UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JOSÉ SERGIO CRUZ DANTAS JUNIOR

SLAM DE UM ROBÔ MÓVEL TERRESTRE UTILIZANDO SENSOR LIDAR E ODOMETRIA COM FILTRO DE KALMAN ESTENDIDO

FORTALEZA

2023

JOSÉ SERGIO CRUZ DANTAS JUNIOR

SLAM DE UM ROBÔ MÓVEL TERRESTRE UTILIZANDO SENSOR LIDAR E ODOMETRIA COM FILTRO DE KALMAN ESTENDIDO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Fabrício Gonzalez Nogueira

Coorientador: Prof. Dr. Bismark Claure Torrico

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

D213s Dantas Júnior, José Sergio Cruz.
SLAM de um Robô Móvel Terrestre Utilizando Sensor LiDAR e Odometria com Filtro de Kalman Estendido / José Sergio Cruz Dantas Júnior. – 2023.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2023.
Orientação: Prof. Dr. Fabrício Gonzalez Nogueira.
Coorientação: Prof. Dr. Bismark Claure Torrico.

1. SLAM. 2. EKF. 3. Robô Móvel. 4. ROS. 5. LiDAR. I. Título.

CDD 621.3

JOSÉ SERGIO CRUZ DANTAS JUNIOR

SLAM DE UM ROBÔ MÓVEL TERRESTRE UTILIZANDO SENSOR LIDAR E ODOMETRIA COM FILTRO DE KALMAN ESTENDIDO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Fabrício Gonzalez Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bismark Claure Torrico (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Me. José Leonardo Nunes da Silva
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Me. Kaio Martins Ramos
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Aos amigos, pelo amparo proporcionado para a realização deste trabalho. Aos meus professores, por todo o conhecimento que me fora concebido para conclusão da minha graduação.

AGRADECIMENTOS

Ao Prof. Dr. Fabrício Gonzalez Nogueira por me orientar em minha pesquisa de inciação científica no Grupo de Pesquisa em Automação, Controle e Robótica (GPAR) e durante o trabalho de conclusão de curso.

À Jeová, por me guiar e me dar força ao longo deste percurso acadêmico. Sua infinita graça e orientação foram fundamentais para a conclusão deste trabalho. Sou grato pelas bênçãos e oportunidades que recebi e por Sua presença constante em minha vida.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Ao Doutorando José Raimundo de Oliveira Júnior e à Graduanda Luana Albuquerque de Oliveira por todo suporte e amparo prestado para o desenvolvimento deste trabalho.

Ao Prof. Dr. José Tarcísio Costa Filho pela orientação no Núcleo de Tecnologia e Qualidade Industrial do Ceará (Nutec) e pela disciplina ministrada de Robótica Móvel, que foi fundamental para a elaboração deste trabalho

Aos amigos de laboratório pelas discussões sobre os assuntos abordados no presente trabalho.

Aos meus pais e meu irmão, que sempre me apoiaram e me incentivaram ao longo de toda a minha jornada acadêmica!

E agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

O SLAM (Simultaneous Localization and Mapping) é uma área de pesquisa em robótica que visa capacitar um robô a construir um mapa do ambiente desconhecido em que está operando, ao mesmo tempo em que estima sua própria localização nesse mapa. Neste estudo, é proposta uma solução baseada na combinação de um sensor LiDAR (Light Detection and Ranging) e odometria filtrada por EKF (Extended Kalman Filter) com uma unidade de medição inercial (IMU), utilizando o ROS (Robot Operating System) para fazer a integração dos sensores. O LiDAR é um sensor óptico que permite ao robô medir distâncias e obter informações sobre o ambiente em forma de nuvem de ponto. A odometria, por sua vez, é uma técnica que utiliza as informações de movimento do robô para estimar sua trajetória. O sensor IMU é composto por acelerômetros e giroscópios, que medem a aceleração linear e a taxa de rotação do robô, respectivamente. O filtro de Kalman estendido (EKF) é um método estatístico que combina as informações provenientes dos sensores, permitindo uma estimativa precisa da posição e orientação do robô, bem como a construção do mapa do ambiente em tempo real. A abordagem estendida do filtro de Kalman é utilizada para lidar com a incerteza e as não linearidades presentes no SLAM. Para validar a proposta, foram realizados experimentos em um ambiente controlado, onde o robô móvel terrestre foi capaz de explorar o espaço, construir um mapa e estimar sua localização simultaneamente. Os resultados obtidos demonstram a eficácia do método proposto, fornecendo um desempenho preciso e robusto na tarefa de SLAM.

Palavras-chave: SLAM. EKF. Robô móvel. LiDAR. Odometria. ROS.

ABSTRACT

Simultaneous Localization and Mapping (SLAM) is a research area in robotics that aims to enable a robot to build a map of an unknown environment while simultaneously estimating its own location within that map. This study proposes a solution based on the combination of a LiDAR sensor (Light Detection and Ranging) and odometry combined with an extended Kalman filter using an inertial measurement unit (IMU), with integration facilitated by the Robot Operating System (ROS). The LiDAR sensor is an optical sensor that allows the robot to measure distances and obtain information about the environment in the form of a point cloud. Odometry, on the other hand, is a technique that uses the robot's motion information to estimate its trajectory. The IMU sensor consists of accelerometers and gyroscopes, which measure linear acceleration and rotational rate of the robot. The extended Kalman filter is a statistical method that combines the information from these sensors, enabling precise estimation of the robot's position and orientation, as well as real-time mapping of the environment. The extended Kalman filter approach is used to handle the uncertainty and non-linearities present in the SLAM problem. To validate the proposed solution, experiments were conducted in a controlled environment, where the mobile ground robot was able to explore the environment, build a map, and simultaneously estimate its location. The results obtained demonstrate the effectiveness of the proposed method, providing accurate and robust performance in the SLAM task.

Keywords: SLAM. EKF. Mobile robot. LiDAR. Odometry. ROS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DC	Direct Current
EKF	Extended Kalman Filter
GPS	Global Positioning System
IMU	Inercial Measurement Unit
LiDAR	Light Detection and Ranging
ROS	Robot Operating System
SLAM	Simultaneous Location and Mapping

LISTA DE SÍMBOLOS

V	Tensão
v_d	Velocidade angular da roda direita
v_e	Velocidade angular da roda esquerda
v	Velocidade linear do robô
ω	Velocidade angular do robô
d	Distância entre as rodas do robô
r	Raio das rodas do robô
Δx	Variação de posição no eixo x
Δy	Variação de posição no eixo y
$\Delta\theta$	Variação de tempo
x	Posição no eixo x
y	Posição no eixo y
k	Instante de tempo discreto
$\hat{x}_k$	Vetor de estado estimado no instante de tempo k
$\hat{x}_{k+1}$	Vetor de estado estimado no instante de tempo k+1
F	Matriz de transição de estado
B	Matriz de controle
u_k	Vetor de controle no instante de tempo k
P_k	Matriz de covariância do estado no instante de tempo k
P_{k-1}	Matriz de covariância do estado no instante de tempo k-1
Q_k	Matriz de covariância do processo no instante de tempo k
R	Matriz de covariância da medida
H	Matriz de transição da medida
K_k	Matriz de gano de Kalman
I	Matriz identidade
A_{k-1}	Matriz jacobiana das derivadas parciais da função de transição de estado em relação ao estado

H_{k-1}	Matriz jacobiana das derivadas parciais da função de observação em relação ao estado estimado
z_k	Medida observada no instante de tempo k
h	Função que relaciona o estado estimado às medidas esperadas
R_k	Matriz de covariância do ruído da medição

SUMÁRIO

1 INTRODUÇÃO

A localização e mapeamento simultâneos (SLAM) têm sido uma área de pesquisa ativa no campo da robótica móvel. Trata-se de um desafio fundamental, pois envolve a capacidade de um robô móvel de estimar sua própria localização e construir um mapa do ambiente desconhecido em tempo real de forma autônoma.

O SLAM desempenha um papel crucial em diversas aplicações, como navegação autônoma, inspeção de ambientes perigosos, exploração subaquática e mapeamento de espaços internos. A literatura científica apresenta uma série de estudos que abordam os temas cruciais no campo do SLAM, assim como os desafios relacionados à fusão de dados de sensores para obter estimativas precisas da posição e orientação de robôs móveis, além da criação de mapas coerentes do ambiente. Por exemplo, em um estudo realizado por Yang (2011), foi proposta a fusão de dados de câmeras e sensores lasers para modelagem de ambientes urbanos. Silva e Kondoz (2018) investigaram a fusão de sensores para detecção de objetos em aplicações de mapeamento móvel. Gao Yu-Xian Gai (2007) discutiram a fusão de sensores em SLAM para robôs móveis autônomos. Esses estudos contribuem significativamente para o avanço do conhecimento na área, fornecendo perspectivas e soluções para aprimorar a precisão e a confiabilidade do SLAM em diferentes cenários de aplicação.

Um dos principais desafios na implementação do SLAM é a fusão de dados de sensores e odometria (informações de deslocamento), para obter estimativas precisas da posição e orientação do robô, além da criação de um mapa coerente do ambiente.

Nesse contexto, o Extended Kalman Filter (EKF) é uma técnica amplamente utilizada para estimar o estado do robô móvel e construir o mapa simultaneamente, como abordado em Saman (2016). O EKF é uma extensão do Filtro de Kalman clássico que lida com a não-linearidade dos modelos e das observações, comuns em problemas de SLAM.

Este trabalho de TCC tem como objetivo o projeto e a implementação de um filtro de Kalman estendido em um robô móvel diferencial, com foco na aplicação em SLAM. A proposta é explorar a capacidade do filtro de Kalman estendido em lidar com as não-linearidades presentes no problema de localização e mapeamento simultâneos, permitindo ao robô móvel obter estimativas precisas de sua localização e construir um mapa coerente do ambiente em tempo real.

Ao investigar e implementar o filtro de Kalman estendido nesse contexto, espera-se contribuir para o avanço do conhecimento na área de SLAM e fornecer uma solução eficiente

para a localização e mapeamento de robôs móveis diferenciais. Os resultados obtidos neste estudo podem ser aplicados em diferentes cenários de robótica móvel, contribuindo para o desenvolvimento de sistemas autônomos mais eficientes e confiáveis.

Este trabalho seguirá uma estrutura que abordará a fundamentação teórica, explorando a estrutura física do robô, o funcionamento do LiDAR e do Inercial Measurement Unit (IMU), o Robot Operating System (ROS), a odometria e o SLAM. No desenvolvimento, será apresentado o estudo e a implementação do EKF integrado ao SLAM. Os resultados serão discutidos, considerando a eficácia do filtro e métricas de desempenho. A conclusão recapitulará os objetivos, ressaltando as contribuições, limitações e possíveis pesquisas futuras. Com isso, o presente estudo busca oferecer uma visão abrangente e consistente sobre o tema, combinando embasamento teórico e implementação prática para melhorar a precisão e confiabilidade do SLAM na robótica móvel.

2 ASPECTOR TEÓRICOS E CONCEITUAIS

Este capítulo tem como objetivo proporcionar um entendimento aprofundada dos aspectos teóricos e conceituais essenciais para o desenvolvimento e compreensão deste trabalho. Na presente seção, serão abordados de forma detalhada a estrutura física do robô, os sensores utilizados, o ROS, a odometria e o SLAM.

A estrutura física do robô será apresentada, descrevendo os componentes fundamentais, tais como sensores, atuadores e sistemas de comunicação, que desempenham um papel crucial no funcionamento e interação do robô com o ambiente.

O ROS será explorado como uma plataforma de referência amplamente utilizada na robótica, oferecendo um ambiente de desenvolvimento e controle para sistemas robóticos com uma vasta gama de ferramentas e bibliotecas que facilitam a criação de aplicações robustas e reutilizáveis.

A odometria será discutida em termos de estimativa do deslocamento do robô com base em informações de sensores, permitindo que o robô tenha uma noção de sua posição e movimento relativo.

Por fim, o SLAM será abordado como um dos principais desafios em robótica móvel, envolvendo a capacidade do robô de se localizar e construir mapas do ambiente de forma simultânea. Serão explorados as técnicas e algoritmos utilizados para estimar a posição e orientação do robô, além da criação de mapas coerentes do ambiente.

Este capítulo fornecerá os fundamentos teóricos necessários para compreender a implementação do Filtro de Kalman Estendido e sua integração com o SLAM, que serão discutidos no capítulo seguinte. Ao final deste capítulo, espera-se que o leitor esteja familiarizado com os conceitos essenciais para o entendimento e desenvolvimento do trabalho, preparado para adentrar no estudo e implementação das técnicas abordadas.

2.1 Estrutura física do robô móvel

Nesta seção, será abordada a estrutura física do robô utilizado neste trabalho. A estrutura física é um dos elementos essenciais para o funcionamento do robô e desempenha um papel crucial na realização das tarefas propostas.

