UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FELYPE DE LUCENA SILVA

INTEGRAÇÃO DE SENSOR LIDAR 2D PARA MAPEAMENTO E LOCALIZAÇÃO DE VEÍCULO AGV

CAMPINA GRANDE, PARAÍBA, BRASIL

FELYPE DE LUCENA SILVA

INTEGRAÇÃO DE SENSOR LIDAR 2D PARA MAPEAMENTO E LOCALIZAÇÃO DE VEÍCULO AGV

Dissertação de Mestrado submetida à Coordenadoria do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Aprovado (a) em: 27 / 03 / 2025 .

BANCA EXAMINADORA

_____

Eisenhawer de Moura Fernandes, D.Sc.

Orientador

_____

Péricles Rezende Barros, Ph.D.

Componente da Banca

_____

George Acioli Júnior, D.Sc.

Componente da Banca

CAMPINA GRANDE, PARAÍBA, BRASIL

S556i Silva, Felype de Lucena.
Integração de sensor LiDAR 2D para mapeamento e localização de veículo AGV / Felype de Lucena Silva – Campina Grande, 2025.
88 f. : il. color.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2025.
"Orientação: Prof. Dr. Eisenhawer de Moura Fernandes."
Referências.

1. Engenharia Elétrica. 2. Processamento da Informação. 2. Interface Interativa. 3. Aplicações SLAM. 4. Técnica ICP. 5. *Iterative Closest Point* (ICP). 6. Sensor LiDAR 2D. I. Fernandes, Eisenhawer de Moura. II. Título.

CDU 621.3(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA ITAPUANA SOARES DIAS GONÇALVES CRB-15/093

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM ENGENHARIA ELETRICA
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

REGISTRO DE PRESENÇA E ASSINATURAS

1 - ATA DA DEFESA PARA CONCESSÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA, REALIZADA EM 27 DE MARÇO DE 2025

(Nº 780)

CANDIDATO(A): **FELYPE DE LUCENA SILVA.** COMISSÃO EXAMINADORA: PÉRICLES REZENDE BARROS, Ph.D., UFCG - Presidente da Comissão e Examinador Interno, EISENHAWER DE MOURA FERNANDES, D.Sc., UFCG - Orientador, GEORGE ACIOLI JUNIOR, D.Sc., UFCG - Examinador Externo. TÍTULO DA DISSERTAÇÃO: INTEGRAÇÃO DE SENSOR LIDAR 2D PARA MAPEAMENTO E LOCALIZAÇÃO DE VEÍCULO AGV : Processamento da Informação. HORA DE INÍCIO: **08h30** – LOCAL: **Auditório do Liec.** Em sessão pública, após exposição de cerca de 45 minutos, o(a) candidato(a) foi arguido(a) oralmente pelos membros da Comissão Examinadora, tendo demonstrado suficiência de conhecimento e capacidade de sistematização, no tema de sua dissertação, obtendo o conceito APROVADO. Face à aprovação, declara o(a) presidente da Comissão, achar-se o examinado, legalmente habilitado a receber o Grau de Mestre em Engenharia Elétrica, cabendo a Universidade Federal de Campina Grande, como de direito, providenciar a expedição do Diploma, a que o(a) mesmo(a) faz jus. Na forma regulamentar, foi lavrada a presente ata, que é assinada por mim, LEANDRO FERREIRA DE LIMA, e os membros da Comissão Examinadora. Campina Grande, 27 de Março de 2025.

LEANDRO FERREIRA DE LIMA
Secretário

PÉRICLES REZENDE BARROS, Ph.D., UFCG
Presidente da Comissão e Examinador Interno

GEORGE ACIOLI JUNIOR, D.Sc., UFCG
Examinador Externo

EISENHAWER DE MOURA FERNANDES, D.Sc., UFCG
Orientador

FELYPE DE LUCENA SILVA
Candidato

2 - APROVAÇÃO

2.1. Segue a presente Ata de Defesa de Dissertação de Mestrado da candidato **FELYPE DE LUCENA SILVA ,** assinada eletronicamente pela Comissão Examinadora acima identificada.

2.2. No caso de examinadores externos que não possuam credenciamento de usuário externo ativo no SEI, para igual assinatura eletrônica, os examinadores internos signatários **certificam** que os examinadores externos acima identificados participaram da defesa da tese e tomaram conhecimento do teor deste documento.

Documento assinado eletronicamente por **LEANDRO FERREIRA DE LIMA**, **SECRETÁRIO (A)**, em 31/03/2025, às 09:28, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **PERICLES REZENDE BARROS**, **PROFESSOR 3 GRAU**, em 31/03/2025, às 10:00, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Felype de Lucena Silva**, **Usuário Externo**, em 31/03/2025, às 10:29, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **EISENHAWER DE MOURA FERNANDES**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 31/03/2025, às 11:45, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **GEORGE ACIOLI JUNIOR**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 01/04/2025, às 09:18, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **5342881** e o código CRC **FA06B695**.

Referência: Processo nº 23096.017189/2025-10 SEI nº 5342881

Dedicatória

Dedico este trabalho à memória da minha querida avó, Albanira, que permanece viva em meu coração. Aos meus pais, Herly e Nádia, e ao meu irmão, Fernando, pelo amor incondicional. À minha noiva, Taiane, pelo companheirismo e apoio. A todos os meus familiares pelo carinho e incentivo constante. A todos meus amigos e professores, que torcem por mim. Acima de tudo, dedico este trabalho ao maravilhoso Deus, a quem entrego toda honra e glória.

"O dom de poder mental vem de Deus, o Ser Divino e se concentrarmos nossas mentes na verdade, ficamos em sintonia com este grande poder."

Nikola Tesla

Agradecimentos

Primeiramente, agradeço a Deus, pois sem a presença dEle em minha vida, não teria chegado até aqui. Sua bênção guiou meus passos e iluminou minha trajetória, proporcionando forças para superar todos os desafios que encontrei.

Aos meus pais, Herly e Nadia, expresso minha profunda gratidão pelos ensinamentos, incentivo constante e apoio incondicional para buscar sempre o melhor para minha vida. Ao meu irmão Fernando, pela torcida e por sempre acreditar no meu sucesso. Agradeço ainda a toda minha família pelo apoio e carinho recebido.

À minha noiva, Taiane, agradeço especialmente por estar ao meu lado durante os últimos seis anos, compartilhando comigo os melhores e mais difíceis momentos da minha vida. Sua presença e apoio foram essenciais para vencer as batalhas enfrentadas durante o mestrado. Sou imensamente grato pela sua compreensão, paciência e pelo sacrifício que fez diversas vezes em prol da minha caminhada acadêmica.

Aos amigos que fiz no laboratório e na UFCG durante esses dois anos de mestrado, agradeço profundamente, pelos conselhos e pela assistência em momentos importantes. Em especial, agradeço aos amigos Levi Pimentel, Matheus Ferreira, Egídio Tadeu, Victor Luiz, Rossana Pontes, Felipe Pimenta e Felipe Cézar.

Expresso meu reconhecimento e gratidão aos meus orientadores. Ao professor José Sérgio, o primeiro que conheci na UFCG e que, enquanto esteve conosco, abriu as portas para minha jornada acadêmica, me acolhendo com generosidade. Ao professor Péricles, que me acolheu e orientou após a partida do professor José Sérgio, proporcionando continuidade em minha pesquisa. Ao professor Eisenhower, que me acompanhou no último ano do curso com orientação. Agradeço a todos pelo aprendizado, pela receptividade e confiança depositada.

Agradeço aos funcionários do laboratório pelo apoio cotidiano, em especial a Dona Nena e Lediane, pela torcida e pelo carinho demonstrado ao longo desta trajetória.

Por fim, agradeço à Universidade Federal de Campina Grande (UFCG), à Coordenação de Pós-Graduação em Engenharia Elétrica (COPELE) por todo suporte, ao Laboratório de Instrumentação Eletrônica e Controle (LIEC) pelo apoio da pesquisa, e ao CNPq pelo suporte financeiro para realização deste trabalho.

Resumo

Os AMRs diferenciam-se dos AGVs por sua navegação autônoma e capacidade de adaptação ao ambiente sem trilhas fixas. A navegação autônoma exige um sistema de percepção preciso de localização e mapeamento. No contexto do SLAM baseado em LiDAR, o *Iterative Closest Point* (ICP) destaca-se como um dos principais algoritmos para refinamento da localização e alinhamento de nuvens de pontos. Este trabalho propõe uma solução baseada em SLAM e ICP para navegação exclusivamente com um sensor LiDAR 2D, sem suporte de sensores adicionais e operando sob restrições computacionais. A solução foi implementada experimentalmente em um veículo AGV em laboratório, possibilitando sua reclassificação de AGV para AMR. As principais contribuições incluem: um estudo comparativo entre métodos ICP (clássico, Gauss-Newton ponto-a-ponto e ponto-a-plano) com dados reais; a implementação incremental do ICP, permitindo o alinhamento contínuo de múltiplos *scans*; a adaptação do sistema para operação em tempo real; e o desenvolvimento de uma interface interativa para visualização da trajetória e do mapa gerado. A técnica ICP proposta demonstrou melhor desempenho e foi validada experimentalmente, indicando seu potencial para expansão em outras aplicações SLAM envolvendo veículos em tempo real.

Palavras-chave: LiDAR 2D; SLAM; AGV; AMR; ICP.

Abstract

AMRs differ from AGVs due to their autonomous navigation capabilities and adaptability to environments without fixed tracks. Autonomous navigation requires precise perception, localization, and mapping systems. In the context of LiDAR-based SLAM, the Iterative Closest Point (ICP) stands out as one of the primary algorithms used for refining localization and point cloud alignment. This study proposes a SLAM and ICP-based solution for navigation exclusively utilizing a 2D LiDAR sensor, without additional sensor support, under computational constraints. The solution was experimentally implemented on an AGV vehicle in a laboratory, enabling its reclassification from an AGV to an AMR. The main contributions include: a comparative study of ICP methods (classical, Gauss-Newton point-to-point, and point-to-plane) using real data; incremental implementation of ICP, allowing continuous alignment of multiple scans; adaptation of the system for real-time operation; and the development of an interactive interface to visualize the trajectory and generated map. The proposed ICP technique demonstrated superior performance and was experimentally validated, highlighting its potential for extension to other real-time SLAM applications involving vehicles.

Keywords: LiDAR 2D; SLAM; AGV; AMR; ICP.

Sumário

Lista de abreviaturas, símbolos, siglas e acrônimos

Abreviaturas

AGV: Automatic Guided Vehicle 8

AMR: Autonomous Mobile Robot 8

CBS: Cyber-Physical Systems 8

IoS: Internet of Services 8

IoT: Internet of Things 8

LiDAR: Light Detection and Ranging 8

ROS: Robot Operating System 8

SLAM: Simultaneous Localization and Mapping 8

Siglas

ASM: Attention-based Scan Matching 23

DBN: Dynamic Bayesian Network 23

DFT: Discrete Fourier Transform 23

EKF: Extended Kalman Filter 23

ICP: Iterative Closest Point 8

LM: Levenberg-Marquardt 23

RBPF: Rao-Blackwellized Particle Filters 23

RGB-D: Red, Green, Blue - Depth 23

RMT: Relative Motion Threshold 23

SPA: Sparse Pose Adjustment 23

SVD: Singular Value Decomposition 23

UFCG: Universidade Federal de Campina Grande 8

WSDF: Weighted Signed Distance Function 23

Acrônimos

DSP: Digital Signal Processor 42

ICR: Instantaneous Center of Rotation 42

IMU: Inertial Measurement Unit 8

LIEC: Laboratório de Instrumentação, Eletrônica e Controle 8

LOAM: Lidar Odometry and Mapping 23

MBD: Model-Based Design 8

ORB: Oriented FAST and Rotated BRIEF 23

PI: Proporcional Integral 8

SDK: Software Development Kit 42

TI: Tecnologia da informação 8

ToF: Time of Flight 42

UART: Universal Asynchronous Receiver-Transmitter 42

USB: Universal Serial Bus 42

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Este capítulo apresenta uma contextualização do problema abordado, seguida pela motivação para o desenvolvimento do estudo. Além disso, são destacados os objetivos e as principais contribuições do trabalho.

1.1 Contextualização

Desde a Primeira Revolução Industrial, iniciada no final do século XVIII, a indústria tem passado por transformações tanto em seus processos produtivos quanto nos modelos de gestão. Atualmente, a Indústria 4.0 representa uma nova fase desse processo evolutivo, caracterizando-se pela convergência entre sistemas físicos e digitais. Essa transformação tecnológica ocorre por meio da incorporação de *Cyber-Physical Systems* (CPS), *Internet of Things* (IoT), *Internet of Services* (IoS), robótica avançada, veículos autônomos, entre outros pontos, impactando diretamente os conceitos atuais de automação industrial (TESSARINI; SALTORATO, 2018).

Os robôs industriais, evoluíram para incorporar maior autonomia e inteligência, permitindo que sejam integrados a sistemas produtivos altamente flexíveis. Essa evolução visa não apenas o aumento da produtividade, mas também a minimização de falhas e a melhoria das condições de trabalho, reduzindo a exposição humana a ambientes insalubres (YAMADA; MARTINS, 2019).

Nesse contexto, os veículos autoguiados e autônomos tornaram-se parte da Industria 4.0

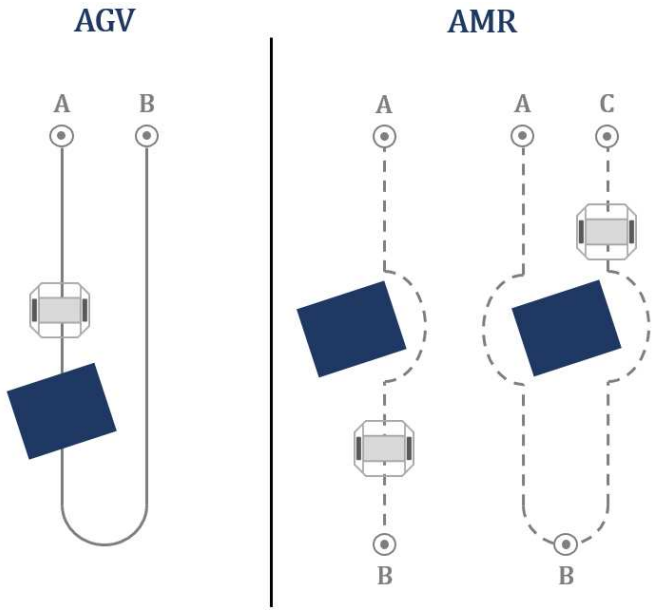
representando uma etapa para a automação, servindo para complementar sistemas integrados de manufaturas na logística e produção.

Os *Automated Guided Vehicles* (AGVs), por exemplo, são robôs móveis projetados para deslocamento em trajetórias pré-definidas, seguindo trilhas que restringem sua movimentação a rotas fixas (SILVA et al., 2024). Essa característica permite que os AGVs operem de maneira previsível e segura, reduzindo riscos operacionais e melhorando a eficiência logística dentro das fábricas (JUNIOR, 2022).

Por outro lado, os *Autonomous Mobile Robots* (AMRs) representam um avanço em relação aos AGVs, pois possuem maior capacidade de navegação autônoma, usando combinação de dados, sistema de navegação e algoritmos de controle (SUTTHI et al., 2023). Diferente dos AGVs, os AMRs não dependem de trilhas fixas, sendo capazes de planejar e ajustar suas rotas em tempo real, adaptando-se ao ambiente de forma dinâmica (SILVA et al., 2024).

Na Figura 1.1 são comparadas as diferenças entre esses dois tipos de robôs móveis. Ambas as categorias apresentam desafios complexos de desenvolvimento, exigindo integração sofisticada de sensores, algoritmos de controle e sistemas de navegação. Com esses avanços, tais robôs têm sido amplamente empregados para o transporte de cargas pesadas e execução de tarefas que envolvem riscos à saúde humana.

Figura 1.1: Diferenças de navegação entre AMR e AGV



Fonte: Autor.

Os sensores utilizados na navegação de robôs móveis industriais são selecionados de acordo com o sistema de navegação empregado pela unidade. Segundo LYNCH et al. (2018), os principais sensores utilizados para essa finalidade incluem o sensor guia magnético e o *scanner* a *laser*, cada um adequado a diferentes abordagens de navegação.

O sensor guia magnético é empregado em AGVs que utilizam sistemas de orientação baseados no acompanhamento de linhas pré-definidas (Figura 1.2). Instalado na parte inferior do veículo, ao identificar a posição do campo magnético, o sensor atua em conjunto com o controlador de motores do AGV, realizando ajustes na trajetória para manter o veículo alinhado ao centro da fita (DIAS, 2020).

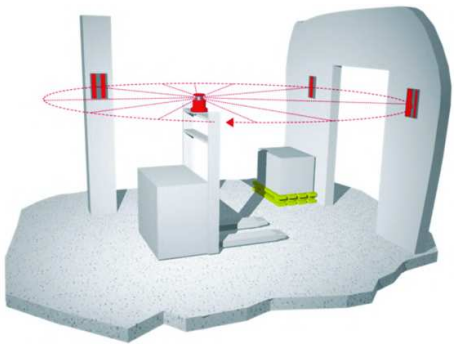
Figura 1.2: Representação do AGV com sensor guia magnético

Fonte: DIAS (2020).

Por outro lado, os sensores de *scanner* a *laser* são projetados para sistemas de navegação livre, sendo amplamente utilizados em AMRs. Diferentemente dos AGVs, que seguem trajetórias fixas, os AMRs são equipados com sensores LiDAR (*Light Detection and Ranging*), que permitem uma percepção detalhada do ambiente (Figura 1.3). Esse sistema consiste em um emissor e receptor de *laser* rotativo capaz de realizar leituras precisas do entorno, identificando tanto objetos estáticos quanto dinâmicos. A partir dessas leituras, o robô pode construir um mapa do ambiente e utilizá-lo para localização e navegação autônoma.

O sensoriamento desempenha um papel fundamental nos sistemas de navegação de robôs móveis, permitindo que eles percebam e interajam com o ambiente de forma eficiente e segura. Além dos sensores mencionados anteriormente, outros dispositivos são frequentemente utilizados para aprimorar a precisão da navegação e a capacidade de detecção de obstáculos, destacam-se os sensores ultrassônicos, os sensores de odometria e as câmeras.

Figura 1.3: Representação do AGV com sensor *scanner* a *laser*

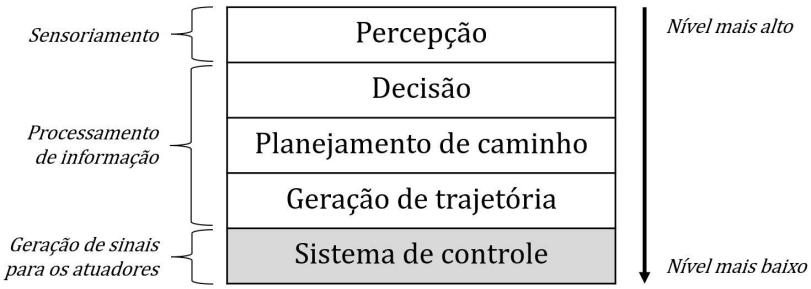


Fonte: DIAS (2020).

1.2 Motivação do trabalho

Segundo VIEIRA (2005), os sistemas de navegação de robôs móveis podem ser organizados hierarquicamente em diferentes níveis funcionais, que vão desde a percepção do ambiente até o controle dos atuadores. Cada nível corresponde a uma área específica da robótica, que pode ser estudada e aprimorada separadamente. Na Figura 1.4, observa-se essa divisão, evidenciando a relação entre os diferentes módulos do sistema.

Figura 1.4: Níveis hierárquicos da arquitetura de navegação para robôs móveis



Fonte: VIEIRA (2005).

A percepção é o nível mais alto da hierarquia e consiste em sensores e algoritmos responsáveis por coletar e interpretar informações do ambiente. Esse nível permite que o robô compreenda elementos essenciais, como localização, obstáculos e velocidade. Sensores como LiDAR, câmeras e ultrassônicos são frequentemente utilizados para fornecer dados brutos que serão processados para a navegação.

No nível de decisão, as informações coletadas na percepção são processadas e analisadas

para permitir a tomada de decisões. Esse processo pode envolver técnicas de inteligência artificial, como redes neurais, aprendizado de máquina e árvores de decisão, garantindo que o robô possa responder de forma autônoma a diferentes cenários.

Com base nas decisões tomadas, o robô precisa determinar a melhor rota a ser seguida dentro do ambiente. Esse nível define o caminho ideal, considerando obstáculos, restrições ambientais e as dimensões do robô.

A geração de trajetória estabelece restrições temporais e define como o caminho previamente planejado será percorrido. Esse processo envolve cálculos que determinam velocidade, aceleração e suavização de trajetórias, empregando técnicas como polinômios de interpolação para garantir movimentos contínuos e seguros.

No nível mais baixo da hierarquia, o sistema de controle é responsável por executar os comandos definidos nos níveis anteriores. Isso ocorre por meio do envio de sinais para os atuadores, garantindo que o robô siga a trajetória planejada com precisão. Controladores como PI (*Proporcional-Integral*) e controle baseado em modelos dinâmicos são comumente aplicados para ajustar a movimentação do robô.

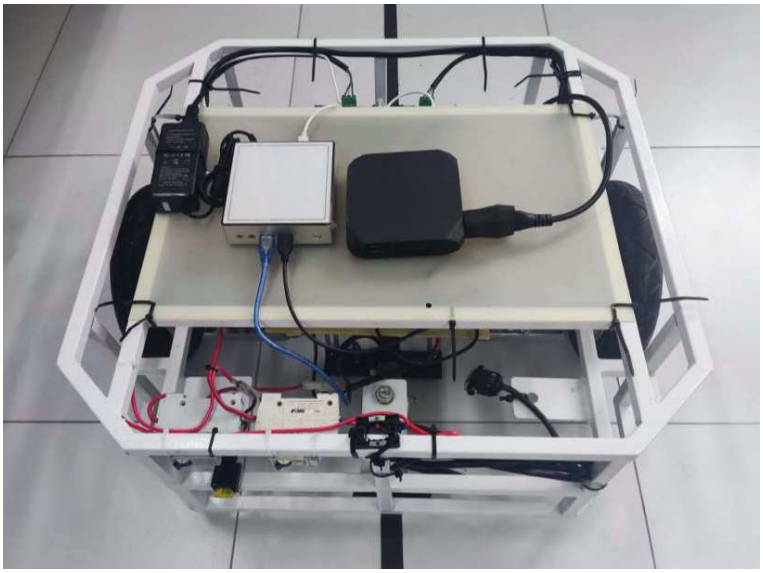
Embora todos os níveis sejam essenciais para a navegação autônoma de robôs móveis, o trabalho concentra-se no nível de percepção, explorando o LiDAR 2D e algoritmos de percepção.

Diante deste cenário, esta pesquisa se insere no estudo e desenvolvimento de um sistema voltado ao suporte da navegação de um veículo AGV em laboratório, comum em ambiente industrial. O objetivo é investigar e implementar soluções com foco na percepção sensorial, localização e mapeamento de ambiente.

O veículo em questão é uma plataforma robótica experimental (Figura 1.5) desenvolvida por iniciativa do Laboratório de Instrumentação Eletrônica e Controle (LIEC) da Universidade Federal de Campina Grande (UFCG), para atividades de ensino e pesquisa. O último trabalho realizado com a plataforma por JUNIOR (2022) abordou, entre outros aspectos, o controle de trajetória para seguimento de trilha magnética, utilizando um sensor guia magnético. No entanto, como discutido anteriormente, esse tipo de sistema de navegação apresenta limitações, pois opera exclusivamente em trajetos fixos e previamente planejados.

É essencial que robôs móveis sejam capazes de oferecer cada vez mais flexibilidade e percepção aprimorada, tornando-os adequados para aplicações em ambientes dinâmicos.

Figura 1.5: Plataforma experimental do AGV (2022)



Fonte: JUNIOR (2022).

Frente à necessidade de expandir a capacidade de navegação da plataforma experimental e visando aumentar sua autonomia, este trabalho propõe integrar um sensor LiDAR 2D ao sistema para permitir sua localização e a construção em tempo real de um mapa do ambiente ao seu redor. Essa modificação não apenas ampliará as funcionalidades da plataforma, mas também possibilitará sua reclassificação para a categoria de AMR, elevando seu nível de autonomia e adaptabilidade a ambientes.

São inúmeras as diferentes abordagens encontradas para solucionar o problema de localização com robôs móveis, cada uma com características e aplicações específicas. A localização pode ser aprimorada por meio do uso de sensores LiDAR 2D ou 3D, câmeras e IMUs (*Inertial Measurement Units*), frequentemente combinados com diferentes técnicas de fusão sensorial e algoritmos de estimativa de posição (CAPRIYANI et al., 2024).

Por meio do uso de sensores apropriados, o SLAM (*Simultaneous Localization and Mapping*) é reconhecida como uma técnica essencial para possibilitar a construção de um mapa do ambiente enquanto estima simultaneamente a posição atual do robô (ASLAM et al., 2020). Dentre as abordagens existentes, o SLAM baseado em LiDAR destaca-se como uma das mais eficazes, pois fornece medições de alcance de alta frequência e elevada precisão.

Dentro desse contexto, o *Iterative Closest Point* (ICP), um método de registro que

busca minimizar a diferença entre duas nuvens de pontos, surge como um dos principais algoritmos para refinamento da localização e alinhamento de nuvens de pontos obtidas por sensores LiDAR. Essa técnica é amplamente utilizada no SLAM para aprimorar a precisão da estimativa de pose e garantir que as informações mapeadas sejam consistentes ao longo do tempo.

A navegação autônoma de robôs móveis requer um sistema de percepção que permita a localização precisa e a construção de mapas detalhados do ambiente. Este trabalho propõe aprimoramentos no sistema de percepção de um AGV, com ênfase na integração de um sensor LiDAR 2D e na implementação de técnicas de mapeamento e localização em tempo real. O objetivo é permitir que o robô desenvolva consciência do ambiente ao seu redor e se localize dentro dele, superando as limitações do sistema anterior, que se baseava exclusivamente em um sensor magnético para seguimento de linha, sem capacidade de percepção do entorno.

1.3 Objetivos

1.3.1 Objetivo geral

O objetivo geral deste trabalho é aprimorar a capacidade de percepção de um veículo AGV por meio da integração de um sensor LiDAR 2D e da aplicação de técnicas de SLAM baseadas em ICP, visando mapear e localizar o robô em tempo real, possibilitando sua reclassificação AGV para AMR.

1.3.2 Objetivos específicos

- Integrar sensor LiDAR 2D a plataforma AGV em laboratório;
- Avaliar variantes do método ICP e identificar abordagem mais adequada;
- Desenvolver sistema de localização e mapeamento de ambiente desconhecido utilizando dados do sensor LiDAR 2D;
- Desenvolver interface interativa para visualizar pose, trajetória e mapa gerado.

