag{2.2}$$

onde $\mathbf{I} \in \mathbb{R}^{3\times3}$ é a matriz identidade.

Figura 2.2: Representação dos ângulos *roll*, *pitch* e *yaw*, com respeito do sistema de coordenadas do robô móvel Husky UGV.



Fonte: Husky Unmanned Ground Vehicle - User Manual, Clearpath Robotics, 2020.

Na robótica, existem várias maneiras de expressar a orientação de um corpo rígido além da matriz de rotação $\mathbf{R}$. As mais comuns são as representações mínimas através de ângulos de Euler, que decompõem a matriz de rotação em três rotações elementares, não consecutivas, em torno dos eixos do próprio sistema de coordenadas do corpo. Outra forma amplamente utilizada é a representação por ângulos de *roll*, *pitch* e *yaw*, que também decompõem a matriz de rotação em três rotações elementares, mas em relação ao sistema de coordenadas inercial, como ilustrado na Figura 2.2.

A orientação do sistema A em relação ao sistema O, utilizando a representação mínima por meio dos ângulos de *roll*, *pitch* e *yaw*, pode ser descrita conforme apresentado em (2.3),

$$\varphi_A^O = \begin{bmatrix} \phi_A^O \\ \theta_A^O \\ \psi_A^O \end{bmatrix}, \tag{2.3}$$

onde ϕ_A^O representa o ângulo *roll*, θ_A^O representa o ângulo *pitch* e ψ_A^O representa o ângulo *yaw*.

A matriz de rotação $\mathbf{R}_A^O$ equivale a multiplicação das matrizes de rotação elementares $\mathbf{R}_x(\phi)$, $\mathbf{R}_y(\theta)$ e $\mathbf{R}_z(\psi)$, ao longo de x, y e z, respectivamente, conforme apresentado em (2.4):

$$\mathbf{R}_A^O = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi). \tag{2.4}$$

A pose de um corpo rígido em A com respeito ao sistema de coordenadas O pode ser definida conforma apresentado em (2.5):

$$\mathbf{x}_A^O = (\mathbf{p}_A^O, \mathbf{R}_A^O), \tag{2.5}$$

sendo composta pela posição $\mathbf{p}_A^O$ e pela orientação $\mathbf{R}_A^O$.

Podendo também ser definida em função da representação mínima de orientação φ_A^O, conforme apresentado em (2.6):

$$\mathbf{x}_A^O = \begin{bmatrix} \mathbf{p}_A^O \\ \varphi_A^O \end{bmatrix}. \tag{2.6}$$

O corpo rígido B com respeito ao sistema de coordenadas A no sistema inercial O pode ser representado utilizando em (2.7) e em (2.8):

$$\mathbf{p}_B^O = \mathbf{p}_A^O + \mathbf{R}_A^O \mathbf{p}_B^A, \tag{2.7}$$

$$\mathbf{R}_B^O = \mathbf{R}_A^O \mathbf{R}_B^A, \tag{2.8}$$

onde $\mathbf{p}_B^O$ é a posição do ponto B com respeito ao sistema inercial O, e $\mathbf{R}_B^O$ é a matriz de rotação do sistema B para o sistema O.

2.1.1 Modelo Cinemático do Robô *Skid-Steer*

Uma classe de robôs móveis amplamente empregado em robôs móveis, veículos industriais e agrícolas é o *skid-steer*. Essa classe é conhecida por não possuir um sistema de direção convencional; em vez disso, ela controla as velocidades relativas das rodas esquerda e direita para mudar a direção do veículo [10].

Os robôs *skid-steer* se deslocam utilizando um conjunto de 4 ou mais rodas, ou às vezes com esteiras. Esta classe de robôs é fundamentado no controle das velocidades lineares $v_{R,d}$ e $v_{R,e}$, correspondentes aos lados direito e esquerdo do robô, respectivamente.

Os primeiros trabalhos desenvolvidos sobre o estudo de localização em robôs da classe *skid-steer* demonstram que os modelos aplicados em veículos de duas rodas com tração diferencial não poderiam ser utilizados para modelar com precisão o

movimento de um robô *skid-steer*, devido ao deslizamento da esteira e das rodas [11]. Para solucionar este problema Martínez e colaboradores desenvolveram a proposta de aproximar a cinemática de robôs *skid-steer* com base no Centro Instantâneo de Rotação (CIR) [12].

A simplicidade do design mecânico e a propriedade de poder fazer curvas com raio zero tornam os robôs *skid-steer* amplamente utilizados tanto pela comunidade de pesquisa científica quanto na indústria robótica comercial. Entretanto, devido ao alinhamento das rodas ao eixo longitudinal do dispositivo, o escorregamento é uma ocorrência constante ao realizar curvas, o que dificulta a precisão da odometria de rodas e o controle de movimento baseado nesse mecanismo de localização [13].

Na modelagem, o sistema de coordenadas do robô é posicionado no centro da área delimitada pela região de contato no plano, onde o eixo x aponta para a frente do robô. Portanto, ao girar, o CIR do robô pode ser descrito no sistema de coordenadas local pelo ponto $p_{CIR} = (x_{CIR}, y_{CIR})$.

Quando o centro instantâneo de rotação p_{CIR} do robô está sobreposto ao eixo y do sistema de coordenadas do robô, ou seja, x_{CIR}=0, a relação entre as velocidades v_R e ω_R são expressas em (2.9):

$$\begin{bmatrix} v_R \\ \omega_R \end{bmatrix} = \frac{\alpha}{2y_{CIR}} \begin{bmatrix} y_{CIR} & y_{CIR} \\ 1 & -1 \end{bmatrix} \begin{bmatrix} v_{R,d} \\ v_{R,e} \end{bmatrix}, \tag{2.9}$$

sendo que, o parâmetro α está associado a um ajuste que considera as particularidades mecânicas na transmissão do movimento para as rodas, somado aos efeitos provenientes do contato com o terreno, a pressão dos pneus e as possíveis imprecisões na transmissão dos movimentos.

Os parâmetros cinemáticos estabelecidos na primeira equação do modelo podem ser determinados por meio de dois experimentos simples. Para obter o y_{CIR}, conforme (2.10), pode-se realizar uma rotação pura em torno do eixo z do robô, no qual ψ_{max} é o ângulo total rotacionado, conforme:

$$y_{CIR} = \frac{\int v_{R,d}(t)dt - \int v_{R,e}(t)dt}{2\psi_{max}}. \tag{2.10}$$

Neste experimento, as velocidades das rodas do lado direito e esquerdo ($v_{R,d}$ e

$v_{R,e}$) são ajustadas com valores de magnitude idêntica, porém com sinais opostos.

Realizando um percurso reto com uma distância l_{max}, a componente α é determinada com apresentado em (2.11):

$$\alpha = \frac{2l_{max}}{\int v_{R,d}(t)dt - \int v_{R,e}(t)dt}. \tag{2.11}$$

A cinemática baseada em CIR é bastante utilizada baseada em sua simplicidade e viabilidade, especialmente para aplicações robóticas em tempo real. Em robôs *Skid-steer* a interação pista-terreno é excepcionalmente complicada, e a conversão entre leituras do codificador de roda e movimento do robô depende do projeto mecânico, pressão de insuflamento dos pneus, carga no centro de massa, condições do terreno, deslizamento e assim por diante. Ao longo da missão utilizando robôs *skid-steer*, a exemplo de uma simples entrega de objetos, os parâmetros cinemáticos podem ser alterados, o que reduz a precisão da localização [14].

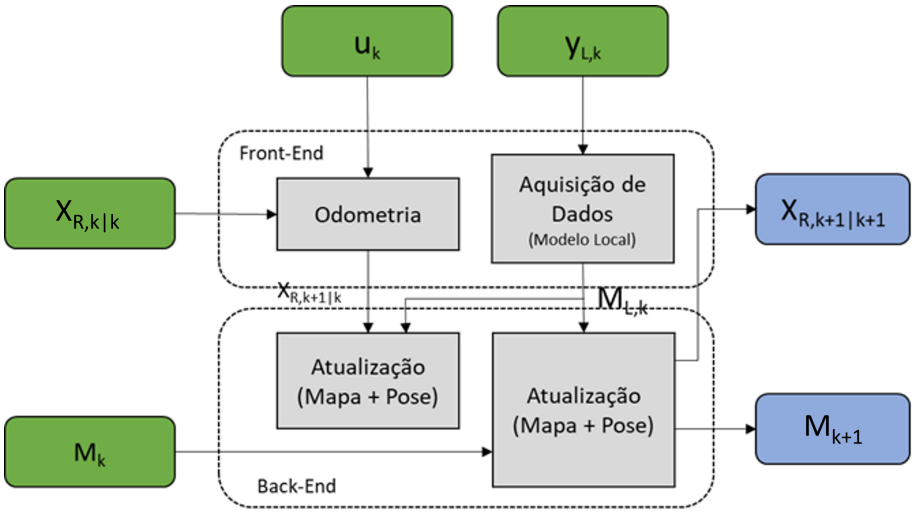
A utilização de modelos cinemáticos, baseados em CIR para ambientes complexos e em movimentações prolongadas, onde as variáveis do terreno podem mudar algumas vezes ao longo da missão, as estratégias utilizando ajuste de posição instantânea com uso de GPS, estratégias baseadas em visão computacional e SLAM apresentam melhores resultados [14].

2.2 Princípios da Localização e Mapeamento Simultâneos

Técnicas de LiDAR SLAM são abordagens para solução do problema do SLAM que utilizam as medidas de sensores *lasers* tanto para estimar a odometria por ego-movimentação entre duas varreduras do sensor, quanto para construir o modelo do ambiente por meio da fusão dos pontos detectado [9]. Este processo combina observações de sensores, pontos de referência e cálculos ao longo do tempo para estimar simultaneamente a localização do robô em um ambiente desconhecido e construir um mapa com características geométricas detectadas.

As técnicas SLAM buscam construir um mapa do ambiente e localizar o dispositivo neste mapa de forma simultânea. Basicamente o problema pode ser separado em 3 etapas [15]. Na primeira etapa, que compõem o *Front-End*, o robô se move por

Figura 2.3: Diagrama básico de funcionamento da estratégia SLAM.



Fonte: Adaptado de Cruz Junior, 2021 [17].

uma distância e a nova pose é calculada e registrada na odometria. Já na segunda etapa, é realizado o alinhamento entre o mapa gerado na estimação atual com o mapa gerado nas etapa anterior, a melhor correspondência é armazenada. E na terceira etapa, a pose com a melhor correspondência se torna a nova pose e o mapa gerado é atualizado em conjunto. Estas duas últimas etapas compõem o *Back-End*. O SLAM ainda possui uma última parte composta pelas técnicas de *loop closure*, que utiliza as informações obtidas em poses anteriores para corrigir a estimativa de pose atual do robô [16]. Na Figura 2.3 pode ser visto o diagrama básico do SLAM com as relações entre entradas e saídas.

O procedimento do SLAM é iterativo e começa a partir da estimativa inicial da localização do robô, podendo ou não envolver um mapa predefinido. Iniciando com a pose estimada do robô, $x_{R,k|k}$, no instante k, o robô avança para o estado a priori $x_{R,k+1|k}$, estimado através de dados de odometria. Esses dados podem ser adquiridos a partir da medição dos *encoders* das rodas, dos comandos de controle, varreduras sequenciais do *laser* e de diferentes métodos combinados. A cada iteração, uma correlação é estabelecida entre as *features* geométricas do mapa M_k identificadas no ambiente, visando determinar a pose do robô $x_{R,k+1|k+1}$. Após a identificação da pose que melhor alinha o mapa local $M_{L,k}$ com o mapa M_k, eles são combinados para formar o mapa M_{k+1}.

A base de um sistema de SLAM está na aquisição de informações do ambiente por meio de sensores, os que são utilizados para identificar *features* do meio, mapear espaço e localizar o dispositivo. Nas técnicas de LiDAR SLAM, os dados provenientes de sensores a em laser são processados na forma de nuvens de pontos, previamente tratadas por um sistema de aquisição de dados.

2.2.1 Sensores LiDAR 3D

Sensores LiDAR são dispositivos que capturam as *features* do ambiente por meio da detecção de feixes de luz medindo o tempo de voo da luz. Essa tecnologia é chamada de ToF (*Time of Flight*) e mede o tempo que o pulso de luz, emitido na forma de um feixe estreito, gasta para ser refletido por uma superfície e retornar. A distância percorrida pode ser calculada por: $d = \frac{1}{2}c\Delta t$, onde d é a distância do objeto, c é a velocidade média da luz e Δt é o tempo entre a emissão do feixe e a sua recepção pelo sensor.

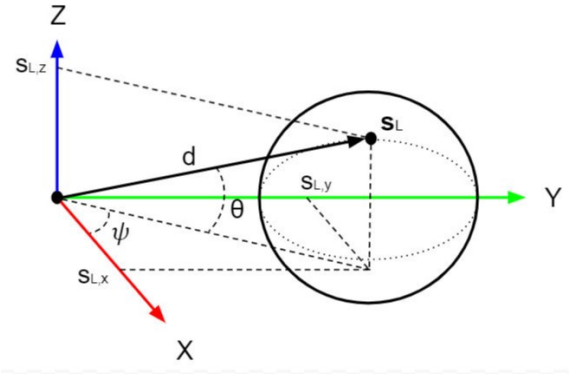
Os sensores *scanning* LiDAR 3D escaneiam múltiplos pontos do ambiente. Devido a sua capacidade de gerar nuvens densas, este são muito utilizados na robótica para gerar modelos tridimensionais aplicados a técnicas de SLAM 3D, como as estudadas nesta dissertação. Estes sensores possuem uma distribuição de *lasers* na vertical que varia de 16 a 128 feixes, como em modelos mais recentes como o Ouster OS2 e o Velodyne VLS-128, e realizam uma varredura de 360° com uma resolução de aproximadamente 0,1° com um alcance de 70 a 300 metros.

Os dados coletados pelos sensores LiDAR 3D são apresentados em diferentes formatos que dependem do fabricante. Existem dois formatos para representação das coordenadas LiDAR (L) que são imagens 3D representadas por $s_L = [i\ j\ d(i,j)]^T$ onde $d(i,j)$ é a distância de um ponto expresso em coordenadas do pixel (i,j), e em coordenadas cartesianas dadas por $s_L = [s_{L,x}\ s_{L,y}\ s_{L,z}]^T$. Portanto, o conjunto de pontos de uma varredura de um sensor LiDAR pode ser organizado em uma lista denominada nuvem de pontos, descrita em (2.12):

$$s_L^{(1:m)} = (s_L^{(1)}, ..., s_L^{(m)}), \tag{2.12}$$

onde m é igual a quantidades de pontos detectados.

Figura 2.4: Coordenadas Esféricas para sensor LiDAR 3D.



Fonte: Cruz Junior, 2021 [17].

As informações 3D coletadas dos ambientes pelos sensores LiDAR são mapeadas para o sistema de coordenadas do sensor segundo a geometria do dispositivo. Um sensor *scanning* LiDAR geralmente possui seu sistema de coordenadas com o eixo x apontando para a frente do sensor e o eixo z para cima, onde os dados coletados do ambiente são expressos em coordenadas esféricas (ψ_L, θ_L, d), como ilustra a Figura 2.4. Os pontos detectados por esse tipo de sensor podem ser convertidos como uma imagem de profundidade representando os ângulos ψ_L e θ_L como coordenadas longitudinais e latitudinais, e em coordenadas cartesianas dadas em (2.13):

$$\begin{bmatrix} s_{L,x} \\ s_{L,y} \\ s_{L,z} \end{bmatrix} = \begin{bmatrix} d\cos\psi_L \cos\theta_L \\ d\,\mathrm{sen}\,\psi_L \cos\theta_L \\ d\,\mathrm{sen}\,\theta_L \end{bmatrix}. \tag{2.13}$$

O sensor LiDAR do tipo Velodyne VLP-16 é pequeno e compacto, com desempenho e potência otimizados para uma variedade de aplicações desde automotiva, mapeamento, robótica, segurança e infraestrutura. O sensor VLP-16 suporta 16 canais, gerando aproximadamente 300.000 pontos/segundo, com campo de visão horizontal de 360° e campo de visão vertical de 30° (+15° a - 15°), e capacidade de alcance de até 100 metros, com resolução de 5 a 20 Hz.

