UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FELIPE JORDÃO FREITAS DE MENDONÇA

PROPOSTA DE ALGORITMO DE LOCALIZAÇÃO POR MEIO DE ODOMETRIA LIDAR

PATO BRANCO

2023

FELIPE JORDÃO FREITAS DE MENDONÇA

PROPOSTA DE ALGORITMO DE LOCALIZAÇÃO POR MEIO DE ODOMETRIA LIDAR

Proposal for Implementation of LiDAR Odometry Algorithm

Trabalho de conclusão de curso apresentada como requisito para obtenção do Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientadora: Profa. Dra. Luciene de Oliveira Marin
Coorientador: Prof. Dr. Jeferson José de Lima

PATO BRANCO

2023

FELIPE JORDÃO FREITAS DE MENDONÇA

PROPOSTA DE ALGORITMO DE LOCALIZAÇÃO POR MEIO DE ODOMETRIA LIDAR

Trabalho de conclusão de curso apresentada como requisito para obtenção do Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de Aprovação: 01 de dezembro de 2023.

Prof. Dr. Jeferson José de Lima
Universidade Tecnológica Federal do Paraná

Profa. Dra. Luciene de Oliveira Marin
Universidade Tecnológica Federal do Paraná

Prof. Dr. Jefferson Tales Oliva
Universidade Tecnológica Federal do Paraná

Prof. Dr. César Rafael Claure Torrico
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2023

Dedico este trabalho a minha família e aos meus amigos, pelos momentos de ausência.

“ Para pequenas criaturas como nós, a vastidão é suportável somente através do amor“

– Carl Sagan

AGRADECIMENTOS

O presente trabalho não poderia ser finalizado sem a ajuda de diversas pessoas e as instituições UTFPR e AutenPRO, às quais presto meus sinceros agradecimentos. Certamente, esses parágrafos não abrangem todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre estas palavras, mas elas podem estar certas que fazem parte do meu pensamento e tem minha gratidão.

Aos meus primeiros amores, Érika Marília e Janner Augusto, pelo carinho, incentivo e total apoio em todos os momentos da minha vida.

Ao meu orientador Prof. Dr. Jeferson Lima e coorientadora Prof. Dra. Luciene Marin, que me mostraram os caminhos a serem seguidos e pela confiança depositada.

A todos os professores e colegas do curso, que ajudaram de forma direta e indireta na conclusão deste curso de graduação. Principalmente ao meu colega Igo Monteiro e minha parceira Jaine Sehnen, que me auxiliaram nos momentos em que não me sentia capaz.

A todos os meus colegas de trabalho, que me incentivaram diariamente nesta batalha.

A todos os demais que de alguma forma contribuíram para meu crescimento pessoal e profissional.

RESUMO

Nas últimas décadas, nota-se um crescente interesse no desenvolvimento de veículos autônomos para a realização de tarefas domésticas, militares, de exploração espacial, entre outras. A robotização de atividades complexas as quais envolvem a navegação em ambientes sem a intervenção humana implica em habilitar o agente da capacidade de adquirir modelos do cenário e estimar sua localização nesta cena. Desta forma, entende-se que uma das necessidades fundamentais de um veículo autônomo é identificar, ao longo do tempo, sua localização e possuir uma interpretação válida do ambiente no qual está empregado. Esta técnica é chamada de SLAM (*Simultaneous Location and Mapping*; Localização e mapeamento simultâneos).

No contexto da visão computacional e robótica móvel, o algoritmo ICP (*Iterative Closest Point*) é amplamente utilizado em aplicações de localização e mapeamento. Essencialmente, a técnica ICP é uma solução para o processamento de dados do tipo nuvem de pontos. Estes dados usualmente são provenientes de sensores laser, radares ou câmeras RGB-D e podem ser representados por meio de coordenadas cartesianas (pontos 3D ou 2D).

O presente trabalho, propõe a implementação do algoritmo ICP para processar os dados de um sensor de escaneamento laser (LiDAR; *Light Detection and Ranging*). Salienta-se que esta abordagem pode ser utilizada para a localização e para o mapeamento, porém, o escopo desta pesquisa é o módulo da localização. Utiliza-se as métricas RMSE (*Root Mean Square Error*; Raiz do erro quadrático médio) e ATE (*Absolute Trajectory Error*) para avaliar a convergência do sistema e o erro das estimações dos parâmetros da localização respectivamente. Para validar a trajetória, foi utilizado o módulo de localização ArUco da biblioteca OpenCV de processamento gráfico, enquanto o algoritmo ICP foi desenvolvido em C++ objetivando o processamento em sistemas embarcados. Adicionalmente, para verificar a possibilidade de embarcar o processamento do algoritmo, avalia-se caso os tempos de execução são condizentes com os parâmetros de engenharia para uma aplicação de localização.

Apesar da implementação desenvolvida obter um bom alinhamento e uma estimativa aceitável da componente angular da localização (média de erro de 6°), as estimativas da translação apresentam um erro significativo o qual torna a trajetória gerada pela abordagem não representativa do movimento real do agente. Entretanto, estes erros são condizentes com resultados de uma abordagem simples de implementação do algoritmo ICP, a qual não conta com filtros probabilísticos ou fusão sensorial para corrigir as estimações. Em adição, foi possível migrar o sistema para o processamento embarcado, porém o tempo de execução não satisfaz as condições de execução em tempo real.

Palavras-chave: SLAM (Localização e mapeamento Simultâneos); Sensor LiDAR; ICP (Iterative Closest Point).

ABSTRACT

In recent decades, there has been a growing interest in the development of autonomous vehicles for various tasks, including domestic, military, and space exploration. The roboticization of complex activities involving navigation in human intervention-free environments implies enabling the agent to acquire models of the scenario and estimate its location in this scene. Thus, it is understood that one of the fundamental needs of an autonomous vehicle is to identify, over time, its location and possess a valid interpretation of the environment in which it is deployed. This technique is known as SLAM (Simultaneous Location and Mapping).
In the context of computer vision and mobile robotics, the Iterative Closest Point (ICP) algorithm is widely used in localization and mapping applications. Essentially, this technique is a solution for processing point cloud data, typically obtained from laser sensors, radars, or RGB-D cameras, and can be represented by Cartesian coordinates (3D or 2D points).
This work proposes the implementation of the ICP algorithm to process data from a laser scanning sensor (LiDAR; Light Detection and Ranging). It is worth noting that this approach can be used for both localization and mapping; however, the focus of this dissertation is on the localization module. The RMSE (Root Mean Square Error) and ATE (Absolute Trajectory Error) metrics are used to assess the system's convergence and the error of location parameter estimates, respectively. To validate the trajectory, the ArUco localization module from the OpenCV graphics processing library was used, while the ICP algorithm was developed in C++, aiming at processing on embedded systems. Finally, to assess the possibility of embedding the algorithm's processing, runtime considerations are evaluated against engineering parameters for a localization application. Although the implemented solution achieves good alignment and an acceptable estimate of the angular component of the location (average error of 6°), translational estimates exhibit a significant error, making the trajectory generated by the approach non-representative of the agent's actual movement. However, these errors are consistent with the results of a simple implementation approach of the ICP algorithm, which lacks probabilistic filters to correct translational estimates. In addition, it was possible to migrate the system to embedded processing, but the runtime does not satisfy the necessary requirements for a localization system.

Keywords: SLAM (Simultaneous Location and Mapping); LiDAR sensor; ICP.

LISTA DE ALGORITMOS

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

Siglas

ATE	*Absolute Trajectory Error*
GPS	*Global Positioning System*
ICP	*Iterative Closest Point*
RMSE	*Root Mean Square Error*
SLAM	*Simultaneous Localization And Mapping*
UTFPR	Universidade Tecnológica Federal do Paraná

LISTA DE SÍMBOLOS

Notações

{A}	Sistema de Coordenadas A	$\left[[x_A,y_A,z_A]^T\right]$
{O}	Sistema de Coordenadas O	$\left[[x_O,y_O,z_O]^T\right]$
{R}	Sistema de Coordenadas R	$\left[[x_R,y_R,z_R]^T\right]$

SUMÁRIO

1 INTRODUÇÃO

No contexto da robótica móvel e visão computacional, o problema SLAM (*Simultaneous Location And Mapping* — Localização e mapeamento simultâneos) é uma das temáticas de pesquisa cuja essência é atingir autonomia robusta e eficaz de uma máquina móvel. Essa habilidade é desejada para várias aplicações e já foi atribuída à automóveis autônomos, transporte e monitoramento industrial não tripulado, robôs domésticos, robôs para exploração espacial (Ilci; Toth, 2020; Cota, 2019; Geromichalos *et al.*, 2020; Li *et al.*, 2021), e outras aplicações nas quais almeja-se uma certa independência na mobilidade do dispositivo.

Entende-se que uma das necessidades fundamentais de um veículo autônomo é identificar, ao longo do tempo, sua localização e possuir uma interpretação válida do ambiente no qual está empregado (Guizilini, 2008) (Dissanayake *et al.*, 2001). Os trabalhos relacionados ao tema SLAM integram a modelagem cinemática e sensorial em uma malha de controle cujo objetivo é suavizar e processar os dados coletados pelo conjunto de sensores acoplados ao robô. É comum o uso de filtros Bayesianos para processar essa informação (Dellaert *et al.*, 1999), pois cada sensor acumula erros ao longo do tempo devido às limitações físicas do aparato. Deste modo, as pesquisas da área apresentam discussões sobre as técnicas de processamento sensorial mais adequadas para cada dispositivo móvel (Cota, 2019).

Entre as técnicas SLAM, é comum a utilização de sensores como câmeras, sonares, sensores LiDAR (*Light Detection and Ranging* — medidores de distância por meio de lasers), IMU (*Inertial Mesurament Unit* — giroscópios e acelerômetros), e também *encoders* de rodas — medidores da rotação das rodas. A configuração de sensores dentro da máquina atua em conjunto para auxiliar na estimação da odometria — nomenclatura para modelo matemático o qual obtém a localização do agente robótico. No trabalho de Cota (2019), é apresentada uma versão do EspeleoRobô, um robô com rodas a qual utiliza uma câmera RGB-D (câmera com senso de profundidade), *encoders* de rodas, um sensor LiDAR e um sensor de unidade inercial (IMU). Essa disposição sensorial permite a correção de erros acumulados em um sensor por meio dos dados de outro sensor. Caso o agente esteja em um local com pouca luz, os dados insignificantes da câmera podem ser compensados pelo sensor LiDAR, o qual consegue operar nessa condição.

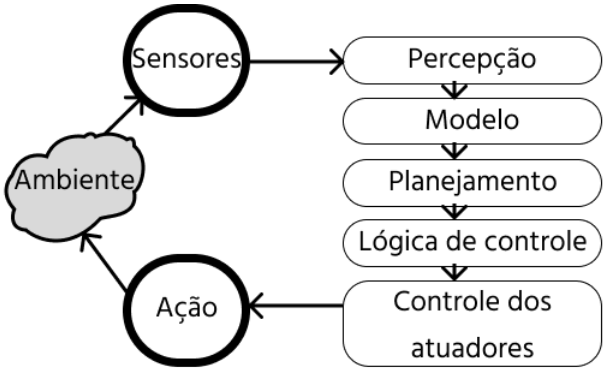
Diversos requisitos precisam ser cumpridos para garantir a robustez e autonomia do veículo móvel. Um dos requisitos é a questão da localização: o robô deve conhecer sua posição e orientação a cada instante num sistema de coordenadas absoluto. A grande problemática dessa abordagem é atingir precisão nas medições de pose, pois a informação dos aparatos sensoriais é

inevitavelmente imprecisa, e caso não processada corretamente, acumula erros os quais distorcem a percepção do robô em relação à realidade do cenário. Consequentemente, o problema da localização e de mapeamento estão inerentemente conectados, pois construir um mapa preciso do ambiente necessita de uma estimação justa da localização do dispositivo, e paralelamente, para melhorar a estimativa da sua posição, precisa-se de um mapa da cena (Cruz Júnior *et al.*, 2021).

Naturalmente, a navegação autônoma é uma habilidade imprescindível para os robôs móveis, porém, requer a integração de uma variedade de técnicas e processos. De acordo com a literatura de Dudek e Jenkin (2010), os principais módulos de um sistema de navegação autônoma são: percepção, planejamento e controle. Dentre os módulos citados, o problema SLAM se encaixa no quadro da percepção, consequentemente, é o processo responsável por interpretar os dados sensoriais.

Tais conceitos podem ser observados em conjuntura e totalidade na malha de controle da Figura 1. A malha em questão é uma das abordagens propostas por Dudek e Jenkin (2010) para o controle de um robô em um ambiente estático e desconhecido. Deste modo, a partir dos sensores e suas leituras, o módulo de percepção organiza a informação coletada. Em seguida, o "Modelo" é responsável por acumular a informação sensorial eficientemente, de forma a construir uma representação pertinente do cenário. A partir dos dois módulos citados, o módulo de planejamento os avalia e decide sobre qual a melhor ação para o instante em questão e para os próximos instantes. Com isto, a lógica de controle traduz o plano para a real movimentação do agente robótico. Por fim, o controle dos atuadores consiste na ativação em baixo nível dos componentes responsáveis pelo movimento do robô. Essa arquitetura de controle é conhecida também pela terminologia SMPA (*Sense-Model-Plan-Act*), cujo significado está centrado no conceito do ciclo sensoriamento, modelo matemático, plano e atuação.

Figura 1 – Malha de controle para autonomia móvel proposta por Dudek e Jenkin (2010)

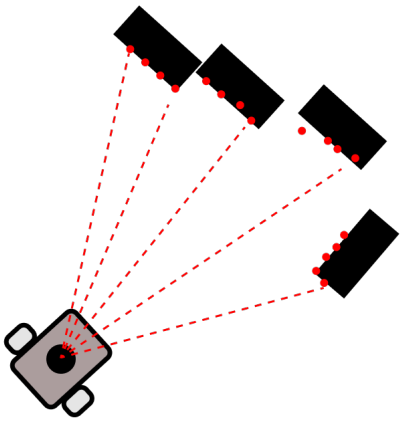


Fonte: Autoria própria (2023).

Nesse âmbito, define-se como escopo do presente trabalho o módulo de percepção. É importante notar que o processamento realizado pelo algoritmo SLAM, numa aplicação integral de navegação autônoma, é usado para realimentar o controle de navegação. Uma abordagem comum para o SLAM é a integração do processamento de imagens, porém, em condições de iluminação precária, as estimações das câmeras se tornam inutilizáveis. Outra solução mais adequada é a aplicação de técnicas SLAM baseadas em sensores LiDAR, os quais possuem medidas precisas de distância e são indiferentes às condições de iluminação (Cruz Júnior *et al.*, 2021).

Os dados provenientes de sensores LiDAR são comumente chamados de nuvens de pontos, pois cada "quadro" de dados do sensor representa o conjunto de pontos do escaneamento atual, o qual assemelha-se a uma nuvem de pontos. Na Figura 2, pode-se visualizar um exemplo de como representa-se o conjunto de pontos referentes ao escaneamento atual do robô. Desta maneira, a técnica SLAM LiDAR aborda um problema de alinhamento de nuvens de pontos para estimar o deslocamento do robô (Cota, 2019).

Figura 2 – Exemplo do escaneamento LiDAR 2D



Fonte: Autoria própria (2023).

Tradicionalmente, a abordagem escolhida para solucionar o alinhamento dos conjuntos de pontos é o algoritmo ICP (*Iterative Closest Point*) (Besl; McKay, 1992). Dadas essas premissas, o vigente trabalho tem como foco a implementação de uma técnica LiDAR SLAM 2D (técnica para localização baseada em um sensor LiDAR de escaneamento 2D) e avaliar a viabilidade de embarcar a solução em um microcontrolador de baixo custo. É importante mencionar que apesar do algoritmo ICP poder ser utilizado para a geração de mapas do ambiente, a geração de mapas não faz parte do escopo desta monografia.

1.1 Objetivos

Descrever, propor e avaliar um algoritmo de odometria baseado em sensor LiDAR e analisar sua viabilidade e desempenho em um microcontrolador de baixo custo.

1.1.1 Objetivos Específicos

- Propor uma estrutura de solução para o problema da localização e justificar sua racionalidade;
- Implementar, avaliar e interpretar resultados de um algoritmo de Odometria LiDAR;
- Propor e descrever a métrica para avaliar a precisão das estimativas de localização da solução implementada em relação a posição real do dispositivo;
- Verificar a viabilidade do algoritmo desenvolvido em sistemas embarcados (STM32F407).

1.2 Justificativa

Desde o momento da aprovação da proposta deste trabalho, algumas alterações de projeto foram necessárias para afunilar o escopo da pesquisa e direcionar a implementação para uma aplicação real. Em um primeiro momento, foi proposto a utilização da plataforma robótica ROS (*Robotic Operating System*) (Laboratory, 2018) e seus recursos (*gazebo*, *turtlebot3*, pacotes ROS, entre outros) para realizar a simulação da solução implementada e avaliar os resultados, entretanto com a possibilidade de desenvolver a solução proposta, adequando-a ao *hardware* presente no ambiente profissional, logo, optou-se por direcionar o desenvolvimento do trabalho ao contexto dos objetivos da empresa AutenPRO.

O sensor LiDAR e outros equipamentos utilizados foram fornecidos pela empresa AutenPRO, a qual se destaca no cenário da automação agrícola, com aplicações únicas para a otimização e redução de insumos da produção agrícola. Apesar de possuir menos de 50 funcionários, atende clientes de estados como São Paulo, Minas Gerais, Rio Grande do Sul, Paraná, Santa Catarina, entre outros. A empresa já utiliza sensores LiDAR para as soluções consolidadas no mercado, este fato incentivou a exploração de novas tecnologias envolvendo o mesmo princípio de sensoriamento laser. Desta forma, o presente trabalho foi incorporado como um projeto da instituição, visando futuramente a inserção no mercado da robótica móvel.