1.4 Contribuições

A proposta estabelece uma base sólida para futuras estratégias de planejamento e controle de trajetória, permitindo que o AGV atue em ambientes estruturados. As principais contribuições deste trabalho são:

- Integração de um sensor LiDAR 2D ao veículo AGV: desenvolvimento da infraestrutura necessária para aquisição e processamento contínuo de dados de varredura, sem depender de sensores adicionais;
- Estudo comparativo entre métodos ICP: comparação experimental entre o ICP clássico e variantes otimizadas com o método de Gauss-Newton, utilizando métricas ponto-a-ponto e ponto-a-plano, com dados reais do sensor utilizado;
- Execução incremental do ICP: desenvolvimento de uma arquitetura que realiza o alinhamento contínuo entre varreduras consecutivas, promovendo a estimação de pose e a construção progressiva do mapa do ambiente;
- Implementação em tempo real com a interface gráfica: adaptação do sistema desenvolvido para operar em um mini-PC embarcado com recursos computacionais limitados, além do desenvolvimento de uma interface gráfica com ferramenta Qt para visualização em tempo real das varreduras, pose estimada e evolução do mapa.

1.5 Organização do texto

Nos próximos capítulos, são abordados tópicos essenciais para a compreensão deste trabalho, organizados em seis capítulos, incluindo este capítulo introdutório:

- Capítulo 2 - Estado da Arte
- Capítulo 3 - Iterative Closest Point (ICP)
- Capítulo 4 - Sensor LiDAR
- Capítulo 5 - Resultados e Discussões
- Capítulo 6 - Conclusões e Sugestões para Trabalhos Futuros

Capítulo 2

Estado da Arte

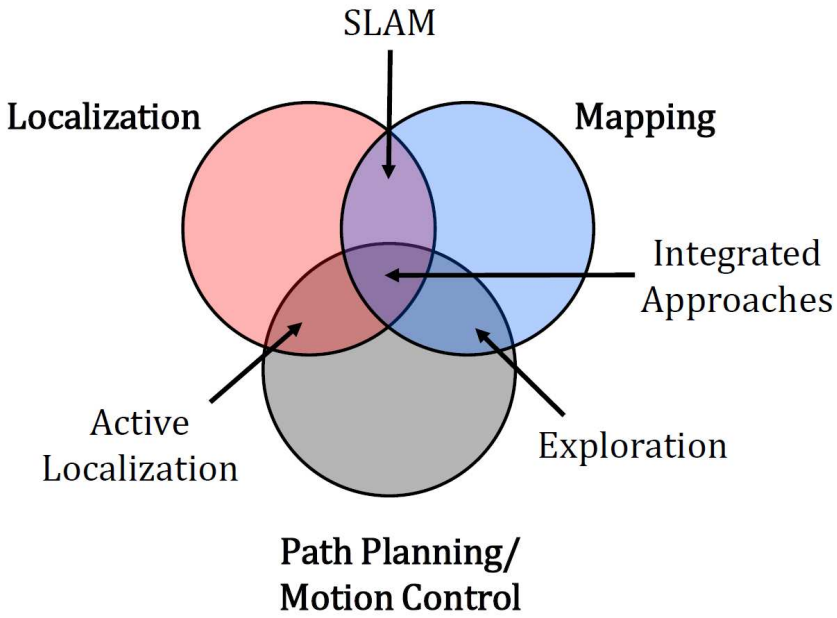
Neste capítulo, é apresentada uma introdução ao problema do SLAM, em seguida, as principais abordagens para resolver o problema. Por fim, o capítulo apresenta uma revisão dos principais trabalhos publicados nos últimos anos relacionados à aplicação do ICP para navegação.

2.1 Introdução ao SLAM

O problema *Simultaneous Localization and Mapping* (SLAM) consiste em construir um mapa do ambiente enquanto se determina, em tempo real, a posição do robô dentro desse mapa (STACHNISS, 2009). No diagrama apresentado na Figura 2.1 são ilustras as tarefas de mapeamento, localização e planejamento de trajetórias, destacando também os desafios combinados nas áreas em que essas funções se sobrepõem. Essas duas tarefas estão intrinsecamente interligadas, tornando inviável resolvê-las de forma isolada. Por isso, o SLAM é frequentemente descrito como o "problema do ovo ou da galinha": um mapa confiável é essencial para uma boa localização, mas uma localização precisa é necessária para construir um bom mapa (STACHNISS, 2009).

A localização de veículos autônomos tem sido um problema altamente complexo e controverso. As tecnologias de localização diferem com base no ambiente e requisitos de eficiência, precisão, velocidade e confiabilidade (KHAN et al., 2021). Atualmente, o SLAM é um campo bem investigado na robótica e existem muitos algoritmos, que são usados ativamente tanto

Figura 2.1: Descrição de tarefas de localização, mapeamento e planejamento de caminho



Fonte: Adaptado de STACHNISS (2009).

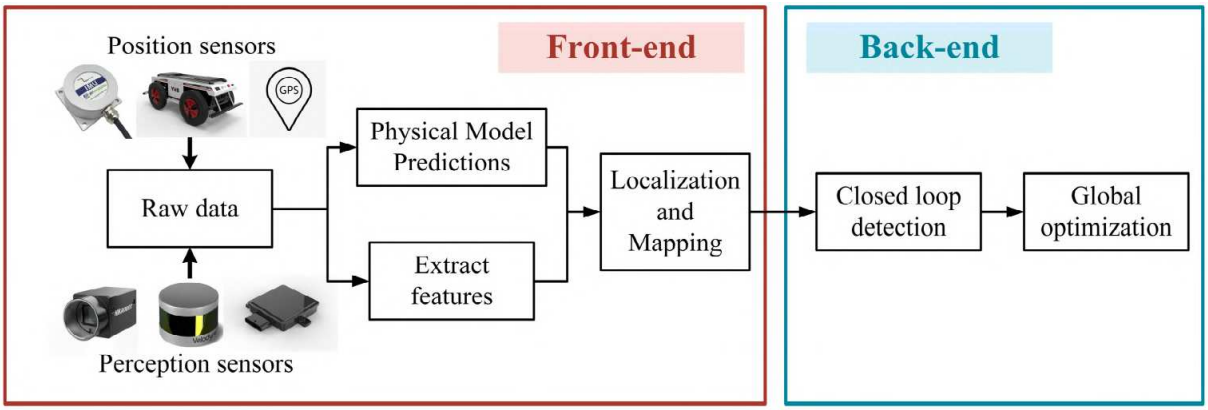
a pesquisa e em problemas aplicados (YAGFAROV; IVANOU; AFANASYEV, 2018).

Os sistemas de SLAM são baseados a partir de dados de sensores, como LiDAR, câmeras, sensores ultrassônicos, radar de ondas milimétricas, entre outros. Desde 1990, a estrutura do SLAM de fusão baseada em recursos foi estabelecida e ainda está em uso hoje (XU et al., 2022). Esta estrutura é baseada no Filtro de Kalman Estendido (EKF, do inglês *Extended Kalman Filter*) para calcular distribuições dorsais sobre a postura do robô, bem como posições de referência. Além disso, o robô identifica sua própria localização e orientação por meio da identificação repetida de características espaciais ao longo do movimento e, em seguida, produz um mapa progressivo da região circundante com base em sua própria posição, atingindo o objetivo de posicionamento simultâneo e formação de mapa (KHAN et al., 2021).

Conforme ilustrado na Figura 2.2, os sistemas de SLAM mais modernos, segundo XU et al. (2022), são divididos em duas partes: *front-end* e *back-end*. O *front-end* é responsável por estimar a posição atual do quadro em tempo real e armazenar as informações correspondentes do mapa. O *back-end* é responsável pela posição em larga escala e otimização do cenário. A detecção de fechamento de *loop* é uma das principais questões do SLAM,

que ajuda o robô a identificar cenários visitados e acionar a correção de desvio em escala global. A otimização global em larga escala também é a principal diferença entre o SLAM e a odometria moderna. Os dois métodos têm muitas semelhanças na estimativa de posição. A maioria das modernas tecnologias de fusão multissensorial atua no *front-end* para obter alta precisão e baixa deriva dos sistemas odométricos por meio da complementação de informações, fusão de pose local, e filtragem de fonte de vários dados.

Figura 2.2: Principais componentes de um sistema SLAM



Fonte: XU et al. (2022).

A tecnologia SLAM tem se mostrado extremamente útil no contexto industrial, especialmente em setores como manufatura, cadeia de suprimentos e gestão de armazéns. Ela é essencial para robôs que operam em ambientes desconhecidos ou parcialmente conhecidos. Essas capacidades são fundamentais para tarefas como evitar obstáculos e planejar rotas. Recentemente, grande parte das pesquisas científicas tem avançado na direção de sistemas de múltiplos robôs (VARGHESE et al., 2023).

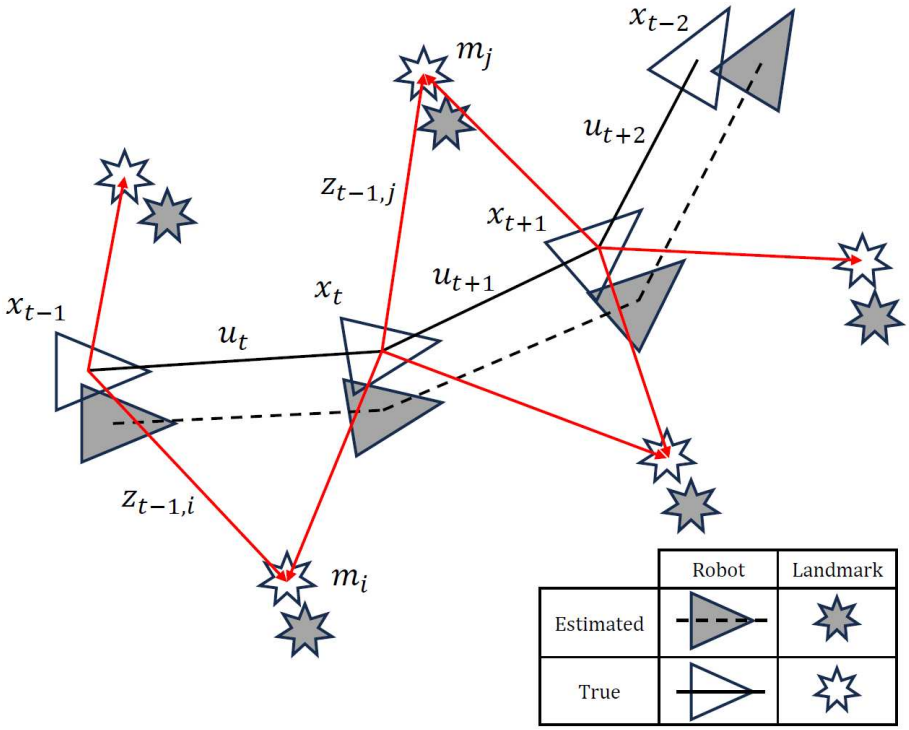
2.1.1 Descrição do problema do SLAM

O princípio fundamental da abordagem SLAM é utilizar o ambiente para corrigir a pose do robô. Isso é realizado por meio da extração de características do ambiente (*Landmark extraction*) e sua reobservação à medida que o robô se desloca (*Landmark update*) (SOULIMAN, 2017). O processo de modelagem do problema de SLAM é representado na Figura 2.3. Portanto, pode ser formulado como a capacidade de um veículo não tripulado de estimar

seu próprio movimento, localizar-se e construir um mapa em um ambiente desconhecido (QIU et al., 2024).

No processo de SLAM, utiliza-se o modelo de movimento, com entradas de controle afetadas por ruídos, e o modelo de observação, com sensores como câmeras, LiDAR, IMU e outros dispositivos de medição quantitativa também sujeitos a ruído, para realizar as estimativas (QIU et al., 2024). O processo de Markov probabilístico pode ser a melhor descrição para o SLAM. A pose atual do robô, x_t, pode ser expressa como uma função probabilística da pose no instante de tempo anterior, x_{t-1}, e do controle u_t executado pelo robô (HOUSEIN; XINGYU, 2021).

Figura 2.3: Diagrama de descrição do problema SLAM



Fonte: Adaptado de QIU et al. (2024), JIA et al. (2022) e SOULIMAN (2017).

Considerando a Figura 2.3, seja X_t o estado posicional do veículo não tripulado no momento t, onde $X_t = (x_t, y_t, \theta_t)^T$. A variável u_t representa o controle do estado do veículo não tripulado no intervalo de tempo de $t-1$ a t. A variável m_i corresponde à posição do i-ésimo ponto de característica, enquanto $z_{t,i}$ denota o valor quantitativo de m_i observado pelo LiDAR ou pela câmera no instante t.

2.1.2 Abordagens SLAM

Gmapping

O Gmapping é um algoritmo amplamente utilizado para SLAM 2D, baseado na técnica de *Rao-Blackwellized Particle Filters* (RBPF). O algoritmo do Gmapping segue a abordagem do filtro de partículas, onde cada partícula representa uma possível pose do robô no ambiente e carrega um mapa individual correspondente. Para aumentar a eficiência, a técnica de reamostragem adaptativa avalia continuamente a dispersão das partículas e executa a reamostragem somente quando necessário, evitando desperdício de recursos computacionais (YAGFAROV; IVANOU; AFANASYEV, 2018). A decisão sobre o momento da reamostragem é feita com base em um limiar definido pela equação:

$$N = \frac{1}{\sum_{m=1}^{M} \omega[m]^2}, \tag{2.1}$$

onde M é o número total de partículas e $\omega[m]$ representa o peso normalizado da m-ésima partícula. Quando N cai abaixo de um determinado limiar ($N/2$), o algoritmo realiza a reamostragem, garantindo uma melhor distribuição das partículas e aumentando a precisão da estimativa de pose (ZOU et al., 2022).

O Gmapping melhora a precisão do mapeamento ao integrar as observações mais recentes dos sensores ao modelo de movimento do robô, reduzindo a incerteza da predição durante a execução do filtro de partículas (YAGFAROV; IVANOU; AFANASYEV, 2018). No entanto, o método depende fortemente da odometria, utilizando-a para restringir o espaço de busca das partículas e reduzir a necessidade de um grande número de amostras. Essa característica pode ser um problema em terrenos irregulares ou inclinados, onde a odometria tende a apresentar maior imprecisão (XUEXI et al., 2019). Outro fator limitante do Gmapping é a necessidade crescente de memória e processamento computacional à medida que o tamanho do mapa aumenta.

Cartographer

O Cartographer é um algoritmo de SLAM em tempo real desenvolvido pelo Google, projetado para operar tanto em 2D quanto em 3D, suportando diversas plataformas, como

drones e robôs móveis. O Cartographer não utiliza abordabem baseada em filtro de partículas, pois esse método demanda um alto custo computacional para mapear grandes áreas. Em vez disso, adota uma estratégia de otimização gráfica, que permite reduzir erros acumulativos e realizar fechamento de loop de forma eficiente (ZOU et al., 2022). Para isso, o método se baseia em três componentes principais: correspondência de *scans*, construção de submapas e otimização gráfica global (YAGFAROV; IVANOU; AFANASYEV, 2018).

O processo de correspondência de *scans* ocorre dentro de submapas, onde cada nova varredura do sensor LiDAR é comparada com um conjunto de varreduras recentes para encontrar a melhor correspondência (ZOU et al., 2022). O alinhamento das varreduras no submapa é feito pelo Ceres Scan Matcher, um otimizador baseado no método de mínimos quadrados não lineares, formulado como:

$$f(t,\theta) = \underset{t,\theta}{\operatorname{argmin}} \sum_{i=1}^{n} (1 - M_{smooth}(t_i, \theta_i))^2, \tag{2.2}$$

onde t_i, θ_i denotam a tradução e a orientação, respectivamente, que são relativas ao quadro do submapa em vez do quadro global do mundo. $M_{\text{smooth}}(\cdot)$ suaviza os valores de probabilidade do mapa local (ZOU et al., 2022).

Ao longo do tempo, os erros acumulativos inerentes à estimativa de pose podem degradar a precisão do mapeamento. Para corrigir isso, o Cartographer emprega um mecanismo de fechamento de *loop* que detecta varreduras previamente registradas e ajusta todas as poses do mapa para minimizar inconsistências. Esse ajuste é feito pelo algoritmo Sparse Pose Adjustment (SPA), que otimiza a posição de todas as varreduras e submapas para garantir a melhor correspondência global (ZOU et al., 2022).

O Cartographer é uma solução robusta para SLAM em tempo real. Entretanto, há algumas limitações a serem consideradas: Requer maior capacidade computacional, devido à otimização gráfica; Possui dependência de múltiplos sensores. Pode até operar apenas com LiDAR 2D, mas a precisão melhora significativamente quando IMU e outros sensores são utilizados.

HectorSLAM

O Hector SLAM é um sistema SLAM baseado exclusivamente em dados de varredura de um sensor LiDAR 2D, sem dependência de odometria, o que o torna uma abordagem diferenciada de navegação em ambientes internos.

O funcionamento do Hector SLAM pode ser dividido em dois subsistemas principais: subsistema 2D SLAM e subsistema de navegação 3D (aplicável a drones e robôs aéreos). Para o SLAM 2D a correspondência dos *scans* é realizada implicitamente com as varreduras anteriores, permitindo a construção do mapa ao longo do tempo (ZOU et al., 2022). O ajuste fino da pose do robô ocorre por meio da otimização Gauss-Newton, garantindo a melhor correspondência entre a nova varredura e o mapa existente (NAGLA, 2020). Este procedimento é formulado por

$$f(t,\theta) = \underset{t,\theta}{\operatorname{argmin}} \sum_{i=1}^{n} [1 - M(t_i,\theta_i)]^2, \tag{2.3}$$

onde t_i, θ_i representam a translação e orientação, respectivamente, da pose nas coordenadas globais 2D no tempo i, $M(t_i,\theta_i)$ transforma a pose nas coordenadas globais nos mapas de grade e $f(t,\theta)$ é a soma dos erros residuais.

Uma das características marcantes do Hector SLAM é sua simplicidade estrutural, pois não inclui um *backend* de otimização de gráfico de poses, diferentemente de abordagens como Gmapping ou Cartographer. Isso o torna altamente eficiente para aplicações em robôs móveis de baixo custo, onde a ausência de odometria pode ser um fator limitante para outras técnicas (XUEXI et al., 2019).

ORB-SLAM (Visual SLAM)

Diferente do SLAM baseado em LiDAR, que depende de varreduras a *laser* para estimar a posição e mapear o espaço, o Visual SLAM (V-SLAM) deriva informações ricas e semânticas do ambiente a partir de imagens capturadas por câmeras monoculares, estereoscópicas ou RGB-D (WANG; MIMA; TOBITA, 2023).

O ORB-SLAM foi desenvolvido com base no uso do detector de características ORB (*Oriented FAST and Rotated BRIEF*), um método rápido e eficiente para extração de pontos

de interesse em imagens, permitindo que o algoritmo funcione em tempo real (IBRAGIMOV; AFANASYEV, 2023).

O ORB-SLAM opera em três estágios principais (ABDELAZIZ; EL-RABBANY, 2022): Rastreamento (*Tracking*), Mapeamento Local (*Local Mapping*) e Fechamento de Loop (*Loop Closing*). O desempenho do ORB-SLAM depende fortemente do tipo de câmera utilizada. As abordagens mais comuns incluem (ZHANG et al., 2023): Monocular, Binocular (Estéreo) e RGB-D (Profundidade + Cor).

O Visual SLAM apresenta diversas vantagens, como, utilizar câmeras convencionais, tornando-o acessível e permitir reconhecimento de objetos e segmentação semântica do ambiente. Porém, o desempenho pode ser prejudicado em ambientes com baixa luminosidade.

LOAM (LiDAR Odometry and Mapping)

O LOAM (*Lidar Odometry and Mapping*) é uma das abordagens mais avançadas para estimativa de pose e mapeamento tridimensional utilizando sensores LiDAR 3D. Desenvolvido com o objetivo de fornecer alta precisão na localização, o LOAM adota uma estratégia baseada em duas etapas independentes: odometria de alta frequência e baixa precisão e mapeamento de baixa frequência e alta precisão (ZOU et al., 2022).

O funcionamento do LOAM pode ser dividido em três principais parte. A primeira, é a extração de características geométricas, onde cada ponto da nuvem é classificado como uma aresta (representam descontinuidades nos objetos do ambiente) ou superfície (áreas mais homogêneas), analisando a rugosidade do entorno (HONDA et al., 2022). A segunda, é odometria LiDAR de alta frequência, que utiliza um modelo de otimização geométrica que minimiza a distância ponto-reta e ponto-plano para estimar a transformação entre varreduras consecutivas (YOUJI et al., 2019). Por fim, o mapeamento de baixa frequência, as nuvens de pontos são alinhadas e refinadas utilizando uma otimização mais precisa, essa etapa garante que o mapa gerado seja consistente, minimizando o erro global.

As maiores vantagens de usar o LOAM são: Alta precisão na estimativa da pose devido à separação entre odometria rápida e mapeamento refinado; Menor sensibilidade a falhas de correspondência de pontos; Pode ser integrado a outros sensores, como IMUs e câmeras. Entretanto, depende de LiDAR 3D e exige alto poder computacional.

Desde sua introdução, diversas extensões do LOAM foram desenvolvidas para aprimorar seu desempenho, como LeGO-LOAM e V-LOAM.

2.1.3 Avaliação das abordagens de SLAM e justificativa para o ICP baseado em SLAM

Foram analisadas diversas abordagens para a solução do problema de SLAM, considerando métodos baseados em LiDAR, visão computacional e otimização gráfica. Cada uma dessas técnicas apresenta vantagens e limitações que as tornam mais adequadas para determinados tipos de aplicações. Para fins de comparação, a Tabela 2.1 resume as principais características dessas abordagens.

Dentre os critérios analisados na comparação, destacam-se:

- Tipo de Sensor Utilizado: Diferentes abordagens dependem de LiDAR 2D/3D, câmeras RGB-D, IMU e odometria para estimar a pose do robô e construir um mapa;
- Dependência de Odometria: Alguns métodos, como Gmapping, exigem odometria confiável para minimizar erros acumulativos, enquanto outros, como Hector SLAM, podem operar apenas com dados do LiDAR;
- Mecanismo de Fechamento de *Loop*: Algumas técnicas possuem um sistema robusto de detecção de ciclos no mapa, como Cartographer e ORB-SLAM, enquanto outras, como Hector SLAM, não possuem um fechamento de *loop* explícito;
- Precisão na Estimativa da Pose: Métodos que utilizam otimização gráfica e refinamento contínuo dos dados, como LOAM e Cartographer, tendem a apresentar maior precisão;
- Complexidade Computacional: Algumas técnicas, como filtros de partículas (Gmapping) e Graph-SLAM (Cartographer), requerem alto poder computacional, o que pode ser um fator limitante para aplicações em robôs embarcados.

Com base na análise comparativa, observa-se que nenhuma abordagem isolada atende a todas as necessidades de um sistema SLAM ideal para aplicações embarcadas utilizando

exclusivamente LiDAR 2D. Abordagens como LOAM e Cartographer demonstram alta precisão na estimativa da pose, mas requerem sensores adicionais, como IMU e LiDAR 3D, além de demandarem maior capacidade computacional devido ao uso de otimização gráfica complexa.

Outro fator crítico é a dependência de odometria. Métodos como Gmapping utilizam informações da odometria para restringir a busca de partículas e aumentar a precisão da localização. No entanto, em terrenos irregulares ou em condições onde a odometria não é confiável, essa dependência pode comprometer a estimativa da pose. Além disso, a complexidade computacional das abordagens analisadas influencia sua aplicabilidade em sistemas embarcados. Técnicas como Cartographer e ORB-SLAM, baseadas em otimização gráfica e mecanismo fechamento de *loop*, são computacionalmente intensivas, exigindo *hardware* mais robusto para operação em tempo real. Por outro lado, métodos como Hector SLAM, embora eficientes e independentes da odometria, não possuem um fechamento de *loop* robusto, o que pode resultar em erros acumulativos e comprometer a consistência do mapa em deslocamentos longos.

Diante do exposto, a escolha por desenvolver uma estrutura ICP baseada em SLAM justifica-se por oferecer um equilíbrio entre precisão, eficiência computacional e independência de sensores adicionais. Inspirado em técnicas como o Hector SLAM, a abordagem desenvolvida permite estimar a pose do robô utilizando exclusivamente dados do LiDAR 2D, sem depender de modelos probabilísticos complexos ou de informações de odometria. Além disso, a técnica baseada em *Iterative Closest Point* (ICP) e otimização Gauss-Newton possibilita um refinamento iterativo da pose estimada, garantindo um alinhamento preciso das nuvens de pontos e minimizando a propagação de erros ao longo do tempo.

Tabela 2.1: Visão comparativa de abordagens SLAM em relação a sensores, métodos e desempenho

Abordagem	Tipo de Sensor	Odometria	Método Principal	Fechamento de Loop	Otimização	Precisão	Complexidade	Aplicações
Hector SLAM	LiDAR 2D	Não	Scan Matching	Não	Gauss-Newton	Média	Média	Robôs móveis, drones
Gmapping	LiDAR 2D + Odometria	Sim	Filtro de Partículas	Sim	FastSLAM	Média	Alta	Robôs móveis
Cartographer	LiDAR 2D /3D + IMU	Sim	Otimização Gráfica	Sim	Graph-SLAM	Alta	Alta	Veículos autônomos
ORB-SLAM	Câmera (Mono, Estéreo, RGB-D)	Não	Rastreamento de Feições	Sim	Bundle Adjustment	Alta	Média	Realidade aumentada, robôs móveis
LOAM	LiDAR 3D + IMU	Sim	Extração de Feições	Sim	Ajuste de Feições	Alta	Alta	Veículos autônomos, drones
ICP-based SLAM	LiDAR 2D	Não	ICP + Gauss-Newton	Não	Gauss-Newton	Alta	Média	Robôs móveis

Fonte: Autor.

O ICP e suas variantes são amplamente aplicados no alinhamento incremental de varreduras LiDAR, atuando em etapas de *scan-matching*, refinamento de pose e atualização de mapas. No entanto, seu uso em SLAM ainda enfrenta desafios como acúmulo de erro (*drift*), degeneração geométrica, custo computacional elevado e dependência da correspondência precisa entre pontos, o que pode comprometer a convergência em ambientes pouco estruturados.

Entre 2022 e 2025, diversos estudos propuseram melhorias no ICP para aumentar a precisão, reduzir efeitos degenerativos e otimizar o desempenho. As inovações incluem novas métricas de erro (ponto-a-ponto, ponto-a-plano e híbridas), métodos de otimização adaptativa, integração com dados inerciais e estratégias robustas para lidar com ambientes dinâmicos.