O robô em questão, ilustrado na Figura 1, possui uma configuração diferencial, o que significa que ele é composto por duas rodas principais, responsáveis pela tração e direção,

Figura 1 – Robô móvel utilizado

Fonte: Lima Marcus Davi do Nascimento Forte (2016)

e outras duas rodas bobas, que garantem o equilíbrio e a estabilidade durante as operações. A distância entre as rodas diferenciais é de 0,4 metros e o raio de cada roda é de 0,08 metros. Essas medidas são fundamentais para a cinemática e o controle de movimento do robô.

No que diz respeito aos sensores, o robô é equipado com um Hokuyo URG LiDAR, um sensor de detecção remota por laser que fornece informações detalhadas do ambiente em

forma de nuvem de pontos. Além disso, é utilizado o MPU 6050, uma IMU, que combina um acelerômetro e um giroscópio para medir a aceleração linear e a taxa de rotação do robô. Dois encoders ópticos são empregados para medir a rotação das rodas e fornecer informações precisas sobre o deslocamento e a velocidade do robô.

Quanto aos atuadores, o robô utiliza dois motores Direct Current (DC), responsáveis pela tração das rodas, permitindo a locomoção em diferentes direções. Para comandar os motores DC, é utilizado o *DC Motor Controller* HDC2460, um *driver* capaz de receber comandos e fornecer energia para eles.

No que se refere aos sistemas de comunicação, destaca-se o uso da Raspberry Pi 3 B como um sistema de computação embarcado central. A Raspberry Pi é responsável pelo processamento de dados, controle do robô e comunicação com os demais componentes, como os sensores, atuadores e controladores.

No aspecto de alimentação, o robô é alimentado por duas baterias de 45Ah em série, proporcionando uma tensão de 24V. Essas baterias fornecem a energia necessária para alimentar os componentes eletrônicos e os motores do robô durante as operações.

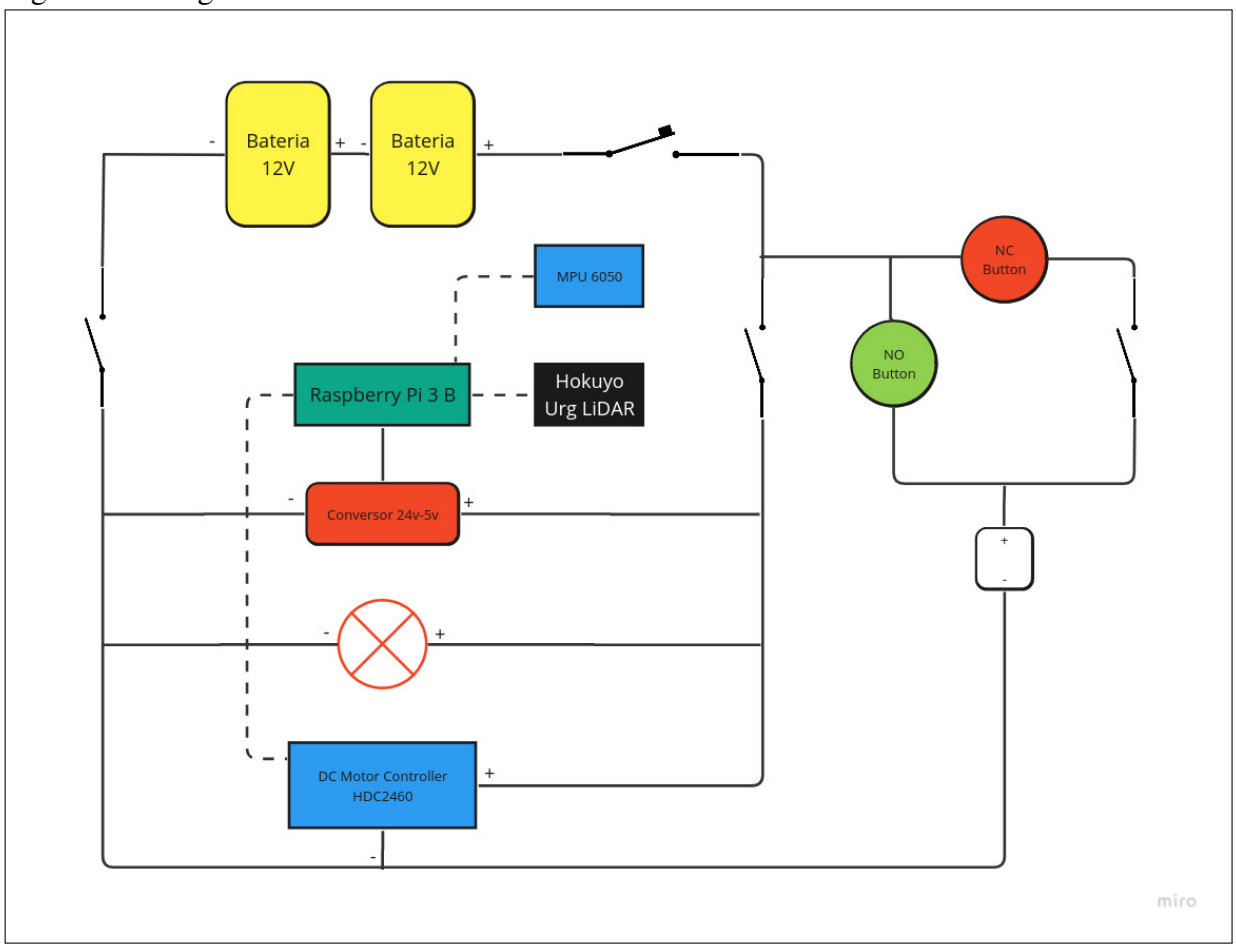
Além disso, são utilizados dispositivos de proteção e controle elétrico para garantir a segurança e o bom funcionamento do robô. Um disjuntor Siemens 5SX11 é empregado como proteção contra sobrecorrente, enquanto um contator WEG CWB25 é utilizado para ligar e desligar o robô de forma segura por meio da configuração contato selo. Também é utilizado um conversor buck de 24V para 5V, que alimenta a Raspberry Pi com a tensão adequada para seu funcionamento.

Essa estrutura física do robô, juntamente com os sensores, atuadores e sistemas de comunicação, proporciona a base necessária para a realização das tarefas de localização e mapeamento simultâneos com o uso do Filtro de Kalman Estendido, que serão abordados posteriormente no desenvolvimento deste trabalho.

2.1.1 Sensor LiDAR

O LiDAR é um sensor óptico que utiliza pulsos de laser para medir a distância entre o sensor e os objetos presentes no ambiente. Esses pulsos de luz são emitidos em direção aos objetos e, em seguida, o sensor mede o tempo que leva para o pulso retornar ao sensor após refletir nos objetos. Com base nessa informação de tempo, juntamente com o conhecimento da velocidade da luz, é possível calcular a distância entre o sensor e os objetos com grande precisão.

Figura 2 – Diagrama elétrico do robô utilizado



Fonte: o autor.

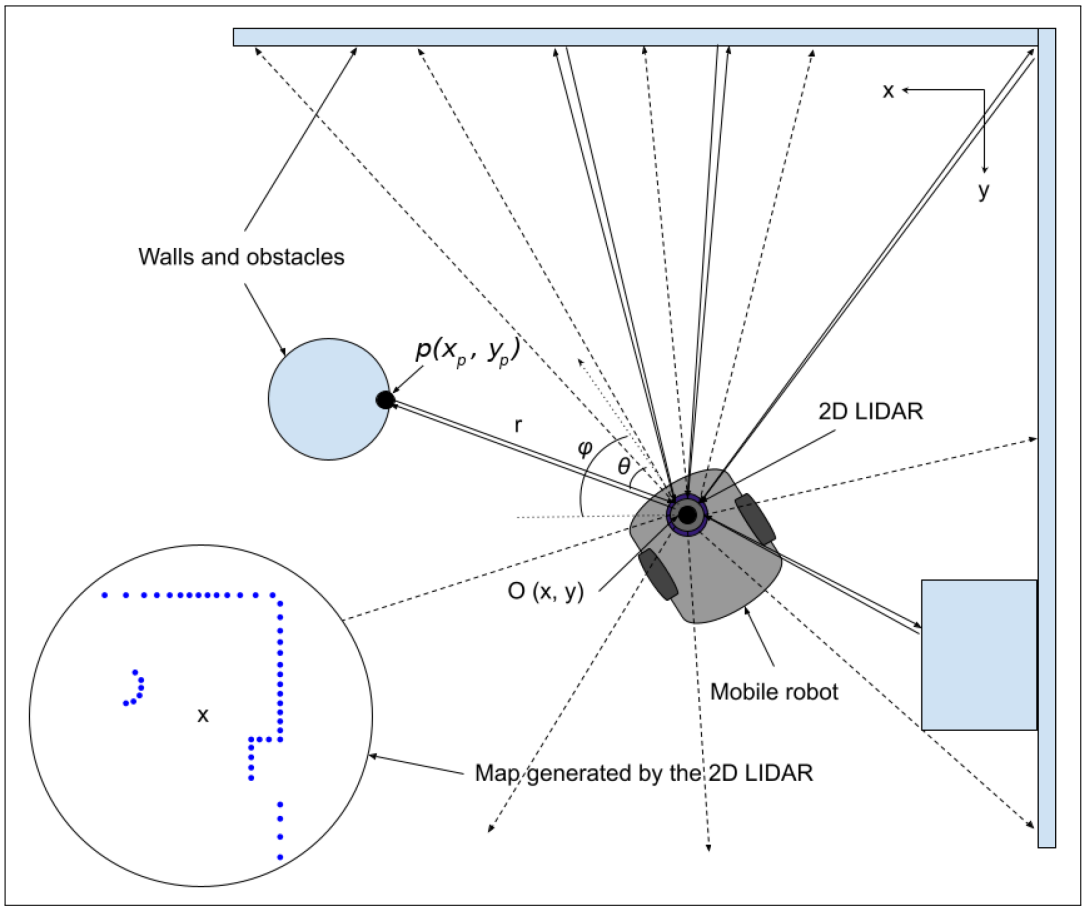
A Figura 3 mostra como o LiDAR pode ser integrado ao robô para detecção de obstáculos.

Uma das principais vantagens dos sensores LiDAR é a capacidade de fornecer medidas tridimensionais ou bidimensionais em tempo real. Ao girar o laser em torno de seu eixo vertical e horizontal, o sensor LiDAR pode mapear o ambiente em múltiplas direções, criando uma nuvem de pontos que representa a geometria dos objetos presentes. Essa nuvem de pontos pode ser utilizada para criar mapas de ambiente detalhados e altamente precisos.

Além da capacidade de mapeamento, os sensores LiDAR também são amplamente utilizados em aplicações de percepção e detecção de objetos. Através da análise dos dados da nuvem de pontos, é possível identificar e classificar objetos, como veículos, pedestres, árvores e obstáculos, permitindo que sistemas autônomos tomem decisões em tempo real com base nessas informações.

Existem diferentes tipos de sensores LiDAR disponíveis, variando em termos de alcance, resolução, taxa de amostragem e precisão. O sensor utilizado nesta pesquisa, que foi citado na seção 2.1, da marca Hokuyo, é capaz de fazer leituras bidimensionais, este foi ilustrado

Figura 3 – Funcionamento de um sensor LiDAR



Fonte: Bouazizi Alejandro Lorite Mora (2023)

Figura 4 – Sensor LiDAR utilizado

Fonte: (HOKUYO,)

na Figura 4. Sensores LiDAR de curto alcance são ideais para aplicações em ambientes internos, enquanto os de longo alcance são mais adequados para aplicações em ambientes externos, como mapeamento de grandes áreas ou navegação autônoma em estradas.

Neste trabalho, será explorado a utilização dos sensores LiDAR como uma ferramenta fundamental para a percepção ambiental em um robô móvel terrestre. Além disso, serão analisadas as técnicas de mapeamento e detecção de objetos aplicadas aos dados obtidos pelo

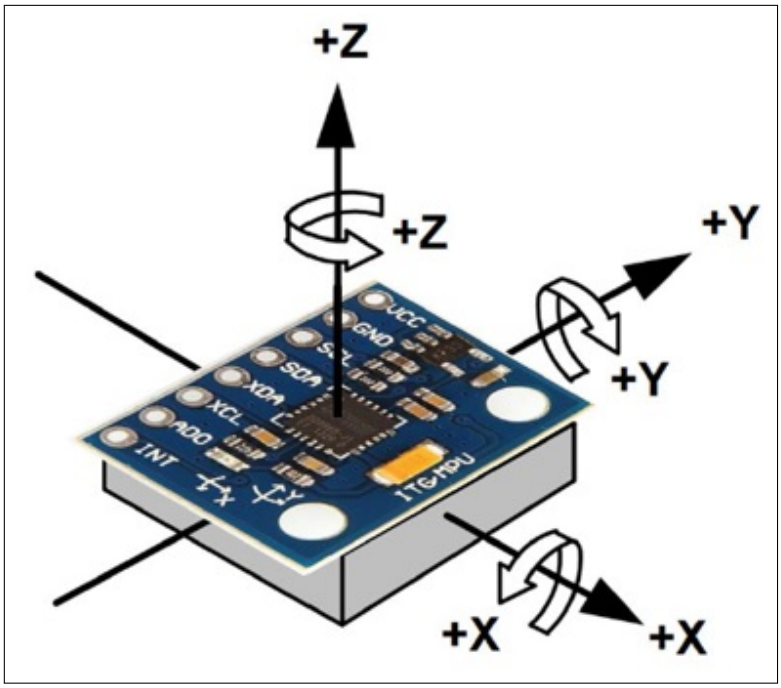
sensor.