Essencialmente, foram alteradas a plataforma de desenvolvimento, a proposta do experimento e a ferramenta para determinar a métrica do erro. Anteriormente, foi apresentado que a implementação e todos os dados para a avaliação do desempenho da solução seriam provenientes da simulação feita na plataforma ROS. Entretanto, retirou-se a simulação no ambiente virtual ROS, e foi realizado um experimento com os dispositivos físicos para verificar o funcionamento da implementação desenvolvida.

É recorrente na literatura sobre o problema SLAM propostas de otimização e alternativas para diminuir o custo computacional da abordagem como em Low (2004), Tiar, Lakrouf e Azouaoui (2015), Cao *et al.* (2021) e Labbe e Michaud (2014). Porém, não é usual encontrar implementações de processamento nuvem de pontos em sistemas embarcados de baixo custo como em Kang *et al.* (2018), apesar de ser comum a utilização de placas de desenvolvimento como a RaspBerry Pi, atualmente, esta não é uma alternativa tão acessível quanto o microcontrolador STM32F407. Logo, pela disponibilidade do dispositivo citado (STM32F407), este foi elegido para avaliar as limitações de um algoritmo de localização em um sistema embarcado.

A mudança citada acima foi motivada pela intenção da empresa em desenvolver um robô móvel de baixo custo semelhante ao proposto por Shan Huang e HongZhong Huang (2022) (soluções as quais serão apresentadas no Capítulo 2). Logo, determinou-se que o primeiro passo para este projeto é o módulo de percepção. Para isto, é necessário eleger o conjunto de sensores do robô, modelar o processamento destes sensores e avaliar se o funcionamento do sistema atende às necessidades de projeto (parâmetros de engenharia). Desta forma, o presente trabalho foca no processamento dos dados provenientes de um sensor LiDAR planar, e como usá-los para estimar a localização do agente robótico.

1.3 Estrutura do trabalho

O presente trabalho possui mais 4 Capítulos. No Capítulo 2 é apresentada a fundamentação teórica publicada sobre robôs móveis, técnicas de SLAM com foco em dados de sensores LiDAR e a estrutura básica dos algoritmos LiDAR SLAM.

O Capítulo 3 discorre sobre os componentes de hardware e software utilizados para o desenvolvimento do trabalho, bem como a metodologia para a avaliação dos resultados.

O Capítulo 4 apresenta os resultados das simulações em PC e no microcontrolador, a comparação entre a trajetória estimada e a real e comparativos do desempenho do algoritmo nas diferentes plataformas.

No Capítulo 5 estará a análise e discussão sobre a implementação e apresenta-se propostas para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este Capítulo apresenta trabalhos relacionados e sensores aplicados nas soluções SLAM. Em seguida, descreve os fundamentos teóricos utilizados para representar a movimentação de robôs móveis, bem como relata as estratégias e passos para compreender as técnicas de localização e mapeamento baseadas em sensores LiDAR.

2.1 A problemática SLAM

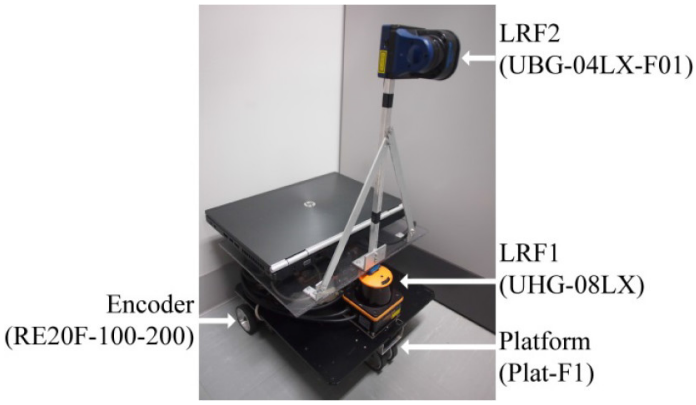
Essencialmente, SLAM é uma abordagem para navegação autônoma baseada em acoplar ao agente robótico a habilidade de identificar sua posição atual e simultaneamente construir mapas do ambiente no qual está inserido. A partir de dados fornecidos por diversos sensores como câmeras, radares, acelerômetros, GPS, um algoritmo SLAM estima a localização do agente, e com uma localização pertinente pode-se construir o mapa representativo do ambiente. A escolha de quais sensores serão acoplados ao robô define as entradas para a abordagem SLAM, e sua saída é o conjunto de parâmetros que descrevem a posição do dispositivo (Gomes, 2015).

Para um veículo autônomo locomover-se com precisão em um ambiente desconhecido é necessário abordar o problema da localização; conhecer a sua posição a cada instante de tempo em um sistema absoluto (inercial) de coordenadas. De forma semelhante, o mapeamento também é essencial, e engloba a habilidade de reconhecer e distribuir as estruturas ao redor do agente. Deste modo, um algoritmo SLAM capacita o dispositivo com a construção de um mapa da cena local e a estimação da própria posição e orientação a partir das informações sensoriais (Dudek; Jenkin, 2010).

Entretanto, existe uma dificuldade intrínseca em conciliar as medições de localização com a posição verdadeira do dispositivo após longos períodos de navegação. O desafio origina-se na imprecisão dos sensores de medidas de odometria (ex: sonares, lasers e câmeras) devido à limitações físicas e imprevisibilidades do sistema. De maneira que, os dados obtidos a partir desses sensores acumulam erros, os quais, se não tratados, delimitam uma clara diferença entre a realidade e a percepção do robô (Guizilini, 2008; Yamauchi; Schultz; Adams, 1998). Essa dificuldade toma a nomenclatura de problemática SLAM. Visto que necessita-se de resultados precisos de localização para a construção de um mapa coerente. Faz-se imprescindível abordar ambos os problemas simultaneamente, eliminando sistematicamente os erros sensoriais antes que acumulem (Guizilini, 2008).

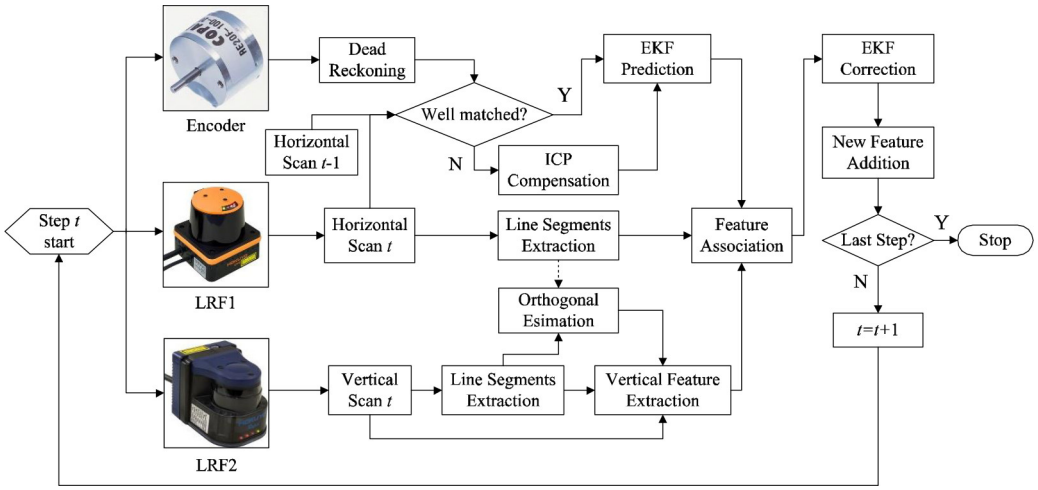
Na solução desenvolvida por Lv *et al.* (2015) foram utilizados dois *encoders* de rodas (capazes de medir a velocidade das rodas) e dois sensores LiDAR (UHG-08LX e UBG-04LX-F01), cujo escaneamento é bidimensional com uma área de detecção de 270° e 240° respectivamente. A arquitetura do dispositivo proposto dispõe os dois sensores laser de forma que um é utilizado para fazer a amostragem horizontal do ambiente enquanto o outro performa o escaneamento vertical para detectar desníveis e escadas. Na Figura 3, observa-se o robô móvel utilizados para os experimentos do trabalho de Lv *et al.* (2015) e sua configuração de sensores.

Figura 3 – Robô móvel e sua disposição de sensores



Fonte: (Lv *et al.*, **2015**).

Figura 4 – Fluxograma da arquitetura da técnica SLAM de Lv *et al.* (2015)



Fonte: (Lv *et al.*, **2015**).

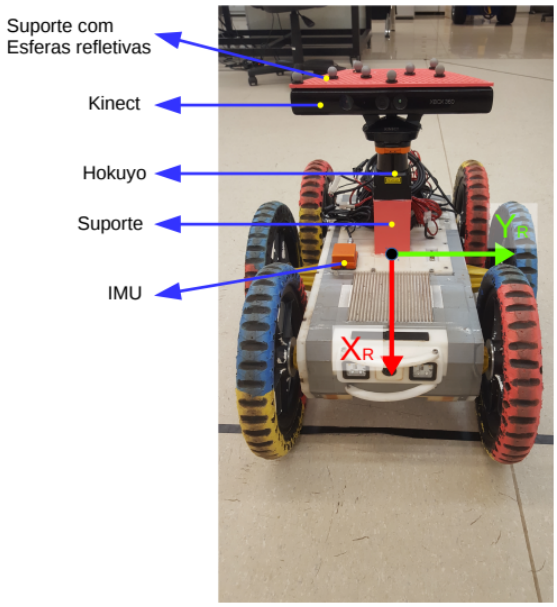
A abordagem dos autores do sistema mencionado acima pode ser dividida em três processos, cada um referente aos dados do respectivo sensor. A etapa *dead reckoning* consiste em capturar os dados dos *encoders* e estimar a transformação que deve ser aplicada ao robô entre o instante $(t-1)$ e (t). Após esta estimativa, aplica-se a transformação encontrada no conjunto de pontos escaneados em $(t-1)$ e avalia-se a qualidade do passo *dead reckoning* (caso haja uma boa sobreposição de pontos). A partir da avaliação citada, decide-se caso seja necessário executar

o Algoritmo ICP para compensar o erro da estimação pelos *encoders*. Por final, é utilizado o filtro EKF (*Extended Kalman Filter* — Filtro Estendido de Kalman) na fase/etapa de predição do próximo estado a partir do atual (Lv *et al.*, 2015).

Mutualmente, as características extraídas (linhas e desníveis) dos sensores de leituras vertical e horizontal são transportadas para entrada da etapa do filtro EKF responsável pela associação das características e correção do erro. Todas essas etapas podem ser visualizadas na Figura 4.

No trabalho de Cota (2019) é utilizado como agente robótico o EspeleoRobô (Figura 5), o qual conta com uma câmera RGB-D (*Kinect*), *encoders* de rodas, um sensor LiDAR (Hokuyo) e um sensor de unidade inercial (IMU). Para a validação da localização desse experimento, utilizou-se 8 câmeras Vicon posicionadas no ambiente de atuação. Essas câmeras identificam reflexos luminosos em esferas reflexivas, desse modo, por meio de triangulação, são capazes de gerar dados de posição e orientação de objetos formados por essas esferas, com acurácia de 1mm.

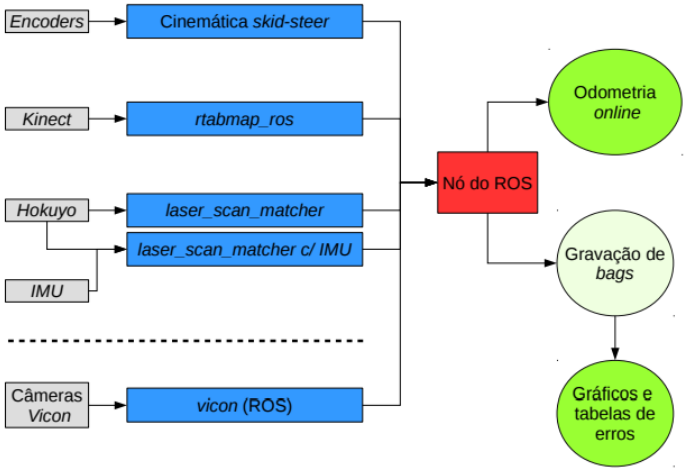
Figura 5 – EspeleoRobô e sua disposição de sensores e suportes



Fonte: (Cota, **2019**).

A abordagem para o experimento de Cota (2019) é evidenciada na Figura 6. Semelhante ao trabalho de Lv *et al.* (2015), cada etapa do sistema lida com os dados do respectivo sensor. Neste arcabouço, os dados dos *encoders* é processado pelo módulo "Cinemática skid-steer" o qual estima a transformação que deve ser aplicada ao robô para localizá-lo, a mesma ideia é aplicada aos dados dos outros sensores (um módulo para cada sensor e um módulo que utiliza dados de dois sensores). Cada módulo estima a transformação de acordo com os dados sensoriais da entrada.

Figura 6 – Fluxograma da arquitetura da técnica SLAM de Cota (2019)



Fonte: (Cota, **2019**).

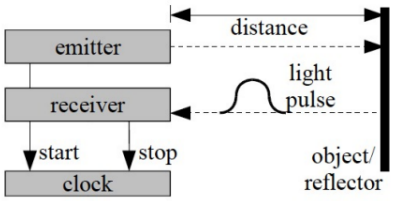
2.1.1 Sensores aplicados ao problema SLAM

Cada projeto robótico utiliza da integração de diferentes sensores para calcular o deslocamento do dispositivo. Nesta seção são apresentados 3 sensores e suas aplicações no problema SLAM.

2.1.1.1 Sensor LiDAR

O sensor LiDAR é composto essencialmente pelo módulo de controle, um espelho e um transmissor/receptor (Braga, 2016). Em suma, o aparato é capaz de emitir e captar pulsos de laser para realizar a medição do intervalo de tempo associado a radiação de retorno da onda emitida; a terminologia dada à tal estratégia é: cálculo *time-of-flight* (Figura 7). Deste modo, é possível computar a distância até o ponto de reflexão. Logo, esse valor pode ser associado à informações de posicionamento e organizado com dados de um sistema de coordenadas mais geral, tal como o GPS (*Global Positioning System*) (Khan *et al.*, 2022).

Figura 7 – Esquema de funcionamento simplificado por meio do princípio *time-of-flight* de um sensor LiDAR



Fonte: (Wallhoff *et al.*, **2007**).

O funcionamento do sensor consiste na emissão de um pulso laser, com alta frequência

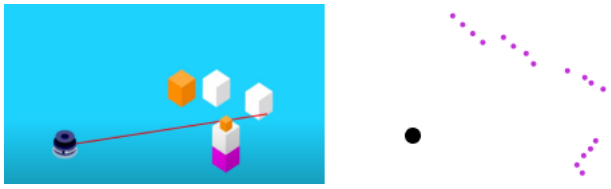
de repetição. O módulo de controle é responsável pelo posicionamento do espelho e pela contagem temporal, garantindo que o pulso seja medido corretamente e assimilado com a distância até o ponto de reflexão (Braga, 2016; Cota, 2019).

De forma simplificada, a distância (d) pode ser descrita em função da velocidade da luz (c) e do intervalo de tempo entre a emissão e detecção do retorno da onda (Equação (1)) (Cota, 2019).

$$d = \frac{c\Delta t}{2} \tag{1}$$

Na implementação realizada nesse trabalho foi utilizado o sensor LiDAR RPLiDAR A1M8, o qual será introduzido com mais profundidade no decorrer deste trabalho. O funcionamento básico do sensor citado pode ser observado na Figura 8, o qual, a partir de medições em vários ângulos, consegue retornar um conjunto de pontos representando contornos do ambiente de atuação.

Figura 8 – Funcionamento básico de um sensor LiDAR 2D



Fonte: Autoria própria (2023).

2.1.1.2 Encoders de rodas

Os *encoders* são aparelhos eletromecânicos capazes de mensurar movimentos rotacionais e gerar um impulso elétrico proporcional ao movimento, o qual é usado como entrada para o modelo matemático da localização. A partir destas medições o sensor pode auxiliar no cálculo de grandezas como velocidade, aceleração, rotação, ângulos, distâncias, entre outras. As aplicações do dispositivo no mercado pode variar de um controle de velocidade de esteira ao controle de posição de um braço robótico (Cota, 2019).

Tradicionalmente, são encontrados *encoders* de dois tipos: incrementais e absolutos. O absoluto permite obter a posição absoluta da roda e é frequentemente utilizado em braços robóticos para localizar o ponto de atuação do manipulador (garra), enquanto o incremental aponta a posição angular relativa ao momento no qual foi energizado, por isso, é comum nas aplicações de robótica móvel para medir a velocidade das rodas.

2.1.1.3 Câmeras RGB

Câmeras RGB desempenham um papel crucial na robótica móvel, desencadeando avanços significativos em técnicas como SLAM. O termo "RGB"refere-se aos três canais de cores principais: vermelho (Red), verde (Green) e azul (Blue), presentes em imagens capturadas por essas câmeras (Dudek; Jenkin, 2010).

No contexto da técnica SLAM, as câmeras RGB desempenham um papel vital na obtenção de dados visuais necessários para a construção de mapas do ambiente e para a localização simultânea do robô. Ao analisar as características visuais das imagens, o SLAM pode extrair informações sobre a geometria e estrutura do ambiente, bem como estimar a posição e orientação do robô em tempo real (Khan *et al.*, 2022).

A informação visual proveniente das câmeras RGB é muitas vezes combinada com dados de outros sensores, como giroscópios, acelerômetros ou sensores de profundidade, para aprimorar a precisão do SLAM. Essa abordagem multimodal permite que os robôs construam mapas mais detalhados e precisos, além de melhorar a capacidade de navegação em ambientes desafiadores (Khan *et al.*, 2022).