A seção seguinte revisa os principais trabalhos recentes, destacando avanços, limitações e lacunas que motivam as contribuições deste estudo.

2.2 Trabalhos relacionados

LEE; LIM; HAN (2025) propõem o GenZ-ICP[1], voltado para odometria com LiDAR 3D. O método introduz pesos adaptativos, ajustados conforme a geometria do ambiente, permitindo a combinação dinâmica das métricas ponto-a-ponto e ponto-a-plano. Essa adaptação melhora o desempenho em ambientes com degeneração direcional, como corredores estreitos. O algoritmo foi testado apenas com datasets previamente coletados, sem validação em tempo real.

VO; HA (2024) aprimoram o ICP com foco em rejeição de outliers, por meio das técnicas *Relative Motion Threshold* (RMT) e *Robust Kernel*. A abordagem visa manter a precisão da pose e a consistência do mapa diante de dados ruidosos. A validação foi feita por simulação no MATLAB com um dataset virtual. Utiliza SVD na otimização da transformação, o que pode causar acúmulo de erro ao longo do tempo.

LI et al. (2024) propõem melhorias no SLAM 2D por meio do método de otimização Levenberg–Marquardt (LM) e introduzem a *Weighted Signed Distance Function* (WSDF) para aumentar a acurácia na reconstrução do mapa. Comparado a métodos como Hector-SLAM

[1] <https://github.com/cocel-postech/genz-icp>

e Cartographer, o WSDF apresenta maior precisão, mas exige alto poder computacional, o que pode limitar seu uso em tempo real. A validação foi feita com robô real utilizando ROS (*Robot Operating System*) e LiDAR 2D.

FILOTHEOU et al. (2023) desenvolvem um método correspondence-less baseado em DFT para scan matching entre mapas, utilizando LiDAR panorâmico. A proposta dispensa correspondência direta entre pontos, aumentando a robustez em ambientes ruidosos. Embora apto para tempo real, não há validação prática em robôs. A aplicabilidade depende da qualidade do mapa, sendo limitada em ambientes ainda não mapeados.

CLOTET; PALACíN (2023) apresentam a biblioteca SLAMICP[2], voltada para otimizar o ICP com detecção de obstáculos integrada. A abordagem reduz em até 36,7% o tempo de processamento em relação à LIBICP, ao eliminar redundâncias na análise da nuvem de pontos. Validado com robô real, o método requer mapa prévio e não realiza otimização global da trajetória, ficando sujeito a *drift*.

YONG et al. (2023) combinam Hector SLAM otimizado com LM e planejamento com A* Dinâmico suavizado, visando aplicações em busca e resgate. A substituição do Gauss-Newton pelo LM melhora a estabilidade e convergência do *scan-matching*. Os testes em simulação com ROS e Gazebo mostraram redução nos erros de pose e melhoria na eficiência da navegação.

HUANG; HUANG (2022) propõem o ASM (*Attention-based Scan Matching*), que prioriza regiões relevantes da cena para melhorar a estimativa de pose com LiDAR 2D. A abordagem foi validada experimentalmente e demonstrou desempenho superior em diferentes cenários. Contudo, depende da seleção adequada de *landmarks* e da calibração dos critérios de troca, sendo sensível a ambientes pouco estruturados.

A Tabela 2.2, destaca pontos relevantes dos trabalhos que abordaram diferentes estratégias. A revisão dos últimos trabalhos (2022-2025) evidenciam contribuições no aprimoramento do *scan-matching* e SLAM baseado em LIDAR 2D e 3D. As pesquisas analisadas propuseram melhorias em eficiência computacional, robustez contra ruído, detecção de *outliers*, redução de *drift* e novas abordagens para otimização da pose. Entretanto, ainda existem desafios não resolvidos, que representam oportunidades para contribuições adicionais.

[2] <http://robotica.udl.cat/slamicp/>

Tabela 2.2: Visão comparativa de trabalhos revisados

Trabalhos	Contribuição	Sensor Principal	Ambiente de Teste	Software Usado	Lacunas do estudo
LEE; LIM; HAN (2025)	Peso adaptativo para lidar com degeneração geométrica no ICP.	LIDAR 3D.	Simulação (com dados reais).	Não mencionado.	Não foi testado em tempo real.
VO; HA (2024)	Rejeição de outliers por Relative Motion Threshold (RMT).	LIDAR 2D.	Simulação (com dados ideais).	MATLAB.	Apenas simulações em MATLAB; não resolve acúmulo de erro (drift).
LI et al. (2024)	Uso de Levenberg-Marquardt e Weighted Signed Distance Function (WSDF).	LIDAR 2D + IMU + odometria.	Protótipo (tempo real).	ROS.	Alto custo computacional.
FILOTHEOU et al. (2023)	Alinhamento sem necessidade de correspondência entre pontos (baseado em Fourier).	LIDAR 2D.	Simulação (com dados reais).	C++ (PCL, Eigen, VGICP).	Depende de um mapa prévio e pode ser sensível a erros no mapa
CLOTET; PALACíN (2023)	Otimização da LIBICP para acelerar a detecção de obstáculos.	LIDAR 2D.	Protótipo (tempo real).	Não mencionado.	Depende de um mapa prévio; não menciona otimização global da trajetória.
YONG et al. (2023)	Melhorias na trajetória do robô com suavização por Bézier.	LIDAR 2D.	Simulação (com dados ideais).	ROS (Gazebo, Ubuntu 18.04).	Apenas testado em simulação.
HUANG; HUANG (2022)	Escolha seletiva de landmarks para minimizar erro acumulado.	LIDAR 2D + odometria.	Protótipo (tempo real).	Provavelmente C++ (Cartographer, PCL).	Dependência da seleção de landmarks adequados.

Fonte: Autor.

2.3 Conclusões

A análise dos trabalhos relacionados evidencia desafios persistentes no uso do ICP para SLAM com LIDAR 2D, como perda de precisão ao longo do tempo, acúmulo de erro (*drift*), dependência de dados pré-coletados e falta de validação em tempo real. Além disso, poucos métodos lidam com o acúmulo excessivo de pontos no mapa global, o que compromete a acurácia do alinhamento em execuções prolongadas.

Diante disso, este trabalho propõe um sistema de *scan-matching* aprimorado, com as principais contribuições:

Execução incremental com referência no scan anterior

O ICP é aplicado de forma incremental, utilizando o *scan* anterior como referência, o que melhora a precisão ao explorar a similaridade natural entre *scans* consecutivos. Isso reduz a sensibilidade a erros acumulados e evita os custos associados à correspondência com mapas densos ao longo do tempo. Essa estratégia favorece a convergência do ICP e permite atualizações contínuas da posição e do mapa, aproximando o sistema de uma arquitetura SLAM funcional.

Execução em tempo real com interface gráfica

A implementação em tempo real permite o processamento contínuo de varreduras de um LIDAR 2D, superando limitações de abordagens baseadas em dados estáticos. Uma interface gráfica, desenvolvida com ferramentas Qt, oferece visualização interativa do alinhamento, da pose estimada e do mapa gerado, facilitando o monitoramento da convergência e a análise do sistema em operação.

O próximo capítulo apresenta os fundamentos do algoritmo ICP, um dos pilares centrais deste trabalho, detalhando sua formulação, estratégias de resolução e as vantagens das métricas de erro no alinhamento.

Capítulo 3

Iterative Closest Point (ICP)

Neste capítulo, é apresentada uma fundamentação teórica sobre o algoritmo ICP, um método amplamente utilizado para o alinhamento de nuvens de pontos. Inicialmente, são discutidos os conceitos básicos do ICP, seguidos pela formulação matemática do problema e as diferentes métricas de erro utilizadas no processo de minimização. Além disso, são abordadas as estratégias de correspondência entre pontos e os critérios de convergência, destacando as diferenças entre as abordagens ponto-a-ponto e ponto-a-plano.

3.1 Introdução

O *Iterative Closest Point* (ICP) é um algoritmo para registrar dois conjuntos de curvas, dois conjuntos de superfície ou duas nuvens de pontos. Esta técnica foi proposta de forma independente por BESL; MCKAY (1992) e ZHANG (1994), em contextos distintos. BESL; MCKAY (1992) desenvolveram o algoritmo ICP para registrar dados parciais de objetos rígidos, obtidos por sensores, com um modelo geométrico ideal, visando a inspeção de formas. Este é um problema de correspondência conjunto-modelo, pois cada ponto sensoreado possui uma correspondência direta no modelo ideal.

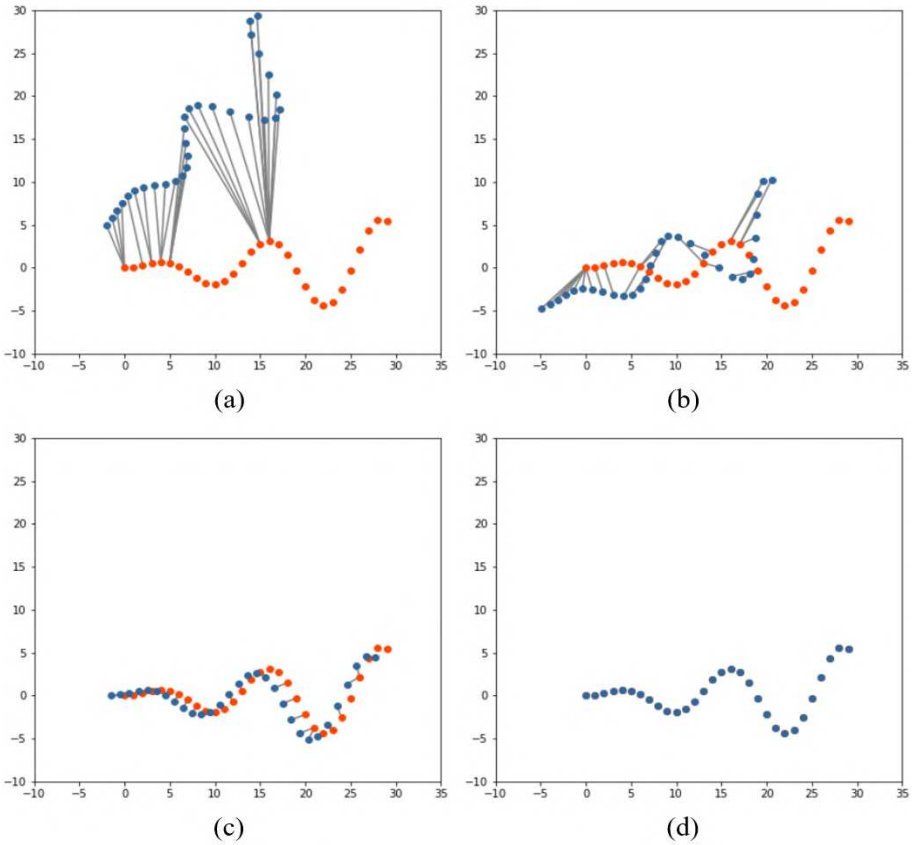
Por outro lado, ZHANG (1994) propôs ICP no contexto da navegação de veículos autônomos em terrenos acidentados utilizando visão computacional. Nesse caso, o algoritmo é aplicado para registrar uma sequência de dados sensoreados, com o objetivo de construir um modelo completo do ambiente e planejar caminhos livres para a navegação. No problema

em questão, trata-se de correspondência conjunto-conjunto, pois uma parte dos dados de um conjunto pode não ter correspondência no outro conjunto.

Para lidar com essas diferenças, ZHANG (1994) incorporou ao ICP um método estatístico baseado na distribuição de distâncias, capaz de tratar *outliers*, oclusões, além do surgimento e desaparecimento de dados. Apesar das diferenças nos contextos e nos detalhes de implementação, ambos os algoritmos compartilham a mesma ideia central: iterativamente associar pontos de um conjunto aos pontos mais próximos no outro conjunto e refinar a transformação entre os dois, buscando minimizar a distância entre as nuvens de pontos (GRAUMAN; LEIBE, 2021).

O funcionamento do algoritmo ICP ao longo de diferentes etapas é mostrado na Figura 3.1, demonstrando o processo de alinhamento de um conjunto de pontos em relação a outro.

Figura 3.1: Etapas do processo de alinhamento utilizando o algoritmo ICP



Fonte: Adaptado de NIUSUS (2021).

No primeiro quadro (a), as linhas conectando os pontos indicam as correspondências iniciais, que frequentemente contêm erros devido ao desalinhamento entre os conjuntos. Com o avanço das iterações, observam-se melhorias progressivas no alinhamento, resultado da aplicação sucessiva das transformações estimadas. No último quadro (d), os dois conjuntos encontram-se devidamente alinhados, evidenciando a convergência do algoritmo.

O método baseia-se na correspondência entre uma nuvem de pontos e uma nuvem de referência, geralmente obtidas por sensores LiDAR 2D ou 3D embarcados em robôs móveis (CLOTET; PALACíN, 2023). O objetivo é estimar uma matriz de transformação que minimize a distância entre os pontos correspondentes, por meio de um processo iterativo que reduz a soma dos erros quadráticos (CLOTET; PALACíN, 2023).

Normalmente, a nuvem coletada representa apenas a vizinhança do robô, enquanto a referência abrange todo o ambiente. Nessas condições, a principal limitação do ICP é o tempo necessário para encontrar, iterativamente, a melhor correspondência entre a nuvem atual e o mapa (CLOTET; PALACíN, 2023).

A seguir, será detalhado o funcionamento do algoritmo ICP, bem como as principais métricas de erro empregadas em seu processo de alinhamento.

3.2 Definição do problema: ICP clássico

Considere dois conjuntos de pontos no espaço Euclidiano $\mathbb{R}^d$ (LEE; LIM; HAN, 2025):

$$P = \{\mathbf{p}_i \in \mathbb{R}^d\}_{i=1}^{N}$$

e

$$Q = \{\mathbf{q}_j \in \mathbb{R}^d\}_{j=1}^{M}$$

onde P é a nuvem de pontos fixa, também chamada de referência, e Q é a nuvem de pontos móvel que será alinhada por meio de uma transformação rígida.

O problema de alinhamento é resolvido iterativamente por meio de duas etapas principais, repetidas até a convergência: associação de correspondências entre os pontos dos dois conjuntos, e estimação da transformação rígida $\mathbf{T}$ que minimiza a distância entre os pares correspondentes (BOGOSLAVSKYI, 2025).

3.2.1 Associação de pontos correspondentes

Para alinhar a nuvem de pontos móvel Q à nuvem fixa P, é necessário identificar para cada ponto $\mathbf{q}_j \in Q$, qual é o ponto mais próximo correspondente na nuvem de referência P. Esse processo é chamado de associação de correspondência.

A métrica utilizada para determinar a correspondência, no caso bidimensional ($\mathbb{R}^2$), é a distância Euclidiana entre dois pontos no plano, definidos como $\mathbf{p}_i = (x_i, y_i)$ e $\mathbf{q}_j = (x_j, y_j)$. Especificamente, a correspondência para um dado ponto $\mathbf{q}_j$ é obtida pela minimização em relação aos pontos da nuvem P, (BESL; MCKAY, 1992):

$$dist(\mathbf{q}_j, P) = \min_{\mathbf{k}\in\{1,..,N\}} \|\mathbf{q}_j - \mathbf{p}_k\|, \tag{3.1}$$

onde $\mathbf{p}_k$ é uma variável de busca que percorre todos os pontos da nuvem fixa P, e $\mathbf{p}_i \in P$ é o ponto que satisfaz $dist(\mathbf{q}_j, \mathbf{p}_i) = dist(\mathbf{q}_j, P)$.

Esse processo de busca define os pares correspondentes $C = \{(i, j)\}$, que são usados na próxima etapa. Após associar os pontos de Q aos pontos correspondentes em P, a transformação $\mathbf{T} = (\mathbf{R}, \mathbf{t})$ é estimada de forma a minimizar a soma dos erros quadráticos pela equação.

3.2.2 Estimação da transformação rígida T

A primeira etapa do alinhamento consiste em calcular os centróides (centros de massa) das nuvens de pontos P e Q com base nas correspondências identificadas. Em seguida, ambas as nuvens são transladadas para que seus centróides coincidam na origem do sistema de coordenadas. Essa centralização facilita a estimativa da rotação ótima entre os conjuntos. Os centróides são dados por:

$$\bar{\mathbf{p}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_i, \quad \bar{\mathbf{q}} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{q}_j. \tag{3.2}$$

e os pontos centralizados são obtidos por:

$$\mathbf{p}'_i = \mathbf{p}_i - \bar{\mathbf{p}}, \quad \mathbf{q}'_j = \mathbf{q}_j - \bar{\mathbf{q}}. \tag{3.3}$$

A segunda etapa determina a rotação ótima entre as nuvens centralizadas por meio da SVD (*Singular Value Decomposition*) para garantir que os conjuntos estejam o mais alinhados possível entre si (BOGOSLAVSKYI, 2025). Caso as nuvens fossem perfeitamente alinhadas, a matriz de covariância cruzada tenderia à identidade. Entretanto, em casos reais, essa matriz é usada como base para aproximar esse alinhamento por meio de transformações iterativas.

Assumindo que as correspondências são conhecidas em uma iteração, define-se o conjunto de pares correspondentes como $C = (i, j) : \mathbf{p}_i \leftrightarrow \mathbf{q}_j$, com $|C| = N$. A matriz de covariância cruzada $\mathbf{W}$ é então dada por:

$$\mathbf{W} = \sum_{i=1}^{N} \mathbf{q}'_j \mathbf{p}'^{\top}_i, \tag{3.4}$$

onde cada termo $\mathbf{q}'_j \mathbf{p}'^{\top}_i$ representa o produto externo entre os vetores $\mathbf{q}'_j$ e $\mathbf{p}'_i$. No caso bidimensional ($\mathbb{R}^2$), a matriz $\mathbf{W}$ assume a forma (os índices i e j foram removidos para simplificar a notação):

$$\mathbf{W} = \begin{bmatrix} cov(p_x, q_x) & cov(p_x, q_y) \\ cov(p_y, q_x) & cov(p_y, q_y) \end{bmatrix}. \tag{3.5}$$

A decomposição SVD é então aplicada à matriz $\mathbf{W}$:

$$\mathbf{SVD}(\mathbf{W}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}, \tag{3.6}$$

onde $\mathbf{U}$ e $\mathbf{V}$ são matrizes ortogonais e $\mathbf{\Sigma}$ é uma matriz diagonal com os valores singulares. A decomposição SVD mostra como rotacionar dados para alinhá-los com sua direção mais proeminente usando $\mathbf{U}\mathbf{V}^{\top}$ e como escalá-los utilizando seus valores singulares $\mathbf{\Sigma}$ (SICILIANO; KHATIBL, 2016). Portanto, A rotação que melhor alinha os conjuntos de pontos é dada por:

$$\mathbf{R} = \mathbf{U}\mathbf{V}^{\top}, \tag{3.7}$$

e a translação correspondente é obtida por:

$$\mathbf{t} = \bar{\mathbf{p}} - \mathbf{R}\bar{\mathbf{q}}. \tag{3.8}$$

Alternativamente, cada iteração do ICP pode ser tratada como um problema de minimização por mínimos quadrados. A função objetivo que deseja-se minimizar é a soma dos quadrados das distâncias entre os pontos correspondentes das duas nuvens (KATAOKA et al., 2021):

$$E(\mathbf{R}, \mathbf{t}) = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{p}_i - (\mathbf{R}\mathbf{q}_j + \mathbf{t})\|^2, \tag{3.9}$$

em que $\mathbf{R} \in SO(d)$ é uma matriz de rotação, $\mathbf{t} \in \mathbb{R}^d$ é o vetor de translação, e $|\cdot|$ representa a norma Euclidiana. Como a rotação introduz não linearidade, a pose também pode ser representada por um vetor de parâmetros mínimos $\mathbf{x} = [x, y, \theta]^\top$, que define a transformação rígida aplicada à nuvem Q para alinhá-la à referência P (BOGOSLAVSKYI, 2025).

3.2.3 Critérios de Convergência

A cada iteração do ICP, os pontos da nuvem móvel são transformados pela rotação $\mathbf{R}^{(k)}$ e translação $\mathbf{t}^{(k)}$, e o alinhamento é avaliado com base no erro médio quadrático:

$$\epsilon^{(k)} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - (\mathbf{R}^{(k)}\mathbf{q}_j + \mathbf{t}^{(k)})\|^2. \tag{3.10}$$

onde k representa a iteração atual. O algoritmo termina quando $\epsilon^{(k)} < \tau$, sendo τ um limiar predefinido, ou quando o número máximo de iterações $k_{\max}$ é atingido.

3.3 ICP baseado no método Gauss Newton

Na prática, existem variantes do ICP que permitem otimizar o algoritmo. Neste caso, requer deixar a abordagem com SVD e partir para uma abordagem geral de mínimos quadrados do problema não linear, que pode ser o método de Gauss-Newton permite incorporar derivadas, o que é chave para obter uma solução iterativa precisa.

Define-se as entradas $\mathbf{q}_j \in Q$ como pontos que devem ser comparados a $\mathbf{p}_i \in P$. Por correspondência, entende-se o processo de encontrar a pose $\mathbf{x} = [x, y, \theta]^\top$, de modo a minimizar a soma dos comprimentos quadrados das correspondências. A pose $\mathbf{x}$ pode ser representada por uma matriz de rotação

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{3.11}$$

e um vetor de translação,

$$\mathbf{t} = [x, y]^\top. \tag{3.12}$$

Essas representações continuarão sendo utilizadas de forma intercalada ao longo do texto e será adotada a seguinte notação $\mathbf{h}_j(\mathbf{x}) = \mathbf{R}\mathbf{q}_j + \mathbf{t}$, para denotar a nuvem de pontos Q transformada com $\mathbf{R}$ e $\mathbf{t}$. Além disso, a função de erro $\mathbf{E}$ é definida como:

$$E(\mathbf{x}) = \sum_{\{i,j\} \in C} \mathbf{e}_{i,j}(\mathbf{x}), \tag{3.13}$$

onde,

$$\mathbf{e}_{i,j}(\mathbf{x}) = \mathbf{p}_i - \mathbf{h}_j(\mathbf{x}) = \mathbf{p}_i - (\mathbf{R}\mathbf{q}_j + \mathbf{t}). \tag{3.14}$$

O método de Gauss-Newton será utilizado para calcular a solução de mínimos quadrados do problema não linear. Para isso, a função é linearizada nas proximidades de $\mathbf{x}$. Resolver mínimos quadrados não lineares é equivalente a resolver o seguinte sistema de equações (NOCEDAL; WRIGHT, 2006):

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{g}, \tag{3.15}$$

onde $\Delta\mathbf{x}$ é o incremento do argumento ($[\Delta x, \Delta y, \Delta\theta]$ neste caso), $\mathbf{H}$ é a matriz Hessiana de $\mathbf{e}_{i,j}$ e $\mathbf{g}$ é a derivada da função $\mathbf{e}_{i,j}$. Calcula-se o gradiente $\mathbf{g}$ da seguinte forma (NOCEDAL; WRIGHT, 2006):

$$\mathbf{g} = \mathbf{J}^\top \mathbf{e}_{\mathbf{i},\mathbf{j}}. \tag{3.16}$$

No método de Gauss-Newton, a função ao redor do ponto considerado é linearizada, o que permite calcular o Hessiano. Os elementos do Hessiano são calculados através da diferenciação dos elementos do gradiente (NOCEDAL; WRIGHT, 2006), ou seja:

$$\mathbf{H} = \mathbf{J}^\top \mathbf{J}. \tag{3.17}$$

Assim, é possível formular o sistema de equações cuja solução resolve o problema. O Hessiano $\mathbf{H}$ e o gradiente $\mathbf{g}$ são inicialmente definidos como zero:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.18}$$

É necessário construir um sistema de equações que, quando resolvido, forneça a pose relativa (BOGOSLAVSKYI, 2025). Para cada par de pontos correspondentes, procede-se do seguinte modo:

$$\mathbf{H} \rightarrow \mathbf{H} + \mathbf{J}^\top \mathbf{J}, \tag{3.19}$$

e

$$\mathbf{g} \rightarrow \mathbf{g} + \mathbf{J}^\top \mathbf{e}_{i,j}. \tag{3.20}$$

Com o sistema de equações formulado, é possível determinar $\Delta\mathbf{x}$, que representa a solução do problema de mínimos quadrados:

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{g} \Longrightarrow \Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}. \tag{3.21}$$

O cálculo do gradiente e do Hessiano requer a Jacobiana $\mathbf{J}$. O método de Gauss-Newton permite definir diferentes métricas de erro, sendo as mais comuns no ICP: a ponto-a-ponto, que minimiza a distância euclidiana entre pares correspondentes, e a ponto-a-plano, que projeta o erro na direção norma à superfície, sendo mais eficaz em regiões planas.

As próximas subseções detalham essas métricas, incluindo a formulação da função de erro e a construção da Jacobiana usada na resolução iterativa do sistema.

