**MESTRADO EM SISTEMAS DE COMPUTAÇÃO**

**HENRIQUE ÁVILA SANTOS**

# UMA AVALIAÇÃO DA CONTRIBUIÇÃO DA FUSÃO DE SENSORES DE POSICIONAMENTO RELATIVO NA PRECISÃO DOS SISTEMAS DE LOCALIZAÇÃO DE ROBÔS MÓVEIS TERRESTRES

Salvador
2020

**HENRIQUE ÁVILA SANTOS**

# UMA AVALIAÇÃO DA CONTRIBUIÇÃO DA FUSÃO DE SENSORES DE POSICIONAMENTO RELATIVO NA PRECISÃO DOS SISTEMAS DE LOCALIZAÇÃO DE ROBÔS MÓVEIS TERRESTRES

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas de Computação da Universidade Salvador UNIFACS, Laureate Internacional Universities, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos.

Salvador
2020

Ficha Catalográfica elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate Internacional Universities.

Santos, Henrique Ávila

Uma avaliação da contribuição da fusão de sensores de posicionamento relativo na precisão dos sistemas de localização de robôs móveis terrestres./ Henrique Ávila Santos.- Salvador, 2021.

105 f.: il.

Dissertação apresentada ao Programa de Pós-Graduação em Sistemas e Computação da Universidade Salvador (UNIFACS), Laureate International Universities, como requisito parcial para obtenção do título de Mestre.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos.

1. Robótica. 2. Sistemas de localização. 3. Fusão de sensores. 4. Odometria. I. Campos, Jorge Alberto Prado de, orient. II. Título.

CDD: 629.892

HENRIQUE ÁVILA SANTOS

# UMA AVALIAÇÃO DA CONTRIBUIÇÃO DA FUSÃO DE SENSORES DE POSICIONAMENTO RELATIVO NA PRECISÃO DOS SISTEMAS DE LOCALIZAÇÃO DE ROBÔS MÓVEIS TERRESTRES

Dissertação apresentada ao Programa de Mestrado em Sistemas e Computação da Universidade Salvador – UNIFACS, Laureate International Universities, como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação e aprovada pela seguinte banca examinadora:

Jorge Alberto Prado de Campos – Orientador _____
Ph.D. in Spatial Information Science, University of Maine, EUA
UNIFACS Universidade Salvador, *Laureate International Universities*

Josemar Rodrigues de Souza _____
Pós-Doutorado em Robótica Autônoma pela, Universidade do Porto, Portugal
Universidade Estadual da Bahia - UNEB

Artur Henrique Kronbauer _____
Doutor em Ciência da Computação pela Universidade Federal da Bahia - UFBA, Brasil
UNIFACS Universidade Salvador, *Laureate International Universities*

Salvador, 15 de dezembro de 2020.

## AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Jorge Alberto Prado de Campos, pela oportunidade, confiança, conselhos e orientações, que foram fundamentais para o desenvolvimento desse trabalho. Agradeço a presença em reuniões semanais para orientar no processo de aprendizagem sempre que foi necessário.

Aos meus pais, Arismar da Silva Santos e Ana Amélia Ávila Santos, pelo amor, carinho e presença durante meu crescimento pessoal e suporte em meu desenvolvimento profissional. Por ter proporcionado em minha vida uma educação de qualidade e ter me ensinado a importância do comprometimento, assim pude me tornar a pessoa que sou hoje.

Ao meu irmão, Roberto Ávila Santos, pela amizade, parceria e fraternidade, o que me ampara em momentos de ansiedade.

À minha companheira, Marina Moor Brandão Lutfi, pelo amor, carinho, apoio e compreensão, por ter me incentivado nos momentos importantes, inclusive durante o desenvolvimento da pesquisa.

Ao Prof. Euclerio Ornellas, por todas as oportunidades concedidas durante a academia, pelo apoio, companheirismo, solidariedade, orientações e ensinamentos que permanecem para a vida.

À equipe do Grupo de Aplicações e Análises Geoespaciais (GANGES) da Universidade Salvador, pelo suporte e conhecimento concedido ao decorrer do projeto.

À equipe do Laboratório de Arquitetura de Computadores e Microcontroladores (ACM) da Universidade Salvador, pelas sugestões, suporte e equipamentos fornecidos sempre que solicitado.

À equipe do Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO) da Universidade do Estado da Bahia pelo apoio e conhecimentos concedidos durante o projeto.

Aos técnicos de laboratório da Universidade Salvador, Washington Mendes e Raony Rios, que sempre estiveram dispostos a colaborar, foram essenciais na construção da plataforma robótica e nas etapas iniciais do trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela concessão da bolsa de estudo do curso de mestrado.

*“O agora importa mais do que qualquer outro momento, pois é o que você está fazendo hoje que está determinando em quem você está se tornando, e quem você está se tornando sempre determinará a qualidade e a direção da sua vida”.*

Hal Elrod

## RESUMO

A localização precisa de um robô é um desafio fundamental e uma das tarefas mais importantes para os sistemas de navegação autônoma. Na navegação autônoma, o robô deve estar ciente de sua localização no ambiente para que possa determinar a trajetória que deve seguir para executar alguma tarefa. Uma das formas mais comuns na robótica móvel para obter a localização do robô é a utilização da odometria baseada no movimento das rodas. O problema deste mecanismo é que ele apresenta uma imprecisão crescente ao longo do tempo e uma sensibilidade elevada quanto ao tipo de pavimento. Uma maneira de melhorar a precisão do posicionamento do robô é através da fusão de outros sensores capazes de medir o deslocamento e rotações do robô. Os sensores mais utilizados para melhorar o conhecimento da posição do robô em plataformas robóticas de baixo custo é a odometria visual e os sensores inerciais. Este trabalho propõe analisar a precisão dos sistemas de posicionamento baseado na odometria obtida das rodas, de sensores inerciais e da odometria visual e o impacto da fusão destes mecanismos na precisão da localização do robô. Para análise da precisão do movimento do robô são comparadas as trajetórias reais de um robô de duas rodas utilizando os sensores individualmente e de forma combinada. O robô utilizado possui os sensores de localização mais usuais em plataformas robóticas de baixo custo e foi construído especialmente para a realização deste experimento. Os resultados do experimento fornecem uma boa indicação do custo-benefício da utilização destes tipos de sensores quando embarcados para realizar a odometria de plataformas robóticas.



**Palavras-chave:** Robótica. Sistemas de localização. Fusão de sensores. Odometria.

## ABSTRACT

The accurate localization of a robot is a fundamental challenge and one of the most important tasks for autonomous navigation systems. In autonomous navigation systems, the robot must keep aware of its location in the environment in order to determine the path it must follow to perform some task. One of the most common ways in mobile robotics to obtain the localization of the robot is the use of odometry based on the movement of the robot's wheels. The problem of this mechanism is that it presents an increasing imprecision over time and a high sensitivity to the type of pavement. One way to improve the robot's positioning accuracy is by fusing data of other sensors capable of measuring the robot's displacement and speed. The most used sensors to improve the knowledge of the robot's position on low-cost robotic platforms are visual odometry and inertial sensors. This paper proposes to analyze the accuracy of the positioning systems based on the odometry obtained from the wheels, inertial sensors and visual odometry and the impact of the fusion of these mechanisms on the accuracy of the robot's localization. To analyze the accuracy of the robot's movement, the real trajectory of a two-wheeled robot is compared using the navigating system's sensors, individually and combined. The robot uses the most common localization sensors on low-cost robotic platforms and was especially built for this experiment. The results of this experiment provide a good indication of the cost-benefit using these types of sensors when embedded to perform the mobile robot's odometry.



**Keywords:** Robotics. Localization systems. Sensor fusion. Odometry.

## LISTA DE FIGURAS

Figura 1 - Braço mecânico industrial Unimate ..... 16
Figura 2 - Exemplos de robôs móveis autônomos: a) Curiosity Rover, b) FlatFish e c) Atlas 17
Figura 3 - O ciclo see-think-act de robôs móveis autônomos ..... 24
Figura 4 - Reprodução de dados observados por sensores do Turtlebot em simulação. ..... 25
Figura 5 - Exemplos de sensores utilizados no mapeamento do ambiente e detecção de obstáculos: a) Sensor de ultrassom, b) Sensor LIDAR e c) Kinect Azure ..... 26
Figura 6 - Veículo autônomo com sensores embarcados ..... 30
Figura 7 - Planejamento de rota para limpeza de ambiente fechado ..... 31
Figura 8 - Exemplo de navegação autônoma em simulação computacional ..... 33
Figura 9 - Plataformas robóticas ..... 37
Figura 10 - Propriedades do ROS ..... 38
Figura 11 - Sistema de arquivos e estruturação de pacotes ..... 40
Figura 12 - Comunicação entre os nós ROS ..... 42
Figura 13 - Representação do grafo computacional do ROS ..... 42
Figura 14 - Simulação do Turtlebot2 com openai_ros e Gazebo ..... 45
Figura 15 - Gráfico de recompensa através de machine learning do openai_ros ..... 45
Figura 16 - Visualização de sensores no Rviz ..... 46
Figura 17 - Exemplos de Grid Map em a) 2D e b) 3D ..... 50
Figura 18 - Exemplo de mapa Point Cloud de uma escada de incêndio ..... 51
Figura 19 - Exemplo de mapa de características utilizando o LIDAR ..... 52
Figura 20 - Experimento utilizando AMCL ..... 56
Figura 21 - MPU 6050 ..... 58
Figura 22 - Diagrama de blocos do IMU no sistema ..... 58
Figura 23 - Motor de corrente contínua e roda com encoder acoplado ..... 59
Figura 24 - Modelo do sistema de odometria do robô ..... 60
Figura 25 - Câmera Kinect ..... 62
Figura 26 - Fluxograma de aquisição da odometria visual ..... 63
Figura 27 - 2WD Wiring ..... 66
Figura 28 - Modelo do sistema eletrônico de alimentação e transmissão de dados ..... 67
Figura 29 - Plataforma Robótica Móvel ..... 68
Figura 30 - Reconfiguração dinâmica do PID ..... 70
Figura 31 - Velocidade linear e velocidade angular em tempo real ..... 70
Figura 32 - Sistema de malha fechada com controlador proporcional ..... 71
Figura 33 - Oscilação sustentada com período Pcr ..... 71
Figura 34 - Resposta do sistema robótico com oscilação sustentada ..... 71
Figura 35 - Teste de odometria ..... 73

Figura 36 - TF da plataforma robótica....74
Figura 37 - Mapa do laboratório de pesquisa construído pela plataforma robótica ....75
Figura 38 - Visão geral: robot_localization e pilha de navegação do ROS....77
Figura 39 - Configurações do ekf_localization_node....77
Figura 40 - Diagrama de blocos dos nós rgbd_odometry e stereo_odometry ....79
Figura 41 - Exemplo de aplicação do OpenCV utilizando o SURF....81
Figura 42 - Correspondência de características com imagem cortada ....82
Figura 43 - Correspondência de características com imagem ampliada ....82
Figura 44 - Percurso realizado pelo robô durante o experimento com a indicação dos erros lineares e angulares....86
Figura *45* - Representação dos vetores inicial e final da rotação do robô ....87
Figura 46 - Odometria resultante do sistema de localização em teste utilizando apenas o encoder ....89
Figura 47 - Odometria resultante do sistema de localização em teste utilizando o encoder e IMU ....92
Figura 48 - Odometria resultante do sistema de localização em teste utilizando o encoder, IMU e Kinect....94

## LISTA DE TABELAS

Tabela 1 – Regras de sintonia de Ziegler-Nichols ..... 72
Tabela 2 - Resultados experimentais utilizando somente o Encoder ..... 89
Tabela 3 - Resultados experimentais utilizando Encoder e Kinect. ..... 90
Tabela 4 - Resultados experimentais utilizando Kinect e IMU ..... 91
Tabela 5 - Resultados experimentais utilizando Encoder e IMU ..... 91
Tabela 6 - Resultados experimentais utilizando Encoder, IMU e Kinect ..... 93
Tabela 7 - Média dos erros RMS linear e angular de cada configuração ..... 95
Tabela 8 - Variação percentual dos erros lineares RMS entre configurações ..... 95
Tabela 9 - Variação percentual dos erros angulares RMS entre configurações ..... 96

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AI | Artificial Intelligence |
| AMCL | Adaptive Monte Carlo Localization |
| EKF | Extended Kalman Filter |
| FPS | Frame per Second |
| F2F | Frame-to-Frame |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| G2O | General Framework for Graph Optimization |
| IMU | Inertial Measurement Unit |
| LIDAR | Light Detection and Ranging |
| LIPO | Lithium Polymer |
| MCL | Monte Carlo Localization |
| NASA | National Aeronautics and Space Administration |
| OPENCV | Open Computer Vision |
| OV | Odometria Visual |
| RANSAC | Random Sample Consensus |
| ROS | Robot Operating System |
| SIFT | Scale-Invariant Feature Transform |
| SLAM | Simultaneous Localization and Mapping |
| STA | See-Think-Act |
| SURF | Speed up Robust Features |
| UKF | Unscented Kalman Filter |
| 2WD | Two Wheel Drive |

**SUMÁRIO**

**1 INTRODUÇÃO ..... 15**
1.1 MOTIVAÇÃO ..... 17
1.2 OBJETIVO ..... 19
1.3 METODOLOGIA ..... 20
1.4 CONTRIBUIÇÕES ..... 21
1.5 AUDIÊNCIA ..... 21
1.6 ESTRUTURA DA MONOGRAFIA ..... 21
**2 SISTEMAS DE NAVEGAÇÃO AUTÔNOMA ..... 23**
2.1 COMPONENTES DA NAVEGAÇÃO AUTÔNOMA ..... 23
**2.1.1 O componente *See*: Percepção, localização e posicionamento ..... 25**
**2.1.2 O Componente *Think*: Robótica cognitiva ..... 30**
**2.1.3 O Componente *Act*: Controle do movimento ..... 32**
2.2 TRABALHOS RELACIONADOS: MELHORIAS NOS SISTEMAS DE NAVEGAÇÃO ..... 33
**3 ROBOT OPERATING SYSTEM ..... 37**
3.1 ARQUITETURA E CONCEITOS DO ROS ..... 39
**3.1.1 Nível do sistema de arquivos ..... 39**
**3.1.2 Nível de grafo computacional ..... 40**
**3.1.3 Nível da comunidade ..... 43**
*3.2 SOFTWARE* PARA SIMULAÇÃO COMPUTACIONAL NO ROS ..... 43
*3.3 SOFTWARE* DE VISUALIZAÇÃO DE DADOS ..... 45
**4 SISTEMAS DE LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEO ..... 48**
4.1 TIPOS DE MAPEAMENTO DO AMBIENTE ..... 48
4.2 MODELOS DE MAPEAMENTO PROBABILÍSTICOS ..... 52
**4.2.1 Filtro de Kalman Estendido ..... 53**
**4.2.2 Localização de Monte Carlo Adaptativa ..... 55**
4.3 Abordagem Multissensorial para Sistemas de Localização ..... 56
**4.3.1 Navegação Inercial ..... 57**
**4.3.2 Odometria ..... 58**
**4.3.3 Odometria Visual ..... 61**
**5 CONSTRUÇÃO E PREPARAÇÃO DA PLATAFORMA ROBÓTICA PARA TESTES ..... 64**
*5.1 HARDWARE* DA PLATAFORMA ROBÓTICA ..... 65
**5.1.1 Esquema de Comunicação e Alimentação ..... 66**
**5.1.2 Construção da plataforma ..... 67**
**5.1.3 Configurações Iniciais ..... 69**

**5.1.4 Sintonização do Controle PID ..... 69**
**5.1.5 Configurações de Odometria ..... 73**
**5.1.6 Parâmetros de Mapeamento ..... 73**
5.2 SOFTWARE PARA SLAM ..... 75
**5.2.1 Posicionamento Relativo com Fusão Sensorial ..... 76**
**5.2.2 Odometria Visual ..... 78**
**5.2.3 Algoritmos utilizados na Odometria Visual ..... 80**
**5 AVALIAÇÃO E RESULTADOS EXPERIMENTAIS ..... 84**
5.1 PERCURSO QUADRADO ..... 85
**5.1.1 Primeira configuração: *Encoder* ..... 88**
**5.1.2 Segunda configuração: *Encoder* e Kinect ..... 90**
**5.1.3 Terceira configuração: Kinect e IMU ..... 90**
**5.1.4 Quarta configuração: *Encoder* e IMU ..... 91**
**5.1.5 Quinta configuração: *Encoder*, IMU e Kinect ..... 93**
5.2 DISCUSSÃO DOS RESULTADOS ..... 94
**6 CONCLUSÃO ..... 98**
6.1 LIMITAÇÕES ..... 99
6.2 TRABALHOS FUTUROS ..... 100
**REFERÊNCIAS ..... 102**

# 1 INTRODUÇÃO

A vontade e a determinação humana para criar um dispositivo com a capacidade de realizar de forma automática tarefas complexas e/ou repetitivas é uma ambição que há muito habita o imaginário das sociedades modernas. A palavra "robô", derivada do termo Tcheco "*robota*", que significa "trabalho forçado", foi introduzida na literatura pela primeira vez em 1920 na peça "*R.U.R.: Rossum's Universal Robots*" escrita por Karel Capek (GASPARETTO; SCALERA, 2019). A palavra "robótica", entretanto, só viria a se popularizar através de histórias publicadas entre 1938 e 1942 pelo escritor Isaac Asimov. Asimov ficou conhecido pela criação das três leis fundamentais para governar o comportamento de um robô. A primeira lei define que um robô não deve machucar um ser humano ou por inatividade permitir que um humano seja ferido; a segunda lei ressalta que um robô deve obedecer às ordens dadas pelos humanos, exceto ordens que entrem em conflito com a primeira lei; a terceira lei determina que um robô deve proteger sua própria existência, contanto que não entre em conflito com a primeira ou a segunda lei. Até a década de 50, a despeito do rápido crescimento tecnológico experimentado à época, o termo robótica só aparecia em histórias de ficção científica e no inconsciente coletivo (GASPARETTO; SCALERA, 2019; HOCKSTEIN et al., 2007).

O primeiro robô real relatado na literatura científica só iria aparecer em meados do século XX. O Unimate, fabricado pela General Motors (Figura 1), foi o primeiro robô industrial desenvolvido para auxiliar na linha de produção automobilística. Mais especificamente, o Unimate era uma máquina que executava uma sequência de comandos programados para realizar tarefas repetitivas. O robô da GM, entretanto, apresentava elevado riscos de acidentes de trabalho, pois não tinha percepção do espaço, não possuía capacidade para avaliar a execução de suas tarefas e também não possuía nenhuma comunicação com o ambiente externo (GASPARETTO; SCALERA, 2019; HOCKSTEIN et al., 2007).

Figura 1 - Braço mecânico industrial Unimate


Fonte: Gasparetto e Scalera (2019).

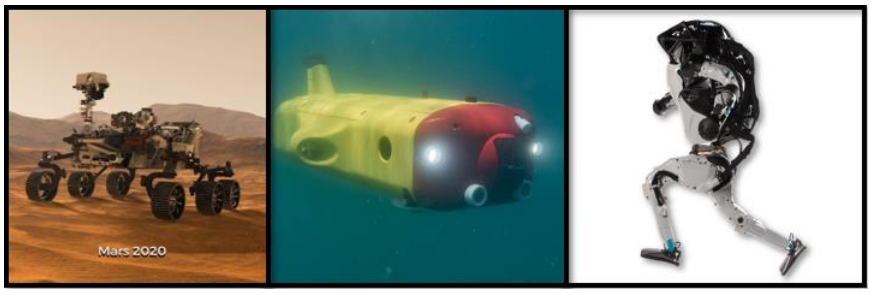
Os avanços tecnológicos, principalmente a partir da segunda metade do século XX, proporcionaram a evolução do ferramental para composição de *hardware* e *software*. Assim, foram desenvolvidas novas linguagens de programação e máquinas com arquitetura mais sofisticadas, o que permitiu que os novos dispositivos passassem a manipular uma grande quantidade de informações e ter a capacidade de receber, processar, armazenar e transmitir um número cada vez maior de dados. O progresso tecnológico impulsionou também a elaboração de novos métodos e recursos computacionais, o que estabeleceu alicerces indispensáveis para o aperfeiçoamento e robustez dos *softwares*, proporcionando o desenvolvimento de produtos que atendem aos crescentes requisitos de qualidade, confiabilidade e portabilidade. Finalmente, mas não menos importante, os avanços tecnológicos resultaram em numerosos impactos sociais, uma vez que a modernização no âmbito científico levou a transformação do estilo de vida, da qualidade de vida e das demandas da sociedade contemporânea.

Na área da robótica, as inovações tecnológicas das últimas décadas têm permitido incorporar aos produtos um certo nível de inteligência. Nesse contexto, a inteligência robótica pode se resumir à capacidade de observar, interagir, analisar, resolver problemas, adaptar-se a novas situações e ter a habilidade de aprender através de experiências, imitando o comportamento cognitivo do ser humano. A implementação de inteligência artificial nos robôs é uma área bastante ativa e existem diversas iniciativas acadêmicas, industriais e governamentais para a pesquisa e desenvolvimento de novos produtos. As pesquisas nesta área são diversas e multidisciplinares, envolvendo tópicos como aprendizado de máquinas, reconhecimento de padrões, visão computacional, redes neurais, entre outros.

A inteligência artificial e a robótica estão virtualmente em todas as áreas de atuação e em todos os ambientes: nas profundezas abissais dos oceanos, em outros planetas do sistema solar e principalmente nos ambientes cotidianos. Somente a título de ilustração, destacamos

alguns robôs móveis autônomos: o Curiosity Rover criado pela NASA para a exploração de Marte (Figura 2.a), o veículo submarino FlatFish projetado pelo SENAI CIMATEC com o objetivo de realizar inspeções visuais tridimensionais na exploração de petróleo e gás em águas profundas (Figura 2.b) e o robô humanoide ATLAS desenvolvido pela Boston Dynamics, uma plataforma de pesquisa projetada para prover serviços emergenciais em operações de busca, salvamento e realizar tarefas como desligar válvulas, abrir portas e operar equipamentos em ambientes onde os seres humanos não poderiam sobreviver (Figura 2.c).

Figura 1 - Exemplos de robôs móveis autônomos: a) Curiosity Rover, b) FlatFish e c) Atlas




a) b) c)

Fonte: a) NASA/JPL-Caltech (2019); b) ARNOLD e MEDAGODA (2018); c) BostonDynamics (2020).

## 1.1 MOTIVAÇÃO

Os robôs móveis variam em tamanho, complexidade e propósito. Uma parcela considerável desses robôs, entretanto, objetiva ser autônoma nas suas decisões e movimentações. Esta característica torna o sistema de navegação dessas máquinas um dos principais componentes para garantir a execução da tarefa e a segurança das pessoas, do ambiente e do próprio robô. Desta forma, é de fundamental importância que o robô mantenha de forma contínua e precisa o conhecimento da sua posição e orientação no espaço, seja capaz de identificar obstáculos e tenha o suporte de um sofisticado algoritmo que permita a sua movimentação de forma eficiente, eficaz e segura.

O conjunto de informações necessárias para determinação da localização e orientação das plataformas robóticas é proveniente de observações sucessivas de sensores embarcados no próprio robô. Esses dispositivos eletrônicos realizam a coleta de amostras analógicas de grandezas físicas (e.g. distância, velocidade, aceleração tridimensional, taxa de rotação), utilizam imagens do ambiente através de câmeras de vídeo digitais simples ou de profundidade ou qualquer outra fonte de informação que permita inferir o quanto o robô se deslocou e em

qual direção.

De forma geral, a navegação de robôs autônomos se resume a responder três perguntas fundamentais: “Onde estou?”, “Para onde vou?” e “Como faço para chegar no meu destino?”. Objetivando solucionar essas questões, pesquisadores e engenheiros desenvolveram sensores, sistemas, tecnologias e técnicas para que a plataforma robótica móvel fosse capaz de conhecer sua localização e orientação, e navegar em determinado ambiente.

Uma das formas mais rudimentares para a determinação da movimentação dos robôs terrestres é a utilização da odometria baseada no movimento das rodas. Este mecanismo, entretanto, apresenta uma imprecisão crescente ao longo do tempo e uma sensibilidade elevada ao tipo de pavimento. Desta forma, é comum associar outros sensores para coletar dados da posição e orientação do robô, tais como as câmeras de vídeo, lasers, sistema de posicionamento global por satélite (GNSS), sensores ultrassônicos e os sensores inerciais, somente para citar os mais usados.

Não existe uma solução única para resolver as questões associadas à navegação do robô. A solução adotada varia em função dos sensores embarcados na plataforma robótica, da complexidade do ambiente, da tarefa a ser executada e da capacidade de processamento da plataforma robótica. A combinação de várias técnicas, entretanto, tem produzido os resultados mais satisfatórios. Um exemplo clássico é a combinação das técnicas de medição de posição relativas com as técnicas de medição de posição absoluta (BORENSTEIN et al., 1997). O grupo de medição de posições relativas, também conhecido como “*dead-reckoning*”, se baseia em dados de odometria e da navegação inercial. Sob outra perspectiva, o grupo de medição de posições absolutas são sistemas baseados em algum tipo de referencial. Nestes sistemas destacam-se as bússolas magnéticas, faróis de rádio (*beacons*), sistemas de posicionamento global, navegação entre pontos de referência (*landmarks*) e casamento de modelos (*Model Matching*). Por um lado, a combinação dos métodos de posicionamento relativo e absoluto proporciona informações mais precisas e confiáveis para os sistemas de localização das plataformas robóticas (BORENSTEIN et al., 1997) e permite amenizar as perturbações provocadas ocasionalmente por algum dos sensores. Por outro lado, o uso combinado de diversos sensores para determinar a movimentação dificulta o tratamento das diferentes incertezas e imprecisões de cada sensor e dificulta a determinação do modelo de observação e movimentação mais apropriado. No intuito de mitigar estes problemas, uma solução bastante utilizada nas plataformas robóticas tem sido a utilização do Filtro de Kalman Estendido (EKF) (DEILAMSALEHY; HAVENS, 2017; HUSSEIN et al., 2016).

O filtro de Kalman é um algoritmo de estimativa de estados de um sistema alimentado

por medições contendo incertezas e fornece recursos para lidar com sistemas não lineares de uma forma eficiente. O EKF é um algoritmo que realiza a fusão de dados, processo que envolve a associação, correlação e combinação de dados provenientes de fontes distintas. Assim, o EKF é uma alternativa interessante para os sistemas de localização de robôs, uma vez que a implementação desse filtro resulta na composição de dados mais precisos, proporcionam a suavização de dados com perturbação, linearização de estados do sistema e estimativa de parâmetros em sistemas de posicionamento global (DEILAMSALEHY; HAVENS, 2017; FARAGHER, 2012).

Desta forma, é pressuposto que quanto maior o número de sensores com parâmetros de localização utilizados no EKF, melhor será a precisão da informação de posicionamento e orientação no resultado de odometria. Esta pressuposição suscita uma das questões de pesquisa norteadoras deste trabalho, que é avaliar a influência que as diferentes configurações de sensores embarcados nos robôs têm na determinação da localização física do robô.

## 1.2 OBJETIVO

Este trabalho propõe analisar o impacto da combinação de diferentes sensores na precisão do sistema de localização de um robô móvel utilizando o algoritmo EKF para a fusão dos dados de odometria. Esta análise busca também identificar como cada sensor afeta os principais aspectos de navegação, isto é, o deslocamento e rotação da plataforma robótica. O objetivo desta análise é auxiliar na tomada de decisões os estudantes, pesquisadores e projetistas de robôs quando da escolha dos sensores a serem utilizados pelo sistema de navegação, subsidiando a avaliação do custo-benefício das diversas combinações e do impacto que cada tipo de sensor tem na precisão do posicionamento do robô.

O objetivo principal desse trabalho é fornecer uma análise quantitativa dos impactos individuais e combinados dos diversos sensores do sistema de localização robótico, utilizando, para isso, ferramentas matemáticas e indicadores estatísticos. Os sensores utilizados nesta avaliação são: 1) *encoders* para mensurar através da rotação das rodas o deslocamento e a orientação da plataforma, 2) a unidade de medida inercial (IMU) composta por giroscópios e acelerômetros, sensores capazes de determinar acelerações, velocidades e deslocamentos nos eixos principais e 3) uma câmera Kinect, que baseada nas imagens capturadas realiza o processo de odometria visual.

Com a finalidade de atingir o objetivo geral deste trabalho foram definidos os seguintes objetivos específicos:

- Identificar na literatura científica os principais sensores utilizados pelos sistemas de navegação de robôs terrestres e definir sensores de baixo custo mais utilizados em robôs autônomos.
- Configurar o simulador de sistemas robóticos Gazebo e o *Robot Operating System* (ROS) para identificar os procedimentos de coleta, armazenamento e visualização de dados obtidos por sensores no ambiente virtual.
- Construir uma plataforma robótica móvel baseada em ROS e com os sensores definidos anteriormente.
- Configurar os componentes atuadores da plataforma robótica para promover movimentos suaves e pontuais.

- Identificar técnicas de fusão sensorial aplicadas a robôs autônomos e definir metodologia de avaliação do sistema de localização.
- Desenvolver metodologia de avaliação do impacto de diferentes sensores na precisão dos sistemas de localização em ambiente físico.
- Executar experimentos, coletar e analisar dados armazenados.

## 1.3 METODOLOGIA

Esta é uma pesquisa de natureza aplicada, com objetivos de caráter explicativo, que emprega uma abordagem quantitativa com procedimentos fundamentados em pesquisa experimental e condução em laboratório.

Trata-se de pesquisa aplicada porque visa o desenvolvimento de uma avaliação direcionada para o problema da precisão dos sistemas de localização de plataformas robóticas móveis em ambientes fechados. É uma pesquisa explicativa, pois tem por objetivo utilizar instrumentos de coleta de dados para observação sistemática da precisão na localização em veículos robóticos. A pesquisa tem abordagem quantitativa, tendo em vista que a validação do modelo possui caráter objetivo, utilizando métodos matemáticos e indicadores estatísticos para os resultados obtidos pela avaliação. Quanto aos procedimentos técnicos, trata-se de uma pesquisa experimental, com o propósito de executar testes controlados em laboratório, analisar os resultados, compará-los e obter informações conclusivas. Quanto à localização, esta é uma pesquisa de laboratório, pois a construção, emprego e validação do projeto foi conduzido em ambiente controlado.