2.1.2 Sensor IMU

A IMU é um dispositivo composto por uma combinação de sensores inerciais, como acelerômetros, giroscópios e magnetômetros. Esses sensores trabalham em conjunto para fornecer medidas precisas das forças e rotações experimentadas pelo objeto onde a IMU está instalada. Os acelerômetros medem a aceleração linear, os giroscópios a taxa de rotação e os magnetômetros o campo magnético do ambiente.

A IMU é amplamente utilizada em diversas aplicações, como robótica, sistemas de navegação inercial, realidade virtual, veículos autônomos, entre outras. Sua capacidade de fornecer medidas precisas e em tempo real do movimento e orientação de um objeto a torna uma ferramenta essencial para a percepção e controle de sistemas autônomos. A Figura 5 mostra o sensor utilizado, citado na seção 2.1, bem como indica os eixos de medida do sensor. Ademais, o sensor utilizado nesta pesquisa não possui magnetômetro.

Figura 5 – Sensor IMU utilizado



Fonte: (RESEARCHGATE, b)

No entanto, apesar das vantagens, as IMUs também apresentam desafios que precisam ser considerados. Os sensores inerciais estão sujeitos a erros devido a fatores como ruído e interferência magnética. A calibração adequada dos sensores e o uso de técnicas avançadas de

fusão de dados são importantes para minimizar esses erros e melhorar a precisão das medidas.

Neste trabalho, será exlorado a importância e as aplicações dos sensores IMU na robótica e em sistemas autônomos. Serão investigados os princípios de funcionamento dos acelerômetros, giroscópios e magnetômetros, bem como as técnicas de processamento de dados utilizadas para estimar a orientação, posição e velocidade do objeto. Além disso, será abordado os desafios e as soluções para a utilização eficiente das IMUs, como calibração, fusão de dados e compensação de erros.

2.2 ROS

A plataforma ROS desempenha um papel fundamental na robótica moderna, sendo amplamente adotada como uma estrutura de desenvolvimento e controle de sistemas robóticos. O ROS não é um sistema operacional convencional, mas sim um conjunto de ferramentas, bibliotecas e convenções projetadas para simplificar e acelerar o desenvolvimento de robôs.

Uma das características distintivas do ROS é a sua arquitetura modular e flexível, abordada em al. (2009). O sistema é projetado em torno de um conceito chamado de grafo de computação, no qual os componentes do sistema (chamados de nós) são conectados por meio de troca de mensagens. Essa abordagem baseada em nós permite que diferentes partes do sistema robótico sejam desenvolvidas e testadas independentemente, facilitando a reutilização de código e a colaboração entre diferentes equipes de pesquisa e desenvolvimento.

O ROS adota um modelo de comunicação assíncrona, no qual os nós podem se comunicar uns com os outros por meio de tópicos. Os tópicos são canais de comunicação que permitem a troca de mensagens em tempo real, enquanto os serviços permitem a chamada de funções remotas em um nó específico. A troca de mensagem é feita por meio dos *Publishers* e *Subscribers*, responsáveis por enviar e receber as mensagens nos tópicos, respectivamente. Além disso, há os *frames*, que funcionam como sistemas de referência para representar a posição e orientação de objetos em um ambiente tridimensional. Essa arquitetura distribuída e baseada em mensagens facilita a criação de sistemas robóticos complexos, nos quais diferentes componentes podem ser desenvolvidos independentemente e interligados de maneira flexível.

Além da arquitetura modular, o ROS oferece uma ampla gama de bibliotecas e pacotes que cobrem uma variedade de funcionalidades. Essas bibliotecas fornecem suporte para controle de movimento, aquisição de dados de sensores, processamento de imagens, reconhecimento de objetos, planejamento de trajetória e muito mais.

No contexto deste trabalho, o ROS será utilizado como a plataforma principal para desenvolver e controlar o robô. Através da utilização do ROS, espera-se obter uma maior flexibilidade e facilidade de integração das diferentes partes do sistema robótico. Além disso, a versão do ROS utilizada neste trabalho é a versão *Noetic*, instalada no sistema operacional Ubuntu Linux LTS 20.04.

Assim, o ROS desempenha um papel fundamental na condução deste trabalho, fornecendo uma estrutura flexível e modular para o desenvolvimento de aplicações robóticas. Sua arquitetura distribuída, baseada em mensagens e sua extensa coleção de bibliotecas tornam o ROS uma escolha popular na comunidade de robótica, permitindo a criação de sistemas inteligentes e eficientes.

2.3 Odometria

Na área de robótica, a odometria desempenha um papel fundamental na estimativa da posição e do deslocamento de um robô em um ambiente (LEE, 1998). Odometria se refere à técnica de estimar a trajetória do robô com base nas informações fornecidas por seus sensores, como *encoders*.

A odometria baseada em *encoders* é uma das abordagens mais comuns para estimar o deslocamento do robô. Os *encoders* são dispositivos que medem a rotação das rodas e fornecem informações sobre a distância percorrida pelos motores. Com base nesses dados, é possível calcular a odometria do robô, estimando sua posição e orientação ao longo do tempo.

A compreensão da odometria é essencial para o desenvolvimento de sistemas de localização e mapeamento simultâneos, pois a odometria fornece uma estimativa inicial da posição do robô, que pode ser combinada com outras informações sensoriais para obter uma localização mais precisa. A odometria também desempenha um papel crucial na navegação autônoma, permitindo que o robô planeje e execute trajetórias eficientes em seu ambiente.

Ao explorar os princípios e métodos de cálculo da odometria, este trabalho busca contribuir para a compreensão e o avanço no campo da odometria robótica, visando à obtenção de estimativas mais precisas e confiáveis da posição e do deslocamento do robô em diferentes ambientes e situações.

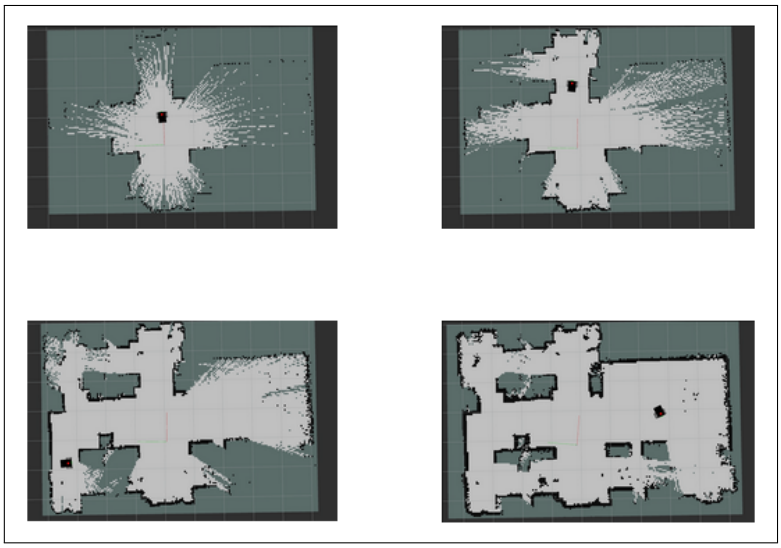
2.4 SLAM

O SLAM é uma área de pesquisa que desempenha um papel crucial em diversas aplicações robóticas, permitindo que um robô explore um ambiente desconhecido, ao mesmo tempo em que constrói um mapa desse ambiente e estima sua própria posição dentro dele. Essa capacidade é fundamental para a navegação autônoma, inspeção de ambientes perigosos, exploração subaquática e mapeamento de espaços internos.

No contexto da SLAM, o ROS desempenha um papel significativo como uma plataforma de desenvolvimento e execução de sistemas robóticos. O ROS fornece uma estrutura flexível e modular que permite a integração de diversos componentes e facilita a implementação de algoritmos avançados de SLAM.

A Figura 6 exemplifica o funcionamento do algoritmo do SLAM.

Figura 6 – Funcionamento do SLAM



Fonte: adaptado de (SADOWSKI,)

Dentre os pacotes disponíveis no ROS para SLAM, destaca-se o pacote Hector_Mapping. O Hector_Mapping é um pacote de mapeamento baseado em grade (grid-based mapping), que utiliza informações de sensores, como LiDAR, para construir um mapa do ambiente em tempo real. Ele oferece suporte para a criação de mapas 2D e 3D, permitindo uma representação precisa do ambiente explorado pelo robô.

Uma das principais vantagens do pacote Hector_Mapping é a sua capacidade de lidar com ambientes dinâmicos e em movimento. Ele possui técnicas robustas de detecção e exclusão de objetos móveis, o que possibilita a criação de mapas mais confiáveis, mesmo em ambientes onde há alterações frequentes.

Além disso, o Hector_Mapping é altamente integrado ao ROS, o que significa que os dados de entrada e saída são facilmente compartilhados com outros componentes do sistema, como a localização do robô e o planejamento de trajetória. Essa integração simplifica o desenvolvimento de sistemas robóticos complexos, tal que o SLAM é apenas uma das funcionalidades necessárias.

Ao utilizar o pacote Hector_Mapping no contexto deste trabalho, será possível explorar as capacidades avançadas de mapeamento e localização fornecidas por esse pacote, contribuindo para a obtenção de resultados precisos e confiáveis no desenvolvimento do filtro de Kalman estendido aplicado à localização e mapeamento simultâneos.

3 METODOLOGIA

Este capítulo apresenta as estratégias e abordagens adotadas neste estudo para alcançar os objetivos propostos de desenvolver um sistema de SLAM em um robô móvel terrestre utilizando sensor LiDAR e odometria com Filtro de Kalman Estendido. Neste capítulo, serão descritos detalhadamente o ambiente de teste e os pacotes de software utilizados, bem como a coleta de dados, a configuração dos parâmetros e as métricas de avaliação empregadas.

Primeiramente, será apresentada uma descrição do ambiente de teste, incluindo suas características físicas, como tamanho, presença de obstáculos e iluminação.

Em seguida, serão detalhados os pacotes de software utilizados para implementar o SLAM e o Filtro de Kalman Estendido. Será explicado como o pacote Hector_SLAM foi configurado e integrado ao robô móvel, bem como a forma que os dados do sensor LiDAR foram adquiridos e processados. Além disso, será abordada a configuração do pacote robot_localization para realizar a fusão dos dados do MPU6050 e da odometria, utilizando o Filtro de Kalman Estendido para estimação da localização.

A coleta de dados será descrita em detalhes, incluindo o protocolo experimental adotado e os movimentos realizados pelo robô móvel durante os testes. Será explicado como os dados do LiDAR e da odometria foram coletados, sincronizados e armazenados para posterior processamento e análise.

Por fim, serão apresentadas as métricas utilizadas para avaliar o desempenho do sistema de SLAM e do Filtro de Kalman Estendido. Será discutida a escolha das métricas e a forma como elas foram calculadas para analisar a qualidade da estimativa da localização e a acurácia do mapa gerado.

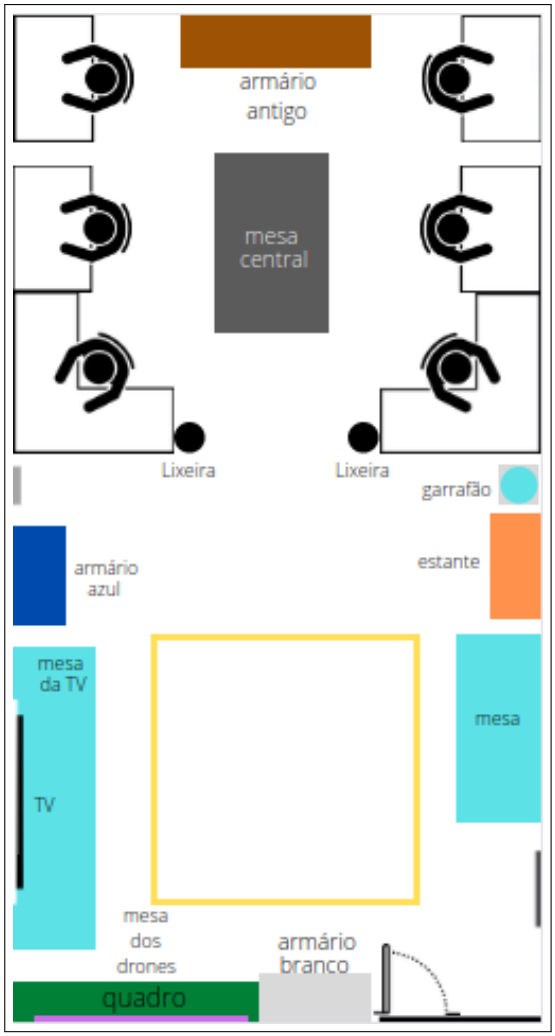
A metodologia adotada neste estudo visa fornecer uma base sólida e consistente para a implementação e avaliação do sistema proposto, permitindo uma análise detalhada do desempenho e resultados obtidos. Através desta descrição minuciosa, busca-se assegurar a reprodutibilidade dos experimentos e a confiabilidade das conclusões apresentadas neste trabalho.