3.3.1 ICP com métrica ponto-a-ponto

A função de erro dada na Equação 3.14 pode ser descrita como:

$$\mathbf{e}_{i,j}(\mathbf{x}) = \begin{bmatrix} p_i^x - (q_x \cos\theta - q_y \sin\theta + x) \\ p_i^y - (q_x \sin\theta + q_y \cos\theta + y) \end{bmatrix}. \tag{3.22}$$

Para a Jacobiana, $\mathbf{e}_{i,j}(\mathbf{x})$ é derivado em relação a $[x, y, \theta]$. Lembrando que a derivada da translação $\mathbf{t} = [x, y]^\top$ é:

$$\frac{\partial \mathbf{e}}{\partial x} = [-1, 0]^\top, \quad \frac{\partial \mathbf{e}}{\partial y} = [0, -1]^\top. \tag{3.23}$$

A rotação aplicada ao ponto $\mathbf{q}_j = [q_x, q_y]^\top$ é:

$$\mathbf{R}(\theta)\mathbf{q}_j = \begin{bmatrix} q_x \cos\theta - q_y \sin\theta \\ q_x \sin\theta + q_y \cos\theta \end{bmatrix}. \tag{3.24}$$

Então a derivada de $\mathbf{R}(\theta)\mathbf{q}_j$ em relação a θ é:

$$\frac{d}{d\theta} \begin{bmatrix} q_x \cos\theta - q_y \sin\theta \\ q_x \sin\theta + q_y \cos\theta \end{bmatrix} = \begin{bmatrix} -q_x \sin\theta - q_y \cos\theta \\ q_x \cos\theta - q_y \sin\theta \end{bmatrix}. \tag{3.25}$$

Consequentemente, todos os elementos necessários estão disponíveis para calcular a Jacobiana $\mathbf{J}$ da seguinte maneira:

$$\begin{aligned} \mathbf{J}_{\text{ponto-a-ponto}} &= \frac{\partial \mathbf{e}}{\partial \mathbf{x}} \\ &= \frac{\partial \mathbf{h}_j(\mathbf{x})}{\partial \mathbf{x}} \\ &= \begin{bmatrix} \frac{\partial \mathbf{e}}{\partial x} & \frac{\partial \mathbf{e}}{\partial y} & \frac{\partial \mathbf{e}}{\partial \theta} \end{bmatrix} \\ &= \begin{bmatrix} -1 & 0 & -q_x \sin\theta - q_y \cos\theta \\ 0 & -1 & q_x \cos\theta - q_y \sin\theta. \end{bmatrix} \end{aligned}. \tag{3.26}$$

Mas é comum inverter o sinal para expressar a variação da função $\mathbf{h}$ em vez da função erro, o que resulta em:

$$\mathbf{J}_{\text{ponto-a-ponto}} = \begin{bmatrix} 1 & 0 & -q_x \sin\theta - q_y \cos\theta \\ 0 & 1 & q_x \cos\theta - q_y \sin\theta \end{bmatrix}. \tag{3.27}$$

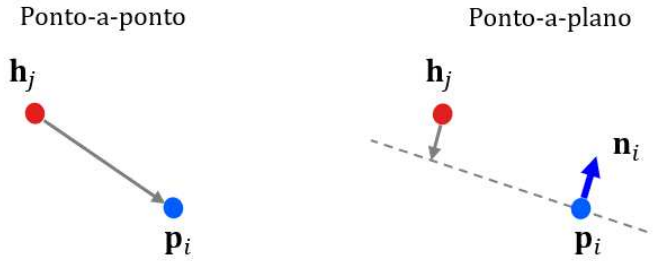
3.3.2 ICP com métrica ponto-a-plano

A métrica ponto-a-ponto utilizada anteriormente pode não ser a mais eficiente. Ela pode exigir mais iterações para que a solução convirja comparada com uma métrica alternativa que funciona melhor, chamada métrica ponto-a-plano.

Para utilizar a métrica ponto-a-plano, é necessário conhecer as normas $\mathbf{n}_i$ dos pontos na nuvem fixa P. Cada norma $\mathbf{n}_i$ representa a orientação do plano local em torno do ponto $\mathbf{p}_i \in P$ e pode ser estimada por ajuste de mínimos quadrados a um pequeno conjunto de vizinhos de $\mathbf{p}_i$ (GUAN; LI; REN, 2018).

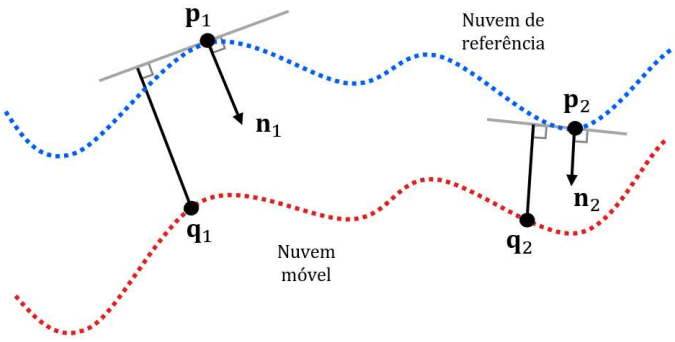
As diferenças entre as métricas são ilustradas na Figura 3.2. Na Figura 3.3, são representados os pontos da nuvem móvel e os planos tangentes à superfície nos respectivos pontos correspondentes do conjunto de destino (YAZDANPOUR; FAN; SHENG, 2017).

Figura 3.2: Diferença da métrica ponto-a-ponto para a métrica ponto-a-plano



Fonte: Autor.

Figura 3.3: Erro ponto-plano entre duas superfícies



Fonte: Autor.

A norma $\mathbf{n}_i$ surge da ideia de que, em regiões planas (como paredes), o erro mais relevante ocorre na direção perpendicular à superfície. Assim, ao alinhar duas nuvens de

pontos, considera-se apenas o componente do erro na direção da norma, resultando em uma convergência mais rápida e precisa.

Formalmente, o erro da métrica ponto-a-plano é minimizado da seguinte forma (YAZDANPOUR; FAN; SHENG, 2017):

$$E(\mathbf{R},\mathbf{t}) = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{n}_i \cdot (\mathbf{p}_i - (\mathbf{R}\mathbf{q}_i + \mathbf{t}))\|^2 . \tag{3.28}$$

Essa formulação projeta o erro apenas na direção norma $\mathbf{n}_i$, ignorando variações menos relevantes ao longo da superfície, o que favorece a estabilidade do alinhamento em ambientes estruturados.

Neste caso, em vez de considerar o vetor de erro completo $\mathbf{e}$, utiliza-se apenas a projeção desse erro na direção da norma $\mathbf{n}_i = [n_x, n_y]^\top$.

O erro escalar, correspondente à projeção do erro sobre a norma, é definido por:

$$e_{i,j}(\mathbf{x}) = \mathbf{n}_i^\top (\mathbf{p}_i - \mathbf{h}_j(\mathbf{x})). \tag{3.29}$$

Como o erro é escalar, a Jacobiana correspondente assume a forma de uma matriz 1×3:

$$\mathbf{J}_{\text{ponto-a-plano}} = \frac{\partial e_{i,j}(\mathbf{x})}{\partial \mathbf{x}} = -\mathbf{n}_i^\top \cdot \frac{\partial \mathbf{h}_j(\mathbf{x})}{\partial \mathbf{x}}. \tag{3.30}$$

Utilizando a expressão da Jacobiana de $\mathbf{h}_j(\mathbf{x})$ já apresentada na Equação 3.27, obtém-se:

$$\mathbf{J}_{\text{ponto-a-plano}} = \begin{bmatrix} n_x & n_y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -q_x \sin\theta - q_y \cos\theta \\ 0 & 1 & q_x \cos\theta - q_y \sin\theta \end{bmatrix} . \tag{3.31}$$

A multiplicação linha por coluna resulta na Jacobiana associada à métrica ponto-a-plano:

$$\mathbf{J}_{\text{ponto-a-plano}} = \begin{bmatrix} n_x & n_y & n_x(-q_x \sin\theta - q_y \cos\theta) + n_y(q_x \cos\theta - q_y \sin\theta) \end{bmatrix} . \tag{3.32}$$

3.4 Conclusões

Na métrica ponto-a-ponto, o vetor de erro é definido na Equação 3.14 Esse vetor mede a distância total entre os pares de pontos, ou seja, em todas as direções do espaço. O algoritmo busca minimizar a soma das distâncias euclidianas entre os pares correspondentes, independentemente da direção do erro.

Contudo, em muitos casos, a direção do erro é mais relevante do que sua magnitude total. Em superfícies planas, por exemplo, deslocamentos ao longo da superfície (tangenciais) não afetam significativamente o alinhamento, enquanto deslocamentos perpendiculares (normas) são críticos.

Geometricamente, considere um ponto $\mathbf{q}_j$ que deve ser alinhado a uma parede plana representada por diversos pontos $\mathbf{p}_i$. Mesmo que $\mathbf{q}_j$ esteja corretamente posicionado ao longo da parede, um pequeno desvio perpendicular resultará em um erro significativo na métrica ponto-a-ponto, pois todo deslocamento, inclusive o tangencial, contribui para o erro.

Na métrica ponto-a-plano, resolve-se essa limitação projetando o erro na direção da norma $\mathbf{n}_i$ associada à superfície, como estabelecido na Equação 3.29. Isso significa que apenas o componente do erro ortogonal à superfície é considerado.

Dessa forma, utilizar $\mathbf{n}_i$ implica assumir que apenas o afastamento em relação à superfície importa, enquanto deslocamentos ao longo dela são irrelevantes para o erro de alinhamento. É essa priorização que torna a métrica ponto-a-plano mais eficiente em ambientes estruturados.

Essa modificação leva a uma Jacobiana ajustada, reduzindo a dimensionalidade do problema. Enquanto a Jacobiana do método ponto-a-ponto tem dimensões 2×3, a Jacobiana ponto-a-plano tem dimensões 1×3, resultando na estabilidade numérica da solução e em uma convergência mais rápida e robusta em comparação com o método ponto-a-ponto.

Capítulo 4

Sensor LiDAR

Os sensores de distância a *laser* são denominados na literatura técnica, como *Laser Scanner* e *Light Detection And Ranging* (LiDAR). Estes sensores são usados para medir a distância até um objeto usando um *laser* como fonte. Eles possuem vantagens de alto desempenho, alta velocidade, aquisição de dados em tempo real, existindo uma variedade de aplicações em relação à medição de distância (PYO et al., 2017). Esse tipo de sensor usa *lasers* para criar imagens unidimensionais, bidimensionais ou tridimensionais, seja ele móvel ou estacionário (MENDO, 2022).

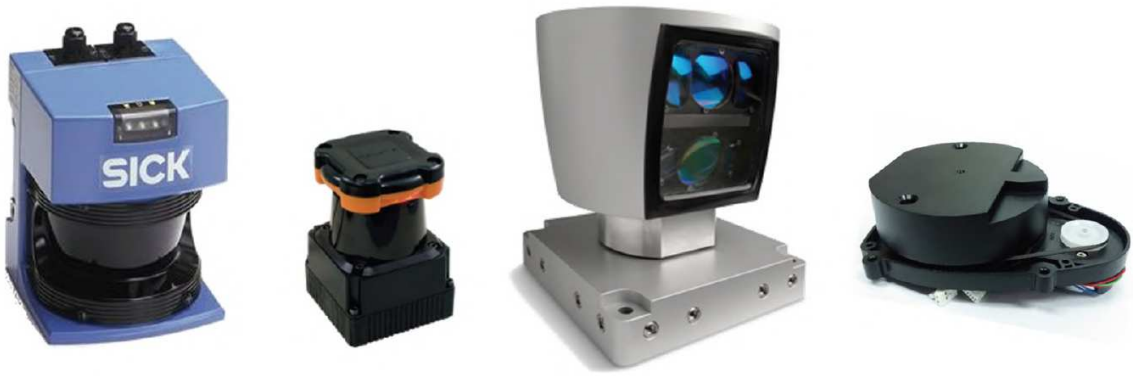
Essa tecnologia é aplicada em vários campos para tarefas de mapeamento, como: agricultura, para exploração e estudo da estrutura do solo; topografia, para criação de fotografias de certas áreas da superfície terrestre; meteorologia para medição e monitoramento de fenômenos atmosféricos; ambiente automotivo, destinado a veículos que possuem o modo piloto automático, o sensor ajuda a detectar a natureza e a proximidade dos objetos que cercam o veículo.

Este capítulo abordará sensores LiDAR que são amplamente utilizado no campo da robótica para reconhecimento de objetos e na direção autônoma em veículos não tripulados baseado em SLAM.

4.1 LiDARs utilizados para navegação autônoma

Existem diversos tipos de LiDAR no mercado, destinados tanto para ambientes externos como internos. Na Figura 4.1, pode ser observado alguns modelos de LiDARs, da esquerda para direita: SICK LMS 210, Hokuyo UTM-30LX, Velodyne HDL-64e e RPLIDAR A1. Os preços variam de produto para produto. O RPLIDAR chega ao mercado a um preço baixo comparado a outros, custando atualmente cerca de R$ 800,00.

Figura 4.1: Típicos LiDARs utilizados para direção autônoma



Fonte: PYO et al. (2017).

4.2 Princípio de medição do LiDAR

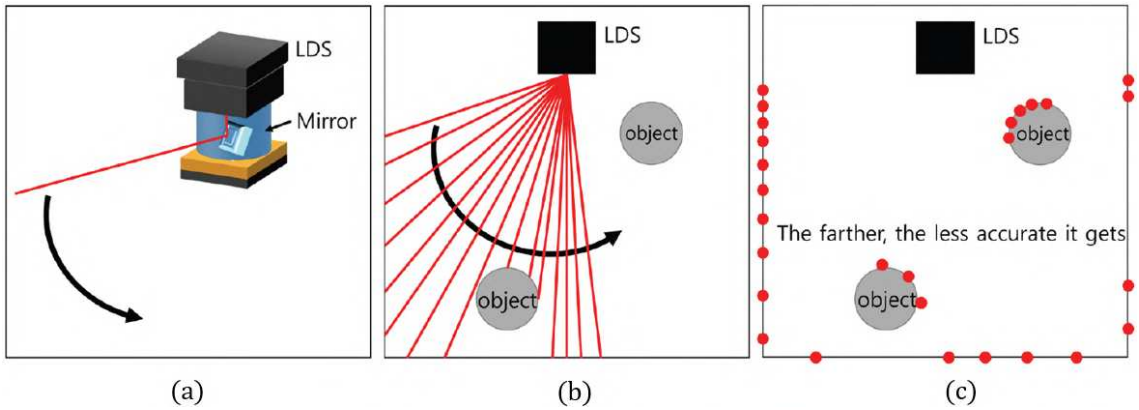
Um LiDAR básico em 1D possui três componentes básicos: um laser, um fotodetector e um cronômetro preciso (MENDO, 2022). O *laser* emite primeiro um pulso, no mesmo momento em que o cronômetro começa a contar. Em seguida, o pulso, à velocidade da luz, atinge o alvo, no caso de carros autônomos por exemplo, um veículo ou um objeto estacionário. Se a superfície não for muito polida ou brilhante, o pulso se dispersará em todas as direções e parte da luz refletida retornará na direção de envio. Dessa forma, a diferença de tempo do pulso de luz desde a emissão até o alvo e o retorno ao receptor a partir do alvo, calculada pelo produto do número de pulsos do relógio n do contador e o intervalo de pulsos τ, o que resulta em:

$$R = \frac{1}{2}cn\tau = \frac{c}{2f}n = ln, \tag{4.1}$$

onde R é a distância até o alvo, c é a velocidade da luz, n é o número de pulsos do relógio, τ é o intervalo de tempo entre os pulsos do relógio, f é a frequência dos pulsos do relógio, l é a base de distância representada por cada pulso do relógio. Essa técnica é chamada de medição de tempo de voo (ToF, do inglês *Time of Flight*) (LIU et al., 2018).

Observa-se na imagem da Figura 4.2 (a) um sensor LiDAR com um *laser* interno e um espelho inclinado em um ângulo. Um motor faz girar o espelho, enquanto o sensor mede o ToF do *laser*. Assim, o espelho rotativo pode ser usado para fazer um escaneamento 2D, em um plano horizontal ao redor do sensor (b) (PYO et al., 2017). Se for adicionado um movimento de inclinação para cima e para baixo, pode-se criar um escaneamento 3D. A coleta de pontos é chamada de "nuvem de pontos", conforme apresentado (c), e é usada para estimativa de estado. Os sensores LiDAR 2D relatam alcance e ângulo de azimute.

Figura 4.2: Medição de distância usando LiDAR 2D



Fonte: PYO et al. (2017).

Como o LiDAR é uma tecnologia que utiliza luz, pode ser usado no escuro. A precisão diminui à medida que a distância aumenta. Normalmente, o escaneamento ocorre em um ângulo de 180 à 360 graus, dependendo do produto (PYO et al., 2017).

4.3 Limitações de uso de sensores LiDAR

Como o sensor é utilizado como fonte de luz, um feixe de *laser* forte pode causar danos aos olhos. Os produtos são classificados com base na fonte de *laser*. Segundo PYO et al. (2017), em geral, o *laser* é classificado de classe 1 a classe 4, sendo que quanto maior o número,

mais perigos pode trazer. A Classe 1, é um produto seguro e não apresenta problema com contato direto nos olhos. A Classe 2, aumenta o risco de exposição prolongada.

Como o LiDAR mede o retorno da fonte de *laser*, é inútil se nada for refletido. Em outras palavras, vidros transparentes, garrafas plásticas e copos de vidro tendem a refletir ou dispersar a fonte de *laser* em muitas direções (PYO et al., 2017). Para espelhos, a luz é refletida de volta ao espelho, tornando a medição imprecisa.

4.4 RPLIDAR A1

O RPLIDAR A1M8 (Figura 4.3) é uma solução de *scanner* a *laser* 2D de 360 graus. Ele pode ser usado para a navegação de um robô móvel e a partir dos dados de "nuvem de pontos" 2D produzidos, podem ser usados em mapeamento, localização e modelagem de objetos ou ambientes. É uma opção de sensor de custo reduzido comparado a outros semelhantes no mercado.

Figura 4.3: Sensor RPLIDAR A1M8

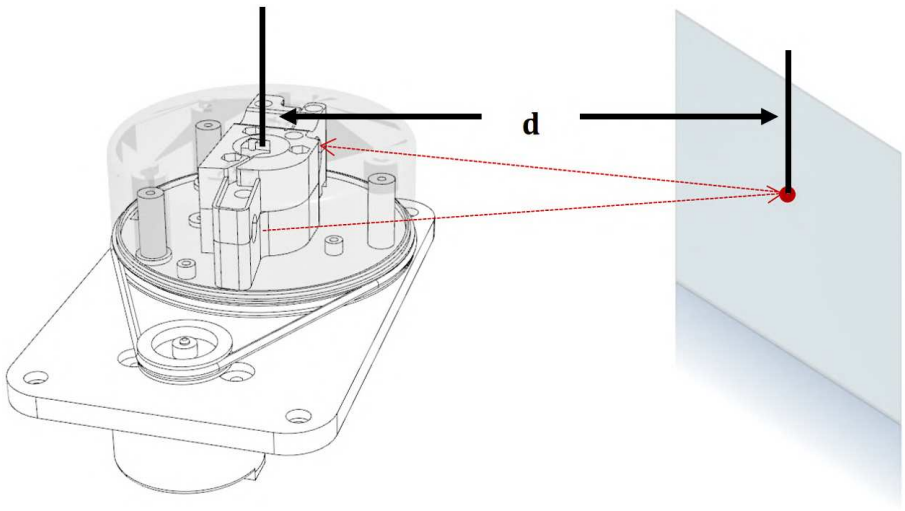
Fonte: Autor.

O sistema físico deste sensor consiste em um *scanner* de alcance e um sistema de moto. Depois de cada subsistema ligado, o *scanner* de alcance rotaciona constantemente no sentido horário a partir do motor conectado a correia, e assim, o usuário pode obter dados de varredura de alcance através da interface de comunicação (UART/USB). O sensor RPLIDAR

A1 conta com um sistema adaptativo e de detecção de velocidade de giro para ajustar a frequência do *scanner* a *laser* automaticamente, de acordo com a velocidade do motor (ASLAM et al., 2020). Logo, o sistema *host* pode obter a velocidade real de giro do RPLIDAR A1 por meio da interface de comunicação.

O RPLIDAR A1 é basicamente um sistema de medição por triangulação a *laser* (SLAMTEC, 2020). Como pode ser observado na Figura 4.4, é emitido um sinal de *laser* infravermelho modulado, e então estes pulsos de luz são refletidos nos objetos ao redor do objeto e retornam ao sensor (BASAVANNA et al., 2021). Dessa maneira, o DSP (*Digital Signal Processor*) embutido no RPLIDAR A1 começa a processar os dados de amostra e tem como saída o valor de distância (mm) e valor de ângulo entre o objeto.

Figura 4.4: Sistema de triangulação a laser do RPLIDAR A1



Fonte: Adaptado de SLAMTEC (2020).

Os fabricantes oferecem o kit de desenvolvimento de *software* (SDK, *Software Development Kit*) para sistemas Windows, Linux e outros. Disponibiliza o *software* RoboStudio para testes e depuração, fornecendo uma interface gráfica de depuração *Framegrabber* para o RPLIDAR.

O RPLIDAR A1 pertence à Classe I em conformidade aos padrões de segurança a *laser*, estabelecidos pela norma internacional IEC 60825-1, utiliza um *laser* infravermelho de baixa potência (< 5 mW), emitindo em um período de tempo muito curto (SLAMTEC, 2020). Isso faz com que funcione adequadamente em diversos tipos de ambientes internos e externos

sem exposição direta à luz solar.

Na Tabela 4.1, são apresentadas descrições mais detalhadas do RPLIDAR A1 segundo os fabricantes.

Tabela 4.1: Especificações do sensor RPLIDAR

Especificações	
Modelo	A1M8
Classe	1
Escaneamento	360°
Interface de comunicação	UART/USB
Tensão de operação	5 V
Corrente de operação	100 mA
Temperatura de operação	20 a 40℃
Frequência de varredura	5,5 Hz
Frequência máxima	10 Hz
Amostra de pontos	1450
Alcance	0,15 até 12 m
Dimensões	96,8 x 70,3 x 55 mm
Peso	170 g

Fonte: SLAMTEC (2020).

4.5 Vantagens, desvantagens e restrições

O sensor apresentado tem como principais vantagens:

- Baixo custo, sendo relativamente acessível em comparação com outros sensores LiDAR no mercado;
- Ampla faixa de detecção de até 12 metros;
- Frequência de varredura com uma taxa suficientemente alta para muitas aplicações em tempo real;
- Compacto e leve.

Já as principais desvantagens e restrições:

- Baixa resolução angular, limitando a detecção de objetos pequenos ou detalhes finos em distâncias maiores;
- Menos precisão em condições de luz intensa ou exposição direta à luz solar;
- Interface de comunicação limitada a porta serial/USB;
- Restrição de uso em ambientes externos.

4.6 Conclusões

Neste capítulo, foi apresentado o princípio de funcionamento dos sensores LiDAR, com ênfase no RPLIDAR A1, destacando seu funcionamento, aplicações e limitações. Por sua precisão, baixo custo e facilidade de integração, o RPLIDAR A1 é uma opção viável para projetos de navegação autônoma baseados em SLAM. A compreensão dessas características é essencial para o uso eficiente da tecnologia nos sistemas.

O próximo capítulo aborda a aplicação prática desses sensores em sistemas de mapeamento e localização, demonstrando como as informações adquiridas podem ser processadas e interpretadas para auxiliar na construção de mapas e na estimativa da pose do robô.

Capítulo 5

Resultados e Discussões

Este capítulo apresenta a metodologia adotada no desenvolvimento do sistema de mapeamento e localização com sensor LiDAR 2D, detalhando as etapas, decisões de projeto e estratégias utilizadas. São descritos os componentes, a estrutura do sistema e a implementação do algoritmo de *scan-matching*.

Em seguida, são apresentados os resultados obtidos, abrangendo: testes iniciais do sensor LiDAR 2D utilizando o ROS; aquisição dos dados do sensor e integração com a interface gráfica; avaliação das abordagens ICP; mapeamento e localização *offline* com dados previamente capturados em ambiente controlado; e, por fim, a implementação do sistema de mapeamento e localização em tempo real ao veículo AGV.

5.1 Descrição geral da implementação experimental

A metodologia adotada neste trabalho foi estruturada visando o desenvolvimento de um sistema eficiente de *scan-matching* baseado no método ICP com otimização por Gauss-Newton, aplicado a um sensor LiDAR 2D. A abordagem proposta busca aprimorar a precisão do alinhamento das varreduras por meio de uma estratégia incremental, na qual cada scan é alinhado diretamente ao *scan* anterior, em vez de utilizar um mapa global acumulativo. Essa escolha reduz a propagação de erro ao longo do tempo e garante que cada alinhamento seja realizado entre varreduras naturalmente semelhantes, melhorando a confiabilidade do processo.

Para atingir os objetivos estabelecidos, a pesquisa foi conduzida seguindo uma sequência estruturada de etapas:

1. Análise preliminar e validação do sensor LiDAR 2D: Utilizou-se pacotes do ROS e do Hector SLAM para observar o comportamento esperado do sensor e validar o conceito, fornecendo uma base para o desenvolvimento da abordagem proposta.

2. Aquisição e integração dos dados do LiDAR: Inicialmente, um leitor de varreduras foi implementado em Windows Forms, mas incompatibilidades com o CLR (*Common Language Runtime*) exigiram a migração para Qt, permitindo o uso eficiente de C++ nativo. Foi criada uma biblioteca intermediária para encapsular as funções essenciais do sensor e garantir a integração estável.

3. Desenvolvimento do ICP: Foram implementadas três versões do ICP (clássico ponto-a-ponto, Gauss-Newton ponto-a-ponto e ponto-a-plano) para comparação de desempenho com dados coletados em posições conhecidas.

4. Mapeamento e localização incremental: Desenvolveu-se um sistema para estimativa da pose do robô e mapeamento contínuo com base em sequências de varreduras, utilizando o ICP Gauss-Newton ponto-a-plano em ambiente controlado para testes do sistema incremental.

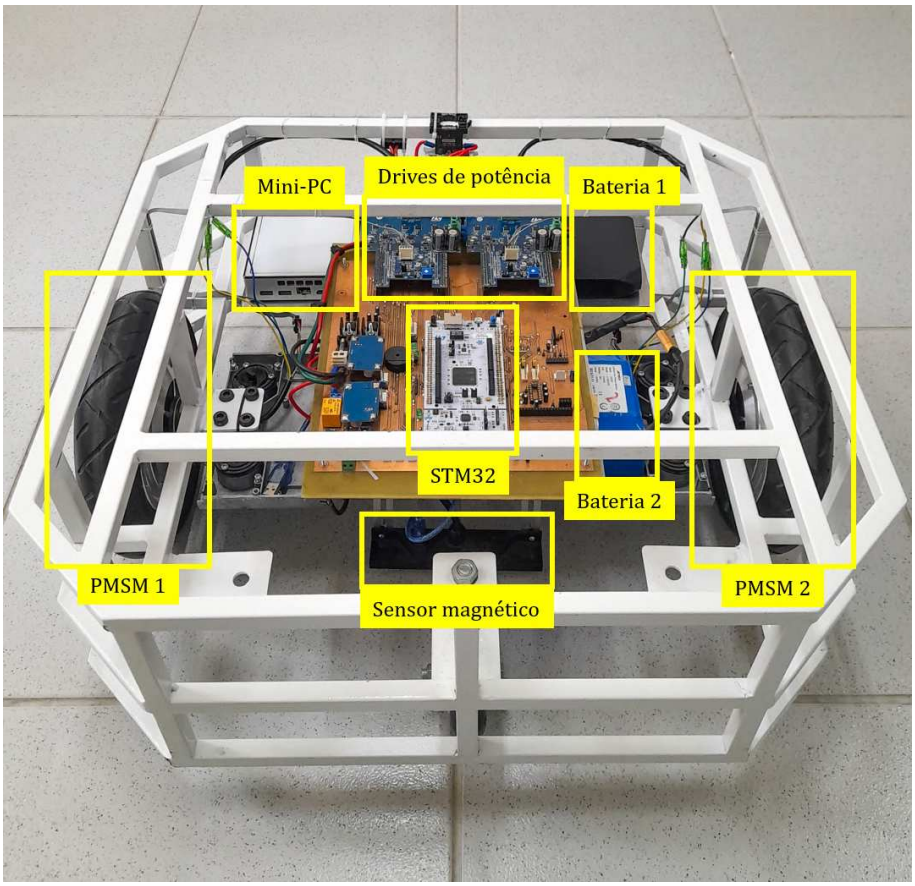
5. Filtragem e suavização do mapa: Para otimizar o mapa global, aplicou-se um filtro baseado em densidade de vizinhos e uma distância mínima entre pontos, removendo *outliers* e redundâncias.