A avaliação da precisão do sistema de localização foi implementada em cinco experimentos, testados em *software* de simulação e executados em um laboratório com um robô físico instruído para navegar em um percurso predefinido. O percurso escolhido para realizar análise dos dados é um quadrado com diagonal de um metro.

O estudo compara os resultados do sistema de localização apurados da fusão de dados utilizando diferentes combinações de sensores: o primeiro teste utiliza somente as informações coletadas pelo *encoder*, o segundo utiliza a combinação de informações do *encoder* e do Kinect, o terceiro utiliza dados do Kinect e IMU, o quarto funde dados do *encoder* e IMU e na última configuração são adotados dados combinados dos três sensores (i.e., *encoders*, IMU e Kinect).

## 1.4 CONTRIBUIÇÕES

Espera-se que a avaliação, resultados e discussões documentadas nessa pesquisa sirvam de referência para estudantes, pesquisadores, projetista e aficionados pela área de robótica, na escolha de sensores para sistemas de localização. Espera-se também que a avaliação da contribuição das diversas configurações de sensores subsidie a tomada de decisão na especificação de veículos robóticos de baixo custo.

## 1.5 AUDIÊNCIA

Esse trabalho tem como audiência um público diversificado. O público-alvo dessa pesquisa são estudantes, engenheiros e pesquisadores na área de robótica, em geral, e dos sistemas de navegação de veículos autônomos, em particular. Esta diversidade de público deve-se principalmente as discussões gerais sobre a robótica móvel, a utilização do Sistema Operacional para Robôs, a documentação do processo de definição e construção de um veículo robótico, a documentação da metodologia para configuração do sistema de localização do robô e, principalmente, pelos experimentos e análises do impacto de diferentes sensores na navegação do robô. Espera-se que os resultados apresentados possam embasar a escolha dos sensores a serem embarcados em robôs autônomos.

## 1.6 ESTRUTURA DA MONOGRAFIA

O restante desta dissertação está organizado em seis capítulos. O Capítulo 2 contém conhecimentos gerais que são importantes para o entendimento desse trabalho, tais como:

percepção, localização, posicionamento, cognição e navegação, além de discutir diversos trabalhos relacionados ao tema.

O Capítulo 3 introduz alguns conceitos fundamentais do ROS, apresenta a estruturação de arquivos, algumas funcionalidades básicas do sistema, orientações sobre compartilhamento de informações e repositórios *online*, sugere programas para simulação computacional e para visualização de dados.

O Capítulo 4 apresenta o problema do Sistema de Localização e Mapeamento Simultâneo (SLAM) e os sensores utilizados na percepção. Também é feita uma discussão sobre os tipos de mapeamento de ambientes utilizando modelos probabilísticos, enfatizando o Filtro de Kalman Estendido e o sistema de localização probabilística que utiliza a abordagem AMCL (*Adaptative Monte Carlo Localization*). Este capítulo finaliza com a demonstração das técnicas de tratamento, os dados que devem ser enviados ao sistema de localização do robô para cada sensor empregado nesse estudo, apresentando as funcionalidades desses dispositivos na teoria, as equações matemáticas e os métodos aplicados para alcançar a odometria individual de cada um dos sensores utilizados.

O Capítulo 5 fornece informações sobre a construção e preparação da plataforma robótica para os testes. Neste capítulo é exposto com detalhes os procedimentos utilizados para implementação do sistema, são indicados repositórios, pacotes e algoritmos utilizados nessa pesquisa e são descriminadas algumas funcionalidades dos projetos *open source* tomados como referência nesse trabalho.

O Capítulo 6 apresenta a metodologia para a realização dos testes, os resultados experimentais, discute as possibilidades de combinação dos sensores, avalia os resultados de cada combinação e realiza uma análise crítica das cinco configurações de sensores no sistema de localização da plataforma robótica baseada nos resultados obtidos nos experimentos.

O Capítulo 7 realiza uma síntese da dissertação retomando os aspectos cruciais do trabalho, apresenta as limitações encontradas no desenvolvimento do projeto e explora possibilidades de extensão dessa pesquisa em trabalhos futuros.

# 2 SISTEMAS DE NAVEGAÇÃO AUTÔNOMA

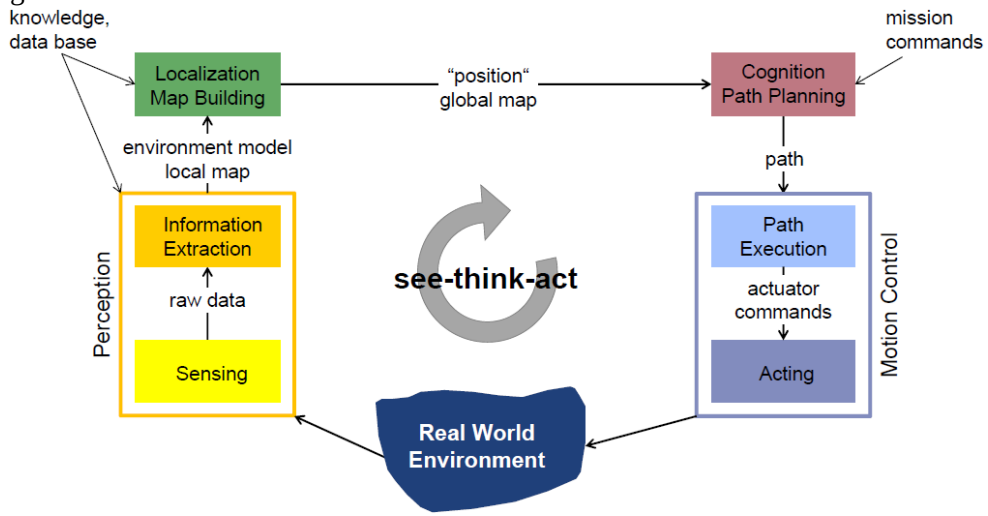
A navegação autônoma de robôs depende de uma série de componentes de *hardware* e *software* atuando de forma cíclica e coordenada para propiciar a movimentação do robô buscando atingir a sua destinação. O processo se inicia com a captura dos dados do ambiente que permitem inferir a posição e orientação do robô (egomoção). Em seguida, estas mesmas informações são utilizadas para alimentar a base de conhecimento que forma o mapa do ambiente na vizinhança do robô. O mapa, associado a um objetivo ou tarefa, alimentam o módulo responsável por definir a melhor trajetória a ser realizada visando atingir o objetivo. Uma vez determinada a melhor trajetória, um algoritmo converte as instruções de navegação em comandos específicos para os atuadores da plataforma robótica em questão, que finalmente executa a ação definida. Enquanto executa a ação, os sensores embarcados capturam os dados do ambiente, iniciando mais um ciclo do processo de navegação.

A próxima seção discute os detalhes de cada módulo da navegação autônoma e os principais trabalhos relacionados ao tema desta dissertação, que trata da combinação ou fusão de sensores com o objetivo de melhorar o sistema de navegação autônoma.

## 2.1 COMPONENTES DA NAVEGAÇÃO AUTÔNOMA

O sistema de navegação de veículos e plataformas robóticas é um complexo agrupamento de equipamentos e programas dedicados a difícil tarefa de movimentar o robô de forma autônoma, segura e eficiente, visando atingir o seu objetivo. Este sistema complexo é bem sumarizado no modelo STA (*See-Think-Act*) apresentado por Siegwart et.al. (2014) (Figura 3).

Figura 3 - O ciclo see-think-act de robôs móveis autônomos




Fonte: Siegwart et.al. (2014).

No modelo STA, o componente "*See*" representa a percepção do robô quanto a sua posição e orientação no ambiente, além da identificação de objetos móveis (e.g. pessoas e outros veículos) e da localização dos objetos imóveis (e.g. paredes e objetos fixos). Ao contrário do que o nome sugere, o "*See*" não se vale apenas de imagens capturadas por câmeras de vídeos, mas também de todo e qualquer dado oriundo dos diversos sensores utilizados para navegação e construção de mapas, como por exemplo: laser, radar, odometria das rodas, GPS, acelerômetro, bússola, giroscópio, somente para citar alguns.

O componente "*Think*" representa processo cognitivo do sistema de navegação do robô. Este componente é responsável pelo processamento de toda informação utilizada para a navegação autônoma, que vai desde o tratamento dos dados brutos dos sensores, passando pela combinação dos dados processados, pela construção de mapas do ambiente e pela identificação obstáculos móveis, culminando com a determinação do melhor caminho ou de caminhos alternativos para alcançar o objetivo de forma eficiente e segura, tanto para as pessoas quanto para o robô e a infraestrutura a sua volta.

Finalmente, o componente "*Act*", converte as instruções de navegação de alto-nível em comandos para os atuadores da plataforma robótica para executar o movimento planejado. Os processos do modelo "*See-Think-Act"* são contínuos e se repetem até que a tarefa ou objetivo de navegação seja atingido. As próximas subseções discutem os detalhes de cada componente do modelo STA.

### 2.1.1 O componente *See*: Percepção, localização e posicionamento

Os avanços tecnológicos e a redução de custos dos componentes eletrônicos, principalmente no que se refere ao desenvolvimento de sensores utilizados no sensoriamento ou percepção do ambiente, vem tornando estes dispositivos cada vez mais baratos, menores e mais precisos, proporcionando, desta forma, a disseminação e popularização desta tecnologia nas mais diversas áreas. Na área industrial, por exemplo, existem sensores para a automação de sistemas e processos de diversos segmentos. Geralmente, os processos industriais necessitam de sensores para medições das grandezas físicas utilizada no controle dos processos de produção, tais como: temperatura, vazão, umidade, pressão, detecção de gases, entre outros. Na área ambiental são utilizadas redes de sensores distribuídos espacialmente em florestas, mares e rios, formando uma extensa rede sensores. Os nós desta rede de sensores são dispositivos eletromecânicos usados para medir características ambientais tais como temperatura, pressão atmosférica, umidade, luminosidade, entre outros.

Na área de robótica, os dispositivos de sensoriamento são utilizados notadamente para duas atividades: i) mapear as características do ambiente e identificar objetos móveis a sua volta e ii) estimar a localização e orientação do robô no ambiente. Ambas as atividades produzem informações fundamentais para o sistema de navegação do robô, pois além de saber para onde ir, é fundamental que o robô saiba onde está e o que encontrará no caminho.

A reconstrução de ambientes tridimensionais baseada em “observações” feitas pelos sensores de percepção são processados para reproduzir parâmetros estruturais ou entidades dinâmicas do ambiente. Utilizando simuladores e ambientes de visualização tridimensionais é possível acompanhar a criação do modelo digital do ambiente (Figura 4). Os ambientes de simulação facilitam o entendimento e a implementação dos sensores de mapeamento em robôs.

Figura 4 - Reprodução de dados observados por sensores do Turtlebot em simulação.

Fonte: Autoria própria.

Para mapeamento do ambiente, os sensores mais utilizados têm tecnologias baseadas em imagens (câmeras de profundidade), som (sonares) e luz (infravermelho ou laser). As câmeras de profundidade são sensores passivos, isto é, os sinais capturados por este sensor são reflexos de uma fonte de energia externa (e.g. o sol). A utilização de câmeras para mapeamento utiliza o princípio da estereopsia ou visão estéreo, que permite tratar o problema da reconstrução da informação tridimensional de objetos a partir de um par de imagens capturadas simultaneamente, mas com um pequeno deslocamento lateral.

Os sensores de ultrassom (Figura 5.a) e o LIDAR (*Light Detection and Ranging*) (Figura 5.b) são sensores ativos, isto é, processam o sinal emitido por uma fonte de energia interna e refletido por uma superfície refletiva. O princípio de funcionamento destes sensores é o mesmo, isto é, consiste em medir o tempo que um sinal emitido pelo dispositivo leva para atingir o obstáculo e retornar ao sensor. Conhecendo-se a velocidade de propagação do sinal e o tempo de ida e volta é possível calcular a distância que o sinal percorreu, que nestes casos é o dobro da distância do sensor à superfície de reflexão.

Atualmente, tem se tornado comum a combinação de diversos sensores em um único dispositivo. A combinação sinergética dos sensores trabalhando em conjunto aumenta a precisão do sistema. O Kinect Azure da Microsoft, por exemplo, possui uma câmera RGB para reconhecimento facial, uma câmera de profundidade, emissor de infravermelho, uma matriz com sete microfones, sensores inerciais e *software* embarcado que permite executar muitas tarefas, entre elas o mapeamento do ambiente.

Figura 2 - Exemplos de sensores utilizados no mapeamento do ambiente e detecção de obstáculos: a) Sensor de ultrassom, b) Sensor LIDAR e c) Kinect Azure

a) b) c)

Fonte: a) Baú da Eletrônica (2020); b) Pedroso (2020); c) Microsoft (2019).

Embora o conhecimento do ambiente onde o robô está inserido seja fundamental para o planejamento de rotas e execução de comandos de navegação sem colidir com obstáculos, é imprescindível que o robô tenha capacidade de se situar no ambiente, conhecendo a sua localização e orientação. Na navegação autônoma, o robô deve manter o conhecimento de sua

posição e a direção do deslocamento ao longo do tempo. O conhecimento preciso da localização em relação ao ambiente é fundamental para o robô calcular a trajetória que deve seguir para executar uma determinada tarefa. Uma localização imprecisa ou incerta pode ocasionar situações que o robô não desempenhe suas tarefas de forma eficaz, o que pode torná-lo ocioso.

No que se refere ao sistema de posicionamento de robôs, existem dois métodos relevantes que merecem destaque: o posicionamento relativo e o posicionamento absoluto. O posicionamento relativo ou "*dead reckoning*" estima a localização do veículo robótico através de velocidades e acelerações, lineares e angulares. Estas grandezas podem ser obtidas por *encoders* acoplados na roda ou por sensores inerciais embarcados. No caso de *encoders*, por exemplo, tomando como referência um ponto inicial, é possível calcular as distâncias lineares percorridas e os giros do robô a partir da taxa de rotação de cada roda.

A utilização de *encoders* de forma isolada não produz bons resultados, pois estes sensores estão sujeitos a erros acumulativos que rapidamente deterioram a precisão do sistema de posicionamento. Pisos irregulares e escorregadios, por exemplo, causam rotações nas rodas dos veículos que não refletem o deslocamento real do robô. Quando isso acontece com a mesma intensidade e simultaneamente nas duas rodas, o sistema de posicionamento infere que o robô percorreu uma certa distância, mas esta distância é superior a distância verificada em campo. Quando somente uma das rodas derrapa, o sistema de posicionamento infere que ocorreu uma rotação na plataforma, isto é, que o robô mudou a direção do seu deslocamento.

Objetivando contornar os problemas causados pelos *encoders* é usual utilizar sensores inerciais para minimizar os erros de rotação. Os sensores inerciais ou IMU (*Inertial Measurement Unit*) são compostos basicamente por um acelerômetro e um giroscópio. Alguns IMUs possuem magnetômetros, o que proporciona uma maior sensibilidade para coletar dados de orientação. A utilização do IMU em combinação com informações vindas dos *encoders*, melhoram a precisão das grandezas angulares no plano de deslocamento. A combinação de *encoders* e sensores inerciais é um técnica relativamente simples, apresenta bons resultados e tem um custo de implementação baixo, quando comparada a outras técnicas que produzem resultados similares (BORENSTEIN et al., 1997; CHO et al., 2011).

Matematicamente falando, é possível se medir distâncias e velocidades através das medições das acelerações. A integração simples da aceleração na direção do deslocamento nos daria a velocidade linear do robô. Integrando-se mais uma vez, obteríamos o deslocamento. Os dados das acelerações, entretanto, possuem níveis elevados de ruído. Desta forma, a utilização do processo de integração numérica aumenta consideravelmente o erro produzido por estes ruídos. Assim, os dados de deslocamentos obtidos por integração dupla das acelerações

apresentam um nível de precisão muito baixo, o que inviabiliza a utilização desta técnica para mensurar distâncias percorridas.

Como alternativa para melhorar a medição dos deslocamentos lineares experimentados pelo robô, alguns pesquisadores têm incorporado a Odometria Visual (OV) aos sistemas de navegação. A técnica da OV consiste no processo de estimar a movimentação do robô através do processamento de sequências de imagens capturadas por câmeras rigidamente fixada na plataforma robótica. A técnica de OV pode ser dividida em quatro etapas: detecção de aspectos (pontos de controle na imagem), correspondência de aspectos, rastreamento de aspectos e estimativa de movimento. Após a realizações destas etapas, as matrizes de rotação e translação entre dois quadros consecutivos de imagens são obtidas. Estas matrizes expressam a variação dos seis graus de liberdade do robô no espaço 3D. A medição contínua dos ângulos e deslocamentos define a trajetória do robô no ambiente.

Em ambientes propícios, isto é, em ambientes que possibilitem a identificação de aspectos ou pontos de controle bem definidos, a odometria visual produz informações de deslocamentos e rotações bastante precisas. Esta técnica, entretanto, possui um custo financeiro e computacional mais elevado, quando comparada com as técnicas que utilizam *encoders* e IMUs. Assim, torna-se imprescindível um estudo de custo benefício mais detalhado, para embasar a decisão de incorporar ou não a técnica de posicionamento baseada em OV no sistema de navegação do robô.

O sistema de posicionamento relativo calcula a posição e orientação do robô a partir de uma posição inicial, tomada como referência. Com o passar do tempo, os erros dos sensores se acumulam e afastam gradativamente a posição real do robô da posição estimada pelo sistema de posicionamento. Os sistemas de posicionamento absoluto, por outro lado, estimam a posição do robô em relação a um sistema de referência (BORENSTEIN et al., 1997; CHO et al., 2011). Esta característica elimina os erros acumulativos, mas não isentam estes sistemas de erros de outra natureza, principalmente em ambientes fechados. Em termos de cobertura, os sistemas de referência variam de pequenas regiões (portos, aeroportos, chão de fábrica) até áreas que cobrem toda a superfície do globo terrestre. Entre os sensores utilizados para o posicionamento absoluto, destacam-se a bússola eletrônica, os faróis de radiofrequência e os sistemas de posicionamento por satélite.

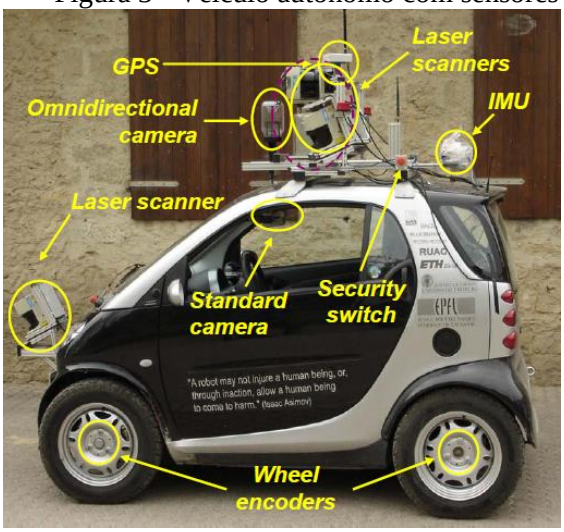
A bússola eletrônica é um dispositivo que mede o ângulo de rotação do dispositivo em relação ao norte magnético. O sistema de referência da bússola é o campo magnético da terra. Este campo é afetado na proximidade de fontes de energia eletromagnéticas (como em linhas de transmissão) e em ambientes internos. A bússola não ajuda em nada na determinação da

localização do robô, mas produz informação relevante no que tange a sua orientação.

Os faróis de radiofrequências são sensores colocados em locais conhecidos que emitem um sinal de rádio. Receptores colocados nos veículos estimam a sua posição através de um processo de triangulação baseado no sinal e na posição dos faróis. Este sistema possui elevada precisão e são bastante utilizados na aproximação de aeronaves nos aeroportos. Outro sistema baseado em triangulação de sinais de rádios são os sistemas de navegação global por satélites (GNSS – *Global Navigation Satellite System*). Os sistemas baseados em satélites possuem cobertura mundial e permitem determinar a posição do dispositivo em qualquer parte da superfície terrestre. Como estes sistemas utilizam sinais provenientes de satélites, a precisão da localização é bastante afetada em ambientes fechados ou grandes obstáculos na vizinhança (e.g. prédios e montanhas). Os sistemas de posicionamento absolutos por radiofrequência e por satélites produzem somente informações sobre a localização e não permitem inferir nada sobre a rotação do dispositivo.

Não se pretende exaurir aqui os diversos sensores, métodos e técnicas para posicionamento do robô. Procurou-se mostrar os sensores mais usuais e suas principais características e limitações. A solução adotada para o componente "*See*" depende do orçamento disponível, das características do ambiente onde o robô irá operar e da tolerância a erros de posicionamento nos sistemas de navegação. É comum, mesmo para robôs de baixo custo, a combinação de diversos sensores e sistemas de posicionamento. Desta forma, os sensores inerciais podem fornecer informações de posicionamento quando o sistema baseado em satélite entra em uma área de sombra, ou que o mesmo IMU e a bússola eletrônica corrijam informações de escorregamentos das rodas dos veículos, que foram registradas de forma equivocada como um giro da plataforma robótica. A Figura 6 ilustra um veículo autônomo com diversos tipos de sensores de posicionamento embarcados. O sistema de navegação do veículo utiliza os dados de todos os sensores para estimar a localização do veículo de forma mais precisa possível.

Figura 3 - Veículo autônomo com sensores embarcados




Fonte: Siegwart et al. (2014).

### 2.1.2 O Componente *Think*: Robótica cognitiva

O componente "*Think*" ou robótica cognitiva é composto essencialmente por *software* e está relacionado com a capacidade do robô de tomar conhecimento do ambiente e decidir qual ação executar para atingir o seu objetivo. A robótica cognitiva consiste na capacidade do robô de "sentir", interpretar, comunicar, interagir e representar o ambiente de forma consistente e eficiente. O robô deve ter a habilidade de imitar o raciocínio humano, ter um comportamento inteligente e aprender através de experiências pretéritas como lidar com ambientes contendo obstáculos estáticos e dinâmicos.

A exploração de ambientes desconhecidos é de extrema importância para realização de diversas tarefas com os robôs móveis (e.g. limpeza de ambientes fechados, operações de resgate, exploração de petróleo). Para que estas tarefas sejam executadas de forma satisfatória são necessárias habilidades cognitivas como a construção automática de mapas do ambiente, habilidade de evitar colisões com obstáculos, planejar e tomar decisões durante a navegação baseada nas informações armazenadas sobre o ambiente (TAI; LIU, 2016).

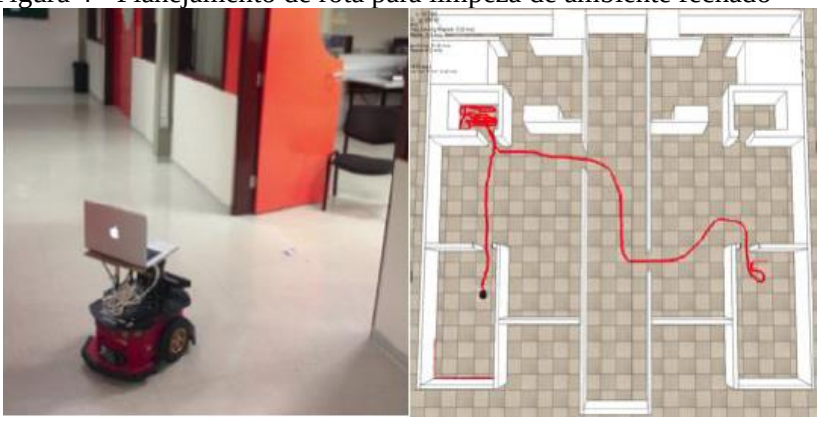
O mapeamento cognitivo nas plataformas robóticas móveis pode ser dividido em duas categorias: topológico e geométrico (VASUDEVAN et al., 2007). O mapeamento topológico registra informações de vizinhança e pertencimento, por exemplo, qual os ambientes vizinhos a uma determinada sala, quais objetos pertencem aquela sala e como estes objetos estão organizados no ambiente. O mapeamento topológico permite estabelecer estratégias de navegação mais abstratas, tais como definir a ordem em que os ambientes serão visitados pelo

robô, qual sala possui o objeto a ser inspecionado pelo robô, entre outras.

O mapeamento geométrico possui informações de distâncias, ou seja, quais as dimensões dos cômodos e dos objetos no seu interior e qual a posição do objeto no mapa. Além dessas informações, podem ser armazenadas informações direcionais, tais como o objeto A está ao norte de B e B está a oeste de C, logo pode-se inferir que C está a sudoeste de A. Informações direcionais podem ser utilizadas para compor mapas abstratos da organização dos objetos na cena.

Os mapas topológicos e geométricos permitem que os algoritmos do componente "*Think*" estabeleçam a melhor estratégia em termos de navegação para que o robô execute a tarefa a ele delegada. A figura 7 apresenta uma abordagem de planejamento de rota utilizando a robótica cognitiva para solucionar as tarefas de limpeza em ambientes fechados (PINHEIRO et al., 2015). A missão do robô é fazer uma limpeza geral em uma pequena despensa localizada na parte superior da planta baixa e limpar uma pequena mancha em um cômodo no canto inferior direito da mesma planta, não necessariamente nesta ordem. O robô se encontra originalmente no cômodo no canto inferior esquerdo. Baseado na organização relativa dos cômodos no ambiente (mapa topológico) e na sua posição inicial (egomoção), o sistema de navegação calcula a menor rota baseada nas dimensões dos cômodos (mapa geométrico) e na missão a ser executada. Esta rota desloca o robô para a despensa, executa uma varredura no cômodo para realizar a limpeza e desloca-se para o cômodo com a pequena mancha no chão (Figura 7). Uma vez estabelecida a rota, entra em ação o componente "*Act*" do modelo STA, que será discutido na próxima subseção.

Figura 4 - Planejamento de rota para limpeza de ambiente fechado



Fonte: Pinheiro et al. (2015).

### 2.1.3 O Componente *Act*: Controle do movimento

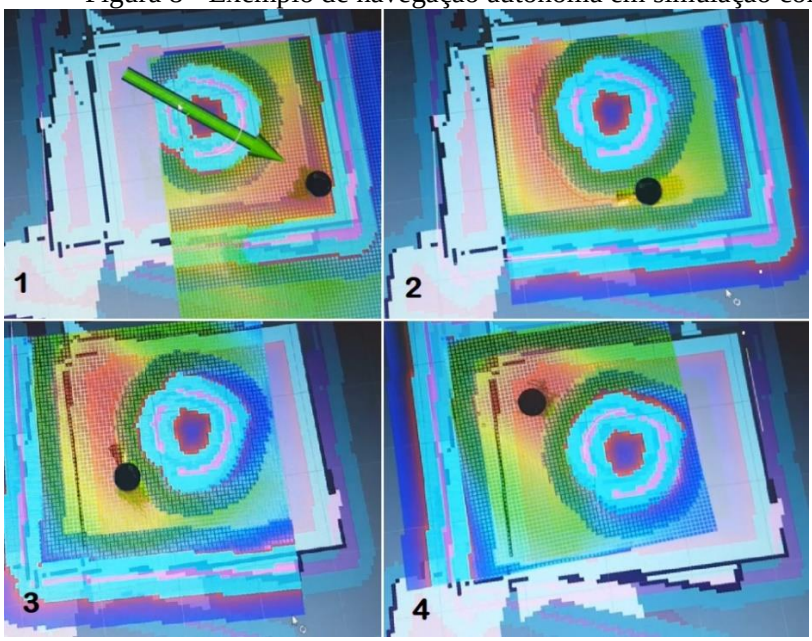
O componente "*Act*", última etapa modelo STA (Figura 3), é responsável por executar os comandos planejados pelo módulo de cognição, fazendo que os atuadores levem o robô até o seu destino. Conforme pôde ser visto até aqui, a tarefa de navegação autônoma possui diversos desafios para ser executada de forma eficiente, pois depende do funcionamento adequado de diversos módulos, tais como o módulo de percepção, os sistemas de localização e mapeamento e os algoritmos do componente de cognição.

Uma vez vencido estes desafios, a última etapa é gerar os comandos de baixo nível para que os atuadores da plataforma produzam o movimento planejado. Esta não é uma tarefa trivial, pois depende de uma variedade de fatores e características do robô, tais como: o meio de deslocamento (aéreo, terrestre ou aquático), a forma de tração (rodas, hélices, membros inferiores, esteiras), entre outros. Para cada tipo de robô existem diversas variações, como o tamanho do robô, o peso, o número de rodas, de membros, de hélices, de articulações, etc.

Mesmo com esta grande variação de tipos de robôs é possível construir módulos de controles genéricos, que atendam, com pequenas modificações, classes de robôs. Assim, é fácil encontrar módulos para veículos com uma, duas e quatro rodas, drones com número variado de hélices e robôs humanoides bípedes ou de animais quadrupedes. Estes modelos de controle, embora genéricos, precisam ser calibrados e ajustados para cada robô, pois dependem de características ainda mais específicas da plataforma robótica, como potência dos motores, consumo das baterias, etc. Está fora do escopo deste trabalho discutir os diversos grupos de módulos de controle existentes.

Todo o modelo STA pode ser observado em programas de simulações, antes de ser implementado em plataformas físicas reais. A Figura 8 demonstra um ambiente simulado em formato retangular, mapeado em tempo real. Nesse teste foi utilizada plataforma robótica *open source*, conhecida como Turtlebot. Os módulos de percepção, localização, cognição e movimento foram controlados por programas gerenciados pelo ROS e a simulação computacional foi executada pelo Gazebo. Nessa simulação, os sensores de percepção detectam um obstáculo circular no centro do quadrado e ao ser enviado o comando contendo a posição do ponto final, destino que o robô deve chegar e a orientação que ele deve parar (seta verde na figura 8.1). O módulo de cognição realiza o planejamento de rota considerando as observações do distanciamento do obstáculo circular através dos sensores de percepção. Ao invés de realizar o menor percurso (colisão com o obstáculo seguindo em linha reta ao destino), a plataforma contorna o obstáculo para concluir a sua tarefa (Figura 8.1, 8.2 e 8.3).

Figura 8 - Exemplo de navegação autônoma em simulação computacional




Fonte: Autoria própria.

Cada componente do modelo STA é um mundo de oportunidade de pesquisa e desenvolvimento. Esta dissertação tem como foco o componente "*See*", ou mais especificamente estudar o impacto da combinação de sensores de localização usuais na precisão dos sistemas de posicionamento. A próxima seção discute trabalhos relacionados que objetivam discutir ou melhorar aspectos da localização do robô no processo da navegação autônoma.