Cada uma das 16 linhas de escaneamento vertical registra até 1875 pontos em um décimo de segundo, o que corresponde a uma resolução angular horizontal de 0,2° para o campo de visão de 360°. Cada um dos 16 *lasers* do sensor é direcionado de forma independente e possui um conjunto exclusivo de parâmetros de medição.

O modelo matemático do VLP-16, para o cálculo das coordenadas x, y e z é dado em (2.14), onde s^i é o fator de escala da distância do *laser* i; D_o^i é o deslocamento de distância do *laser* i; δ_i é a correção vertical do *laser* i; β_i é a correção horizontal do *laser* i; H_o^i é o deslocamento horizontal da origem da estrutura do sensor para o *laser* i; V_o^i é o deslocamento vertical da origem da estrutura do sensor para o *laser* i; R_i é a distância bruta para o *laser* i; e ε é o ângulo mensurado pelo *encoder*. A matriz para cálculo das coordenadas do VLP-16 pode ser calculada conforme:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (s^i.R_i + D_o^i).\cos(\delta_i).[\text{sen}(\varepsilon).\cos(\beta_i) - \cos(\varepsilon_i)] \\ -H_o^i.[\cos(\varepsilon).\cos(\beta_i) + \text{sen}(\varepsilon).\text{sen}(\beta_i)] \\ (s^i.R_i + D_o^i).\cos(\delta_i).[\cos(\varepsilon).\cos(\beta_i) + \text{sen}(\varepsilon_i)] \\ +H_o^i.[\text{sen}(\varepsilon).\cos(\beta_i) - \cos(\varepsilon).\text{sen}(\beta_i)] \\ (s^i.R_i + D_o^i).\text{sen}(\delta_i) + V_o^i \end{bmatrix}. \tag{2.14}$$

Os seis primeiros parâmetros são utilizados para calibrar o sistema, e podem ser encontrados na ficha de dados do sensor LiDAR, já os parâmetros R_i e o ε são provenientes coletados durante a medição.

A Figura 2.5a apresenta uma fotografia de uma cena escaneada com o mesmo ângulo de visão, e a Figura 2.5b mostra o ponto típico de nuvem de pontos produzida pelo VLP-16 com destaque vertical e resoluções angulares horizontais. A baixa resolução vertical limita a aplicação do VLP-16 a aplicações terrestres [18].

2.2.2 Extração de *Features* Geométricas do Ambiente

Os sensores LiDAR detectam pontos que oferecem informações sobre as distâncias no espaço mapeado, a partir das medições realizadas. No entanto, para que essas informações sejam empregadas na resolução do SLAM, é necessário extrair certos recursos do ambiente, conhecidas como *features* geométricas. As técnicas de LiDAR SLAM frequentemente utilizam pontos 3D, planos, linhas 3D e segmentos de reta como *features* para o mapeamento do ambiente.

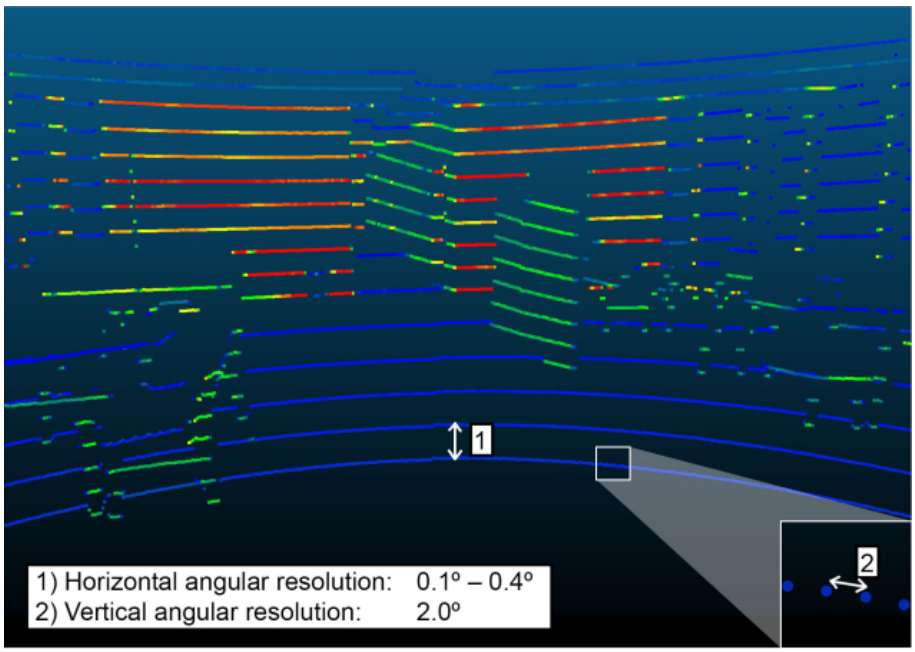
As *features* representadas por pontos 3D descrevem saliências como quinas e portas, encontro de duas ou três superfícies e protuberâncias, assim como podem ser usados também para identificar superfícies planas como pisos, paredes, tetos e objetos.

Uma solução para classificação de pontos em bordas ou planos é encontrada em

Figura 2.5: (a) Fotografia do ambiente escaneado; (b) escaneamento típico criado pelo VLP-16, as cores representam a intensidade de retorno de sinal.

(a)

(b)



Fonte: Bula et al, 2020

Zhang and Singh (2017) [8], que utilizam uma equação de suavidade normalizada, definida em (2.15), no qual $s_{L,k}^{(i)}$ é o vetor da coordenada 3D de um ponto i na varredura k do sensor LiDAR, $s_{L,k}^{(j)}$ é o vetor de um ponto j da varredura k e, S_L é o conjunto de pontos nas proximidades do ponto i. Desta forma, os pontos são classificados como bordas quando o valor s é maior que um limite máximo, ou pertencentes a um plano quando o valor s é menor que um limite mínimo, dado por:

$$s = \frac{1}{|S_L| \; ||s_{L,k}^{(i)}||} || \sum_{j \in S_L, j \neq i} (s_{L,k}^{(i)} - s_{L,k}^{(j)}) ||. \tag{2.15}$$

2.2.3 Estimativa de Pose e Mapeamento

A estimativa de pose corresponde ao cálculo da posição e orientação de um corpo com respeito a um sistema de coordenadas inercial. Nas técnicas LiDAR SLAM, a pose é calculada pelo alinhamento entre duas nuvens de pontos escaneadas pelo sensor ou entre uma nuvem de pontos e o mapa. Três soluções possíveis são a estimação de pose por um conjunto de pontos, por plano e por linhas 3D, ou por segmentos de reta. As três são resolvidas minimizando uma função de custo onde as *features* atuam como restrições.

Uma solução para estimação de pose dada por um conjunto de pontos é o algoritmo *Iteractive Closest Points* (ICP) [19]. Esse método é iterativo e usa uma distância máxima d_{max} para analisar os pontos mais próximos, determinando um conjunto de pontos parcialmente sobrepostos. Dados dois conjuntos de pontos 3D independentes a pose relativa $x_L = (p_L, R_L)$ é determinada minimizando a função de custo, conforme (2.16),

$$e(p_L, R_L) = \sum_j ||s_{L,k-1}^{(j)} - (R_L s_{L,k}^{(j)} - p_L)||_N^2, \tag{2.16}$$

no qual $e(p_L, R_L)$ equivale ao erro a ser minimizado, $s_{L,k}^{j}$ é a coordenada do ponto j para a varredura k, e N é uma matriz positiva definida de ganhos.

As entradas do algoritmo são dois conjuntos de pontos que, para o caso da estimação entre dois escaneamentos, são os mapas locais gerados por varreduras consecutivas, $s_{L,k-1}^{(1:m)}$ e $s_{L,k}^{(1:m)}$. A saída é a pose estimada entre os dois conjuntos, dada

uma matriz de rotação R_L e um vetor de posição p_L. Para cada ponto $s_{L,k}^{(j)}$ pertencente ao conjunto $S_{L,k}$ o algoritmo encontra o ponto mais próximo dentro os pontos da sua vizinhança rastreados dentro de um raio d_{max}. Na sequência é calculada a transformação (p_L, R_L) que minimiza o erro dado pela função custo conforme a (2.16).

Para isso as médias $\mu_{L,k-1}$ e $\mu_{L,k}$ são determinadas para cada conjunto. Sendo assim, são construídas então duas matrizes, no qual Q e D, (2.17) e (2.18), respectivamente:

$$Q = [\{s_{L,k-1}^1 - \mu_{L,k-1}\}...\{s_{L,k-1}^m - \mu_{L,k-1}\}], \tag{2.17}$$

$$D = [\{s_{L,k}^1 - \mu_{L,k}\}...\{s_{L,k}^m - \mu_{L,k}\}], \tag{2.18}$$

formadas pelos vetores dados pela diferença entre os pontos de cada varredura e sua respectiva média, gerando um conjunto centralizado.

Para o cálculo da matriz de rotação R_L deve ser realizada a decomposição de valores singulares, conforme (2.19):

$$svd(DQ^T) = UEV^*, \tag{2.19}$$

no qual U é uma matriz unitária, E é uma matriz diagonal retangular e V^* é a matriz conjugada transposta da matriz unitária V.

Portando, a matriz de rotação R_L é calculada por (2.20).

$$R_L = UV^* \tag{2.20}$$

Sendo assim, a translação p_L pode ser calculada usando os valores das médias de cada conjunto $\mu_{L,k-1}$ e $\mu_{L,k}$ com a matriz de rotação R_L dada por (2.21):

$$p_L = \mu_{L,k} - R\mu_{L,k-1} \tag{2.21}$$

Estimada a transformação (p_L, R_L), está é aplicada ao conjunto $s_{L,k}^{(1:m)}$ e então é calculado o erro quadrático, conforme (2.22).

$$||e_{k-1}(p_L, R_L) - e_k(p_L, R_L)|| < e_{min} \tag{2.22}$$

Quando essa diferença é menor que um certo limiar inferior pré-determinado, o algoritmo finaliza.

O mapeamento de ambientes escaneados por um sensor LiDAR nada mais é que o registro de múltiplas varreduras ou, dependendo da característica extraída, de múltiplos conjuntos de pontos, linhas ou planos. Para realizar a construção do mapa, as nuvens de pontos escaneadas devem estar expressas no sistema de coordenadas inercial. Essa transformação é dada pela pose do dispositivo robótico que pode ser estimada pela odometria LiDAR.

Com o sensor LiDAR rigidamente acoplado ao dispositivo, os pontos s_L podem ser representados com respeito ao sistema de coordenadas do robô segundo (2.23):

$$s_L^R = H_L^R s_L, \tag{2.23}$$

onde H_L^R é a matriz de transformação homogênea do sensor para o sistema de coordenadas do robô, e s_L^R são os pontos do LiDAR descritos com respeito ao sistema de coordenadas do robô.

A partir da pose estimada com respeito ao sistema inercial O expressa por uma matriz de transformação homogênea H_R^O é possível relacionar os pontos lidos pelo sensor ao mapa por (2.24):

$$s_L^O = H_R^O H_L^R s_L. \tag{2.24}$$

Os pontos obtidos pelo escaneamento do ambiente, após transformados, são associados ao mapa aplicando técnicas de registro de nuvem de pontos como algoritmo ICP. Porém ao invés de utilizar duas varreduras consecutivas como entradas do algoritmo, é utilizado o modelo do ambiente e a varredura atual do sensor LiDAR. Desta forma a pose do dispositivo é determinada com respeito ao sistema de coordenadas inercial e os pontos escaneados podem ser registrados no mapa.

Capítulo 3

ANÁLISE DAS TÉCNICAS LIDAR SLAM (LeGO-LOAM, A-LOAM e F-LOAM

Este capítulo descreve a implementação de três variantes do algoritmo LOAM-Velodyne, que é uma versão adaptada para sensor Velodyne do algoritmo desenvolvido por Zhang e Singh (2017) [8], utilizados para estimação da odometria e geração do mapeamento. As técnicas descritas a seguir são a LeGO-LOAM, A-LOAM e F-LOAM. Cada uma dessas abordagens oferece estimativas de pose em tempo real, cria mapas representativos e de baixo consumo computacional, sendo compatíveis com o processamento no computador embarcado ao robô.

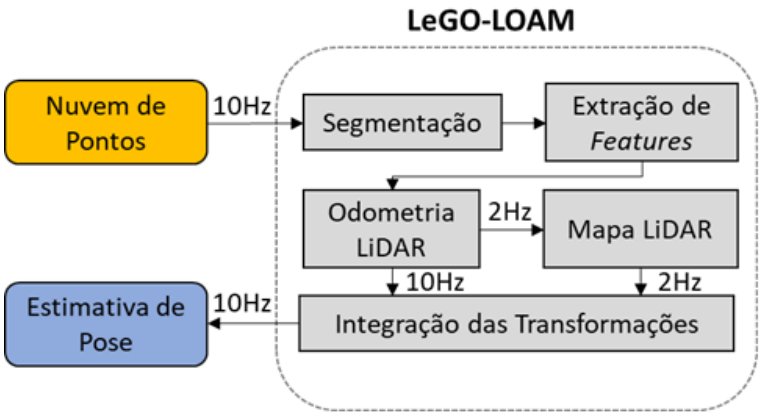
3.1 LeGO-LOAM

A técnica LeGO-LOAM (*Lightweight and Ground Optimized LOAM*) apresentada por Shan and Englot (2018) [9] é uma versão mais leve do LOAM, otimizada para veículos terrestres que obtém estimativas de pose em tempo real, podendo ser utilizada em um sistema embarcado com menor consumo de energia. As principais diferenças entre as técnicas são a opção de uso de *loop closure*, e a estimação da pose do robô. A metodologia fornece as estimações de odometria LiDAR a cada 10 Hz, e o mapa é gerado a cada 2 Hz. A técnica LeGO-LOAM possui 5 módulos sendo eles: Segmentação, Extração de *Features*, Odometria LiDAR, Mapeamento LiDAR

e integração das transformações [1].

A Figura 3.1 descreve o esquema de funcionamento da técnica, apresentando a entrada como a nuvem de pontos e saída a estimativa de pose, além do mapa gerado.

Figura 3.1: Esquema de funcionamento da técnica LeGO-LOAM.



Fonte: Adaptado de Shan and Englot, 2018.

A nuvem de pontos do sensor LiDAR inicialmente é processada no módulo de Segmentação, onde são filtrados os *outliers*. É coletada uma única nuvem de pontos de varredura, que é projetada em uma faixa de imagem para segmentação, agrupando os pontos e vários *clusters*.

Como terrenos inclinados são comuns em muitos ambientes o solo não pode ser assumido como plano. Uma avaliação de coluna de faixa de imagens é realizada para estimar o plano de solo, e os pontos classificados como pontos de solo são excluídos da segmentação [9]. A proposta é reduzir o número de pontos a serem utilizados no processo de extração de *features*, para otimização do desempenho da estratégia.