6. Implementação em tempo real: O sistema foi integrado a um veículo AGV do LIEC, processando dados em tempo real e validando o desempenho da abordagem sob condições reais.

Para o estudo e desenvolvimento desta pesquisa, foi utilizado um desktop equipado com um processador Intel® Core™ i7-13700 e 16 GB de RAM. Esse ambiente computacional foi empregado para a implementação, testes de desempenho do algoritmo e desenvolvimento da interface gráfica.

Para a validação do sistema desenvolvido, foi utilizada a plataforma robótica experimental do LIEC, onde o algoritmo foi testado em um ambiente real com dados adquiridos diretamente do sensor LIDAR 2D. A plataforma é composta por diversos componentes essenciais do sistema de controle e navegação do robô, conforme ilustrado na Figura 5.1.

Figura 5.1: Principais componentes da plataforma



Fonte: Autor.

O principais componentes da plataforma experimental, são:

- Mini-PC: microcomputador equipado com processador Intel® Core™ i5-5200U e 8 GB de memória RAM;
- Microcontrolador ARM 32-bits cortex M7 modelo STM32F767ZIT6U utilizado por meio da placa de desenvolvimento STM32 Nucleo-F767ZI;
- Driver de potência da STM32 X-Nucleo-IHMO8M1 para motores com 3 fases PMSM com tensão nominal de 10 V à 48 V;

- Sensor magnético MGS1600GY da RoboteQ;
- Motores PMSM (*Permanent Magnet Synchronous Motor*);
- Banco de baterias com inversor Omni 20: com uma saída 120 V 60 Hz 100 W e duas portas USB 5 V 3 A, possui capacidade de 20.400 mAh;
- Banco de baterias *lithium-ion* 10S2P: com uma saída 36 V 158 Wh, possui capacidade de 4400 mAh.

Para integrar o sensor LiDAR 2D à plataforma robótica, uma estrutura metálica foi projetada para ser fixada no robô. Essa estrutura atua como suporte para o sensor, garantindo que o sensor fique posicionado de forma centralizada. A configuração final do suporte pode ser observada na Figura 5.2.

Figura 5.2: Estrutura metálica com sensor para alocar na plataforma

Fonte: Autor.

A implementação foi realizada utilizando C++ devido ao seu desempenho computacional

superior, sendo desenvolvida no Visual Studio 2022, um ambiente *open source*. O projeto utiliza Eigen para operações de álgebra linear e Qt versão estudante para visualização gráfica dos resultados. A escolha dessas ferramentas permite um sistema independente de bibliotecas externas como ROS, garantindo maior portabilidade e facilidade de configuração. Além disso, optou-se por não utilizar MATLAB, pois, apesar de sua ampla aplicação acadêmica, ao executar o software ele apresenta já alto processamento computacional e exige licença paga, o que reduziria a acessibilidade e a flexibilidade do sistema.

5.1.1 Estratégia de avaliação

Para comparar o desempenho dos algoritmos ICP Clássico e ICP baseado em Gauss-Newton, utilizando as métricas ponto-a-ponto e ponto-a-plano e validar a melhor abordagem para o sistema, os experimentos foram conduzidos em um ambiente controlado, dentro de uma sala de laboratório. Duas nuvens de pontos foram capturadas com o sensor em posições distintas, permitindo a aplicação dos diferentes métodos de alinhamento sob as mesmas condições. Antes da aquisição dos dados, realizou-se a medição manual das coordenadas da pose real, posição (x, y) e orientação θ, a fim de fornecer uma base confiável para comparação dos resultados.

Para a avaliação foi considerado os seguintes critérios: precisão na estimativa da pose, erro médio de alinhamento, número de iterações, tempo de execução, tempo gasto por CPU e uso de CPU.

Assume-se que, uma vez identificada a abordagem mais adequada e plausível por meio da análise comparativa, ela poderá ser integrada diretamente ao sistema incremental de SLAM, dispensando a necessidade da avaliação do deslocamento real da pose em cada experimento.

5.1.2 Programação do algoritmo scan-matching

O ICP Clássico é um algoritmo amplamente utilizado para alinhamento de nuvens de pontos. O método segue uma abordagem ponto-a-ponto, onde cada ponto do conjunto móvel (Dataset2) é associado ao ponto mais próximo no conjunto fixo (Dataset1), e a transformação necessária para minimizar a diferença entre os conjuntos é iterativamente refinada. O Algoritmo 1 apresenta uma versão resumida da implementação desenvolvida neste trabalho,

construída com base nos princípios descritos na literatura.

Algorithm 1: ICP Clássico (Fonte: Autor)

Entradas: Dataset1: Nuvem de pontos fixa; Dataset2: Nuvem de pontos móvel; ϵ: Tolerância; $maxIter$: Número máximo de iterações.
Inicialização: Carregar os datasets, balancear os tamanhos, definir $currentIteration \leftarrow 0$, $converged \leftarrow false$, $robotPose \leftarrow (0, 0, 0)$.
while $currentIteration < maxIter$ ***e não*** $converged$ **do**
    Calcular os centróides $\bar{p}$, $\bar{q}$ e a matriz de correlação W.
    Obter a rotação R e a translação t usando SVD.
    Aplicar a transformação ao Dataset2.
    Calcular o erro de alinhamento: $E \leftarrow \frac{1}{N} \sum \text{dist}(p_i, q_i)$.
    if $E \leq \epsilon$ **then**
        $converged \leftarrow true$. **Encerrar.**
    end
    Atualizar $robotPose$ usando R, t e incrementar $currentIteration$.
end
Saída: Exibir o Dataset2 alinhado, a pose do robô e o gráfico do erro.

Observa-se que centróides das nuvens de pontos $\bar{p}$, $\bar{q}$ são calculados, e a matriz de correlação W (Equação 3.5) é construída considerando a relação entre os pontos correspondentes nos conjuntos fixo e móvel. A matriz de transformação T é determinada a partir da SVD (Equação 3.6). O resultado fornece a melhor matriz de rotação R e o vetor de translação t, que são aplicados à nuvem móvel para alinhar os pontos com a referência. A convergência é avaliada pelo erro médio de alinhamento, definido como E (Equação 3.9). Se o erro for inferior a um limite predefinido ϵ ou não houver melhora significativa entre iterações consecutivas, o processo é interrompido.

A cada iteração, a transformação calculada é incorporada à estimativa da pose do robô (x, y, θ), de modo que, ao término do alinhamento, seja possível determinar o deslocamento total acumulado.

O ICP Gauss-Newton é uma variação do ICP Clássico que busca minimizar o erro de alinhamento por meio de um método de otimização não-linear. Diferente da abordagem

baseada em SVD, esta versão do ICP refina iterativamente a transformação por meio do método de Gauss-Newton. Foram analisadas duas métricas de erro com esse método, ponto-a-ponto (Equação 3.27), que minimiza a distância euclidiana entre pares correspondentes, e ponto-a-plano (Equação 3.32), que minimiza a projeção do erro ao longo das normais da nuvem fixa. O Algoritmo 2 apresenta uma versão resumida da implementação desenvolvida neste trabalho, com base nos conceitos da literatura.

Algorithm 2: ICP com Otimização de Gauss-Newton (Fonte: Autor)

Entradas: Dataset1: Nuvem de pontos fixa; Dataset2: Nuvem de pontos móvel; ϵ: Tolerância de convergência; $maxIter$: Máximo de iterações do ICP.
Inicialização: Carregar os datasets, balancear os tamanhos, definir $currentIteration \leftarrow 0$, $converged \leftarrow false$, $robotPose \leftarrow (0, 0, 0)$.
while $currentIteration < maxIter$ ***e não*** $converged$ **do**
    Definir transformação inicial $T \leftarrow I$.
    Encontrar o ponto mais próximo em Dataset1 para cada ponto em Dataset2.
    Obter a norma associada ao ponto correspondente em Dataset1.
    Calcular o erro projetado na direção da norma.
    Construir a Jacobiana J com base nas normas e nas coordenadas dos pontos transformados.
    Resolver $\Delta T \leftarrow (J^T J)^{-1} J^T r$.
    Atualizar $T \leftarrow \Delta T \cdot T$.
    if $\|\Delta T\| < \epsilon$ **then**
        $converged \leftarrow true$. **Encerrar**.
    end
    Aplicar T ao Dataset2, calcular erro de alinhamento $E \leftarrow \frac{1}{N} \sum \text{dist}_{norma}(p_i, q_i)$, atualizar $robotPose$ com T, incrementar $currentIteration$.
end
Saída: Exibir Dataset2 alinhado, pose do robô (x, y, θ) e gráfico do erro ao longo das iterações.

A matriz de transformação inicial T é definida como a matriz identidade I, assumindo, inicialmente, que não há deslocamento ou rotação entre os conjuntos de dados. Em cada

iteração do ICP, o método de Gauss-Newton é aplicado para refinar a transformação T. Esse processo ocorre em três etapas.

A primeira etapa é o cálculo das normas e correspondências. As normas dos pontos da nuvem fixa são calculadas a partir de pontos vizinhos locais. Para cada ponto da nuvem móvel transformada, encontra-se o ponto correspondente na nuvem fixa e sua norma associada.

A segunda etapa é a construção do sistema linear. A matriz Jacobiana J é composta pelas derivadas parciais da função de erro em relação à pose. Sua forma depende da métrica adotada: na métrica ponto-a-ponto, utiliza-se a Equação 3.27; na ponto-a-plano, aplica-se a Equação 3.32, que considera apenas a projeção escalar do erro na direção da norma da superfície. O vetor de resíduos r também é calculado conforme essa projeção.

A terceira etapa é a atualização da transformação. A correção incremental ΔT é obtida resolvendo o sistema (Equação 3.21). A matriz T é atualizada aplicando ΔT, refinando a transformação entre as nuvens. O processo se repete até atingir o número máximo de iterações do Gauss-Newton ou até que a variação da transformação seja inferior ao limiar ϵ, indicando convergência. Ao final do processo, a pose do robô (x, y, θ) é obtida considerando a transformação final estimada.

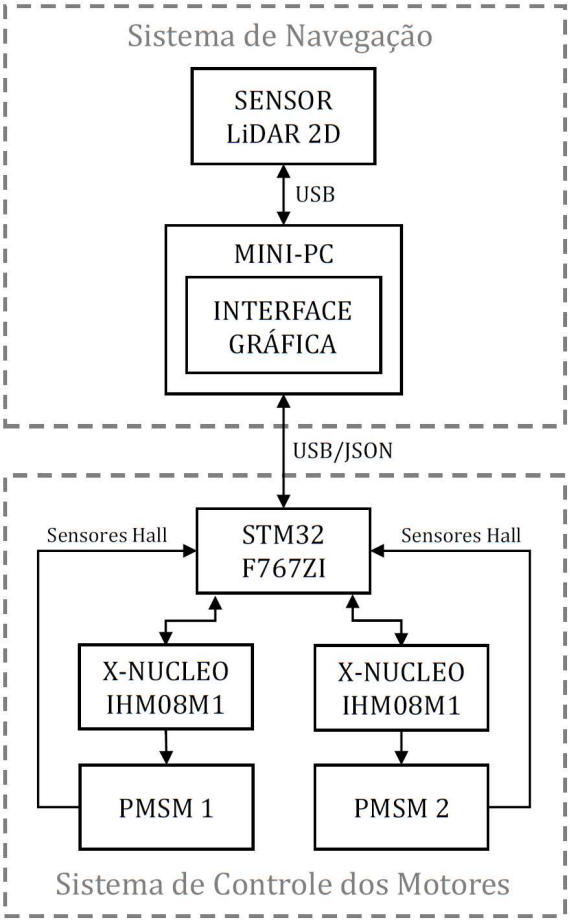
5.1.3 Arquitetura do sistema

Observa-se na Figura 5.3, o esquema básico de interligação dos componentes da plataforma experimental para este trabalho. O mini-PC atua como unidade central de processamento, sendo responsável pelo gerenciamento e controle de toda a plataforma.

O sensor LiDAR 2D opera como principal fonte para percepção do ambiente, sendo responsável pela aquisição de varreduras. Ele recebe comandos do mini-PC para iniciar ou interromper as medições e, em seguida, transmite os dados de scan para processamento.

O controle dos motores PMSM é realizado por meio do microcontrolador STM32, que recebe comandos do mini-PC e é responsável por gerar sinais para os módulos de potência (X-NUCLEO), acionando os motores. Além disso, o STM32 recebe *feedback* em tempo real através dos sensores de efeito Hall embutidos nos motores, permitindo a monitoração da movimentação do robô.

Figura 5.3: Sistemas da plataforma robótica experimental



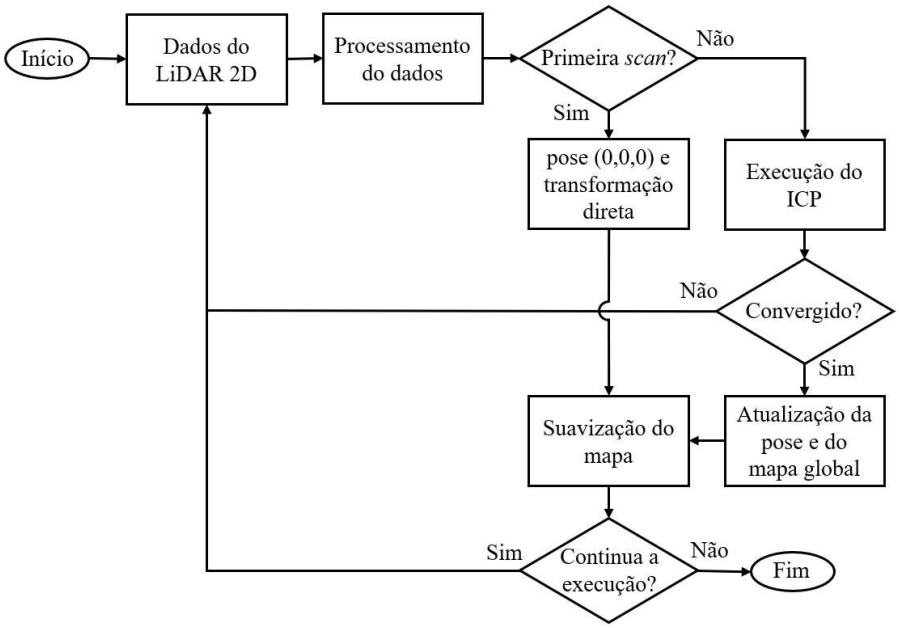
Fonte: Autor.

A comunicação entre o mini-PC e o STM32 ocorre por meio de conexão USB utilizando o protocolo JSON (*Java Script Object Notation*), através do qual o STM32 recebe os parâmetros de velocidade estabelecidos pelo sistema de navegação. Em contrapartida, o microcontrolador também envia dados de odometria ao mini-PC, porém, devido à baixa confiabilidade dos sensores de efeito Hall, essas informações não serão utilizadas neste trabalho. Destacando o proposito do trabalho que é desenvolver um sistema para uso exclusivo do LiDAR 2D, solucionando e sendo alternativa para robôs de baixo custo (que utilizam apenas de sensor LiDAR 2D).

Na Figura 5.4, é possível visualizar o fluxograma simplificado do algoritmo utilizado no sistema de mapeamento e localização com LIDAR 2D baseado no ICP otimizado por Gauss-Newton. O fluxograma descreve a sequência de etapas seguidas para processar os

dados do sensor, realizar o alinhamento e atualização do mapa global.

Figura 5.4: Estrutura simplificada do algoritmo



Fonte: Autor.

O sistema inicia com a leitura e pré-processamento dos pontos coletados pelo sensor LIDAR 2D, garantindo que a nuvem de pontos esteja preparada para o processamento subsequente. Após a aquisição, os dados são filtrados com base nas informações de qualidade proveniente do próprio sensor sobre cada ponto para eliminar medições de baixa confiabilidade.

O primeiro *scan* capturado é diretamente incorporado ao mapa global e, simultaneamente, armazenado como referência para os alinhamentos subsequentes. Na sequência, uma nova nuvem de pontos é adquirida e submetida ao mesmo processo de filtragem baseado no critério de qualidade. Para garantir maior precisão no alinhamento ambas as nuvens são balanceadas para que tenham o mesmo número de pontos e evitar discrepâncias estruturais no alinhamento. Caso haja diferenças no tamanho das nuvens, são preenchidas com base no primeiro ponto lido de maior qualidade. Esse procedimento assegura que o algoritmo ICP possa ser executado de forma eficiente, estimando a transformação para alinhar o *scan* atual (conjunto móvel) ao *scan* de referência (conjunto fixo).

Com esses requisitos é possível seguir para a próxima etapa que é iniciar o processo que

executa o algoritmo ICP para o alinhamento das nuvem móvel com a fixa, realizando os cálculos a cada iteração para determinar a transformação, ou seja, o quanto a nuvem móvel se deslocou (x, y, θ). Se o erro de alinhamento não atender à tolerância estabelecida e o número máximo de iterações for atingido, o processo é reiniciado para a captura um novo *scan*, e assim, o scan anterior é mantido para ser processado com outra nova varredura. A convergência é alcançada a quando a variação do erro entre uma iteração e outra atinge uma tolerância especificada e corresponde ao menor erro registrado ao longo das iterações.

Em seguida, a pose do robô é atualizada com base no deslocamento medido durante o alinhamento entre as nuvens de pontos. Os novos pontos são então transformados para o sistema de coordenadas do mapa global, utilizando informações da pose acumulada, de modo que se sobreponham aos pontos já existentes no mapa. Para evitar o acúmulo excessivo de dados, aplica-se um processo de suavização, que inclui a filtragem de pontos redundantes e a remoção de *outliers*, preservando apenas as informações mais relevantes. Esse procedimento assegura que o mapa global permaneça preciso e eficiente, sem ser sobrecarregado por dados desnecessários. O algoritmo verifica se há mais varreduras a serem processadas. Se houver, retorna à etapa inicial para adquirir novos dados do sensor e continuar o mapeamento. Caso contrário, o processo é finalizado.

Na seção seguinte, são apresentados experimentos realizados com um sensor LiDAR 2D aplicado a um sistema SLAM implementado com pacotes do ROS, permitindo observar e analisar os resultados esperados do comportamento do sistema.

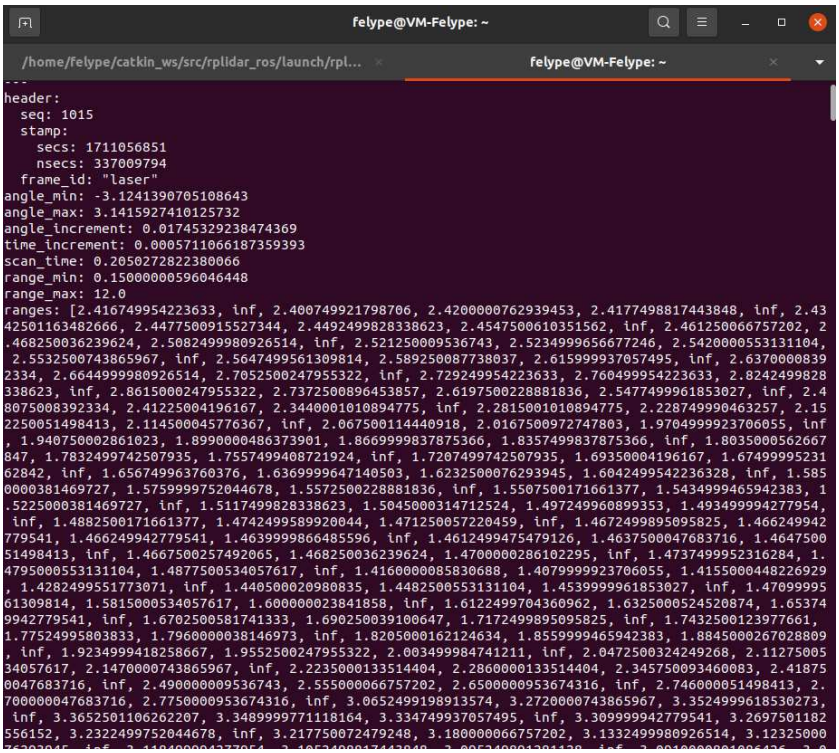
5.2 Análise preliminar: experimentos com LiDAR 2D no ROS

Após revisão bibliográfica sobre o ROS e instalação do software ROS Noetic em um computador com Linux Ubuntu 20.04, configurou-se um workspace para inicialização do LiDAR e etapas subsequentes da pesquisa. Utilizou-se o pacote `rplidar_ros`[1] para aquisição de dados do sensor RPLIDAR A1. O comando `rostopic echo /scan` permitiu exibir, em tempo real, os dados numéricos do sensor (Figura 5.5), apresentados no formato *ROS*

[1] <https://github.com/robopeak/rplidar_ros>

PointCloud Message, empregado no ROS. Os principais campos desse formato são:

Figura 5.5: Dados numéricos do sensor RPLIDAR A1



Fonte: Autor.

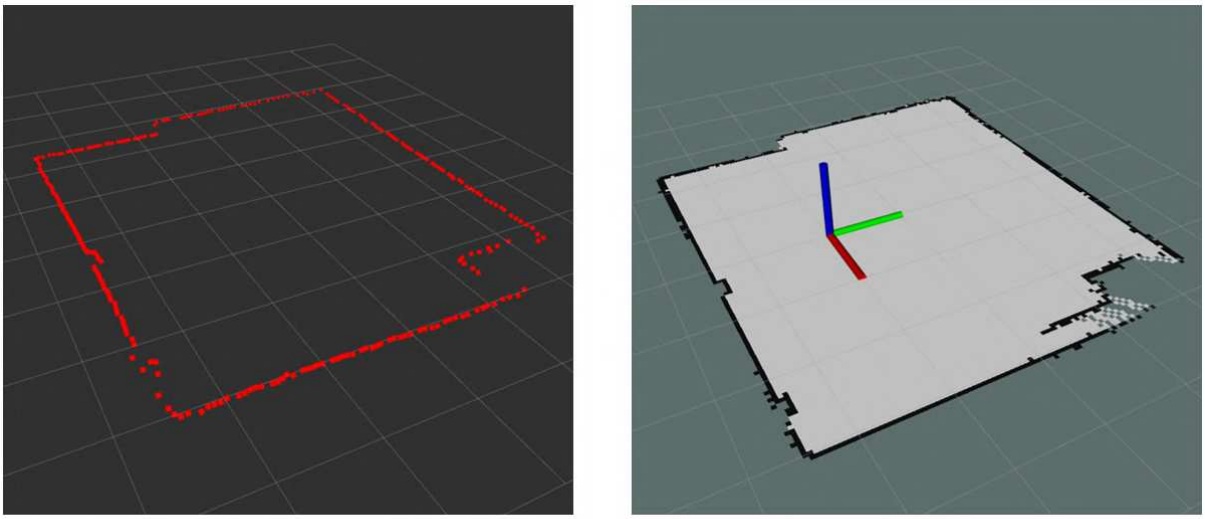
- `Header`: informações gerais da mensagem, incluindo número de sequência (`seq`) e carimbo de tempo (`stamp`);
- `angle_min`: ângulo mínimo da varredura, em radianos;
- `angle_max`: ângulo máximo da varredura, em radianos;
- `angle_increment`: resolução angular entre medições consecutivas, em radianos;
- `time_increment`: intervalo temporal entre medições consecutivas, em segundos;
- `scan_time`: tempo total para completar uma varredura, em segundos;
- `range_min`: distância mínima detectável pelo sensor, em metros;
- `range_max`: distância máxima detectável pelo sensor, em metros;

- `ranges`: lista das distâncias medidas pelo sensor, com valores `inf` indicando ausência de leitura;
- `intensities`: lista das intensidades de retorno das medições, úteis para identificação de superfícies.

Esses dados permitem a reconstrução do ambiente e a tomada de decisões na navegação robôs móveis. Por meio do software Rviz, visualizou-se graficamente a dispersão dos pontos captados pelo sensor (Figura 5.6 (a)).

Após essa etapa inicial, avançou-se para o mapeamento e localização simultâneos, utilizando o pacote `hector_slam`[2]. Realizaram-se ajustes específicos nos arquivos de lançamento do pacote, adequando-os ao modelo LiDAR utilizado. Em seguida, iniciaram-se o ROS e o sensor RPLIDAR A1, e executou-se o arquivo do HectorSLAM. Os dados do ambiente foram capturados e visualizados no Rviz, resultando na geração do mapa do local e na determinação da localização e orientação do sensor (Figura 5.6 (b)). O Hector SLAM utiliza a abordagem *Occupancy Grid*, onde as células em preto representam objetos ou obstáculos, e as células em cinza claro indicam espaço livre.

Figura 5.6: Dados do sensor RPLIDAR A1 visualizados no Rviz



(a) (b)

Fonte: Autor.

[2] <https://github.com/tu-darmstadt-ros-pkg/hector_slam?tab=readme-ov-file>

5.3 Aquisição e integração: dados do LiDAR 2D e Interface Gráfica

Após verificar o funcionamento esperado de um sistema SLAM, a pesquisa avançou para a etapa de aquisição de dados e integração com a interface gráfica desenvolvida.

Para a aquisição dos dados do sensor LiDAR 2D, foi utilizado o SDK público do fabricante[1], desenvolvido em C++ e disponibilizado como código aberto sob a licença GPLv3. Esse SDK oferece a biblioteca principal para comunicação com o sensor, além de aplicativos de demonstração que facilitam o desenvolvimento e a integração.

Com base nos aplicativos de demonstração disponibilizados pelo SDK do sensor, foi desenvolvido em C++, utilizando o Visual Studio 2022, um aplicativo de console para leitura dos dados provenientes do sensor pela porta USB. Para isso, configurou-se o projeto com vínculo à biblioteca estática fornecida (`rplidar_driver.lib`), ajustando-se o ambiente para a opção `Multi-threaded Debug (/MTd)` devido ao tipo estático da biblioteca utilizada.

Dessa forma, tornou-se possível inicializar o sensor LiDAR e obter os dados coletados, incluindo a confiabilidade da leitura indicada pelo nível de qualidade (Figura 5.7). As informações exibidas no console consistem em uma sequência de dados que incluem o ângulo (graus) no momento da captura, a distância correspondente (milímetros) e a qualidade do sinal, que varia de 0 a 15. Esse valor de qualidade indica o nível de confiabilidade da medição realizada.