## 2.2 TRABALHOS RELACIONADOS: MELHORIAS NOS SISTEMAS DE NAVEGAÇÃO

Diversas iniciativas tratam a questão da localização de plataformas robóticas móveis utilizando metodologias diferentes, geralmente empregam técnicas de fusão de sensores e métodos de localização probabilística para a estimação da posição e orientação de robôs. Estas técnicas têm como objetivo mitigar as perturbações dos sensores e obter uma localização mais precisa do robô.

O trabalho de Nguyen *et al.* (2019) propõe um sistema de localização melhorado para robôs móveis autônomos utilizando a técnica de fusão de sensores através do EKF. Nesta abordagem são utilizados dados de um sensor de posicionamento relativo, os *encoders* nas rodas do robô, e dois sensores para posicionamento absoluto, o GPS e a bússola. Neste trabalho são comparados os dados de posicionamento real com os dados de posicionamento obtidos por diversas combinações dos sensores, a saber: i) somente o *encoder*; ii) *encoder* + GPS; iii) *encoder* + bússola e iv) *encoder* + GPS + bússola. As combinações ou fusão dos dados dos

sensores são feitas com e sem o emprego do filtro de Kalman estendido (EKF). Os resultados apresentados demonstram que a fusão de sensores utilizando EKF conduzem a uma maior precisão quando comparada a combinação de sensores sem utilizar esta técnica.

A despeito de evidenciar a importância do uso do filtro de Kalman na fusão dos sensores, os resultados apresentados podem ser utilizados também para avaliar o impacto de cada sensor na precisão do sistema de posicionamento. Os resultados indicam que o GPS e a bússola quando associados com o *encoder* produzem uma melhora significativa e bastante similar, com uma ligeira vantagem para a combinação ii (*encoder* + GPS). Como esperado a combinação iii *(encoder* + GPS + bússola) produz o melhor resultado e o ganho na precisão justifica plenamente esta escolha.

É importante ressaltar que esta combinação de sensores é indicada para navegações em ambientes *outdoors*, pois tanto a qualidade dos dados oriundos do GPS quanto da bússola se deteriora drasticamente em ambientes *indoors*, chegando muitas vezes a níveis inaceitáveis para a maioria das aplicações. Em ambientes abertos, as informações de posição dos sistemas robóticos obtidas por sistemas de posicionamento por satélite (GPS) são extremamente relevantes, uma vez que esse sensor fornece dados precisos de posicionamento global e proporciona uma referência confiável para os sistemas de localização. Por outro lado, em ambientes fechados ou com muitos obstáculos, os sistemas de navegação por satélite são extremamente ineficazes, pois nestas situações o sistema torna-se instável e impreciso.

Em Hussein *et al*. (2016) apresenta-se um sistema com fusão de sensores para detecção de obstáculos em ambientes externos em uma aplicação para navegação autônoma *off-road*. A plataforma de teste é um carro de golfe elétrico que foi modificado mecânica e eletricamente para operar no modo sem motorista. Os sensores utilizados no módulo de percepção para localização e posicionamento são uma bússola eletrônica, GPS e os *encoders* nas rodas. Estes sensores são combinados para produzir dados de localização da maneira usual, isto é, sem técnicas sofisticadas para fusão de sensores como o filtro de Kalman estendido. Os sensores de percepção para identificação de obstáculos são uma câmera binocular *stereo-vision* e um laser *rangefinder*. Este trabalho apresenta uma técnica para fusão destes sensores objetivando melhorar a detecção de obstáculos na rota do veículo. Os resultados indicam que a percepção derivada da câmera de visão estéreo aumenta a eficácia do sistema de detecção de obstáculos do sensor de laser *rangefinder*, contribuindo para uma melhoria do sistema de navegação.

Kim and Kim (2013) desenvolveram um sistema de localização para robôs móveis autônomos em ambientes *indoor* utilizando múltiplos sensores ultrassônicos. O algoritmo proposto nesse trabalho utiliza medições de odometria e estimativa de distância através de

observações capturadas pelos sensores. Para estimar a localização do robô, foram fixados três transceptores (Tx) no teto de uma sala com posições conhecidas no sistema de coordenadas global e três receptores (Rx) na parte superior do robô, colocados equidistantes em relação ao centro do robô. O sistema de localização utiliza a fusão de sensores ultrassônicos com os dados da odometria das rodas para estimar a posição e orientação precisa do robô em um laboratório.

Deilamsalehy and Havens (2017) apresentam uma abordagem de estimação de posição e orientação de uma plataforma robótica móvel utilizando IMU, LIDAR e câmera de vídeo. Neste trabalho foi utilizada a técnica de fusão de sensores aplicando o filtro de Kalman com o objetivo de proporcionar um sistema de localização melhorado. Como método de validação foi utilizado a avaliação do erro médio com a utilização de três sensores, combinados da seguinte forma: IMU com a câmera somente, IMU com a câmera e com o LIDAR em 2D (escaneamento a laser no plano xy) e por último o IMU, a câmera e o LIDAR em 3D (escaneamento a laser em múltiplos planos). Neste trabalho, as posições estimadas da câmera e do LIDAR foram utilizadas na fusão dos sensores utilizando EKF como forma de melhorar a estimativa da orientação da unidade inercial. Os estudos resultados demostram que a inclusão do LIDAR no sistema reduz o erro de localização de forma significativa, o que sugere ser uma solução técnica e economicamente viável, considerando o custo de aquisição do LIDAR.

O trabalho de Loevsky and Shimshoni (2010) descreve um sistema de localização preciso e robusto para ambientes *indoors* baseado em marcadores artificiais. Sistemas de localização baseado em marcos geográficos são bastante utilizados para ajustar os sistemas de posicionamento relativos, permitindo que sejam anulados os erros cumulativos destes sistemas. Neste trabalho, entretanto, os marcadores são utilizados para calcular por meio de triangulação de pontos a posição de um veículo guiado automaticamente (AGV). O veículo possui uma câmera de vídeo que identifica os marcadores no ambiente. Os pesquisadores propõem uma metodologia para calcular o posicionamento do veículo que contempla as etapas de estimativas de distância, triangulação e utilização do filtro EKF para uma estimativa precisa da posição do veículo. O sistema ainda é capaz de navegar em ambientes fechados, realizar planejamento de rotas e evitar colisões com obstáculos e locais desconhecidos.

Considerando um grupo de trabalhos que priorizam o uso de imagens para introduzir melhorias no componente de percepção e nos sistemas de navegação, Marín-Plaza *et al.* (2016) apresentam uma abordagem para a detecção de obstáculos estáticos e para realizar a construção de um mapa local de grade de ocupação através de uma câmera de vídeo estéreo utilizando o ROS. Os experimentos foram realizados em um carro de golfe elétrico denominado iCab (Automóvel inteligente de campus), com sensores e atuadores embarcados. Os estudos de

mapeamento de ambientes *outdoor* são de extrema importância para as tarefas de navegação autônoma em veículos não-tripulados.

A abordagem de Alatise and Hancke (2017) busca melhorar o desempenho da navegação e obter uma estimativa melhorada da localização de robôs móveis utilizando o algoritmo EKF e a fusão de dados da odometria visual de uma câmera monocular e dados de aceleração e orientação de um IMU. Em Breitenmoser, Kneip and Siegwart (2011) os pesquisadores analisaram a localização relativa 6D completa em três robôs e-puck utilizando módulos de visão monocular e módulos de rastreamento. Nesse trabalho, foi realizada uma avaliação de precisão das medições de posicionamento em centímetros e graus e comprovaram através de experimentos a aplicabilidade dos módulos de localização relativa em um sistema multi-robôs.

Os estudos sobre o problema de localização e mapeamento simultâneo (SLAM), mostram que apesar de serem objetivos distintos, o mapeamento e a localização estão intrinsecamente entrelaçados. O mapa é indispensável para conhecer a localização do robô, da mesma forma que a estimação da posição do robô é fundamental para garantir a precisão do mapeamento (BAILEY; DURRANT-WHYTE, 2006). A análise dos principais trabalhos relacionados mostra que os sensores inerciais são os mais utilizados para a melhoria dos sistemas de localização, sendo empregados em todos os tipos de ambiente, seja ele aberto ou fechado. Quando os robôs possuem rodas, os dados provenientes dos *encoders* são também sempre utilizados, a despeito da sensibilidade deste sensor ao tipo de pavimentação e à topografia do terreno. Outra técnica bastante utilizada para melhorar a localização do robô é a odometria visual. Na aplicação de mapeamento, é usual a utilização de um sensor de visão computacional ou de sistemas baseados em laser (e.g. LIDAR).

Este trabalho propõe realizar uma análise da contribuição dos principais sensores utilizados na melhoria da estimativa da localização dos robôs, isto é, sensores inerciais, *encoders* e uma câmera RGB-D (cor e profundidade). Neste trabalho optou-se pela utilização de uma câmera Kinect RGB-D, que capta imagem e profundidade do espaço e fornece as características do ambiente para o sistema. Desta forma, os dados de processamento de imagens foram utilizados para a obtenção da odometria visual, ou seja, para estimar o posicionamento e orientação do veículo a partir de um ponto de referência inicial.

A próxima seção aborda alguns aspectos necessários ao entendimento do funcionamento das tecnologias empregadas neste trabalho.

# 3 ROBOT OPERATING SYSTEM

A redução de custos dos dispositivos robóticos (e.g. sensores, atuadores, microcontroladores e engrenagens), o oferecimento de *kits* para construção de robôs pelo próprio usuário e a disponibilização de programas e bibliotecas de *software* para manipulação dos robôs têm popularizado a área de robótica e disseminado os conhecimentos básicos da área nos diversos níveis de formação, incluindo a educação básica. Além disso, pequenas empresas e aficionados podem agora desenvolver soluções robóticas que antes pertenciam ao domínio de atuação de médias e grandes empresas. Se por um lado estas facilidades promoveram a popularização da robótica, por outro o espírito inventivo do ser humano tem produzido uma grande variedade de robôs (Figura 9).

A diversidade de plataformas robóticas introduziu grandes desafios no desenvolvimento de *software* para robôs. Os programas embarcados na plataforma robótica precisam considerar as características físicas de cada plataforma e implementar soluções específicas para uma determinada composição. A diversidade de *hardwares* interfere diretamente no *software* dos robôs, o que dificultava o reaproveitamento dos algoritmos entre plataformas (QUIGLEY et al., 2009).

Figura 9 - Plataformas robóticas


Fonte: Adaptado de Martinez e Fernández (2013).

Diversas inciativas têm buscado minimizar o problema de desenvolvimento de *software* para um espectro tão variado e diverso de robôs. Várias comunidades têm utilizado a Internet para disseminar conhecimento na área, criar fóruns de discussão e troca de informações e compartilhar algoritmos, *frameworks* e *drivers* para diversos domínios e níveis de

complexidade. Nenhuma dessas iniciativas *open source*, entretanto, é tão genérica e estruturada quanto o *Robot Operating System* (ROS), que conta com o suporte de uma equipe numerosa pesquisadores, estudantes, profissionais e aficionados (O'KANE; KANE, 2013). ROS é um *framework open source* utilizado para criação de aplicações robóticas. De acordo com O'Kane and Kane (2013), a descrição oficial do ROS é a seguinte:

> O ROS é um sistema de meta-operação de código aberto para o seu robô. Ele fornece os serviços que você esperaria de um sistema operacional, incluindo abstração de *hardware*, controle de dispositivo de baixo nível, implementação de funcionalidade comumente usada, passagem de mensagens entre processos e gerenciamento de pacotes. Ele também fornece ferramentas e bibliotecas para obter, criar, escrever e executar código em vários computadores.

O ROS fornece os recursos necessários para criar aplicações robóticas e que podem ser reutilizadas por outros robôs. As características do *framework* viabilizam a abstração do *hardware* das plataformas robóticas e fornecem uma coleção de ferramentas e bibliotecas que facilitam o desenvolvimento de sistemas de controle para diversos tipos de robôs (JOSEPH, 2015).

Segundo Joseph (2015), o ROS é uma combinação de transmissão de mensagens (*Plumbing*), ferramentas para depurar e visualizar dados (*Tools*), recursos do robô como localização, mapeamento, navegação e manipulação (*Capabilities*) e o ecossistema (*Ecosystem*), como ilustrado na Figura 10.

Figura 10 - Propriedades do ROS


Fonte: Joseph (2015).

O ecossistema do ROS possui características interessantes que facilitam a compreensão e acessibilidade de diversas funcionalidades no desenvolvimento de aplicações robóticas. Dentre as principais características do ROS, destacam-se: i) a organização interna e gerenciamento dos seus componentes, ii) a interface de transmissão de mensagens, iii) a facilidade de incorporar bibliotecas externas e iv) o suporte nativo ao processamento de recursos distribuídos. A organização interna dos seus componentes em nós e pacotes, facilita a implementação incremental dos sistemas de controle e o gerenciamento dos pacotes, o que torna o desenvolvimento mais sistemático e estruturado. A interface de transmissão de mensagens é imprescindível a comunicação entre processos e permite a troca de dados entre subsistemas

internos, programas e nós. A integração com diversas bibliotecas de terceiros, como OpenCV, PCL, OpenNI, entre outras. Esta facilidade permite utilizar funções complexas, robustas e amplamente testadas em algoritmos mais sofisticados, a exemplo dos algoritmos de processamento digital de imagens. No outro extremo, a utilização de código de terceiros facilita o desenvolvimento de programas com controle de baixo nível, realizando o envio de dados e utilizando portas seriais e pinos I/O. A arquitetura do ROS favorece também a implementação de recursos distribuídos, o que permite processar os dados mais rapidamente. Além disso, o ROS fornece independência de linguagem de programação, escalonamento para tarefas complexas e *framework* para fácil realização de testes (JOSEPH, 2015).

## 3.1 ARQUITETURA E CONCEITOS DO ROS

A arquitetura do ROS é dividida em três níveis de conceito:

1. O nível do sistema de arquivos.
2. O nível de grafo computacional.
3. O nível da comunidade.

O nível do sistema de arquivos estabelece a organização interna ROS, que é basicamente uma hierárquica de pacotes, pastas e arquivos. Para o entendimento desse nível deve-se conhecer o papel dos pacotes, os tipos de mensagens e os tipos de serviços presentes no sistema. O nível de grafos computacionais é o nível que ocorre a comunicação entre processos e sistemas, portanto, é preciso conhecer os principais elementos funcionais do ROS: os nós, o mestre, as mensagens e os serviços. O nível de comunidade é onde encontra-se a aprendizagem de utilização do sistema, incluindo explicações de ferramentas e conceitos, compartilhamento do conhecimento, de algoritmos e de pacotes de qualquer desenvolvedor (JOSEPH, 2015; MARTINEZ; FERNÁNDEZ, 2013). As próximas subseções detalham os níveis conceituais do ROS.

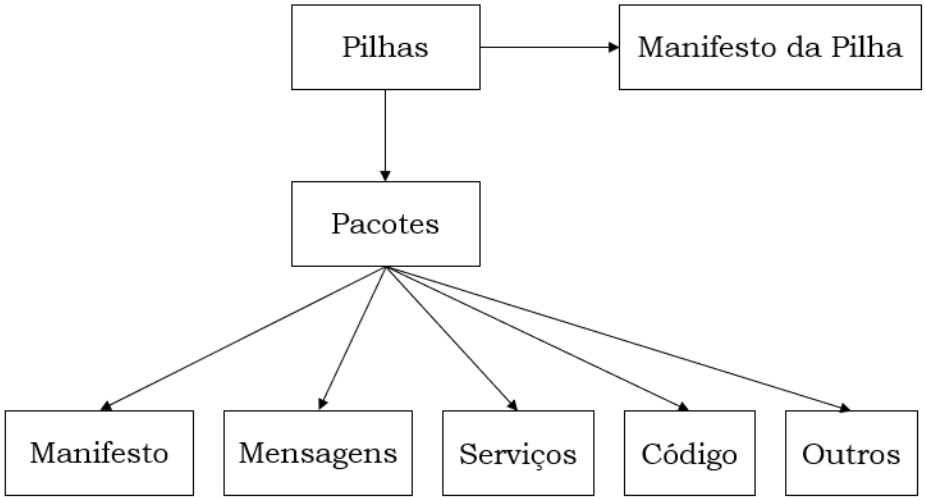
### 3.1.1 Nível do sistema de arquivos

O nível do sistema de arquivos é estruturado hierarquicamente de forma semelhante a maioria dos sistemas operacionais. Os programas do ROS são organizados em pastas que contêm arquivos, os quais descrevem as suas funcionalidades.

Em um nível mais abstrato, o *software* ROS é organizado em pacotes (Figura 11). Um pacote é uma coleção coerente de pastas e arquivos, geralmente incluindo código fonte de nós,

bibliotecas de terceiros e arquivos de configuração. Tais conteúdos são estruturados de forma a: facilitar o processo de reuso dos algoritmos, fornecer de forma transparentes as funcionalidades do pacote e possibilitar o compartilhamento e reutilização dos pacotes.

Figura 11 - Sistema de arquivos e estruturação de pacotes




Fonte: Adaptado de Martinez e Fernández (2013).

Para o desenvolvimento de projetos ROS é fundamental o entendimento de pelo menos três elementos fundamentais: as pastas de manifesto, mensagens (*msg*) e serviços (*srv*). Os arquivos de manifesto do pacote descrevem seus detalhes, fornecendo informações como nome, descrição, licença e dependências. A pasta *msg* contém as mensagens que serão trocadas entre processos. Os tipos de mensagens definem a estrutura dos dados que serão transmitidos. As pastas de serviços (*srv*) estruturam os serviços implementados nos sistemas. De forma análoga aos tipos de mensagens, os tipos de serviços estruturam o conteúdo da solicitação (fornecida pelo cliente ao servidor) e a resposta (enviado pelo servidor de volta ao cliente) (JOSEPH, 2015; MARTINEZ; FERNÁNDEZ, 2013; O'KANE; KANE, 2013).

Por fim, é importante mencionar que todos os pacotes presentes no ROS são organizados em estruturas do tipo pilha (Figura 11). As pilhas são o mecanismo primário do ROS para o *software* de distribuição e seu principal objetivo é simplificar o processo de compartilhamento de códigos.

### 3.1.2 Nível de grafo computacional

O nível de grafo computacional do ROS é a rede de conexão ponto-a-ponto para todos os processos do sistema. Os dados transmitidos no *framework* são compartilhados e acessados

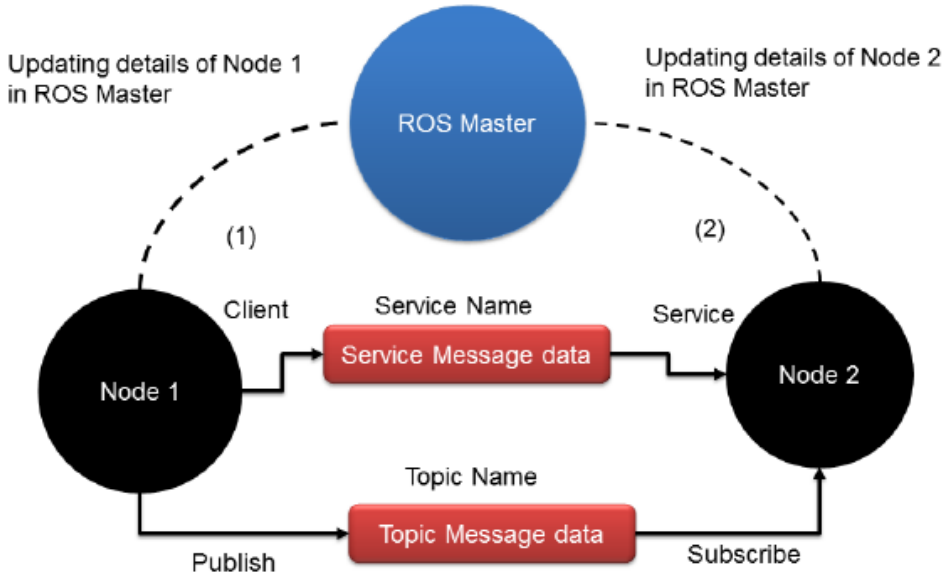
através dessa rede, em um ou múltiplos computadores, a qual gerencia todas as comunicações entre os nós (HERNANDEZ et al., 2015; JOSEPH, 2015; MARTINEZ; FERNÁNDEZ, 2013).

O mestre (ROS Master) é o programa que habilita e gerencia a interconexão dos nós durante a execução do sistema, fornecendo nome, registro e realizando a busca de outros nós vinculados (JOSEPH, 2015; MARTINEZ; FERNÁNDEZ, 2013). Os nós são pequenos programas independentes e executáveis que realizam tarefas simples, e quando articulados de forma coerente podem ser bastante eficientes para resolver tarefas mais complexas.

A comunicação entre os nós é realizada através de tópicos, que são canais de transmissão de mensagens publicadas ou subscritas na rede. O compartilhamento de informações de um nó é realizado através da publicação de uma mensagem subscrita em um ou mais tópicos. As mensagens contêm dados estruturados pelo usuário ou com características padronizadas do sistema. em um ou mais tópicos e essa mensagem pode ser recebida por um ou mais nós a partir dos tópicos subscritos (JOSEPH, 2015; MARTINEZ; FERNÁNDEZ, 2013; O'KANE; KANE, 2013).

Analogamente aos tópicos, os serviços ROS são um sistema cliente/servidor e funcionam de forma semelhante à comunicação dos tópicos na transmissão de solicitação e resposta. O serviço é definido por um nome e um par de mensagens, uma das mensagens é a solicitação e a outra mensagem a resposta. A principal diferença entre eles é que os tópicos utilizam fluxo de dados unidirecionais, enquanto os serviços são síncronos e bidirecionais. O mestre é o programa que garante o elo de comunicação entre os nós, seja por tópico ou serviço. Essa interação pode ser visualizada na Figura 12.

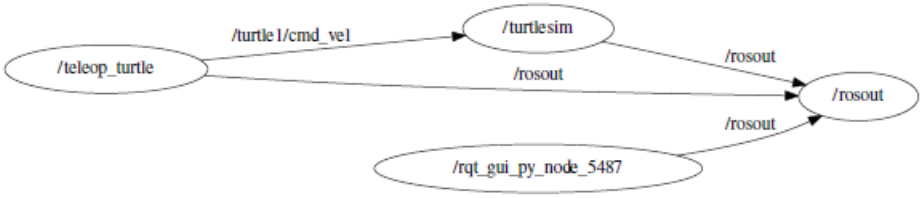
Figura 12 - Comunicação entre os nós ROS




Fonte: Joseph (2015).

Uma representação gráfica do nível de grafos computacionais é observada na Figura 13, na qual é possível identificar os nós (aqueles que estão sendo representados pelas elipses), os tópicos (flechas que representam o relacionamento publicar/subscrever) e a comunicação entre nós (qual nó está publicando e qual nó está subscrito no tópico). Esta visualização do grafo computacional do ROS é fornecida pelo plugin *rqt_graph* da interface gráfica do utilizador (*GUI*). Esta figura representa o grafo computacional da teleoperação do turtlesim, que é uma ferramenta de simulação criada para o ensinamento do ROS e de seus pacotes. O turtlesim, quando executado, exibe uma nova janela no monitor contendo uma pequena tartaruga no centro e é possível controlá-la através das setas direcionais do teclado através do nó "/teleop_turtle". O nó "/rosout" representa o mecanismo de relatório de logs do sistema e o "/rqt_gui_py_node_5487" a interface gráfica do utilizador.

Figura 13 - Representação do grafo computacional do ROS




Fonte: O'Kane e Kane (2013).

Vale ressaltar que também é possível realizar a movimentação do turtlesim criando novos nós programados para publicar mensagens no tópico "/turtle1/cmd_vel".

### 3.1.3 Nível da comunidade

O nível da comunidade ROS é composta pela criação, manutenção de pacotes e compartilhamento de informações relacionadas a pacotes existentes pelos desenvolvedores e pesquisadores que utilizam o sistema (JOSEPH, 2015; MARTINEZ; FERNÁNDEZ, 2013). Neste nível é preciso conhecer as distribuições do ROS, que são as versões do sistema que podem ser instaladas. Desta forma, é necessário analisar as necessidades do projeto antes de escolher a versão desejada, dado que existem diversos projetos *open source* que disponibilizam conteúdo para o ROS uma determinada versão, requerendo que sejam realizadas várias alterações para adaptar um pacote de uma versão antiga para uma versão mais nova.

No nível da comunidade ROS, é possível explorar os repositórios online do sistema, pois é o local onde os pacotes são criados, armazenados, e reutilizados pelos usuários da comunidade. Por último, é extremamente relevante pesquisar o conteúdo documentado e disponibilizado na ROS Wiki (*http://wiki.ros.org*), a qual fornece a transmissão de conhecimentos básicos e avançados para trabalhar com o ROS.

## *3.2 SOFTWARE* PARA SIMULAÇÃO COMPUTACIONAL NO ROS

As simulações computacionais são ferramentas essenciais para o desenvolvimento de robôs. Existem alguns simuladores eficientemente integrados ao ROS (e.g. Gazebo e V-REP). Estes simuladores proporcionam a implementação de diversas tarefas relevantes para o domínio da robótica e para realização de pesquisas, tais como projeção de plataformas robóticas, teste de algoritmos, treinamento de sistemas com inteligência artificial. Além disso, os próprios simuladores podem ser usados no processo de ensino-aprendizagem do ROS, facilitando o entendimento da manipulação do ROS com a observação da atuação dos sensores na interação com ambiente virtual através da visualização dos tópicos.

O Gazebo é um simulador *open source* e gratuito que pode representar robôs complexos navegando em ambientes virtuais *indoor* e *outdoor*, além de ser capaz de criar gráficos e renderizações de alta qualidade (JOSEPH, 2015). Para trabalhar com as simulações no Gazebo é necessário projetar um modelo robótico próprio ou utilizar uma plataforma robótica *open source* disponível no ROS para fins educacionais ou de pesquisa, como é o caso do Turtlebot 2

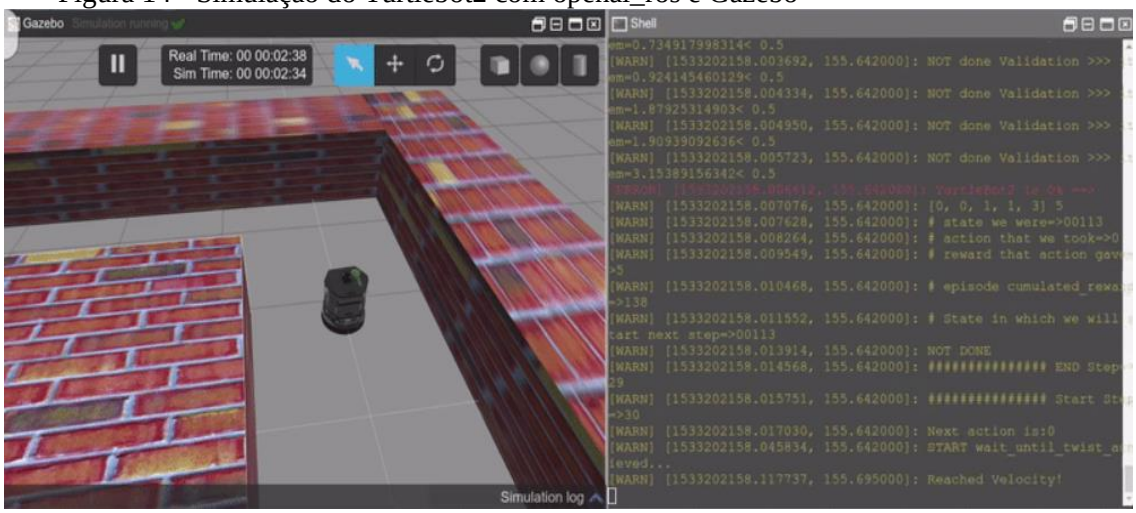
para a distribuição do ROS Kinetic Kame.

O Turtlebot 2 é um robô móvel com tração diferencial, sensor de profundidade (Kinect) e sensores interruptores de para-choque. As simulações do Turtlebot 2 no Gazebo permitem a visualização dos tópicos dos sensores embarcados. Assim, é possível utilizar esses dados da câmera, do comando de movimento e do interruptor de para-choque para desenvolver algoritmos de controle, teleoperação, mapeamento, navegação autônoma, entre diversas outras tarefas. Para execução da teleoperação da plataforma utilizando as mensagens do tópico ROS Twist. Este tópico permite o controle das velocidades linear e angular do robô. Além do *Twist*, existem diversos projetos de código aberto nos repositórios *online* que disponibilizam algoritmos de controle remoto para problemas recorrentes no domínio da robótica.

Um dos desafios relevantes no contexto da robótica móvel é a tarefa de aprendizagem por experiência utilizando a inteligência artificial. Nesse contexto, há um tutorial na ROS Wiki conhecido como "*Turtlebot2 with openai_ros*" que utiliza o Turtlebot 2 para aprender a se mover sem colidir em um labirinto simples simulado no Gazebo. Os ambientes e scripts de treinamento são fornecidos pelos pacotes do openai_ros, entretanto, é informado no tutorial que o script de treinamento é totalmente independente do ambiente, o que torna esse projeto bem flexível em relação a alterações no ambiente ou nos scripts de treinamento, já que são independentes.

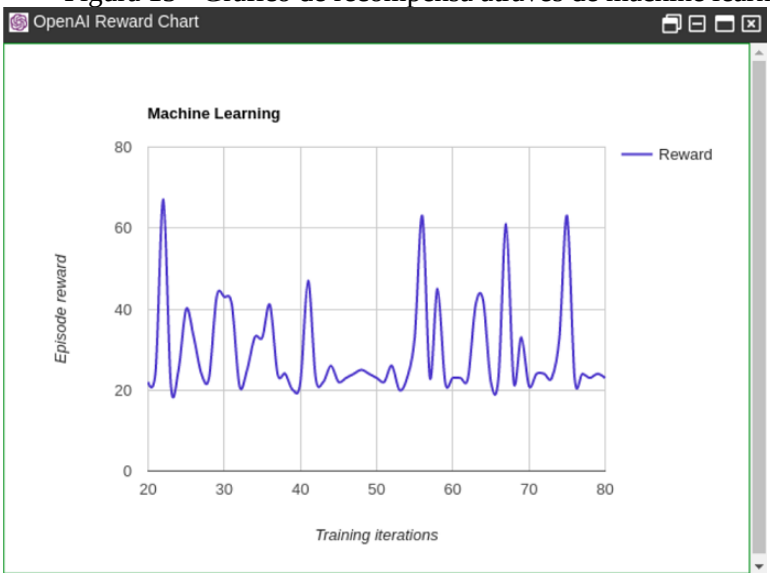
A figura 14 demonstra o Turtlebot 2 executando a tarefa de treinamento no labirinto e os logs do algoritmo de treinamento sendo impressos na janela ao lado. Já a figura 15 representa o aprendizado obtido pelo robô através do algoritmo em cada simulação, em um gráfico que compara a recompensa que equivale ao aprendizado e a iteração (uma simulação em específico).