3.1 Descrição do ambiente de testes

O ambiente de teste utilizado neste estudo consiste em um laboratório de pesquisa, representado na Figura 7, com dimensões aproximadas de 5 metros por 12,5 metros. O espaço do laboratório é caracterizado por uma área com várias mesas, cadeiras e armários, com piso

plano e sem grandes variações de iluminação.

Figura 7 – Representação do ambiente de teste



Fonte: o autor.

Ressalta-se que o laboratório possui iluminação artificial estável, com luzes de teto difusas e uma intensidade luminosa constante. A ausência de variações significativas de iluminação contribui para a estabilidade dos dados obtidos pelo sensor LiDAR.

Durante os testes, o acesso ao laboratório foi restrito a fim de evitar movimentações que pudessem ser captadas pelo LiDAR e mudanças no ambiente, gerando interferências durante as execuções.

Todas as medidas foram tomadas para garantir um ambiente controlado e propício à realização dos experimentos, assegurando que as condições do ambiente não influenciassem negativamente nos resultados obtidos e na avaliação do desempenho do SLAM.

3.2 Estimação de posição

A odometria é uma técnica essencial na robótica para estimar a posição e o deslocamento de um robô com base nas leituras dos encoders das rodas. Esta subseção apresenta de forma matemática como calcular a odometria, considerando a distância entre as rodas, o raio das rodas e as velocidades angulares das rodas direita e esquerda. No entanto, é importante considerar fatores como deslizamento das rodas e erros de medição dos sensores, que podem afetar a precisão da odometria.

Seja:

- d a distância entre as rodas do robô,
- r o raio das rodas,
- v_d a velocidade angular da roda direita, e
- v_e a velocidade angular da roda esquerda.

Para simplificar, considere que o robô se move em um plano bidimensional, em que sua posição é representada por um par ordenado (x,y) e sua orientação é representada pelo ângulo θ em relação a um sistema de coordenadas global.

A velocidade linear do robô v pode ser calculada pela média das velocidades angulares das rodas direita e esquerda:

$$v = \frac{v_d + v_e}{2} \tag{3.1}$$

A velocidade angular do robô ω pode ser calculada pela diferença das velocidades angulares das rodas direita e esquerda dividida pela distância entre as rodas:

$$\omega = \frac{v_d - v_e}{d} \tag{3.2}$$

Agora, pode-se calcular as variações na posição $(\Delta x, \Delta y)$ e na orientação $\Delta\theta$ do robô em um pequeno intervalo de tempo Δt usando as seguintes equações de movimento, seguindo a equação de movimento retilíneo uniforme:

$$\Delta x = v \cdot \cos(\theta) \cdot \Delta t \qquad \Delta y = v \cdot \sin(\theta) \cdot \Delta t \qquad \Delta\theta = \omega \cdot \Delta t \tag{3.3}$$

Finalmente, pode-se atualizar a posição e a orientação do robô somando as variações calculadas:

$$x_{\text{novo}} = x_{\text{antigo}} + \Delta x \qquad y_{\text{novo}} = y_{\text{antigo}} + \Delta y \qquad \theta_{\text{novo}} = \theta_{\text{antigo}} + \Delta\theta \tag{3.4}$$

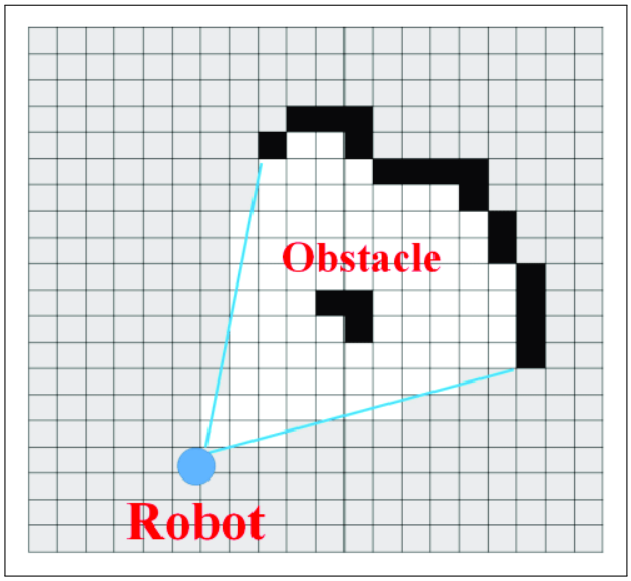
Essas equações permitem estimar a posição e a orientação do robô com base nas leituras do encoder.

3.3 Configuração do SLAM

A metodologia utilizada neste estudo emprega o pacote Hector_SLAM, que é baseado no método de mapeamento de grade de ocupação, ilustrado na Figura 8, para realizar o mapeamento de ambientes desconhecidos. Esse pacote é amplamente utilizado na área de robótica e tem demonstrado eficácia no mapeamento em tempo real.

O algoritmo do Hector_SLAM, conforme descrito em Anandito (2019), utiliza a probabilidade de ocupação das células como base para seu funcionamento. O objetivo é criar um mapa preciso e detalhado do ambiente em que o robô está operando.

Figura 8 – Representação do método de mapeamento de grade de ocupação



Fonte: (RESEARCHGATE, a)

O funcionamento do algoritmo se baseia na divisão do ambiente em uma grade de células. Cada célula na grade representa uma pequena região do ambiente, e inicialmente todas as células são consideradas desconhecidas. A probabilidade de ocupação de cada célula é calculada e atualizada à medida que o robô se move e coleta dados dos sensores.

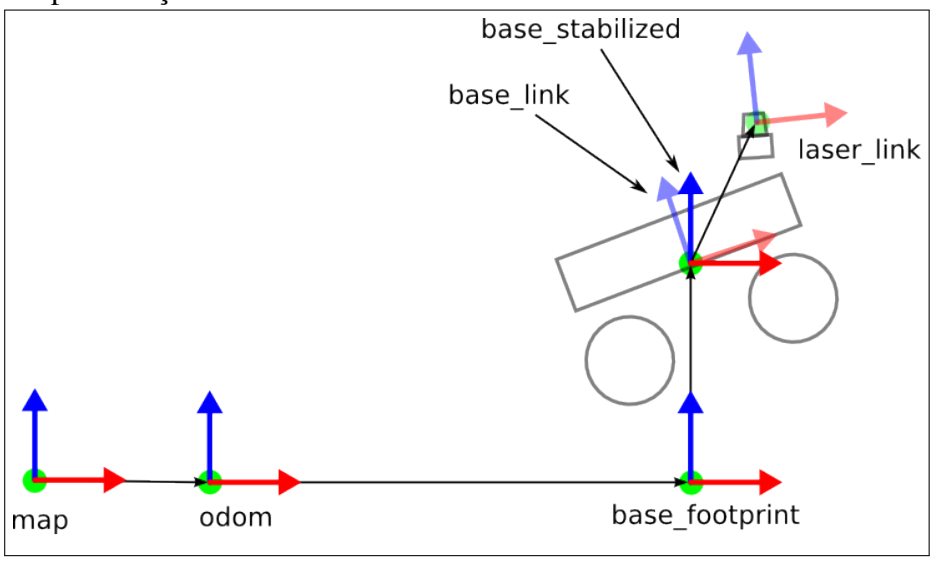
Quando o robô observa uma célula, os dados do sensor são utilizados para determinar se a célula está ocupada ou livre. Se o sensor detectar a presença de um objeto ou obstáculo na célula, a probabilidade de ocupação dessa célula será aumentada. Por outro lado, se o sensor não detectar nada na célula, a probabilidade de ocupação será diminuída.

Ao longo do tempo, à medida que o robô explora o ambiente e coleta mais dados sensoriais, as probabilidades de ocupação das células são atualizadas e refinadas. O algoritmo utiliza técnicas de filtragem estendida, como o filtro de Kalman estendido, para combinar as informações dos sensores e calcular as probabilidades de ocupação de forma precisa e eficiente.

Com base nas probabilidades de ocupação das células, o 'Hector_SLAM' gera um mapa 2D ou 3D do ambiente, onde as células ocupadas são representadas por obstáculos e as células livres são representadas por áreas transitáveis. Esse mapa fornece uma representação visual detalhada do ambiente mapeado e é atualizado em tempo real à medida que o robô se movimenta.

O pacote 'Hector_SLAM' utiliza o nó 'Hector_Mapping' para criar um mapa do ambiente enquanto estima a posição 2D da plataforma em tempo real, com base nas leituras do scanner a laser. A Figura 9 ilustra os *frames* de referência relevantes em uma visão 2D simplificada de um robô se movendo em terreno acidentado, o que resulta em movimentos de inclinação e rotação da plataforma:

Figura 9 – Representação do ambiente de teste



Fonte: (KOHLBRECHER,)

A relação geral entre os *frames* de referência 'map', 'odom' e 'base_link' já foi descrita em Meeussen (2010).

Para utilizar o pacote Hector_SLAM, os comandos a seguir precisam ser executados no terminal do Ubuntu para obter os mapas gerados pela leitura do LiDAR e com a leitura do LiDAR com odometria. Os arquivos *launch* estão presentes no Apêndice A e Apêndice B, respectivamente.

```
roslaunch hector_slam_launch hectorslam_lidar.launch
roslaunch hector_slam_launch hectorslam_odom.launch
```

3.4 O Filtro de Kalman

Ao realizar a estimativa, que é um processo ou técnica utilizada para calcular ou aproximar um valor desconhecido ou incerto com base em informações disponíveis (ALBUQUERQUE, 2018), do estado de um sistema, especialmente em ambientes sujeitos a incertezas e ruídos, é crucial contar com métodos eficientes que possam combinar informações observadas com estimativas prévias, visando obter uma estimativa mais precisa e confiável. Nesse contexto, o Filtro de Kalman emerge como uma ferramenta fundamental na área de estimativa de estado.

O filtro apresentado em Kalman (1960) tem se mostrado um método poderoso e amplamente utilizado em diversas aplicações, desde a navegação espacial até a robótica móvel. Sua utilidade reside na capacidade de realizar uma estimativa ótima do estado de um sistema dinâmico, mesmo quando as medidas observadas estão sujeitas a erros e as equações que governam o sistema são não lineares.

O princípio fundamental do Filtro de Kalman está na combinação adequada das informações disponíveis, combinando as medidas observadas em tempo real com estimativas prévias do estado do sistema, ponderando-as de acordo com suas respectivas incertezas. Isso permite obter uma estimativa mais precisa, adaptando-se continuamente às informações mais recentes.

Através da combinação das medidas observadas e estimativas prévias, o método estima o estado do sistema em cada instante de tempo, além de fornecer informações sobre a incerteza associada a essa estimativa. Dessa forma, é possível acompanhar e prever a evolução do sistema com maior confiança.

Para alcançar essa estimativa otimizada, o Filtro de Kalman se baseia em equações matemáticas que descrevem a dinâmica do sistema, a relação entre as medidas e o estado, bem como as incertezas envolvidas. As principais equações do filtro são a equação de previsão (predição) e a equação de correção (atualização). A etapa de previsão propaga o estado estimado e sua covariância para o próximo instante de tempo, enquanto a etapa de correção combina a medida observada com a estimativa prévia, atualizando o estado estimado e a covariância.

A covariância é uma medida estatística que descreve o grau de associação linear entre duas variáveis aleatórias. Ela indica como as duas variáveis variam conjuntamente em

relação às suas médias (ALBUQUERQUE, 2018) e é calculada a partir do produto das diferenças entre os valores observados das variáveis e suas médias correspondentes.

Formalmente, a covariância entre duas variáveis aleatórias X e Y é definida como:

$$\text{cov}(X,Y) = E[(X - E[X])(Y - E[Y])] \tag{3.5}$$

onde $E[.]$ representa o operador de esperança, ou valor esperado.

No decorrer deste trabalho, será explorado, de forma resumida, o Filtro de Kalman como uma solução para a correção da odometria em um sistema robótico. Será utilizada suas propriedades fundamentais, juntamente com uma extensão conhecida como Filtro de Kalman Estendido (EKF), que permite lidar com sistemas não lineares. Além disso, será feito uso do pacote 'robot_localization' do ROS para implementar o EKF e obter uma estimativa mais precisa da posição do robô.