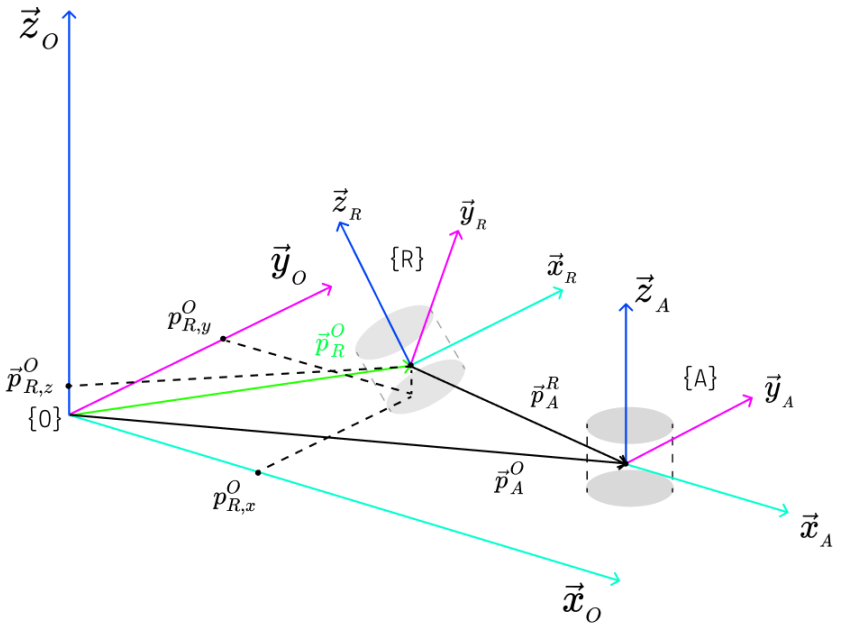
2.2 Cinemática de corpo rígido

Um corpo rígido é definido como um conjunto de moléculas agrupadas as quais possuem distâncias fixas entre si. Deste modo, estuda-se a movimentação do objeto considerando um único ponto, costumeiramente denominado como o centro de massa inercial. Para simplificar a interpretação dos modelos matemáticos utiliza-se o centro de massa do corpo para estudar a movimentação do dispositivo. No âmbito da robótica móvel, a cinemática é descrita como a determinação da posição e orientação do agente em função do deslocamento das rodas (Cruz Júnior *et al.*, 2021). Também é utilizado o termo "pose" para referenciar o conjunto de parâmetros: posição e orientação.

Na intenção de montar o modelo de posições de um objeto, torna-se imprescindível estabelecer um sistema de coordenadas cartesiano global inercial, $\{O\}$, e atribuir um sistema de coordenadas local acoplado ao corpo, $\{R\}$. Considera-se que a origem do sistema $\{R\}$ está localizada no próprio centro de massa do objeto, tal como na Figura 9.

Precisa-se de duas informações para descrever a pose de um corpo rígido em $\{R\}$ com respeito ao sistema $\{O\}$; a posição e a orientação do corpo em relação ao sistema global O.

Figura 9 – Representação gráfica da posição de um corpo rígido acoplado ao sistema de coordenadas $\{R\}$ com respeito ao sistema de coordenadas inercial $\{O\}$, e da posição de um corpo rígido em $\{A\}$ com respeito ao sistema de coordenadas $\{R\}$, o qual respeita o sistema $\{O\}$



Fonte: Autoria própria (2023).

É usado os vetores $\overrightarrow{p^O_R} e \overrightarrow{p^O_A} \in \mathbb{R}^3$ para representar a posição do centro de massa do corpo em relação ao sistema $\{O\}$, como mostra a Equação (2). Na imagem da Figura 9, o vetor $\overrightarrow{p^O_R}$ está indicado com a cor verde claro e é equacionado da seguinte forma:

$$\overrightarrow{p^O_R} = \begin{bmatrix} p^O_{R,x} \\ p^O_{R,y} \\ p^O_{R,z} \end{bmatrix}. \tag{2}$$

As coordenadas $(p^O_{R,x}, p^O_{R,y}, p^O_{R,z})$ representam as componentes (x,y,z) do vetor $\overrightarrow{p^O_R}$ expressadas em relação aos vetores unitários dos eixos do sistema $\{O\}$.

A orientação de um corpo em $\{R\}$ é definida pela matriz de rotação do seu sistema de coordenadas com respeito ao sistema de referência $\{O\}$ (Cruz Júnior *et al.*, 2021), denotada nesse trabalho por $\mathbf{R}^O_R$.

Para expressar a matriz de rotação, utiliza-se a representação mínima por ângulos de Euler $(\phi, \theta, \psi) = (roll, pitch, yaw) =$ (Rolamento, Arfagem, Guinada) correspondentes às rotações nos eixos formados por $\vec{x}_R, \vec{y}_R, \vec{z}_R$, respectivamente (Figura 10). Essa representação permite decompor a matriz de rotação em 3 rotações elementares em torno dos eixos do sistema de coordenadas $\{R\}$. Deste modo, a representação mínima φ^O_R assume o seguinte formato:

$$\varphi_R^O = \begin{bmatrix} \phi_R^O \\ \theta_R^O \\ \psi_R^O \end{bmatrix}. \tag{3}$$

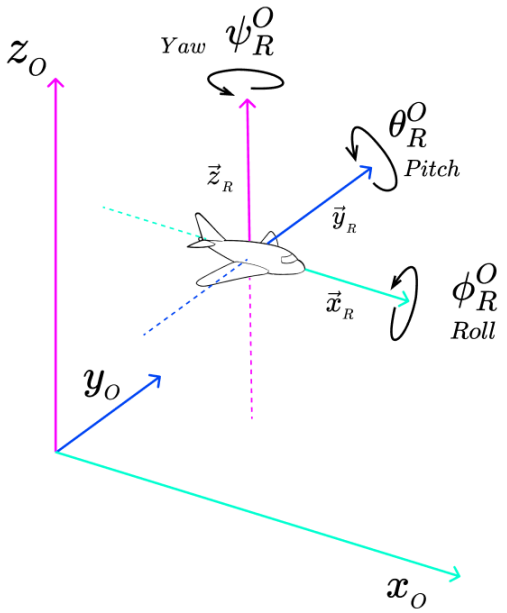
Para melhor entendimento, apresenta-se abaixo a forma das matrizes de rotação 3D em relação a cada eixo nas Equações 4, 5 e 6.

$$\mathbf{R}_{z_R}(\psi_R^O) = \begin{bmatrix} \cos(\psi_R^O) & -\sin(\psi_R^O) & 0 \\ \sin(\psi_R^O) & \cos(\psi_R^O) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\mathbf{R}_{y_R}(\theta_R^O) = \begin{bmatrix} \cos(\theta_R^O) & 0 & \sin(\theta_R^O) \\ 0 & 1 & 0 \\ -\sin(\theta_R^O) & 0 & \cos(\theta_R^O) \end{bmatrix} \tag{5}$$

$$\mathbf{R}_{x_R}(\phi_R^O) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_R^O) & -\sin(\phi_R^O) \\ 0 & \sin(\phi_R^O) & \cos(\phi_R^O) \end{bmatrix} \tag{6}$$

Figura 10 – Representação dos ângulos roll, pitch e yaw, com respeito a um robô móvel.



Fonte: Autoria própria (2023).

Proceduralmente, tem-se que a resultante da multiplicação das matrizes de rotação elementares euclidianas é dada na forma:

$$\mathbf{R}_R^O = \mathbf{R}_{z_R}(\psi_R^O)\mathbf{R}_{y_R}(\theta_R^O)\mathbf{R}_{x_R}(\phi_R^O), \tag{7}$$

$$\mathbf{R}_R^O \in SO(3). \tag{8}$$

Logo, com as duas informações, de posição e de rotação, pode-se descrever a pose $\mathbf{X}_R^O$ de um corpo no sistema $\{R\}$ com respeito à $\{O\}$ com a seguinte representação (Equação (9)) :

$$\mathbf{X}_R^O = \left(\overrightarrow{p_R^O}, \quad \mathbf{R}_R^O\right). \tag{9}$$

Dadas as condições acima, pode-se escrever a pose $\mathbf{X}_A^O$ de um corpo rígido em $\{A\}$ em relação ao sistema $\{R\}$ o qual é expressado no sistema inercial $\{O\}$ por meio das seguintes transformações (Equação (10) e Equação (11)) (Cruz Júnior *et al.*, 2021):

$$\overrightarrow{p_A^O} = \overrightarrow{p_R^O} + \mathbf{R}_R^O \cdot \overrightarrow{p_A^R} \quad , \tag{10}$$

$$\mathbf{R}_A^O = \mathbf{R}_R^O \mathbf{R}_A^R \, . \tag{11}$$

$$\mathbf{X}_A^O = \left(\overrightarrow{p_A^O}, \quad \mathbf{R}_A^O\right). \tag{12}$$

Outra maneira de representar transformações entre sistemas de coordenadas é fazer uso da matriz de transformação homogênea, a qual pertence ao grupo especial euclidiano de dimensão 3, sua definição segue o formato (Cruz Júnior *et al.*, 2021):

$$\mathbf{H}_R^O = \begin{bmatrix} \mathbf{R}_R^O & \overrightarrow{p_R^O} \\ \mathbb{O}_{1x3} & 1 \end{bmatrix}. \tag{13}$$

Tem-se que $\mathbf{H}_R^O$ é a matriz de transformação homogênea do sistema de coordenadas $\{\mathrm{R}\}$ para o sistema $\{\mathrm{O}\}$, $\overrightarrow{p_R^O}$ representa o vetor que leva um ponto na origem de $\{\mathrm{O}\}$ para a origem de $\{\mathrm{R}\}$, e por final, a matriz de rotação $\mathbf{R}_R^O$ a qual aplica as rotações elementares em relação a cada eixo para alinhar a direção e sentido dos eixos de $\{\mathrm{R}\}$ em $\{\mathrm{O}\}$. A partir dessas elaborações, constrói-se o sistema

$$\overrightarrow{p_A^O} = \begin{bmatrix} \mathbf{R}_R^O & \overrightarrow{p_R^O} \\ \mathbb{O}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \overrightarrow{p_A^R} \\ 1 \end{bmatrix} = \mathbf{H}_R^O \overrightarrow{p_A^R} \tag{14}$$

e por meio desta representação extrai-se a posição 3D do objeto ($\overrightarrow{p_O^A}$), juntamente à sua orientação, a qual é extraída da matriz de rotação $\mathbf{R}_R^O$.

Na Equação (14), $\overrightarrow{p_A^O}$ representa a posição de um ponto em {A} dada em relação ao sistema $\{R\}$ com respeito ao sistema inercial $\{O\}$ (Cruz Júnior *et al.*, 2021).

$$\overrightarrow{p_A^O} = \begin{bmatrix} p_{A,x}^O \\ p_{A,y}^O \\ p_{A,z}^O \end{bmatrix}. \tag{15}$$

$$\overrightarrow{p_A^R} = \begin{bmatrix} p_{A,x}^R \\ p_{A,y}^R \\ p_{A,z}^R \end{bmatrix}. \tag{16}$$

$$\overrightarrow{p_R^O} = \begin{bmatrix} p_{R,x}^O \\ p_{R,y}^O \\ p_{R,z}^O \end{bmatrix}. \tag{17}$$

Para esclarecimento, a Equação (15) relaciona o vetor $\overrightarrow{p_A^O}$ com os vetores unitários ortogonais que definem o sistema de coordenadas $\{O\}$. Semelhantemente, o vetor posição $\overrightarrow{p_A^R}$ é relacionado com os vetores unitários do sistema $\{R\}$ na Equação (16). Analogamente, tem-se a Equação (17), já apresentada em (2).

De maneira semelhante, é possível fazer rotações e translações em sistemas de coordenadas cartesianas $(x,y) \in \mathbb{R}^2$. Para aplicar uma rotação ψ em um ponto $\mathbf{q} = (x_0, y_0)$ o qual está expressado em relação ao sistema de coordenadas $\{O\}$ basta realizar a seguinte operação:

$$\mathbf{q}' = \mathbf{R}_{2D}(\psi)\mathbf{q}, \tag{18}$$

de maneira que $\mathbf{R}_{2D}(\psi)$ é a matriz de rotação 2D a qual rotaciona pontos bidimensionais de acordo com um ângulo ψ e mantém a magnitude do vetor $\overrightarrow{p_q}$ e $\mathbf{q}'$ é o resultado da operação. A representação gráfica desta transformação pode ser observada na Figura 11 e a matriz de rotação é da seguinte forma:

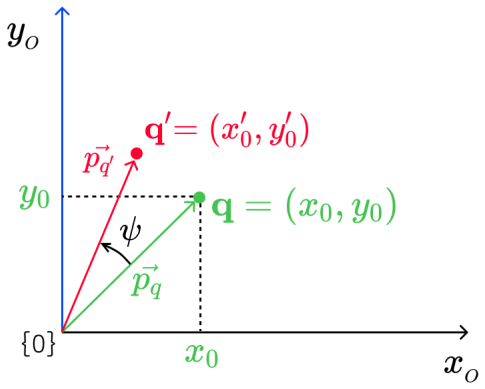
$$\mathbf{R}_{2D}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}. \tag{19}$$

Logo, equaciona-se

$$\mathbf{q}' = \begin{bmatrix} x_0' \\ y_0' \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \mathbf{R}_{2D}(\psi)\mathbf{q}' \tag{20}$$

para melhor visualização do sistema de equações.

Figura 11 – Representação gráfica da operação de rotação bidimensional



Fonte: Autoria própria (2023).

De forma análoga ao espaço tridimensional, também é possível a utilização de matrizes de transformação homogênea no espaço cartesiano. Essa operação é dada como:

$$\mathbf{T}(\mathbf{t}, \psi) = \begin{bmatrix} \mathbf{R}_{2D}(\psi) & \mathbf{t} \\ \mathbb{O}_{1x2} & 1 \end{bmatrix}, \tag{21}$$

de maneira que $\mathbf{t}$ é a translação $\left[t_x, t_y\right]^T$ e $\mathbf{R}_{2D}(\psi)$ é a matriz de rotação 2D em função do ângulo ψ.

2.2.1 Modelos cinemáticos de robôs com rodas

O modelo cinemático de um veículo descreve o comportamento do sistema mecânico empregado. Um dispositivo robótico movimentando-se em um plano possui um espaço de configurações $\mathbb{R}^2 \times \mathbb{S}^1$. O conjunto $\mathbb{R}^2$ é o espaço das configurações de um plano cartesiano, e $\mathbb{S}^1$ representa o espaço das configurações de um círculo. A configuração do monociclo é representada por $\mathbf{X}_R$, como demonstrado na Equação:

$$\mathbf{X}_R = \begin{bmatrix} p^O_{R,x} \\ p^O_{R,y} \\ \psi^O_R \end{bmatrix}, \tag{22}$$

de forma que $p^O_{R,x}$ e $p^O_{R,y}$ são coordenadas planares da origem do sistema de coordenadas acoplado ao dispositivo ($\{R\}$) em relação ao sistema inercial $\{O\}$, e ψ^O_R é o ângulo de rotação em torno do eixo $\vec{z}_R$ do robô.

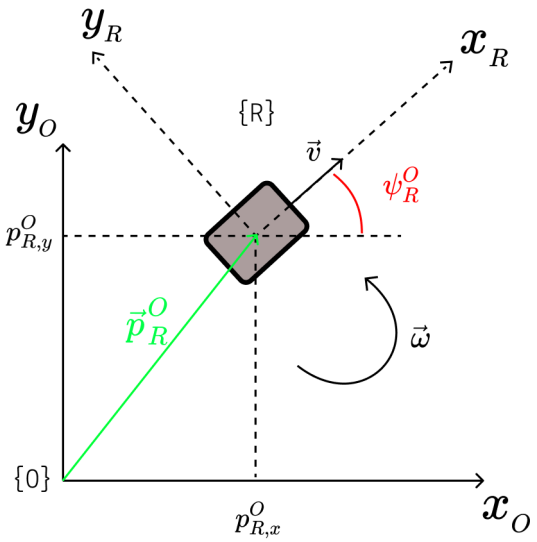
Para introduzir o modelo aplicado, a seguir apresenta-se o comumente aplicado modelo cinemático do monociclo. E na subseção consequente, explica-se o modelo diferencial o qual será implementado.

2.2.1.1 Modelo monociclo

Para poder aplicar o modelo monociclo parte-se de alguns princípios para a modelagem do problema:

- O robô se move em uma superficie plana;
- Não ha deslizamento;
- Não ha deformação nas rodas;
- Os eixos de orientação são perpendiculares ao solo.

Figura 12 – Representação da cinemática do modelo monociclo



Fonte: Autoria própria (2023).

Considerando a Figura 12 e o modelo planar (plano $x_O - y_O$), tem-se que a pose do robô

é dada pela Equação (22).

Entende-se que $(p^O_{R,x}, p^O_{R,y})$ são as componentes (x_O, y_O) do vetor $\overrightarrow{p^O_R}$ e ψ^O_R é o ângulo de rotação em torno do eixo z_O do robô. O modelo cinemático do monociclo permite relacionar as velocidades linear $\vec{v}$ e a angular $\vec{\omega}$, relativas ao sistema $\{R\}$, com as velocidades lineares $\dot{p}^O_{R,x}$ e $\dot{p}^O_{R,y}$ e a velocidade angular $\dot{\psi}^O_R$. É importante comentar que tal modelagem assume que existem limitações físicas atreladas ao movimento do monociclo (não há movimentos laterais). Essas restrições são chamadas de não-holonômicas e para o monociclo é dada por:

$$\dot{p}^O_{R,x} \sin(\psi^O_R) - \dot{p}^O_{R,y} \sin(\psi^O_R) = 0. \tag{23}$$

Partindo da Equação (23) e efetuando a integração numérica de primeira ordem de Euler (Cruz Júnior *et al.*, 2021; Sanz, 2009), a estimativa da pose do robô, em tempo discreto, é dada na forma:

$$\mathbf{X}_{R,k+1} = \begin{bmatrix} p^O_{R,x,k+1} \\ p^O_{R,y,k+1} \\ \psi^O_{R,k+1} \end{bmatrix} = \begin{bmatrix} v_k \cos(\psi^O_{R,k}) + p^O_{R,x,k} \\ v_k \sin(\psi^O_{R,k}) + p^O_{R,y,k} \\ \omega_k + \psi^O_{R,k} \end{bmatrix}, \tag{24}$$

tendo que:

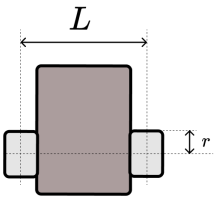
- v_k é a velocidade linear do robô no instante de tempo discreto k;
- ω_k é a velocidade angular do robô no instante de tempo discreto k;
- $p^O_{R,x,k+1}$ é a componente da pose do robô no eixo x_O, dada no instante $k+1$;
- $p^O_{R,x,k}$ é a componente da pose do robô no eixo x_O, dada no instante k ;
- $p^O_{R,y,k+1}$ é a componente da pose do robô no eixo y_O, dada no instante $k+1$;
- $p^O_{R,y,k}$ é a componente da pose do robô no eixo y_O, dada no instante k ;
- $\psi^O_{R,k+1}$ é a componente da pose do robô no eixo z_O (ângulo de rotação em torno do eixo z_O), dada no instante $k+1$;
- $\psi^O_{R,k}$ é a componente da pose do robô no eixo z_O (ângulo de rotação em torno do eixo z_O), dada no instante k.