Figura 14 - Simulação do Turtlebot2 com openai_ros e Gazebo




Fonte: ROS Wiki (2020).

Figura 15 - Gráfico de recompensa através de machine learning do openai_ros




Fonte: ROS Wiki (2020).

## *3.3 SOFTWARE* DE VISUALIZAÇÃO DE DADOS

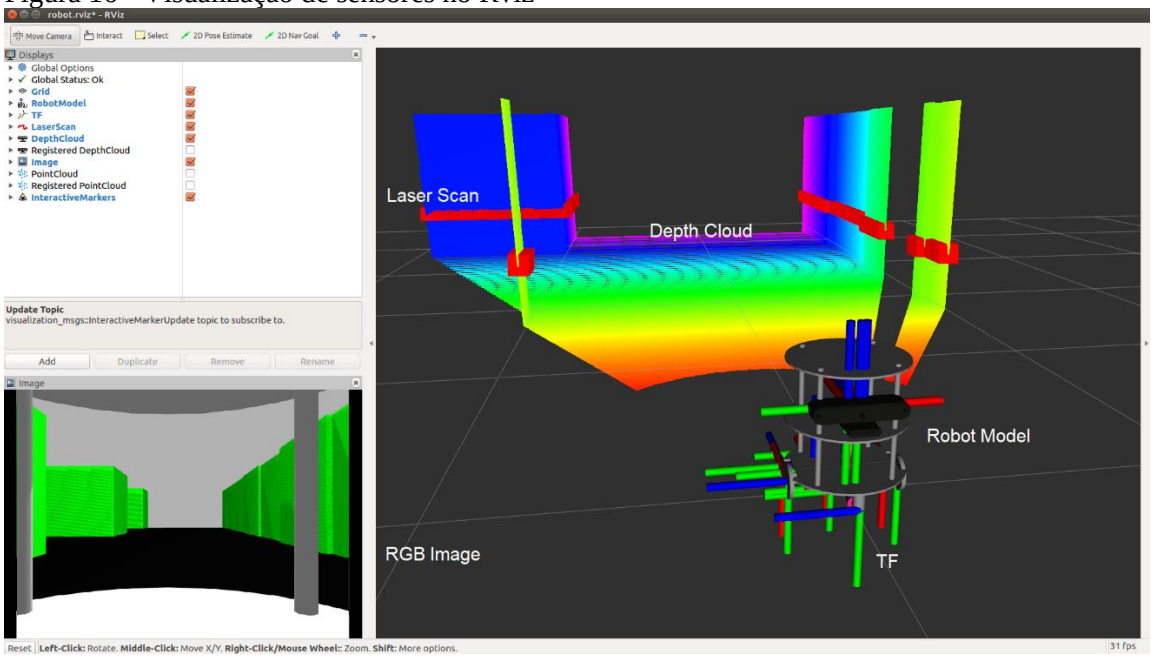
O ROS possui uma ferramenta para visualização 2D/3D de todo o ambiente robótico. O Rviz permite exibir uma grande quantidade de informações, de acordo com as configurações feitas pelos usuários, das informações coletadas pelos sensores e dos dados oriundos da operação do robô, tudo isso de forma simultânea em relação ao tempo de simulação (O'KANE; KANE, 2013).

O Rviz permite a visualização de diversos tipos de informações dos sensores, por

exemplo, algumas possibilidades de exibição são os dados de sensores de visão computacional, como imagens de câmeras RGB, dados de câmera com camada de profundidade (*depth cloud)* ou informações de telêmetro a laser para escaneamento de ambiente (*laser rangefinder*) (JOSEPH, 2015).

Também é possível configurar o modelo da plataforma robótica para visualização na ferramenta e acompanhar vários quadros de coordenadas ao longo do tempo (TF) com os dados obtidos por sensores inerciais embarcados ou *encoders* na roda (JOSEPH, 2015). A Figura 16 representa a utilização do Rviz simulando dados dos sensores mencionados em um ambiente virtual.

Figura 16 - Visualização de sensores no Rviz




Fonte: Joseph (2015).

A figura 16 é um experimento de navegação autônoma realizado em (JOSEPH, 2015). No canto inferior esquerdo pode ser visualizada a "*RGB Image*", uma imagem com diversos cubos verdes, esses objetos foram colocados propositalmente no espaço para simular um ambiente com nove mesas. Essa simulação foi realizada com o intuito que o robô fosse capaz de entregar comida em qualquer uma das mesas.

É importante destacar alguns aspectos dessa ilustração, por exemplo, a parcela colorida da simulação representa o "*Depth Cloud"*, ou seja, uma nuvem de pontos contendo informações de profundidade coletadas pelo Kinect do modelo, cujo gradiente, das cores quentes para cores frias, é equivalente a medições de distâncias de aspectos do ambiente, de mais próximos para distantes, respectivamente. Por outro lado, o termo "*Laser Scan*" representa o escaneamento de

distâncias do ambiente em uma única linha, demonstrado por uma linha vermelha à uma determinada altura na figura 16. Também é relevante conhecer o *"TF",* que de forma resumida são as relações entre as coordenadas e configurações de alguns parâmetros do robô que auxilia na identificação de alguns componentes no *software* de visualização, como a base do robô, sensores e atuadores fixados nessa base (cilindros verdes, azuis e vermelhos).

# 4 SISTEMAS DE LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEO

Considere um robô posicionado em um local completamente desconhecido e que não existe nenhuma informação prévia da sua localização ou das características do ambiente a sua volta. Deseja-se saber se é possível uma plataforma robótica construir um mapa coerente do ambiente e, de forma simultânea, conhecer a sua localização dentro desse mapa. A Localização e Mapeamento Simultâneo (SLAM) busca solucionar problema. Apesar do mapeamento e localização serem problemas distintos, estão intrinsecamente entrelaçados, uma vez que o mapa é indispensável para conhecer a localização do robô, assim como a estimação da posição do robô é necessária para realizar o mapeamento (BAILEY; DURRANT-WHYTE, 2006; DURRANT-WHYTE; BAILEY, 2006).

O SLAM é um tópico antigo na área da robótica, mas com a evolução das técnicas de localização e dos sensores de percepção, continua bastante ativo na pesquisa e desenvolvimento de veículos autônomos e robôs móveis.

Atualmente, os robôs que possuem sistemas SLAM tem a capacidade de construir mapas, e ao mesmo tempo, estimar sua localização no mapa criado. Esses sistemas são imprescindíveis para múltiplas implementações em ambientes diversificados, inclusive em locais que não é possível a utilização de sistemas com GPS, como tarefas em ambientes submarinos ou embaixo da água (rio, piscina, dutos, entre outros), em ambientes fechados (indústria, edifício, casa, entre outros), embaixo da terra (exploração de minas) ou até em outros planetas (como o robô Curiosity Rover da NASA para exploração em Marte).

O método SLAM também é fundamental em ambientes abertos. Nestes casos, as informações de GPS são extremamente relevantes, uma vez que esse sensor fornece dados preciso de posicionamento global em áreas amplas, proporcionando uma referência confiável para os sistemas de localização de plataformas robóticas e em aplicações de navegação autônoma em robôs terrestres e veículos aéreos (tripulados ou não-tripulados).

As duas primeiras seções deste capítulo, tratam dos tipos de mapas e técnicas de mapeamento. A terceira seção discute a questão dos sensores utilizados na localização, mas especificamente dos sensores utilizados neste trabalho.

## 4.1 TIPOS DE MAPEAMENTO DO AMBIENTE

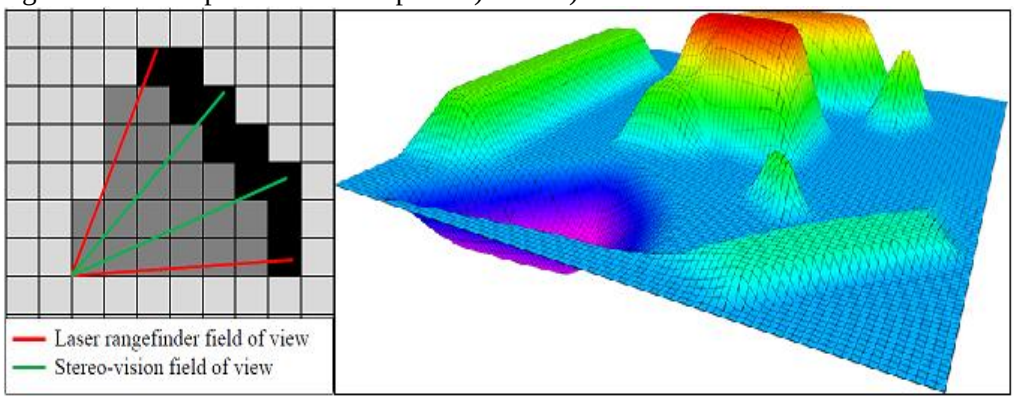
Os mapas originados pelas plataformas robóticas móveis podem ser classificados em mapas simbólicos e mapas sub-simbólicos. Segundo Fernández-Madrigal e Claraco (2013), os

mapas simbólicos incluem objetos do mundo real que podem ser separados pelo nível de complexidade das informações sensoriais coletadas, discrimina-se o objeto pela detecção de muitos dados, em comparação a região que é muito menos informativa ao seu redor. Esse tipo de mapa necessita de um grande esforço computacional para a detecção, rastreamento e armazenamento dos elementos, em troca de uma menor necessidade do espaço de armazenamento. Os mapas simbólicos não serão aprofundados nesse estudo. Por outro lado, o mapa sub-simbólico representa todas as percepções amostradas do ambiente utilizando dados métricos (como distância dos obstáculos, pontos de referência, espaço livre), portanto reduz o esforço computacional da plataforma e, consequentemente, aumenta a necessidade do espaço de armazenamento (FERNÁNDEZ-MADRIGAL; CLARACO, 2013).

O tipo de mapa que deve ser utilizado depende de alguns aspectos específicos relacionados ao robô e ao ambiente. A escolha da representação adequada tem um certo nível de complexidade, uma vez que é preciso levar em consideração as tarefas do robô, a escala do ambiente, as características dos sensores e os recursos computacionais disponíveis. Os mapas mais conhecidos e mencionados na literatura são os seguintes: o mapa de grades de ocupação (*Grid Maps*), o mapa de pontos (*Point Clouds* ou *Point-Based Maps*) e o mapa de características (*Feature Maps* ou *Landmark Maps*) (FERNÁNDEZ-MADRIGAL; CLARACO, 2013).

Os mapas de grade de ocupação fracionam o espaço dimensional (2D ou 3D) em células equivalentes. Cada célula é analisada individualmente e são atribuídos valores que representam a sua ocupação por algum obstáculo, como objetos, paredes e portas, indicando a possibilidade ou não do robô atravessar livremente aquela célula. Desta forma, o *Grid Map* é obtido através do conjunto de células classificadas como ocupada ou livre. É importante ressaltar que esse tipo de mapa é pouco simbólico (sub-simbólico), uma vez que consome um grande espaço de armazenamento devido à alta densidade de informação na interação das células, o que pode impossibilitar o mapeamento de cenários de larga escala. A Figura 17 representa exemplos de mapa de grades de ocupação em 2D e 3D, respectivamente.

Figura 17 - Exemplos de Grid Map em a) 2D e b) 3D



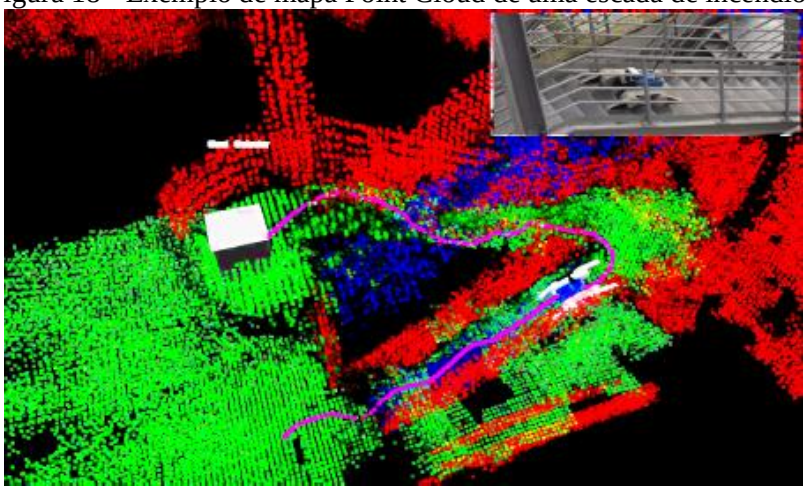

a) b)

Fonte: a) Hussein et al. (2016); b) ROS Wiki (2018).

Outro tipo de mapa sub-simbólico são os mapas de “nuvem de pontos” (*Point Cloud*). *Point Cloud* são representações métricas discretas que não incluem o espaço livre, apenas recebem as informações de distância dos obstáculos através dos sensores originando uma “nuvem de pontos” ao redor dos objetos no ambiente. Esse tipo de mapa é capaz de identificar e representar as bordas dos obstáculos com maior precisão, coletando os aspectos de qualquer objeto no ambiente de forma mais aproximada as características reais, como contornos de geometria circular.

A Figura 18 ilustra os dados no formato *Point Cloud* coletados por sensores embarcados em uma plataforma robótica. O robô processa essa nuvem de pontos, analisa a possibilidade de prosseguir no caminho e realiza o planejamento de rotas a partir das informações 3D no intuito de executar a tarefa de navegação autônoma em terrenos irregulares, como em escadas de incêndio (MENNA et al., 2014).

Figura 18 - Exemplo de mapa Point Cloud de uma escada de incêndio



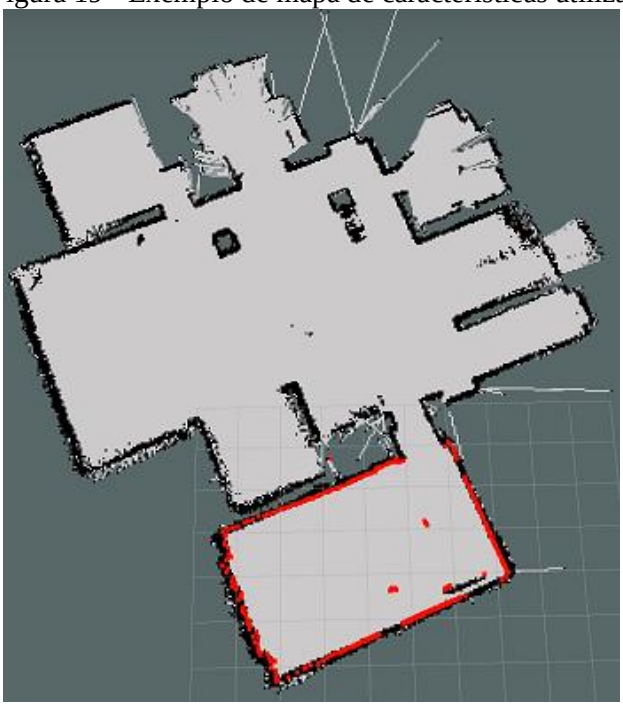
Fonte: Menna et al. (2014).

Os mapas *Point Cloud* podem tornar-se mais simbólico com um pouco mais de processamento computacional, diminuindo, desta forma, o espaço demandado para o seu armazenamento. Vale ressaltar que em ambos os mapas, grade de ocupação ou nuvem de pontos, os dados dos mapas refletem a informação proveniente dos sensores com muito pouco processamento. Contudo, é possível enriquecer os dados amostrados no processo de mapeamento através de algoritmos de extração de características importantes do ambiente, tornando os mapas mais simbólicos.

Em todos ambientes existem elementos físicos que são bastante significativos para os processos relativos à navegação. Estes elementos, denominados de características (*features*) ou marcos (*landmarks*) são essenciais para construção de mapas mais ricos e precisos. As características podem ser identificadas em qual tipo de ambiente (*indoor* ou *outdoor*) e elas são mapeadas não somente como obstáculos (e.g. paredes, portas, pilastras, árvores e veículos), mas também como elementos de referência para a navegação (marcações de pistas, pontes e cruzamentos).

Os mapas de características são construídos através de sensores adequados capazes de distinguir particularidades do ambiente (e.g. lasers de escaneamento 2D ou 3D, câmeras de distância 3D, câmeras de profundidade RGBD, sensores ultrassônicos, entre outros). Utilizando os dados destes sensores e empregando algum algoritmo de detecção é possível realizar extração das características mais relevantes do local (FERNÁNDEZ-MADRIGAL; CLARACO, 2013). A Figura 19 exemplifica um mapa de características de um espaço físico utilizando um sensor de escaneamento a laser 2D conhecido como LIDAR. Neste mapa é possível identificar paredes, móveis e aberturas de portas.

Figura 19 - Exemplo de mapa de características utilizando o LIDAR



Fonte: ROS Wiki (2019).

Ainda que o mapa de características seja definido como sub-simbólico, este mapa é que mais se aproxima a um mapa simbólico. Atualmente, esse tipo de mapa é um dos mais utilizados para a robótica móvel *indoor*. Vale salientar que com alguns ajustes do algoritmo de mapeamento, a exemplo da utilização de processamento cognitivo e abstração de conceitos, é possível tornar o mapa de características em uma representação simbólica do ambiente (FERNÁNDEZ-MADRIGAL; CLARACO, 2013).

## 4.2 MODELOS DE MAPEAMENTO PROBABILÍSTICOS

Segundo Durrant-Whyte e Bailey, “as soluções para o problema probabilístico do SLAM envolvem encontrar uma representação apropriada para o modelo de observação e o modelo de movimento que permita o cálculo eficiente e consistente da distribuição de probabilidade na transição de estados”.

A representação mais comum para os mapas está na forma de espaço de estados com perturbação Gaussiana, que é a solução adotada no Filtro de Kalman Estendido (EKF) para solucionar o problema do SLAM. Outra alternativa é o uso da descrição do modelo de movimento, isso leva a um conjunto de amostras que correspondem a uma distribuição de probabilidade não gaussiana. Esta é a solução utilizada no filtro de partículas ou o algoritmo FastSLAM para solucionar o mesmo problema (DURRANT-WHYTE; BAILEY, 2006).

Uma terceira alternativa para localização de robôs em ambientes dinâmicos é algoritmo de localização de Markov, que busca computar as distribuições de probabilidade de todas posições possíveis no espaço de estados do robô. Esse algoritmo implementa dois modelos probabilísticos distintos para serem atualizados, o de movimento do robô referente a cinemática da plataforma e o modelo de percepção referente às leituras dos sensores embarcados. Existem algumas abordagens fundamentadas no algoritmo de localização de Markov, com pequenas diferenças para mitigar limitações específicas, como a cadeia topológica de Markov, a cadeia de Markov baseada em grades e as cadeias de Markov de Monte Carlo (FOX; BURGARD ; DELLAERT et al., 1999; FOX ; BURGARD; THRUN, 1999).

O algoritmo de localização de Monte Carlo (MCL) é um método de estimação eficiente para determinar a posição de robôs móveis. O MCL é uma versão dos algoritmos de localização de Markov que utiliza métodos de amostragem para aproximar as distribuições de probabilidade arbitrárias e promover a redução do esforço de processamento e memória. Essa técnica é capaz de adaptar o tamanho do conjunto das amostras conforme as circunstâncias a qual está inserido, por exemplo, na localização global quando uma plataforma robótica está completamente perdida utiliza-se uma quantidade muito grande de amostras para preencher totalmente o espaço de estados tridimensional, até alcançar um certo nível de precisão nos dados. No caso da tarefa de rastreamento da posição, com um histórico do ambiente e da localização global, é necessário um menor número de amostras, uma vez que as incertezas são muito menores (FOX e BURGARD e DELLAERT et al., 1999).

### 4.2.1 Filtro de Kalman Estendido

O Filtro de Kalman é um algoritmo de estimativa de estados de um sistema, geralmente alimentado por medições que contém incertezas ou que são coletadas indiretamente. Este algoritmo fornece recursos para lidar com sistemas não lineares de uma forma eficiente. O Filtro de Kalman é um algoritmo que realiza a fusão de dados, processo que envolve a associação, correlação e combinação de dados provenientes de fontes distintas. Tais funcionalidades são implementadas em sistemas robustos que demandam informações altamente coerentes, uma vez que a aplicação desse filtro resulta em dados mais precisos, proporcionam a suavização de dados com perturbação, linearização de estados do sistema, estimativa de parâmetros em sistemas de posicionamento global, entre outras vantagens (DEILAMSALEHY; HAVENS, 2017; FARAGHER, 2012; NGUYEN et al., 2019).

É importante mencionar que existem variantes desse filtro, como o filtro de Kalman

estendido (EKF) e o filtro de Kalman *unscented* (UKF), que são considerados atualmente os mais importantes algoritmos para os sistemas de localização e são frequentemente aplicados em robôs, veículos autônomos e *smartphones* (FARAGHER, 2012).

As variantes EKF e UKF são métodos matemáticos incrementais geralmente utilizados para estimativa de estados e medidas em sistemas não lineares. Por exemplo na robótica móvel, o filtro de Kalman Estendido é amplamente utilizado em diversas aplicações, principalmente na estimativa de posição e orientação de uma plataforma robótica móvel com diversos sensores embarcados e inserido em ambientes tridimensionais (DEILAMSALEHY ; HAVENS, 2017).

No domínio da robótica, o EKF é uma ferramenta muito poderosa e bastante útil. Como já foi mencionado, esse filtro permite a obtenção de informações de posicionamento e orientação de plataformas robóticas altamente precisas através da fusão de dados correlacionados (como medições inerciais, odometria, dados visuais, etc.), o que viabiliza a autonomia do robô nas tarefas de navegação. No ROS, por exemplo, existe um pacote chamado robot_localization que implementa o algoritmo geral do Filtro de Kalman Estendido.

Segundo Moore e Stouch (2014), a implementação do EKF no ROS tem o objetivo de estimar a posição completa em três dimensões (seis graus de liberdade) e a velocidade do robô ao longo do tempo. Este processo pode ser descrito de acordo com as seguintes equações:

$$x_k = f(x_{k-1}) + w_{k-1} \quad (1)$$

Na equação (1), a variável $x_k$ representa o vetor de estados contendo as informações importantes para o sistema no tempo *k*; *f* é uma função de transição de estados não-linear; $w_{k-1}$ é o vetor contendo as perturbações do processo para cada parâmetro do vetor de estado. O vetor de estados em *x* é composto pela posição e orientação do robô em três dimensões e as respectivas velocidades. Essas informações são armazenadas no seguinte modelo:

$$z_k = h(x_k) + v_k \quad (2)$$

Onde $z_k$ é a amostra de medições no tempo *k*; a variável h simboliza a matriz de transformação que mapeia os parâmetros do vetor de estados para o domínio das medições; $v_k$ corresponde ao vetor que contém as perturbações de cada amostra.

$$\hat{x}_k = f(x_{k-1}) \quad (3)$$

$$\hat{P}_k = F P_{k-1} F^{\mathrm{T}} + Q \quad (4)$$

O algoritmo envolve dois estágios, previsão e atualização de medições. As equações (3) e (4) representam a etapa inicial que executa um passo de previsão e projeta a estimativa do

estado atual e a covariância de erro no tempo futuro. Para o nó ekf_localization_node contido no pacote robot_localization, *f* em (3) é um modelo cinemático derivado pelas leis da mecânica de Newton; na equação (4), *P* é o erro estimado de covariância, projetado pela matriz Jacobiana de *f* (*F*) e adicionado à matriz de covariância das perturbações do processo *Q*.

Logo, a etapa de correção é dada por (5) através de (7) e as equações de atualização das medições são apresentadas por (6) e (7).

$$K = \hat{P}_k H^{\mathrm{T}} \left( H \hat{P}_k H^{\mathrm{T}} + R \right)^{-1} \quad (5)$$

$$x_k = \hat{x}_k + K(z - H\hat{x}_k) \quad (6)$$

$$P_k = (I - KH)\hat{P}_k (I - KH)^{\mathrm{T}} + KRK^{\mathrm{T}} \quad (7)$$

Nessa proposta, o ganho de Kalman é calculado utilizando a matriz de observações *H*, as medições de covariância *R* e $\hat{P}_k$. O ganho é utilizado para a atualização do vetor de estados e da matriz de covariância (MOORE ; STOUCH, 2014).

A melhor estimativa de localização do robô é obtida através da análise de duas informações essenciais: a predição, que é baseada na última posição e velocidades conhecidas do robô e das medições, que são referentes ao conjunto de informações de localização providas por diferentes sensores embarcados. A fusão desses dados, de estimação e medição, fornece a melhor estimativa de localização da plataforma robótica (FARAGHER, 2012).

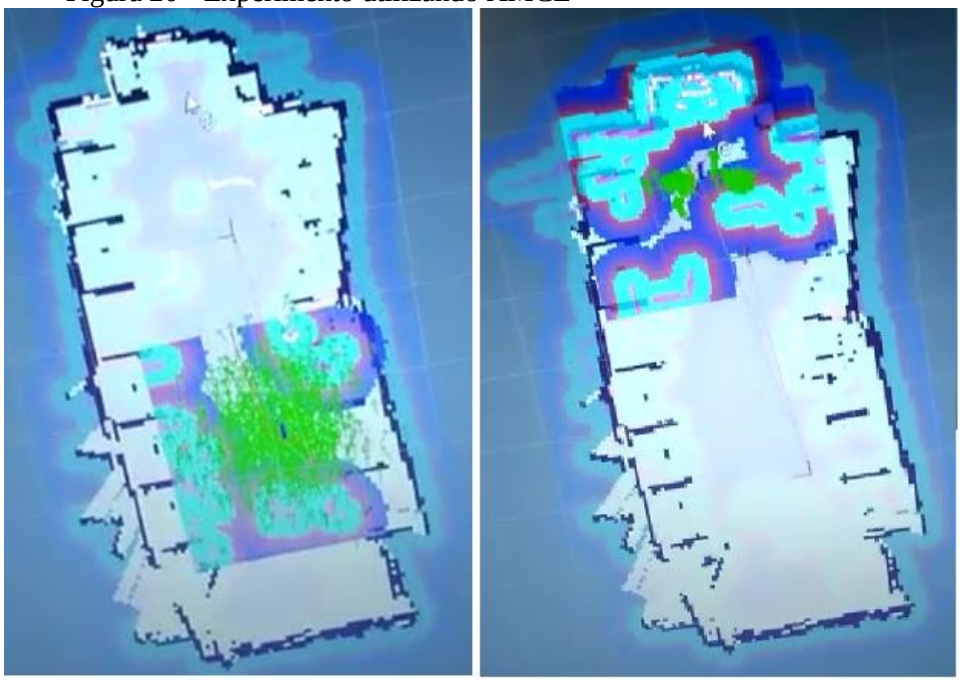
### 4.2.2 Localização de Monte Carlo Adaptativa

A abordagem da Localização de Monte Carlo ou Filtro de partículas utiliza partículas virtuais para estimar a posição do robô no ambiente. O algoritmo de localização de Monte Carlo Adaptativa ajusta dinamicamente esse número de partículas, em determinado período, durante a navegação do robô no mapa. A cada elemento virtual são determinados atributos, como peso, posição e orientação, e representam um palpite da localização do robô. O peso da partícula é o resultado da diferença entre a posição atual do robô e a posição prevista da partícula. Essas partículas são amostradas toda vez que o robô anda e coleta informações do ambiente e podem ser visualizadas na Figura 20, contornando o robô com pequenas setas verdes (DAS, 2018).

Nesse algoritmo, são agregadas as informações de sensores de movimento e visuais em um processo de amostragem que só é interrompida quando o limiar da soma de pesos é excedido. Caso o robô esteja perdido em relação a sua posição, consequentemente o conjunto de amostras é bem maior e as observações contêm perturbações. Por outro lado, caso a

odometria esteja sintonizada corretamente, a soma dos pesos das partículas virtuais são maiores e mais precisos, o que leva a um menor número de amostras (FOX; BURGARD ; DELLAERT et al., 1999).

Figura 20 - Experimento utilizando AMCL



Fonte: Autoria própria.

O experimento da Figura 20 demonstra a tarefa de navegação autônoma em uma plataforma robótica real e simultaneamente permite a visualização da localização do robô e do mapa pelo Rviz. Também é possível observar o algoritmo AMCL (setas verdes) atualizando-se em tempo real enquanto o robô segue para o seu destino. Os diversos algoritmos que capacitam o robô a navegar no mapa consiste em múltiplas informações, analisadas e processadas em tempo real para que o robô seja capaz de chegar ao seu destino, evitando todos os obstáculos do ambiente. As regiões azul claro, roxo e azul escuro no mapa representam os obstáculos presentes no ambiente. Estes obstáculos são “vistos” pelos sensores de percepção e processados nos algoritmos de navegação.

## 4.3 ABORDAGEM MULTISSENSORIAL PARA SISTEMAS DE LOCALIZAÇÃO

A abordagem multissensorial é adotada para obter uma maior precisão nos sistemas de localização do robô. Este trabalho utiliza a fusão de três tipos usuais de sensores em plataformas robóticas de baixo custo: IMU, *encoder* das rodas e câmera Kinect. Em conjunto, estes sensores coletam informações suficientes para obter dados relacionados a localização e posicionamento

da plataforma robótica utilizando duas técnicas de localização, a localização relativa ou "*dead reckoning*" e a localização absoluta.

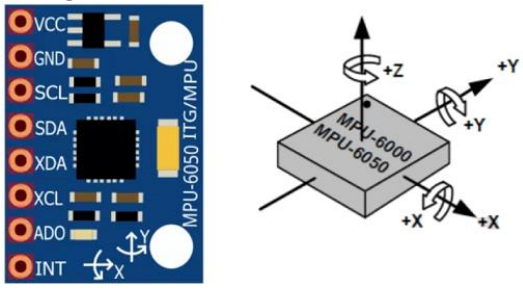
O sensor inercial e os *encoders* acoplados nas rodas contribuem com informações para aprimorar o "*dead reckoning"*, o que significa que estimam a localização do robô tomando como referência um ponto inicial e através dos dados coletados atualizam seu posicionamento relativo. No caso da câmera Kinect, ela pode ser utilizada tanto como sensor de posicionamento absoluto ou quanto no posicionamento relativo. No posicionamento absoluto, a câmera Kinect é utilizada na detecção e descrição de características do ambiente através de imagens com base nas posições absolutas de objetos contidos no ambiente (e.g. paredes, portas e mesas). No posicionamento relativo, a câmera Kinect é utilizada para a realização da odometria visual, isto é, a medição de rotações e deslocamentos da plataforma robótica pela comparação das mudanças ocorridas em duas imagens consecutivas do ambiente. Neste trabalho, em particular, como estamos interessados em medir a precisão do sistema de posicionamento relativo, utilizamos somente os dados da odometria visual da câmera Kinect.