Ao compreender e aplicar corretamente tal filtro, será possível melhorar significativamente a qualidade das estimativas de estado em sistemas dinâmicos, contribuindo para avanços em diversas áreas, como a robótica, automação e processamento de sinais.

3.4.1 Etapas do Filtro de Kalman

O Filtro de Kalman é composto por duas etapas essenciais: a etapa de previsão e a etapa de correção. Cada uma dessas etapas desempenha um papel fundamental na estimativa do estado do sistema com base nas medidas observadas e nas estimativas prévias.

A etapa de previsão é responsável por propagar o estado estimado e a covariância para o próximo instante de tempo, levando em consideração as informações de controle aplicadas ao sistema. Essa etapa utiliza a matriz de transição de estado (**F**), de dimensão MxM, e a matriz de controle (**B**), de dimensão MxN, que relacionam o estado atual ao próximo estado com base nas entradas externas aplicadas. Durante a previsão, o estado estimado $\hat{\mathbf{x}}_k$, de dimensão Mx1, e a covariância do estado $\mathbf{P}_k$, de dimensão MxM, são atualizados de acordo com as seguintes equações:

$$\hat{\mathbf{x}}_k = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \qquad \mathbf{P}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{3.6}$$

Nessas equações, $\hat{\mathbf{u}}_k$, de dimensão Nx1, é o vetor de controle aplicado no instante de tempo k e (**Q**), de dimensão MxM, que representa a incerteza associada à dinâmica do sistema.

Após a etapa de previsão, passa-se para a etapa de correção, na qual combina-se a medida observada com a estimativa prévia para atualizar o estado estimado e a covariância. Nessa etapa, utiliza-se a matriz de transição da medida (**H**), de dimensão NxM, que mapeia o estado para o espaço das medidas, e a matriz de covariância da medida (**R**), de dimensão NxN, que representa a incerteza associada às medidas. A equação de correção é dada por:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k) \tag{3.7}$$

Nessa equação, $\hat{\mathbf{z}}_k$, de dimensão Nx1, é o vetor de medida observada no instante de tempo k. A matriz de ganho de Kalman $\hat{\mathbf{K}}_k$, de dimensão MxN, é calculada de forma a otimizar a combinação entre a estimativa prévia e a medida observada, e é dada por:

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1} \tag{3.8}$$

Além disso, a matriz de covariância do estado ($\hat{\mathbf{P}}_k$) também é atualizada no estado de correção de acordo com a equação:

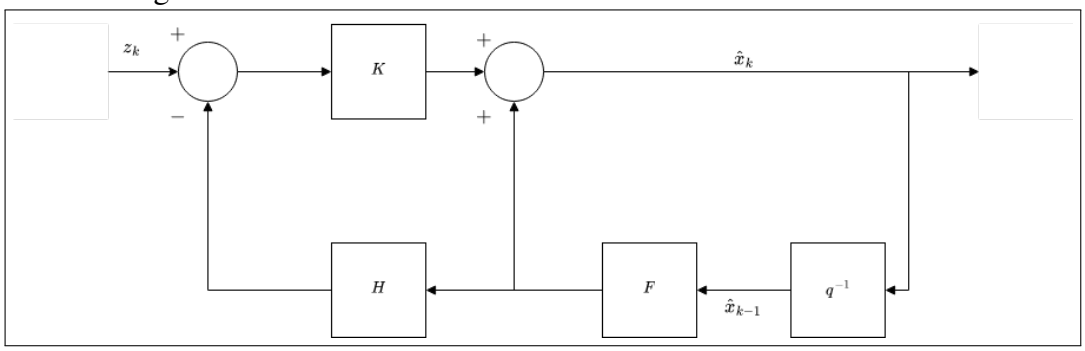
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k \tag{3.9}$$

Na Equação 3.9, *I* representa a matriz identidade com dimensão MxM.

Assim, a etapa de previsão do Filtro de Kalman propaga as estimativas do estado e da covariância com base nas informações de controle, enquanto a etapa de correção combina as medidas observadas com a estimativa prévia para atualizar o estado estimado e a covariância. Esse processo iterativo de previsão e correção é fundamental para obter uma estimativa precisa do estado do sistema, levando em consideração tanto a dinâmica do sistema quanto as informações disponíveis por meio das medidas observadas. A Figura 10 ilustra o funcionamento do Filtro de Kalman por diagrama de blocos.

No próximo tópico, será explorado uma extensão do Filtro de Kalman, que é especialmente útil para sistemas não lineares, como a odometria de um robô móvel, e introduz modificações nas equações do Filtro de Kalman para lidar com essa não linearidade. Será abordado como o EKF pode ser aplicado na correção da odometria e como o pacote 'robot_localization' do ROS pode ser utilizado para implementar o EKF de forma eficiente e prática.

Figura 10 – Diagrama de blocos do Filtro de Kalman



Fonte: o autor.

3.4.2 O Filtro de Kalman Estendido

Nesta subseção, será analisado o Filtro de Kalman Estendido como uma extensão do Filtro de Kalman convencional. Será abordado sua aplicação em sistemas não lineares, com ênfase na odometria de um robô móvel. Além disso, será discutido a necessidade de modificar o filtro de Kalman básico para lidar com a não linearidade.

O Filtro de Kalman é amplamente utilizado para estimar o estado de sistemas dinâmicos, assumindo que as equações do sistema são lineares e que o ruído segue uma distribuição gaussiana. No entanto, muitos sistemas do mundo real exibem comportamento não linear, o que torna o uso do filtro de Kalman convencional impraticável.

Uma solução para lidar com sistemas não lineares é o EKF (BUCY, 1961), que é uma extensão do filtro de Kalman que permite estimar o estado de sistemas não lineares aproximando-os por meio de uma série de pontos de operação. Essa abordagem é efetiva para sistemas cuja dinâmica não linear pode ser aproximada localmente por um modelo linear.

A odometria de um robô móvel é um exemplo de aplicação comum do EKF, utilizado para estimar a posição e orientação do robô com base nas informações da odometria, que geralmente é afetada por erros sistemáticos e ruído. A não linearidade surge devido às relações complexas entre as variáveis de estado do robô, como posição, velocidade e orientação.

A modificação do filtro de Kalman básico é necessária para lidar com a não linearidade presente na odometria do robô móvel. Por meio da linearização local das equações do sistema em cada instante de tempo, o EKF permite a atualização iterativa do estado estimado e da covariância, fornecendo uma estimativa mais precisa em sistemas não lineares.

Ao adotar a extensão do método, é possível aproveitar as suas vantagens para lidar com sistemas não lineares, como a odometria de um robô móvel. A partir de agora, será explorado as modificações específicas no filtro de Kalman básico que permitem sua aplicação

em sistemas não lineares e discutido em detalhes as equações do EKF.

A principal modificação realizada no método extendido está relacionada à linearização das equações do sistema não linear. Essa linearização ocorre em cada instante de tempo, aproximando localmente as equações do sistema não linear por meio de uma série de pontos de operação. Isso permite que o EKF utilize as equações lineares do filtro de Kalman convencional, que são mais simples de manipular e calcular.

Um componente fundamental para realizar a linearização no EKF é a matriz jacobiana, desenvolvida na Equação 3.15, que se resume a uma matriz de derivadas parciais que descreve as taxas de variação das equações não lineares em relação às variáveis de estado. Através do cálculo da matriz jacobiana, é possível obter uma aproximação linear das equações do sistema, viabilizando a aplicação do filtro de Kalman convencional.

A matriz jacobiana é essencial para o EKF, pois fornece informações sobre as relações entre as variáveis de estado e como elas evoluem no tempo. Ela é utilizada tanto na etapa de previsão quanto na etapa de correção do filtro extendido, permitindo a atualização iterativa do estado estimado e da covariância. Portanto, a matriz jacobiana desempenha um papel crucial na adaptação do filtro de Kalman às não linearidades do sistema.

Ao lidar com sistemas não lineares, o EKF e suas modificações no filtro de Kalman convencional permitem obter estimativas mais precisas do estado do sistema. Através da linearização das equações não lineares utilizando a matriz jacobiana, o filtro estendido pode fornecer resultados confiáveis mesmo em situações em que o filtro de Kalman convencional não seria aplicável.

Na próxima subseção, será explicado em detalhes as equações do EKF.

3.4.3 Etapas do Filtro de Kalman Estendido

Nesta seção, será abordado as etapas essenciais do EKF. As suas equações serão apresentadas, incluindo a de previsão, de correção e de covariância prevista, além da covariância posterior, a matriz jacobiana e o ganho de Kalman.

O EKF é composto por duas etapas principais: previsão e correção. Na etapa de previsão, o método propaga a estimativa do estado e a covariância do estado para o próximo instante de tempo. Essa propagação é realizada utilizando uma versão linearizada do modelo não linear do sistema, por meio da matriz jacobiana.

A equação de previsão nesse filtro pode ser representada da seguinte forma:

$$\hat{\mathbf{x}}_k^- = f(\hat{x}_{k-1}, u_{k-1}) \tag{3.10}$$

onde $\hat{\mathbf{x}}_k^-$ é a estimativa do estado, com dimensão Nx1, no instante de tempo k antes da correção, f é a função que descreve a dinâmica do sistema não linear e $\mathbf{u}_{k-1}$ é o vetor de controle, com dimensão Px1, aplicado ao sistema no instante de tempo $k-1$.

Além disso, a covariância do estado prevista, denotada como $\mathbf{P}_k^-$, com dimensão NxN, também é atualizada durante a etapa de previsão. A equação para a covariância prevista é dada por:

$$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \tag{3.11}$$

onde $\mathbf{A}_{k-1}$, com dimensão NxN, é a matriz jacobiana das derivadas parciais da função de transição de estado em relação ao estado, $\mathbf{P}_{k-1}$ é a covariância do estado estimada no instante de tempo anterior e $\mathbf{Q}_{k-1}$, com dimensão NxN, é a matriz de covariância do ruído do processo.

Na etapa de correção, o EKF combina a estimativa prévia do estado com a medida observada, ajustando a estimativa e a covariância do estado com base na diferença entre a medida real e a estimativa prevista. A equação de correção pode ser expressa da seguinte forma:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)) \tag{3.12}$$

onde $\hat{\mathbf{x}}_k$ é a estimativa corrigida do estado, com dimensão Nx1, no instante de tempo k, $\mathbf{K}_k$ é o ganho de Kalman, com dimensão NxM, no instante de tempo k, $\mathbf{z}_k$ é a medida observada, com dimensão Mx1, no instante de tempo k, e h é a função que relaciona o estado estimado às medidas esperadas.

A covariância do estado corrigida, denotada como $\mathbf{P}_k$, também é atualizada durante a etapa de correção. A equação para a covariância posterior é dada por:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{3.13}$$

onde $\mathbf{I}$ é a matriz identidade de dimensão NxN, $\mathbf{K}_k$ é o ganho de Kalman e $\mathbf{H}_k$ é a matriz jacobiana das derivadas parciais da função de observação em relação ao estado estimado, com dimensão MxN.

O vetor de estados, x, no contexto deste estudo, é uma matriz 15x1, definido por:

$$\mathbf{x}_k = \begin{bmatrix} X_k & Y_k & Z_k & \text{roll}k & \text{pitch}k & \text{yaw}k & \dot{X}k & \dot{Y}k & \dot{Z}k & \dot{\text{roll}}_k \\ & & & \dot{\text{pitch}}k & \dot{\text{yaw}}k & \ddot{X}k & \ddot{Y}k & \ddot{Z}_k \end{bmatrix}^T. \tag{3.14}$$

Com isso, a matriz jacobiana $\mathbf{H}_k$ é calculada da seguinte forma:

$$\mathbf{H}_k = \begin{bmatrix} \frac{\partial h(X_k^-)}{\partial X_k^-} & \frac{\partial h(Y_k^-)}{\partial X_k^-} & \cdots & \frac{\partial h(\ddot{Z}_k^-)}{\partial X_k^-} \\ \frac{\partial h(X_k^-)}{\partial Y_k^-} & \frac{\partial h(Y_k^-)}{\partial Y_k^-} & \cdots & \frac{\partial h(\ddot{Z}_k^-)}{\partial Y_k^-} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h(X_k^-)}{\partial \ddot{Y}_k^-} & \frac{\partial h(Y_k^-)}{\partial \ddot{Y}_k^-} & \cdots & \frac{\partial h(\ddot{Z}_k^-)}{\partial \ddot{Y}_k^-} \\ \frac{\partial h(X_k^-)}{\partial \ddot{Z}_k^-} & \frac{\partial h(Y_k^-)}{\partial \ddot{Z}_k^-} & \vdots & \frac{\partial h(\ddot{Z}_k^-)}{\partial \ddot{Z}_k^-} \end{bmatrix}. \tag{3.15}$$

O ganho de Kalman $\mathbf{K}_k$ desempenha um papel crucial na etapa de correção do EKF. Ele determina a contribuição relativa da medida observada e da estimativa prévia do estado na atualização do estado corrigido. O ganho de Kalman é calculado da seguinte forma:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{3.16}$$

em que $\mathbf{R}_k$, com dimensão MxM, é a matriz de covariância do ruído de medição.