Os dados originais do sensor estão em forma polar. Para convertê-los ao sistema cartesiano, utilizam-se as seguintes relações, considerando-se r como a distância medida e θ o ângulo em radianos, segue-se

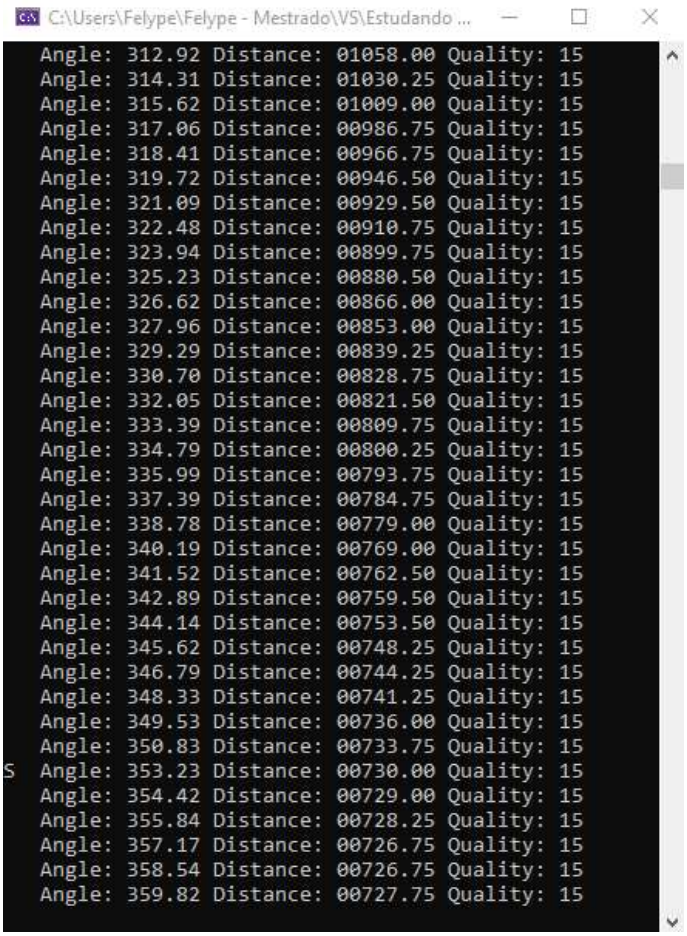
$$x = r \cdot \cos(\theta_{\text{rad}}), \tag{5.1}$$

$$y = r \cdot \sin(\theta_{\text{rad}}). \tag{5.2}$$

Em seguida, esses dados convertidos são integrados à interface gráfica desenvolvida, que exibe as leituras do sensor LiDAR em forma gráfica, conforme ilustra a Figura 5.8, que apresenta o momento do plote de uma varredura realizada em um ambiente.

[1] <https://github.com/Slamtec/rplidar_sdk>

Figura 5.7: Leitura de informações do LiDAR



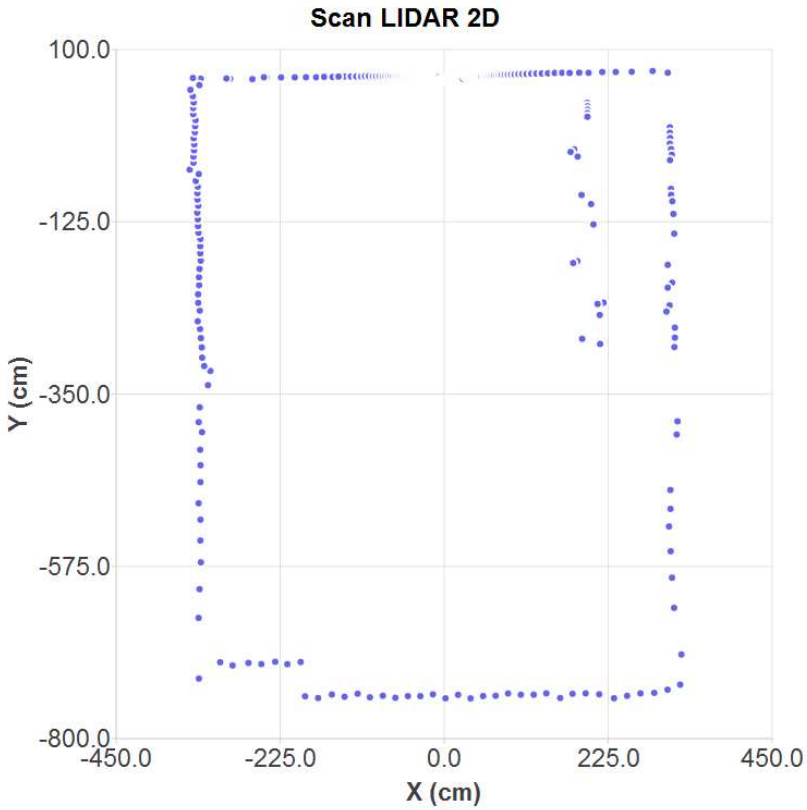
Fonte: Autor.

Durante o desenvolvimento inicial do projeto, identificou-se um problema de incompatibilidade no tempo de execução entre a biblioteca estática (`rplidar_driver.lib`) do sensor LiDAR e a configuração do projeto com interface gráfica Qt no Visual Studio 2022 (*runtime* MDd). Para resolver essa incompatibilidade, foi desenvolvida uma biblioteca dinâmica (DLL)[2] intermediária, que encapsula as funções da biblioteca estática original. Dessa forma, tornou-se possível operar o sensor sem modificar diretamente sua biblioteca estática, garantindo a compatibilidade com o *runtime* da aplicação Qt.

A DLL foi criada com o arquivo estático integrado e exporta apenas as funções essenciais para utilização do LiDAR, solucionando o erro `LNK2038` referente à incompatibilidade entre *runtimes* estático (MTd) e dinâmico (MDd). Ao concluir essa etapa, o sensor foi operado

[2] <https://github.com/flucena01/rplidar_driver_dll>

Figura 5.8: Scan LiDAR 2D



Fonte: Autor.

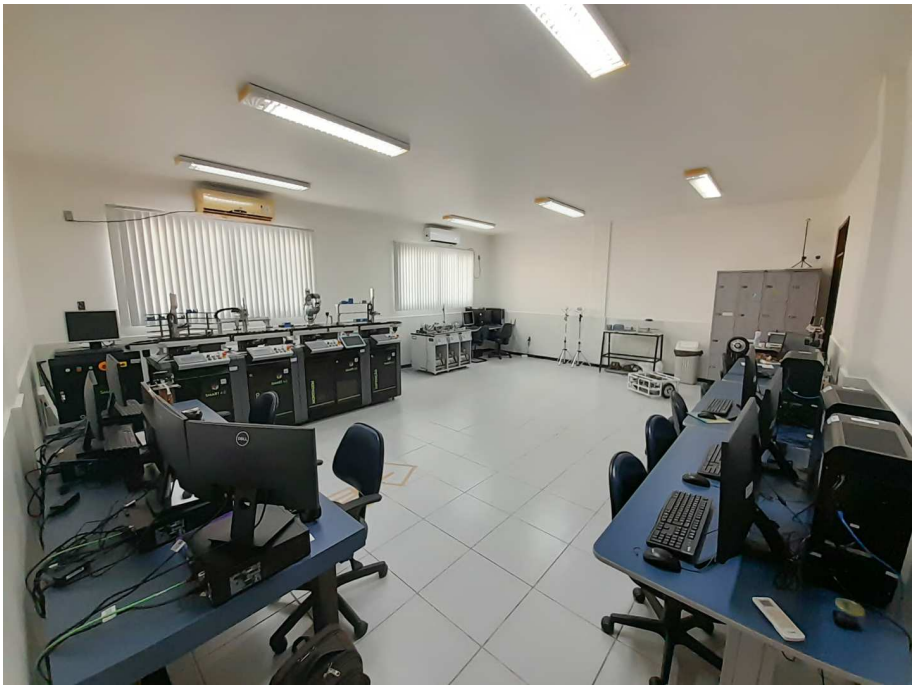
com sucesso pela interface gráfica. Uma tentativa anterior de implementação manual da leitura direta da porta serial revelou-se inviável devido à complexidade na decodificação dos dados brutos recebidos.

5.4 Análise dos algoritmos ICP

Para avaliar a eficiência do método proposto, ICP Gauss-Newton (GN) com métrica ponto-a-plano, foi comparado com o ICP clássico ponto-a-ponto e o ICP otimizado com Gauss-Newton com métrica ponto-a-ponto.

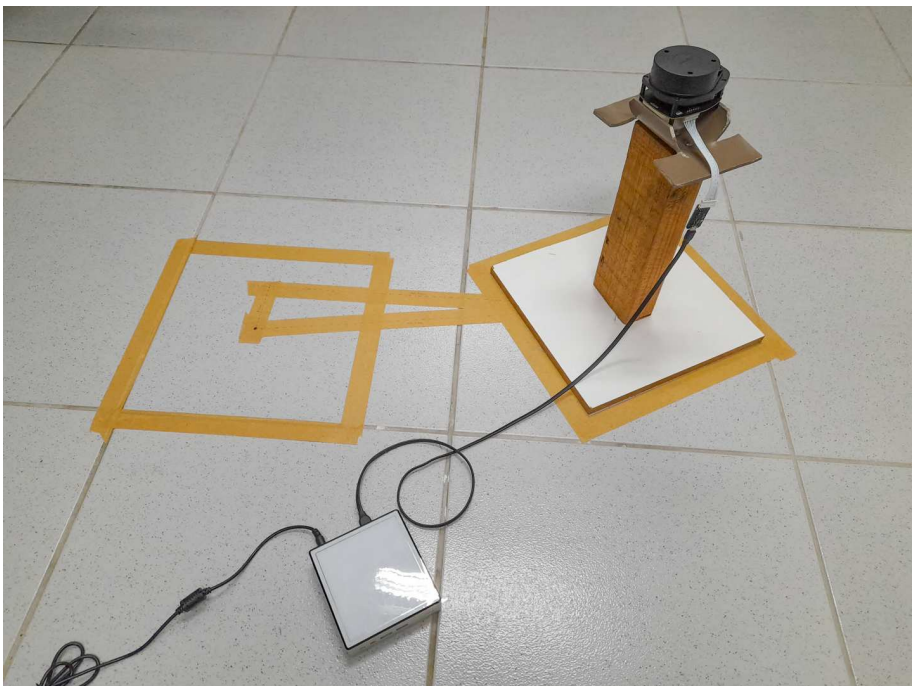
Previamente, foram coletados os dados de uma sala do laboratório, cujo ambiente pode ser observado na Figura 5.9. O sensor LiDAR 2D foi posicionado sobre um suporte improvisado no chão, conforme ilustrado na Figura 5.10, em uma altura semelhante àquela em que será instalado na plataforma experimental.

Figura 5.9: Ambiente de teste para avaliação de pose estimada



Fonte: Autor.

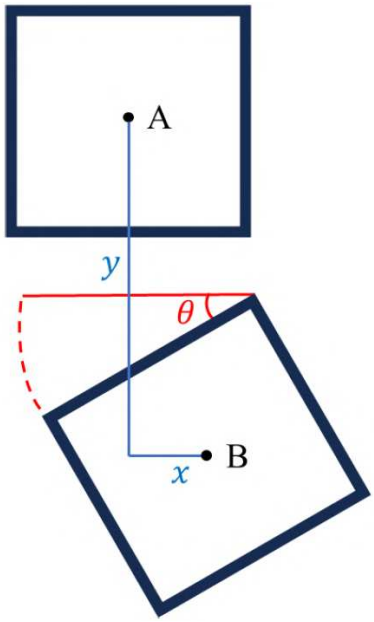
Figura 5.10: Coleta de dados com o sensor LiDAR para teste de eferência de pose



Fonte: Autor.

Para garantir precisão na avaliação, foram realizadas marcações no piso utilizando a base quadrada do suporte. Foram definidas duas posições distintas A e B, permitindo a medição da distância cartesiana nos eixos x e y, além da rotação angular, como representado na Figura 5.11. Na posição A, a base do suporte foi alinhada paralelamente à cerâmica. Já na posição B, o suporte foi rotacionado propositalmente 30°, utilizando as linhas da cerâmica como referência para garantir a consistência na medição do ângulo de rotação.

Figura 5.11: Representação esquemática para calcular pose



Fonte: Autor.

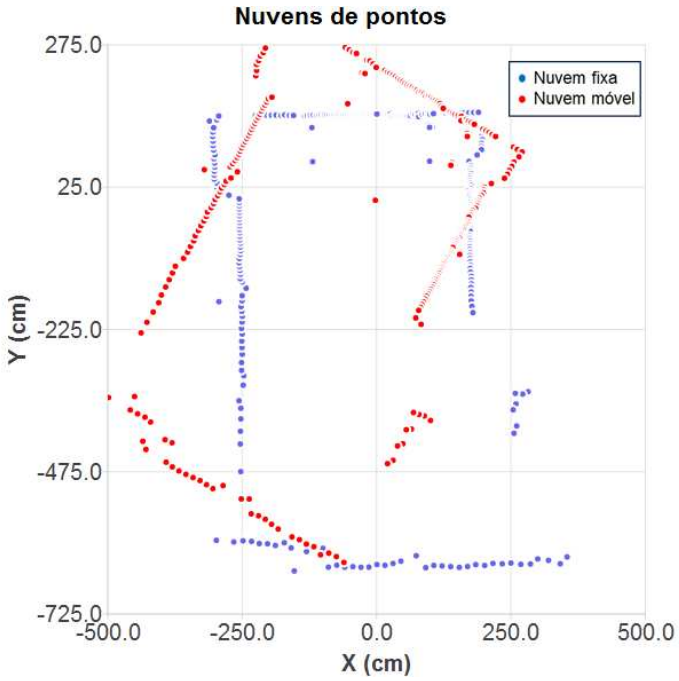
A partir dessas marcações, a medição foi realizada considerando o ponto central do suporte, que representa o centro do sensor LiDAR 2D. Os valores obtidos foram: $x = 9,2$ cm, $y = 55,5$ cm e 30°.

Após essas marcações, os dados foram capturados nas duas posições distintas utilizando um mini-PC, escolhido por proporcionar um melhor manuseio. Em seguida, os dados foram organizados e armazenados no formato `csv` para o processamento. Com esses procedimentos concluídos, foi possível testar os algoritmos desenvolvidos no Visual Studio 2022 nesse cenário experimental.

5.4.1 Resultados experimentais com algoritmos ICP

Na Figura 5.12, são apresentadas as duas nuvens de pontos capturadas pelo sensor no ambiente. A nuvem fixa (em azul) serve como referência para o alinhamento da nuvem móvel (em vermelho). Esses conjuntos de dados foram utilizados para teste dos três algoritmos avaliados.

Figura 5.12: Posição de origem das nuvem de pontos

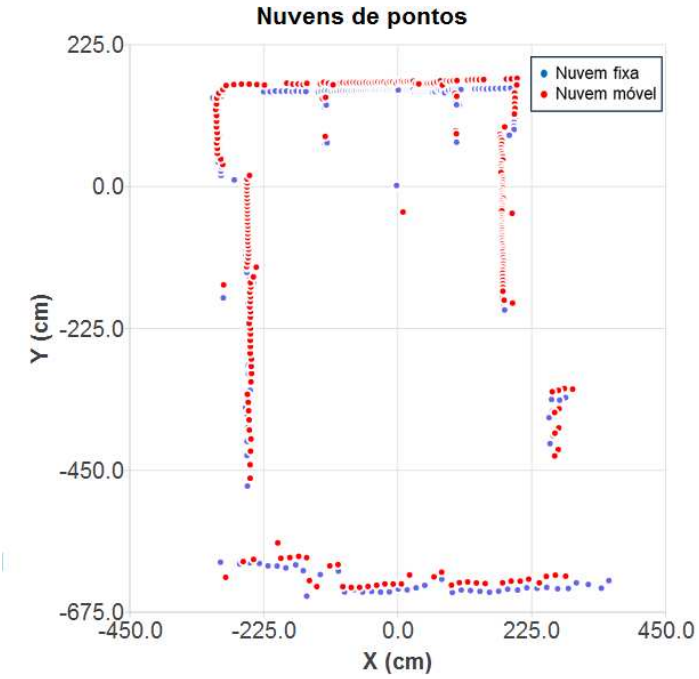


Fonte: Autor.

Nas Figuras 5.13, 5.14 e 5.15, são apresentados os resultados das nuvens de pontos convergidas utilizando três abordagens diferentes do algoritmo ICP. A Figura 5.13 ilustra o resultado do ICP clássico com a métrica ponto-a-ponto, enquanto a Figura 5.14 exibe a aplicação do ICP com otimização Gauss-Newton para a mesma métrica. Por fim, a Figura 5.15 mostra os resultados obtidos ao utilizar o ICP com Gauss-Newton e a métrica ponto-a-plano.

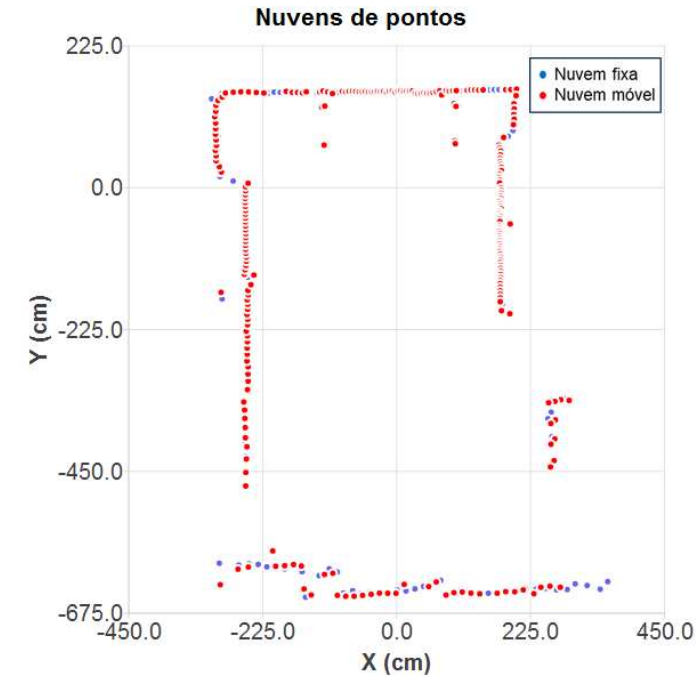
Visualmente, a comparação entre os métodos evidencia que a abordagem clássica ponto-a-ponto (Figura 5.13) pode apresentar um desalinhamento maior, perceptível, em relação às abordagens com otimização Gauss-Newton (Figuras 5.14 e 5.15). No método clássico, os pontos da nuvem móvel nem sempre se alinham corretamente com a nuvem fixa, resultando

Figura 5.13: Nuvem de pontos convergida usando o algorítimo ICP Clássico Ponto-a-Ponto



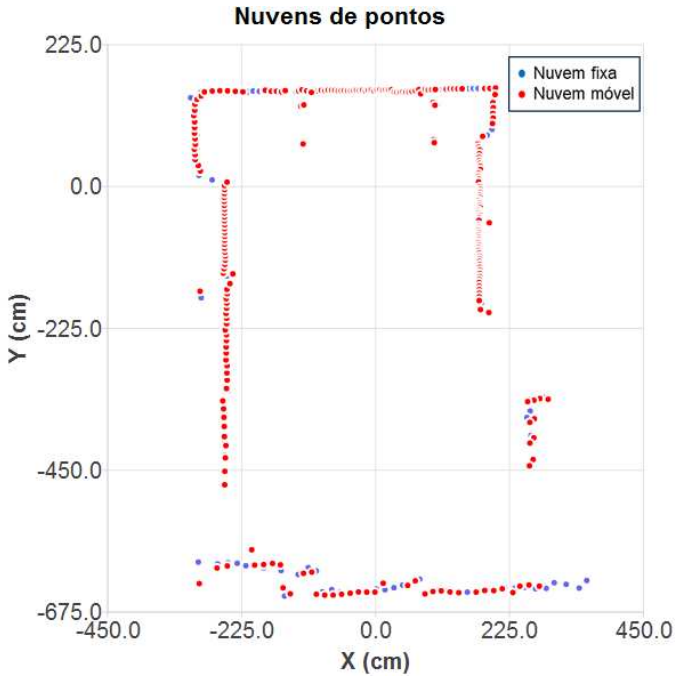
Fonte: Autor.

Figura 5.14: Nuvem de pontos convergida usando o algorítimo ICP GN Ponto-a-Ponto



Fonte: Autor.

Figura 5.15: Nuvem de pontos convergida usando o algorítimo ICP GN Ponto-a-Plano



Fonte: Autor.

em um encaixe menos preciso. Já nas abordagens otimizadas, a transformação estimada consegue sobrepor grande parte da nuvem móvel à fixa, melhorando o ajuste.

Esse comportamento está diretamente relacionado à forma como o sensor LiDAR 2D captura os pontos. Ele registra valores dos ângulos que variam ligeiramente a cada ciclo de aquisição. Essa variação ocorre devido a pequenos desvios no mecanismo de rotação e nas características do sensor, fazendo com que os pontos coletados não correspondam exatamente às posições das leituras anteriores, gerando então uma não linearidade.

Dessa forma, o método clássico ponto-a-ponto se torna inadequado, pois depende da correspondência direta entre os pontos mais próximos sem considerar a orientação da superfície local. Em contrapartida, a métrica ponto-a-plano (Figura 5.15) leva em consideração a orientação das superfícies, garantindo um ajuste mais preciso ao minimizar o erro ao longo das normas. Essa abordagem é mais robusta para lidar com variações naturais nas leituras do sensor, resultando em um alinhamento mais estável e confiável.

As Figuras 5.16, 5.18 e 5.20 ilustram com maior clareza a eficiência e precisão de cada método, mostrando o comportamento do erro ao longo das iterações.

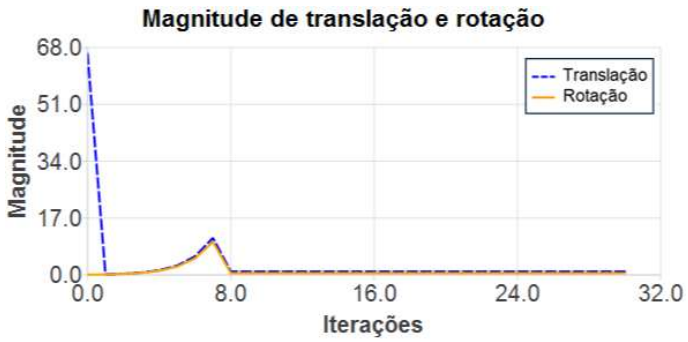
O algoritmo ICP Clássico foi configurado para limitar a rotação e a translação ao atingir

um determinado erro, o que justifica o comportamento do erro na Figura 5.16. Inicialmente, há um rápido decaimento, seguido de uma estabilização. Essa estratégia foi implementada devido à natureza do algoritmo, que realiza uma busca exaustiva e considera a convergência quando o erro atinge um limiar predefinido. O gráfico na Figura 5.17 mostra um comportamento oscilatório em alguns momentos. A magnitude inicial da translação é alta, em seguida acompanha com a magnitude da rotação até atingir um pico. Neste ponto, o limite de movimento é ativado, levando a uma estabilização até a convergência. Foi necessário usar essa técnica devido ao mecanismo de busca exaustiva deste algoritmo.

Figura 5.16: Erro médio algorítimo ICP Clássico Ponto-a-Ponto

Fonte: Autor.

Figura 5.17: Magnitude do algorítimo Clássico ICP Ponto-a-Ponto



Fonte: Autor.

A tolerância de erro foi ajustada para 12, pois esse valor proporcionou o menor erro final. Como resultado, o algoritmo necessitou de 29 iterações para convergir e alcançar o erro 11,42.

Esse comportamento reforça a limitação do método clássico em lidar com diferenças não

lineares entre as nuvens de pontos, exigindo mais iterações para alcançar um alinhamento não satisfatório.

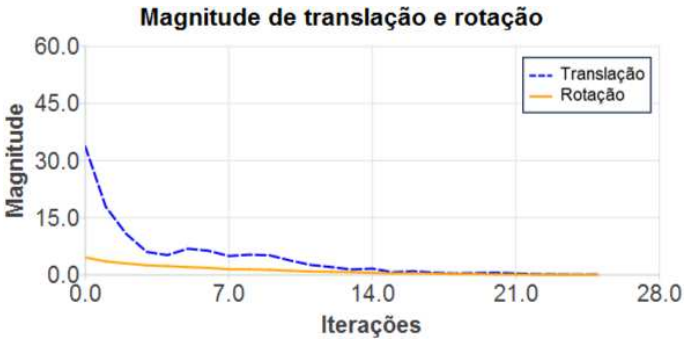
No algoritmo ICP com otimização Gauss-Newton e métrica ponto-a-ponto, a convergência ocorre de forma eficiente, a taxa de redução do erro diminui e a curva se estabiliza, sugerindo que o método atingiu um mínimo estável. conforme a Figura 5.18. Comparado ao método clássico, o erro cai de forma abrupta e se estabiliza em torno de 3,69 após a iteração 23, demonstrando um alinhamento mais rápido e mais preciso que o anterior.

O gráfico da Figura 5.19 evidencia um comportamento mais estável na magnitude da translação e rotação, com uma rápida aproximação, nas primeiras iterações.

Figura 5.18: Erro médio algorítimo ICP GN Ponto-a-Ponto

Fonte: Autor.

Figura 5.19: Magnitude do algorítimo ICP GN Ponto-a-Ponto



Fonte: Autor.

Já no método ICP Gauss-Newton ponto-a-plano, o erro, mostrado na Figura 5.20, atinge seu menor valor na iteração 7, estabilizando o erro em torno de 3,65. Esse comportamento reforça sua a eficácia em cenários onde as normas associadas dos pontos são bem defini-

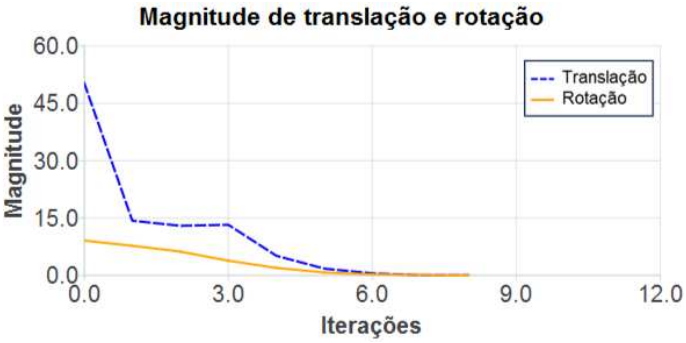
das, permitindo um refinamento direto da transformação sem oscilações ou necessidade de múltiplas iterações.

Além disso, o gráfico da magnitude da translação e rotação, apresentado na Figura 5.21, destaca que o ajuste do ocorre de maneira rapidamente, com uma queda significativa nas primeiras iterações, devido a redução da dimensionalidade do problema com a métrica ponto-a-plano (Equação 3.32).