### 4.3.1 Navegação Inercial

A navegação inercial fundamenta-se em informações de taxa angular provenientes do giroscópio e da taxa de velocidade fornecida pelo acelerômetro. A navegação inercial é um dos sistemas mais utilizados nos sistemas de localização porque é independente de plataforma, de baixo custo, não necessita de informações de componentes externos para o posicionamento, fornece medições dinâmicas, rápidas e de baixa latência, isto é, o tempo entre a leitura do dado e a geração de informações de movimentação é muito pequeno (BORENSTEIN et al., 1996).

O sensor inercial, conhecido como IMU, é capaz de realizar medições em tempo real relativas a variantes inerciais de movimento. O IMU é composto por um acelerômetro, um giroscópio e em alguns casos um magnetômetro. O funcionamento do IMU consiste em transformar os fenômenos físicos associados à inércia em sinais elétricos analógicos, que podem ser processados e utilizados por dispositivos computacionais de baixa capacidade de processamento, como os microcontroladores. A fusão sensorial dos dados do MPU6050 (sensor inercial utilizado nesse estudo, ilustrado na Figura 21) resulta em informações de grande valor para obter a localização relativa de um robô, como a orientação, velocidade, deslocamento e velocidade angular.

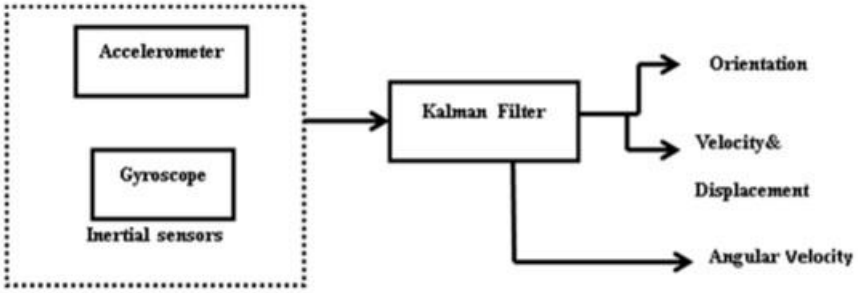
Figura 21 - MPU 6050




Fonte: Souza (2020).

O sistema de referência dos eixos de coordenadas considera o corpo do robô como a origem (centro do IMU) e o plano XY como plano de deslocamento para estimação da posição e orientação do robô em relação ambiente global (Figura 21). Os dados oriundos dos sensores inerciais permitem estimar as velocidades lineares e angulares experimentadas pelo dispositivo, além dos deslocamentos e rotações em torno dos eixos principais, respectivamente (Figura 22).

Figura 22 - Diagrama de blocos do IMU no sistema




Fonte: Alatise e Hancke (2017).

Para a estimativa da localização do robô é usual a utilização somente do valor da aceleração nos três eixos principais e o valor da taxa de rotação em torno dos mesmos eixos (ALATISE; HANCKE, 2017; LIGORIO; SABATINI, 2013). Como os deslocamentos são obtidos por integração dupla das acelerações, os erros provenientes da integração numérica são amplificados, tornando a medição de distâncias bastante imprecisas em sensores de baixo custo. Em submarinos, por exemplo, que podem ficar dias submersos, os sofisticados sensores inerciais fornecem informações precisas de deslocamento e rotações, garantindo uma precisa localização do veículo.

### 4.3.2 Odometria

A odometria das rodas consiste na integração dos dados incrementais de movimento ao longo do tempo e é um dos métodos mais utilizados na navegação para determinar o

posicionamento de veículos robóticos. Esta técnica é bastante utilizada por utilizar sensores de baixo custo, quase nenhum esforço computacional para calcular os deslocamentos e rotações, por fornecer boa precisão e permitir altas frequências de amostragem. Em contrapartida, essa alternativa de localização acarreta o acúmulo de erros, o que implica que quanto maior o tempo de navegação, maiores serão os erros de orientação e consequentemente de posição da plataforma robótica (BORENSTEIN et al., 1996).

A odometria fundamenta-se no conceito de que a rotação das rodas pode ser convertida em deslocamento linear em relação ao solo. Entretanto, existem diversos tipos de erros sistemáticos e não-sistemáticos, que podem interferir nessa concepção, como a derrapagem da roda no solo ou até um desalinhamento entre as rodas, que podem se transformar em dados imprecisos para o sistema de localização (BORENSTEIN et al., 1996).

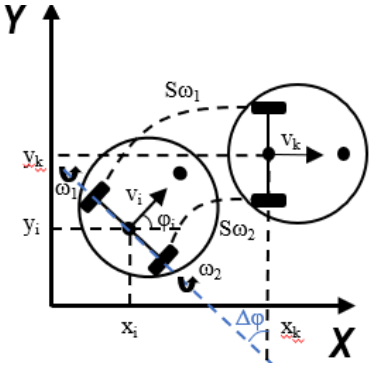
O *encoder* é um dispositivo acoplado à roda do robô com a finalidade de converter amostragens analógicas da taxa angular do rotor em sinais digitais (Figura 23). O sistema de odometria coleta uma quantidade alta de pulsos durante a revolução da roda e, de acordo com a quantidade de pulsos amostrados por cada sensor em unidade de tempo, é possível estimar grandezas físicas como a velocidade, orientação do robô através dos ângulos de rotação e a posição em relação ao ponto inicial do movimento (CHO et al., 2011).

Figura 23 - Motor de corrente contínua e roda com encoder acoplado


Fonte: Uxcell (2018).

As equações do sistema de odometria do robô podem ser implementadas utilizando os dados incrementais amostrados pelos *encoders* nas rodas.

Figura 24 - Modelo do sistema de odometria do robô




Fonte: Autoria própria.

A Figura 24 demonstra uma perspectiva do robô de duas rodas (2WD) com as variáveis relacionadas ao movimento das rodas para obter um modelo cinemático do seu sistema de odometria. Com estes dados o robô pode executar o "*dead-reckoning*" utilizando equações geométricas para calcular a localização em determinado instante em relação a um ponto inicial conhecido. Segundo (Cho et al. (2011) e Nguyen et al. (2019), considerando que Sω1 corresponde a distância percorrida pela roda esquerda, Sω2 a distância percorrida pela roda direita e $D_w$ é a distância entre as rodas, pode-se obter a velocidade, posição e ângulos de rotação pela medição diferencial dos *encoders* através das equações:

$$\Delta S = \frac{S\omega 1 + S\omega 2}{2} \quad (8)$$

$$\Delta \varphi = \frac{S\omega 1 - S\omega 2}{D_w} \quad (9)$$

$$\Delta x = \Delta S * cos\left(\varphi + \frac{\Delta \varphi}{2}\right) \quad (10)$$

$$\Delta y = \Delta S * sen\left(\varphi + \frac{\Delta \varphi}{2}\right) \quad (11)$$

Desta forma, pode-se afirmar que o modelo do sistema de odometria do robô no tempo t é governado por:

$$x_{t+1} = x_t + \Delta x_t \quad (12)$$

$$y_{t+1} = y_t + \Delta y_t \quad (13)$$

$$\varphi_{t+1} = \varphi_t + \Delta \varphi_t \quad (14)$$

### 4.3.3 Odometria Visual

Os sensores utilizados para mapeamento do ambiente (e.g. câmeras, lasers e sensores ultrassônicos) também podem ser utilizados para o posicionamento do robô. Estes sensores coletam uma quantidade tão grande de informações sobre o ambiente que é natural se pensar em algoritmos que permitam inferir mudanças na posição do veículo baseados nestes dados. As câmeras de vídeo, por exemplo, captam continuamente as imagens do ambiente enquanto o robô se movimenta. Algoritmos de processamento digital de imagens identificam pontos característicos na cena que são usados como referência para o rastreamento da localização do robô. O processo de determinação da localização através da comparação entre imagens consecutivas denomina-se odometria visual. A maioria dos algoritmos para realização da odometria visual possui cinco etapas:

1. Captura de duas imagens consecutivas (IT e IT+1)
2. Correção da distorção e retificação das imagens.
3. Identificação de características nas imagens.
4. Correlação das características entre duas imagens consecutivas.
5. Cálculo das rotações e translações

Após a captura de duas imagens consecutivas, as imagens passam por um processo de correção da distorção (*undistortion*). Este processo corrige a distorção causada pela lente da câmera. A correção é única para cada tipo e modelo de câmera e os parâmetros de correção devem ser obtidos através de um processo de calibração. Após a correção da distorção, as imagens passam por um processor de retificação (*rectification*). Neste processo, a imagem é transformada de tal forma que as linhas retas no mundo real e que aparecem curvas na imagem são corrigidas e voltam a ficar retas. A retificação da imagem facilita a busca por características na imagem, que é o próximo passo do algoritmo.

Uma vez retificada a imagem, utiliza-se um algoritmo para a detecção rápida de características na imagem. Uma característica é um ponto notável na imagem que pode ser facilmente identificado através de algoritmos simples de processamento digital de imagens, a exemplo dos algoritmos para detecção de cantos e arestas. O algoritmo para detecção de cantos FAST (*Features from Accelerated Segment Test*) ou SURF (*Speed Up Robust Features*) são os usualmente utilizados nesta etapa da odometria visual. Este algoritmo é utilizado nas duas imagens e identifica uma grande quantidade de pontos candidatos a se tornarem características na imagem. Para poderem ser utilizados no processo de odometria, uma característica

identificada em uma imagem precisa ser correlacionada com a mesma característica na imagem seguinte (*feature matching*).

A massa de pontos com as características que foram encontradas e que possuem correlação em dois quadros consecutivos da imagem são utilizados para determinar a matriz de rotação e o vetor de translação das plataformas robóticas. O vetor de translação representa o deslocamento da plataforma robótica ao longo dos três eixos principais, enquanto a matriz de rotação representa a rotação em torno destes mesmos eixos.

Quando se trata do processo da odometria visual em veículos terrestre, a única rotação que interessa é em torno do eixo perpendicular ao plano de deslocamento e a translação na direção do deslocamento. Estas restrições facilita o processamento da odometria visual e aumenta a precisão dos resultados.

Todas as etapas do processo de odometria visual vale para a utilização de uma ou duas câmeras. Na odometria visual com uma única câmera (monocular) é possível determinar com precisão a matriz de rotação, mas o vetor de translação possui escala indeterminada, o que impede de calcular a distância real percorrida, tendo somente ideia da distância percorrida multiplicada por um fator de escala desconhecido. Na odometria visual com duas câmeras é possível se resolver a ambiguidade da escala e obter as distâncias reais percorridas.

O sensor de imagem utilizado para determinar o posicionamento do robô nesse trabalho é uma câmera Kinect RGBD (Figura 25). Este sensor é utilizado para realizar a processo de odometria visual, recurso que estima a movimentação do robô através da observação de uma sequência de imagens capturadas do ambiente.

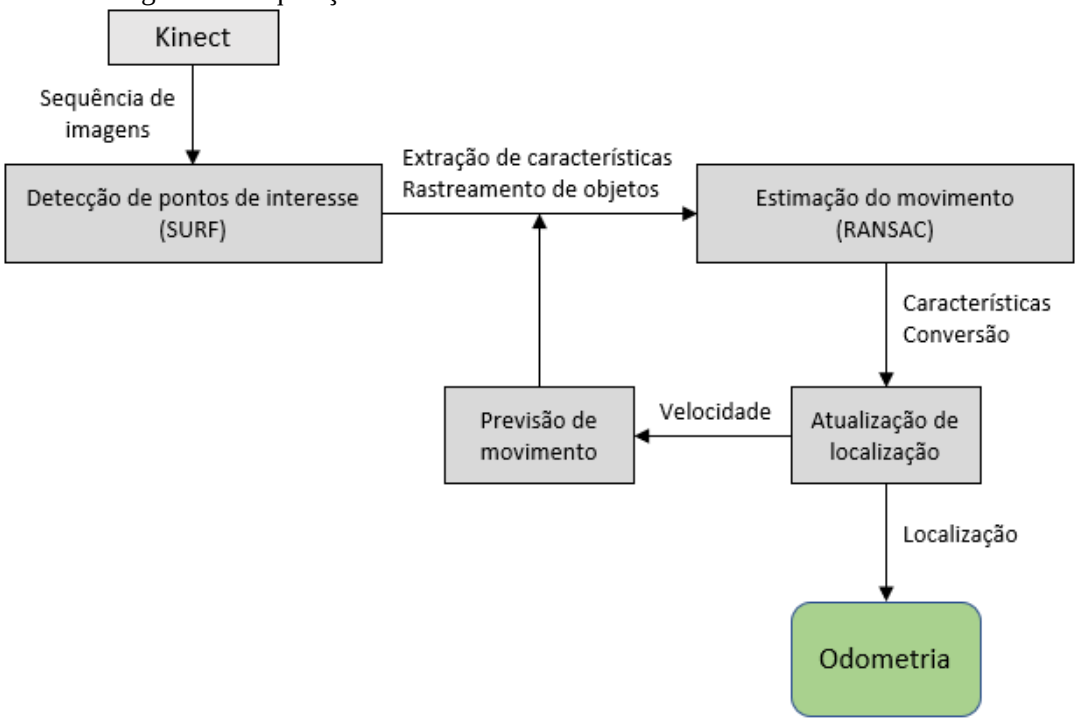
Figura 25 - Câmera Kinect


Fonte: Surur (2018).

Na sua implementação mostrou-se necessário a utilização de alguns algoritmos conhecidos na literatura, como SURF (*Speed Up Robust Features*) que é utilizado na detecção de pontos de interesse, extração de características da imagem e rastreamento dos pontos de interesse nos quadros de imagens e RANSAC (*Random Sample Consensus*) que realiza a filtragem dos dados resultantes do SURF, eliminando os pontos muito distantes na comparação

de imagens. A Figura 26 representa a metodologia implementada para a aquisição de dados de OV (ALATISE ; HANCKE, 2017; FABIAN; CLAYTON, 2014).

Figura 26 - Fluxograma de aquisição da odometria visual




Fonte: Autoria própria.

# 5 CONSTRUÇÃO E PREPARAÇÃO DA PLATAFORMA ROBÓTICA PARA TESTES

Este trabalho tem como objetivo avaliar a contribuição que os sensores de navegação têm na precisão do sistema de localização relativa de um robô terrestre. Os sensores escolhidos para esta avaliação foram os *encoders* das rodas, uma unidade IMU e uma câmera Kinect. A inclusão dos *encoders* neste estudo comparativo se justifica por ser este um sensor ubíquo em veículos terrestres. Já a inclusão do IMU e da câmera a de vídeo foi motivada por serem estes sensores facilmente encontrados nos *smartphones*, o que permite estender a contribuição deste trabalho a plataformas robóticas que utilizam *smartphones* como computadores de bordo, uma solução bastante interessante para o desenvolvimento de robôs de baixo custo.

Os *encoders* das rodas podem ser considerados o sensor universal dos veículos terrestres com rodas. A despeito dos problemas inerentes a utilização da odometria com *encoders*, eles são sempre importantes no processo de localização relativa, notadamente quando associados a outros sensores.

Os sensores inerciais (IMU) é outra classe de sensor que tem sido muito utilizada para auxiliar a odometria de plataformas robóticas. Os sensores IMU são mais abrangentes que os *encoders* de rodas, pois além de poderem ser utilizados em veículos terrestres sem rodas (e.g. robôs humanoides ou quadrúpedes), podem ser igualmente úteis na odometria de veículos aéreos, aquáticos e subaquáticos.

O Kinect representa a classe dos sensores baseados em imagens. No que se refere a odometria visual, os sensores baseados em imagens podem ser classificados em dois grandes grupos: monocular e estéreo. Na odometria monocular é possível determinar com precisão os ângulos de rotação ou giros sofridos pelo robô enquanto se movimenta. É possível se determinar também o número de unidades de deslocamentos percorridas pelo robô. Utilizando uma única câmera, entretanto, não é uma tarefa trivial determinar o fator de escala que deve ser aplicada as unidades de deslocamento para transformá-las em unidades métricas. Desta forma, a odometria visual monocular contribui somente com informações de rotações, pois as distâncias reais percorridas não são determinadas. Na odometria estéreo são necessárias duas câmeras de vídeo posicionadas lado a lado. Além da necessidade de duplicar o *hardware*, tem-se também um aumento considerável na carga de processamento para realização da odometria. Na odometria visual estéreo é necessário comparar duas sequências de imagens, uma de cada câmera, além de ter que correlacionar características que aparecem nas duas câmeras simultaneamente. Este incremento de *hardware* e *software* demandado pela odometria estéreo é recompensado com informações reais de deslocamentos.

De forma a conduzir os experimentos para avaliar a contribuição dos sensores mencionados anteriormente, foi necessária a construção de uma plataforma robótica real e completamente funcional, isto é, capaz de realizar uma navegação autônoma utilizado os *encoders*, IMU e Kinect como sensores de posicionamento relativo. As próximas seções relatam todos os passos para construção da plataforma robótica, deste a seleção do modelo do robô, passando pela construção, configuração e calibração do *hardware* e finalizando com a escolha, instalação e configuração do *software* embarcado.

## *5.1 HARDWARE* DA PLATAFORMA ROBÓTICA

A comunidade ROS é formada por um grupo variado de pessoas, incluindo pesquisadores, estudantes, empresas e aficionados. O conteúdo das pesquisas *open source* da comunidade são disponibilizados pelos seus autores em repositórios *online* sem restrições de acesso. Nestes repositórios é encontrada uma grande variedade de pacotes, bibliotecas, mensagens, nós, projetos de plataformas robóticas, entre outros.

O desenvolvimento da plataforma robótica usada no nosso experimento é baseado no projeto *open source* da comunidade ROS conhecido como Linorobot. O Linorobot foi utilizado como sistema de base para este trabalho pois apresenta a documentação necessária para construir uma plataforma robótica móvel terrestre de baixo custo. O projeto Linorobot disponibiliza plataformas com configurações 2WD (Tração em duas rodas) ou 4WD (Tração em quatro rodas) para rodas comuns ou esteiras. Além disso, o Linorobot atende aos principais requisitos funcionais e não-funcionais deste projeto, que era ser baseado no ROS e possuir suporte para todos os tipos de sensores de localização para a implementação nesse trabalho.

O Linorobot não é um projeto fechado. A documentação do projeto apresenta diversas opções para os sensores, microcontroladores, atuadores e sistema de alimentação, mas pode ser facilmente estendida. Atualmente, existe documentação para utilização dos seguintes itens:

- Sensores a laser para capturar informações do ambiente em sua volta: XV11 Lidar, RPLidar, YDLIDAR X4, Hokuyo (SCIP 2.2 Compliant), Intel RealSense R200 e Kinect;
- IMUs: GY-85, MPU6050, MPU9150, MPU9250;
- Microcontrolador: Teensy (nas versões 3.1/3.2/3.5/3.6);
- Computadores de desenvolvimento baseados em ARM: Raspberry Pi 3 B+, Odroid XU4, Jetson TK1, Jetson TX1, Radxa Rock Pro;

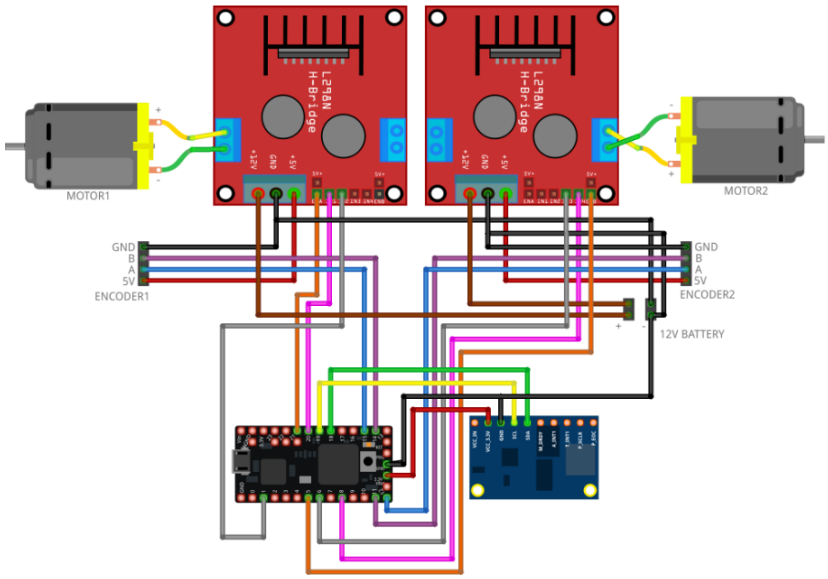
- Drivers para o motor: L298, BTS7960, ESC com Reverse;
- Motores indicados: Ackerman e Mecanum;
- Baterias indicadas: LIPO de três células (3S) ou quatro células (4S).

Vale ressaltar que a documentação do Linorobot fornece um tutorial com todos os passos para construção da plataforma móvel capaz de realizar a tarefa de navegação autônoma, incluindo montagem e configuração de *hardware* e *software*. Todas essas informações estão disponibilizadas no portal digital do Linorobot (https://linorobot.org/).

### 5.1.1 Esquema de Comunicação e Alimentação

A construção da plataforma teve como referência os sensores, *drivers*, câmeras e microcontroladores suportados nesse sistema. O circuito eletrônico foi organizado conforme as conexões do *2WD Wiring* publicado no Github do Linorobot e pode ser visualizado na Figura 27.

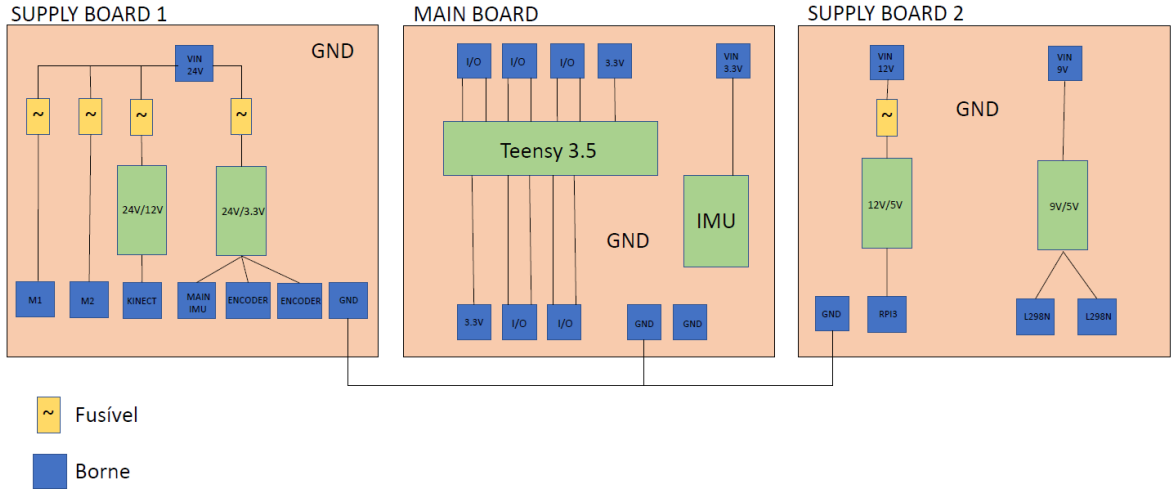
Figura 27 - 2WD Wiring




Fonte: Jimeno (2018).

Tendo em vista o circuito da Figura 27, foram projetadas algumas placas de circuito impresso contendo todos os elementos eletrônicos do 2WD *Wiring*, inserindo novos componentes de proteção e reduzindo o número de cabos entre dispositivos. A prototipagem e o design eletrônico foram desenvolvidos na ferramenta Eagle e a Figura 28 representa o modelo das placas desenvolvidas para o robô.

Figura 28 - Modelo do sistema eletrônico de alimentação e transmissão de dados




Fonte: Autoria própria.

O sistema de alimentação do robô se resume a duas placas de circuito impresso. Na *supply board* 1 são utilizadas duas baterias LIPO de 11.1V de tensão nominal e 2200mah de corrente nominal, conectadas em série para fornecer uma tensão de aproximadamente 22.2V, com o intuito de prover energia aos motores. São utilizados dois reguladores de tensão, o primeiro para reduzir a tensão de 24V para 12V com o objetivo de alimentar a câmera Kinect, e o segundo regulador converte a tensão de 24V para 3.3V como suprimento dos *encoders* e o IMU. A *supply board* 2 possui uma bateria LIPO com as mesmas características elétricas descritas anteriormente, um regulador de tensão de 12V para 5V para alimentar a Raspberry Pi 3 e indiretamente a Teensy 3.5, pois a Teensy está conectada na Raspberry Pi. Já a *main board* conecta todas as portas de saída e entrada (Pinos I/O) no IMU, *encoder* e driver ponte H L298 para realizar a leitura de dados e enviar pulsos indicando comandos de movimento. Vale salientar que o *Ground* é conectado nas três placas para dispor do mesmo potencial de tensão e foi inserido fusíveis de proteção em todos os circuitos para evitar correntes maiores do que previstas em projeto.

### 5.1.2 Construção da plataforma

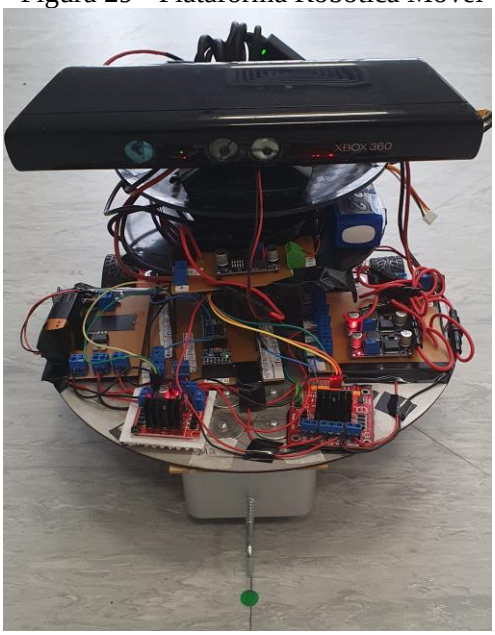
A construção da plataforma teve início com a implantação da placa Raspberry Pi 3 Modelo B+ que possui um processador de 64bits, 1.4GHZ, quatro núcleos e 1GB de SDRAM. Esta placa foi escolhida por ser um bom modelo de minicomputador de baixo custo e atender as nossas necessidades de processamento. Foi adicionado a placa Raspberry Pi um cartão de memória de 64GB classe 10 para aumentar o espaço de armazenamento do dispositivo e manter o poder de processamento, uma vez que Micro SD classe 10 fornece uma velocidade mínima

de transmissão de dados de 10 MB/s e alta velocidade de gravação.

A memória da Raspberry Pi 3 é pequena e limitada, logo, para instalação do Ubuntu MATE 16.04 e a instalação de outros componentes, foi necessário aumentar o espaço de memória virtual no Linux como extensão de memória RAM com a atribuição da seguinte configuração "*CONF_SWAPSIZE=1024".* O próximo passo foi instalar a distribuição, módulos de terceiros e *softwares* adjacentes (e.g. Linorobot, OpenCV, Rtabmap, entre outros). No computador de desenvolvimento instalou-se o ROS Kinetic Kame, Gazebo, Ubuntu Xenial, etc. A versão do ROS Kinetic Kame é recomendada desde maio de 2016 e ficará ativa até abril de 2021.

A plataforma desenvolvida (Figura 29) é um robô com tração em duas rodas com dois motores de corrente continua 24V 350RPM com *encoders* (CHIHAI MOTOR), IMU (MPU6050) e Kinect V1 que contém uma câmera RGB com resolução de 640x480 pixels@30 fps, uma câmera de profundidade 320x240 pixels@30fps para leitura de dados. Os controladores utilizados foram a Teensy 3.5 para coleta de dados em baixo nível e a Raspberry Pi 3 B+ para controle em alto nível utilizando o ROS e o Ubuntu 16.04 e cartão de memória 64GB classe 10. O sistema de alimentação e movimento contém três Baterias LIPO 11.1V, alguns módulos com circuitos para proteção do sistema (fusível, reguladores de tensão LM2596, Ponte H L298). Para começar a trabalhar com o ROS Kinect foi necessário instalar o Ubuntu 16.04 na Raspberry Pi 3 contendo um cartão de memória 64GB classe 10, como foi descrito com mais detalhes anteriormente nessa seção.

Figura 29 - Plataforma Robótica Móvel




Fonte: Autoria própria.

### 5.1.3 Configurações Iniciais

Uma vez montada a plataforma robótica é necessário configurar todos os dispositivos. O primeiro passo é realizar uma configuração de rede para possibilitar a comunicação entre o computador de bordo no robô e o computador de desenvolvimento. Para isso é coletado o endereço IP dos dois dispositivos e configurado parâmetros que definem o *host* e o *master* via terminal em ambos sistemas. O *master* é a plataforma robótica e o *host* o computador de desenvolvimento.

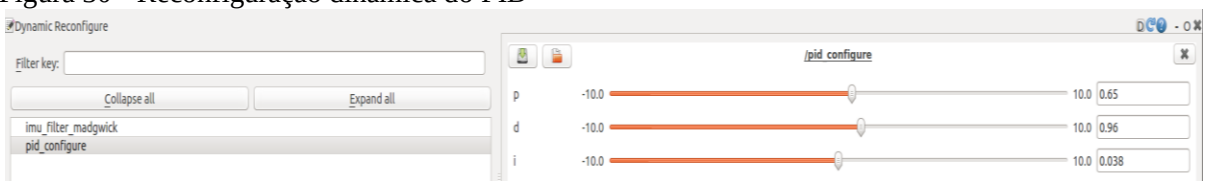
As características do veículo robótico também são importantes para configurar o *software* do Linorobot. Para robôs de dois motores e um rodízio, como é nesse caso, é necessário definir a base como "*DIFFERENTIAL_DRIVE*". Para esta configuração é necessário ter informações precisas de parâmetros como número máximo de rotações por minuto do motor (RPM), contagens do *encoder* por revolução da roda, diâmetro da roda, resolução do microcontrolador, distância entre rodas, entre outros aspectos referentes aos sensores. O IMU deve ser calibrado corretamente, e para isso, todos os circuitos e plataformas do robô devem estar firmes e fixados na base, pois na calibração o robô deve ser colocado em diversas posições, até mesmo de cabeça para baixo.