Por meio da linearização das equações do sistema não linear utilizando a matriz jacobiana e do cálculo do ganho de Kalman, o EKF é capaz de lidar com sistemas não lineares e fornecer estimativas do estado mais precisas. Essas modificações permitem a sua aplicação em uma ampla gama de aplicações onde a não linearidade está presente.

3.4.4 Configuração do EKF

Para a utilização do EKF integrado ao ROS, foi utilizado o pacote 'robot_localization', que é um pacote que fornece uma estrutura para estimativa de estado e fusão de sensores em robôs móveis. Ele é projetado para lidar com o problema de estimativa de estado em tempo real, combinando dados de sensores de forma robusta e eficiente.

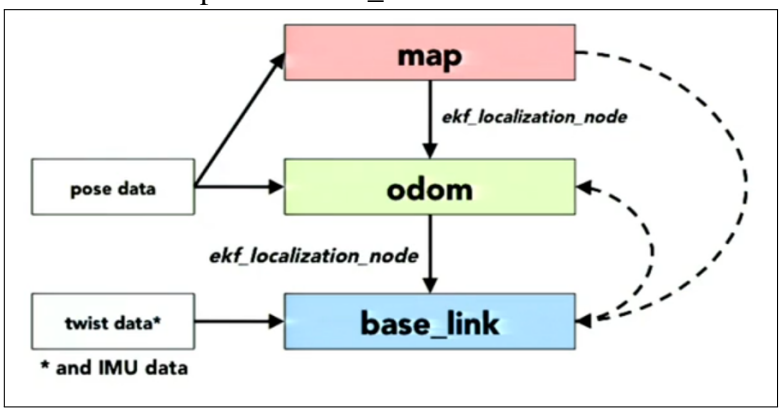
Formalmente, o pacote "robot_localization"oferece uma implementação modular e configurável de filtro de Kalman estendido e filtro de Kalman unscented. No caso deste trabalho, foi utilizado sua configuração com EKF.

O pacote suporta a fusão de dados de diversos sensores, como odometria, Global Positioning System (GPS), IMU, encoders, entre outros, permitindo uma integração flexível e adaptável a diferentes configurações de robôs.

Além disso, o pacote "robot_localization"inclui ferramentas para configurar e sintonizar os filtros de Kalman, bem como para visualizar e analisar os resultados da estimativa de estado. Ele também oferece recursos para lidar com casos de uso mais avançados, como a correção de deriva acumulada ao longo do tempo e a fusão de múltiplas instâncias do mesmo tipo de sensor.

O pacote 'robot_localization' funciona por meio da subscrição nos tópicos publicados pelos sensores e, a partir disso, realiza a fusão por meio do algorítmo utilizando EKF. A Figura 11 ilustra a funcionalidade deste pacote.

Figura 11 – Funcionalidade do pacote 'robot_localization'



Fonte: (MOORE; STOUCH, 2014)

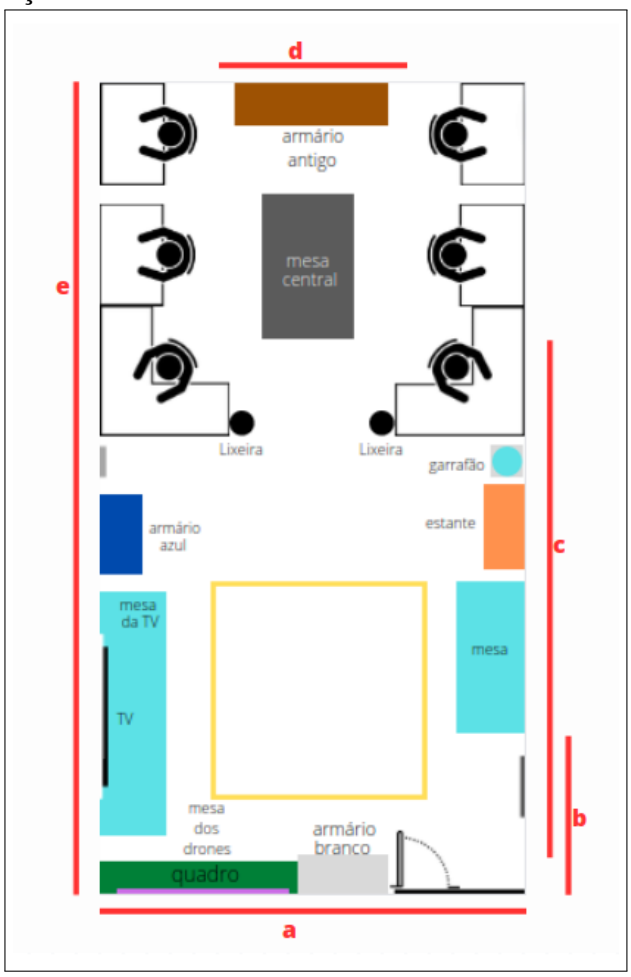
Para utilizar este pacote, o comando a seguir precisa ser executados no terminal do Ubuntu. O arquivo *launch* está presente no Apêndice C.

```
roslaunch robot_localization odom_ekf.launch
```

4 RESULTADOS

Neste capítulo, serão apresentados os resultados obtidos a partir da aplicação do SLAM em um ambiente de teste. Para isto, foram gerados três mapas, que serão explicados posteriormente, e comparados com o a representação do ambiente de teste representado na seção 3.1. A Figura 12 indica medidas que foram tiradas, manualmente, do espaço estudado. Além disso, a trajetória feita pelo robô foi a mesma para todos os testes para gerar uma avaliação equalitária.

Figura 12 – Representação do ambiente com indicadores de medidas



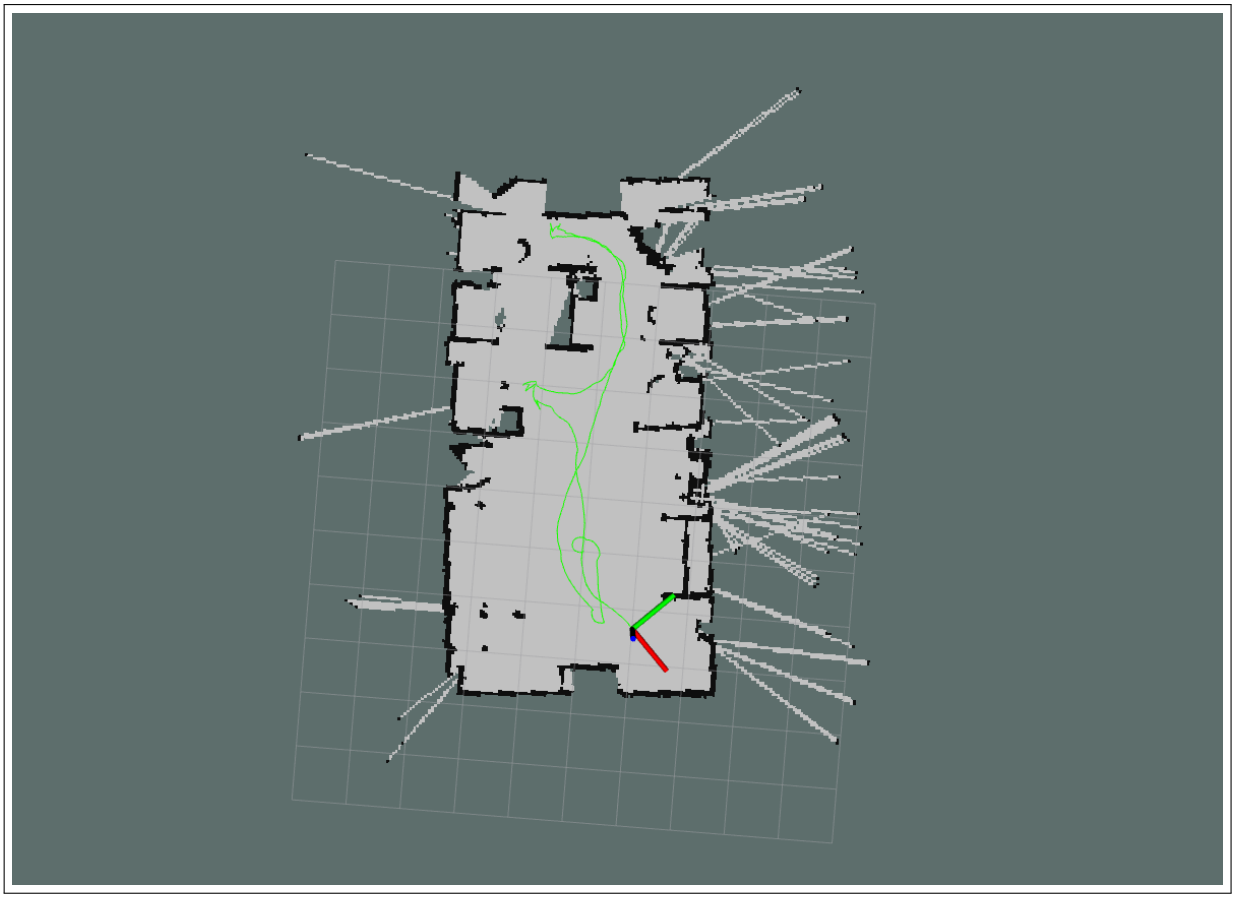
Fonte: o autor.

4.1 SLAM com LiDAR

Neste teste, foi realizado o mapeamento do ambiente de teste utilizando como entrada do algorítmo de SLAM apenas a leitura do sensor LiDAR. A linha verde no mapa representa o percurso realizado pelo robô dentro do mapa. Assim, a Figura 13 mostra o resultado final do

processo de mapeamento.

Figura 13 – Mapa (1) gerado pelo SLAM apenas com LiDAR



Fonte: o autor.

A Tabela 1 a seguir compara os valores das medidas reais dos indicadores da Figura 12 com medidas tiradas pelo mapa gerado nesta etapa.

Tabela 1 – Comparação das medidas entre o mapa 1 e o real

Trecho analisado	Mapa real (cm)	Mapa slam (cm)	Erro (cm)
a	50	45	5
b	18	17	1
c	58	52	6
d	20	18	2
e	97	90	7

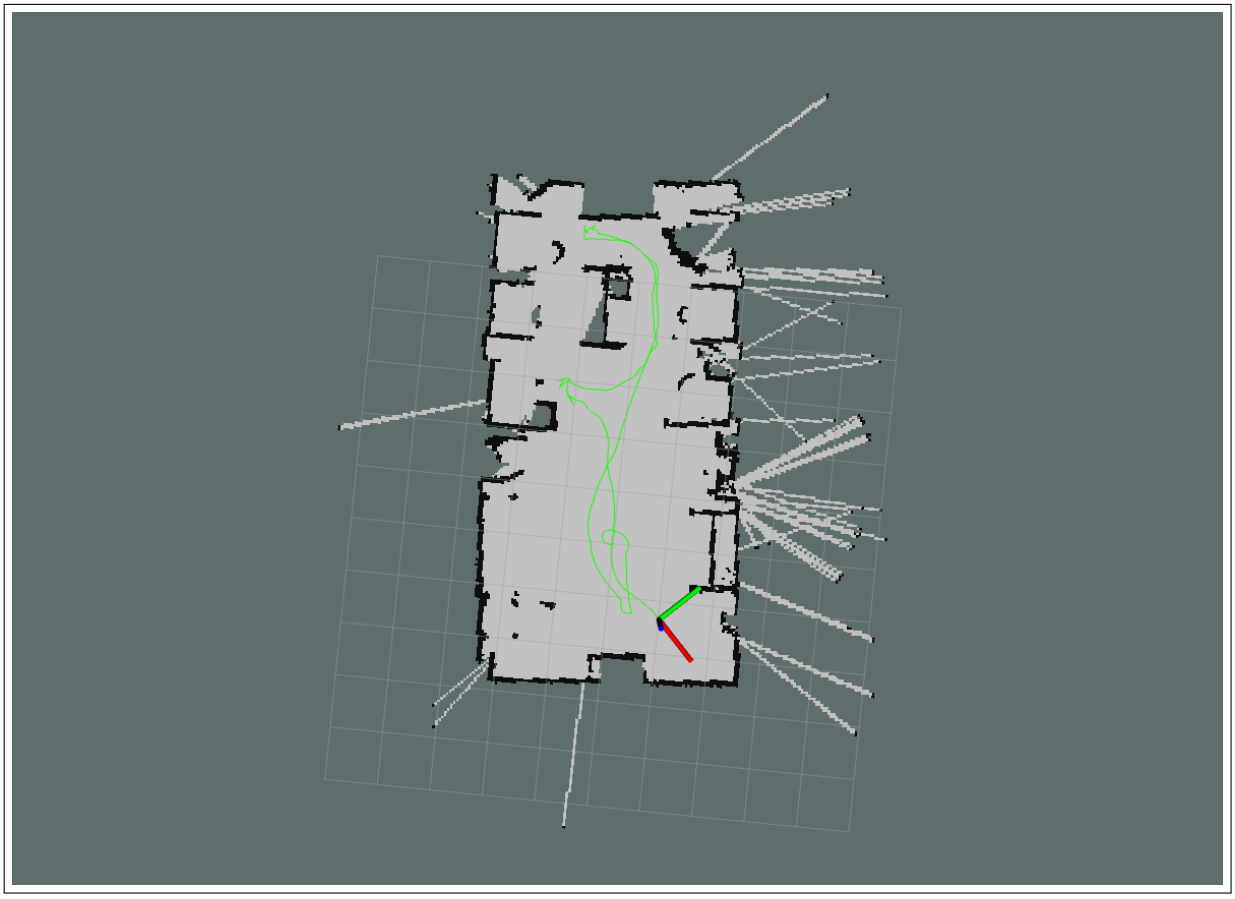
Fonte: o autor.