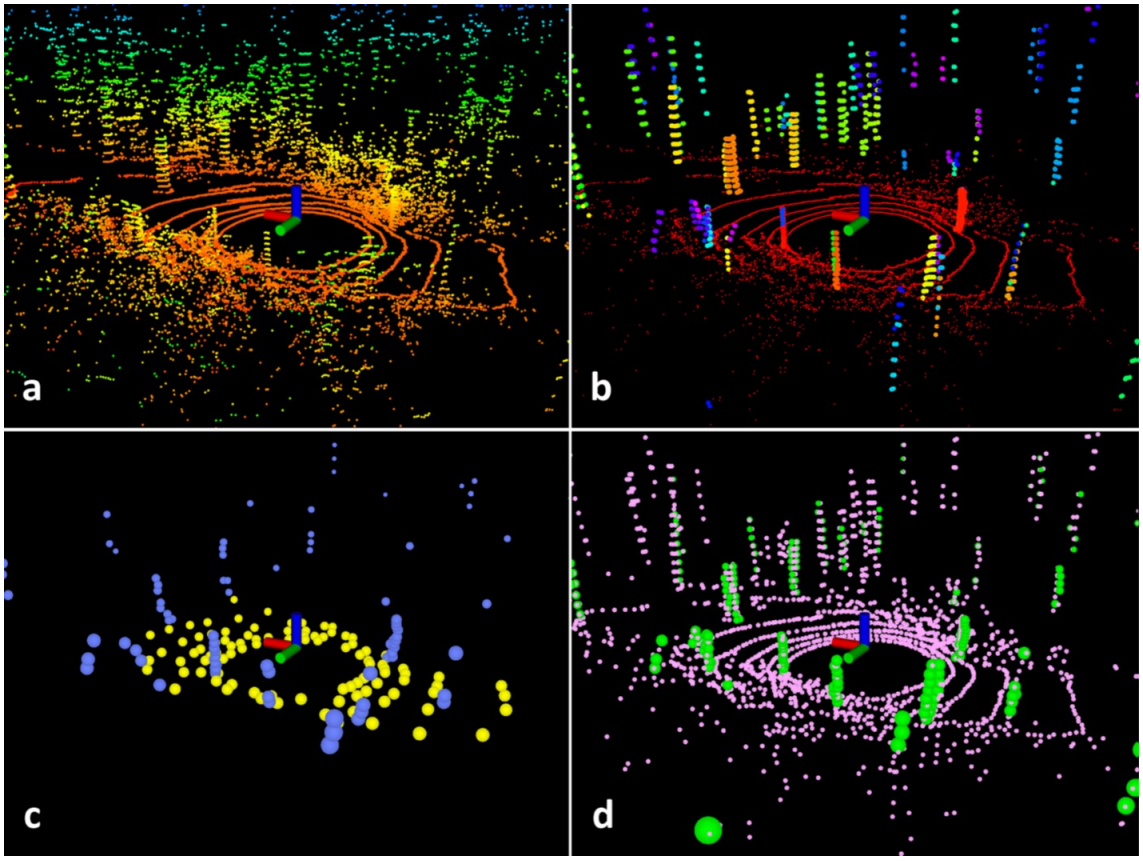
Pontos do mesmo *cluster* recebem o mesmo rótulo. Os *clusters* com menos de 30 pontos são excluídos da análise, de forma a garantir um desempenho rápido e confiável [9].

Em seguida, os dados são enviados para o módulo de Extração de *features* geométricas, onde são separados em dois conjuntos, um para pontos presentes em quinas e bordas e outros para pontos presentes em planos. O método utilizado para extração

[1] O pacote LeGO-LOAM está disponível em https://github.com/RobustFieldAutonomyLab/LeGO-LOAM

de *features* é similar a estratégia LOAM utilizado por Zhang e Shingh [8], com a diferença de utilizar de não utilizar dados brutos, utilizando dados segmentados. A Figura 3.2 apresenta o processo de extração de *features* a partir da nuvem de pontos coletada.

Figura 3.2: Processo de extração de *features* da técnica LeGO-LOAM. A nuvem de pontos original é demonstrada em (a). Em (b) os pontos vermelhos correspondem a detecção de solo. Em (c) os pontos azuis e amarelos correspondem as *features* de borda e de plano, na varredura atual, respectivamente. Em (d) os pontos verdes e rosas representam as *features* de borda e plano, da varredura final, respectivamente.



Fonte: Adaptado de Shan and Englot, 2018.

O módulo de odometria estima a movimentação do sensor entre duas varreduras consecutivas. A estimativa é gerada por correspondência entre as *features* identificadas no passo anterior.

O método de Levenberg-Marquardt (L-M) é um algoritmo de otimização numérica amplamente utilizado para resolver problemas de mínimos quadrados não lineares.

Em particular, é eficaz em situações onde a solução precisa minimizar a diferença (ou erro) entre dois conjuntos de dados, o que é frequentemente o caso em registro de varreduras de sensores, como LiDAR, câmeras, ou outros dispositivos de medição [19].

O método L-M é aplicado em duas etapas, sendo $[t_z, \theta_{roll}, \theta_{pitch}]$ estimada pelas *features* planares na varredura atual e seu correspondente na varredura anterior, e $[t_x, t_y, \theta_{yaw}]$ é estimada em seguida, pelas *features* de borda da varredura atual e seu correspondente na varredura anterior. Finalmente, a transformação 6D entre duas varreduras consecutivas é encontrada pela fusão entre $[t_z, \theta_{roll}, \theta_{pitch}]$ e $[t_x, t_y, \theta_{yaw}]$, o que reduz o custo computacional em 35%, conforme descrito por Shan and Englot (2018) [9].

Os dados referentes a pose, bem com os conjuntos de pontos selecionados, são enviados para o módulo de Mapeamento LiDAR, onde ao invés de salvar um único mapa de nuvem de pontos, são salvos conjuntos de *features* individuais, que são registrados utilizando um grafo de poses e fundidos ao mapa pela técnica Bayes-Tree [20]. Por fim, as poses geradas pela Odometria LiDAR e Mapeamento LiDAR são integradas no módulo de Integração das Transformações e a pose final é estimada.

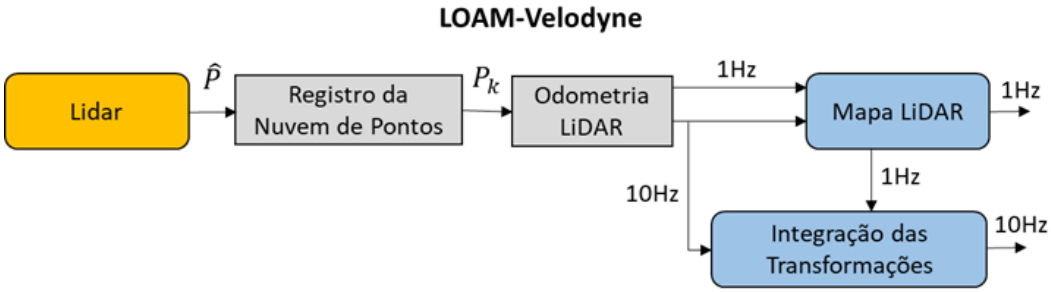
3.2 A-LOAM

A-LOAM (*Advanced LOAM*) é uma implementação avançada da técnica LOAM-Velodyne [2], com o uso de *Eigen* e *Ceres Solver* para simplificar a estrutura do código. O código é direto e simples sem derivações matemáticas complexas e operações redundantes [21].

A técnica LOAM-Velodyne é uma versão adaptada para sensor Velodyne da proposta desenvolvida por Zhang e Singh (2014) [7], que possui baixo erro de desvio de odometria e dispensa o uso de técnicas de loop closure. O LOAM-Velodyne fornece as estimações de odometria LiDAR a cada 10 Hz, e o mapa é gerado cada 1 Hz. A metodologia é subdivida em 4 etapas: Registro da Nuvem de Pontos, Odometria LiDAR, Mapeamento LiDAR e integração de transformações, conforme ilustrado na Figura 3.3.

[2] O pacote A-LOAM está disponível em https://github.com/HKUST-Aerial-Robotics/A-LOAM

Figura 3.3: Esquema de funcionamento da técnica LOAM-Velodyne.



Fonte: Adaptado de Zhang and Singh, 2014.

No módulo Registro da Nuvem de Pontos o algoritmo permite o ajuste da distorção da nuvem causada na ego-movimentação por meio dos dados fornecidos pela IMU, onde as *features* geométricas são extraídas destacando as bordas e os planos de acordo a equação da suavidade. No módulo Odometria LiDAR, o método de Levenberg-Marquardt (L-M) é utilizado em uma única etapa para determinar a pose 6D do dispositivo, ou seja, a posição e orientação do sensor no espaço tridimensional. A pose 6D é calculada considerando um sistema de equações que incorpora tanto as restrições de borda quanto as restrições de plano, elementos fundamentais no processo de alinhamento de varreduras LiDAR.

Para o mapeamento LiDAR é utilizando o filtro voxel-grid em uma estrutura do tipo KD-Tree, onde o alinhamento do mapa é feito utilizando um método similar ao ICP [21]. A técnica L-M é utilizada para otimizar o processo de cálculo da pose entre dois conjuntos de dados 3D, compostos por pontos de bordas e planos, dentro de uma área restrita (neste caso, um volume de $10m^3$ do mapa). A área selecionada é escolhida com base na proximidade da última posição estimada do sensor, garantindo que o cálculo da pose seja feito em uma região do mapa onde os dados de varredura mais recentes têm maior relevância. No módulo de Integração das Transformações, as poses retornadas pelos módulos de Odometria LiDAR e Mapeamento LiDAR são integrados gerando uma pose de saída.

3.3 F-LOAM

Embora os trabalhos existentes sobre LiDAR SLAM alcancem boa performance na avaliação com uso de banco de dados público, ainda existem algumas limitações para aplicações práticas. Uma das principais limitações é a robustez em diferentes ambientes, do ambiente indoor para o ambiente outdoor e do ambiente estático para o dinâmico.

Em muitos robôs, como os UAVs (*Unmanned Aerial Vehicle*), os recursos computacionais são limitados, onde a unidade de processamento deve ter alta frequência de localização e planejamento de caminho ao mesmo tempo.

Na estratégia LOAM, que extrai as *features* geométricas de plano e de bordas, e calcula a pose minimizando a distância de ponto para o plano e ponto para a borda. Entretanto, ambas, a compensação de distorção e a odometria a *laser* requer um cálculo iterativo, que continua sendo computacionalmente caro.

A estratégia F-LOAM (*Fast LiDAR Odometry and Mapping*), que busca uma solução de menor custo computacional e em tempo real, utilizando um método não iterativo, de dois estágios de compensação de distorção. No primeiro estágio a velocidade linear e angular é assumida como constante, por um curto intervalo de tempo, para prever a movimentação e corrigir as distorções. No segundo estágio as distorções são recalculadas, após a estimativa de pose e as *features* geométricas não distorcidas são atualizadas no mapa final [3].

A estratégia permite alcançar um desempenho em tempo real em 20 Hz em uma unidade de computação embarcada de baixo consumo de energia. Os experimentos realizados confirmaram que o método de compensação de dois estágios proporciona precisão semelhante, com um custo computacional consideravelmente inferior.

Para estimativa de pose as *features* de borda e planas, não distorcidas, são alinhadas com o mapa global de *features*, pois o mapa de *features* de borda e de plano são atualizados e mantidos separadamente. Para reduzir o custo computacional, os mapas de borda e de planos são armazenados em uma árvore K-D.

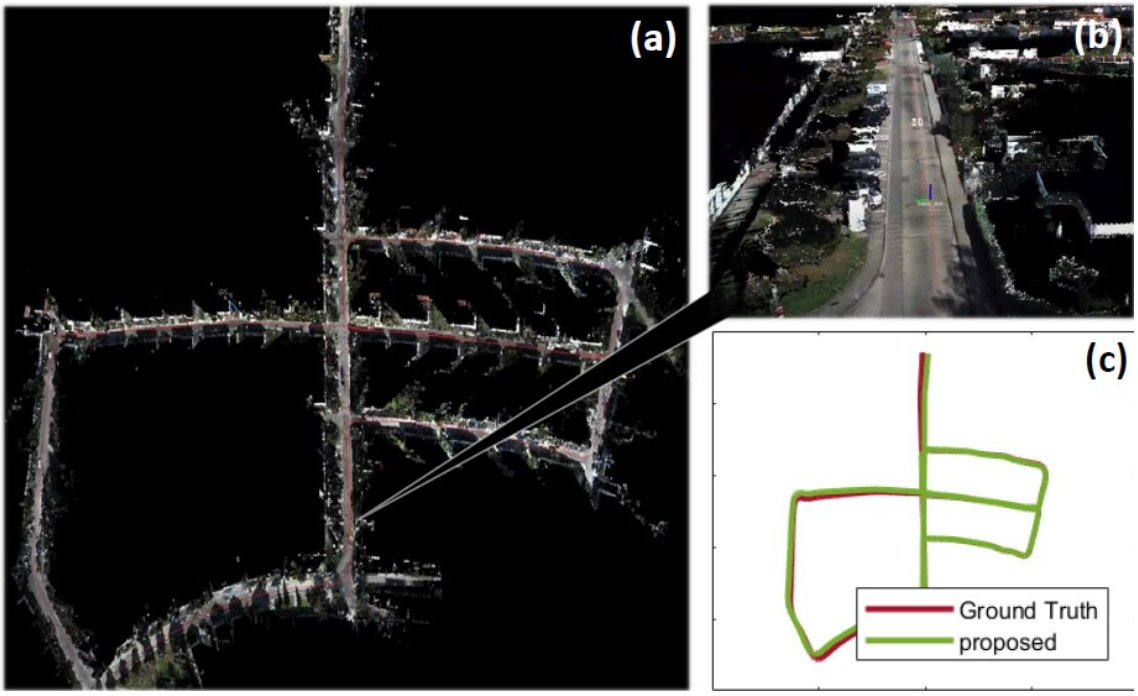
Similar ao descrito por Zhang e Suganthan [22], a linha e plano global podem ser estimados pela coleta de pontos próximos dos mapas de *features* de borda e de plano.

[3] O pacote F-LOAM está disponível em https://github.com/wh200720041/floam

Quando os pontos estão distribuídos em uma linha, a matriz de covariância contem um autovalor que é muito maior, já a norma do plano global possui um autovetor com autovalor associado menor. Esta correspondência é utilizada para estimar a pose ideal entre a referência atual e o mapa global, minimizando a distância entre os pontos característicos e o mapa de global (bordas e planos).

A Figura 3.4 abaixo, ilustra o resultado de de um mapeamento 3D e odometria utilizando o algoritmo F-LOAM. Os resultados foram gerados a partir do dados disponibilizados pelo KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute), que consiste em um banco de dados aberto, amplamente utilizado para análise de algoritmos de robótica e direção autônoma.

Figura 3.4: Exemplo de aplicação do algoritmo F-LOAM com dados abertos disponibilizados no site do KITTI. (a) mapa gerada pelo algoritmo F-LOAM. (b) reconstrução 3D da rua integrada com a visão da câmera. (c) plotagem gráfica da trajetória gerada pelo algoritmo F-LOAM comparado com o *ground truth*.



Fonte: Adaptado de Wang e colaboradores, 2021.

3.4 Comparação entre as principais características dos algoritmos estudados

A Tabela 3.1 abaixo apresenta as principais características dos algoritmos LeGO-LOAM, A-LOAM e F-LOAM, que são amplamente utilizados para mapeamento e localização simultânea baseados em LIDAR [23]. Cada algoritmo apresenta um equilíbrio diferente entre precisão, eficiência e adequação a aplicações específicas. As escolhas dependem do ambiente e dos recursos computacionais disponíveis.

Tabela 3.1: Tabela comparativa entre as principais características dos algoritmos LeGO-LOAM, A-LOAM e F-LOAM.

Algoritmo	Descrição Geral	Pontos Fortes	Limitações	Aplicações Típicas
LeGO-LOAM	Versão otimizada do LOAM para ambientes com terrenos planos. Segmenta pontos do solo para facilitar o cálculo da pose.	Redução de complexidade computacional; Adequado para robôs terrestres.	Pior desempenho em terrenos irregulares ou ambientes complexos.	Veículos autônomos e robôs móveis terrestres.
A-LOAM	Versão aprimorada do LOAM, com melhorias na eficiência e na estimativa da odometria. Usa melhor segmentação de características.	Melhor precisão na odometria em relação ao LOAM; mais eficiente em ambientes dinâmicos.	Mais complexo de configurar; requer hardware potente.	Navegação em drones e robôs em cenários mistos.
F-LOAM	Focado em alto desempenho e eficiência em tempo real. Reduz a carga computacional com algoritmos mais leves.	Adequado para sistemas embarcados; processamento em tempo real.	Pode sacrificar precisão em comparação com o LOAM original.	Aplicações em robôs de baixo custo e drones de alta velocidade.

Fonte: Adaptado de Ren e colaboradores, 2022.