A relação em (24) foi retirada e interpretada dos trabalhos de Cruz Júnior *et al.* (2021) e Cota (2019).

2.2.1.2 Modelo diferencial

A modelagem diferencial consiste em interpretar a movimentação do robô assumindo que este possui duas rodas paralelas de raio r, as quais estão separadas por uma distância L (Figura 13). Desta forma, o dispositivo é capaz de realizar curvas e giros em torno do próprio eixo dependendo das velocidades lineares das rodas direita e esquerda: $\vec{v}_d$ e $\vec{v}_e$ (Figura 14).

Figura 13 – Representação do modelo diferencial

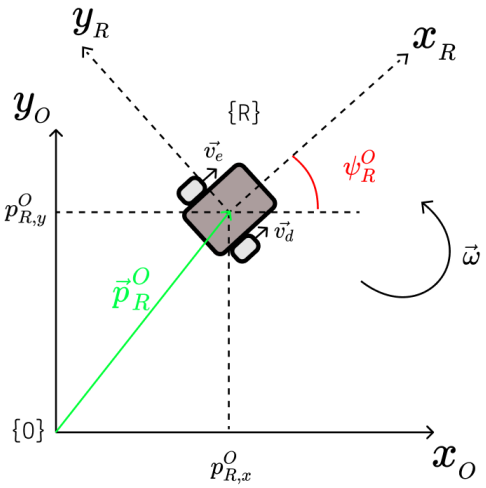


Fonte: Autoria própria (2023).

No modelo diferencial é dada a relação (Cota, 2019):

$$\begin{bmatrix} \vec{v} \\ \vec{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} \vec{v}_d \\ \vec{v}_e \end{bmatrix}. \tag{25}$$

Figura 14 – Representação da cinemática do modelo diferencial



Fonte: Autoria própria (2023).

Por meio das Equações ((25) e (24)) tem-se que as velocidades lineares ($\dot{p}_{R,x}^O$ e $\dot{p}_{R,y}^O$) e a velocidade angular ($\dot{\psi}_R^O$) do robô são da seguinte forma (Cota, 2019):

$$\begin{bmatrix} \dot{p}^O_{R,x} \\ \dot{p}^O_{R,y} \\ \dot{\psi}^O_R \end{bmatrix} = \begin{bmatrix} \cos(\psi^O_R) & 0 \\ \sin(\psi^O_R) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} v_d \\ v_e \end{bmatrix} \tag{26}$$

Realizando a integração numérica de 1ª ordem pelo método de Euler (Cota, 2019; Sanz, 2009) tem-se as seguintes relações:

$$p^O_{R,x,k+1} = \frac{(v_{d,k} + v_{e,k})}{2} \cos(\psi^O_{R,k}) + p^O_{R,x,k} \tag{27}$$

$$p^O_{R,y,k+1} = \frac{(v_{d,k} + v_{e,k})}{2} \sin(\psi^O_{R,k}) + p^O_{R,y,k} \tag{28}$$

$$\psi^O_{R,k+1} = \frac{(v_{d,k} - v_{e,k})}{L} + \psi^O_{R,k} \tag{29}$$

Apesar da modelagem cinemática ser uma etapa crucial para a solução do problema SLAM, é um fato que ao passar do tempo as estimativas geradas por tal modelagem ficam obsoletas, pois a presunção de que não há escorregamento nas rodas torna os cálculos imprecisos. Logo, para atenuar os erros acumulados pela etapa da odometria de rodas pode-se utilizar informações de outros sensores, tais como sensores laser.

2.3 Princípios de funcionamento do LiDAR SLAM

As técnicas LiDAR SLAM são abordagens para a solução do problema de localização e mapeamento as quais utilizam medições oriundas de sensores lasers para estimar a odometria do robô (Cruz Júnior *et al.*, 2021).

2.3.1 Sistema de aquisição de dados

A essência de um sistema SLAM está na aquisição de informações do ambiente por via de sensores, o processamento destas informações e a geração da localização do dispositivo juntamente ao mapeamento do espaço (Cruz Júnior *et al.*, 2021). Os sensores LiDAR são capazes de escanear múltiplos pontos do ambiente e prover dados na forma de "nuvem de pontos"(conjuntos de pontos).

Os dados coletados pelos sensores lasers são apresentados em diferentes formatos

Figura 15 – Sensor RPLiDAR A1M8

Fonte: (Gurel, **2018**).

dependendo do fabricante. O sensor é capaz de apresentar dados do ambiente em formato de nuvem de pontos, portanto, o conjunto de pontos de uma varredura de um sensor LiDAR pode ser organizado em uma lista como:

$$s_L = s_L^{(1:m)} = \{s_L^{(1)}, \ \ldots, \ s_L^{(m)}\}. \tag{30}$$

Na Equação (30), tem-se que *m* é a quantidade de pontos detectados e $s_L^{(i)}$ são coordenadas cartesianas as quais respeitam as medições do sensor. No escopo desse trabalho, as coordenadas são bidimensionais, pois o A1M8 é um sensor laser 2D capaz de fazer 360 leituras espaçadas por aproximadamente 1º.

2.3.2 Odometria com sensor LiDAR

A odometria é o processo o qual, a partir de dados sensoriais sequenciais, interpreta-se as medições de deslocamento do robô com a finalidade de criar modelos da posição e orientação atual do agente (Dudek; Jenkin, 2010).

O robô deve conhecer sua posição e orientação a cada instante em um sistema de coordenadas absoluto. Para isso, é preciso manipular corretamente as medições do sensor laser e usá-las para estimar a pose do dispositivo. Uma das soluções propostas para estimação da pose por um conjunto de pontos é o algoritmo ICP (*Iteractive Closest Points*) (Besl; McKay, 1992), o qual é atualmente uma abordagem tradicional para esta finalidade.

2.3.2.1 Algoritmo ICP

Nas técnicas de LiDAR SLAM, a pose é calculada pelo alinhamento entre duas nuvens de pontos escaneadas pelo sensor ou entre a nuvem e a informação do mapa construído. No Algoritmo 1, a comparação é entre duas varreduras sequenciais.

O algoritmo ICP é uma das soluções para estimação da pose cuja essência é a comparação entre nuvens de pontos. Para isso, considera-se dois conjunto de pontos; $\mathrm{s}_{L,k}$ correspondente à varredura atual, e $\mathrm{s}_{L,k-1}$ corresponde ao conjunto de pontos que representa o escaneamento imediatamente anterior. A partir desses dados, estima-se de maneira iterativa a translação e rotação relativas $X_L = (\overrightarrow{p_L}, \mathbf{R}_L)$ as quais minimizam o erro quadrático entre os conjuntos. A minimização da função de custo é dada por (Cruz Júnior *et al.*, 2021) :

$$e(\overrightarrow{p_L}, \mathbf{R}_L) = \sum_j |\mathrm{s}^{(j)}_{L,k-1} - (\mathbf{R}_L\ \mathrm{s}^{(j)}_{L,k} - \overrightarrow{p_L})|^2. \tag{31}$$

Tem-se que $e(\overrightarrow{p_L}, R_L)$ é o erro a ser minimizado, $\mathrm{s}^{(j)}_{L,k-1}$ representa as coordenadas do ponto j para a varredura $k-1$, semelhantemente $\mathrm{s}^{(j)}_{L,k}$ para o escaneamento k, e $\mathbf{R}_L$ é a matriz de rotação da pose estimada pela odometria LiDAR com respeito ao sistema $\{R\}$ acoplado ao robô. Nota-se que os símbolos $\mathrm{s}^{(j)}_{L,k-1}$ e $\mathrm{s}^{(j)}_{L,k}$ correspondem às nuvens de pontos geradas por leituras consecutivas. Logo a saída do algoritmo deve retornar a rotação ($\mathbf{R}_L$) e posição ($\overrightarrow{p_L}$) estimada pela minimização do erro quadrático da função de custo descrita na Equação (31).

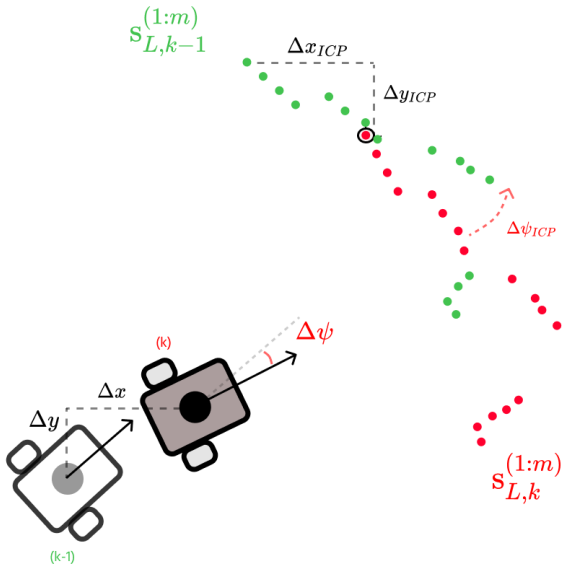
Algoritmo 1 – Algoritmo ICP

Requer: $\mathrm{s}^{(1:m)}_{L,k}$ **e** $\mathrm{s}^{(1:m)}_{L,k-1}$
Garantir: $X_L = (\overrightarrow{p_L}, \mathbf{R}_L)$
para $i = 0$ até N **faz**
  N : número máximo de iterações
  para $\mathrm{s}^{(j)}_{L,k} \in \mathrm{s}^{(1:\mathrm{m})}{}_{L,k}$ **faz**
    Encontrar o ponto mais próximo no raio (d_{max}) no conjunto $\mathrm{s}^{1:m}_{L,k-1}$.
  finaliza para
  Calcula a transformação $(\overrightarrow{p_L}, R_L)$ que minimiza o erro da Equação (31).
  Aplica a transformação nos pontos $\mathrm{s}^{(1:\mathrm{m})}{}_{L,k}$.
  Calcula a diferença do erro quadrático $|e_{i-1}(\overrightarrow{p_L}, \mathbf{R}_L) - e_i(\overrightarrow{p_L}, \mathbf{R}_L)|$.
  Se a diferença for menor que um limiar e_{min}, finaliza.
finaliza para

Fonte: Funcionamento geral do ICP (Cruz Júnior *et al.*, 2021) .

Para entender a Figura 16 faz-se que $\overrightarrow{p_L} = (\Delta x_{ICP}, \Delta y_{ICP})$ e $\mathbf{R}_L = \mathbf{R}_{z_L}(\Delta\psi_{ICP})$. Tendo que $\mathbf{R}_{z_L}(\Delta\psi_{ICP})$ é a matriz de rotação elementar em torno do eixo z_L do sensor LiDAR e $(\Delta x_{ICP}, \Delta y_{ICP})$ são coordenadas planares as quais correspondem a translação estimada pelo

Figura 16 – Descrição gráfica do objetivo do Algoritmo ICP



Fonte: Autoria própria (2023).

algoritmo ICP.

No Algoritmo 1, as linhas 1 e 2 definem o número máximo de iterações para o cálculo do alinhamento das nuvens. Em seguida, as linhas 3, 4 e 5 determinam a etapa nomeada de encontrar correspondências, a qual elege quais pontos serão comparados na etapa seguinte, ou seja, estipula para cada ponto na leitura atual um ponto no escaneamento anterior o qual será usado para o cálculo do erro (função de custo). Após estabelecer as correspondências, pode-se expressar o erro como uma função dependente da distância entre os pontos pareados, deste modo, tenta-se minimizar esta diferença aplicando uma transformação (2D) a qual atualiza a posição dos pontos do escaneamento anterior. Em seguida, computa-se novamente o erro com a posição atualizada da nuvem $k-1$. Na iteração a qual a diferença entre os erros calculados for menor que um limiar definido, o algoritmo retorna a transformação a qual sobrepõe as duas nuvens de pontos de entrada.

Para estabelecer qual transformação minimiza o erro $e(\overrightarrow{p_L}, R_L)$ é preciso solucionar o problema de mínimos quadrados não linear apresentado na Equação (31), com isto em mente, simplifica-se-a aplicando as seguintes modificações:

$$s_k^{(j)} = \mathbf{q}_i, \tag{32}$$

a qual representa um ponto i na nuvem de pontos atual;

$$\mathrm{s}_{k-1}^{(j)} = \mathbf{p}_i \ , \tag{33}$$

a qual representa um ponto i na nuvem de pontos anterior;

$$\overrightarrow{p_L} = \mathbf{t} \ , \tag{34}$$

a qual representa a translação estimada;

$$\mathbf{R}_L = \mathbf{R} \ , \tag{35}$$

a qual representa a matriz de rotação 2D estimada;

$$e(\overrightarrow{p_L}, \mathbf{R}_L) = f_e, \tag{36}$$

a qual representa o erro a ser minimizado.

Logo, é possível reescrever a Equação (31) e expressá-la da seguinte forma:

$$f_e = \sum_i |\mathbf{q}_i - \mathbf{R}\mathbf{p}_i + \mathbf{t}|^2 \ , \tag{37}$$

tendo que:

$$\mathbf{t} = \begin{bmatrix} t_x & t_y \end{bmatrix}^T \tag{38}$$

representa a translação que minimiza o erro f_e;

$$\mathbf{q}_i = \begin{bmatrix} q_x^i & q_y^i \end{bmatrix}^T \tag{39}$$

define um ponto i no conjunto $\mathbf{q}$ o qual corresponde à nuvem de pontos atual;

$$\mathbf{p}_i = \begin{bmatrix} p_x^i & p_y^i \end{bmatrix}^T \ , \tag{40}$$

representa um ponto i no conjunto $\mathbf{p}$ o qual corresponde à nuvem de pontos anterior;

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \ , \tag{41}$$

caracteriza a matriz de rotação a qual alinha as duas nuvens de entrada $\mathbf{q}$ e $\mathbf{p}$.

Uma outra modificação é aplicada à Equação (37) para configurá-la da maneira apropriada para construção do sistema de equações desejado. Isto é possível fazendo:

$$\mathbf{T} = \mathbf{T}(\mathbf{t}, \mathbf{R}) = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{42}$$

tendo que $\mathbf{T}$ é a matriz de transformação 2D a qual engloba a rotação $\mathbf{R}$ e a translação $\mathbf{t}$ apresentadas na Equação (31). Logo pode-se reequacionar a Equação (37) da seguinte forma:

$$f_e = \sum |\mathbf{q_i} - \mathbf{T}\mathbf{p_i}|^2. \tag{43}$$

A Equação (43) é uma função exponencial não linear a qual a caracteriza a expressão como um problema de quadrados mínimos não lineares. A primeira etapa para solucionar esse tipo de sistema é linearizar a função, desta forma, aplica-se a aproximação proposta por Low (2004), na qual assume-se que: $\theta \approx 0 \implies \sin\theta \approx \theta,\ \cos\theta \approx 0$.

Com a linearização, a Equação (37) toma a seguinte forma:

$$f_e = |\mathbf{T}\mathbf{p}_i - \mathbf{q}_i|^2 = |\mathbf{q}_i - \mathbf{T}\mathbf{p}_i|^2 = \left| \begin{bmatrix} 1 & -\theta & t_x \\ \theta & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x^i \\ p_y^i \\ 1 \end{bmatrix} - \begin{bmatrix} q_x^i \\ q_y^i \\ 1 \end{bmatrix} \right|^2. \tag{44}$$

Logo, a partir do sistema acima pode-se isolar as variáveis independentes e configurá-lo da seguinte maneira:

$$\mathbf{T}\mathbf{p}_i - \mathbf{q}_i = \begin{bmatrix} 1 & 0 & -p_y^i \\ 0 & 1 & p_x^i \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ \theta \end{bmatrix} - \begin{bmatrix} q_x^i - p_x^i \\ q_y^i - p_y^i \\ 1 \end{bmatrix} = \mathbf{A}_i\mathbf{X} - \mathbf{b}_i\ , \tag{45}$$

tendo que $\mathbf{X} = [t_x \ \ t_y \ \ \theta]^T$ é a resposta do algoritmo ICP, ou seja, representa a translação e rotação a qual melhor alinha as nuvens de pontos $\mathbf{q}$ e $\mathbf{p}$.

Desta forma, a Equação (45) assemelha-se ao caso básico da minimização de erro quadrático $|\mathbf{A}\mathbf{X} - \mathbf{b}|^2 = 0$. Com isto, pode-se chegar no seguinte sistema:

$$f_e = \sum |\mathbf{A}_i\mathbf{X} - \mathbf{b}_i|^2 = |\mathbf{A}\mathbf{X} - \mathbf{b}|^2. \tag{46}$$

Entende-se que a expressão acima se trata de um problema de mínimos quadrados linear

em que **A** e **b** são da forma:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -p_y^i \\ 0 & 1 & p_x^i \\ 0 & 0 & 1 \\ . & . & . \\ . & . & . \\ . & . & . \\ 1 & 0 & -p_y^m \\ 0 & 1 & p_x^m \\ 0 & 0 & 1 \end{bmatrix}, \tag{47}$$

$$\mathbf{b} = \begin{bmatrix} q_x^i - p_x^i \\ q_y^i - p_y^i \\ 1 \\ . \\ . \\ . \\ q_x^m - p_x^m \\ q_y^m - p_y^m \\ 1 \end{bmatrix}. \tag{48}$$

Assim, resolve-se a Equação (45) decompondo a matriz **A** por meio do cálculo SVD (*Singular Value Decomposition*) (Huang, S.; Huang, H., 2022) e encontra-se a pose relativa **X** a qual minimiza o erro quadrádico f_e.