4.2 SLAM com LiDAR e Odometria

Neste teste, foi realizado o mapeamento do ambiente de teste utilizando como entrada do algorítmo de SLAM a leitura do sensor LiDAR combinado com a informação de odometria.

A linha verde no mapa representa o percurso realizado pelo robô dentro do mapa. Desse modo, A Figura 14 mostra o resultado final do processo de mapeamento.

Figura 14 – Mapa (2) gerado pelo SLAM com LiDAR e odometria



Fonte: o autor.

A Tabela 2 a seguir compara os valores das medidas reais dos indicadores da Figura 12 com medidas tiradas pelo mapa gerado nesta etapa.

Tabela 2 – Comparação das medidas entre o mapa 2 e o real

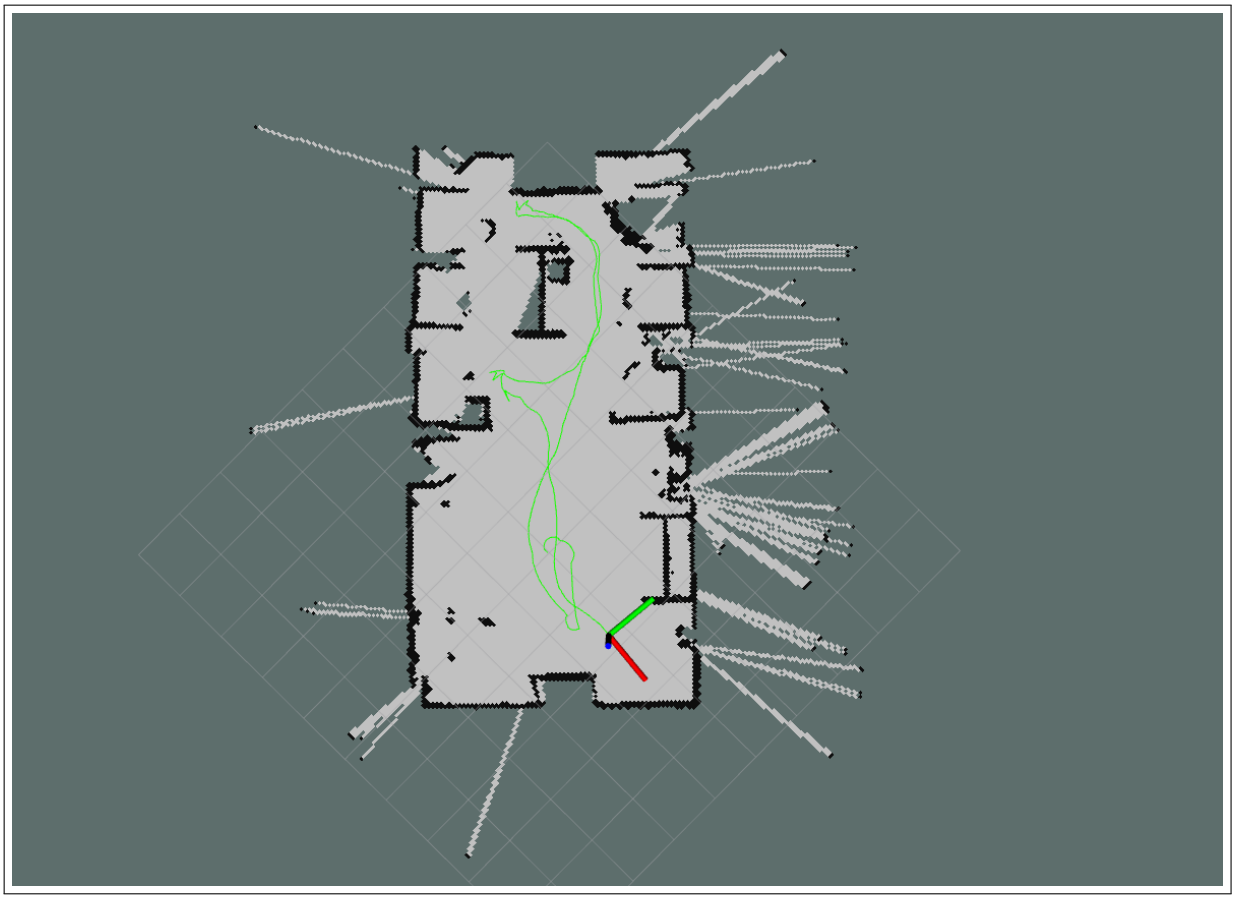
Trecho analisado	Mapa real (cm)	Mapa slam (cm)	Erro (cm)
a	50	48	2
b	18	21	3
c	58	54	4
d	20	19	1
e	97	93	4

Fonte: o autor.

4.3 SLAM com LiDAR e Odometria Filtrada

Neste teste, foi realizado o mapeamento do ambiente de teste utilizando como entrada do algorítmo de SLAM a leitura do sensor LiDAR combinado com a informação de odometria filtrada com EKF por meio da fusão sensorial entre odometria e o sensor IMU. A linha verde no mapa representa o percurso realizado pelo robô dentro do mapa. Com isso, a Figura 15 mostra o resultado final do processo de mapeamento.

Figura 15 – Mapa (3) gerado pelo SLAM com LiDAR e odometria filtrada



Fonte: o autor.

A Tabela 3 a seguir compara os valores das medidas reais dos indicadores da Figura 12 com medidas tiradas pelo mapa gerado nesta etapa.

Tabela 3 – Comparação das medidas entre o mapa 3 e o real

Trecho analisado	Mapa real (cm)	Mapa slam (cm)	Erro (cm)
a	50	48	2
b	18	20	2
c	58	56	2
d	20	19	1
e	97	95	2

Fonte: o autor.

4.4 Análise dos resultados

Conforme apresentado nos seção 4.1 a seção 4.3 deste trabalho, um mesmo ambiente teste foi submetido à diferentes métodos de fusão sensorial para otimização do SLAM. Assim, para aferição da eficiência de cada técnica utilizada, medidas obtidas manualmente de trechos específicos do laboratório foram comparadas aos valores estimados em cada teste.

Desse modo, foi possível confirmar pelas Tabela 1 a Tabela 3 que o uso de mais sensores no mapeamento da região garantiu um resultado mais eficiente, uma vez que o erro das medições foi reduzido. Comprovando a eficiência na técnica proposta.

Além disso, foi possível perceber que em trechos mais próximos do ponto inicial de deslocamento do robô houve erros maiores. Isto se deve à convergência do EKF, que precisa de um certo tempo, a depender do modelo, para aproximar o erro de zero.

5 CONCLUSÕES E TRABALHOS FUTUROS

No decorrer deste trabalho, o objetivo foi implementar a técnica de Simultaneous Localization and Mapping em conjunto com fusão sensorial utilizando o Filtro de Kalman Estendido em um robô móvel terreste utilizando LiDAR e odometria. O trabalho envolveu abordagem sobre o robô utilizado nos testes, os sensores utilizados, o Robotic operational System como ferramenta para integração do sistema robótico, a configuração e integração do pacote Hector_SLAM ao robô e a configuração do EKF por meio do pacote robot_localization.

Ao longo deste estudo, pode-se obter resultados promissores que demonstram a viabilidade e eficácia do uso do SLAM para estimar a localização e mapear o ambiente em tempo real. Os experimentos mostraram que a combinação do giroscópio/acelerômetro e da odometria, utilizando o Filtro de Kalman Estendido, resultou em estimativas de posição mais precisas e em mapas de ocupação detalhados.

Uma das principais contribuições deste trabalho foi a integração bem-sucedida do pacote Hector_SLAM ao robô móvel, permitindo a realização do SLAM de forma eficiente e confiável. A configuração cuidadosa dos parâmetros do pacote, levando em consideração as características do sensor LiDAR e do ambiente de teste, possibilitou a obtenção de resultados de alta qualidade.

Além disso, pode ser identificada algumas limitações durante a execução deste estudo, como o efeito negativo na precisão da odometria causado pelo deslizamento das rodas. Como este efeito gera um erro integrativo, a leitura de posição dada pela odometria torna-se mais imprecisa quanto mais o robô se movimenta. Para isto, foi abordado a utilização do Filtro de Kalman Estendido, por meio do pacote robot_localization para minimizar o erro gerado pela odometria, integrando o giroscópio/acelerômetro na estimação de posição.

Em trabalhos futuros, pode ser explorada a utilização de estratégias avançadas de controle de tragetória, que permitam o robô otimizar sua navegação em tempo real. A integração de um *costmap* também pode ser uma área interessante para a pesquisa, uma vez que o *costmap* é um mapa que reflete a acessibilidade do robô no que diz respeito aos obstáculos presentes.

Em suma, este trabalho demonstrou que a técnica de SLAM integrada com o Filtro de Kalman Estendido apresenta resultados promissores para a estimativa de localização e mapeamento em tempo real. A integração bem-sucedida ao robô móvel, juntamente com a análise dos resultados obtidos, contribui para o conhecimento científico e pode ser aplicada em diversos contextos da robótica e da automação.

REFERÊNCIAS

AL., Q. M. et. Ros: an open-source robot operating system. **ICRA workshop on open source software**, v. 3, n. 3.2, 2009.

ALBUQUERQUE, J. P. d. A. e. e. a. **Probabilidade, Variáveis Aleatórias e Processos Estocásticos**. [S.l.]: Editora Interciência, 2018. v. 2.

ANANDITO, S. F. A. E. W. D. S. P. E. B. U. M. A. Research study of occupancy grid map mapping method on hector slam technique. **2019 International Electronics Symposium (IES)**, IEEE, v. 1, n. 1, p. 238–241, 2019.

BOUAZIZI ALEJANDRO LORITE MORA, T. O. M. A 2d-lidar-equipped unmanned robot-based approach for indoor human activity detection. **Sensors**, Sensors, v. 23, n. 2534, p. 326–330, 2023.

BUCY, R. K. R. New results in linear filtering and prediction theory. **Journal of Basic Engineering**, v. 83, n. 1, p. 95–108, 1961.

GAO YU-XIAN GAI, S. F. L.-F. Simultaneous localization and mapping for autonomous mobile robots using binocular stereo vision system. **2007 International Conference on Mechatronics and Automation**, IEEE, v. 1, n. 1, p. 326–330, 2007.

HOKUYO. **Products Detail**. Detalhes técnicos do sensor LiDAR HOKUYO URG-04LX-UG01. Disponível em: <https://www.hokuyo-aut.jp/search/single.php?serial=166>. Acesso em: 05 jul. 2023.

KALMAN, R. A new approach to linear filtering and prediction problems. **Journal of Basic Engineering**, v. 1, n. 82, p. 35–45, 1960.

KOHLBRECHER, S. **How to set up hector slam for your robot**. Descrição dos frames de referência do pacote hector slam. Disponível em: <https://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot>. Acesso em: 05 jul. 2023.

LEE, K. P. H. C. J. G. Dead reckoning navigation for autonomous mobile robots. **IFAC Proceedings Volumes**, v. 31, n. 3, p. 219–224, 1998.

LIMA MARCUS DAVI DO NASCIMENTO FORTE, F. G. N. B. C. T. T. A. Trajectory tracking control of a mobile robot using lidar sensor for position and orientation estimation. **12th IEEE International Conference on Industry Applications (INDUSCON)**, IEEE, v. 1, n. 1, p. 1–6, 2016.

MEEUSSEN, W. Coordinate frames for mobile platforms. **ROS Documentation**, ROS, v. 1, n. 1, p. 105, 2010.

MOORE, T.; STOUCH, D. A generalized extended kalman filter implementation for the robot operating system. In: **Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)**. [S.l.]: Springer, 2014.

RESEARCHGATE. **A 2.5D Map-Based Mobile Robot Localization via Cooperation of Aerial and Ground Robots**. Ilustração do método de mapeamento de grade de ocupação. Disponível em: <https://www.researchgate.net/figure/Occupancy-grid-map_fig1_321326800>. Acesso em: 12 jul. 2023.