Capítulo 4

METODOLOGIA

Este Capítulo descreve as metodologias aplicadas para a análise do desempenho das técnicas LiDAR SLAM selecionadas, sendo apresentado os resultados obtidos em ambientes simulados e nos nos experimentos reais. Tanto para a análise de ambientes simulados quando para análise de ambientes reais inicialmente foram apresentados os recursos utilizados. Em seguida, é realizada uma descrição dos ambientes estudados e a justificativa para a sua escolha. Na sequência são apresentadas as técnicas LiDAR SLAM avaliadas, finalizando com os métodos de aquisição de dados e análise de resultados.

4.1 Análise em Ambientes Simulados

4.1.1 Recursos Utilizados

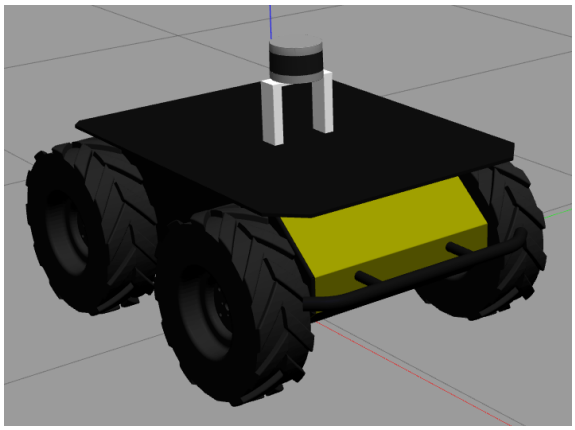
Para realização dos experimentos, a partir dos dados simulados, foi utilizado o *framework* ROS (*Robot Operating System*), distribuição Noetic, juntamente com o simulador Gazebo e o visualizador 3D RViz. Foi utilizado o pacote Clearpath Husky [1], desenvolvido pela fabricante do robô Husky, a Clearpath Robotics. O Husky é um robô móvel da classe *Skid-Steer*, cuja movimentação se baseia no controle diferencial das rodas. Apesar de sua mecânica simples e robusta, esse tipo de sistema apresenta desafios significativos para odometria e controle de movimento, especialmente em terrenos irregulares. O deslizamento das rodas e a deformação do solo geram discrepâncias entre o modelo cinemático ideal e o comportamento real do

[1] https://github.com/husky/husky.git

robô, dificultando a previsão precisa da trajetória.

No modelo do robô disponível no pacote Clearpath Husky estão disponíveis alguns sensores LiDAR 2D e 3D e câmeras de visão computacional. Os sensores podem ser utilizados de forma isolada ou em conjunto, sendo possível posicioná-los em qualquer posição com relação a estrutura do robô. A Figura 4.1 ilustra o modelo do Husky no gazebo, com o sensor LiDAR 3D Velodyne VLP-16 posicionado sobre a base superior do robô.

Figura 4.1: Modelo do robô móvel Husky acompanhado do sensor LiDAR SLAM Velodyne VLP 16.



Fonte: Elaborada pelo autor.

Foi escolhido para realização das análises o sensor LiDAR 3D Velodyne VLP-16, pelo fato de se tratar de um sensor com campo de visão horizontal de 360°, distância de medição de até 100 metros, com acurácia em torno de +/- 3 cm, compacto, com elevada autonomia, além de ser amplamente utilizado em aplicações reais e em trabalhos científicos para análise de desempenho de algoritmos LOAM [4][9].

O modelo físico do robô foi descrito utilizando o formato de descrição de robô URDF (*Unified Robot Description Format*), que foi implementado em um arquivo XML que contém as especificações técnicas dos componentes do robô, incluindo links, juntas, atuadores e sensores. Cada componente do robô é descrito com um sistema de coordenadas próprio, que é definido em relação ao sistema de coordenadas principal, chamado de base_link. A relação entre esses componentes é representada por uma árvore de transformações, que foi implementada utilizando a ferramenta TF do ROS.

A frequência de varredura configurada no Velodyne VLP-16 foi de 10 Hz para todas as situações simuladas, pois os algoritmos estudados utilizam uma frequência de aquisição de dados de 10 Hz, a exceção para o F-LOAM, que suporta aquisições de até 20 Hz. As etapas de segmentação, estimação de odometria e mapeamento apresentam taxas de processamento diferentes entres as técnicas estudadas, sendo um dos objetivos de estudo verificar a influência do tempo de processamento e o erro de estimação encontrado.

Para aquisição de dados o robô móvel Husky posicionado inicialmente na posição zero em x e y, e com o sensor LiDAR SLAM posicionado na base superior, localizada a 0,081 metros no eixo x e 0,409 metros do eixo z, do centro de massa do robô, que é utilizado como referência para o sistema de coordenadas móvel do robô.

O robô foi deslocado por 4 ambientes com características diferentes, com o objetivo de avaliar a influência do número de objetos e detalhes presentes na cena, e os resultados da estimação de odometria e mapeamento. A aquisição de dados foi realizada pela coleta do arquivo .bag durante a movimentação do robô com velocidade fixa em 0,5 m/s para os 2 corredores retos e variável para os outros 2 ambientes, nos quais o robô foi teleoperado com o objetivo de percorrer um caminho e retornar a um ponto próximo a pose inicial.

4.1.2 Características dos Ambientes Simulados Avaliados

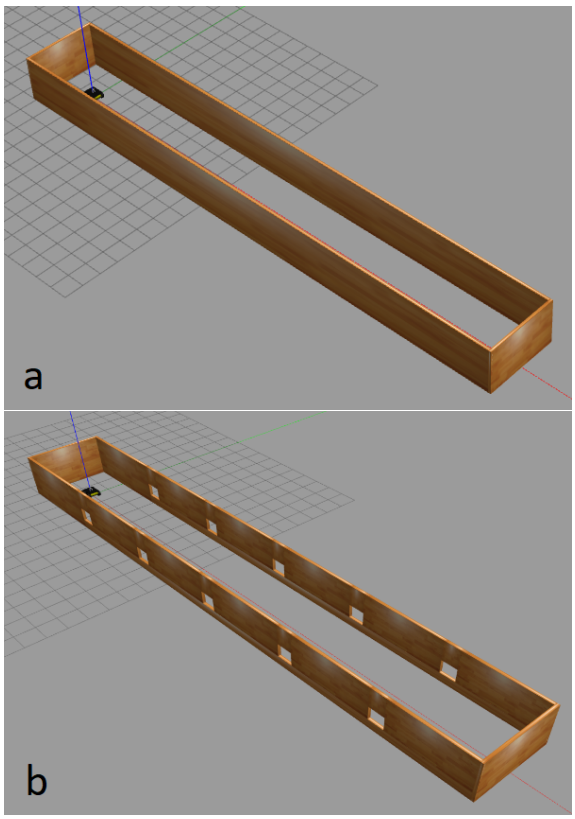
Para avaliação do desempenho dos algoritmos LiDAR SLAM, em ambientes simulados, foram utilizados quatro ambientes no simulador Gazebo. O objetivo foi avaliar o comportamento dos algoritmos em ambientes com poucas e com diversas *features* geométricas e a influência do deslocamento em linha reta e na presença de curvas.

O primeiro ambiente criado foi um corredor reto de 30 metros de comprimento, com 4 metros de largura, com paredes lisas com 2,5 metros de altura, sem qualquer objeto no caminho, conforme ilustrado na Figura 4.2a. O segundo ambiente foi um corredor reto, com as mesmas dimensões do primeiro, porém com 1 janela a cada 5 metros em ambas as paredes laterais, com o objetivo de simular alguns referenciais estáticos, conforme ilustrado na Figura 4.2b.

O terceiro ambiente criado foi um corredor fechado, em formato quadrado, com

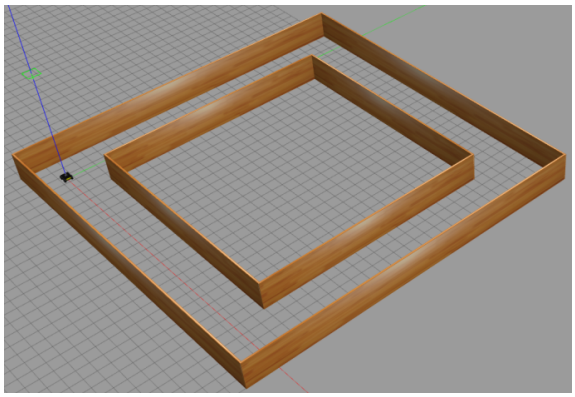
paredes lisas, 30 metros de comprimento, em cada aresta, e largura de 5 metros, conforme ilustrado na Figura 4.3. O quarto ambiente simulado foi o Laboratório de Robótica da UFBA, que foi criado com as mesmas dimensões e a mesma disposição e quantidade de mesas e cadeiras do ambiente real, conforme apresentado na Figura 4.4. A proposta de realizar o experimento no laboratório modelado no Gazebo foi comparar os resultados de localização e mapeamento obtidos no ambiente simulado com o ambiente real, como forma de validar a utilização de ambientes simulados para análise de desempenho dos algoritmos LiDAR SLAM.

Figura 4.2: Modelo de mundo no gazebo para simulação de movimentação do robô Husky em um corredor reto sem janelas (a) e em um corredor reto com janelas (b).



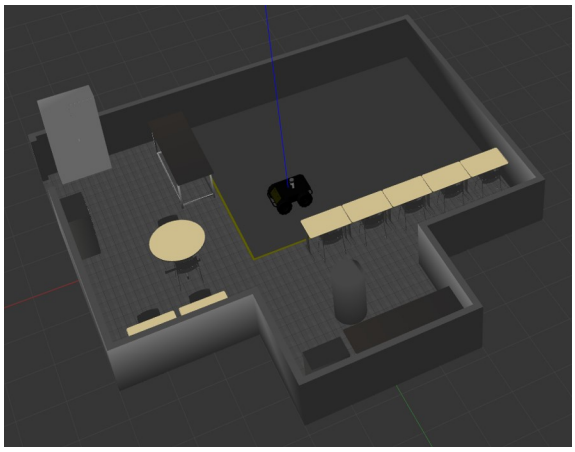
Fonte: Elaborada pelo autor.

Figura 4.3: Modelo de mundo no gazebo para simulação de movimentação do robô Husky em um corredor quadrado fechado sem janelas.



Fonte: Elaborada pelo autor.

Figura 4.4: Modelo do Laboratório de Robótica da UFBA desenvolvido no Gazebo.



Fonte: Elaborada pelo autor.

4.2 Análise em Ambiente Real

4.2.1 Recursos Utilizados

Para os experimentos realizados em ambientes reais foi utilizado o robô móvel Husky UGV A200, disponível no LaR - Laboratório de Robótica do Departamento de Engenharia Elétrica e Computação da UFBA, conforme apresentado na Figura 4.5. O Husky A200 foi projetado para trabalhar em ambientes externos pesados, com sistema de transmissão 4x4 e pneus robustos, que se adaptam a todo tipo de

terreno, o robô possui as especificações listadas na Tabela 4.1.

O robô foi equipado com um sensor LiDAR Velodyne VLP-16 e com diversos marcadores esféricos para estimativa de pose pelo sistema de captura de movimento *Optitrack*[2], um marcador visual (*ground truth* foi posicionado exatamente em cima do sensor LiDAR, conforme apresentado na Figura 4.5. Desta forma é possível obter comparações entre o *ground truth* e a estimativa de posição calculada pelos algoritmos LOAM.

Figura 4.5: Robô Husky equipamento com sensor LiDAR Velodyne VLP-16 e marcadores esféricos para estimativa de pose pelo *Optitrack*.

Fonte: Elaborada pelo autor

[2]https://www.optitrack.com/cameras/flex-3/

Tabela 4.1: Especificações Técnica do Robô Husky UGV A200 (Fonte: Clearpath Robotics).

Característica	Valor
Dimensões (Comprimento x Largura x Altura)	990 x 670 x 390 mm
Peso	50 kg
Velocidade Máxima	1 m/s
Tempo de Operação	3 horas em operação normal (máximo de 8 horas)
Potência de Pico	1.000 W
Potência Média	400 W
Bateria	24 V 20 Ah
Comunicação	RS-232 115200 (Baud rate)
Encoders	$>$ 200.000 pulsos/min
Modos de Controle	Tensão elétrica
	Velocidade
	Velocidade das rodas
	Torque
Drives/APIs	ROS
	MOOS-IvP
	LabVIEW
	Python

A sistema de localização *Optitrack*, modelo Flex 3, que possuem resolução de 640 x 480 (0,3 MP), acurácia de ±0,5 mm e taxa de captura de 100 FPS, foi utilizado como *Ground Truth*. O sistema instalado no LaR é composto de 8 câmeras fixadas no teto, de forma a garantir uma captura de movimentos em uma região de projeção quadrada, com aproximadamente 5 metros de lado e 2 metros de altura, levando em consideração que as câmeras possuem FOV Horizontal de 57,5° e FOV Vertical de 34,7°.

A conexão entre as câmeras e o computador é feita via cabos USB 2.0, sendo este mesmo cabo responsável pela alimentação elétricas das câmeras. A Figura 4.6, ilustra o posicionamento das câmeras do dispositivo em relação a região de interesse,

onde o robô está localizado.

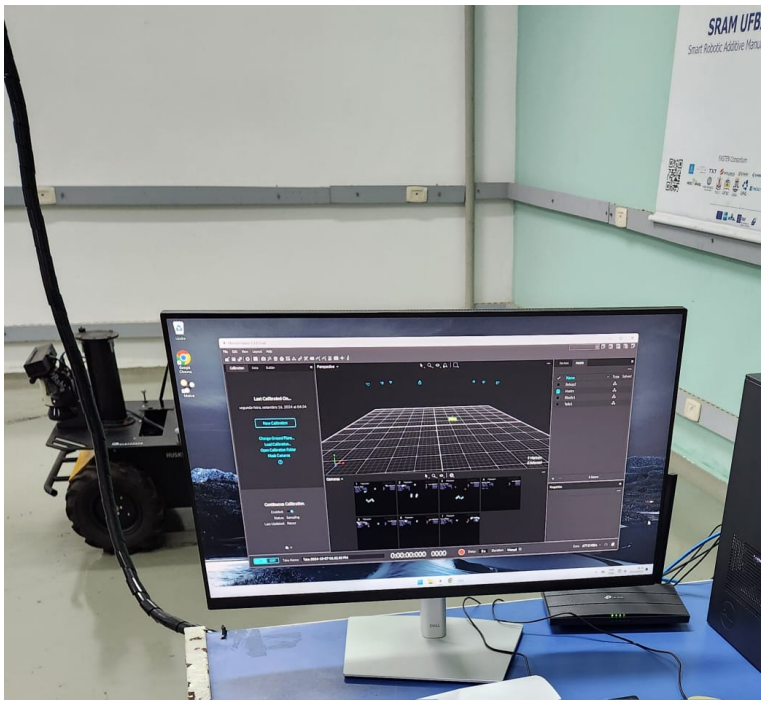
Figura 4.6: Posicionamento das câmeras do sistema de localização *Optitrack*.

Fonte: Elaborada pelo autor.

O computador de processamento da poses geradas pelo sistema *Optitrack* foi instalado próximo a região de interesse no laboratório de robótica, de forma a permitir acompanhar os dados gerados pelo sistema em tempo real e identificar as câmeras que estão capturando o movimento de sistemas robóticos a cada instante. A Figura 4.7 apresenta a tela de visualização do sistema *Optitrack* e a posição do computador de processamento em relação a região de interesse do laboratório.

A aquisição de dados foi realizada com o robô Husky sendo teleoperado utilizando controle remoto, percorrendo a região monitorada pelo sistema *Optitrack*, realizando curvas e percorrendo trechos retos, de forma a analisar o comportamento dos algoritmos avaliados nas mudanças de sentido de deslocamento e deslizamento do robô. A análise dos resultados foi realizada a partir de um arquivo .bag gravado durante a varredura do ambiente, sendo possível analisar o desempenho dos algoritmos de forma isolada e de forma simultânea.

Figura 4.7: Computador de processamento do sistema *Optitrack* com a visualização instantânea do posicionamento do robô no ambiente.



Fonte: Elaborada pelo autor.