Figura 5.20: Erro médio algorítimo ICP GN Ponto-a-Plano

Fonte: Autor.

Figura 5.21: Magnitude do algorítimo ICP GN Ponto-a-Plano



Fonte: Autor.

Nos algoritmos com otimização Gauss-Newton, a convergência foi determinada pelo critério de variação mínima do erro, definida como 1e-6. Diferentemente, no método clássico, o critério de convergência incluiu a verificação da iteração seguinte, se o erro aumentasse, a transformação anterior seria mantida como a melhor solução encontrada.

5.4.2 Avaliação do desempenho

A análise dos resultados apresentados na Tabela 5.1 permite uma avaliação do desempenho de cada abordagem em termos de precisão, eficiência computacional e convergência.

Em temos de acurácia, ao comparar a pose estimada com a pose verdadeira, observa-se que o ICP Clássico apresenta os maiores desvios às dimensões (x, y). Já o ICP GN ponto-a-ponto melhora a acurácia, com erro reduzido e estimativa próxima da posição real. No entanto, ainda apresenta um desvio notável no eixo x e y. O ICP GN ponto-a-plano apresenta a melhor acurácia relativa a pose verdadeira entre as abordagens comparadas, atingindo 99,42% para x, 99,68% para y e 99,99% para θ.

Em temos de eficiência, o ICP GN ponto-a-plano obteve o menor erro médio (3,6578), seguido pelo ICP GN ponto-a-ponto (3,6930), enquanto o ICP Clássico apresentou um erro muito maior (11,6636), reforçando sua menor eficácia. O ICP Clássico necessitou 29 iterações para convergir, enquanto o ICP GN ponto-a-ponto reduziu esse número para 23 iterações, e o ICP GN ponto-a-plano foi ainda mais eficiente, alcançando a convergência em apenas 7 iterações.

Quanto ao tempo total de execução dos métodos, fator essencial para aplicações em tempo real, o ICP GN ponto-a-plano, apesar de apresentar menor número de iterações e maior precisão, não foi o mais rápido, com tempo de execução de 278 ms. O cálculo das normas associadas na métrica ponto a plano, adiciona um custo computacional extra. O ICP Clássico foi mais rápido, com tempo de 215 ms, enquanto o ICP GN ponto-a-ponto foi o mais lento, levando 547 ms.

Analisando o tempo gasto por CPU, o ICP Clássico foi o mais leve (6,88 ms), seguido pelo ICP GN ponto-a-ponto (25,83 ms) e pelo ICP GN ponto-a-plano (27,56 ms). Isso mostra que, embora a otimização Gauss-Newton exija um maior esforço computacional, sua rápida convergência compensa essa demanda.

A utilização média da CPU, por sua vez, mostra que o ICP Clássico teve um consumo de 5,60% do processamento, o ICP GN ponto-a-ponto 14,10%, e o ICP GN ponto-a-plano 13,62%. Esse resultado sugere que, apesar do maior tempo de execução do ICP GN ponto-a-plano, sua utilização de CPU foi mais eficiente em comparação ao ICP GN ponto-a-ponto.

Os resultados indicam que o ICP GN ponto-a-plano foi o mais preciso e eficiente em

termos de convergência, exigindo menos iterações e fornecendo uma estimativa mais próxima da pose verdadeira. Entretanto, seu tempo de execução foi equilibrado comparados as abordagens comparadas, o que pode ser um fator relevante para aplicações em tempo real.

Já o ICP GN ponto-a-ponto apresenta um equilíbrio entre precisão e custo computacional, sendo uma alternativa viável caso não haja necessidade de um refinamento extremo do alinhamento. O ICP Clássico, por outro lado, mostrou-se limitado tanto em precisão quanto em eficiência, necessitando de muitas iterações para convergir e ainda assim apresentando maior erro.

Portanto, os resultados confirmam a superioridade do ICP Gauss-Newton com métrica ponto-a-plano, sendo a melhor escolha para cenários que exigem equilíbrio entre alta precisão e rápida convergência. Embora o erro não atinja valores próximos de zero, isso é esperado e aceitável, pois a métrica de erro considera a diferença entre a nuvem de pontos anterior e a nova nuvem adquirida, que pode conter novos pontos a serem incorporados ao mapa global. Esse aspecto é essencial para garantir uma representação progressivamente mais consistente à medida que novos dados são processados.

Tabela 5.1: Resultados de desempenho de cada abordagem

Avaliação	ICP Clássico	ICP GN Ponto-a-Ponto	ICP GN Ponto-a-Plano
Pose (x, y, θ) verdadeira (cm)	9,5; -55,5; 30	9,5; -55,5; 30	9,5; -55,5; 30
Pose (x, y, θ) estimada (cm)	10,8063; -42,0348; 30,5742	9,4059; -54,9814; 30,0578	9,5554; -55,3249; 30,0014
Acurácia Relativa (x, y, θ) (%)	86,25; 75,75; 98,09	99,01; 99,07; 99,81	99,42; 99,68; 99,99
Erro médio (cm)	11,4264	3,6972	3,6528
Iterações	29	23	7
Tempo total de execução (ms)	215	641	278
Tempo gasto por CPU (ms)	6,8834	25,8368	27,5617
Utilização média da CPU (%)	5,6029	14,1001	13,6279

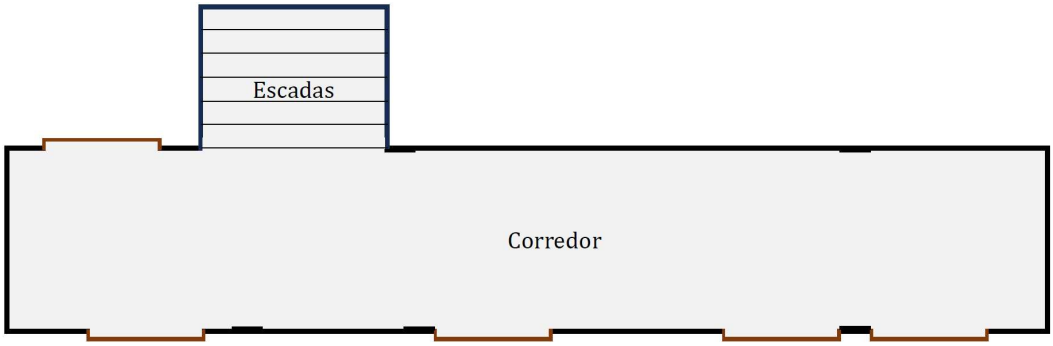
Fonte: Autor.

5.5 SLAM offline: processamento de dados capturados em diferentes posições

A abordagem baseada no ICP com métrica ponto-a-plano e otimização por Gauss-Newton apresentou melhor desempenho na estimativa da pose do robô, sendo considerada a mais adequada dentre as avaliadas. Com base nesses resultados, foi implementado um sistema incremental de mapeamento e localização (SLAM), adotando-se o método que demonstrou maior eficiência.

O sistema foi desenvolvido para processar um banco de dados contendo 16 conjuntos de pontos (armazenados em arquivos `csv`) capturados por um sensor LiDAR 2D, posicionados em diferentes locais ao longo de um corredor de laboratório, para simular leituras sequenciais em um cenário equivalente à operação contínua de um robô móvel. Tal procedimento viabilizou a criação de um ambiente controlado para o desenvolvimento, depuração e validação do sistema antes de sua aplicação em tempo real com o veículo AGV. O formato da área onde o experimento foi conduzido é ilustrado na Figura 5.22, e o processo de aquisição no ambiente do experimento é mostrado na Figura 5.23.

Figura 5.22: Formato do ambiente do experimento



Fonte: Autor.

Cabe destacar que, nessa etapa, não foram realizadas medições de referência para as poses reais durante cada varredura. Assim, a avaliação do sistema incremental ICP foi conduzida de forma qualitativa, com base na coerência visual das posições estimadas e mapa gerado, e no conhecimento prévio da geometria do ambiente. Como o estudo comparativo já havia validado a acurácia da abordagem adotada com dados reais, o foco dessa etapa

Figura 5.23: Registro da captura de dados

Fonte: Autor.

esteve na integração e no comportamento incremental do sistema.

Conforme ilustrado na Figura 5.4, a estratégia adotada baseia-se no alinhamento entre varreduras consecutivas, o algoritmo desenvolvido inicia transformando a primeira nuvem de pontos diretamente para o sistema de coordenadas do mapa global, uma vez que ainda não há uma referência anterior disponível. A partir da segunda varredura, a nuvem previamente processada é utilizada como referência (nuvem fixa), enquanto a nova varredura, obtida a partir de uma pose distinta, é tratada como a nuvem móvel.

O algoritmo de ICP é então processado para estimar a transformação rígida, composta por translação e rotação (x, y, θ), que melhor alinha a nuvem móvel à nuvem fixa. A transformação resultante é utilizada tanto para atualizar a estimativa de pose quanto para reposicionar a nuvem móvel em relação ao mapa global. Após o alinhamento, a nuvem móvel é transformada para o sistema de coordenadas global utilizando os parâmetros estimados e incorporada ao mapa acumulado. Em seguida, essa nuvem transformada passa a ser considerada como a nova referência (nuvem fixa), e o processo é repetido iterativamente para as próximas varreduras, enquanto houver dados disponíveis.

Após a transformação dos pontos alinhados pelo ICP para o sistema de coordenadas global, foi usado um filtro de redundância de pontos. O objetivo é evitar a inserção de pontos muito próximos entre si, prevenindo a redundância e o excesso de dados semelhantes no mapa. Assim foi estabelecido o critério, dado um novo ponto $p_{new} = (x_{new}, y_{new})$, que será adicionado ao mapa global, é verificado sua distância euclidiana em relação a todos os pontos já presentes no conjunto:

$$d(p_{\text{new}}, p_i) = \sqrt{(x_{\text{new}} - x_i)^2 + (y_{\text{new}} - y_i)^2}. \tag{5.3}$$

Se existir qualquer ponto p_i tal que $d(p_{\text{new}}, p_i) < D_{min}$, o ponto é descartado. No código, esse limiar está definido como $D_{min} = 10$ (10 cm).

Os resultados da execução do algoritmo de SLAM baseado no ICP Gauss-Newton com métrica ponto-a-plano são apresentados nos gráficos das Figuras 5.24, 5.25 e 5.26 que mostram a evolução do mapeamento, da posição estimada e da trajetória ao longo do processamento das varreduras capturadas. A pose estimada do robô é representada pela seta azul, os pontos em preto correspondem ao mapa gerado, e a linha contínua indica a trajetória percorrida.

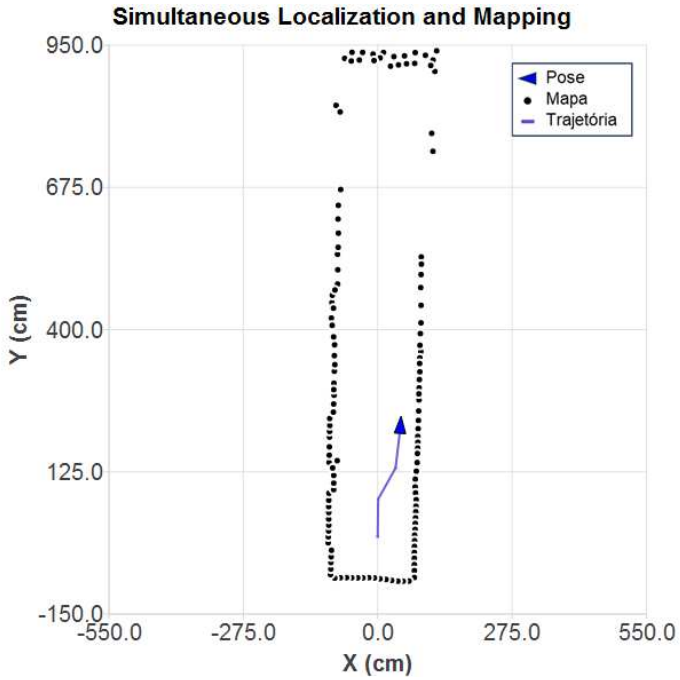
Na Figura 5.24, observa-se os primeiros estágios da construção do mapa. Nota-se, na região superior do gráfico, algumas discrepâncias devido a imprecisões nas leituras do sensor, especialmente em distâncias maiores, onde o erro tende a se acentuar.

Quando o robô se aproxima das escadas (Figura 5.25), há um aumento significativo na quantidade de *outliers*. Apesar do sistema estar filtrando os dados considerados de melhor qualidade pelo próprio sensor, ainda é possível perceber pontos incoerentes. Essa inconsistência está associada às limitações do sensor, conforme especificado pelo fabricante. O rodapé da escada é revestido com granito, um material reflexivo, o que compromete a precisão das medições do sensor LiDAR.

Na etapa final, mostrado na Figura 5.26, após o processamento de todas as varreduras, nota-se a presença de dispersões de pontos, reforçando a interferência causada pelo material reflexivo nas escadas. No entanto, a estimativa da pose do robô e a trajetória permaneceram fiéis com as posições reais das medições.

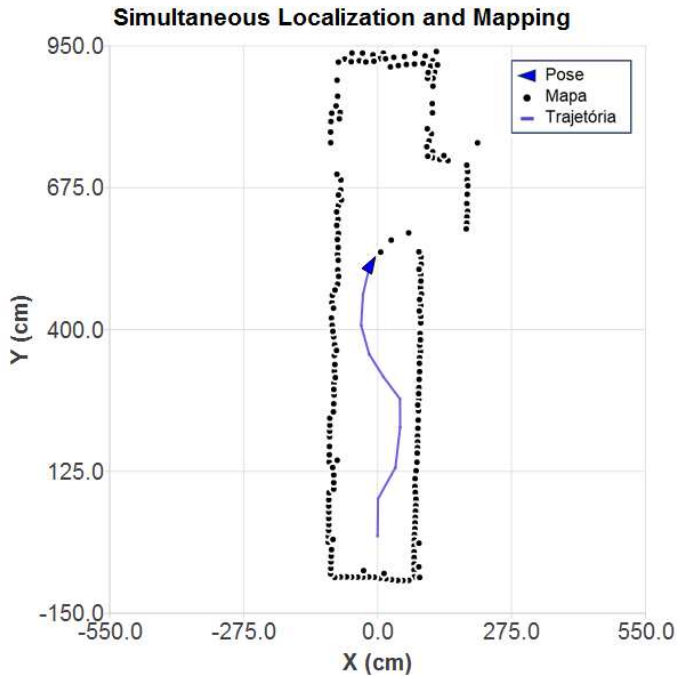
Para mitigar as limitações do sensor LiDAR 2D, foi implementado filtro de *outliers* para

Figura 5.24: Simulação SLAM com dados reais sem filtragem de *outliers* (estágio 1)



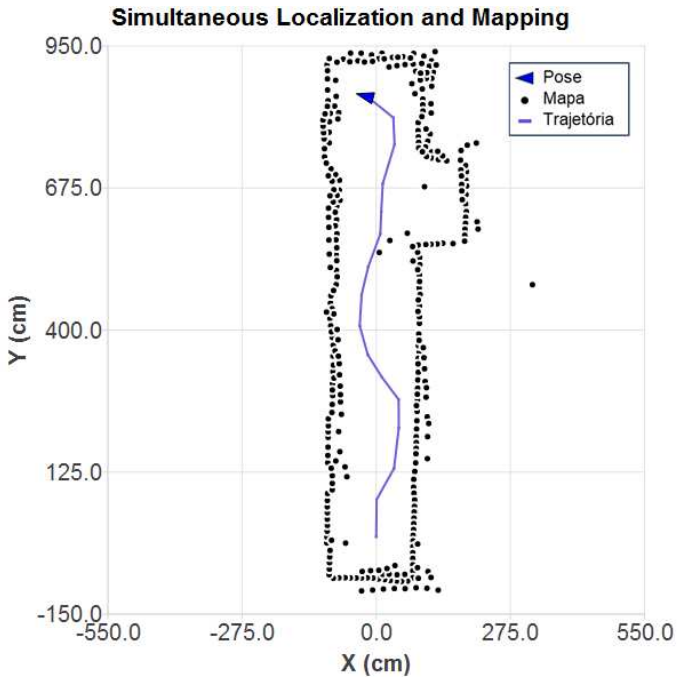
Fonte: Autor.

Figura 5.25: Simulação SLAM com dados reais sem filtragem de *outliers* (estágio 2)



Fonte: Autor.

Figura 5.26: Simulação SLAM com dados reais sem filtragem de *outliers* (estágio 3)



Fonte: Autor.

aprimorar a suavização do mapa global à medida que novos pontos são adicionados. O objetivo é identificar e eliminar pontos que estão isolados, ou seja, que não têm vizinhos próximos, pois geralmente são ruídos provenientes de erros no sensor. O critério definido foi, para cada ponto $p_i = (x_i, y_i)$, conta-se quantos vizinhos p_j (com $j \neq i$) estão dentro de um raio R. A contagem é feita conforme:

$$\left(d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right) < R. \tag{5.4}$$

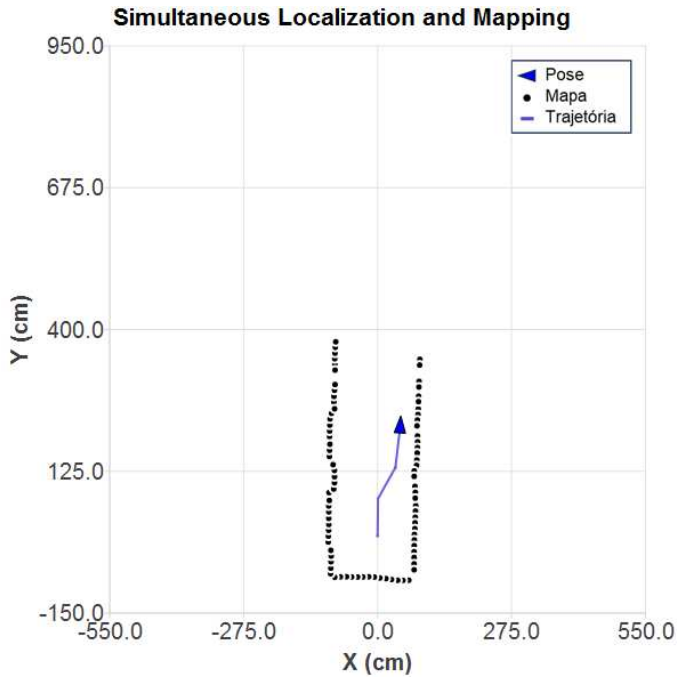
Se o número de vizinhos de p_i for menor que um limiar N_{min}, então p_i é considerado *outlier* e descartado da nuvem de pontos global. No código, $R = 14$ e $N_{min} = 1$.

Fazer esse procedimento assegura que os pontos mais relevantes condizentes com a estrutura do ambiente sejam mantidos no mapa global. Além disso, evita possíveis redundâncias e sobrecarga de dados desnecessários.

Nas Figuras 5.27, 5.28 e 5.29, são apresentados os resultados da execução do algoritmo SLAM com a aplicação da filtragem dos *outliers*.

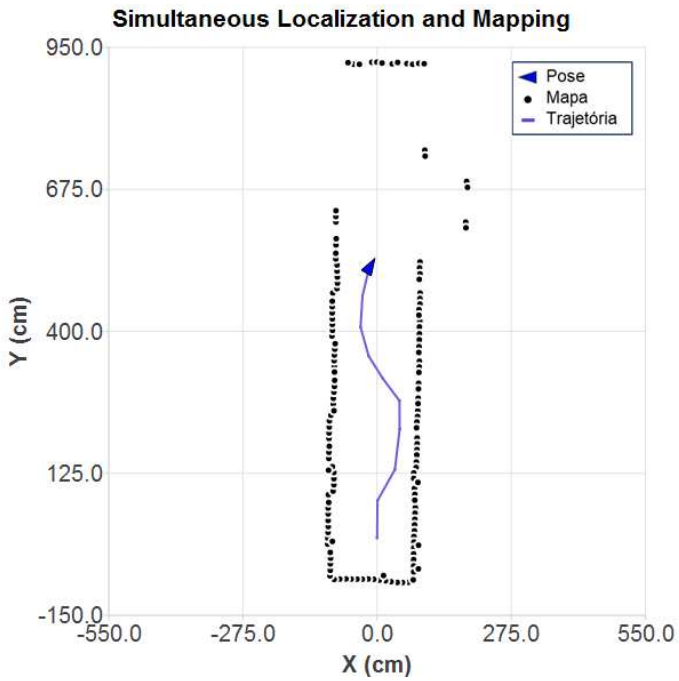
Comparando com os resultados anteriores, percebe-se que a dispersão de pontos espúrios

Figura 5.27: Simulação SLAM com dados reais com filtragem de *outliers* (estágio 1)



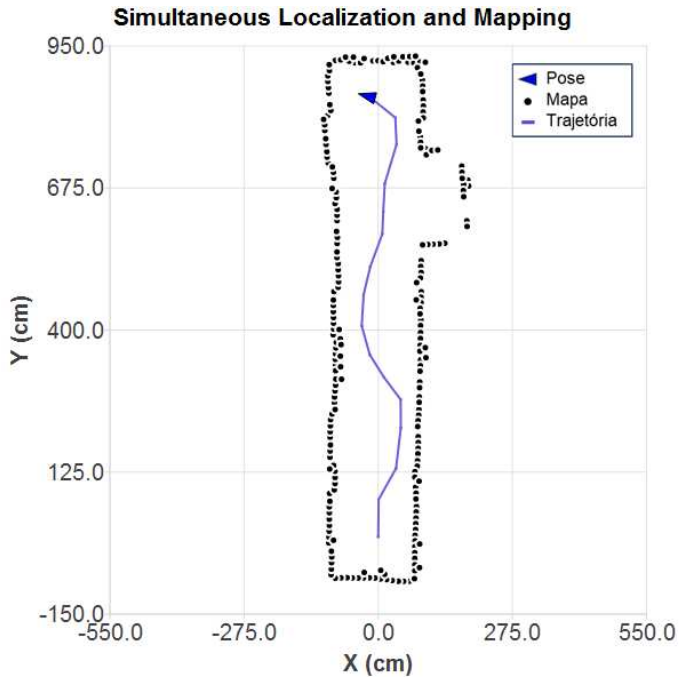
Fonte: Autor.

Figura 5.28: Simulação SLAM com dados reais com filtragem de *outliers* (estágio 2)



Fonte: Autor.

Figura 5.29: Simulação SLAM com dados reais com filtragem de *outliers* (estágio 3)



Fonte: Autor.

foi reduzida, tornando a estrutura do ambiente mais bem definida. Isso indica que a filtragem de pontos redundantes e de *outliers* foram eficazes na remoção de ruídos e na preservação apenas dos pontos mais relevantes. Observa-se a ausência de pontos na região das escadas (Figura 5.29). Essa ausência ocorre devido ao material reflexivo presente nessa área, que compromete a confiabilidade das leituras do sensor LiDAR.

O resultado final sugere que as técnicas de suavização aplicadas contribuem para um mapeamento mais confiável, reduzindo os efeitos de leituras imprecisas do sensor LiDAR e garantindo que apenas dados confiáveis sejam incorporados ao mapa. Os detalhes deste experimento são apresentados numericamente, a título de informação, na Tabela 5.2.

Tabela 5.2: Resultado experimental do sistema SLAM com suavização do mapa

Varredura	Quantidade de pontos do scan	Pose (x, y, θ)	Total de pontos no mapa global
1	273	0,0; 0,0; 0,0	48
2	263	0,9007; 72,0047; 11,3461	61
3	264	36,4819; 132,107; 12,2955	72
4	266	46,3461; 210,361; 3,3761	85
5	266	45,9615; 265,514; 334,318	90
6	268	12,0747; 308,191;325,807	100
7	265	-17,241; 352,112; 339,943	105
8	265	-33,8101; 408,244;1,4096	120
9	268	-29,8845; 468,539; 10,63	139
10	266	-16,332; 522,507; 30,6144	147
11	259	8,4066; 585,931; 5,8433	166
12	259	10,3514; 629,644; 358,738	178
13	259	13,4733; 683,434; 14,3188	195
14	258	37,6158; 759,465; 358,611	204
15	257	35,0859; 811,158; 328,649	213
16	258	-17,8156; 851,666; 285,088	221

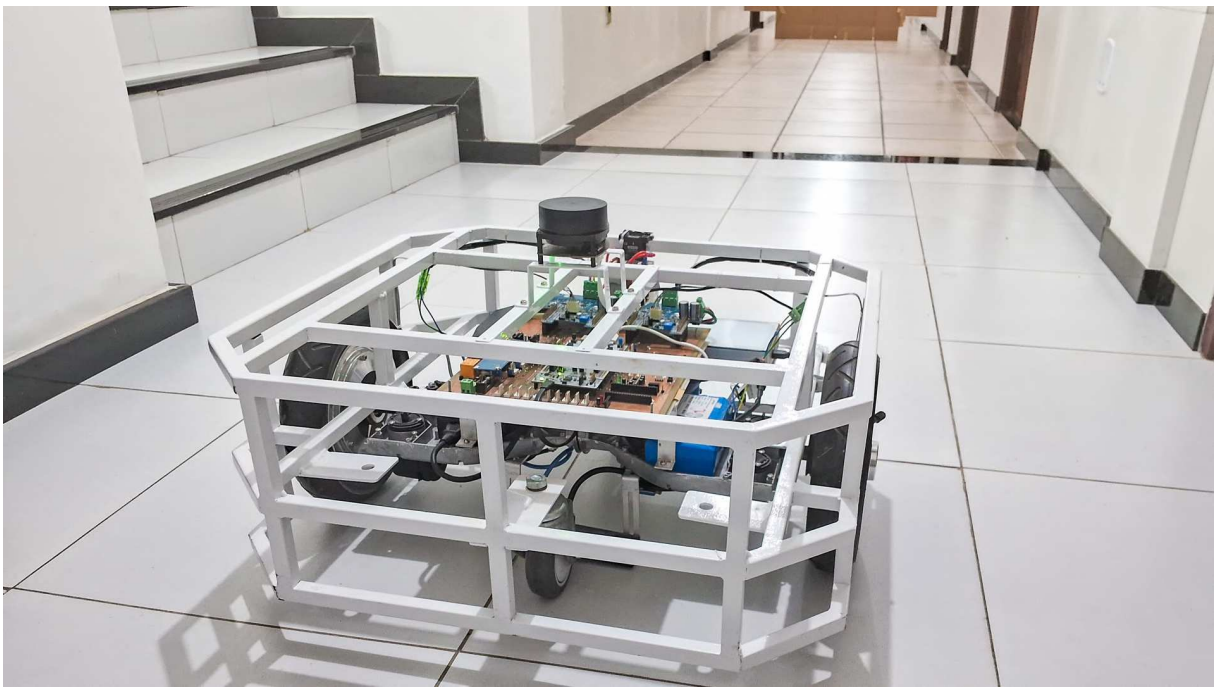
Fonte: Autor.