RESEARCHGATE. **Stewart-Gough Platform: Design and Construction with a Digital PID Controller Implementation**. Imagem do sensor MPU 6050 indicando as 3 direções de medições. Disponível em: <https://www.researchgate.net/figure/Gyroscope-sensor-and-accelerometer-MPU-6050_fig5_347285831>. Acesso em: 05 jul. 2023.

SADOWSKI, M. **Hands on with slam_toolbox**. Representação em sequencia de imagem do SLAM. Disponível em: <https://msadowski.github.io/hands-on-with-slam_toolbox>. Acesso em: 12 jul. 2023.

SAMAN, A. H. L. A. B. S. H. M. An implementation of slam with extended kalman filter. **2016 6th International Conference on Intelligent and Advanced Systems (ICIAS)**, IEEE, v. 1, n. 1, p. 1–4, 2016.

SILVA, J. V. D.; KONDOZ, A. Fusion of lidar and camerasensor datafor environment sensing in driverless vehicles. **Sensors**, arXiv.org, abs/1710.06230, n. 1, 2018.

YANG, M. Y. Fusion of camera images and laser scans for wide baseline 3d scene alignment in urban environments. **ISPRS Journal of Photogrammetry and Remote Sensing**, ISPRS, v. 66, n. 1, p. 52–61, 2011.

APÊNDICE A – HECTORSLAM_LIDAR.LAUNCH

```
<?xml version="1.0"?>

<launch>

  <arg name="geotiff_map_file_path" default="$(find hector_geotiff)/maps"/>

  <param name="/use_sim_time" value="true"/>

  <node pkg="tf" type="static_transform_publisher" name="base_link_2_laser"
  args="0 0 0 0 0 0 /base_link /laser 100"/>

  <node pkg="tf" type="static_transform_publisher" name="odom_2_base_link"
  args="0 0 0 0 0 0 /map /base_link 100"/>

  <node pkg="rviz" type="rviz" name="rviz"
    args="-d $(find hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>

  <arg name="tf_map_scanmatch_transform_frame_name" default="scanmatcher_frame"/>
  <arg name="base_frame" default="base_link"/>
  <arg name="odom_frame" default="base_link"/>
  <arg name="pub_map_odom_transform" default="true"/>
  <arg name="scan_subscriber_queue_size" default="5"/>
  <arg name="scan_topic" default="scan"/>
  <arg name="map_size" default="2048"/>

  <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping"
  output="screen">

    <!-- Frame names -->
    <param name="map_frame" value="map" />
    <param name="base_frame" value="$(arg base_frame)" />
```

```
<param name="odom_frame" value="$(arg odom_frame)" />

<!-- Tf use -->
<param name="use_tf_scan_transformation" value="true"/>
<param name="use_tf_pose_start_estimate" value="false"/>
<param name="pub_map_odom_transform" value="$(arg pub_map_odom_transform)"/>

<!-- Map size / start point -->
<param name="map_resolution" value="0.050"/>
<param name="map_size" value="$(arg map_size)"/>
<param name="map_start_x" value="0.5"/>
<param name="map_start_y" value="0.5" />
<param name="map_multi_res_levels" value="2" />

<!-- Map update parameters -->
<param name="update_factor_free" value="0.4"/>
<param name="update_factor_occupied" value="0.9" />
<param name="map_update_distance_thresh" value="0.4"/>
<param name="map_update_angle_thresh" value="0.06" />
<param name="laser_z_min_value" value = "-1.0" />
<param name="laser_z_max_value" value = "1.0" />

<!-- Advertising config -->
<param name="advertise_map_service" value="true"/>

<param name="scan_subscriber_queue_size"
value="$(arg scan_subscriber_queue_size)"/>
<param name="scan_topic" value="$(arg scan_topic)"/>

<!-- Debug parameters -->
<!--
  <param name="output_timing" value="false"/>
```

```
    <param name="pub_drawings" value="true"/>
    <param name="pub_debug_output" value="true"/>
  -->
  <param name="tf_map_scanmatch_transform_frame_name"
  value="$(arg tf_map_scanmatch_transform_frame_name)" />
</node>

<arg name="trajectory_source_frame_name" value="scanmatcher_frame"/>
<arg name="trajectory_update_rate" default="4"/>
<arg name="trajectory_publish_rate" default="0.25"/>
<arg name="map_file_path" value="$(arg geotiff_map_file_path)"/>
<arg name="map_file_base_name" default="hector_slam_map"/>

<node pkg="hector_trajectory_server" type="hector_trajectory_server"
name="hector_trajectory_server" output="screen">
  <param name="target_frame_name" type="string" value="/map" />
  <param name="source_frame_name" type="string" value="$(arg trajectory_source_fr
  <param name="trajectory_update_rate" type="double"
  value="$(arg trajectory_update_rate)" />
  <param name="trajectory_publish_rate" type="double"
  value="$(arg trajectory_publish_rate)" />
</node>

<node pkg="hector_geotiff" type="geotiff_node" name="hector_geotiff_node"
output="screen" launch-prefix="nice -n 15">
  <remap from="map" to="/dynamic_map" />
  <param name="map_file_path" type="string" value="$(arg map_file_path)" />
  <param name="map_file_base_name" type="string"
  value="$(arg map_file_base_name)" />
  <param name="geotiff_save_period" type="double" value="0" />
  <param name="draw_background_checkerboard" type="bool" value="true" />
  <param name="draw_free_space_grid" type="bool" value="true" />
```

```
    <param name="plugins" type="string"
    value="hector_geotiff_plugins/TrajectoryMapWriter" />
  </node>

</launch>
```

APÊNDICE B – HECTORSLAM_ODOM.LAUNCH

```
<?xml version="1.0"?>

<launch>

  <arg name="geotiff_map_file_path" default="$(find hector_geotiff)/maps"/>

  <param name="/use_sim_time" value="true"/>

  <node pkg="tf" type="static_transform_publisher" name="base_link_2_laser"
  args="0 0 0 0 0 0 /base_link /laser 100"/>

  <node pkg="tf" type="static_transform_publisher" name="odom_2_base_link"
  args="0 0 0 0 0 0 /map /base_link 100"/>

  <node pkg="tf" type="static_transform_publisher" name="map_2_odom"
  args="0 0 0 0 0 0 /map /odom 100"/>

  <node pkg="rviz" type="rviz" name="rviz"
    args="-d $(find hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>

  <arg name="tf_map_scanmatch_transform_frame_name"
  default="scanmatcher_frame"/>
  <arg name="base_frame" default="base_link"/>
  <arg name="odom_frame" default="base_link"/>
  <arg name="pub_map_odom_transform" default="true"/>
  <arg name="scan_subscriber_queue_size" default="5"/>
  <arg name="scan_topic" default="scan"/>
  <arg name="map_size" default="2048"/>

  <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping"
  output="screen">
```

```
<!-- Frame names -->
<param name="map_frame" value="map" />
<param name="base_frame" value="$(arg base_frame)" />
<param name="odom_frame" value="$(arg odom_frame)" />

<!-- Tf use -->
<param name="use_tf_scan_transformation" value="true"/>
<param name="use_tf_pose_start_estimate" value="false"/>
<param name="pub_map_odom_transform"
value="$(arg pub_map_odom_transform)"/>

<!-- Map size / start point -->
<param name="map_resolution" value="0.050"/>
<param name="map_size" value="$(arg map_size)"/>
<param name="map_start_x" value="0.5"/>
<param name="map_start_y" value="0.5" />
<param name="map_multi_res_levels" value="2" />

<!-- Map update parameters -->
<param name="update_factor_free" value="0.4"/>
<param name="update_factor_occupied" value="0.9" />
<param name="map_update_distance_thresh" value="0.4"/>
<param name="map_update_angle_thresh" value="0.06" />
<param name="laser_z_min_value" value = "-1.0" />
<param name="laser_z_max_value" value = "1.0" />

<!-- Advertising config -->
<param name="advertise_map_service" value="true"/>

<param name="scan_subscriber_queue_size"
value="$(arg scan_subscriber_queue_size)"/>
```

```
  <param name="scan_topic" value="$(arg scan_topic)"/>

  <!-- Debug parameters -->
  <!--
    <param name="output_timing" value="false"/>
    <param name="pub_drawings" value="true"/>
    <param name="pub_debug_output" value="true"/>
  -->
  <param name="tf_map_scanmatch_transform_frame_name" value="$(arg
  tf_map_scanmatch_transform_frame_name)" />
</node>

<arg name="trajectory_source_frame_name" value="scanmatcher_frame"/>
<arg name="trajectory_update_rate" default="4"/>
<arg name="trajectory_publish_rate" default="0.25"/>
<arg name="map_file_path" value="$(arg geotiff_map_file_path)"/>
<arg name="map_file_base_name" default="hector_slam_map"/>

<node pkg="hector_trajectory_server" type="hector_trajectory_server"
name="hector_trajectory_server" output="screen">
  <param name="target_frame_name" type="string" value="/map" />
  <param name="source_frame_name" type="string" value="$(arg
  trajectory_source_frame_name)" />
  <param name="trajectory_update_rate" type="double" value="$(arg
  trajectory_update_rate)" />
  <param name="trajectory_publish_rate" type="double" value="$(arg
  trajectory_publish_rate)" />
</node>

<node pkg="hector_geotiff" type="geotiff_node" name="hector_geotiff_node"
output="screen" launch-prefix="nice -n 15">
  <remap from="map" to="/dynamic_map" />
```

```
    <param name="map_file_path" type="string" value="$(arg map_file_path)" />
    <param name="map_file_base_name" type="string"
    value="$(arg map_file_base_name)" />
    <param name="geotiff_save_period" type="double" value="0" />
    <param name="draw_background_checkerboard" type="bool" value="true" />
    <param name="draw_free_space_grid" type="bool" value="true" />
    <param name="plugins" type="string"
    value="hector_geotiff_plugins/TrajectoryMapWriter" />
  </node>

</launch>
```

APÊNDICE C – ODOM_EKF.LAUNCH

```
<launch>
    <!-- Odom node (Encoders + IMU) -->

   <node pkg="robot_localization" type="ekf_localization_node"
   name="ekf_odom_node" output="screen" >

       <param name="frequency" value="20"/>

       <param name="sensor_timeout" value="0.1"/>

       <param name="two_d_mode" value="true"/>

       <remap from="odometry/filtered" to="odom/ekf/enc_imu"/>

       <param name="map_frame" value="map"/>

       <param name="odom_frame" value="odom_ekf"/>

       <param name="base_link_frame" value="base_link"/>

       <param name="world_frame" value="odom_ekf"/>

       <param name="transform_time_offset" value="0.0"/>

       <param name="odom0" value="/odometria"/>

       <param name="odom0_differential" value="false" />

       <param name="odom0_relative" value="false" />

       <param name="odom0_queue_size" value="10" />
```

```
<rosparam param="odom0_config">[false,  false, false,
                                false, false, false,
                                true, true, false,
                                false, false, true,
                                false, false, false]</rosparam>
<param name="imu0" value="/mpu_data2"/>
<param name="imu0_differential" value="false" />
<param name="imu0_relative" value="true" />
<param name="imu0_queue_size" value="10" />
<param name="imu0_remove_gravitational_acceleration" value="true" />
<rosparam param="imu0_config">[false,  false, false,
                               false, false, false,
                               false, false, false,
                               true , true, true,
                               true, true, true]</rosparam>
<param name="print_diagnostics" value="true" />
```

```
<param name="debug" value="false" />

<param name="debug_out_file" value="debug_odom_ekf.txt" />

<rosparam param="process_noise_covariance">
[0.05, 0,    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    0, 0, 0,


 0, 0.05, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0,

 0, 0, 0.06, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0,

 0, 0, 0, 0.03, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0,

 0, 0, 0, 0, 0.03, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0.06, 0, 0, 0,  0, 0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0.025,0, 0,  0, 0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0.025,0,  0, 0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0.04, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0,  0.01, 0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0.01, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0.02, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0.01, 0, 0,
```

```
 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0.01, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0.005]</rosparam>

<rosparam param="initial_estimate_covariance">
[1e-9, 0,    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    0, 0,

 0, 1e-9, 0, 0, 0, 0, 0, 0, 0, 0,    0, 0, 0, 0, 0,

 0, 0, 1e-9, 0, 0, 0, 0, 0, 0, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 1e-9, 0, 0, 0, 0, 0, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 1e-9, 0, 0, 0, 0, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 1e-9, 0, 0, 0, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 1 , 0, 0, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 1 , 0, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 1e-9, 0,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,    0, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    1, 0, 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    0, 1 , 0, 0, 0,

 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    0, 0, 1, 0, 0,
```

```
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    0, 0, 0, 1e-9, 0,

        0, 0, 0, 0, 0, 0, 0, 0, 0, 0,    0, 0, 0, 0, 1e-9]</rosparam>

    </node>
</launch>
```