Com todos os parâmetros de especificações analisados e configurados, deve ser realizada a sintonização do controle PID (Proporcional-Integral-Derivativo). O controlador PID possui valores padrões sugeridos pela plataforma, mas estes valores devem ser ajustados para se tornarem mais adequados para as características de cada robô. Por fim, todas as configurações são enviadas para a teensy e são atualizadas nos arquivos de controle de baixo nível. A próxima subseção detalha os ajustes do controlador PID.

### 5.1.4 Sintonização do Controle PID

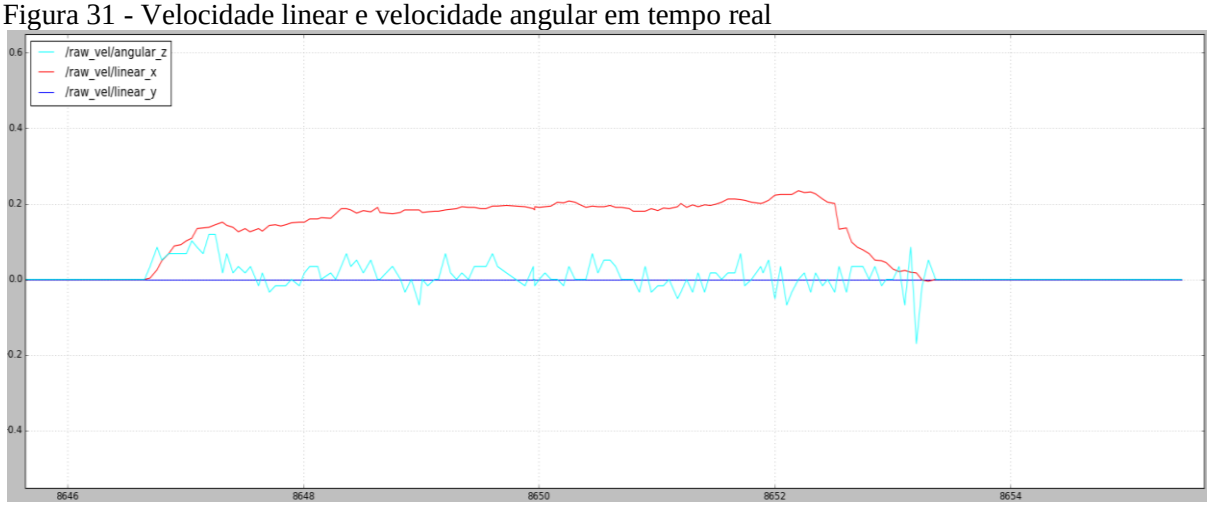
O Linorobot contém as instruções que devem ser executadas para que o usuário seja capaz de sintonizar o controle PID adequado para sua plataforma. Esse repositório fornece os comandos do ROS para teleoperar a plataforma robótica, visualizar as velocidades lineares no eixo x e y, e a velocidade angular no eixo z, e reconfigurar os valores de $K_p$, $K_i$ e $K_d$ em tempo real, como ilustra as Figuras 30 e 31. A calibração do PID é imprescindível para alcançar uma boa navegação do robô.

Figura 30 - Reconfiguração dinâmica do PID




Fonte: Autoria própria.

Figura 31 - Velocidade linear e velocidade angular em tempo real




Fonte: Autoria própria.

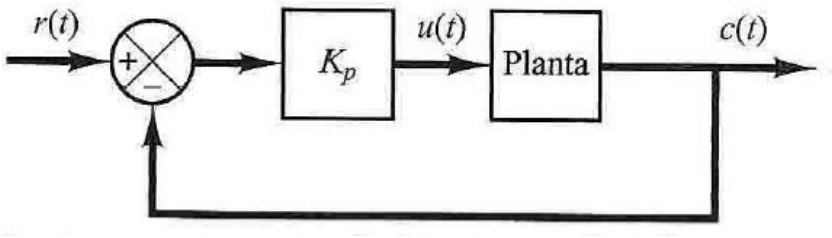
Existem dois métodos de sintonia dos controladores PID básicos com regras propostas por Ziegler e Nichols que são comumente utilizadas em sistemas de automação industrial e na robótica. Estes métodos são muito utilizados pois podem ser aplicadas em plantas com modelos matemáticos conhecidos ou desconhecidos. No primeiro método de Ziegler-Nichols é necessário obter a resposta do sistema através de uma entrada de degrau unitário (diretamente no sistema, ou seja, o controle PID deve ser zerado) (OGATA, 2003). Para a proposta de veículo robótico, o primeiro método não satisfaz as condições, pois quando as variáveis do controle PID são zeradas o robô não responde a comandos de teleoperação.

O segundo método de Ziegler-Nichols define na equação (15), $T_i = \infty$ e $T_d = 0$:

$$G_c(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \quad (15)$$

Isso implica que ao fazer os testes de PID deve-se zerar os valores de $K_i$ e $K_d$. Portanto, utilizando apenas o controle proporcional deve-se aumentar o valor de $K_p$ de 0 até um estado crítico, denominado $K_{cr}$.

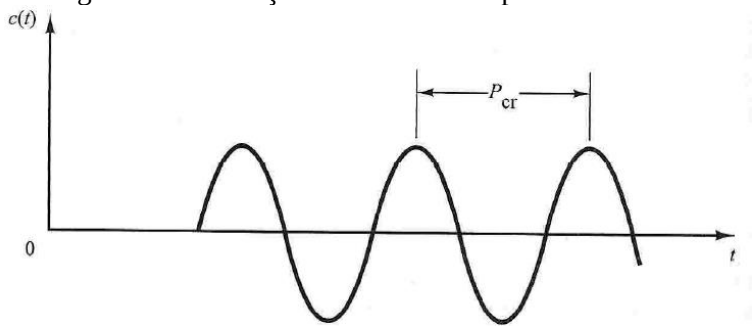
Figura 32 - Sistema de malha fechada com controlador proporcional




Fonte: Ogata (2003).

O valor de $K_{cr}$ será o valor de $K_p$ quando a saída do sistema robótico (velocidades) apresentarem uma oscilação sustentada pela primeira vez, e o período dessa oscilação é conhecida como período crítico $P_{cr}$ (OGATA, 2003). O $P_{cr}$ pode ser coletado pela diferença de tempos entre as cristas das ondas, como demonstra a Figura 33.

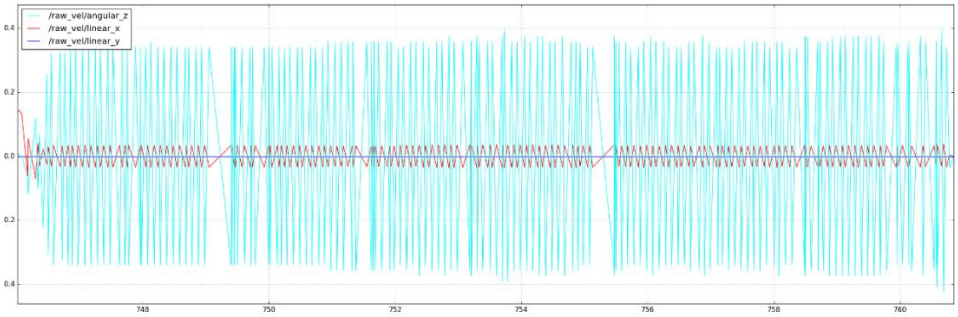
Figura 33 - Oscilação sustentada com período Pcr




Fonte: Ogata (2003).

Na implementação do segundo método das regras de Ziegler-Nichols para sintonia dos controladores PID no robô foi encontrado o valor crítico do controlador proporcional igual a 3,8 e a oscilação sustentada com uma média de período crítico de 128 amostras igual a aproximadamente 0,100, como ilustrado na Figura 34.

Figura 34 - Resposta do sistema robótico com oscilação sustentada




Fonte: Autoria própria.

A análise desses valores só foi viável após coletar e transferir o conjunto dados contendo informações do tempo do sistema e as velocidades linear e angular para uma planilha digital, e isso foi feito executando os seguintes comandos no ROS, antes do início dos testes e após os testes terem sido realizados, respectivamente:

*$ rosbag record raw_vel*

*$ rostopic echo -b nome_do_arquivo.bag -p /nome_do_tópico > nome_desejado.txt*

É importante salientar que o primeiro comando armazena todos os valores presentes do tópico /raw_vel em um arquivo de log do ROS e o segundo comando exporta as informações armazenadas no log para um arquivo no formato ".txt". As regras de sintonia de Ziegler-Nichols baseada no ganho crítico $K_{cr}$ e no período crítico $P_{cr}$ são representadas na Tabela 1:

Tabela 1 – Regras de sintonia de Ziegler-Nichols

| **Tipo de controlador** | **$K_p$** | **$T_i$** | **$T_d$** |
|---|---|---|---|
| **P** | $0{,}5 * K_{cr}$ | $\infty$ | 0 |
| **PI** | $0{,}45 * K_{cr}$ | $\frac{1}{1{,}2} P_{cr}$ | 0 |
| **PID** | $0{,}6 * K_{cr}$ | $0{,}5 * P_{cr}$ | $0{,}125 * P_{cr}$ |

Fonte: Adaptado de Ogata (2003).

Utilizando as regras da Tabela 1, o controlador PID da plataforma robótica foi sintonizado com os seguintes valores:

$$K_p = 0{,}6 * K_{cr} = 0{,}6 * 3{,}8 = 2{,}28$$

$$K_i = \frac{K_p}{T_i} = \frac{2{,}28}{0{,}5 * P_{cr}} = \frac{2{,}28}{0{,}5 * 0{,}100} \cong 45{,}459$$

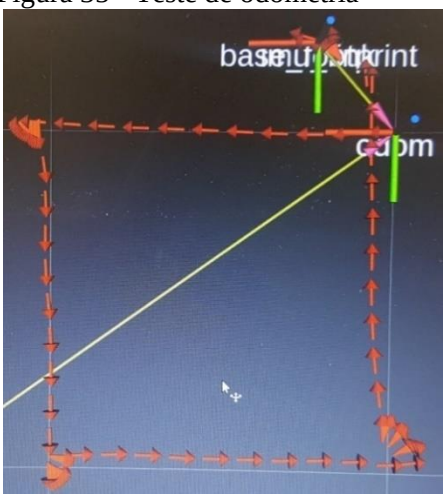
$$K_d = K_p * T_d = 2{,}28 * 0{,}125 * P_{cr} = 2{,}28 * 0{,}125 * 0{,}100 \cong 0{,}0286$$

Os valores encontrados na constante proporcional e derivativa ($K_p$ e $K_d$) foram adotados no controlador PID do robô. Entretanto, realizou-se ajustes na constante integrativa ($K_i$) após os cálculos, pois esse valor não foi adequado ao cenário proposto o que resultou em instabilidade para o sistema. Desta forma, a configuração foi feita manualmente. A metodologia utilizada para determinar a constante integrativa procedeu da seguinte forma: iniciar com um valor bem pequeno para $K_i$ (por exemplo 0,001) e incrementou-se progressivamente este parâmetro até mitigar os erros existentes na teleoperação do robô. Para essa proposta, foi adotado um $K_i$ de 0,2.

### 5.1.5 Configurações de Odometria

A implementação da odometria é um processo que demanda observação, pois para configurar os parâmetros do veículo é necessário executar alguns testes, acompanhar pelo terminal os tópicos de posicionamento (e.g. *position*, *orientation*, *linear*, *angular*) e o caminho percorrido pelo robô no Rviz verificando as grades virtuais. A análise tem como objetivo adequar a escala de sensibilidade de distância e orientação do modelo, em comparação ao mundo real. É importante mencionar que as grades virtuais do Rviz são predefinidas para corresponder a uma distância de um metro, ou seja, são configuradas para representar um metro quadrado, logo tornam-se uma referência para testes e ajustes das informações de odometria.

Figura 35 - Teste de odometria



Fonte: Autoria própria.

A Figura 35 representa um teste de odometria utilizando o Linorobot e a plataforma robótica real, no qual os dados estão sendo expressos no Rviz em um percurso programado de um quadrado com lado de um metro. É possível observar pelas grades virtuais que existem um erro significativo de navegação, uma vez que as setas vermelhas que representam a odometria do robô não coincidem perfeitamente com a referência do *software*.

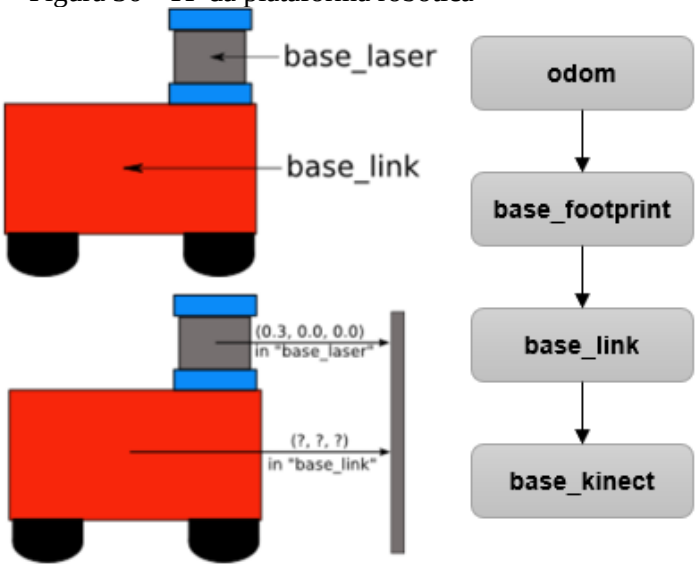
### 5.1.6 Parâmetros de Mapeamento

A câmera Kinect é responsável por coletar dados do ambiente externo através de imagens e profundidade, assim é possível detectar obstáculos e paredes no ambiente. Essa imagem obtida pela câmera é tridimensional, entretanto, foi utilizado um algoritmo no ROS para converter essa imagem 3D em uma visualização 2D, para trabalhar apenas com as

distâncias dos objetos e extremidades, possibilitando uma redução de complexidade do mapeamento do ambiente *indoor*.

O Linorobot oferece uma gama de opções de sensores para realizar a percepção do ambiente, a maioria deles são lasers e câmeras, e devem ser selecionados corretamente nos arquivos de configuração. Nesse projeto realizou-se a instalação da seguinte forma: "*./install 2wd kinect"*, o que identifica que será utilizado algoritmos que consta o método de acionamento diferencial dos motores e o sensor de coleta de dados é o Kinect. Também é importante para o mapeamento determinar as relações entre as coordenadas, conhecido como TF (*Transform Configuration*), uma vez que são esses parâmetros que identifica no *software* de visualização a referência que relaciona a base do robô ao piso e a base do robô à base da câmera, como pode ser visualizado na Figura 36.

Figura 36 - TF da plataforma robótica



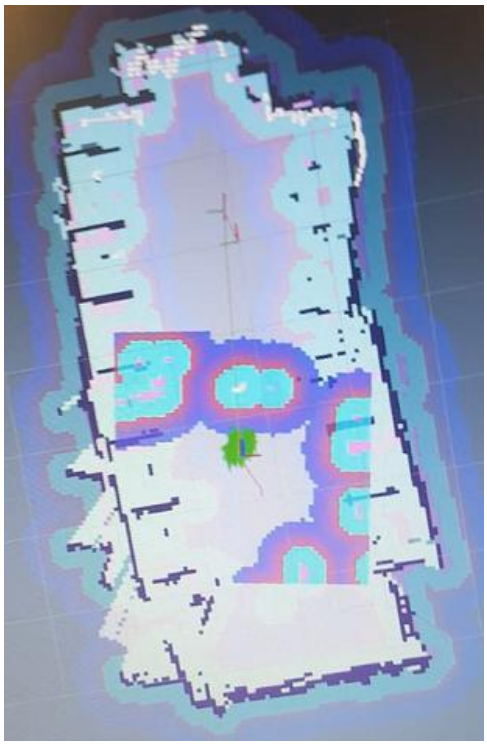

Adaptado de ROS Wiki (2015).

Essa configuração da árvore de transformação é estática e define compensações em termos de rotação e translação entre diferentes quadros de coordenadas. Em outras palavras, esta configuração cria mecanismos para evitar colisões do robô, pois o centro da câmera pode ser diferente do centro da plataforma, logo deve-se compensar essa diferença, ajustando os parâmetros de offset para corrigir as distâncias entre o centro da base do robô e determinado obstáculo.

A tarefa de mapeamento deve ser executada manualmente, portanto, é realizada a teleoperação do robô por todo o local de forma lenta, para que sejam coletadas e atualizadas as características importantes do ambiente, o que reflete em um processo que demanda tempo e observações consistentes. Durante o processo a câmera Kinect armazena os pontos relacionados

a distância entre o robô e os obstáculos estáticos de um ambiente *indoor* (nesse caso, mesas, armários, paredes, cadeiras) e através da navegação no ambiente são armazenados novos pontos, até que se complete o mapa do ambiente. A Figura 37 demonstra o mapa do laboratório de pesquisa no qual foram realizados os experimentos desse projeto.

Figura 37 - Mapa do laboratório de pesquisa construído pela plataforma robótica



Fonte: Autoria própria.

É importante ressaltar que todas as características e detalhes desse mapa foram adquiridas durante um processo de teleoperação e mapeamento, o qual durou cerca de uma hora, e o robô navegou pelo ambiente diversas vezes até que fosse concluído um mapa satisfatório e condizente com o ambiente físico.

## 5.2 SOFTWARE PARA SLAM

As plataformas robóticas baseadas no Linorobot são capazes de realizar a navegação autônoma. Para que esta habilidade seja implementada com um nível de precisão satisfatório, entretanto, é preciso realizar a configuração dos sensores de localização, os ajustes dos parâmetros intrínsecos da plataforma robótica (e.g. tamanho da plataforma, sintonização do controle PID, entre outros) e a construção do mapa do ambiente. Uma vez executadas estas tarefas, faz-se necessário instalar os pacotes e bibliotecas que suportem a navegação autônoma

do robô, isto inclui a fusão dos sensores de percepção para melhorar a estimativa da posição relativa do robô e o mecanismo para mapeamento do ambiente. No centro desta funcionalidade estão o pacote robot_localization e biblioteca RTAB-Map, discutidos nas próximas subseções.

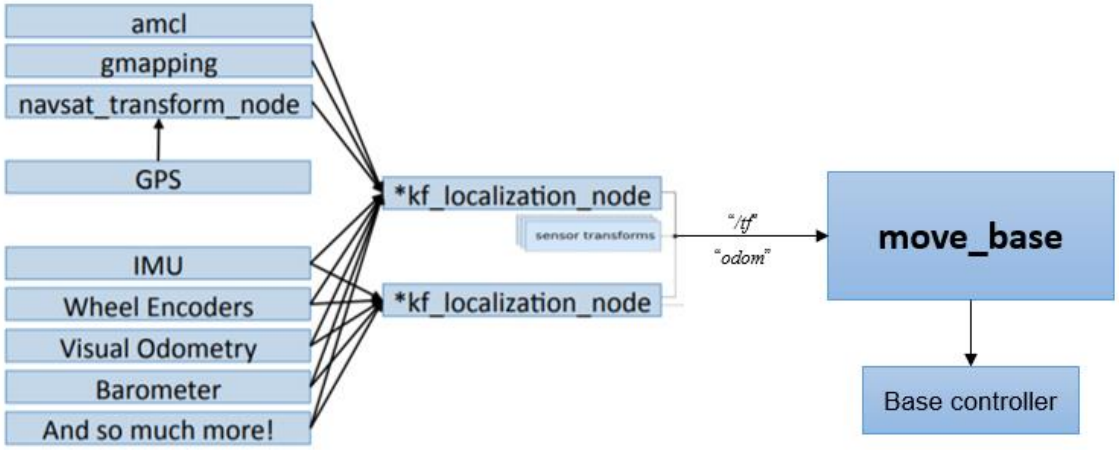
### 5.2.1 Posicionamento Relativo com Fusão Sensorial

O robot_localization é um pacote composto por nós de estimativa de estados não lineares para robôs que navegam em um espaço tridimensional. Este pacote contém a implementação do algoritmo AMCL (*Adaptative Monte Carlo Localization*) e de dois estimadores de estados bastante utilizados na robótica, os filtros EKF e UKF, discutidos no capítulo 4.

AMCL é um algoritmo de localização global que funde dados de diversos sensores de odometria para estimar a posição de um robô com respeito a mapa de referência global. No pacote robot_localization pode é possível utilizar os dados de um número de sensores ilimitado como fonte de entrada dos nós e todos serão aceitos, tais como: GPS, IMU, *encoders*, odometria visual, entre outros (Figura 38).

Como funcionalidade acessória, o pacote robot_localization permite a personalização dos dados, permitindo a inclusão de informações pertinentes à pesquisa e a exclusão de mensagens desnecessárias. É importante destacar que o pacote suporta vários tipos de mensagens do ROS e o seu processo de estimação é contínuo, ou seja, mesmo que o pacote pare de receber as mensagens de entrada, o filtro continuará estimando o estado do robô através do modelo de movimento da plataforma. A Figura 38 ilustra a interação entre os sensores, os estimadores de estados contidos no pacote robot_localization e a pilha de navegação do ROS.

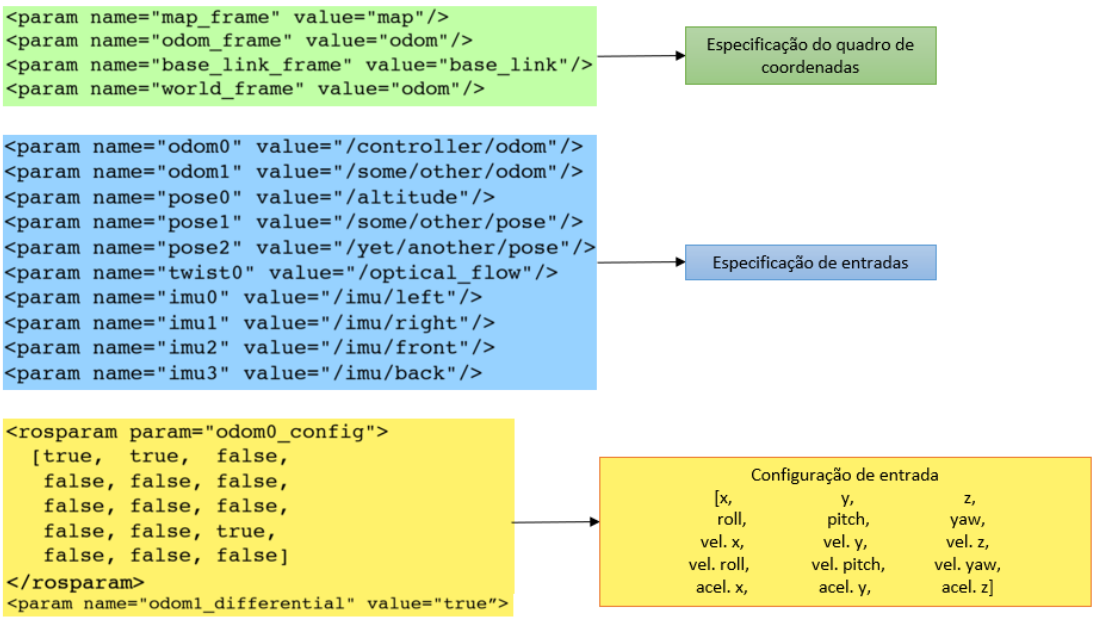
Figura 38 - Visão geral: robot_localization e pilha de navegação do ROS




Fonte: Adaptado de Moore (2015).

O processo de configuração do pacote de localização do robô resume-se a definir os parâmetros do filtro de estimativa de estados (EKF) relacionados ao quadro de coordenadas, a especificação e configuração das variáveis das fontes de entrada, como ilustra a Figura 39.

Figura 39 - Configurações do ekf_localization_node




Fonte: Adaptado de Moore (2015).

Os valores presentes na especificação de entradas (quadro azul da Figura 39) indicam os nomes utilizados na identificação de um quadro de coordenada ou fonte de entrada (*param name*) e e os endereços dos tópicos para onde as mensagens relacionadas a determinado parâmetro serão enviadas (*value*).

As configurações de entrada (quadro azul da Figura 39) são configuradas de forma

booleana (*true* ou *false*) para selecionar quais grandezas físicas serão utilizadas na obtenção de dados de determinado sensor. Sabe-se, por exemplo, que *encoders* coletam dados relevantes de velocidade linear e angular, logo, os valores de velocidade nos eixos x e y (o eixo z não precisa ser considerado para veículos terrestres) e a velocidade angular de giro yaw devem ser consideradas "*true*" e os dados restantes podem ser descartados, no caso "*false*". Os dados da odometria oriundos do sensor IMU descartas todos os dados de velocidade e deslocamentos lineares e só consideram informações de rotações. No caso de veículos terrestres, rotações em torno do eixo perpendicular ao plano de deslocamento (yaw). A odometria visual colabora com informações de deslocamentos e rotações, isto é, deslocamentos no plano xy e rotações em torno de z.

Os dados da odometria visual são obtidos pelo processamento das imagens oriundas das câmeras de vídeo instaladas na plataforma. Nesse caso, o Kinect tem duas funções essenciais para a navegação: produzir dados de odometria visual e a criação do mapa do ambiente.

### 5.2.2 Odometria Visual

O RTAB-Map é uma biblioteca de código aberto que fornece diversas implementações de soluções para aplicações robóticas tridimensionais e bidimensionais para diferentes robôs e sensores. Existem estudos com a utilização do RTAB-Map para análises quantitativas e qualitativas de abordagens utilizando *datasets* bem conhecidos na robótica (e.g. KITTI, EuRoC, entre outros). Estes estudos apontam as vantagens e limitações da computação visual aplicada a navegação autônoma de plataformas robóticas (LABBÉ ; MICHAUD, 2013).

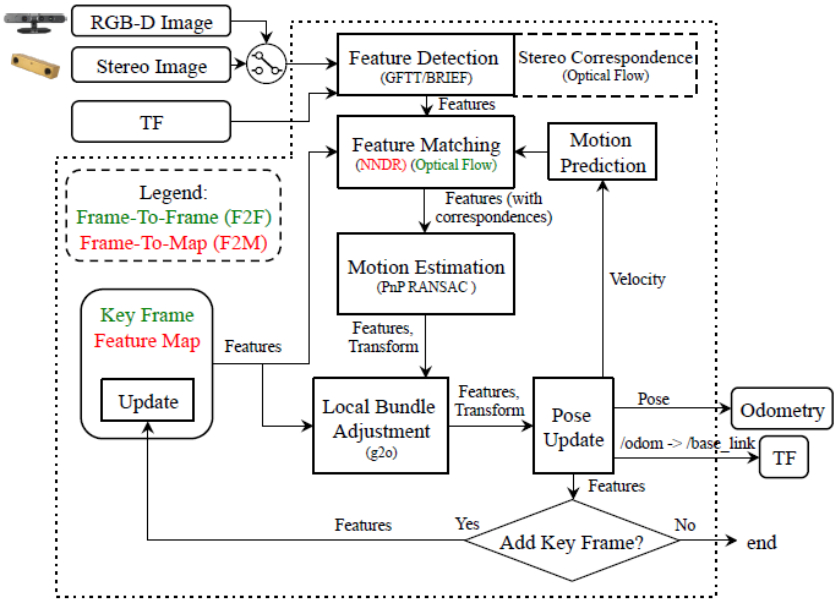
Dentre as vantagens da utilização RTAB-Map no nosso robô destacam-se a possibilidade de realizar o mapeamento e localização visual (SLAM). Iremos discutir nesta subseção somente os detalhes de implementação da odometria visual, responsável pelo posicionamento. O mapeamento visual, embora imprescindível para a navegação autônoma, não tem impacto nos resultados do experimento que avalia a contribuição dos sensores de localização.

A odometria visual do RTAB-Map permite o uso de câmeras RGB-D e câmeras estéreo como entrada do processo. A figura 40 mostra o diagrama de blocos dos nós rgbd_odometry (câmeras RGB-D) e stereo_odometry (câmera estéreo). Conforme pode ser visto no diagrama, os dois tipos de câmeras podem ser utilizados simultaneamente, o que melhora a precisão do sistema. Nesse trabalho adotamos somente a câmera RGB-D Kinect, portanto, o componente *Stereo Correspondence (optical flow)* não se aplica.

O processo para o posicionamento visual é cíclico, alimentado continuamente por imagens oriundas das câmeras. O primeiro estágio após o recebimento da câmera é o processo de detecção de características (*Feature Detection*).

O processo de detecção de características é seguido pelo módulo de correspondência de características (*Feature Matching*). Este processo busca identificar a correspondência das características encontradas em um quadro de imagem no instante $T_R$ com as características de um quadro de imagem no instante $T_S$, onde $T_R - T_S$ é o tempo transcorrido entre dois quadros sucessivos de imagens enviados pela câmera. Se o robô estiver parado, as imagens em dois quadros consecutivos serão idênticas e os pares de características correspondentes nas duas imagens estarão na mesma posição. Se o robô se movimentar, haverá uma mudança na posição das características correspondentes. Os deslocamentos das características correspondentes são utilizados para estimar o movimento realizado pelo robô (*Motion Estimation*).

Figura 40 - Diagrama de blocos dos nós rgbd_odometry e stereo_odometry




Fonte: Labbé e Michaud (2013).

O módulo de correspondência de características (*Feature Matching*) pode operar com duas abordagens distintas: *Frame-To-Frame* (F2F) ou *Frame-To-Map* (F2M). A técnica F2F compara a correspondência das características com quadros consecutivos (*keyframes*). A técnica F2M compara a correspondência das características com um mapa de características criado informações obtidas de quadros anteriores. O mapa de características guarda a história da movimentação das características considerando um certo número de *keyframes*. A técnica F2M possui maior precisão, entretanto, demanda um maior poder de processamento e tem um

maior atraso no tempo de resposta (LABBÉ; MICHAUD, 2013). Utilizamos na nossa implementação a técnica F2F, motivados unicamente pela capacidade de processamento limitada.

Após a estimação de movimento é realizado alguns ajustes e eliminação de erros grosseiros nos dados (*Local Bundle Adjustment*) e calculada a nova posição e orientação do robô (*Pose Update*) (Figura 40). A nova posição retroalimenta o processo como informação de predição do movimento (*Motion Prediction*) e é enviada para publicação, sendo consumida pelo nó responsável pela odometria do sistema de navegação.