4.2.2 Ambiente Avaliado

A análise de desempenho em um ambiente real foi realizada no LaR, que possui uma área dedicada para testes de robótica móvel, conforme já apresentada na Figura 4.6, que possui piso liso e sem a presença de obstáculos no trajeto realizado, porém com diversos objetos espalhados pelo ambiente, como mesas, cadeiras, computadores, dentro outros. A região de interesse selecionada, onde foi realizada a movimentação do robô e varredura simultânea, é uma região plana, com aproximadamente 25 m^2,

4.3 Análise de Resultados

Para análise quantitativa dos resultados foram utilizadas 7 métricas. Sendo que a primeira métrica analisada foi a média e desvio padrão do número de *features* geométricas de plano e de borda detectadas a cada pose publicada. O cálculo para

obtenção do média e desvio padrão do número de *features* de borda é realizado conforme (4.1) e (4.2):

$$\mu_{f_b} = \frac{1}{N}\sum_{k=1}^{N}(f_{b,k}), \tag{4.1}$$

$$\sigma_{f_b} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(f_{b,k} - \mu_{f_b})^2}, \tag{4.2}$$

o cálculo da média e desvio padrão do número de *features* de plano detectada a cada iteração é calculado conforme (4.3) e (4.4):

$$\mu_{f_p} = \frac{1}{N}\sum_{k=1}^{N}(f_{p,k}), \tag{4.3}$$

$$\sigma_{f_p} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(f_{p,k} - \mu_{f_p})^2}, \tag{4.4}$$

sendo que, $f_{b,k}$ e $f_{p,k}$ representam o número de *features* de borda e de plano respectivamente, a cada iteração k gerada.

A segunda métrica analisada foi a estimativa do caminho total percorrido pelo robô estimado cada algoritmo, foi calculada a distância percorrida de referência através dos dados do *Ground Truth* do Husky. A distância percorrida ($D_{percorrida}$) foi calculada pelo somatório do módulo das distâncias entre cada pose estimada no plano xy, conforme (4.5). O mesmo método será utilizado para o cálculo da distância percorrida nas estimativas calculadas pelo LeGO-LOAM, A-LOAM e F-LOAM.

$$D_{percorrida} = \sum_{i=1}^{N}\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{4.5}$$

A terceira métrica analisada foi o erro de posição ao final da simulação será calculado pelo módulo da distância entre as coordenadas x ($x_{estimado}$) e y ($y_{estimado}$) pelo modelo avaliado em relação a posição final obtida pelo *Ground Truth (GT)* do simulador. O módulo do erro no plano xy (Erro_{xy}), das técnicas analisadas em relação a referência, foi calculado conforme (4.6):

$$Erro_{xy} = \sqrt{(x_{estimado} - x_{GT})^2 + (y_{estimado} - y_{GT})^2}. \tag{4.6}$$

A quarta métrica analisada foi o erro de orientação final absoluta referente ao ângulo *yaw* (ψ) foi calculado pela diferença da orientação estimada ($\psi_{estimado}$) pela orientação final obtida pelo *ground truth* (ψ_{GT}), conforme (4.7):

$$Erro_{\psi} = \sqrt{(\psi_{estimado} - \psi_{GT})^2}. \tag{4.7}$$

A análise de orientação apenas pelo ângulo ψ foi escolhida pelo fato do robô ser terrestre, e as simulações terem sido realizadas em ambientes planos.

A quinta e a sexta métricas fornecem a estimativa de variação dos resultados obtidos pelas diferentes técnicas de localização, calculando o Erro Médio Absoluto (EMA) e o Erro Quadrático Médio (EQM) com respeito às poses obtidas pelo *ground truth*, conforme (4.8) (4.9):

$$EMA = \frac{1}{N}\sum_{k=1}^{N}(p_{gt,k} - p_{L,k}), \tag{4.8}$$

$$EQM = \frac{1}{N}\sum_{k=1}^{N}(p_{gt,k} - p_{L,k})^2, \tag{4.9}$$

sendo $p_{L,k}$, a pose estimada pela técnica LiDAR SLAM, e $p_{gt,k}$, a pose obtida pelo *ground truth* do simulados.

A sétima métrica foi análise do Ruído Médio de Localização no Plano xy (RM_{xy}), que foi calculada pelo menor distância entre a posição $p_{R,k}$ no instante de tempo k, em relação ao segmento de reta formado pelas posições $p_{R,k-n}$ e $p_{R,k+n}$, onde $q_{k-n,k+n}$ é o vetor formado pelas posições $p_{R,k-n}$ e $p_{R,k+n}$, e $q_{k,k-n}$ é o vetor formado pelas posições $p_{R,k}$ e $p_{R,k-n}$, conforme (4.10):

$$RM_{xy} = \frac{1}{N-2}\sum_{k=2}^{N-1}\frac{\|q_{k-n,k+n}\| \times \|q_{k,k-n}\|}{\|q_{k-n,k+n}\|}. \tag{4.10}$$

Capítulo 5

RESULTADOS

Esta Capítulo está dividido em 2 Seções que descrevem os resultados dos algoritmos em ambientes simulados e no ambiente real. A primeira Seção apresenta os resultados de mapeamento e odometria nos 4 ambientes simulados, com o objetivo de avaliar o desempenho das técnicas LiDAR SLAM LeGO-LOAM, A-LOAM e F-LOAM, em comparação ao *ground truth* do simulador Gazebo, em condições com volume de *features* geométricas diferentes, com deslocamento em linha reta e com presença de curvas e obstáculos. Na segunda Seção são apresentados os resultados das técnicas estudadas em um ambiente real, comparando os resultados com a localização gerada pelo sistema de captura de movimentos *Optitrack*.

5.1 Resultados dos Algoritmos LOAM Velodyne em Ambientes Simulados

Nos experimentos realizados no corredor reto, com e sem janela, o robô foi controlado através da biblioteca rospy, escrevendo a velocidade linear de 0,5 m/s no tópico /cmd_vel, que publica em uma frequência de 10 Hz, e após 500 iterações percorre uma distância aproximada de 25 metros orientada no eixo x do mundo.

Para os experimentos realizados no corredor quadrado fechado, em formato quadrado e no ambiente LaR modelado no Gazebo, o robô foi teleoperado pelo teclado do computador utilizando o pacote Husky_control, teleop_keyboard. Para o corredor quadrado fechado sem janela a velocidade média do robô foi de 0,69 m/s, considerando uma distância total percorrida de 95,81 metros em 138,6 segundos, e

no ambiente LaR simulado a velocidade média do robô foi de 0,37 m/s, percorrendo uma distância total de 18,28 metros em 49,48 segundos. Todas as implementações realizadas para a análise de desempenho dos algoritmos nos ambientes selecionados foram compartilhadas publicamente em um repositório na plataforma *Github*[1].

Os resultados serão apresentados a seguir estão divididos em erro de distância percorrida e erro de posição final no plano xy. Para melhor comparação dos resultados, os mesmos estão descritos em gráficos e tabelas. Nas tabelas os ambientes A, B, C e D, representam o corredor reto sem janela, o corredor reto com janela, o corredor quadrado fechado sem janela e o ambiente LaR simulado respectivamente, e os melhores valores comparados estão destacados nas tabelas.

5.1.1 Número médio de *features* geométricas de borda e de plano capturadas por algoritmo

As Tabelas 5.1 e 5.2 apresentam o número de *features* de borda capturadas pelo algoritmos durante as simulações realizadas. Os algoritmos A-LOAM e F-LOAM apresentaram um maior número de *features* de borda capturadas em todos os ambientes simulados.

Durante a simulação realizada no corredor quadrado fechado sem janelas foi capturado o menor número médio de *features* de bordas, seguido do corredor reto sem janelas, o que apresenta relação direta com o desempenho dos algoritmos para as correções da estimação de localização. O número médio de *features* de borda capturado no ambiente LaR simulado foi maior para os algoritmos A-LOAM e F-LOAM em comparação ao resultado do LeGO-LOAM, o que similar ao encontrado nos outros ambientes simulados.

Com relação ao número médio de *features* planas capturadas, no ambiente do LaR simulado, como esperado, devido a ausência de corredores contínuos, apresentaram os menores valores. O algoritmo F-LOAM apresentou um volume capturado de *features* planas de 8 a 10 vezes superior aos demais algoritmos devido ao método menos restrito de seleção de superfícies planas, o que permite a inclusão de um número maior de pontos no conjunto de *features* planares. O algoritmo F-LOAM prioriza a eficiência em comparação aos algoritmos LeGO-LOAM e A-LOAM, que

[1]https://github.com/mecespinheira/LOAM_Velodyne

Tabela 5.1: Média e desvio padrão do número de *features* de **borda** capturadas durante a movimentação do robô.

Ambiente	LeGO-LOAM	A-LOAM	F-LOAM
	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
A	158,27±25,20	**293,40±40,74**	293,31±40,49
B	240,96±42,56	351,85±53,58	**352,26±53,90**
C	135,82±16,25	**240,31±66,94**	240,41±66,65
D	171,09±42,30	**452,51±105,95**	452,26±105,07

priorizam a precisão e descartam redundâncias [4].

ao método não iterativo de extração de *features*, que ocorre em duas etapas, o que leva a um desempenho superior, chegando a uma taxa de aquisição de 20 Hz, com custo computacional inferior.

Tabela 5.2: Média e desvio padrão do número de *features* de **plano** capturadas durante a movimentação do robô.

Ambiente	LeGO-LOAM	A-LOAM	F-LOAM
	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
A	2694,36±254,31	2785,50±240,94	**27931,15±334,08**
B	2585,84±225,27	2799,25±235,37	**26026,17±668,15**
C	3105,21±206,24	3337,56±282,07	**27507,60±247,42**
D	1680,83±117,16	2117,73±141,38	**19632,30±1506,40**

5.1.2 Erro de Distância Total Percorrida

O erro de distância total percorrida estimada pelos algoritmos avaliados, que leva em consideração apenas a distância, sem avaliação das poses, está apresentado na Tabela 5.3 abaixo. Na tabela é possível observar que o algoritmo A-LOAM apresentou o melhor desempenho para o corredor reto com e sem janelas (Ambiente A e B respectivamente), e no ambiente LaR simulado (Ambiente D). O algoritmo LeGO-LOAM apresentou o melhor desempenho para o corredor quadrado fechado sem janela (Ambiente C), mesmo apresentando o menor número de *features* de borda detectadas em comparação aos demais algoritmos, porém, neste ambiente, quando

somada as *features* de borda e de plano, foi detectado um maior número total de *features* pelo LeGO-LOAM.

Tabela 5.3: Comparação do erro de distância total percorrida entre cada um dos algoritmos avaliados.

		Erro relativo a distância de referência(%)		
Ambiente	Distância(m)	LeGO-LOAM	A-LOAM	F-LOAM
A	24,980	12,570	**1,237**	1,244
B	24,980	0,297	**0,532**	1,345
C	95,813	**0,018**	0,706	0,730
D	18,276	2,899	**0,213**	0,338

5.1.3 Erro de Posição e Orientação Final

O erro na localização final estimada pelos algoritmos avaliados, que calcula o módulo da distância entre a posição final estimada no plano xy por cada técnica LiDAR SLAM e a posição final obtida pelo GT.

O erro localização final estimada pelos algoritmos avaliados, que calcula o módulo da distância entre a posição final estimada no plano *xy* por cada técnica LiDAR SLAM e a posição final obtida pelo *ground truth*, está apresentado na Tabela 5.4 abaixo. Na tabela é possível observar que o algoritmo A-LOAM apresentou o melhor desempenho para o corredor reto sem janelas (Ambiente A) e para o ambiente LaR simulado (Ambiente D). O algoritmo LeGO-LOAM apresentou o melhor desempenho para o corredor reto com janelas (Ambiente B) e para o corredor quadrado fechado sem janelas (Ambiente C).

O algoritmo LeGO-LOAM apresentou o melhor desempenho geral para estimativa de posição final no corredor reto com janelas (Ambiente B), e o pior desempenho geral para o corredor reto sem janelas (Ambiente A). Analisando o resultados apresentados na Tabela 5.1, é possível realizar uma correlação entre o desempenho do algoritmo na determinação da posição final e o número médio de *features* de borda detectadas, onde no corredor reto com janelas foi capturado o maior número de *features* de borda, o que determina uma melhor correção da estimativa de posição ao longo da trajetória.

Tabela 5.4: Comparação entre as estimativas de pose final no plano xy, com valores apresentados de acordo ao ambiente e algoritmo utilizado.

	Erro de posição final no plano xy (metros)		
Ambiente	LeGO-LOAM	A-LOAM	F-LOAM
A	3,274	**0,411**	0,405
B	**0,088**	0,163	0,417
C	**0,110**	0,199	0,253
D	0,198	**0,178**	0,182

O erro absoluto, em graus, para o ângulo *yaw* (ψ), estão apresentados na Tabela 5.5, sendo possível atestar a precisão dos algoritmos no que se refere a orientação do robô. O algoritmo LeGO-LOAM apresentou melhores resultados na previsão da orientação final no corredor reto com janelas, no corredor quadrado fechado sem janelas e no ambiente LaR simulado, ambientes B, C e D respectivamente. O algoritmo A-LOAM apresentou melhor resultado na previsão de orientação final no corredor reto sem janelas (Ambiente A).

Tabela 5.5: Comparação entre as estimativas de orientação final para o ângulo *yaw* (ψ), com valores apresentados de acordo ao ambiente e algoritmo utilizado.

	Erro absoluto de orientação final para o ângulo *yaw* ψ (°)		
Ambiente	LeGO-LOAM	A-LOAM	F-LOAM
A	0,025	**0,001**	0,059
B	**0,006**	0,030	0,109
C	**0,165**	0,415	0,518
D	**0,157**	0,193	0,271

5.1.4 Análise do Erro Médio Absoluto e do Erro Quadrático Médio das Poses Geradas

Conforme apresentado na Tabela 5.6 abaixo, o erro médio absoluto no corredor reto sem janelas no eixo x para o algoritmo LeGO-LOAM apresentou valor de 2,532 metros, o que para uma distância final percorrida pelo robô, que foi de 25 metros, se

apresenta como um erro importante, especialmente em comparação aos algoritmos A-LOAM e F-LOAM, que apresentaram valore de erro médio absoluto de 0,216 metros e 0,274 metros, respectivamente. Entretanto, na Tabela 5.7, o erro médio absoluto para o corredor reto com janela para o algoritmo LeGO-LOAM apresentou o melhor resultado para posição no plano xy e orientação no eixo ψ, o que apresenta relação com o volume de *features* de borda capturado pelo algoritmo.

Tabela 5.6: Média e desvio padrão dos erros médios absolutos em relação ao *ground truth* para o **corredor reto sem janelas.**

Algoritmo	$\mu_x \pm \sigma_x$ (m)	$\mu_y \pm \sigma_y$ (m)	$\mu_\psi \pm \sigma_\psi(^\circ)$
LeGO-LOAM	2,532±0,711	0,004±0,002	**0,014±0,016**
A-LOAM	**0,216±0,138**	**0,003±0,002**	0,016±0,007
F-LOAM	0,274±0,096	0,015±0,005	0,052±0,038

Tabela 5.7: Média e desvio padrão dos erros médios absolutos em relação ao *ground truth* para o **corredor reto com janelas.**

Algoritmo	$\mu_x \pm \sigma_x$ (m)	$\mu_y \pm \sigma_y$ (m)	$\mu_\psi \pm \sigma_\psi(^\circ)$
LeGO-LOAM	**0,022±0,017**	**0,003±0,002**	**0,007±0,011**
A-LOAM	0,099±0,037	**0,003±0,002**	0,017±0,009
F-LOAM	0,141±0,069	0,015±0,011	0,048±0,032

Para a análise no corredor quadrado fechado sem janelas, conforme apresentado na Tabela 5.8, houve uma tendência para menor erro absoluto na coordenada x e no ângulo ψ para o algoritmo LeGO-LOAM. Para o ambiente LaR do gazebo o algoritmo A-LOAM apresentou tendência a melhor erro absoluto para a coordenada y e para o ângulo ψ, conforme apresentado na Tabela 5.9.