3 MATERIAIS E MÉTODOS

A presente seção discorre sobre o arcabouço experimental utilizado nas simulações e na validação do algoritmo implementado para investigar a técnica de odometria LiDAR proposta. Inicialmente será descrito sobre os materiais usados para os experimentos e extração de dados, destacando o sensor LiDAR A1M8 e a placa de desenvolvimento de baixo custo baseada no microprocessador STM32F407VET6. Em seguida apresentam-se as bibliotecas, linguagens e IDE's utilizadas no desenvolvimento e verificação da solução.

Ao fim dessa seção é discorrido sobre a metodologia para a implementação do algoritmo proposto e as métricas para avaliar sua precisão e viabilidade de ser processado em um microcontrolador.

3.1 Materiais

O experimento desenvolvido para a avaliação do algoritmo proposto foi conduzido utilizando os dispositivos físicos apresentados no Quadro 1. As linhas do Quadro 1 expõem os *Hardwares* utilizados, as especificações e a finalidade dos respectivos componentes, enquanto, o Quadro 2 lista as ferramentas de *Softwares* utilizadas.

3.1.1 Placa de desenvolvimento STM32F407VGT6

O processador STM32F407VGT6 é um dispositivo de baixo custo o qual possui a frequência máxima de operação de 168MHz e memória RAM de 192Kbytes e foi utilizado para verificar se a implementação é computacionalmente eficiente. O componente contém um processador com arquitetura Arm Cortex-M4 32-bit RISC, possui suporte para operações de ponto flutuante e a fabricante disponibiliza o *Software* STM32CubeIDE (STMicroelectronics, 2023) o qual possibilita carregar códigos em linguagem C/C++ nos microcontroladores STM32. Integrada a plataforma STM32CubeIDE, está a biblioteca HAL (*Hardware Abstraction Layer*) a qual fornece uma abstração das camadas de baixo nível do processador e permite a codificação de projetos embarcados a partir de suas funções de interface (presentes para todos os periféricos dos processadores da fabricante).

Quadro 1 – Materiais utilizados no desenvolvimento do experimento

Componente	**Especificações**	**Finalidade**
Desktop	CPU Intel Core i3-6100, placa mãe GIGABYTE H110M-S2H-CF, GPU NVIDIA GeForce GTX 1050 Ti e SO Windows 10.	Testes e simulações
Sensor LiDAR A1M8	Sensor de triangulação laser, ecaneamentos 2D em 360 graus com a definição em 1º e 12 metros de alcance.	Entrada do algoritmo ICP
Microcontrolador STM32F407VET6	Núcelo ARM-Cortex-M4 de 32 bits, frequência de clock de 168MHz, memória Flash integrada de até 512 KB, memória RAM de 192 KB e periféricos e interfaces como Timers, UART, I2C, SPI	Processar dados LiDAR
Conversor TTL USB p/ Serial CH340	Permite que o computador ou outro dispositivo com porta USB se conecte a dispositivos que utilizam comunicação serial UART em níveis lógicos TTL (Transistor-Transistor-Logic)	Gravar dados do sensor LiDAR
Samsung Galaxy M53 5G	Câmera:108 MP, f/1.8, (wide), PDAF	Gravação do vídeo do experimento
STlinkV2	Interface de comunicação SWIM (*Single Wire Interface Module*) e JTAG	Passa código para o microcontrolador
Notebook Lenovo Ideapad	AMD Ryzen™ 5 7520U 256GB QLC M.2 PCIe SSD	Ajudar na captura dos dados LiDAR

Fonte: Autoria própria (2023).

3.1.2 Sensor LiDAR A1M8

O sensor RpLiDAR A1M8 pode ser configurado e operar com frequências e definições diferentes, porém, neste trabalho ele opera em aproximadamente 10Hz, fazendo 10 escaneamentos de 360 distâncias (e 360°) por segundo. Nesta configuração, cada ponto no escaneamento está

espaçado em um ângulo aproximadamente 1°.

3.1.3 Bibliotecas e ferramentas

Todas as ferramentas de *Software* estão listadas no Quadro 2 juntamente a licença de cada sistema e a finalidade do utilitário dentro do presente trabalho. Essencialmente, a implementação da odometria visual foi feita com a linguagem de alto nível Python e a utilização da biblioteca de processamento gráfico OpenCV. Entretanto, como o objetivo é embarcar a odometria laser em um microcontrolador, elegeu-se a linguagem C++ pela compatibilidade com as plataformas de desenvolvimento escolhidas e um bom desempenho no cenário do microcontrolador. Para implementar as operações algébricas da solução, escolheu-se a biblioteca Eigen (Benoît Jacob, 2021) a qual oferece prontas as funções de operações matriciais necessárias para a realização do algoritmo ICP. Paralelamente, a biblioteca OpenCV contém as operações de álgebra linear para os cálculos da abordagem visual.

Quadro 2 – Softawares utilizados no desenvolvimento do experimento

Ferramenta	**Licença**	**Finalidade**
Python	ZERO-CLAUSE BSD LICENSE	Odometria visual
C++ (g++ 12.2.0)	BSD-3-Clause	Odometria LiDAR
OpenCV 4.8.0	Apache 2 License	Odometria visual
Eigen 3.4.0	Mozilla Public License 2.0	Odometria LiDAR
Adafruit CircuitPython PyLiDAR	MIT License	Gravação de dados LiDAR
VSCode	MIT License	Edição dos códigos
STM32CubeIDE	Mix Ultimate Liberty+OSS+3rd-party V1 SOFTWARE LICENSE AGREEMENT	Implementação no microcontrolador

Fonte: Autoria própria (2023).

3.2 Métodos

Para avaliar o funcionamento do algoritmo desenvolvido, este trabalho foi divido em 3 etapas: a coleta de dados do experimento, o processamento dos dados do experimento e a avaliação do desempenho do algoritmo ICP implementado. A coleta dos dados é imprescindível pois é preciso avaliar o comportamento temporal do código na plataforma embarcada e na máquina

pessoal apresentadas no Quadro 1, além de verificar caso as estimativas da implementação corroboram com o movimento real da base do sensor LiDAR.

Na etapa da coleta de dados, tem-se dois sensores gerando dados: o sensor LiDAR A1M8 e a câmera de celular (Samsumg Galaxy M58). Os dados do experimento consistem em 19,5 segundos de captura dos aparatos sensoriais citados acima (19,5s de vídeo e escaneamentos).

Na etapa de processamento de dados, tem-se dois fluxos diferentes para o processamento dos dados: as imagens da câmera alimentam o processo da odometria visual e a odometria ICP processa os dados do sensor laser. Também é coletado o tempo de execução do processamento de cada conjunto de dois escaneamentos sequenciais para definir a aceitabilidade do sistema em tempo real. O fluxo mencionado pode ser visualizado na Figura 17 e cada etapa é destacada pelo respectivo retângulo pontilhado.

Consequentemente, para verificar a viabilidade da implementação do sistema desenvolvido são propostas duas métricas: uma em relação ao erro da trajetória estimada pelo algoritmo ICP em comparação com a trajetória real (vinda da odometria visual) e a outra avalia o alinhamento aplicado pelo algoritmo ICP. Adicionalmente, propõe-se investigar o tempo de execução do algoritmo em duas plataformas: no microcontrolador STM32F407 e na máquina do acadêmico cujas especificações estão na primeira linha do Quadro 1.

Apesar da modelagem diferencial de robô de duas rodas apresentada na fundamentação teórica, a técnica não foi aplicada pois não foi desenvolvido um modelo físico para o robô de duas rodas. Logo o experimento foi conduzido utilizando apenas o sensor LiDAR sendo deslocado por meio de uma cadeira.

3.2.1 Coleta de dados do experimento

O sensor LiDAR foi acoplado a uma caixa de papelão a qual foi colocada em cima de uma cadeira com rodas que se locomove pelo ambiente de acordo com o movimento feito pelo acadêmico; a cadeira é manipulada para simular o movimento de um robô. Entende-se que a estrutura de locomoção da base não é ideal, porém, a engenharia envolvida nos sistemas de atuadores de um robô móvel não faz parte do escopo deste trabalho. Logo, foi necessário simplificar ao máximo a estrutura na qual o sensor LiDAR seria acoplado.

É preciso manter visível a base do sensor em relação à imagem capturada pela câmera do celular. Consequentemente, posiciona-se a câmera logo acima da posição inicial do sensor LiDAR. Após iniciar a coleta de dados dos dois sensores citados acima movimenta-se a estrutura

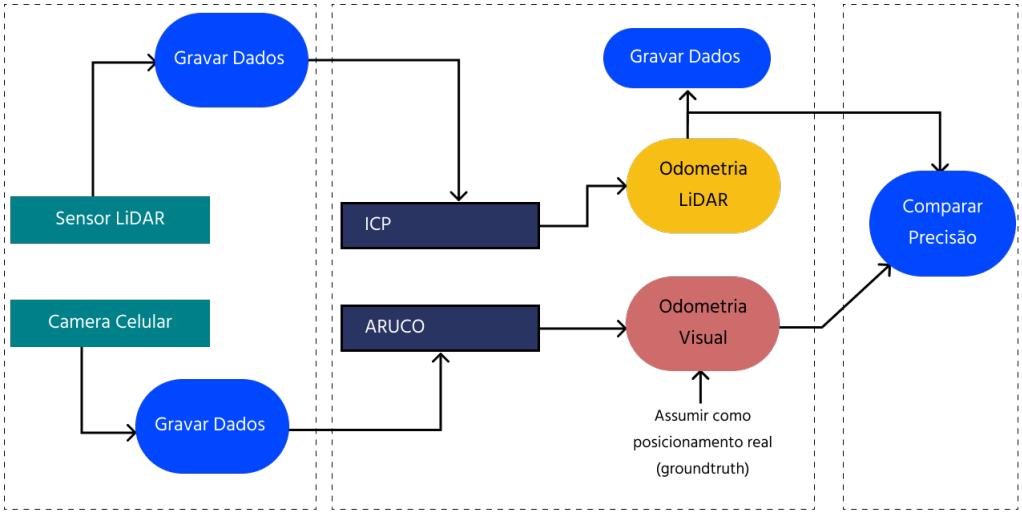
da base para localizá-la por meio das abordagens propostas. O experimento ocorreu numa sala da empresa AutenPRO e espera-se que os escaneamentos LiDAR representem um corte seccional da planta da sala.

A coleta dos dados do sensor laser RpLiDAR A1M8 ocorre por meio do módulo de conversão serial (UART) para USB e da biblioteca *adafruit rplidar* a qual possui funções de fácil implementação para capturar cada escaneamento (frame) do sensor. Com o cabo conectado ao sensor e um fio para puxar a base do sensor LiDAR foi realizada a coleta de dados do movimento da base.

Para a coleta das imagens, coloca-se a câmera de celular em uma posição estática, elevada e direcionada ao sensor LiDAR e sua base. O objetivo do posicionamento da câmera é manter a visibilidade do marcador ARUCO para que a odometria visual possa gerar os resultados do posicionamento do marcador.

Entende-se que o mesmo movimento é capturado pelo sensor laser e pela câmera, desta maneira, é possível comparar os resultados das duas abordagens e avaliar a precisão do algoritmo desenvolvido.

Figura 17 – Fluxograma do arcabouço experimental para a avaliação da precisão da proposta



Fonte: Autoria própria (2023).

3.2.2 Odometria visual

A odometria visual é a etapa na qual as imagens provenientes da câmera são processadas para determinar a localização da base do sensor laser. Neste trabalho, o principal foco é comparar a

odometria visual com a odometria LiDAR, assumindo que o processamento visual disponibilizado pela biblioteca OpenCV determina a trajetória real (*groudtruth*) do objeto em questão.

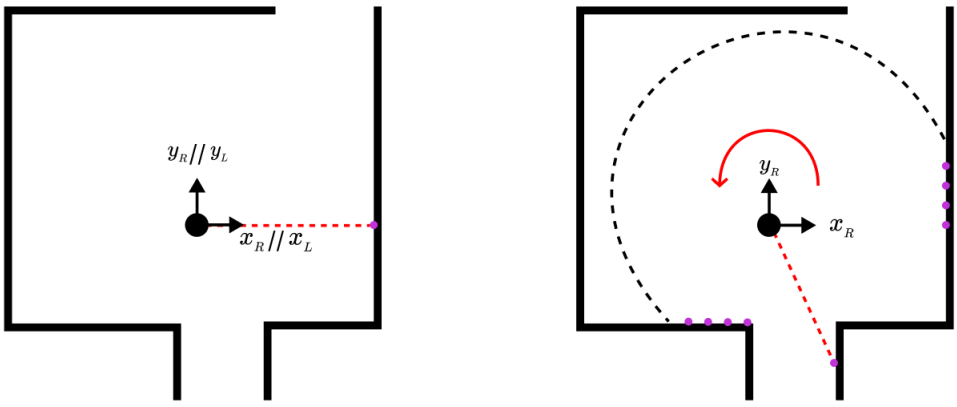
É importante mencionar que nas abordagens as quais utiliza-se câmeras é esperado boas condições de iluminação, umidade e temperatura para obter resultados coerentes. Como o experimento foi realizado no ambiente interno da empresa e com boas condições de iluminação, é possível aceitar os dados da câmera com um bom grau de confiança.

3.2.3 Odometria LiDAR

Tendo-se em mente que o sensor LiDAR RPLiDAR A1M8 consegue fazer leituras de distância entre 150-12.000 mm e escanear 360° com uma definição de aproximadamente 1°, manipula-se a informação contida na mensagem de escaneamento do sensor para tirar conclusões sobre a pose atual do robô. Nota-se que caso não houver pulso laser de retorno dentro de 23.33ns após a emissão, a leitura de distância na respectiva direção será "inf"e terá o significado de espaço vazio e não entra para o processamento do ICP.

Em cada varredura é possível medir uma distância a cada 1° de rotação, assim, uma varredura completa é composta por 360 distâncias entre o sensor e pontos no ambiente. Dessa forma, cada escaneamento gera uma nuvem de pontos que representa a forma do ambiente no momento da leitura. Logo, para calcular a odometria entre duas leituras, deve-se minimizar a distância entre os pontos desses escaneamentos calculando a pose relativa $\mathbf{X}_L = (\overrightarrow{p_L}, \mathbf{R}_L)$ entre os dois conjuntos de pontos (Cruz Júnior *et al.*, 2021).

Figura 18 – Imagem descritiva do funcionamento do sensor *LiDAR*

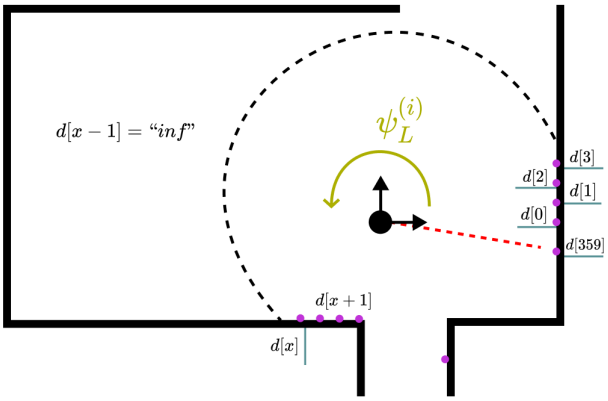


Fonte: Autoria própria (2023).

A Figura 18 expõe o funcionamento de um sensor LiDAR 2D o qual é capaz de gerar

a nuvem de pontos evidenciada pela coloração lilás por meio do tempo de reflexão dos pulsos lasers (tracejado vermelho). Nesta figura, o sensor é denotado pelo círculo preto no centro do ambiente o qual é representado pelas linhas pretas mais espessas. As duas representações na Figura 18 fazem parte do mesmo escaneamento equivalente a um instante t.

Figura 19 – Imagem descritiva do funcionamento do sensor *LiDAR LDS-01*



Fonte: Autoria própria (2023).

A partir da mensagem transmitida pelo sensor laser, extrai-se do vetor de distâncias (*ranges*) de tamanho $m_r = 360$ as informações necessárias para popular a lista $s_L^{(1:m_r)}$. Na Figura 19, observa-se que a informação de distância postada pelo sensor é denotada nessa proposta por $d[1:m_r]$. O vetor $d[1:m_r]$ é populado de acordo com as leituras do sensor laser, as quais são transmitidas por uma mensagem do tipo *LaserScanner* .

Caso não haja leitura de distância, tem-se que o vetor de distâncias no índice (i) terá o valor "inf", o qual é associado à interpretação de espaço vazio. Caso exista leitura de distância, a informação contida em $d[i]$ é usada para popular a lista $s_L^{(1:m)}$.