Todos os módulos do nó rgbd_odometry já vem equipado com algoritmos consagrados para a realização de detecção e correspondência de características e estimativa de movimento. Nada impede, entretanto, que sejam utilizadas bibliotecas de terceiros e que os algoritmos para o processamento digital de imagens sejam substituídos por algoritmos mais eficientes e adequados ao contexto da aplicação. A biblioteca RTAB-Map trabalha em conjunto com a biblioteca OpenCV. Isto permite que qualquer algoritmo da biblioteca OpenCV seja facilmente integrado.

A biblioteca OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de código aberto que contém mais de 500 funções e algoritmos de computação visual. O principal foco da OpenCV são aplicações em tempo real, mas já existem soluções utilizando a visão computacional nos setores de inspeção de produtos na indústria, estudos de imagem na medicina, segurança, interface do usuário, calibração de câmera, visão estéreo, robótica, entre outros (BRADSKI; KAEHLER, 2008). A próxima subseção discute os principais algoritmos de computação visual configurados na nossa plataforma para a realização da odometria visual.

### 5.2.3 Algoritmos utilizados na Odometria Visual

O primeiro passo na odometria visual é utilizar algum algoritmo de detecção de características do ambiente no quadro de imagens coletado (*Feature Detection*). Existe na biblioteca OpenCV três algoritmos capazes de executar a detecção de características: *Features from Accelerated Segment Test* (FAST), *Scale-Invariant Feature Transform* (SIFT) e *Speed-Up Robust Features* (SURF). Após analisar as características de cada algoritmo, optamos pela utilização do SURF.

O SURF é um detector e descritor de pontos de interesse para a busca por correspondências entre duas imagens da mesma cena. O SURF é invariante em escala e rotação, possui alto desempenho tanto no quesito velocidade quanto precisão, robustez, repetibilidade e

complexidade reduzida em relação ao SIFT, ainda que apresente propriedades semelhantes (BAY et al., 2006; OYALLON; RABIN, 2015).

O algoritmo SURF é executado em duas etapas consecutivas. A primeira etapa consiste em detectar características do ambiente como cantos, bolhas e junções em T e selecionar esses pontos de interesse presentes em determinada imagem, como demonstra a Figura 41. O método identifica uma região reproduzível para os pontos de interesse e então, através do cálculo de respostas Haar-Wavelet nas direções x e y e em regiões vizinhas ao ponto detectado, as características selecionadas são representadas por um vetor contendo pontos de interesse (BAY et al., 2006). Em seguida, é construída uma região quadrada alinhada com a orientação, extraído o descritor SURF, e por fim os vetores descritores são correspondidos entre imagens diferentes baseado na distância dos vetores. Segundo Karami et al. (2017) os quadrados são usados para aproximação, uma vez que a convolução com quadrado é muito mais rápida se a imagem integral for usada. Logo, o SURF usa um detector baseado na matriz Hessiana para encontrar os pontos de interesse. Na atribuição de orientação é utilizada respostas de *wavelets* nas direções horizontal e vertical, aplicando pesos gaussianos adequados nas respostas obtidas e representadas para obter o descritor de recursos do SURF (KARAMI et al., 2017).

A Figura 41 representa uma aplicação do OpenCV empregando o SURF em uma imagem, é um detector de bolhas nas regiões de pontos de interesse utilizando as ferramentas matemáticas descritas anteriormente.

Figura 41 - Exemplo de aplicação do OpenCV utilizando o SURF

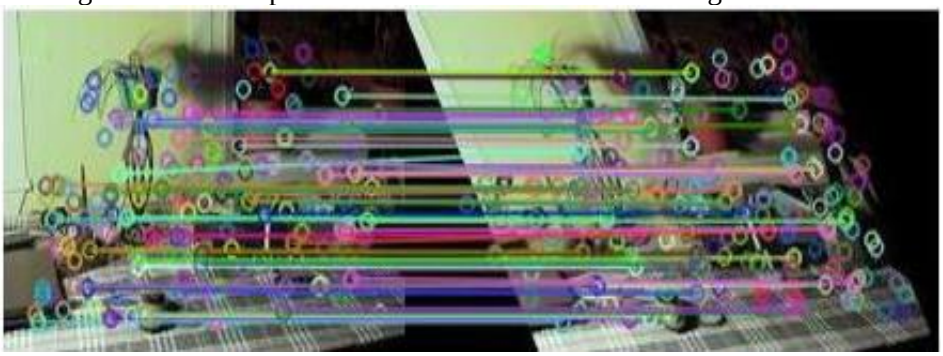
Fonte: Oyallon e Rabin (2015).

Após a detecção dos pontos de interesse é necessário determinar, através das características selecionadas, os locais que são correspondentes entre imagens distintas do mesmo ambiente. É importante ressaltar que o algoritmo de odometria visual fundamenta-se no

acúmulo de movimentos relativos estimados, isto é, de características correspondentes nas imagens adquiridas enquanto o robô está em movimento.

As Figuras 42 e 43 representam as correspondências de pontos de interesse entre duas imagens diferentes da mesma cena. Na Figura 42 a imagem foi cortada e no caso da Figura 43 a imagem foi ampliada, ambas com o objetivo de avaliar o teste de correspondência de características. Estas operações buscam sintetizar mudanças que ocorrem na imagem durante o processo de odometria visual. Quando o robô se movimenta para frente ou para trás a imagem sofre mudanças em termos de escala, isto é, é ampliada ou reduzida (teste de correspondência com imagem ampliada). Quando o robô rotaciona em torno do seu eixo, a imagem sofre um deslocamento lateral, mas ainda algumas características em comum com a imagem anterior (teste de correspondência com imagem cortada).

Figura 42 - Correspondência de características com imagem cortada



Fonte: Karami et al. (2017).

Figura 43 - Correspondência de características com imagem ampliada

Fonte: Karami et al. (2017).

Após a detecção e correspondência das características da imagem, o processo de odometria visual atinge a fase de estimativa da posição, que é realizado pelo RANSAC e G2O.

O *Random Sample Consensus* (RANSAC) é um algoritmo para realizar a estimativa de parâmetros de um modelo através de um conjunto de dados amostrados contendo valores

discrepantes. No caso do conjunto de dados observados em imagens, esse método é empregado para ajustar os atributos no conjunto de correspondências esparsas entre as imagens, ou seja, é uma maneira de lidar com dados discrepantes decorrentes de correspondências incorretas.

De acordo com (NIST, 2003), o paradigma RANSAC é uma solução algébrica robusta que baseia-se em um conjunto mínimo de amostras aleatórias retirado das observações para ajustar e estimar o modelo probabilístico de movimento.

O *General Graph Optimization* (G2O) é um *framework* de código aberto que fornece ferramentas para executar a otimização de problemas probabilísticos não-lineares que podem ser representados como um grafo. Essa ferramenta foi selecionada pela sua popularidade na resolução de problemas probabilísticos, eficiência computacional, bom desempenho e aplicabilidade na robótica (KUMMERLE et al., 2011). A estimativa de parâmetros do modelo de movimento é recebida do RANSAC, transformada e refinada utilizando o ajuste de pacote local (G2O) sobre os recursos do último quadro chave (F2F) para obter a atualização do posicionamento do veículo robótico (LABBÉ; MICHAUD, 2013).

# 5 AVALIAÇÃO E RESULTADOS EXPERIMENTAIS

O objetivo do presente experimento é avaliar a contribuição que os sensores de localização relativo têm na precisão do posicionamento do robô para efeito do sistema de navegação. Os sensores utilizados são os *encoders* das rodas, sensor IMU e câmera Kinect. Estes sensores são responsáveis pela realização de três tipos de posicionamento relativo: odometria tradicional (*encoders*), odometria inercial (IMU) e odometria visual (Kinect). A motivação para a escolha destes sensores é baseada primeiramente na popularidade deles. Esta popularidade vem do custo reduzido e da facilidade de uso, quando comparadas a outras técnicas de posicionamento relativo. Os *encoders*, por exemplo, são sensores baratos e estão presentes em virtualmente todos os veículos terrestres de rodas ou esteiras. Quando associado a um sensor auxiliar para a odometria, o IMU é sempre a primeira opção. Este sensor é um dos sensores mais utilizados em todos os tipos de plataformas robóticas (aéreas, terrestres, aquática e subaquática). A odometria visual vem ganhando espaço com a possibilidade de utilizar tanto a odometria monocular, em plataformas com reduzido poder de processamento, quanto a odometria estéreo, em plataformas com maior capacidade de processamento e que demande uma maior precisão na localização.

A segunda motivação para utilização do IMU e câmera de vídeo neste experimento é o fato dos *smartphones* possuírem estes tipos de sensores. Nosso grupo de pesquisa estuda a possibilidade de utilizar *smartphones* como computador de bordo de plataformas robóticas, explorando não somente a capacidade de processamento destes dispositivos, como a riqueza de sensores que eles possuem. A maioria dos *smartphones* possui pelo menos uma câmera de vídeo, o que possibilita a realização da odometria visual monocular. Alguns modelos mais sofisticados possuem duas, três e até quatro câmeras de vídeo, o que permite realizar a odometria estéreo.

Analisar a contribuição dos *encoders*, sensores inerciais e de câmeras de vídeo na odometria é um importante conhecimento para as plataformas robóticas de baixo custo e para as plataformas robóticas baseadas em *smartphones*, subsidiando as tomadas de decisões que envolvem a análise do custo-benefício de utilizar estes sensores, tanto do ponto de vista financeiro, quanto do ponto de vista computacional.

Como não havia nenhuma plataforma robótica no grupo de pesquisa para ser utilizada neste experimento, foi imprescindível o desenvolvimento do robô. O robô construído para este experimento é voltado para navegação em ambientes internos. Neste tipo de ambiente, os sensores de localização absolutos baseados em satélites não se aplicam. Desta forma, a

navegação do robô é baseada em sensores de localização relativa e no conhecimento do mapa do ambiente (SLAM).

O fato de realizarmos o experimento em um ambiente interno (laboratório de pesquisa), nos permite um maior controle sob alguns aspectos que poderiam interferir na precisão dos sensores. A questão do piso, por exemplo, é fundamental para a precisão dos *encoders* da roda. Pisos escorregadios ou desnivelados causam erros consideráveis na odometria das rodas. No laboratório o piso é uniforme, nivelado e possui uma boa aderência. Assim, os erros dos *encoders* podem ser creditados somente as características intrínsecas do sensor e não causada por alguma perturbação ou ocorrência externa.

O experimento foi idealizado de forma a poder comparar a posição da plataforma robótica realizando um percurso conhecido com a posição da plataforma registrada pelo sistema de navegação do robô. Por simplificação, assumimos que o erro linear e angular da posição e orientação do robô é causada por imprecisão dos sensores de localização relativas ou pela combinação deles. Acreditamos que erros sistemáticos e externos aos sensores são desprezíveis e não comprometem os resultados obtidos.

A próxima seção descreve o percurso utilizado no experimento e como são calculadas as discrepâncias entre os valores reais medidos em campo com os valores coletados pelo sistema de navegação. Nesta seção são apresentadas as medições realizadas com cinco configurações de sensores, a saber: i) *encoder*; ii) *encoder* e Kinect; iii) Kinect e IMU; iv) *encoder* e IMU; e v) *encoder*, IMU e Kinect. Finalmente, a última seção discute os resultados obtidos.

## 5.1 PERCURSO QUADRADO

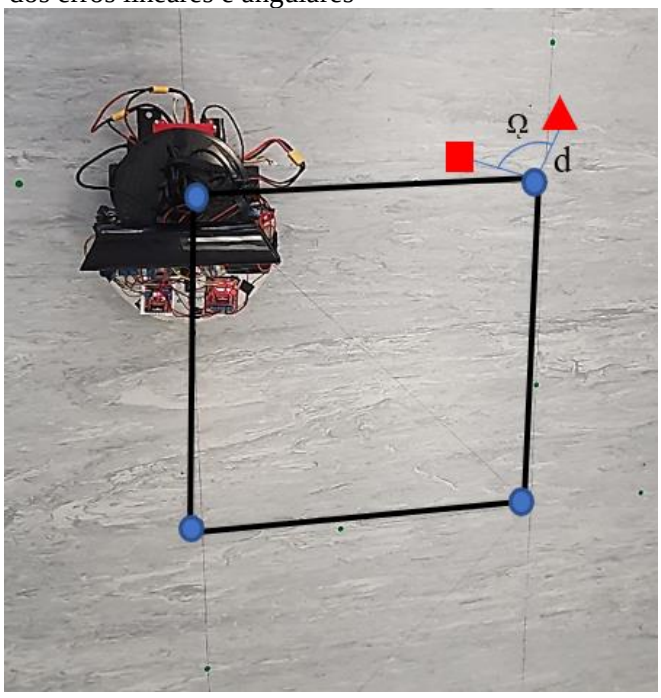
O percurso programado para o experimento é um quadrado de lado medindo 70.7 centímetros, o que resulta em uma diagonal de aproximadamente um metro. O robô inicia o percurso em um dos vértices do quadrado e percorre cada um dos seus lados, realizando giros de 90 graus ao passar em *checkpoints* (vértices do quadrado). Em cada *checkpoint* são medidos os erros lineares e angulares. O erro linear é a distância euclidiana medida do vértice em questão até a posição do robô em campo. O erro angular é diferença entre o ângulo de rotação do robô e o ângulo reto de um quadrado.

A Figura 44 mostra o percurso realizado pelo robô durante o experimento. Os *checkpoints* (círculos azuis) coincidem com os vértices do quadrado. Para facilitar a coleta dos dados de campo, foi inserido na base robótica um parafuso em L utilizado como ponto de referência para as medições em campo (pode ser visualizado na Figura 29). O ponto de

referência deveria coincidir com os vértices do quadrado se não existissem erros na localização. Quando um robô alcança um *checkpoint* são realizadas duas medições. A primeira medição é feita antes da rotação e corresponde a distância euclidiana do ponto de referência ao *checkpoint* (triângulo vermelho na Figura 44). A segunda medição é feita após a rotação do robô. O quadrado vermelho na Figura 44 indica a posição do ponto de referência após a rotação. O ângulo indica o ângulo de rotação medido em campo e o erro angular é a diferença deste ângulo para o ângulo de 90 graus, previsto no percurso programado.

Para o experimento foram realizadas cinco medições completas no percurso descrito para cada configuração de sensores. A primeira configuração utilizou a apenas informações da odometria da roda coletada através dos codificadores incrementais. A segunda configuração utilizou a fusão dos dados do *encoder* e da odometria visual obtida através da câmera Kinect. A terceira configuração utilizou os resultados da fusão dos dados de odometria visual e do IMU. A quarta configuração foi composta pelo codificador incremental e pelo sensor inercial. A quinta e última configuração contém todos os sensores embarcados para coleta de dados, nesse caso o *encoder* (codificador incremental), IMU (sensor inercial) e Kinect (odometria visual), e realiza a fusão sensorial de todas essas fontes. Com os dados obtidos em campo é possível identificar a contribuição de cada sensor na precisão do sistema de localização do robô.

Figura 44 - Percurso realizado pelo robô durante o experimento com a indicação dos erros lineares e angulares




Fonte: Autoria própria.

Vale salientar que no início de cada um dos testes gravou-se os dados resultantes de

odometria do sistema de localização em *rosbags*. O pacote *rosbag* é um conjunto de ferramentas para gravação e reprodução de tópicos do ROS. Desta forma, foi possível armazenar essas informações, converter em arquivos de texto e realizar análises matemáticas em planilhas eletrônicas.

Os comandos do ROS para gravação dos tópicos, reprodução dos dados e conversão dos arquivos "bag" em arquivos ".txt", podem ser executados respectivamente por:

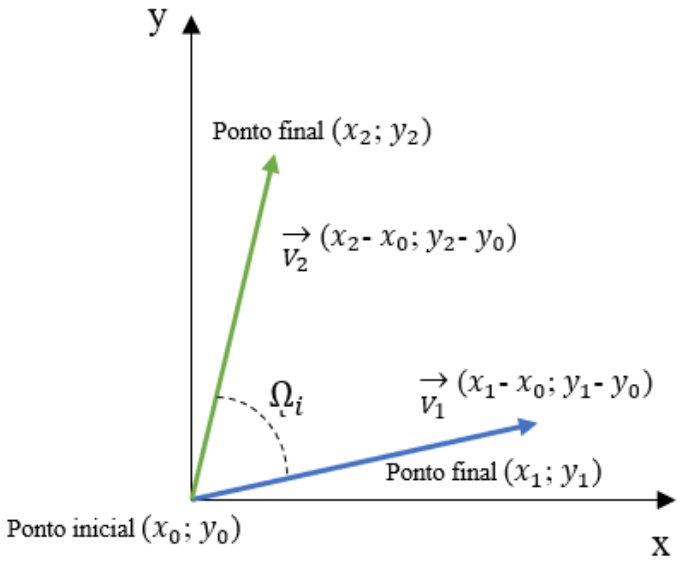
*$ rosbag record nome_do_tópico1 nome_do_tópico2 nome_do_tópicoX*

*$ rosbag play nome_do_arquivo.bag*

*$ rostopic echo -b nome_do_arquivo.bag -p /nome_do_tópico > nome_desejado.txt*

A aferição do ângulo $\Omega_i$ ilustrado na Figura 45 é calculado baseada na posição do vértice do quadrado ($X_0$,$Y_0$) na medição da localização dos pontos de referência antes ($X_1$,$Y_1$) e depois ($X_2$,$Y_2$) da rotação do robô.

Figura *45* - Representação dos vetores inicial e final da rotação do robô




Fonte: Autoria própria.

As coordenadas dos pontos ($X_0$,$Y_0$) e ($X_1$,$Y_1$) define o vetor de direção $V_1$ do robô antes da rotação e as coordenadas dos pontos ($X_0$,$Y_0$) e ($X_2$,$Y_2$) define o vetor de direção $V_2$ do robô após a rotação. O ângulo entre os vetores $V_1$ e $V_2$ representa a rotação efetiva do robô ($\Omega$), medida no *checkpoint*.

O ângulo de rotação efetiva, em graus decimais, é calculado pela equação (16):

$$\Omega_i = \frac{180}{\pi} * \left( \cos^{-1} \left( \frac{x_1 * x_2 + y_1 * y_2}{\sqrt{x_1^2 + y_1^2} * \sqrt{x_2^2 + y_2^2}} \right) \right) \quad (16)$$

Para cada medição foram calculados os erros quadráticos médios lineares e angulares ($RMSE_{linear}$ e $RMSE_{angular}$, respectivamente). É importante ressaltar que em um sistema de localização ideal o robô irá executar a trajetória perfeita, ou seja, os erros seriam nulos. Entretanto, os experimentos físicos estão propensos a erros mecânicos e sistemáticos que são acumulativos ao longo do tempo. Desta forma, quanto mais um erro aproxima-se de zero, mais o robô se aproxima do sistema de localização ideal.

As equações (17) e (18) demonstram como foram calculados os erros totais para cada teste. Na equação (17), $y_i$ e $x_i$ são os valores reais coletados durante a execução dos testes e as variáveis $\hat{X}_i$ e $\hat{Y}_i$ são os valores das coordenadas dos *checkpoints*. Na equação (18), $\Omega_i$ é o ângulo de rotação realizado pelo robô e 90 é o ângulo reto do quadrado. Em ambas equações, N é o número de amostras.

$$RMSE_{linear} = \frac{1}{N}\sum_{i}^{N}\sqrt{((x_i - \hat{X}_i)^2 + (y_i - \hat{Y}_i)^2)} \quad (17)$$

$$RMSE_{angular} = \frac{1}{N}\sum_{i}^{N}\sqrt{((\Omega_i - 90)^2)} \quad (18)$$

### 5.1.1 Primeira configuração: *Encoder*

A primeira configuração do sistema de localização utilizou somente os *encoders* das rodas para obter a odometria. Este sensor fornece dados que possibilitam a extração de informações do movimento linear e angular. Como mencionado na Seção 5, o *encoder* estima a velocidade, orientação e a posição do robô em relação ao ponto inicial do movimento através das equações 12, 13 e 14 do sistema de odometria do robô.

A Tabela 2 mostra os erros quadráticos médios lineares e angulares para a configuração utilizando somente o *encoder*. Como era de se esperar, a utilização de um único sensor produz erros RMSE linear e RMSE angular elevados. Em média, o robô se afastou 16.37 centímetros do ponto de medição em um percurso total de pouco menos de três metros. Os erros angulares médios foram de 37.20 graus, para curvas de 90 graus que o robô deveria realizar. Estes valores são inaceitáveis para a maioria das aplicações práticas, o que reforça a ideia que a localização de robôs não pode se basear somente nos dados oriundos dos *encoders*.

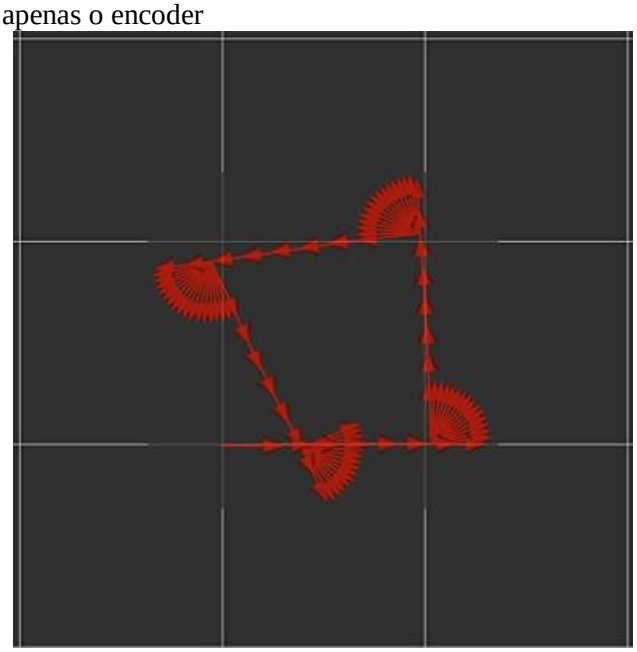
Tabela 2 - Resultados experimentais utilizando somente o Encoder

| **Sensor** | ***Encoder*** | |
|---|---|---|
| **Testes** | RMSE Linear | RMSE Angular |
| **1** | 20.90 | 45.64 |
| **2** | 17.43 | 47.73 |
| **3** | 15.48 | 29.53 |
| **4** | 11.70 | 24.06 |
| **5** | 16.36 | 39.05 |
| **Média** | 16.37 | 37.20 |

Fonte: Autoria própria.

O desempenho do *encoder* como único sensor do sistema de localização do robô para a maioria das tarefas, que demandam precisão do posicionamento, é ineficiente. Pode-se visualizar na Figura 46 como o acúmulo de erros lineares e angulares afetam a navegação da plataforma robótica móvel e desloca-se da trajetória desejada em um pequeno percurso predefinido. A malha do grid é composta de retângulos de 70,7x70,7 cm.

Figura 46 - Odometria resultante do sistema de localização em teste utilizando apenas o encoder



Fonte: Autoria própria.

Vale salientar que os sensores de localização relativas acumulam erros durante a navegação, ou seja, quanto mais longo é o percurso, maior é o erro. No caso dos *encoders*, estes erros podem ser ocasionados por derrapagem de uma roda, desalinhamento físico entre as rodas, entre outros fatores. Como realizamos o experimento em ambiente controlado e percursos curtos, consideramos desprezíveis as contribuições dos erros incrementais nos resultados do experimento.

As demais configurações dos sensores de posicionamento utilizaram a técnica de fusão

sensorial e combinaram dados oriundos de pelo menos dois sensores.

### 5.1.2 Segunda configuração: *Encoder* e Kinect

A segunda configuração do sistema de localização fundamentou-se na técnica de fusão dos dados diferenciais dos *encoders* e das informações coletadas pela câmera Kinect. Os dados de odometria visual obtidos pelo nó *rgbd_odometry* foram combinadas com os dados da odometria coletada pelos codificadores incrementais. A fusão dos sensores utilizou o Filtro de Kalman Estendido e pacote robot_localization para estimar a posição do robô, resultando em um sistema de localização mais preciso em relação ao experimento anterior.

Tabela 3 - Resultados experimentais utilizando Encoder e Kinect.

| **Sensor** | ***Encoder e Kinect*** | |
|---|---|---|
| **Testes** | RMSE Linear | RMSE Angular |
| **1** | 4.03 | 4.22 |
| **2** | 6.43 | 11.11 |
| **3** | 6.57 | 16.51 |
| **4** | 4.42 | 4.69 |
| **5** | 6.27 | 12.40 |
| **Média** | 5.55 | 9.49 |

Fonte: Autoria própria.

A Tabela 3 demonstra os erros quadráticos médios lineares e angulares para a configuração utilizando *encoder* e Kinect. É possível visualizar mudanças relevantes nos erros RMSE linear e RMSE angular em relação a primeira configuração. Em média, o robô se afastou 5.55 centímetros do ponto de medição em sua trajetória. Os erros angulares médios foram de 9.49 graus, para as curvas de 90 graus pretendidas. Estes valores ainda são ineficazes para a maioria das aplicações, no entanto, é possível observar uma melhora considerável no sistema de localização com a fusão sensorial.

### 5.1.3 Terceira configuração: Kinect e IMU

A terceira configuração do sistema de localização utilizou a fusão sensorial dos dados de odometria visual da câmera RGB-D e das informações de orientação, velocidade, deslocamento e velocidade angular provenientes dos sensores inerciais. Esta configuração é a única que não utiliza os dados dos *encoders*, configuração muito improvável de ser encontrada na prática.

A Tabela 4 demonstra os erros quadráticos médios lineares e angulares para a configuração utilizando Kinect e IMU. É possível visualizar uma mudança favorável e

considerável nos erros RMSE angular em relação a primeira configuração e um aumento demasiado e prejudicial no RMSE linear em relação as duas configurações anteriores, o que indica a imprecisão desses dois sensores em coletar os movimentos lineares. Em média, o robô se afastou 26.55 centímetros do ponto de medição em seu percurso. Os erros angulares médios foram de 8.72 graus, para as curvas de 90 graus que o robô deveria realizar. Estes valores são insuficientes para a maioria das tarefas que necessitam de uma alta precisão nos sistemas de localização.

Tabela 4 - Resultados experimentais utilizando Kinect e IMU

| **Sensor** | ***Kinect e IMU*** | |
|---|---|---|
| **Testes** | RMSE Linear | RMSE Angular |
| **1** | 35.05 | 4.75 |
| **2** | 18.12 | 12.65 |
| **3** | 13.58 | 9.80 |
| **4** | 40.08 | 4.69 |
| **5** | 25.94 | 11.69 |
| **Média** | 26.55 | 8.72 |

Fonte: Autoria própria.

### 5.1.4 Quarta configuração: *Encoder* e IMU

A quarta configuração do sistema de localização utilizou a fusão sensorial dos dados de odometria do movimento diferencial dos codificadores incrementais (*encoders*) combinados à estimativas de posicionamento do robô e os dados de orientação, velocidade, deslocamento e velocidade angular do sensor inercial (IMU). Esta talvez seja uma das configurações mais comumente encontradas em plataformas robóticas terrestres de baixo custo.

Tabela 5 - Resultados experimentais utilizando Encoder e IMU

| **Sensor** | ***Encoder e IMU*** | |
|---|---|---|
| **Testes** | RMSE Linear | RMSE Angular |
| **1** | 4.53 | 6.67 |
| **2** | 3.53 | 5.10 |
| **3** | 4.95 | 8.52 |
| **4** | 3.55 | 5.39 |
| **5** | 4.24 | 7.25 |
| **Média** | 4.16 | 6.59 |

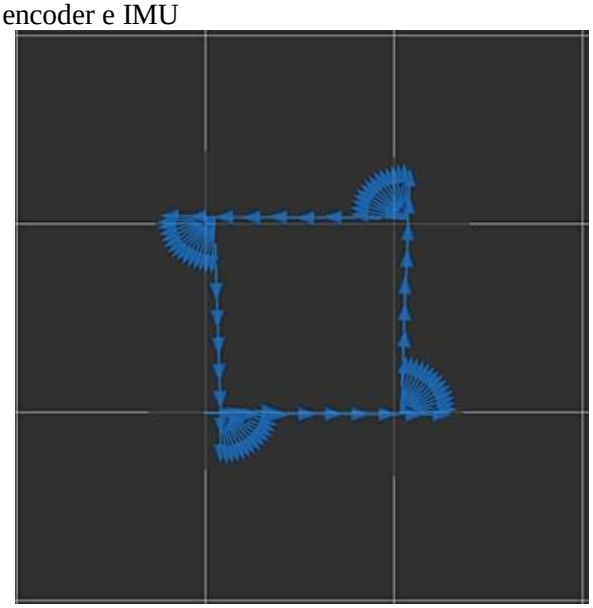
Fonte: Autoria própria.

A Tabela 5 apresenta os resultados dos erros lineares e angulares quando se realiza a fusão do *encoder* e o IMU. Comparando os valores dos erros das Tabelas 2, 3 e 4, nota-se que a fusão desses sensores proporciona uma melhora significativa ao sistema de localização. Em média, o robô apresentou desvios de 4.16 centímetros em relação as coordenadas dos

*checkpoints*. Vale destacar que o robô apresentou um desvio médio angular de aproximadamente 6.59 graus. Estes valores são plausíveis para as aplicações robóticas que necessitam de navegação autônoma e uma precisão aceitável para alguns sistemas de localização.

A melhoria do erro angular já era esperada nesta configuração. Como o IMU já realiza a fusão dos dados dos seus sensores internos (acelerômetros e giroscópios), informações mais precisas relacionadas as rotações sofridas pelo robô eram esperadas. A melhora no erro linear, entretanto, superou as expectativas. Os deslocamentos do robô obtidos somente através de sensores inerciais dependem da integração dupla das acelerações na direção do deslocamento. Os dados dos acelerômetros possuem elevado nível de ruído, que são potencializados pelo processo de integração. Diante deste fato, não se imaginava que o sensor IMU pudesse contribuir significativamente com a precisão linear do movimento.

Figura 47 - Odometria resultante do sistema de localização em teste utilizando o encoder e IMU



Fonte: Autoria própria.

É importante mencionar que essa configuração apresenta os melhores resultados encontrados neste experimento para configurações com apenas dois sensores. A Figura 47 ilustra a odometria de um dos testes do sistema de localização utilizando o *encoder* e IMU.

Como pode ser observado na Figura 47, a plataforma robótica desloca-se pouco de sua rota original, com pequenos erros lineares e angulares em sua trajetória percorrida, entretanto, é um resultado satisfatório considerando os resultados apresentados nas Tabelas 3, 4 e 5, que também possuem configurações de sistemas com dois sensores.