Tabela 5.8: Média e desvio padrão dos erros médios absolutos em relação ao *ground truth* para o **corredor quadrado fechado sem janelas.**

Algoritmo	$\mu_x \pm \sigma_x$ (m)	$\mu_y \pm \sigma_y$ (m)	$\mu_\psi \pm \sigma_\psi(^\circ)$
LeGO-LOAM	**0,125±0,083**	0,097±0,073	**1,767±19,324**
A-LOAM	0,148±0,089	0,106±0,078	2,133±23,491
F-LOAM	0,180±0,098	**0,096±0,070**	1,983±21,435

Tabela 5.9: Média e desvio padrão dos erros médios absolutos em relação ao *ground truth* para o **ambiente LaR simulado.**

Algoritmo	$\mu_x \pm \sigma_x$ (m)	$\mu_y \pm \sigma_y$ (m)	$\mu_\psi \pm \sigma_\psi(^\circ)$
LeGO-LOAM	**0,108±0,098**	0,091±0,079	1,345±2,636
A-LOAM	0,135±0,080	**0,056±0,037**	**0,939±1,399**
F-LOAM	0,146±f0,087	0,057±0,037	1,728±16,162

Com relação ao erro quadrático médio, que valoriza o impacto do desvio padrão devido a sua fórmula que eleva o erro ao quadrado, o algoritmo A-LOAM apresentou o melhor desempenho para o corredor reto sem janela, conforme observado na Tabela 5.10, e o algoritmo LeGO-LOAM apresentou o melhor resultado no corredor reto com janelas, conforme apresentado na Tabela 5.11.

Tabela 5.10: Erro Quadrático Médio (EQM) em relação ao *ground truth* para o **corredor reto sem janelas.**

Algoritmo	EQM_x (m)	EQM_y (m)	$EQM_\psi(^\circ)$
LeGO-LOAM	6,913	1,846e-05	1,599e-05
A-LOAM	**0,066**	**1,331e-05**	**5,473e-06**
F-LOAM	0,084	2,544e-04	7,221e-05

Tabela 5.11: Erro Quadrático Médio (EQM) em relação ao *ground truth* para o **corredor reto com janelas.**

Algoritmo	EQM_x (m)	EQM_y (m)	$EQM_\psi(^\circ)$
LeGO-LOAM	**0,001**	**9,510e-08**	**5,449e-06**
A-LOAM	0,011	1,194e-07	6,841e-06
F-LOAM	0,025	9,997e-07	5,728e-05

Na simulação realizada no corredor quadrado fechado sem janelas, houve uma tendência a melhor resultado para o algoritmo LeGO-LOAM, conforme observado na Tabela 5.12, onde o Erro Quadrático Médio foi menor para a coordenada x e para o ângulo ψ. Na análise do Erro Quadrático Médio para o ambiente LaR simulado, conforme apresentado na Tabela 5.13, foi observado uma tendência a melhor resultado para o algoritmo A-LOAM.

Tabela 5.12: Erro Quadrático Médio (EQM) em relação ao *ground truth* para o **corredor quadrado fechado sem janelas.**

Algoritmo	EQM_x (m)	EQM_y (m)	$EQM_\psi(^\circ)$
LeGO-LOAM	**0,022**	0,015	**6,553**
A-LOAM	0,030	0,017	9,704
F-LOAM	0,042	**0,014**	8,082

Tabela 5.13: Erro Quadrático Médio (EQM) em relação ao *ground truth* para o **ambiente LaR simulado.**

Algoritmo	EQM_x (m)	EQM_y (m)	$EQM_\psi(^\circ)$
LeGO-LOAM	**0,021**	0,015	0,293
A-LOAM	0,025	**0,005**	**0,050**
F-LOAM	0,029	**0,005**	4,611

5.1.5 Ruído Médio de Localização no Plano *xy* por algoritmo

Conforme apresentado na Tabela 5.14, com relação ao Ruído Médio de Localização no Plano *xy*, o algoritmo A-LOAM apresentou o melhor resultado para o corredor reto sem janela, para o corredor reto com janela e para o corredor quadrado fechado sem janelas (Ambientes A, B e C respectivamente). Para o ambiente LaR simulado o algoritmo LeGO-LOAM apresentou o melhor desempenho para o ruído de posição médio, comparado aos resultados obtidos com os algoritmos A-LOAM e F-LOAM.

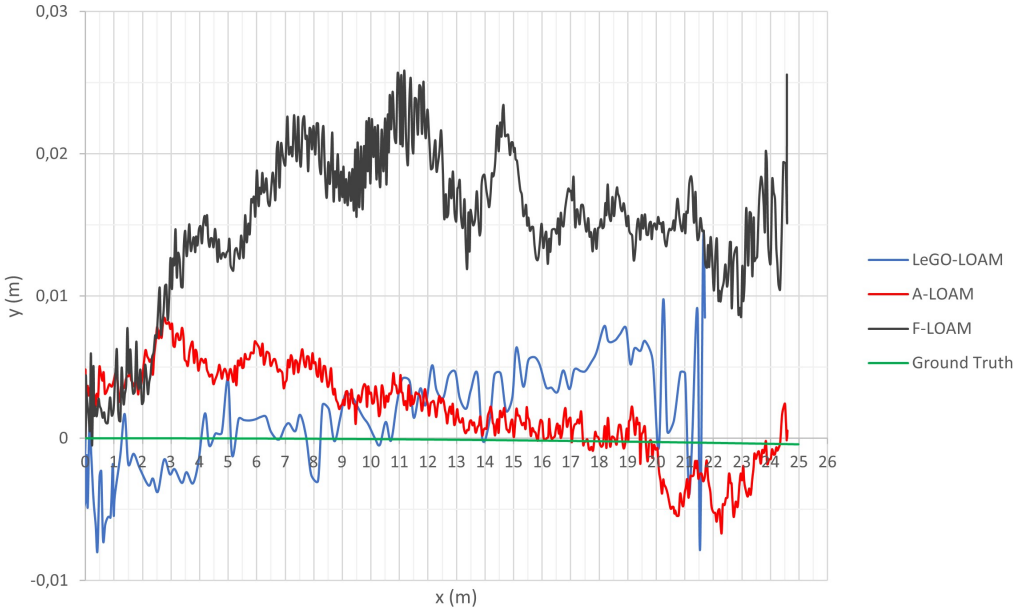
Tabela 5.14: Ruído de posição no plano *xy* em metros.

Ambiente	LeGO-LOAM	A-LOAM	F-LOAM
A	1,833e-03	**4,735e-04**	1,369e-03
B	1,483e-03	**4,298e-04**	1,180e-03
C	1,135e-02	**1,245e-03**	1,345e-03
D	**1,553e-02**	3,061e-03	3,138e-03

5.1.6 Análise Gráfica Comparativa entre os Algoritmos

Esta seção apresenta a análise gráfica dos algoritmos em cada situação testada. As Figuras 5.1, 5.2, 5.3 e 5.4, apresentam o gráfico comparativo de cada algoritmo para a ambiente com corredor reto sem janela, corredor reto com janela, corredor quadrado fechado sem janela e no ambiente LaR simulado.

Figura 5.1: Gráfico comparativo das estimativas de posição dos algoritmos testados no corredor reto sem janela.

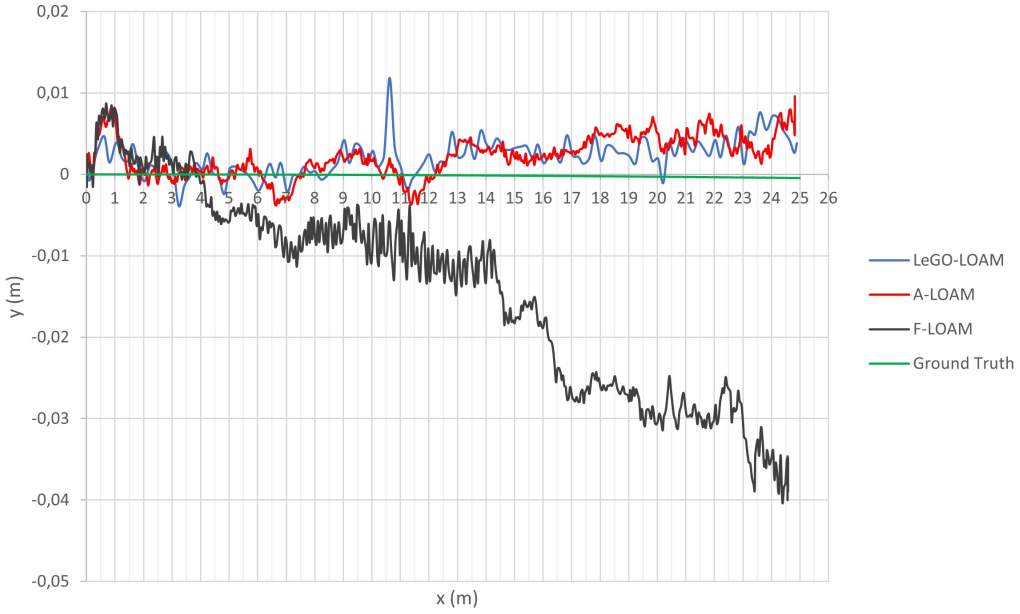


Fonte: Elaborado pelo autor.

A análise gráfica dos algoritmos nos corredores retos permite observar o grau de variação da estimativa de posição, sendo que a cada nova iteração uma nova pose estimada com base nos dados da nuvem de pontos e das *features* geométricas identificadas no ambiente.

No gráfico da Figura 5.1, obtido pelo resultado da simulação do robô se movimentando em um corredor reto sem janela, é possível observar a falha no desempenho do algoritmo LeGO-LOAM na presença de poucas *features* geométricas, na determinação da posição final, em comparação com os demais algoritmos LiDAR SLAM. No gráfico da Figura 5.2 é possível observar o melhor desempenho dos algoritmos para determinação da posição final no eixo x, com a presença de referenciais estáticos

Figura 5.2: Gráfico comparativo das estimativas de posição dos algoritmos testados no corredor reto com janela.



Fonte: Elaborado pelo autor.

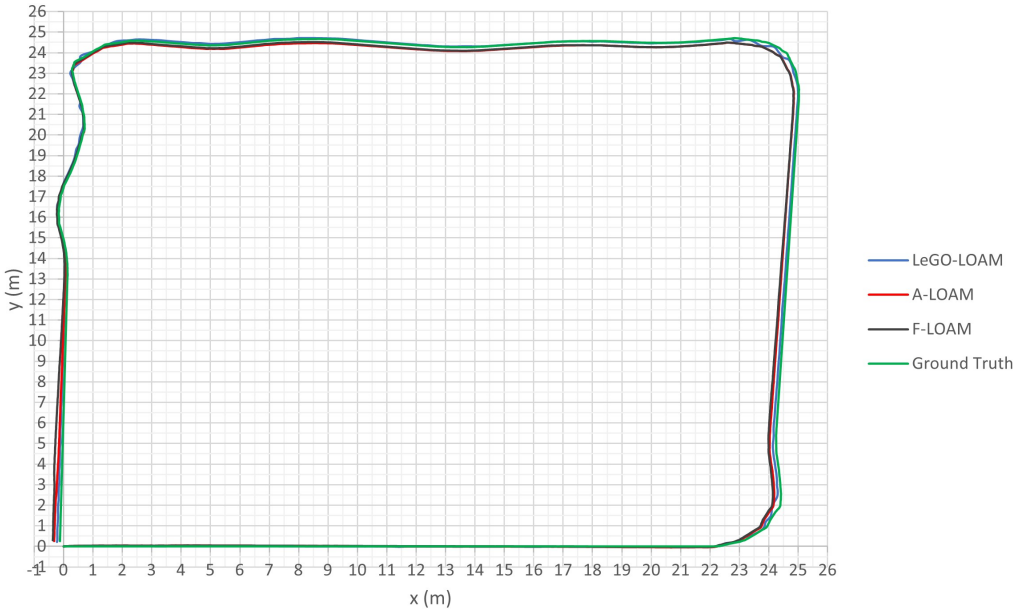
simples, como janelas espaçadas simetricamente.

Nos gráficos das Figuras 5.1 e 5.2 é possível observar um desvio acumulado, no eixo y, no resultado do mapeamento com a estratégia F-LOAM, o que apresenta relação com o maior ruído na estimativa de localização instantânea da técnica. A estratégia A-LOAM apresentou o menor ruído na estimativa de localização instantânea, o que deve estar relacionado à utilização da biblioteca de otimização não-linear Ceres Solver.

No gráfico da Figura 5.3, obtido pelo resultado da simulação do robô se movimentando em um corredor quadrado fechado sem janelas, é possível observar que não houve diferença significativa entre os resultados dos algoritmos, quando comparado ao *ground truth*. No gráfico da Figura 5.4, referente à estimativa de localização obtida no ambiente LaR simulado, é possível observar que todos os algoritmos apresentaram desvio importante da estimativa de localização nas curvas, e que o algoritmo LeGO-LOAM apresentou a melhor estimativa de posição ao final da simulação.

Com base nos gráficos apresentados é possível inferir que as técnicas A-LOAM

Figura 5.3: Gráfico comparativo das estimativas de posição dos algoritmos testados no corredor quadrado fechado sem janela.



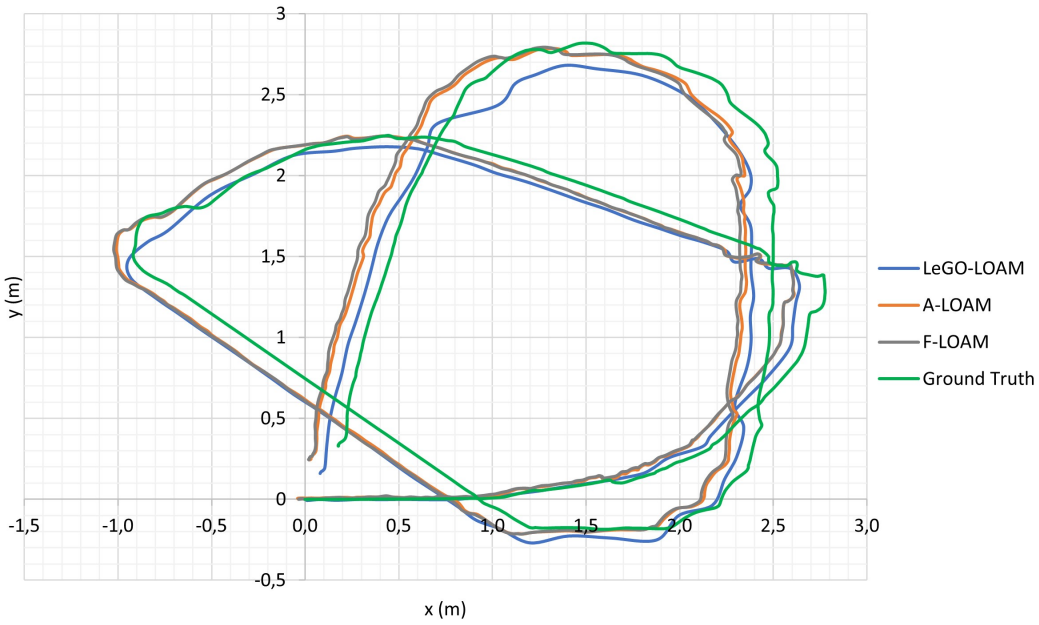
Fonte: Elaborado pelo autor.

e F-LOAM apresentaram os melhores resultados para a simulação do corredor reto sem janelas, onde o algoritmo LeGO-LOAM apresentou um erro considerável na estimativa de posição final. Para os ambientes corredor quadrado fechado sem janela e LaR simulado, o algoritmo LeGO-LOAM apresentou os melhores resultados, com maior convergência de estimativa de localização nos trechos retos e nas curvas, em comparação aos algoritmos A-LOAM e F-LOAM.

5.1.7 Resultados do Mapeamento

As Figuras 5.5, 5.6 e 5.7 apresentam o resultado do mapeamento por nuvem de pontos dos algoritmos no ambiente LaR simulado no gazebo, que foi selecionado para análise qualitativa do mapeamento devido ao maior número de referenciais estáticos e para comparação com os resultados obtidos no ambiente real. O ambiente modelado no gazebo apresenta alguma restrições em comparação ao ambiente real, pois os objetos modelados no gazebo não possuem o mesmo nível de detalhamento dos objetos reais, sendo formados por formas geométricas simples, além disso, as paredes

Figura 5.4: Gráfico comparativo das estimativas de posição dos algoritmos testados na ambiente LaR simulado.