Desta forma, associa-se a lista de pontos $s_L^{(1:m)}$ com as coordenadas $(s_{L,x}^{(i)}, s_{L,y}^{(i)})$ a partir da relação:

$$s_L^{(1:m)} = \{(s_{L,x}^{(1)}, s_{L,y}^{(1)}),\ \ldots,\ (s_{L,x}^{(m)}, s_{L,y}^{(m)})\}. \tag{49}$$

É necessário relacionar as distâncias medidas (*ranges*) à coordenadas cartesianas. Nessa etapa são usadas as seguintes relações e funções:

$$\begin{bmatrix} s_{L,x}^{(i)} \\ s_{L,y}^{(i)} \end{bmatrix} = \begin{bmatrix} d^{(i)} \cos(\psi_L^{(i)}) \\ d^{(i)} \sin(\psi_L^{(i)}) \end{bmatrix}, \tag{50}$$

a qual representa a conversão para coordenadas cartesianas;

$$distancia\_medida(d[0:m_r]) = \{d^{(1)}, \ldots, d^{(m)}\} = d^{(i)} \quad , \tag{51}$$

a função a qual deve retornar uma lista com as distancias medidas filtrando os valores inválidos ïnf ¨;

$$indice(d[j]) = j = indice(d^{(i)}) \quad , \tag{52}$$

a função a qual retorna o índice j da distância $d[j]$ no vetor de distâncias;

$$\psi_L^{(i)} = \frac{\pi(indice(d^{(i)}))}{180}[rad] \quad , \tag{53}$$

a equação que calcula o ângulo de incidência do laser;

$$angulo\_da\_medida(d[0:m_r]) = \{\psi_L^{(1)}, \ldots, \psi_L^{(m)}\} = \psi_L^{(i)} \quad , \tag{54}$$

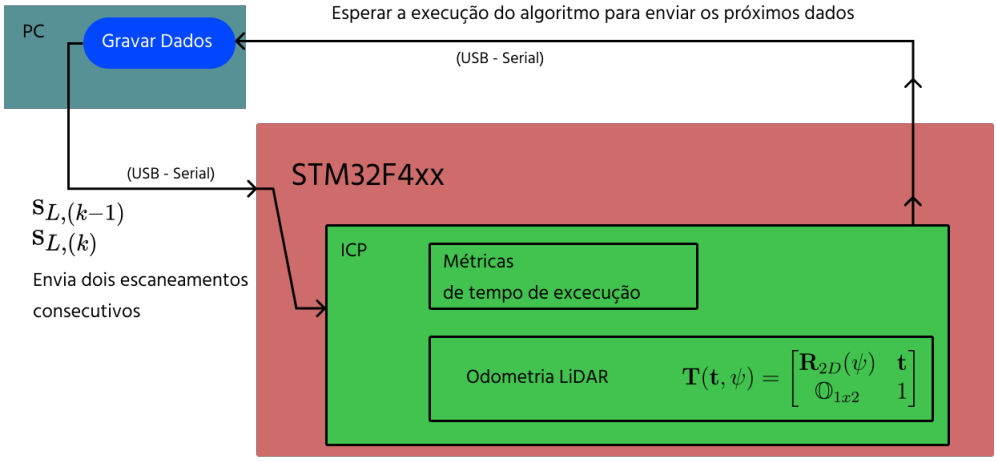
a função que calcula o ângulo de acordo com a fórmula mostrada na Equação (53)

A partir das relações em (49) e (50), consegue-se os escaneamentos sequenciais necessários como entrada para o algoritmo ICP descrito no Capítulo 2.

Após salvos os arquivos do experimento, são utilizados como entrada do algoritmo ICP, para avaliar o desempenho em relação ao tempo de execução e em relação ao alinhamento das nuvens e precisão das estimativas.

O arcabouço para a execução do algoritmo com processamento em sistema embarcado é apresentado na Figura 20, a qual ilustra o fluxo de dados da abordagem proposta. Inicialmente, um script em python é executado para estabelecer comunicação serial com a Placa de desenvolvimento, e assim, inicia-se a transmissão dos dados dos escaneamentos capturados do sensor LiDAR. No momento em que dois escaneamentos consecutivos de 360 distâncias forem acumulados na memória RAM do microcontrolador, será executado o algoritmo ICP desenvolvido. O algoritmo retorna a transformação a ser aplicada ao escaneamento anterior para alinhá-lo ao escaneamento atual e utiliza destes dados para montar a trajetória efetuada pela base do sensor LiDAR. Após a resposta do código ICP, grava-se todas as transformações estimadas pela abordagem embarcada para ser comparada com a Odometria Visual proveniente da biblioteca AruCo de visão computacional.

Figura 20 – Fluxograma do arcabouço experimental para a avaliação da viabilidade de processamento embarcado para o algoritmo proposto



Fonte: Autoria própria (2023).

3.2.4 Métricas para avaliação

Para avaliar o funcionamento do algoritmo ICP utiliza-se a métrica RMSE (*Root Mean Square Error*; Raiz do Erro Quadrático médio) e verifica-se caso o algoritmo converge para uma solução aceitável. Essa variável representa a raiz da distância média quadrática dos pontos alinhados pelo ICP e informa sobre a qualidade da sobreposição do algoritmo. Com este parâmetro é possível observar quantas iterações o algoritmo leva até a convergência da estimativa e caso atinja um nível aceitável de sobreposição (alinhamento).

O equacionamento do RMSE é da forma:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{q}_i - \mathbf{p}_i)^2}, \tag{55}$$

tendo que N é a quantidade de pontos da nuvem de entrada $\mathbf{q}$ a qual representa o escaneamento atual e $\mathbf{p}$ é a nuvem de pontos do escaneamento anterior.

De acordo com Sturm *et al.* (2012), é possível utilizar a métrica ATE (*Absolute Trajectoy Error*) para verificar o erro entre as trajetórias geradas pelas duas abordagens de localização e computar as diferenças de poses absolutas. Para o cálculo dos parâmetros ATE, compara-se uma sequência de poses estimadas pela odometria visual P_1, ..., P_n com as respectivas poses fornecidas pela odometria laser Q_1, ..., Q_n em relação a cada parâmetro estimado (x, y, θ). Esta comparação somente é possível após sincronizar as trajetórias, ou seja, aplicar a reamostragem

na trajetória Aruco para equiparar com as amostras da trajetória ICP.

4 RESULTADOS

Neste capítulo são apresentados as coletas de dados do experimento, as trajetórias estimadas pela Odometria visual e Odometria LiDAR, ou seja, a geração do *ground thruth* (pose real) e a saída do algoritmo ICP, em seguida, serão expostos gráficos do erro RMSE e do erro ATE o qual será apresentado em relação à cada parâmetro (x, y, θ) da saída do algoritmo implementado comparado com a abordagem visual. Em todas as métricas são utilizados os módulos dos erros em relação ao *ground truth*. Em seguida, mostra-se as tabelas de tempo de execução da chamada (i, j) do algoritmo ICP para alinhar a nuvem de pontos i com a j, sendo estas correspondentes a escaneamentos sequenciais da captura do experimento.

4.1 Experimento e pré tratamento dos dados

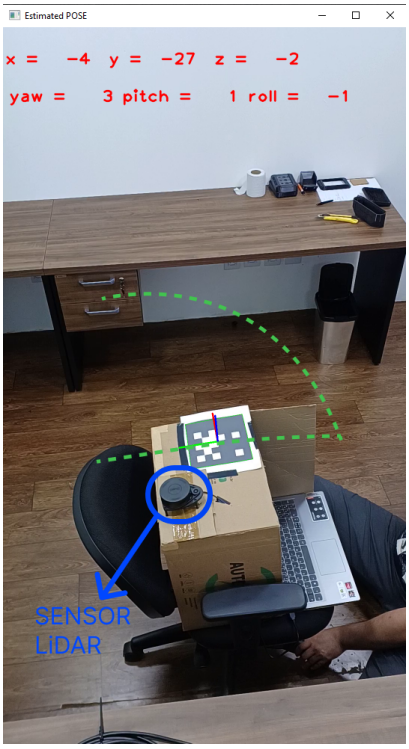
A Figura 21 representa um frame do vídeo do experimento gravado pela câmera do celular juntamente aos parâmetros de localização estimados pela abordagem visual (Biblioteca Aruco; parâmetros em vermelho) e a aproximação da trajetória efetuada no experimento (verde pontilhado). Como era necessário um cabo estar conectado ao sensor LiDAR para capturar os dados, a estrutura da base do sensor consiste numa caixa de papelão em cima de uma cadeira com rodas a qual é movimentada pelo acadêmico.

O teste durou 19,5 segundos e o vídeo e escaneamentos serão disponibilizados online para livre acesso. Para as finalidades deste trabalho, faz-se que o sistema absoluto de coordenadas possui a origem na posição inicial do sensor LiDAR. Desta forma, é possível comparar corretamente as trajetórias de acordo com a métrica ATE.

É evidente que o fato do sensor LiDAR operar na frequência de 10Hz e a câmera fazer capturas de 30 quadros por segundo (30Hz) tem-se uma questão de sincronização de amostras e precisa-se adequar os dados para o cálculo da métrica ATE. A Figura 22 mostra a reamostragem aplicada aos parâmetros estimados pela odometria visual (somente os parâmetros relevantes para a comparação ATE). Para efetuar a reamostragem, foi usada função *resample* da biblioteca *scipy*.

Ao observar os gráficos da Figura 22 percebe-se de início uma inconsistência na estimativa do parâmetro θ da orientação do objeto próximo ao final do experimento (entre $t \approx 15$ e $t \approx 17$) (Figura 22). Acredita-se que este comportamento ruidoso é devido à distância do marcador AruCo até a câmera, pois nos momentos finais do experimento o rótulo AruCo está em movimento e na parte da trajetória mais longínqua em relação à câmera. Nota-se que

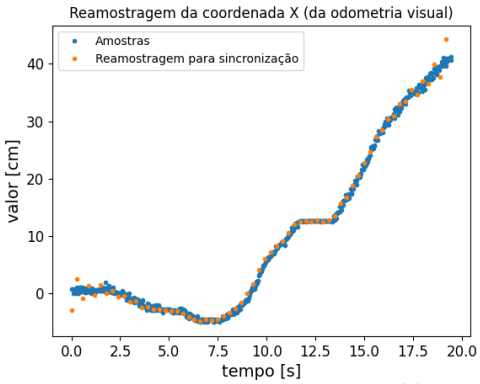
Figura 21 – Imagem tirada do vídeo do experimento gravado pelo celular



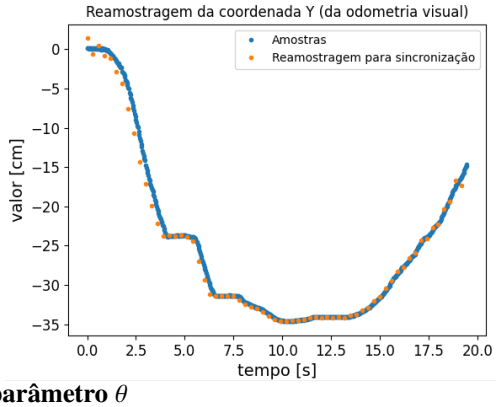
Fonte: Autoria própria (2023).

Figura 22 – Reamostragem dos parâmetros para compor a trajetória 2D AruCo

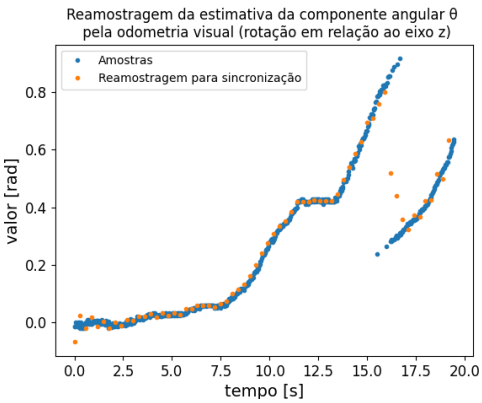
(a) Reamostragem do parâmetro x



(b) Reamostragem do parâmetro y



(c) Reamostragem do parâmetro θ



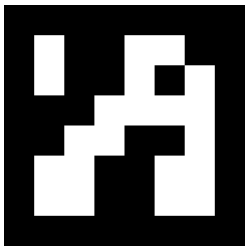
Fonte: Autoria própria (2023).

nos últimos momentos do experimento ($t > 17$) as estimativas retornam ao comportamento devido, menciona-se que nestes instantes o marcador está diminuindo de velocidade até entrar em repouso.

4.2 Validação por odometria visual

O processamento visual proposto consiste na utilização de marcadores ARUCO para localizar o objeto. Na base do sensor LiDAR é colocado uma folha com um marcador ARUCO impresso. Este rastreador é bem conhecido pela comunidade de processamento de imagens e é utilizado para detectar vários tipos de objetos por meio de imagens de câmeras (Blachut *et al.*, 2022). Um marcador ARUCO típico é composto por vários quadrados pretos e brancos configurados na área de um quadrado maior como apresentado na Figura 23. Os quadrados pequenos definem o "ID" do registrador o qual é um número inteiro que identifica o respectivo marcador. Neste trabalho utiliza-se um marcador ARUCO de dimensões 8×8 *cm* e ID = 24. É importante informar as dimensões do registrador pois permite que as distâncias expressas em pixels na imagem da câmera possam ser transformadas em coordenadas tridimensionais do espaço 3D euclidiano (domínio das imagens para coordenadas reais).

Figura 23 – Marcador Aruco



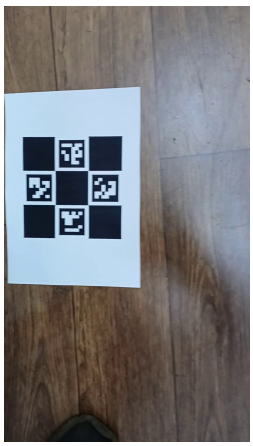
Fonte: (Itseez, 2015).

4.2.1 Calibração AruCo

Para que o processamento visual ocorra corretamente, é necessário calibrar a câmera utilizada. No experimento desta monografia foi proposto o uso do chamado quadro CHARUCO, disponível na biblioteca OpenCV (Itseez, 2015). Este quadro pode ser observado na Figura 24 e é uma das abordagens possíveis para a calibração da câmera. O objetivo da calibração é obter os coeficientes de distorção da câmera e remover as deformidades intrínsecas da lente da câmera para determinar com mais precisão a detecção e localização dos marcadores (Blachut *et al.*,

2022).

Figura 24 – Imagem retirada do vídeo de calibração Aruco



Fonte: Autoria própria (2023).

4.2.2 Detecção AruCo

Após correção da distorção da câmera, utiliza-se a função "*detectMarkers*" da biblioteca OpenCV, a qual detecta a quantidade e os cantos dos rótulos ARUCO presentes na imagem de entrada. Após detectados, executa-se a função "*estimatePoseSingleMarkers*" para determinar a posição e orientação do marcador no sistema de coordenadas associado à câmera (ou seja, a origem do sistema é a lente da câmera). A função "estimatePoseSingleMarkers" retorna um vetor com as coordenadas do centro do rótulo ARUCO juntamente aos ângulos euclidianos que determinam a orientação do marcador.

Para os objetivos deste trabalho, o essencial é guardar a trajetória feita no plano do sensor (no espaço de configurações $\mathbb{R}^2 \times \mathbb{S}^1$), porém a saída do processamento visual é no espaço tridimensional euclidiano o qual é representado por 6 parâmetros ($\mathbb{R}^3 \times \mathbb{S}^3$ — coordenadas 3D e 3 ângulos de Euler). Para comparar as trajetórias das duas abordagens propostas é necessário tratar os dados da localização dos marcadores tanto em relação à dimensão quanto à transformação de sistema de coordenadas. Para isto, utiliza-se os conceitos discutidos na secção cinemática de corpo rígido e será detalhado na comparação dos resultados.

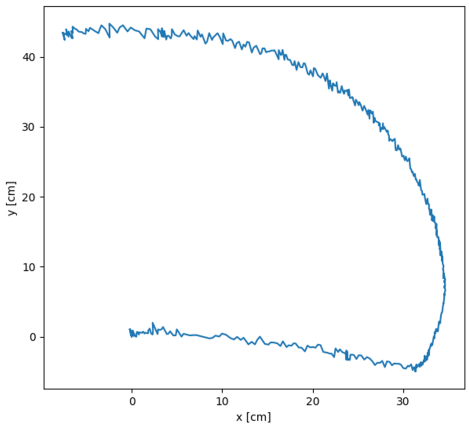
Essencialmente, o módulo visual gera uma trajetória representativa do movimento do marcador pois é capaz de localizar, em cada frame (imagem) o rótulo ARUCO em relação a lente da câmera. Isto implica que os parâmetros estimados pela detecção estão no sistema de coordenadas o qual possui a origem na lente da câmera. Entretanto, para gerar a trajetória do marcador com o sistema de coordenadas absoluto localizado na posição inicial do sensor, é

preciso aplicar a matriz de transformação homogênea a qual transforma as coordenadas em relação a câmera para coordenadas em relação a posição inicial.

O código para a odometria visual foi desenvolvido em linguagem Python utilizando a a biblioteca OpenCV juntamente a um de seus pacotes de processamento de imagens intitulado de detecção ArUco (Rafael Muñoz, 2014) . A linguagem Python e a plataforma OpenCV facilitam a implementação das transformações de sistemas de coordenadas e a visualização dos dados coletados (imagens da câmera, dados do sensor LiDAR, as transformações estimadas, entre outros dados necessários para o desenvolvimento do trabalho).

Por meio das imagens da câmera e da ferramenta de detecção AruCo, é possível localizar o marcador visual em cada frame de imagem da câmera. Com essa informação de localização, ou seja, a posição 3D e as rotações elementares, pode-se construir a trajetória feita pelo marcador. A Figura 25 apresenta a trajetória do rótulo Aruco projetada no plano 2D, paralelo a superfície do marcador, no instante t=0. Isto é possível aplicando a matriz de transformação homogênea que muda o sistema de coordenadas o qual antes tinha a origem no centro da lente da câmera para a posição inicial do rótulo AruCo. Deste modo, pode-se apresentar a trajetória já em relação sistema de coordenadas absoluto escolhido.