5.6 SLAM em tempo real: implementação direta com sensor LiDAR 2D

O presente experimento teve como objetivo desenvolver e testar um sistema SLAM em tempo real, utilizando diretamente os dados obtidos do sensor LiDAR integrado à plataforma robótica para a execução do algoritmo *scan-matching* proposto. Foi desenvolvida uma interface gráfica em Qt/C++ para visualização em tempo real das varreduras do sensor, estimativa da pose e construção do mapa.

A plataforma com o LiDAR 2D integrado é mostrada na Figura 5.30 durante um teste no ambiente do experimento. A construção inicial do mapa, é apresentada na Figura 5.31, sendo possível já identificar a estrutura geral do ambiente, enquanto na Figura 5.32, à medida que o deslocamento avança, é mostrado a progressão do mapeamento com a incorporação sucessiva de novas varreduras ao mapa global.

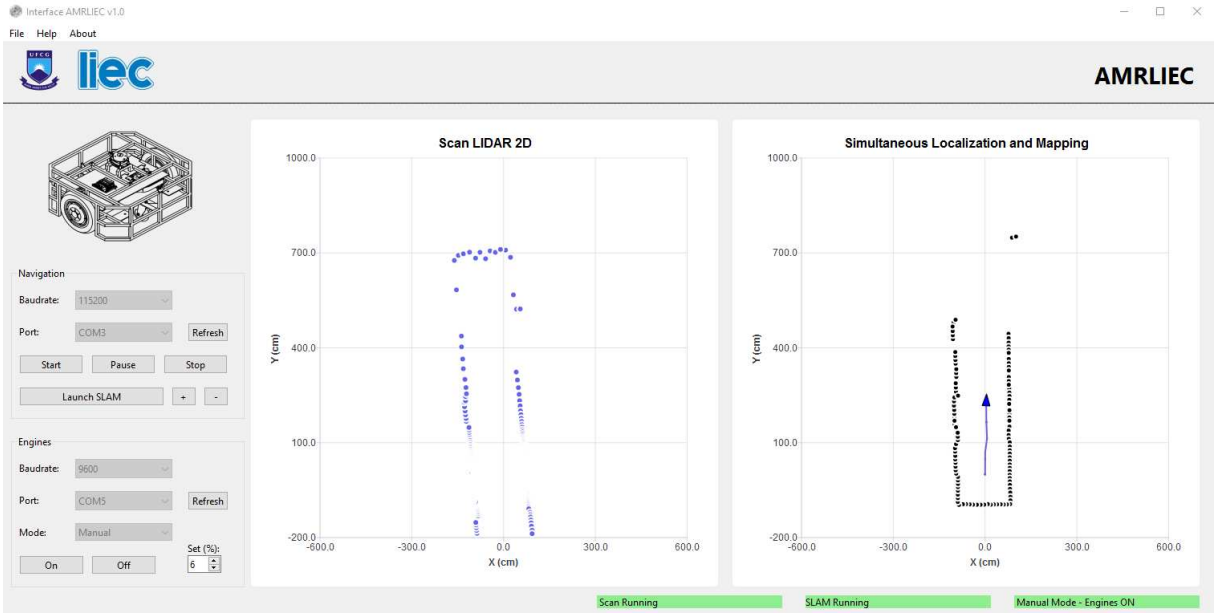
Figura 5.30: Registro visual do experimento com o veículo



Fonte: Autor.

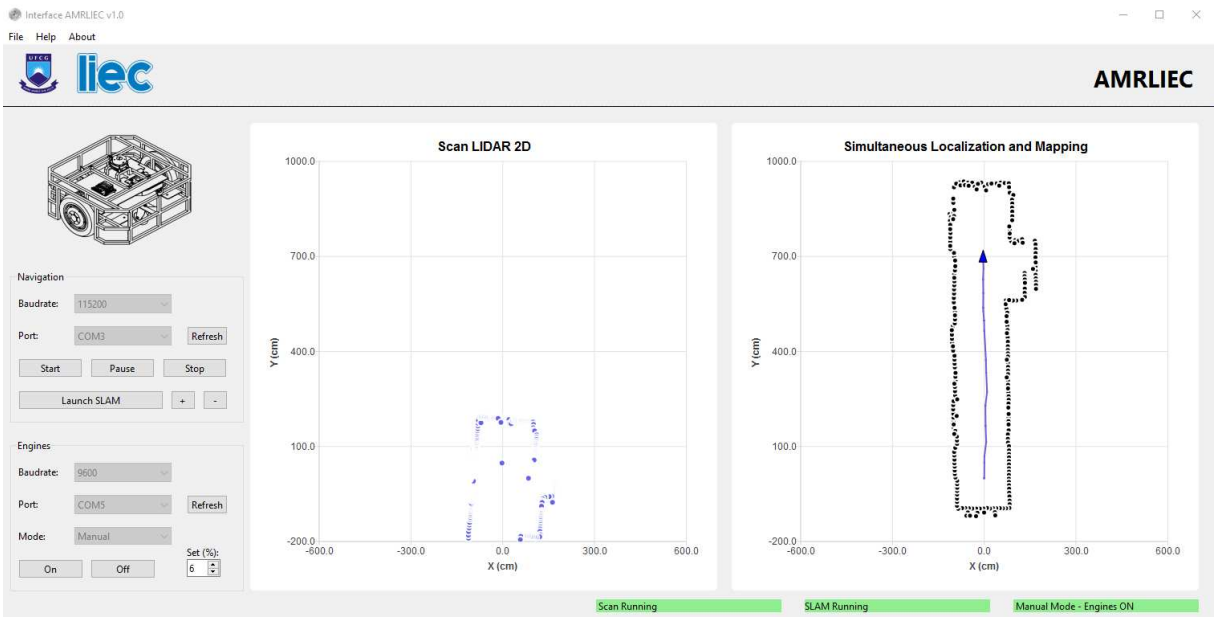
A interface oferece opções para configurar os parâmetros de comunicação do sensor LiDAR (baudrate e porta serial), além de botões dedicados para iniciar o LiDAR e o processo

Figura 5.31: Interface gráfica do usuário executando o SLAM em tempo real (estágios iniciais)



Fonte: Autor.

Figura 5.32: Interface gráfica do usuário executando o SLAM em tempo real (estágios finais)



Fonte: Autor.

SLAM. Adicionalmente, a interface possui uma seção específica para configuração da comunicação com o microcontrolador responsável pelo controle dos motores. Nessa seção, é

possível configurar os parâmetros (baudrate e porta serial) e selecionar entre três modos de operação: Manual (partida direta dos motores), Keyboard (controle dos motores através das teclas do teclado) e LiDAR 2D (controle dos motores utilizando dados obtidos pelo LiDAR).

Devido às limitações de *hardware* do mini-PC embarcado, a atualização da localização e do mapa ocorre a cada 3 s. O sistema utiliza *threads* assíncronas para melhor aproveitamento dos núcleos de processamento. No entanto, ao testar uma taxa de atualização de 1 s, a interface parou de responder, pois a classe responsável pelo ICP ainda estava em execução enquanto novos dados do LiDAR continuavam chegando. A taxa de 3 s foi considerada ideal, pois o ICP leva entre 2 e 2,5 s para convergir, permitindo a atualização da pose e a adição de novos pontos ao mapa sem comprometer a estabilidade do sistema.

Um dos principais desafios enfrentados durante os experimentos foi a limitação do sistema de controle dos motores da plataforma robótica. A ausência de controle de torque dificultou a execução de movimentos em baixa velocidade, o que impactou diretamente a capacidade de conduzir o robô por comandos manuais ou de forma automática com dados do LiDAR. Essa limitação comprometeu a qualidade de alguns experimentos, uma vez que o desempenho de sistemas SLAM tende a ser melhor em velocidades reduzidas e movimentos mais controlados.

O sistema SLAM em tempo real demonstrou consistência com os objetivos propostos, mas sua performance pode variar conforme o ambiente. Em locais com obstáculos finos, como mesas e cadeiras, a representação do mapa pode ser comprometida, assim como a estimativa da pose. Isso ocorre porque, na varredura do LiDAR, esses objetos geram pontos dispersos ou distorcidos, dificultando o alinhamento preciso no algoritmo ponto a plano. Além disso, a velocidade de movimentação do robô também influencia, deslocamentos mais suaves resultam em uma localização e um mapeamento mais consistentes.

5.7 Conclusões

Os experimentos comparativos entre os métodos ICP confirmaram a superioridade da métrica ponto-a-plano em relação às demais, com acurácia de 99,42%, 99,68% e 99,99% nas estimativas de x, y e θ, respectivamente, em um cenário controlado com poses conhecidas. Esse estudo foi essencial para validar a precisão dos algoritmos de alinhamento, permitindo

escolher a abordagem mais confiável para aplicação no sistema incremental.

Diferentemente do estudo comparativo, em um cenário simulado (SLAM offline) e em tempo real (aplicado ao AGV), não foram realizadas medições da pose a cada varredura. Assim, assume-se que as posições verdadeiras são desconhecidas. No entanto, com base na confiança obtida no estudo inicial, na coerência visual observada nos gráficos de localização e mapeamento, considerou-se que o sistema operou conforme esperado. As estimativas de pose mostraram-se plausíveis com os locais de aquisição das varreduras, enquanto a filtragem aplicada ao mapa global resultou em representações mais limpas e confiáveis, atenuando os efeitos de ruído do sensor.

O desenvolvimento prévio em ambiente controlado foi fundamental para ajustes finos e depuração do sistema. A partir disso, o sistema foi adaptado para operar em tempo real, utilizando dados diretos do sensor LiDAR embarcado no AGV e integrando-se à interface gráfica desenvolvida. Os testes demonstraram a capacidade do sistema de mapear progressivamente o ambiente enquanto estimava sua posição, validando sua aplicação prática da proposta.

Embora a taxa de atualização seja moderada (3 s), o desempenho é compatível com aplicações em robôs móveis de baixa velocidade, onde movimentos suaves são preferíveis para garantir segurança operacional e estabilidade nas estimativas. Essa limitação é, inclusive, uma característica comum em sistemas SLAM, os quais tendem a ser sensíveis a variações bruscas de movimento.

Cabe destacar que a avaliação quantitativa da precisão do sistema incremental desenvolvido permanece como uma possibilidade para estudos futuros, por meio da obtenção de poses reais durante a execução. Ainda assim, os resultados obtidos até aqui demonstram um desempenho preciso e promissor para aplicações práticas de mapeamento e localização.

Capítulo 6

Conclusões e Sugestões para Trabalhos Futuros

6.1 Conclusões

Todos os objetivos específicos propostos neste trabalho foram alcançados: a integração do sensor LiDAR 2D ao AGV em laboratório, a avaliação de variantes do método ICP, o desenvolvimento de um sistema de mapeamento e localização em tempo real, e o desenvolvimento de uma interface gráfica interativa.

O estudo comparativo confirmou que o algoritmo ICP com métrica ponto-a-plano, associado à otimização via Gauss-Newton, é a abordagem mais precisa e eficiente dentre as avaliadas, exigindo menos iterações e fornecendo estimativas mais próximas da pose real. Apesar de seu maior custo computacional, os ganhos em robustez e confiabilidade justificam sua aplicação, especialmente em contextos onde a precisão é prioritária.

Com base nessa análise, o sistema SLAM incremental foi desenvolvido, permitindo o mapeamento progressivo e a estimativa contínua da pose do robô a partir dos dados do sensor LiDAR 2D. Mesmo sem medições externas para validação da pose durante os testes em simulação e tempo real com o veículo, os resultados demonstraram coerência visual nas estimativas, com mapas consistentes e compatíveis com os deslocamentos reais observados.

A solução proposta mostrou-se eficaz para ambientes estruturados, validando sua aplicabilidade em cenários reais, inclusive como base para futuras etapas de planejamento de

trajetória e controle de navegação. A arquitetura modular do sistema, com separação clara entre interface e processamento, também facilita a integração com outros sensores ou algoritmos no futuro.

Por outro lado, observou-se que a qualidade das varreduras pode ser impactada negativamente por fatores como obstáculos finos, superfícies reflexivas e movimentos abruptos. Tais limitações são comuns em sistemas baseados apenas em LiDAR e reforçam a importância de se considerar suavidade no deslocamento e possíveis estratégias de fusão sensorial para aumentar a robustez do sistema.

Com os avanços demonstrados neste trabalho, acredita-se que este trabalho contribui para o desenvolvimento de plataformas móveis mais autônomas, precisas e adaptáveis, impulsionando aplicações industriais, acadêmicas e educacionais no campo da robótica móvel.

6.2 Sugestões para trabalhos futuros

Alguns pontos não tratados podem servir como sugestões para futuros estudos nessa área:

- Avaliação quantitativa do sistema SLAM incremental desenvolvido neste trabalho com medições reais de pose durante a aquisição dos dados, a fim de validar a precisão relativa do sistema em operação;
- Adoção de técnicas baseadas em *Occupancy Grid* para uma representação mais eficiente do ambiente, uma vez que esse modelo reduz a complexidade do sistema ao utilizar uma matriz 2D binária que distingue de forma direta células ocupadas e livres.
- Extensão do sistema com a integração de dados de outros sensores, como odometria por *encoders* ou câmeras para fusão sensorial no contexto de SLAM.
- Desenvolvimento de módulos complementares para planejamento de trajetória e controle de navegação baseados no mapa gerado, ampliando a autonomia e aplicabilidade prática da plataforma robótica.

Referências bibliográficas

ABDELAZIZ, Nader; EL-RABBANY, Ahmed. Lidar/visual slam-aided vehicular inertial navigation system for gnss-denied environments. *2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, p. 1–5, 2022. DOI: 10.1109/ICCSPA55860.2022.10019210. <https://ieeexplore.ieee.org/document/10019210>, Último acesso em 31 Jan. 2025.

ASLAM, Muhammad Saad et al. An rplidar based slam equipped with imu for autonomous navigation of wheeled mobile robot. *2020 IEEE 23rd International Multitopic Conference (INMIC)*, p. 1–5, 2020.

BASAVANNA, M. et al. Ros based 3d mapping of an indoor environment using fusion of orbbec astra camera and lidar on turtlebot mobile robot. *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, p. 323–327, 2021.

BESL, P.J.; MCKAY, Neil D. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 14, n. 2, 1992.

BOGOSLAVSKYI, Igor. *Non-linear Least-squares based ICP*. 2025. <https://github.com/niosus/notebooks/blob/master/icp.ipynb>. Acesso em: 31 Jan. 2025.

CAPRIYANI, Desak Made Indira et al. Mecanum-Wheeled Autonomous Mobile Robot for Flexible Manufacturing System. *2024 IEEE International Conference on Advanced Telecommunication and Networking Technologies (ATNT)*, v. 1, p. 1–4, 2024. DOI: 10.1109/ATNT61688.2024.10719179. <https://ieeexplore.ieee.org/document/10719179>, Último acesso em 31 Jan. 2025.

CLOTET, Eduard; PALACíN, Jordi. Slamicp library: Accelerating obstacle detection in mobile robot navigation via outlier monitoring following icp localization. *Sensors*, v. 23, n. 15, 2023.

DIAS, Christian C. *Projeto Baseado em Modelo Aplicado ao Desenvolvimento de um Sistema de Controle para Veículo Autoguiado*. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal de Campina Grande (UFCG), Campina Grande, Brasil, 2020.

FILOTHEOU, Alexandros et al. Correspondenceless scan-to-map-scan matching of 2D panoramic range scans. *Sensors*, v. 18, p. 100288, 2023. DOI: 10.1016/j.array.2023.100288. <https://doi.org/10.1016/j.array.2023.100288>, Último acesso em 31 Jan. 2025.

GRAUMAN, Kristen; LEIBE, Bastian (Ed.). *Computer Vision: A Reference Guide*. [S.l.]: Springer, 2021.

GUAN, Wei; LI, WenTao; REN, Yan. Point cloud registration based on improved ICP algorithm. *2018 Chinese Control And Decision Conference (CCDC)*, p. 1461–1465, 2018. DOI: 10.1109/CCDC.2018.8407357. <https://ieeexplore.ieee.org/document/8407357>, Último acesso em 31 Jan. 2025.

HONDA, Kohei et al. Generalized LOAM: LiDAR Odometry Estimation With Trainable Local Geometric Features. *IEEE Robotics and Automation Letters*, v. 7, p. 12459–12466, 2022. DOI: 10.1109/LRA.2022.3219022. <https://ieeexplore.ieee.org/document/9935115>, Último acesso em 31 Jan. 2025.

HOUSEIN, Alaa; XINGYU, Gao. Simultaneous localization and mapping using differential drive mobile robot under ros. *Journal of Physics: Conference Series*, v. 1820, p. 012015, 03 2021.

HUANG, Shan; HUANG, Hong-Zhong. A Frame-to-Frame Scan Matching Algorithm for 2D Lidar Based on Attention. *Applied Sciences*, v. 12, p. 4341, 2022. DOI: 10.3390/app12094341. <https://www.mdpi.com/2076-3417/12/9/4341>, Último acesso em 31 Jan. 2025.

IBRAGIMOV, Ilmir Z.; AFANASYEV, Ilya M. Comparison of ros-based visual slam methods in homogeneous indoor environment. *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, p. 1–6, 2023. DOI: 10.1109/WPNC.2017.8250081. <https://ieeexplore.ieee.org/document/8250081>, Último acesso em 31 Jan. 2025.

JIA, Guanwei et al. Visual-slam classical framework and key techniques: A review. *Sensors*, v. 22, 2022. DOI: 10.3390/s22124582. <https://www.mdpi.com/1424-8220/22/12/4582>, Último acesso em 31 Jan. 2025.

JUNIOR, José A. N. Barbosa. *Sitonia Iterativa Baseada em Dados da Malha Fechada de Trajetória de um AGV Utilizando Margens de Ganho e de Fase*. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal de Campina Grande (UFCG), Campina Grande, Brasil, 2022.

KATAOKA, Ryosuke et al. ICP-based SLAM Using LiDAR Intensity and Near-infrared Data. *2021 IEEE/SICE International Symposium on System Integration (SII)*, p. 100–104, 2021. DOI: 10.1109/IEEECONF49454.2021.9382647. <https://ieeexplore.ieee.org/document/9382647>, Último acesso em 31 Jan. 2025.

KHAN, Misha Urooj et al. A comparative survey of lidar-slam and lidar based sensor technologies. *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, Karachi, Pakistan, p. 1–8, 2021.

LEE, Daehan; LIM, Hyungtae; HAN, Soohee. GenZ-ICP: Generalizable and Degeneracy-Robust LiDAR Odometry Using an Adaptive Weighting. *IEEE Robotics and Automation Letters*, v. 10, p. 152–159, 2025. DOI: 10.1109/LRA.2024.3498779. <https://ieeexplore.ieee.org/document/10753079>, Último acesso em 31 Jan. 2025.

LI, Zhuoran et al. An Optimization on 2D-SLAM Map Construction Algorithm Based on LiDAR. *Journal of Intelligent and Robotic Systems*, v. 110, p. 1–6, 2024. DOI: 10.1007/s10846-024-02123-1. <https://link.springer.com/article/10.1007/s10846-024-02123-1>, Último acesso em 31 Jan. 2025.

LIU, Jingyun et al. TOF Lidar Development in Autonomous Vehicle. *2018 IEEE 3rd Optoelectronics Global Conference (OGC)*, p. 185–190, 2018. DOI: 10.1109/OGC.2018.8529992. <https://ieeexplore.ieee.org/document/8529992>, Último acesso em 31 Jan. 2025.

LYNCH, Liam et al. Automated ground vehicle (AGV) and sensor technologies - a review. *2018 12th International Conference on Sensing Technology (ICST)*, p. 347–352, 2018.

MENDO, Aina E. Trabalho de Conclusão de Curso, *Investigation into the principles of LiDAR and use in modern electronic systems*. Barcelona, Catalunya: [s.n.], 2022.

NAGLA, Shubham. 2d hector slam of indoor mobile robot using 2d lidar. *2020 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, p. 1–4, 2020. DOI: 10.1109/ICPECTS49113.2020.9336995. <https://ieeexplore.ieee.org/document/9381521>, Último acesso em 31 Jan. 2025.

NIUSUS, Nicolas. *Non-linear Least-squares based ICP*. 2021. Último acesso em 28 nov. 2024. Disponível em: <https://nbviewer.org/github/niosus/notebooks/blob/master/icp.ipynb#Non-linear-Least-squares-based-ICP>.

NOCEDAL, Jorge; WRIGHT, Stephen J. *Numerical Optimization*. [S.l.: s.n.], 2006. ISBN 9780387303031.

PYO, YoonSeok et al. *ROS Robot Programming*. GeumCheon-gu, Seoul, Republic of Korea: ROBOTIS Co.,Ltd., 2017. ISBN: 979-11-962307-1-5.

QIU, Hao et al. Research on unmanned vehicle obstacle avoidance technology based on lidar and depth camera fusion. *Applied Mathematics and Nonlinear Sciences*, v. 9, 2024. DOI: 10.2478/amns.2023.2.00575. <https://www.researchgate.net/publication/374616864_Research_on_unmanned_vehicle_obstacle_avoidance_technology_based_on_LIDAR_and_depth_camera_fusion>, Último acesso em 31 Jan. 2025.

SICILIANO, Bruno; KHATIBL, Oussama. *Handbook of Robotics*. [S.l.: s.n.], 2016. ISBN 978-3-319-32550-7.

SILVA, Rita Tomé et al. AGVs vs AMRs: A Comparative Study of Fleet Performance and Flexibility. *2024 7th Iberian Robotics Conference (ROBOT)*, v. 1, p. 1–4, 2024. DOI: 10.1109/ATNT61688.2024.10719179. <https://ieeexplore.ieee.org/document/10719179>, Último acesso em 31 Jan. 2025.

SLAMTEC. *RPLIDAR A1: Introduction and Datasheet*. 2020. Rev. 3.0.

SOULIMAN, Aya. *Mapping and Localization for Indoor Mobile Robot Based on RGB-D Camera*. Tese (Doutorado), Agosto 2017.

STACHNISS, Cyrill. *Robotic Mapping and Exploration*. [S.l.: s.n.], 2009. v. 55. ISBN 978-3-642-01096-5.

SUTTHI, Siritakorn et al. Designing of Delta Manipulator as Human-Robot Interaction for Collaborative Mobile Robot. *2023 62nd Annual Conference of the Society of Instrument and Control Engineers (SICE)*, p. 204–209, 2023. DOI: 10.23919/SICE59929.2023.10354095. <https://ieeexplore.ieee.org/document/10354095>, Último acesso em 31 Jan. 2025.

TESSARINI, Geraldo; SALTORATO, Patrícia. Impactos da indústria 4.0 na organização do trabalho: uma revisão sistemática da literatura. *Revista Produção Online*, v. 18, p. 743–769, 2018.

VARGHESE, Glace et al. Multi-robot system for mapping and localization. *2023 8th International Conference on Robotics and Automation Engineering (ICRAE)*, Karachi, Pakistan, p. 79–84, 2023.

VIEIRA, Frederico Carvalho. *Controle Dinâmico de Robôs Móveis com Acionamento Diferencial*. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Rio Grande do Norte (UFRN), Natal, Brasil, Fevereiro 2005.

VO, Trung-Kien; HA, Manh-Tuan. Implement ICP Algorithm for SLAM Using Relative Motion Threshold and Robust Kernel. *2024 IEEE 11th International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*, p. 1–6, 2024. DOI: 10.1109/ICCC62278.2024.10582930. <https://ieeexplore.ieee.org/document/10582930>, Último acesso em 31 Jan. 2025.

WANG, Qiang; MIMA, Kazuhiro; TOBITA, Kazuteru. Semantic mapping based on orb-slam and yolo in indoor scenes. *2023 IEEE 5th Eurasia Conference on IOT, Communication and Engineering (ECICE)*, p. 956–959, 2023. DOI: 10.1109/ECICE59523.2023.10383074. <https://ieeexplore.ieee.org/document/10383074>, Último acesso em 31 Jan. 2025.

XU, Xiaobin et al. A Review of Multi-Sensor Fusion SLAM Systems Based on 3D LIDAR. *Remote Sensing*, v. 14, p. 1–27, 2022.

XUEXI, Zhang et al. Slam algorithm analysis of mobile robot based on lidar. *2019 Chinese Control Conference (CCC)*, p. 4739–4745, 2019.

YAGFAROV, Rauf; IVANOU, Mikhail; AFANASYEV, Ilya. Map comparison of lidar-based 2d slam algorithms using precise ground truth. *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Singapore, p. 18–21, 2018.

YAMADA, Viviane Y.; MARTINS, Luís M. Indústria 4.0: um comparativo da indústria brasileira perante o mundo. *Revista Terra e Cultura: Cadernos de Ensino e Pesquisa*, v. 34, p. 95–109, 2019.

YAZDANPOUR, Mahdi; FAN, Guoliang; SHENG, Weihua. Real-time volumetric reconstruction of Manhattan indoor scenes from depth sequences. *2017 IEEE Visual Communications and Image Processing (VCIP)*, p. 1–4, 2017. DOI: 10.1109/VCIP.2017.8305073. <https://ieeexplore.ieee.org/document/8305073>, Último acesso em 31 Jan. 2025.

YONG, Zhang et al. An Autonomous Navigation Strategy Based on Improved Hector SLAM With Dynamic Weighted A* Algorithm. *IEEE Access*, v. 11, p. 79553–79571, 2023. DOI: 10.1109/ACCESS.2023.3299293. <https://ieeexplore.ieee.org/document/10195970>, Último acesso em 31 Jan. 2025.

YOUJI, Zhu et al. A SLAM Method Based on LOAM for Ground Vehicles in the Flat Ground. *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, p. 546–551, 2019. DOI: 10.1109/ICPHYS.2019.8780378. <https://ieeexplore.ieee.org/document/8780378>, Último acesso em 31 Jan. 2025.

ZHANG, Xinyu et al. Research on visual SLAM technology of industrial inspection robot based on RGB-D. *2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS)*, p. 51–55, 2023. DOI: 10.1109/ISCTIS58954.2023.10213089. <https://ieeexplore.ieee.org/document/10213089>, Último acesso em 31 Jan. 2025.

ZHANG, Zhengyou. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, v. 13, n. 2, 1994.

ZOU, Qin et al. A comparative analysis of lidar slam-based indoor navigation for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, v. 23, p. 6907–6921, 2022. DOI: 10.1109/TITS.2021.3063477. <https://ieeexplore.ieee.org/document/9381521>, Último acesso em 31 Jan. 2025.