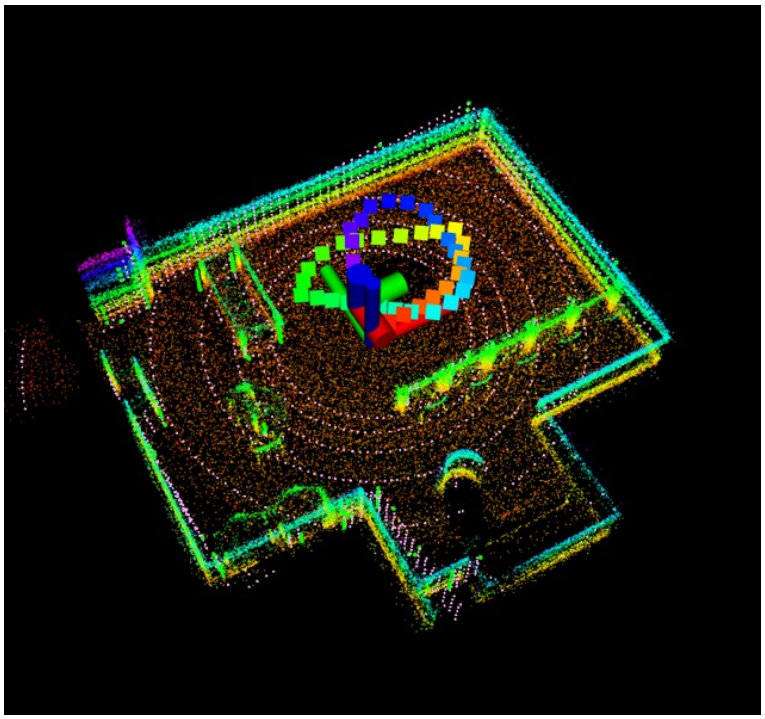
Fonte: Elaborado pelo autor.

não foram modeladas até a altura do teto e diversos objetos existentes no mundo real foram suprimidos, a exemplo dos computadores, que ficam sobre as mesas localizadas ao redor da região onde o robô realizou a trajetória, de dois manipuladores robóticos e de uma impressora 3D que no mundo real estão posicionados sobre bancadas localizadas próximo a região de interesse.

A expectativa é de que o maior número de *features* geométricas identificadas aprimore os resultados obtidos com as técnicas SLAM , devido as correções das distorções da odometria e do mapeamento passo a passo na simulação. As *features* geométricas servem de referência para determinação mais assertiva de novas poses, recalculando a distância entre elas.

Visualmente todos os algoritmos LiDAR SLAM foram capazes de mapear o ambiente LaR simulado, com destaque para o LeGO-LOAM, que conseguiu mapear com maior detalhamento alguns objetos presentes do ambiente, a exemplo das mesas e dos pilares presentes na cena. A orientação das estruturas mapeadas no ambiente permaneceu fixa durante todo o mapeamento, não sendo observada presença de ro-

Figura 5.5: Resultado do mapeamento por nuvem de pontos no ambiente LaR simulado no Gazebo utilizando o algoritmo LiDAR SLAM LeGO-LOAM.



Fonte: Elaborada pelo autor.

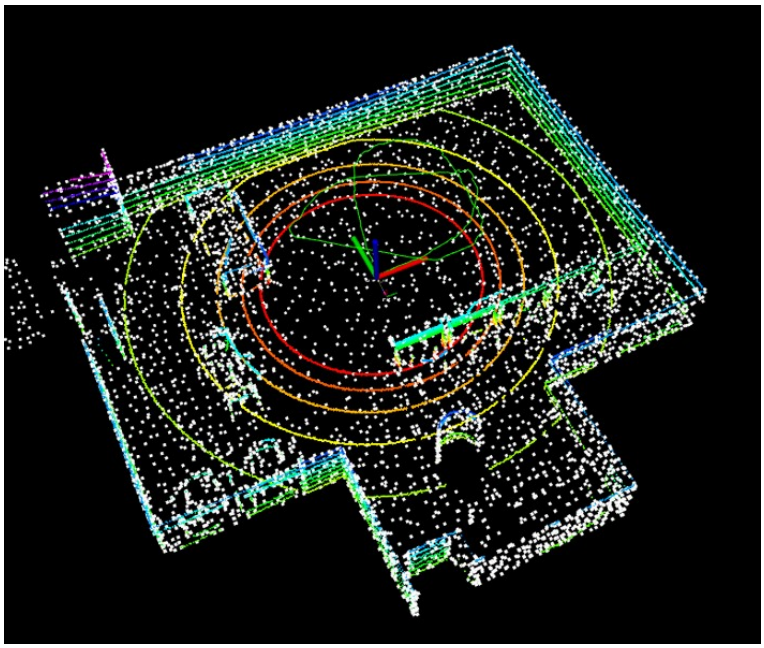
tações no mapa final, o que pode ser constatado pelo alinhamento das mesas e nas arestas das paredes.

A qualidade do mapeamento seria melhor avaliada em uma reconstrução 3D que poderia ser realizada com os dados da nuvem de pontos obtida pelos algoritmos, onde seria melhor visualizado as falhas de preenchimento das estruturas. Neste trabalho, o análise do mapeamento ficou restrita a posicionamento e rotação das estruturas mapeadas.

5.2 Resultados dos algoritmos LOAM Velodyne em Ambientes Reais

Para a realização dos experimentos em um ambiente real o robô foi teleoperado com uso de um controle remoto, onde os movimentos realizados foram uma combinação de trajetórias retas e curvas, com o objetivo de avaliar o comportamento dos

Figura 5.6: Resultado do mapeamento por nuvem de pontos no ambiente LaR simulado no Gazebo utilizando o algoritmo LiDAR SLAM A-LOAM.



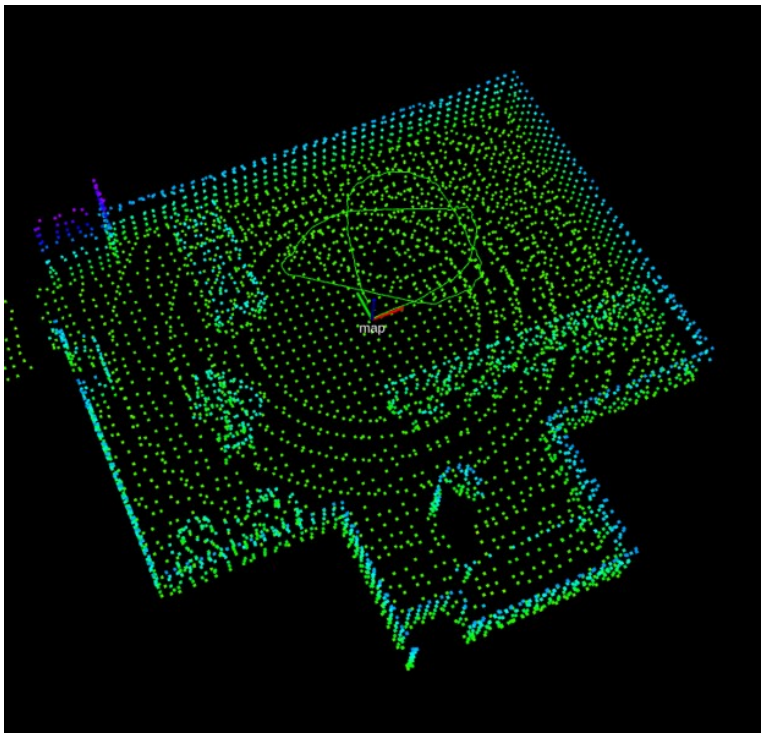
Fonte: Elaborada pelo autor.

algoritmos em situações com e sem deslizamento. A velocidade de movimentação do robô não foi fixada, mas ficou em torno de 0,29 m/s considerando a distância de 22,59 metros percorrida, em 78,71 segundos, em uma área restrita, com trecho reto percorrido inferior a 4 metros.

5.2.1 Número Médio de *Features* Geométricas de Borda e de Plano Capturadas por Algoritmo

Com relação ao número de *features* geométrica capturadas, conforme apresentado na Tabela 5.15, o algoritmo F-LOAM apresenta uma maior número médio de *features* de borda e de plano capturadas a cada pose. É importante observar que o número de *features* de plano capturas pelo algoritmo F-LOAM é cerca de 6 a 9 vezes superior ao obtido pelos demais algoritmos, similar aos resultados apresentados na Tabela 5.2.

Figura 5.7: Resultado do mapeamento por nuvem de pontos no ambiente LaR simulado no Gazebo utilizando o algoritmo LiDAR SLAM F-LOAM.



Fonte: Elaborada pelo autor.

Tabela 5.15: Média e desvio padrão do número de *features* capturadas durante o movimentação do robô.

Algoritmo	N° de *Features* de Borda $\mu \pm \sigma$	N° de *Features* de Plano $\mu \pm \sigma$
LeGO-LOAM	439,729±38,361	2523,696±149,730
A-LOAM	988,802±144,090	3550,419±195,914
F-LOAM	**989,793±144,707**	**22837,906±964,896**

5.2.2 Erro de Distância Total Percorrida

Com relação a distância total percorrida, conforme apresentado na Tabela 5.16, o algoritmo A-LOAM apresentou o melhor resultado, porém com as diferenças encontradas dos demais algoritmos em relação a distância estimada pelo *Optitrack* foi inferior a 1% de erro. O erro de distância final percorrida encontrado foi pequeno o

suficiente para todos os algoritmos estudados, sendo necessário avaliar os resultados em distância maiores e condições diferentes para garantir que não haveria diferenças significativas devido a presença de erro acumulado.

Tabela 5.16: Comparação do erro de distância total percorrida entre cada um dos algoritmos avaliados.

Algoritmo	Distância Percorrida (m)	Erro % em relação a referência
LeGO-LOAM	22,416	0,776
A-LOAM	22,697	**0,466**
F-LOAM	22,705	0,501
Optitrack	22,592	-

5.2.3 Erro de Posição e Orientação Final

A Tabela 5.17 apresenta o erro de posição final absoluta e o erro de orientação final para os três algoritmos estudados. Para o erro de posição final absoluta o algoritmo LeGO-LOAM apresentou o melhor resultado, já para a orientação final o algoritmo A-LOAM apresentou o melhor resultado, sendo o maior erro de orientação no ângulo ψ obtido pelo algoritmo LeGO-LOAM, que foi de 4,843°.

Tabela 5.17: Erro de posição e orientação final.

Algoritmo	Erro Posição Final em Módulo (m)	Erro Orientação Ângulo ψ (°)
LeGO-LOAM	**0,021**	4,843
A-LOAM	0,035	**3,891**
F-LOAM	0,044	3,903

5.2.4 Análise do Erro Médio Absoluto e do Erro Quadrático Médio das Poses Geradas

A Tabela 5.18 apresenta o Erro Médio Absoluto (EMA) obtido pelo algoritmos estudados em relação à estimativa obtida pelo *Optitrack*. Nesta tabela é possível observar que o algoritmo LeGO-LOAM apresentou o melhor resultado em relação

ao EMA na coordenada x e ângulo ψ, já em relação à coordenada y os resultados obtidos pelos algoritmos A-LOAM e F-LOAM foram superiores.

Tabela 5.18: Erro Médio Absoluto (EMA) e Desvio Padrão do Algoritmos Estudados em Relação ao *Optitrack*.

Algoritmo	$\mu_x \pm \sigma_x$ (m)	$\mu_y \pm \sigma_y$ (m)	$\mu_\psi \pm \sigma_\psi(^\circ)$
LeGO-LOAM	**0,016±0,011**	0,012±0,015	**3,653±1,129**
A-LOAM	0,037±0,020	**0,010±0,014**	3,975±12,773
F-LOAM	0,036±0,019	**0,010±0,014**	4,397±18,080

A Tabela 5.19 apresenta os resultados do Erro Quadrático Médio (EQM) para os três algoritmos, onde o algoritmo LeGO-LOAM apresentou o melhor resultado em relação à coordenada x e ao ângulo ψ, já o algoritmo A-LOAM apresentou o melhor resultado para a coordenada y.

Tabela 5.19: Erro Quadrático Médio (EQM) em relação ao *Optitrack*.

Algoritmo	EQM_x (m)	EQM_y (m)	$EQM_\psi(^\circ)$
LeGO-LOAM	**3,857e-04**	3,516e-04	**0,255**
A-LOAM	0,002	**2,800e-04**	3,120
F-LOAM	0,002	0,010	6,035

5.2.5 Ruído Médio de Localização no Plano *xy* por Algoritmo

Com relação ao Ruído Médio de Localização instantânea no Plano xy, em relação a posição estimada pelo *Optitrack*, os algoritmos A-LOAM e F-LOAM apresentaram o menor valor. Este resultado permite afirmar que para estimativa de posição instantânea os algoritmos A-LOAM e F-LOAM apresentam o melhor desempenho, sendo este resultado importante caso seja desejado a implementação de controle baseado no erro de posicionamento instantâneo, conforme apresentado na Tabela 5.20.

5.2.6 Análise Gráfica Comparativa entre os Algoritmos

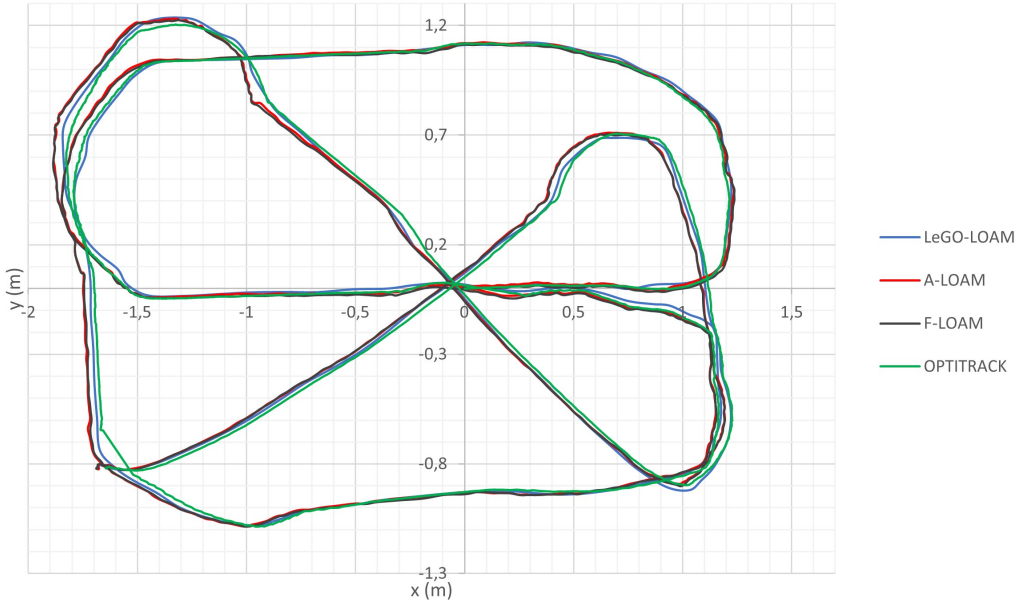
O gráfico apresentado na Figura 5.8 apresenta as estimativas de odometria gerada pelo algoritmos estudados, sendo que robô iniciou o percusso na coordenada x=0

Tabela 5.20: Ruído Médio de Localização no Plano *xy* em metros.

Algoritmo	Ruído de Posição (m)
LeGO-LOAM	0,009
A-LOAM	**0,002**
F-LOAM	**0,002**

e y=0 e terminou na posição x=-1,642 e y=-0,798. Neste gráfico pode ser observado que erros das estimativas de posição obtidos pelos algoritmos nas curvas foi maior, sendo reduzido nos trechos retos, o que permite inferir que os algoritmos não acumulam erro de posição ao longo do percusso. A partir da análise gráfica é possível observar que a estimativa obtida pelo algoritmo LeGO-LOAM apresenta maior convergência com a localização fornecida pelo sistema *Optitrack*.

Figura 5.8: Gráfico comparativo das estimativas de odometria dos algoritmos testados no laboratório de robótica da Universidade Federal da Bahia.