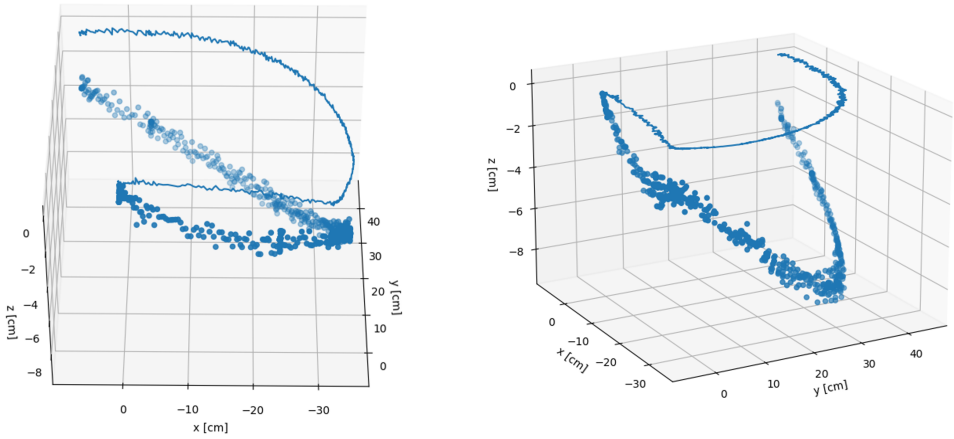
Figura 25 – Plotagem da trajetória 2D da localização estimada pela detecção AruCo



Fonte: Autoria própria (2023).

Semelhantemente, a Figura 26 apresenta em pontos azuis a trajetória do marcador em 3 dimensões (dado bruto da detecção AruCo) juntamente a projeção 2D da trajetória no plano do sensor LiDAR. Na Figura 26 fica evidente que existe ruído na estimação da coordenada “z” com amplitude máxima de 8cm, porém é um erro aceitável para a estimação desta coordenada.

Figura 26 – Plotagens da localização estimada pela detecção AruCo



Fonte: Autoria própria (2023).

4.3 Resultados da odometria LiDAR

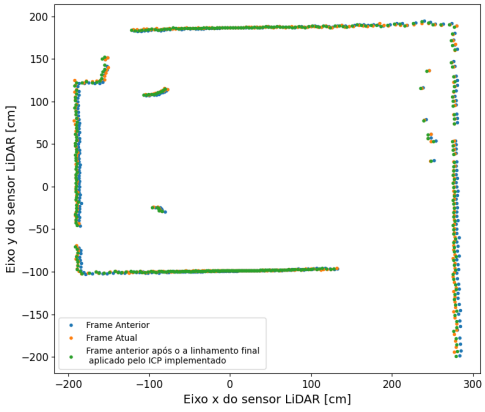
O código do algoritmo ICP foi inicialmente simulado na máquina do aluno (Quadro 1) em linguagem C++ e utilizando a biblioteca de álgebra linear Eigen (Quadro 2). A escolha da linguagem e biblioteca de operações matriciais foi motivada pela simplicidade na codificação e pelo suporte online de tutoriais. A ferramenta da simulação é imprescindível para os projetos de engenharia e testar a implementação em um computador com bons recursos de processamento antes de implementar a versão para o microcontrolador foi essencial para verificar o funcionamento e calibrar o algoritmo ICP.

Como o experimento durou 19,5 segundos e o sensor LiDAR opera em 10Hz, tem-se 195 escaneamentos sequenciais. Estes escaneamentos também podem ser chamados de frame, pois capturam o estado do ambiente em um instante t. Para os fins deste trabalho, cada chamada do ICP utiliza como entrada um frame k e o frame $k+3$, desta maneira, será intitulada como chamada $(k, k+3)$, tendo que quando $k = 0$, caracteriza a primeira chamada com os frames (0, 3) .

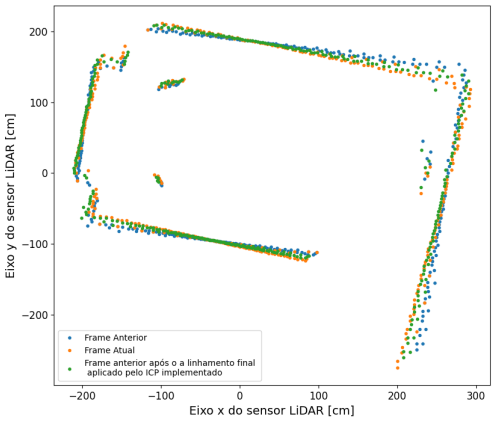
As Figuras 27.(a), 27.(b) e 27.(c) representam as entradas do algoritmo ICP em alguns momentos do experimento e o alinhamento final efetuado pela estimativa da transformação ideal a qual minimiza o erro quadrático $e(\overrightarrow{p_L}, \mathbf{R}_L)$ apresentado no Capítulo 2. Nestas figuras observa-se que a abordagem é capaz de alinhar as nuvens de entrada correspondentes ao experimento, entretanto atenta-se que existe um erro quadrático médio (RMSE) de aproximadamente 1cm entre as nuvens após o alinhamento final do ICP para algumas entradas.

Figura 27 – Plotagens do alinhamento efetuado pelo algoritmo ICP: (a) Plotagem das nuvens (48, 51) de entrada do algoritmo e a transformação final aplicada pelo ICP, (b) Plotagem das nuvens (90, 93) de entrada do algoritmo e a transformação final aplicada pelo ICP, (c) Plotagem das nuvens (177, 180) de entrada do algoritmo e a transformação final aplicada pelo ICP

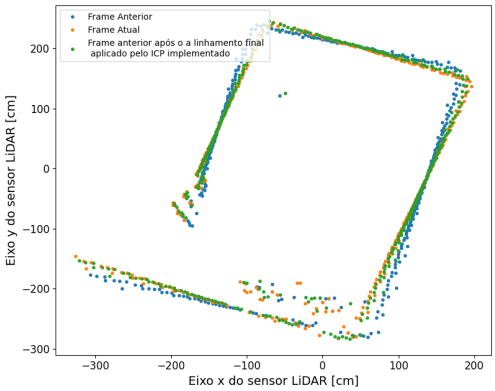
(a) Plotagem das nuvens (48, 51) de entrada do algoritmo e a transformação final aplicada pelo ICP



(b) Plotagem das nuvens (90, 93) de entrada do algoritmo e a transformação final aplicada pelo ICP



(c) Plotagem das nuvens (177, 180) de entrada do algoritmo e a transformação final aplicada pelo ICP



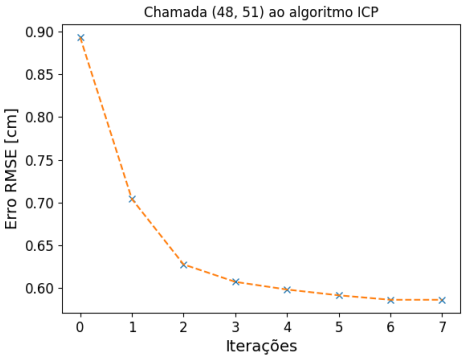
Fonte: Autoria própria (2023).

Para avaliar o funcionamento do algoritmo ICP utiliza-se a métrica RMSE e verifica-se caso o algoritmo converge para uma solução aceitável. Nos gráficos da Figura 28 é possível observar o comportamento do erro RMSE ao longo das iterações do algoritmo ICP. Nota-se que o algoritmo necessita de mais iterações para convergir quando existe uma rotação entre os frames, adicionalmente, quando há rotação, o erro RMSE estabiliza próximo à 1cm, porém, isto é devido ao fato do movimento de rotação gerar mais ruídos na entrada do que o movimento de translação. Tendo em mente que o movimento do objeto causa o surgimento de novos pontos nos escaneamentos, este erro é esperado. Evidentemente, os novos pontos não terão uma sobreposição exata com os pontos anteriores, visto que não estavam presentes no frame anterior.

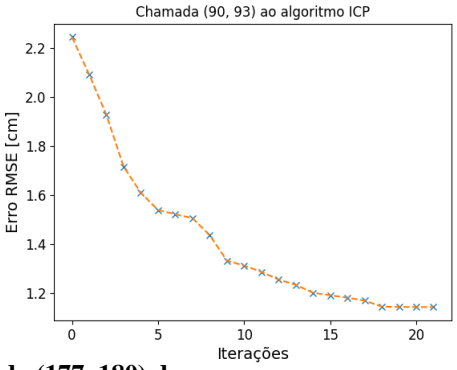
Em sequência, apresenta-se na Figura 29 a trajetória estimada pela odometria LiDAR, a qual é construída por meio das poses relativas calculadas pelo algoritmo ICP. Nesta imagem,

Figura 28 – Gráficos do erro RMSE em diferentes chamadas do algoritmo ICP: (a) Métrica RMSE na chamada (48, 51) do algoritmo ICP, (b) Métrica RMSE na chamada (90, 93) do algoritmo ICP, (c) Métrica RMSE na chamada (177, 180) do algoritmo ICP

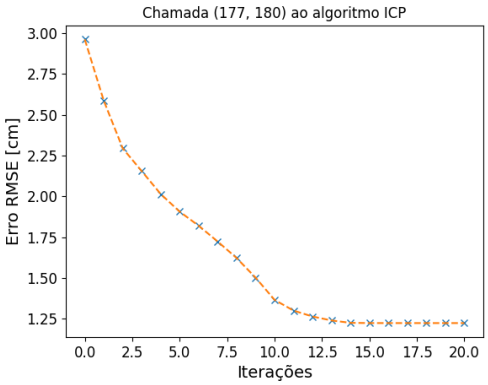
(a) Métrica RMSE na chamada (48, 51) do algoritmo ICP



(b) Métrica RMSE na chamada (90, 93) do algoritmo ICP



(c) Métrica RMSE na chamada (177, 180) do algoritmo ICP



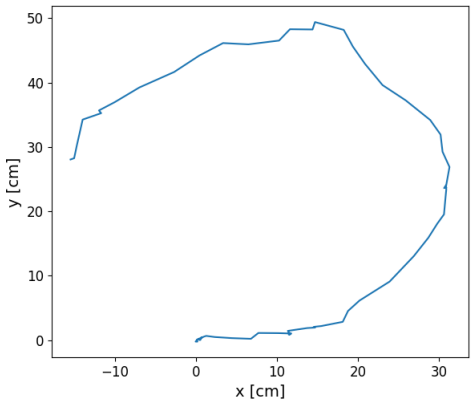
Fonte: Autoria própria (2023).

nota-se uma clara discrepância com a trajetória da odometria visual, porém, é um resultado condizente com o esperado para uma abordagem sem a aplicação de filtros, ou fusão sensorial. Como o algoritmo efetua um bom alinhamento das nuvens, é coerente afirmar que a incrementação de um filtro tal como o filtro estendido de Kalman possa ser capaz de ajustar as estimativas da solução (Cruz Júnior *et al.*, 2021).

Entretanto, uma das possíveis causas para o erro observado é o cálculo da translação final do algoritmo ICP. Os escaneamentos não possuem a mesma quantidade de pontos, logo, como utiliza-se o centro de massa das nuvens para estimação da translação inicial e final, a abordagem diverge da realidade do movimento.

Por final, expõe-se as tabelas de tempo de execução do algoritmo ICP, fazendo uma comparação entre o tempo de execução no microcontrolador e na máquina do acadêmico. O intuito da Tabela 1 é apresentar, primeiramente, a existência da diferença entre tempos de execução do algoritmo quando existe uma rotação entre os frames de entrada como nos frames (90, 93) e (177, 180) e quando não há: frames (0, 3), (3, 6), (48, 51).

Figura 29 – Plotagem da trajetória gerada pelas estimações do algoritmo ICP



Fonte: Autoria própria (2023).

A condição mínima para a aceitabilidade do algoritmo ICP em relação ao tempo de execução é a capacidade de processar dois frames consecutivos antes de escaneamentos novos serem efetuados. No caso do funcionamento em 10Hz do sensor LiDAR, o processamento dos dados deve ocorrer em menos de 100ms, entretanto, como os frames foram comparados na forma $(k, k+3)$, define-se o tempo máximo de execução do algoritmo ICP como 300ms. Logo, nota-se pela Tabela 1 que o algoritmo não atende ao parâmetro de engenharia definido para uma aplicação de localização.

Tabela 1 – Tabela da comparação do tempo de execução entre a máquina pessoal e o microcontrolador STM32F407xx.

Frames processados	Tempo de execução no PC [ms]	Tempo de execução no micro. [ms]
(0, 3)	149	21168
(3, 6)	151	21166
(48, 51)	151	21171
(90, 93)	415	55996
(177, 180)	449	53513

Fonte: Autoria própria (2023).

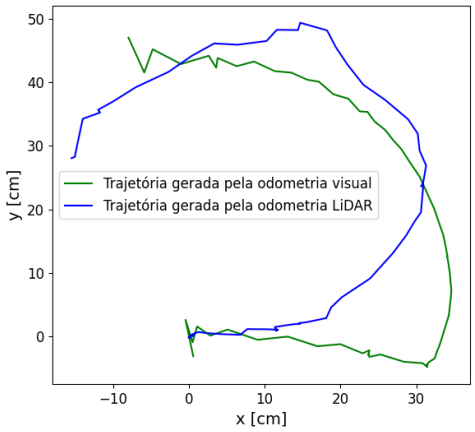
4.4 Comparação das abordagens

É uma tarefa difícil efetuar a comparação exaustiva de abordagens para a localização, pois pequenas variações nos parâmetros de convergência podem levar a melhorias de performance, porém para casos limitados. Consequentemente, limita-se a avaliação da comparação das trajetórias em uma variável apenas: a ATE (*Absolute Trajectory Error*), apresentada no Capítulo 3. É importante salientar que para o cenário bidimensional a variável ATE possui 3 dimensões: o

erro em relação à coordenada x, y e a orientação θ.

Entretanto, antes é preciso visualizar as trajetórias geradas pelas abordagens. A Figura 30 expõe, no sistema de coordenadas absoluto, as duas trajetórias estimadas. Em seguida, a Figura 31 apresenta as mesmas trajetórias juntamente às distâncias translacionais de cada pose comparada. Nota-se que existe uma certa semelhança nas trajetórias geradas, mas as estimações da translação ICP não condizem com o movimento real do sensor laser no experimento. É importante salientar que este resultado é semelhante aos testes do trabalho de Cota (2019), nos quais as abordagens que processam dados de apenas um sensor (*encoder*, LiDAR e câmera RGB-D) divergem do *groundtruth*. Outro resultado semelhante ao presente trabalho é o de Silva do Monte Lima, Uchiyama e Taniguchi (2019), no qual foi desenvolvido um sistema para processamento de dados de um sensor IMU. Entretanto, apesar das camadas de processamento da abordagem, os ruídos de leitura do sensor IMU levam o sistema à estimar poses relativas imprecisamente. Logo, os erros intrínsecos ao aparelho de medição impulsionam o uso de outros sensores ou mais camadas de processamento para atingir uma acurácia melhor.

Figura 30 – Plotagem das duas trajetórias geradas pelas abordagens exploradas

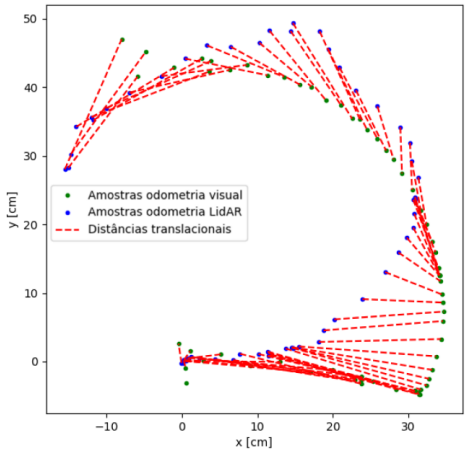


Fonte: Autoria própria (2023).

Desta forma, a Figura 32 apresenta-se os gráficos do erro absoluto ao longo do tempo em relação a cada parâmetro da pose estimada pelo algoritmo ICP. Logo, é possível investigar as estimações ICP de maneira relativa à odometria visual. Consequentemente, ao examinar os gráficos da Figura 32, acredita-se que o erro notado nos instantes de $t \approx 13{,}0s$ à $t \approx 17{,}5s$ do parâmetro θ (Na Figura 32.(c)) seja devido à falha que ocorre na estimativa AruCo durante aproximadamente o mesmo intervalo e pode ser observada na Figura 22.(c).

As estimativas do ICP em relação ao movimento translacional (Figura 32.(a) e Figura 32.(b)), por sua vez, apresentam estimativas as quais não corroboram com a trajetória real do

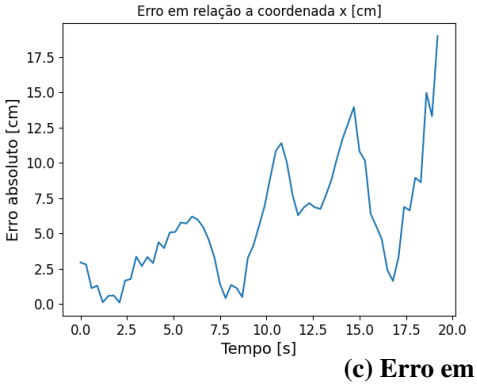
Figura 31 – Plotagem das distâncias entre as amostras das duas trajetórias geradas pelas abordagens exploradas



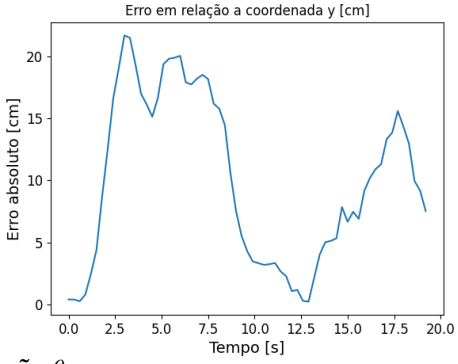
Fonte: Autoria própria (2023).

Figura 32 – Gráficos dos parâmetros do erro ATE: (a) Erro em relação à coordenada x, (b) Erro em relação à coordenada y, (c) Erro em relação à orientação θ

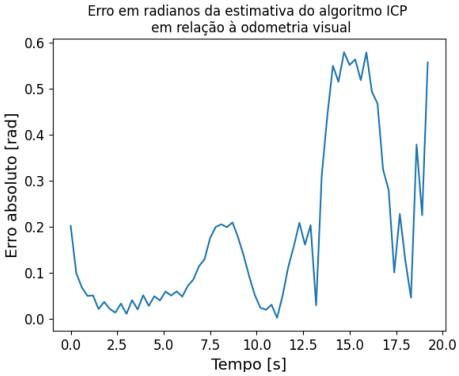
(a) Erro em relação à coordenada x



(b) Erro em relação à coordenada y



(c) Erro em relação à orientação θ



Fonte: Autoria própria (2023).