### 5.1.5 Quinta configuração: *Encoder*, IMU e Kinect

A quinta configuração do sistema de localização e última fase do experimento utilizou a fusão de todos os três sensores: *encoder*, IMU e Kinect. É natural que esta configuração proporcione melhorias nos erros lineares e angulares. A Tabela 6 mostra que em média o robô apresentou desvio de 2.17 centímetros em relação as coordenadas dos *checkpoints* e o ângulo de rotação 3.63 graus em relação aos 90 graus previstos.

Tabela 6 - Resultados experimentais utilizando Encoder, IMU e Kinect

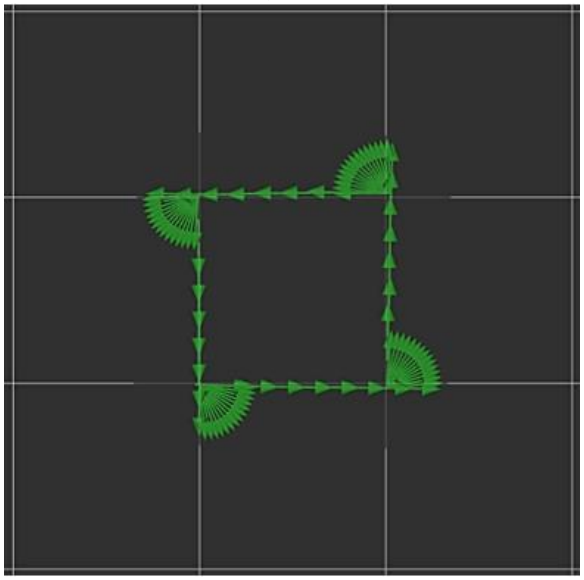
| **Sensor** | ***Encoder, IMU e Kinect*** | |
|---|---|---|
| **Testes** | RMSE Linear | RMSE Angular |
| **1** | 2.05 | 4.08 |
| **2** | 2.06 | 4.71 |
| **3** | 2.70 | 4.72 |
| **4** | 2.20 | 2.83 |
| **5** | 1.85 | 1.83 |
| **Média** | 2.17 | 3.63 |

Fonte: Autoria própria.

A configuração com fusão de dados de três sensores se constitui na configuração mais precisa entre todas as configurações testadas. A odometria resultante da fusão de dados desses sensores é satisfatória para a maioria das tarefas robóticas em que a navegação é indispensável, uma vez que possui erros RMSE linear e angular bem menores e estáveis, os quais representam a qualidade do modelo e precisão do sistema de localização.

Apesar dos resultados insatisfatórios em termos de precisão na segunda e terceira configurações que utilizam o Kinect para executar a tarefa de navegação autônoma, a contribuição desse sensor nos resultados de odometria da quinta configuração é considerável. É importante destacar que a introdução da odometria visual estéreo do Kinect tem o potencial de amenizar tanto os erros lineares quanto os erros angulares quando combinados com o *encoder* e IMU. A Figura 48 ilustra a odometria resultante do sistema de localização do robô móvel utilizando todos os sensores desse estudo, os quais proporcionam uma ótima precisão nos sistemas de localização para desempenhar a navegação.

Figura 48 - Odometria resultante do sistema de localização em teste utilizando o encoder, IMU e Kinect



Fonte: Autoria própria.

## 5.2 DISCUSSÃO DOS RESULTADOS

Os resultados do estudo conduzem a algumas conclusões sobre as vantagens e desvantagens na fusão dos sensores para navegação de robôs móveis em ambientes fechados. Os dados do experimento podem ser analisados individualmente e em conjunto para a comparação entre as diferentes configurações. Estas análises têm o objetivo de identificar características importantes sobre cada um dos sensores utilizados e indicações dos ganhos efetivos quando combinados.

A Tabela 7 indica os erros quadráticos médios lineares e angulares para cada configuração. Adotando a configuração com o *encoder* somente como configuração de referência, pode-se perceber que a única configuração que possui um erro RMS linear maior que a configuração de referência é a configuração 3 (Kinect e IMU). Este resultado indica que o *encoder* individualmente possui uma precisão em movimentos lineares melhor do que a combinação entre Kinect e IMU. Por outro lado, a cominação Kinect e IMU melhora significativamente os erros angulares do movimento, quando comparado a configuração de referência.

Tabela 7 - Média dos erros RMS linear e angular de cada configuração

| **Configurações** | **Erros** | |
|---|---|---|
| | RMSE Linear | RMSE angular |
| ***1 - Encoder*** | 16,37 | 37,20 |
| ***2 - Encoder e Kinect*** | 5,55 | 9,49 |
| ***3 - Kinect e IMU*** | 26,55 | 8,72 |
| ***4 - Encoder e IMU*** | 4,16 | 6,59 |
| ***5 - Encoder, IMU e Kinect*** | 2,17 | 3,63 |

Fonte: Autoria própria.

Conforme já verificado no parágrafo anterior e reforçado pela coluna com os erros angulares (Tabela 7), a configuração 1 (*encoder*) é a menos indicado para coletar informações angulares fidedignas. Em todas as outras configurações os resultados demonstram erros médios RMS angular menores do que 10 graus, enquanto a média do *encoder* operando individualmente é de 37.20 graus, o que representa um erro RMS angular de aproximadamente 292% maior do que o segundo maior erro RMS angular desse experimento (configuração 2).

As comparações entre configurações ficam mais evidentes quando se apresentam os dados lineares e angulares tomando como referência cada uma das configurações. A tabela 8, por exemplo, indica a variação do erro médio tomando cada uma das configurações como referência. Assim, na primeira linha da tabela, a configuração 1 é tomada como referência e os valores na linha indicam a variação dos erros lineares (em percentual) para as demais configurações. Valores positivos indicam um aumento no erro médio e valores negativos uma redução no erro médio. Desta forma, considerando a configuração 1 como referência, constata-se que as configurações 2, 4 e 5 melhoram a precisão das medições lineares em 66%, 75% e 87%, respectivamente, enquanto a configuração 3 reduz a precisão das medições lineares em 62%.

Tabela 8 - Variação percentual dos erros lineares RMS entre configurações

| **Configuração** / **Configuração de Referência** | 1 – ***Encoder*** | 2 – ***Encoder e Kinect*** | 3 – ***IMU e Kinect*** | 4 – ***Encoder e IMU*** | 5 – ***Encoder, IMU e Kinect*** |
|---|---|---|---|---|---|
| 1 – ***Encoder*** | | -66% | 62% | -75% | -87% |
| 2 – ***Encoder e Kinect*** | 195% | | 378% | -25% | -61% |
| 3 – ***IMU e Kinect*** | -38% | -79% | | -84% | -92% |
| 4 – ***Encoder e IMU*** | 294% | 33% | 538% | | -48% |
| 5 – ***Encoder, IMU e Kinect*** | 654% | 156% | 1.124% | 92% | |

Fonte: Autoria própria.

As mesmas comparações entre as diversas as configurações também podem ser feitas para os erros angulares (Tabela 9). Tomando novamente como base a configuração com

*encoder* somente (configuração 1), todas as demais configurações apresentam melhoria na precisão do sistema, com erros RMS caindo 74%, 77%, 82% e 90%, para as configurações 2, 3, 4 e 5, respectivamente.

Tabela 9 - Variação percentual dos erros angulares RMS entre configurações

| Configuração / Configuração de Referência | 1 – *Encoder* | 2 – *Encoder e Kinect* | 3 – *IMU e Kinect* | 4 – *Encoder e IMU* | 5 – *Encoder, IMU e Kinect* |
|---|---|---|---|---|---|
| 1 – *Encoder* | | -74% | -77% | -82% | -90% |
| 2 – *Encoder e Kinect* | 292% | | -8% | -31% | -62% |
| 3 – *IMU e Kinect* | 327% | 9% | | -24% | -58% |
| 4 – *Encoder e IMU* | 464% | 44% | 32% | | -45% |
| 5 – *Encoder, IMU e Kinect* | 925% | 161% | 140% | 82% | |

Fonte: Autoria própria.

Analisando as configurações 2 e 4, onde os sensores Kinect e IMU foram associados individualmente ao *encoder*, constata-se um ganho significativo nas medições lineares e angulares, quando comparados com a configuração 1. A redução dos erros lineares e angulares observados nas configurações 2 e 4 estão na mesma ordem de grandeza, com uma ligeira vantagem para a configuração com IMU (configuração 4). Enquanto o Kinect diminui o erro linear em 66%, o IMU alcança 75% de redução. Para os erros angulares, o Kinect melhora a precisão do erro angular em 74% e o IMU em 77%. Pelo resultado desta análise, conclui-se que se a intenção é acrescentar apenas mais um sensor em um sistema com *encoders* somente, a melhor opção é a utilização de sensores inerciais. Estes sensores, além de melhorarem a precisão da localização linear e angular mais do que o sensor Kinect, são mais baratos que os sensores de imagem e requerem menor poder de processamento do computador de bordo para realizar a odometria.

Considerando os erros lineares somente, a configuração 3 (IMU e Kinect) é a pior configuração, mesmo quando comparada com a configuração 1, que possui um único sensor. Sabe-se que os sensores IMU contribuem com informações precisas sobre rotações, mas não são os mais indicados quando se trata de medir distâncias lineares. É possível mensurar a ineficiência relativa da configuração 3, pois as configurações 1, 2, 4 e 5 são melhores que ela em 38%, 79%, 84% e 92%, respectivamente. Em se tratando de erros angulares, a configuração 3 perde em precisão para as configurações 1 e 2. A configuração 1 apresenta um aumento de 327% no erro angular, já a configuração 2 um aumento de 292%, quando comparada com a configuração 3.

Corroborando o que sugere o senso comum, a configuração 5 (utilização de todos os sensores) produz os melhores resultados, tanto no que refere os deslocamentos lineares quanto

no tange às rotações. Assim, as informações mais importantes na análise comparativa desta configuração é o percentual de ganho efetivo quando comparada com as demais configurações. Com relação a configuração 1, por exemplo, a diminuição dos erros lineares e angulares foi de 654% e 925%, respectivamente. A melhora acentuada nos erros angulares já era esperada, pois sabe-se que o *encoder* produz informações mais precisas quando se trata de medir distâncias lineares do que os ângulos de rotações.

Quando configuração 5 é comparada com as configurações 2 e 4, percebe-se uma melhoria na precisão lineares e angulares na mesma ordem de grandeza, mas a melhoria com relação a configuração 4 (*encoder* e IMU) é ligeiramente menor que a melhoria com relação a configuração 2 (*encoder* e Kinect).

A comparação entre a melhor combinação de sensores (configuração 5) com a pior combinação (configuração 3) produz resultados significativos. O erro linear quando se retira o *encoder* da configuração 5 aumenta 1.124%. Já o erro angular é menos impactado, com aumento de 140%.

Baseado nos dados do experimento, pode-se chegar a algumas recomendações de ordem prática. As plataformas robóticas terrestres com rodas ou esteiras devem sempre instalar *encoders* nas rodas para realização da odometria. Quando utilizamos sensores para odometria inercial e visual sem a presença de *encoders* (configuração 3) os erros lineares e angulares são inaceitáveis para a maioria das aplicações.

Caso se deseje incrementar o sistema de odometria que possua o *encoder* somente, sugere-se a inclusão de um sensor inercial ou câmera de profundidade, com preferência para o primeiro tipo de sensor. A melhor solução entreato, é a utilização dos três tipos de sensores, pois a melhora na precisão da orientação e localização do robô valem o custo de implementação desta solução. A melhora do erro linear, quando comparada a segunda melhor opção é de 92%. Para os erros angulares, esta melhora atinge 82%. Um erro linear total de 2,17 cm em um percurso de 282 cm, equivale a um erro acumulado de ~0,75%, e um erro angular total de 3,63º após a realização de 4 rotações de 90º (360º) equivale a um erro acumulado de ~1%. Esta precisão é aceitável para a maioria das aplicações de robótica.

# 6 CONCLUSÃO

Um dos principais desafios da navegação autônoma para a robótica móvel é o conhecimento do mapa do ambiente onde será realizada a navegação e o conhecimento da posição e orientação do robô neste ambiente. Além disso, o robô deve possuir sensores capazes de perceber no ambiente a presença de objetos móveis ou obstáculos que não constam do mapa do ambiente. Todas estas informações são relevantes para que o sistema de navegação possa definir a melhor rota para o robô atingir seu objetivo e impedir que o robô colida com pessoas ou outros objetos no caminho.

O mapa do ambiente pode ser carregado na memória do robô ou construído enquanto o robô navega e faz o reconhecimento do ambiente (SLAM). A detecção de obstáculos é feita por sensores de percepção, tais como, ultrassom, lasers e câmeras de vídeo. A localização do robô é feita por sensores de posicionamento relativos ou absolutos. Este trabalho propõe analisar a precisão dos sistemas de posicionamento baseado na odometria obtida das rodas, de sensores inerciais e da odometria visual e o impacto da fusão destes mecanismos na precisão da localização do robô. O mapeamento do ambiente e a detecção de obstáculos estão foras do escopo deste trabalho e não foram incluídos na análise.

Este trabalho apresentou uma avaliação dos sistemas de localização para ambientes fechados implementado em robôs móveis utilizando a técnica de fusão sensorial com Filtro de Kalman Estendido. Os experimentos de laboratório foram realizados com a finalidade de extrair informações de dados provenientes da odometria da roda, de sensores inerciais e do sequenciamento de imagens (convertidos em odometria visual). As fontes de dados selecionadas foram baseadas em sensores mencionados com frequência em trabalhos que envolvem a navegação de robôs terrestres.

Objetivando avaliar a precisão relativa obtidas pelas diversas combinações de sensores, foi realizado um experimento com cinco configurações de sensores. A configuração mais básica utiliza apenas dados dos *encoders* (odometria das rodas). A segunda configuração adiciona as informações da odometria visual da câmera de vídeo. A terceira configuração remove o *encoder* do sistema e utiliza a câmera Kinect e o sensor inercial para obter a localização do robô. A quarta configuração consiste em dados provenientes do IMU (odometria inercial) e *encoder* (odometria das rodas). A última configuração incorpora os dados provenientes de todos os sensores avaliados nessa pesquisa, nesse caso são os *encoders*, o sensor inercial e a câmera RGB-D Kinect.

A avaliação do impacto que as diferentes configurações de sensores têm sobre a

localização do robô foi analisado em um percurso quadrado predefinido com diagonal medindo aproximadamente um metro. Nesse experimento foi possível analisar na navegação a precisão do movimento linear e angular. Na análise foram utilizados os erros médios quadráticos linear e angular para mensurar a precisão da navegação. Esses indicadores estatísticos foram utilizados como forma de obter uma medida quantitativa da qualidade da navegação em cada configuração dos experimentos. Os resultados do experimento podem ser utilizados subsidiar a tomada de decisão e a análise de custo-benefício quanto a utilização de determinada combinação de sensores.

## 6.1 LIMITAÇÕES

Este trabalho executou todos os testes em uma plataforma real. Esta plataforma robótica real não existia no nosso grupo de pesquisa, sendo necessário a construção do robô a partir do zero. Esta foi uma dificuldade encontrada no desenvolvimento deste trabalho, pois demandou tempo para a manufatura da plataforma propriamente dita e para a instalação, configuração e entendimento de todos os programas instalados na plataforma. Embora a construção da plataforma robótica não seja uma contribuição no objeto desta dissertação, esta plataforma fica como legado para trabalhos futuros, que poderão focar inteiramente nos seus objetivos, tendo a disposição um robô inteiramente funcional e operacional.

Os experimentos desse estudo foram realizados no laboratório de pesquisa Ganges, onde existe uma limitação de espaço físico para realizar percursos longos. Considerando esta realidade, esse trabalho realizou a análise dos sistemas de localização somente em um curto percurso. Entendemos que experimentos em percursos longos sejam fundamentais para avaliar o comportamento dos sensores a longo termo e como a evolução dos erros lineares e angulares evoluem e se acumulam à medida que a distância percorrida cresce e número de rotações aumenta.

Embora os experimentos desse estudo tenham sido realizados com sucesso, foi observado durante os testes uma limitação do desempenho da Raspberry Pi 3. Apesar desse microcomputador ter sido capaz de executar as atividades designadas nesse projeto, verificou-se que nos experimentos com três sensores o dispositivo ficou no limiar de sua performance. É provável que a adição de muitos sensores pode sobrecarregar o microcomputador e seja necessário modificar o computador de bordo para experimentos mais robustos.

## 6.2 TRABALHOS FUTUROS

Este é um trabalho com múltiplas possibilidades de extensão. A título de trabalhos futuros sugere-se aumentar a quantidade de sensores de localização com a incorporação de tecnologias distintas às tecnologias utilizadas neste experimento, o aumento no número de combinações de sensores e a realização de percursos maiores e mais elaborados.

Uma das possibilidades de implementação é a utilização de câmeras de vídeo simples para realizar a odometria visual monocular. A odometria monocular apresenta como vantagem necessitar de um *hardware* relativamente barato e de fácil instalação, além de não demandar elevada capacidade de processamento do computador de bordo. Acreditamos que os dados provenientes da odometria visual monocular pode aumentar significativamente a precisão dos movimentos de rotação do robô. Não se espera, no entanto, melhoras significativas no movimento linear, pois é difícil inferir uma escala para o deslocamento e inferir a distância real percorrida.

Outra opção interessante é a utilização de *smartphones* para coleta de dados para o ROS, uma vez que esses dispositivos se tornaram mais potentes, funcionais e financeiramente acessíveis para a sociedade. As características dos *smartphones* incluem processadores *multicore* de alto desempenho, arquiteturas multimodais, versatilidade nos recursos de rede, diversos sensores integrados, como acelerômetros, giroscópios (mesmos sensores que compõem o IMU), câmeras monoculares e em alguns casos duas câmeras (permitindo coletar a odometria estéreo), além de custos relativamente baixos, o que torna viável a utilização desses dispositivos integrados à plataforma robótica em pesquisas. Os *smartphones* também possibilitam tratar os dados dos sensores de localização no próprio dispositivo, o que reduziria a carga de processamento no computador de bordo. Em um cenário ainda mais desejável a médio prazo seria a incorporação do *smartphone* como computador de bordo da plataforma robótica.

No que tange ao caminho realizado pelo robô, considera-se imprescindível aumentar os percursos tanto em tamanho e quanto em complexidade. Alguns percursos considerados no desenvolvimento desta pesquisa foram: percurso serrilhados, trajetórias em escada, caminhos senoidais, rotas circulares, entre outros. As adaptações para tornar isso viável seria a alteração da localização dos *checkpoints* em cada experimento e ponderar das referências angulares para cálculo de orientação, o que pode ser analisado primeiramente em simulações no Gazebo e então implementada as referências no sistema.

Finalmente, consideramos importante acrescentar um sensor de localização absoluto,

como por exemplo o sensor de localização por satélite (GPS). A incorporação de um sensor de localização absoluto irá incluir a configuração de percursos em ambientes abertos.

# REFERÊNCIAS


ALATISE, Mary B. ; HANCKE, Gerhard P. Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended kalman filter. **Sensors**, v. 17, n. 10, p. 2164, 2017.

ARNOLD, Sascha ; MEDAGODA, Lashika. Robust Model-Aided Inertial Localization for Autonomous Underwater Vehicles. *In*: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2018. **Proceedings** […] 2018. p. 4889–4896.

BAILEY, Tim ; DURRANT-WHYTE, Hugh. **Si**multaneous localization and mapping (SLAM): Part II. **IEEE Robotics and Automation Magazin**e, v. 13, n. 3, p. 108–117, 2006.

BAÚ DA ELETRÔNICA. **Sensor de Distância Ultrassônico HC-SR04**. Disponível em: https://www.baud ELETRÔNICA aeletronica.com.br/modulo-de-sensor-ultrassonico-hc-sr04.html. Acesso em: 25 jan. 2020.

BAY, Herbert ; TUYTELAARS, Tinne ; GOOL, Luc Van. LNCS 3951 - SURF: Speeded Up Robust Features. *In*: EUROPEAN CONFERENCE ON COMPUTER VISION. ECCV 2006, 2006. **Proceedings** […] 2006. p. 404–417.

BORENSTEIN, J. e colab. Mobile robot positioning: Sensors and techniques. **Journal of Robotic Systems,** v. 14, n. 4, p. 231–249, 1997.

BORENSTEIN, J.; EVERETT, H. R. ; FENG, L. **Where am I? Sensors and Methods for Mobile Robot Positioning**. 1996.

BOSTONDYNAMICS. **ATLAS**. Disponível em: https://www.bostondynamics.com/atlas. Acesso em: 25 jan. 2020.

BRADSKI, Gary ; KAEHLER, Adrian. **Learning OpenCV: C**omputer Vision with the OpenCV Library. [*S.l.*]: O'Reilly Media, 2008.

BREITENMOSER, A. ; KNEIP, L. ; SIEGWART, R. A monocular vision-based system for 6D relative robot localization. *In*: **2011 IEEE/RSJ I**NTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. IEEE, 2011. **Proceedings** […] 2011.p. 79-85.

CHO, Bong Su e colab. A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. **Journal of Mechanical Science and Technology**, v. 25, n. 11, p. 2907–2917, 2011.

DAS, Sagarnil. Robot localization in a mapped environment using Adaptive Monte Carlo algorithm. **International Journal of Scientific & Engineering Research,** v. 9, n. 10, p. 77–85, 2018.

DEILAMSALEHY, Hanieh; HAVENS, Timothy C. Sensor fused three-dimensional localization using IMU, camera and LiDAR. *In*: IEEE SENSORS. 2017. **Proceedings** […] 2017. p. 1–3.

DURRANT-WHYTE; Hugh e BAILEY, Tim. **S**imultaneous localization and mapping: Part I. IE**EE Robotics and Automation Magazin**e, v. 13, n. 2, p. 99–108, 2006.

FABIAN, Joshua ; CLAYTON, Garrett M. Error analysis for visual odometry on indoor, wheeled mobile robots with 3-D sensors. **IEEE/ASME Transactions on Mechatronics,** v. 19, n. 6, p. 1896–1906, 2014.

FARAGHER, Ramsey. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes**]**. **IEEE Signal Processing Magazine**, v. 29, n. 5, p. 128–132, 2012.

FERNÁNDEZ-MADRIGAL;, Juan-Antonio ; CLARACO, José Luis Blanco. **Simultaneous Localization and Mapping for Mobile Robots:** Introduction and Method**s**. [*S.l.*]: Information Science Reference, IGI Global, 2013.

IEEE SENSORS. 2

FOX, Dieter ; BURGARD, Wolfram ; DELLAERT, Frank e colab. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots Dieter Fox, Wolfram Burgard. *In*: AAAI-99 1999. **Proceedings** […], p. 7, 1999.

FOX, Dieter ; BURGARD, Wolfram; THRUN, Sebastian. **M**arkov Localization for Mobile Robots in Dynamic Environments. **Journal of Arti ficial Intelligence Research**, p. 391–427, 1999.

GASPARETTO, A. ; SCALERA, L. A Brief History of Industrial Robotics in the 20th Century. **Advances in Historical Studies**, v. 8, n. 1, p. 24–35, 2019.

GREICIUS, Tony. **Two Rovers to Roll on Mars Again:** Curiosity and Mars 2020. Disponível em: https://www.nasa.gov/feature/jpl/two-rovers-to-roll-on-mars-again-curiosity-and-mars-2020. Acesso em: 20 jun. 2020

HERNANDEZ, Sergi ; FERNANDO, Juan e COTARELO, Herrero. **Multi-master ROS systems**. [S.l.]: **Institut de Robotics and Industrial Informatics**, 2015.

HOCKSTEIN, N. G. e colab. A history of robots: From science fiction to surgical robotics. **Journal of Robotic Surgery,** v. 1, n. 2, p. 113–118, 2007.

HUSSEIN, Ahmed e colab. Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection. *In*: IEEE INTELLIGENT VEHICLES SYMPOSIUM, , 4., 2016. **Proceedings** […] 2016-Augus. p. 104–109.

JIMENO, Juan Miguel. **Base Controller**. Disponível em: https://github.com/linorobot/linorobot/wiki/2.-Base-Controller. Acesso em: 22 jul 2020.

JOSEPH, Lentin. **Learning Robotics Using Python**. [*S.l: s.n.*], 2015. v. 44.

KARAMI, Ebrahim; PRASAD, Siva ; SHEHATA, Mohamed. **Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images**. Computational and Mathematical Methods in Medicine, 2017. **arXiv preprint arXiv:1710.02726**, 2017.

KIM, Seong Jin ; KIM, Byung Kook. Dynamic ultrasonic hybrid localization system for indoor mobile robots. **IEEE Transactions on Industrial Electronics,** v. 60, n. 10, p. 4562–4573, 2013.

KUMMERLE, R. e colab. $G^2o$: A General Framework for Graph Optimization. 2011, [S.l:

s.n.], 2011. p. 3607–3613.

LABBÉ, Mathieu ; MICHAUD, François. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. **Journal of Field Robotics,** 2013.

LIGORIO, Gabriele; SABATINI, Angelo Maria. Extended Kalman filter-based methods for pose estimation using visual, inertial and magnetic sensors: Comparative analysis and performance evaluation. **Sensors,** Switzerland, v. 13, n. 2, p. 1919–1941, 2013.

LOEVSKY, I. ; SHIMSHONI, I. **R**eliable and efficient landmark-based localization for mobile robots. **Robotics and Autonomous Systems**, v. 58, n. 5, p. 520–528, 2010.

MARÍN-PLAZA, Pablo e colab. Stereo Vision-based Local Occupancy Grid Map for Autonomous Navigation in ROS. *In*: JOINT CONFERENCE ON COMPUTER VISION, IMAGING AND COMPUTER GRAPHICS THEORY AND APPLICATIONS, 11., 2016. **Proceedings** […] 2016. p. 701–706.

MARTINEZ, Aaron ; FERNÁNDEZ, Enrique. **Learning ROS for Robotics Programming**. [S.l: s.n.], 2013.

MENNA, Matteo e colab. Real-time autonomous 3D navigation for tracked vehicles in rescue environments. *In*: IEEE INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2014. **Proceedings** […] 2014. p. 696–702.

MICROSOFT. **Azure Kinect DK**. Disponível em: https://www.microsoft.com/en-us/p/azure-kinect-dk/8pp5vxmd9nhq?activetab=pivot%3Aoverviewtab. Acesso em: 22 jul 2020.

MOORE, Thomas ; STOUCH, Daniel. **A Generalized Extended Kalman Filter Implementation for the Robot Operating System**. [*S.l.*]: IAS, 2014.

NGUYEN, Lan Anh e colab. Improving the accuracy of the autonomous mobile robot localization systems based on the multiple sensor fusion methods. *In*: INTERNATIONAL CONFERENCE ON RECENT ADVANCES IN SIGNAL PROCESSING, TELECOMMUNICATIONS AND COMPUTING, SIGTELCOM 2019, 3., 2019. **Proceedings** […] 2019. p. 33–37.

NIST, David. Preemptive RANSAC for Live Structure and Motion Estimation Corresponding Author : Preemptive RANSAC for Live Structure and Motion Estimation. **Machine Vision and Applications**, v. 16, n. Iccv, p. 1–29, 2003.

O'KANE, Jason M.; KANE, Jason M. O. **A gentle introduction to ROS**. [*S.l: s.n.*], 2013.

OGATA, Katsuhiko. **Engenharia de Controle Moderno.** 4 ed. [*S.l.*]: Prentice Hall, 2003.

OYALLON, Edouard ; RABIN, Julien. An Analysis of the SURF Method. **Image Processing,** v. 5, n. 2004, p. 176–218, 2015.

PEDROSO, Ana Luiza. **Sensores Lidar - Entenda o que são e como funcionam!** Disponível em: https://mundoconectado.com.br/artigos/v/15382/sensores-lidar-entenda-o-que-sao-e-como-funcionam. Acesso em: 22 jul. 2020.

PINHEIRO, Paulo e colab. Cleaning Task Planning for an Autonomous Robot in Indoor Places with Multiples Rooms. *International Journal of Machine Learning and Computing*, v. 5, n. 2, p. 86–90, 2015.

QUIGLEY, Morgan e colab. ROS: an open-source Robot Operating System. **Icra,** v. 3, n. Figure 1, p. 5, 2009.

ROS_WIKI. **Grid Map**. Disponível em: http://wiki.ros.org/grid_map. Acesso em: 22 jul. 2020.

ROS_WIKI. **Learn the basics of openai_ros using a Turtlebot2 simulation.** Disponível em: <http://wiki.ros.org/openai_ros/TurtleBot2 with openai_ros>.

ROS_WIKI. **rplidar**. Disponível em: http://wiki.ros.org/rplidar. Acesso em: 22 jul. 2020.

ROS_WIKI. **Setting up your robot using tf**. Disponível em: http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF. Acesso em: 22 jul. 2020.

SIEGWART, Roland e colab. **Mobile Robots | Introduction and Lecture Overview Autonomous Mobile Robots**. Zurique: [*s.n.*], 2014

SOUZA, Fábio. **Arduino - Interface com acelerômetro e giroscópio.** Disponível em: https://www.embarcados.com.br/arduino-acelerometro-giroscopio/. Acesso em: 21 jul. 2020.

SURUR. **Most popular Xbox Kinect feature coming back next year**. Disponível em: https://mspoweruser.com/most-popular-xbox-kinect-feature-coming-back-next-year. Acesso em: 22 jul. 2020.

T. MOORE. **Working with the robot_localization Package**. Disponível em: http://wiki.ros.org/move_base. Acesso em: 22 jul. 2020.

TAI, Lei ; LIU, Ming. **Towards Cognitive Exploration through Deep Reinforcement Learning for Mobile Robots**. n. 16206014, 2016. **arXiv preprint arXiv:1610.01733**, 2016.

UXCELL. **uxcell DC 12V 350RPM Encoder Gear Motor with Mounting Bracket 65mm Wheel Kit for Smart Robot DIY**. Disponível em: https://www.amazon.com/uxcell-350RPM-Encoder-Mounting-Bracket/dp/B078HYJ3F8/ref=sr_1_25?crid=1O8NDG0KEKCIC&dchild=1&keywords=dc+motor+wheel+kit&qid=1612734598&sprefix=kit+motor+dc+%2Caps%2C432&sr=8-25. Acesso em: 22 jul. 2020.

VASUDEVAN, Shrihari e colab. Cognitive maps for mobile robots-an object based approach. **Robotics and Autonomous Systems**, v. 55, n. 5, p. 359–371, 2007.