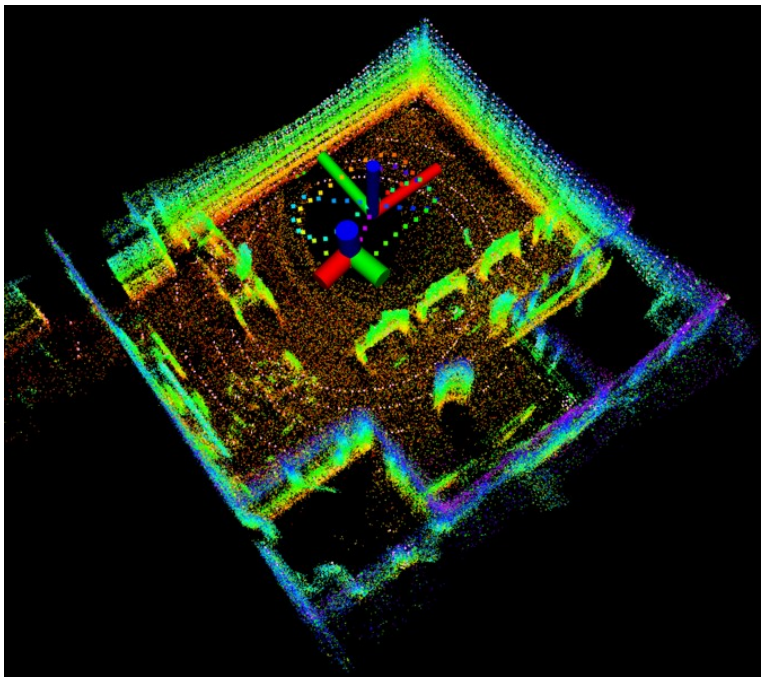
Fonte: Elaborado pelo autor.

5.2.7 Resultados do Mapeamento

Com relação aos resultados do mapeamento foi realizada apenas uma análise qualitativa dos mapas geradas pelos algoritmos LeGO-LOAM, A-LOAM e F-LOAM, conforme apresentado na Figuras 5.9, 5.10 e 5.11, respectivamente. Com bases na análise das imagens geradas, especialmente com atenção aos objetos presentes no laboratório, é possível afirmar que o mapeamento gerado pelo algoritmo LeGO-LOAM apresentou melhor resultado, pois as estruturas estão melhor delimitadas, sem presença de rotações e distorções significativas, o que inclui as mesas e as janelas, o que apresenta relação com o método de seleção de pontos que devem ser corrigidos e armazenados no mapeamento.

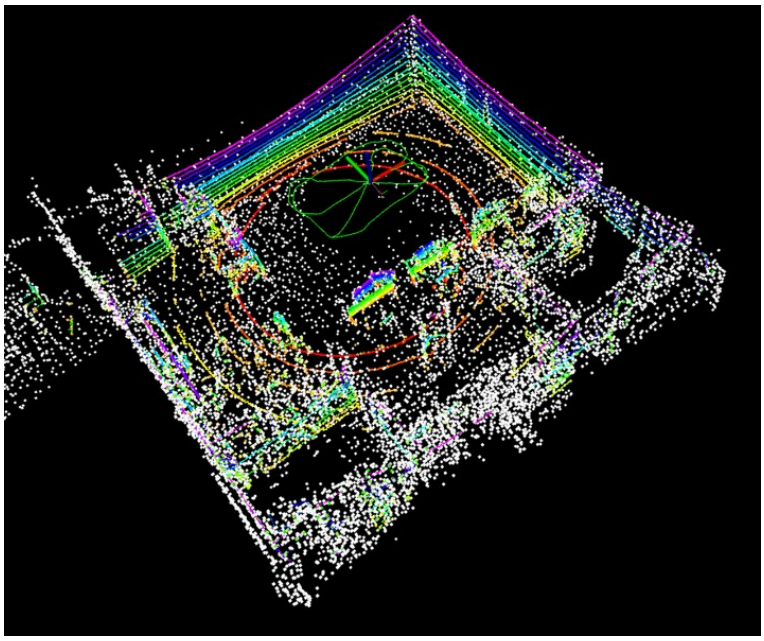
A percepção qualitativa do mapeamento não tem a intenção de inferir nada sobre precisão de medidas e formas, apenas sobre a capacidade de identificar as estruturas presentes no laboratório durante a varredura do ambiente, o que mostra a capacidade do algoritmo em estimar os limites dos objetos.

Figura 5.9: Resultado do mapeamento por nuvem de pontos do LaR utilizando o algoritmo LiDAR SLAM LeGO-LOAM.



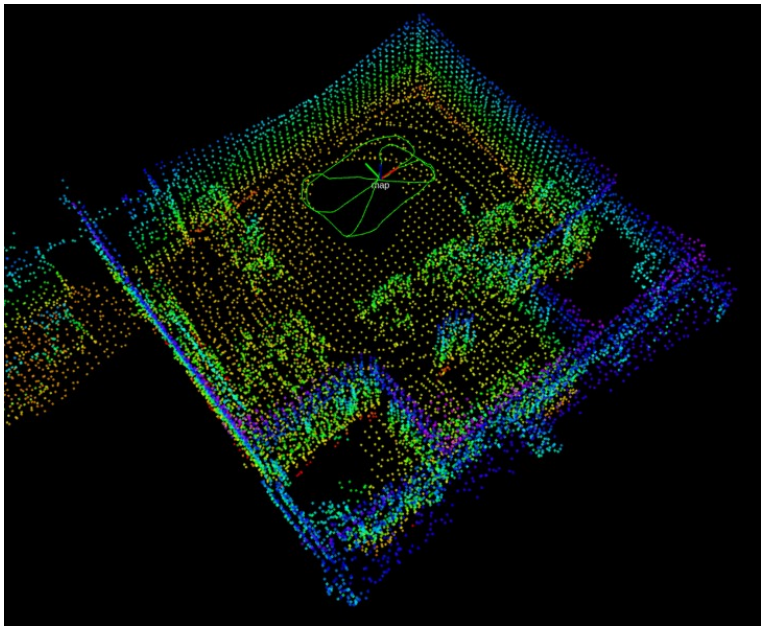
Fonte: Elaborada pelo autor.

Figura 5.10: Resultado do mapeamento por nuvem de pontos LaR utilizando o algoritmo LiDAR SLAM A-LOAM.



Fonte: Elaborada pelo autor.

Figura 5.11: Resultado do mapeamento por nuvem de pontos do LaR utilizando o algoritmo LiDAR SLAM F-LOAM.



Fonte: Elaborada pelo autor.

Capítulo 6

CONCLUSÃO

Neste Capítulo serão apresentadas as considerações finais da dissertação, com comentários relativos aos achados dos experimentos realizados em ambientes simulados e achados dos experimentos realizados com dados do ambiente real. Além disso, serão apresentadas as perspectivas para trabalhos futuros levando em consideração os achados alcançados com os experimentos realizados, as limitações do trabalho e as oportunidades de contribuição científica identificadas.

6.1 Considerações Finais

A escolha de comparar as técnicas A-LOAM, LeGO-LOAM e F-LOAM se justifica pela necessidade de avaliar diferentes abordagens de SLAM para LiDAR, cada uma otimizada para cenários específicos. O A-LOAM foi selecionado por sua capacidade de adaptação e equilíbrio entre precisão e desempenho, sendo ideal para ambientes dinâmicos e situações em que a flexibilidade da odometria é crucial. O LeGO-LOAM, com foco na otimização para ambientes terrestres, é particularmente relevante para cenários de grande escala, onde a precisão no mapeamento a longo prazo é essencial. Assim, é uma escolha robusta para mapeamento de áreas amplas com superfícies planas. Por fim, o F-LOAM foi escolhido por sua ênfase na eficiência computacional, tornando-se uma alternativa interessante para plataformas com recursos limitados, como drones ou robôs de pequeno porte, onde a velocidade de processamento em tempo real é mais importante do que a precisão absoluta.

Quanto à varredura realizada em ambientes simulados, os resultados confirmam a

eficácia dos algoritmos LiDAR SLAM avaliados na determinação da localização de robôs móveis e no mapeamento de ambientes, especialmente em cenários com maior número de referenciais estáticos. Em ambientes com poucos referenciais estáticos, os algoritmos também apresentaram desempenho satisfatório, com exceção do LeGO-LOAM no corredor reto sem janela, que registrou um erro de posição final de 12,57% ao longo do eixo x.

Na análise do Erro Médio Absoluto e Erro Quadrático Médio, o LeGO-LOAM apresentou um erro significativo no eixo x durante o deslocamento no corredor reto sem janela. Esse erro reflete uma estimativa acumulada ao longo do percurso, não corrigida devido à limitação na quantidade de referenciais estáticos. Já na simulação realizada no ambiente Lar, simulado no Gazebo, o erro médio absoluto e o erro quadrático médio demonstraram uma tendência de menores valores para o LeGO-LOAM.

No experimento realizado no Laboratório de Robótica da UFBA, os resultados dos algoritmos foram satisfatórios para todas as métricas analisadas, com destaque para a localização e o mapeamento obtidos pela técnica LeGO-LOAM, que apresentou o menor desvio de localização nas curvas. O erro de localização final também foi inferior ao das demais técnicas.

Como resultado geral da análise gráfica, observou-se uma tendência ao aumento do erro de posição no plano xy nas curvas, mas uma correção das estimativas de posição nos trechos retos. Essa análise sugere que os algoritmos LiDAR SLAM estudados não exibem tendência a erro acumulado ao longo da trajetória, ao contrário do que ocorre com algoritmos baseados em sensores proprioceptivos.

A análise qualitativa dos mapas gerados evidencia o potencial dos algoritmos estudados para o mapeamento de ambientes. Em todos os mapas, foi possível identificar os objetos presentes nas cenas. No entanto, na comparação entre os algoritmos, o LeGO-LOAM destacou-se por apresentar um maior nível de detalhamento, facilitando a percepção dos limites das estruturas mapeadas.

6.2 Trabalhos Futuros e Publicações

Esta dissertação apresentou resultados satisfatórios na aplicação de técnicas LiDAR SLAM para obtenção da odometria e mapeamento de ambientes. Os ambientes simulados foram criados com o objetivo de avaliar os resultados das técnicas em condições com baixo número de *features* geométricas, a exemplo de corredores longos e simétricos, que podem causar subestimação da pose do robô.

Como perspectiva de trabalhos futuros podem ser realizadas tentativas de fusão sensorial, com odometria das rodas, LiDAR e IMU, para obtenção mais precisa da pose do robô. Sendo possível utilizar dados de outros sensores para compensar as limitações de ambientes com poucas *features*.

Outra possibilidade em torno da fusão sensorial seria o uso de inteligência artificial para proporcionar a melhor estimativa de pose, incluindo a possibilidade de análise do ambiente pelo algoritmo e definição da melhor estratégia de localização, que poderia alternar de acordo as variações de luz, tipo de solo, volume de *features*, dentre outras. O sistema ajustaria os métodos de localização com base na confiabilidade das informações fornecidas pelos sensores, de forma similar a como um ser humano recorre ao tato em um ambiente escuro.

Uma abordagem alternativa seria aproveitar as poses instantâneas geradas pelos algoritmos para desenvolver sistemas de controle aplicados à robótica móvel ou manipuladores. Esse método permitiria ajustar automaticamente a trajetória para alcançar a posição final desejada. Além disso, seria possível integrar a segmentação de objetos detectados no mapa gerado e o planejamento simultâneo de trajetórias, evitando colisões com obstáculos e criando rotas otimizadas para atingir o objetivo de forma eficiente.

Como contribuição científica parte desta dissertação de mestrado foi apresentado oralmente, na *26th International Conference on Climbing and Walking Robots and Support Technologies for Mobile Machines* (CLAWAR 2023), o artigo intitulado *Comparative Analysis of LiDAR SLAM Techniques in Simulated Environments in ROS Gazebo*. Após o evento o artigo foi publicado na integra no ebook Synergetic Cooperation between Robots and Human [24], contribuindo significativamente para o avanço das pesquisas na área de localização robótica com o uso da técnica LiDAR SLAM.

Referências Bibliográficas

[1] CRUZ JUNIOR, G.P. et al. Investigação de técnicas lidar slam para um dispositivo robótico de inspeção de ambientes confinados. *Revista da SBA*, v. 2, n. 1, 2020.

[2] YAHYAEI, M.; et al. Review of exteroceptive sensors for autonomous driving. *IEEE 25th International Conference on Intelligent Transportation Systems (ITSC).*, 2022.

[3] ZHANG, Y.; SHI, P.; LI, J. 3D LiDAR SLAM: A survey. *The Photogrammetric Record.*, v. 39, p. 457-517, 2024.

[4] WANG, H. et al. F-LOAM: Fast lidar odometry and mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[5] BESL, P.J.; MCKAY, N.D. Method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 14, n. 2, p. 239-256, 2020.

[6] HONG, S.; HO, H.; KIM, J. Vicp: Velocity updating iterative closest point algorithm. *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[7] ZHANG, J.; SINGH, S. LOAM: Lidar odometry and mapping in real-time. *Robotics: Science and Systems*, v. 2, n. 9, 2014.

[8] ZHANG, J.; SINGH, S. Low-drift and real-time lidar odometry and mapping. *Autonomous Robot*, v. 41, n. 2, p. 401-416, 2017.

[9] SHAN, T.; ENGLOT, B. Lego-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 4758-4765, 2018.

[10] PENTZER, J.; BRENNAN, S.; REICHARD, K. Modelbased prediction of skid-steer robot kinematics using online estimation of track instantaneous centers of rotation. *Journal of Field Robotics*, v. 31, n. 3, 2014.

[11] ANOUSAKI, G.; KYRIAKOPOULOS, K. J. A dead-reckoning scheme for skid-steered vehicles in outdoor environment. *IEEE International Conference on Robotics and Automation.*, v. 1, p. 580-585, 2004.

[12] MARTÍNEZ, J. L.; et al. Approximating kinematics for tracked mobile robots. *International Journal of Robotics Research.*, v. 24, p. 867-878, 2005.

[13] MANDOW, A.; et al. Experimental kinematics for wheeled skid-steer mobile robots. *International Conference on Intelligent Robots and Systems*, p. 1222-1227, 2007.

[14] ZUO, X.; et al. Visual-based kinematics and pose estimation for skid-steering robots: Algorithm and theory. *IEEE Transactions on Automation Science and Engineering.*, v. 21, p. 91-105, 2022.

[15] BEN-ARI, M.; MONDADA, F. Elements of robotics. *Springer Nature*, 2017.

[16] XU, Y.; OU, Y.; XU, T. Slam of robot based on the fusion of vision and lidar. *International Conference on Cyborg and Bionic Systems (CBS), IEEE*, p. 121 – 126, 2018.

[17] CRUZ JUNIOR, G.P. Localização e mapeamento para robôs móveis em ambientes confinados baseado em fusão de lidar com odometrias de rodas e sensor inercial. 2021. 131 f. *Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Minas Gerais, Escola de Engenharia, Belo Horizonte*, 2021.

[18] BULA, J.; DERRON, M.H.; MARIETHOZ , G. Dense point cloud acquisition with a low-cost velodyne VLP-16. *Geosci. Instrum. Method. Data Syst.*, v. 9, p. 385-396, 2020.

[19] BESL, P.J.; MCKAY, N.D. Method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 14, n. 2, p. 239 - 256, 1992.

[20] KAESS, M.; et al. iSAM2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research.*, v. 31(2), p. 216-235, 2012.

[21] TONG, Q.; SHAOZU, C. Advanced implementation of loam. *Disponível em: https://github.com/HKUST-Aerial-Robotics/ALOAM*, Acesso em: 12 de Dezembro de 2022.

[22] ZHANG, L.; SUGANTHAN, P. N. Robust visual tracking via co-trained kernelized correlation filters. *Pattern Recognitioh.*, v. 69, p. 82-93, 2017.

[23] REN, W.; et al. Lidar-only 3d slam system comparative study. *IEEE 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2022.

[24] YOUSSEF, E. S. E.; et al. Synergetic cooperation between robots and humans - preceedings of the clawar 2023 conference. *Springer.*, v. 2, p. 91-105, 2023.