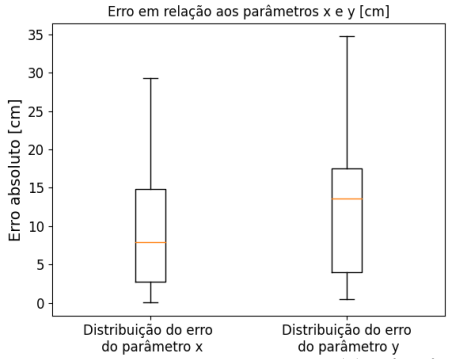
experimento. Este fenômeno é explorado com mais profundidade na próxima subsecção, porém acredita-se que o erro translacional é devido a desuniformidade da distribuição dos pontos nas nuvens de entrada.

Para uma análise mais profunda dos parâmetros ATE, foram gerados os gráficos da Figura 33, com estas visualizações, pode-se observar o valor máximo, mínimo, a mediana, o

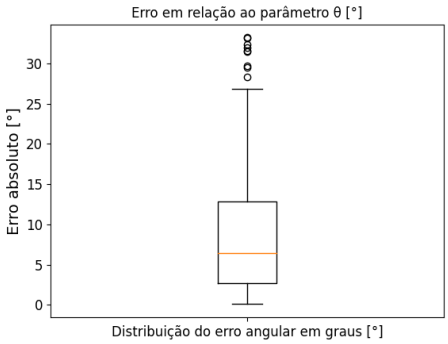
desvio padrão e os *outliers* de cada dimensão do erro ATE. O erro máximo do parâmetro θ é de 0,62 radianos (36°), entretanto a média é de 0,104 radianos (6°), o qual caracteriza um erro admissível para este parâmetro. Entretanto, os parâmetros x e y não apresentam erros médios aceitáveis e caracterizam uma inconsistência do sistema desenvolvido.

Figura 33 – Gráficos das distribuições do erro de cada parâmetro da métrica ATE: (a) Distribuiçaõ do erro em relação às coordenadas x e y, (b) Distribuição do erro em relação ao parâmetro θ em graus °, (c) Distribuição do erro em relação ao parâmetro θ em radianos

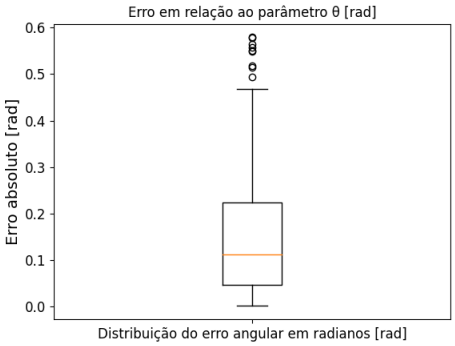
(a) Distribuiçaõ do erro em relação às coordenadas x e y



(b) Distribuição do erro em relação ao parâmetro θ em graus °



(c) Distribuição do erro em relação ao parâmetro θ em radianos



Fonte: Autoria própria (2023).

Os *outliers* observados nas Figuras 33.(b) e 33.(c) evidencia e corrobora com a hipótese do comportamento inesperado do módulo da estimativa AruCo nos momentos finais do experimento, indicando um mal funcionamento temporário do módulo AruCo. Esse comportamento pode ser devido às condições de iluminação do ambiente ou à distância entre o marcador e a câmera.

4.5 Discussões

Um dos desafios do presente trabalho foi a concepção do experimento, pois, sem a estrutura de um robô, é difícil tanto simular o movimento quanto realizar o posicionamento de todos os aparatos necessários: o notebook, o sensor LiDAR com o cabo de dados e o celular. A maior dificuldade foi em relação ao cabo de dados do sensor laser, pois, caso o cabo do sensor for

capturado pelas leituras laser, isto implica um ambiente dinâmico, o qual não possui tratamento na implementação desenvolvida. Apesar do acadêmico ser responsável pelo movimento da base do sensor, como não há intersecção do aluno com o plano de escaneamento do aparato, não houve dinamicidade no ambiente do experimento.

Ao investigar o funcionamento do algoritmo ICP nota-se que a abordagem sobrepõe as nuvens de pontos de entrada, porém não estima corretamente a translação do objeto em questão. Isto é devido ao fato da translação ser calculada por meio do centro de massa dos pontos e as nuvens de ponto não possuem densidade uniforme de uma leitura para outra. Esse fenômeno fica notável quando o sensor está próximo de uma parede e longe da outra; os pontos mais próximos do sensor serão mais densos e os pontos longínquos estarão mais espaçados (Figura 27). Isto leva o algoritmo à estimar uma translação que não corresponde ao movimento real do objeto móvel. O centroide é uma boa estimação inicial, mas ser utilizado para o alinhamento final não condiz com a realidade das amostras de pontos deste trabalho.

Em relação a geração da trajetória do algoritmo ICP, visto o alinhamento das entradas do algoritmo, a estimação condiz com as expectativas de uma implementação sem fusão sensorial e filtros probabilísticos. A abordagem desenvolvida captura o formato (geometria) do deslocamento do objeto, porém, nota-se que a desuniformidade das nuvens de pontos de entrada interfere na estimação da translação, e consequentemente, leva o algoritmo a gerar uma trajetória incoerente em relação ao movimento real.

Durante o desenvolvimento do algoritmo ICP, notou-se uma clara dificuldade na questão do tempo de execução do código. Apesar da escolha da linguagem C++ e biblioteca matemática Eigen ser direcionada à performance do código, a implementação não atende às necessidades mínimas de tempo de execução de uma aplicação de localização. Isto ocorre pois, quando existe uma rotação significativa no agente robótico, o algoritmo precisa de mais iterações para convergir. Proceduralmente, ao investigar as etapas do algoritmo no quesito de tempo de execução, identificou-se que a função mais custosa é a de encontrar correspondências (pela distância ponto a ponto), por ter a complexidade $O(n^2)$, sendo n a quantidade de pontos da nuvem de entrada.

Para melhorar o tempo de execução pode-se implementar a busca KD-tree para encontrar rapidamente as correspondências. Com esta abordagem é possível encontrar os pontos mais próximos de maneira mais eficiente, diminuindo a complexidade da busca para $O(log(n))$ (Yang *et al.*, 2015).

Uma outra questão que surgiu durante o trabalho foi sobre o desempenho e a otimização da biblioteca Eigen para sistemas embarcados. Apesar dessa ferramenta poder ser utilizada

em vários cenários, não foi a escolha correta para o microcontrolador STM32F407. Logo, é preciso verificar outras alternativas de bibliotecas de álgebra para tornar a implementação viável. Entretanto, salienta-se que a biblioteca Eigen foi escolhida pois ela pode ser embarcada e também é amplamente usada dentro de outras plataformas como a OpenCV, ROS, TensorFlow, entre outras.

5 CONCLUSÃO

Duas abordagens para a localização foram exploradas nesse trabalho, na abordagem visual foi possível obter com precisão a localização (posição e orientação) do marcador, enquanto na abordagem ICP apenas a estimação da orientação apresentou resultados coerentes com o movimento do objeto. Visto que a estimativa da translação não é capaz de tratar a desuniformidade dos dados, os resultados da trajetória gerada pela odometria LiDAR divergem consideravelmente do movimento real do objeto, entretanto, isto era esperado do sistema implementado, desta maneira, apesar de não ser adequado às necessidades mínimas de uma solução completa de localização, é um passo fundamental do projeto.

Uma estratégia possível para tratar a desuniformidade dos dados LiDAR é representar a nuvem de pontos por meio de uma equação de superfície, e desta maneira, alinhar as superfícies as quais representam os escaneamentos sequenciais (Holz; Behnke, 2014). A partir desta equação de superfície é admissível assumir uma distribuição uniforme e realizar a operação do centro de massa. Outra solução é utilizar a estrutura e o modelo cinemático do robô de duas rodas juntamente ao método de filtragem apropriado para corrigir as estimativas incoerentes.

Em relação ao tempo de execução, a implementação não atende às necessidades de projeto, pois não seria capaz de gerar a transformação rígida antes do próximo escaneamento chegar na memória do microcontrolador. Isto ocorre tanto para a versão implementada em PC quanto para a do microcontrolador. Uma das propostas para trabalhos futuros é utilizar outra biblioteca de álgebra linear, mais específica para o cenário embarcado, pois acredita-se que a biblioteca Eigen, apesar de compatível, não é otimizada o suficiente para o processamento em baixo nível.

Entende-se que a implementação foi capaz de sobrepor corretamente as nuvens de pontos e essa habilidade representa o primeiro passo para a construção de um mapa do ambiente. A partir dos dados de alinhamento, é possível registrar os pontos que representam o cenário, de modo que, somente os pontos com o melhor alinhamento serão registrados como pontos do mapa.

Visto os desafios e limitações do projeto desenvolvido, propõe-se para trabalhos futuros a adição do filtro estendido de Kalman para corrigir as estimativas do algoritmo ICP, a migração das operações matemáticas para uma biblioteca mais eficiente e direcionada à performance, e também o desenvolvimento do robô com duas rodas para realizar os testes e aplicar a modelagem cinemática apresentada no Capítulo 2.

REFERÊNCIAS

BENOÎT JACOB, G. **Eigen**. 2021. Disponível em: https://eigen.tuxfamily.org/index.php?title=Main_Page.

BESL, P. J.; MCKAY, N. D. A Method for Registration of 3-D Shapes. **IEEE Trans. Pattern Anal. Mach. Intell.**, v. 14, p. 239–256, 1992.

BLACHUT, K. *et al.* Automotive Perception System Evaluation with Reference Data from a UAV's Camera Using ArUco Markers and DCNN. **Journal of Signal Processing Systems**, Springer, v. 94, n. 7, p. 675–692, 2022.

BRAGA, J. R. G. **Navegação autônoma de VANT por imagens *LiDAR***. 2016. 307 f. Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada – Programa de Pós Graduação em Computação Aplicada, Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, SP, 2016. Disponível em: https://bdtd.ibict.br/vufind/Record/INPE_6c3da09faadf55b8a428896dcb51afc1. Acesso em: 5 dez. 2022.

CAO, Y. *et al.* Indoor SLAM Algorithm Based on PL-ICP and Map Matching. In: 2021 1ST INTERNATIONAL CONFERENCE ON CONTROL AND INTELLIGENT ROBOTICS. **Proceedings [. . .].** 2021. P. 295–299.

COTA, E. **Implementação e avaliação de técnicas de odometria aplicadas a um dispositivo robótico móvel.** 2019. 109 f. Dissertação de Mestrado em Instrumentação, Controle e Automação de Processos de Mineração – Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto, Ouro Preto, MG, 2019. Disponível em: https://bdtd.ibict.br/vufind/Record/UFOP_f74380802d9c68adc8baa853557bf6d2. Acesso em: 5 dez. 2022.

CRUZ JÚNIOR, G. P. da *et al.* **Localização e mapeamento para robôs móveis em ambientes confinados baseado em fusão de *LiDAR* com odometrias de rodas e sensor inercial**. 2021. 132 f. Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 2021. Disponível em: https://bdtd.ibict.br/vufind/Record/UFMG_535e03f444fe3127caaca808ee2cec26. Acesso em: 5 dez. 2022.

DELLAERT, F. *et al.* Monte Carlo localization for mobile robots. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION. **Proceedings [. . .].** Detroit, MI, USA: IEEE, 1999. v. 2, 1322–1328 vol.2. DOI: 10.1109/ROBOT.1999.772544.

DISSANAYAKE, M. *et al.* A solution to the simultaneous localization and map building (SLAM) problem. **IEEE Transactions on Robotics and Automation**, v. 17, n. 3, p. 229–241, 2001. DOI: 10.1109/70.938381.

DUDEK, G.; JENKIN, M. **Computational Principles of Mobile Roboticss**. Cambridge University Press, 2010. 406 p. ISBN 9780521871570. Disponível em: https://books.google.com.br/books?id=FpnWjgEACAAJ.

GEROMICHALOS, D. *et al.* SLAM for autonomous planetary rovers with global localization. eng. **Journal of field robotics**, Wiley Subscription Services, Inc, Hoboken, v. 37, n. 5, p. 830–847, 2020. ISSN 1556-4959.

GOMES, D. d. F. **Mapas auto-organizáveis de topologia variante no tempo para SLAM visual em espaço de aparências**. 2015. 114 f. Tese (Doutorado em Inteligência Computacional) – Programa de Pós Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Campinas, SP, mar. 2015. Disponível em: https://bdtd.ibict.br/vufind/Record/UFPE_6c5890610312cae119c2f23d0cb76ad0. Acesso em: 5 dez. 2022.

GUIZILINI, V. C. **Localização e mapeamento simultâneos com auxílio visual omnidirecional.** 2008. 76 f. Dissertação de Mestrado Acadêmico em Engenharia de Controle e Automação Mecânica – Faculdade de Engenharia de Controle e Automação Mecânica, Escola Politécnica da Universidade de São Paulo, São Paulo, SP, 12 ago. 2008. DOI: https://doi.org/10.11606/D.3.2008.tde-17082009-095800. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3152/tde-17082009-095800/pt-br.php. Acesso em: 5 dez. 2020.

GUREL, C. S. **REAL-TIME 2D AND 3D SLAM USING RTAB-MAP, GMAPPING, AND CARTOGRAPHER PACKAGES**. Ago. 2018. DOI: 10.13140/RG.2.2.14901.99049.

HOLZ, D.; BEHNKE, S. Registration of non-uniform density 3D point clouds using approximate surface reconstruction. In: 41ST INTERNATIONAL SYMPOSIUM ON ROBOTICS. **ISR/Robotik 2014**. 2014. VDE, p. 1–7.

HUANG, S.; HUANG, H. A Frame-to-Frame Scan Matching Algorithm for 2D Lidar Based on Attention. **Applied Sciences**, v. 12, n. 9, 2022. ISSN 2076-3417. DOI: 10.3390/app12094341. Disponível em: https://www.mdpi.com/2076-3417/12/9/4341.

ILCI, V.; TOTH, C. High Definition 3D Map Creation Using GNSS/IMU/LiDAR Sensor Integration to Support Autonomous Vehicle Navigation. **Sensors**, v. 20, n. 3, 2020. ISSN 1424-8220. DOI: 10.3390/s20030899. Disponível em: https://www.mdpi.com/1424-8220/20/3/899.

ITSEEZ. **Open Source Computer Vision Library**. 2015. https://github.com/itseez/opencv.

KANG, J. M. *et al.* An enhanced map-matching algorithm for real-time position accuracy improvement with a low-cost GPS receiver. **Sensors**, MDPI, v. 18, n. 11, p. 3836, 2018.

KHAN, M. *et al.* Investigation of Widely Used SLAM Sensors Using Analytical Hierarchy Process. **Journal of Sensors**, v. 2022, p. 1–15, jan. 2022. DOI: 10.1155/2022/5428097.

LABBE, M.; MICHAUD, F. Online global loop closure detection for large-scale multi-session graph-based SLAM. In: 2014 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. **Proceedings [...].** 2014. IEEE, p. 2661–2666.

LABORATORY, S. A. I. **Robotic Operating System**. 23 mai. 2018. Disponível em: https://www.ros.org.

LI, F. *et al.* Real-Time 2-D Lidar Odometry Based on ICP. **Sensors**, v. 21, n. 21, 2021. ISSN 1424-8220. DOI: 10.3390/s21217162. Disponível em: https://www.mdpi.com/1424-8220/21/21/7162.

LOW, K.-L. Linear least-squares optimization for point-to-plane icp surface registration. **Chapel Hill, University of North Carolina**, Citeseer, v. 4, n. 10, p. 1–3, 2004.

LV, J. *et al.* Indoor Slope and Edge Detection by Using Two-Dimensional EKF-SLAM with Orthogonal Assumption. eng. **International journal of advanced robotic systems**, SAGE Publications, London, England, v. 12, n. 4, p. 44, 2015. ISSN 1729-8806.

RAFAEL MUÑOZ, S. **AruCo**. 2014. Disponível em: https://mecaruco2.readthedocs.io/en/latest/notebooks_rst/Aruco/Aruco.html.

SANZ, P. Robotics: Modeling, planning, and control (siciliano, b. et al; 2009)[on the shelf]. **IEEE Robotics & Automation Magazine**, IEEE, v. 16, n. 4, p. 101–101, 2009.

SILVA DO MONTE LIMA, J. P.; UCHIYAMA, H.; TANIGUCHI, R.-i. End-to-end learning framework for imu-based 6-dof odometry. **Sensors**, MDPI, v. 19, n. 17, p. 3777, 2019.

STMICROELECTRONICS, H. **STM32CubeIDE**. 2023. Disponível em: https://www.st.com/en/development-tools/stm32cubeide.html.

STURM, J. *et al.* A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. **Proceedings [. . .].** 2012. IEEE, p. 573–580.

TIAR, R.; LAKROUF, M.; AZOUAOUI, O. Fast ICP-SLAM for a bi-steerable mobile robot in large environments. In: 2015 IEEE INTERNATIONAL WORKSHOP OF ELECTRONICS, CONTROL, MEASUREMENT, SIGNALS AND THEIR APPLICATION TO MECHATRONICS (ECMSM). **Proceedings [. . .].** 2015. IEEE, p. 1–6.

WALLHOFF, F. *et al.* Surveillance and Activity Recognition with Depth Information. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO. **Proceedings [...]** 2007. P. 1103–1106. DOI: 10.1109/ICME.2007.4284847.

YAMAUCHI, B.; SCHULTZ, A.; ADAMS, W. Mobile robot exploration and map-building with continuous localization. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION. **Proceedings [. . .].** IEEE, 1998. v. 4, 3715–3720 vol.4. DOI: 10.1109/ROBOT.1998.681416.

YANG, H. *et al.* An improved iterative closest points algorithm. **World Journal of Engineering and Technology**, Scientific Research Publishing, v. 3, n. 03, p. 302, 2015.