



**CENTRO UNIVERSITÁRIO SENAI CIMATEC**

**PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM**

**COMPUTACIONAL E TECNOLOGIA INDUSTRIAL**

**Mestrado em Modelagem Computacional e Tecnologia Industrial**

**Dissertação de mestrado**

**Fusão de Sensores para Localização: Desenvolvimento de Modelos para  
um Veículo Robótico Subaquático**

Apresentada por: Lucas Marins Batista

Orientadora: Prof<sup>a</sup> Dr<sup>a</sup> Valéria Loureiro da Silva

Março de 2022

Lucas Marins Batista

**Fusão de Sensores para Localização: Desenvolvimento de Modelos para um  
Veículo Robótico Subaquático**

Dissertação de mestrado apresentada ao Programa de Pós-Graduação Stricto Sensu do Centro Universitário SENAI CIMATEC como requisito final para a obtenção do título de **Mestre em Modelagem Computacional e Tecnologia Industrial**.

Orientadora: Prof<sup>a</sup> Dr<sup>a</sup> Valéria Loureiro da Silva

Salvador  
2022

Ficha catalográfica elaborada pela Biblioteca do Centro Universitário SENAI CIMATEC

B333f Batista, Lucas Marins

Fusão de sensores para localização: desenvolvimento de modelo para um veículo robótico subaquático / Lucas Marins Batista. – Salvador, 2022.

98 f. : il. color.

Orientadora: Prof.<sup>ª</sup> Dr.<sup>ª</sup> Valéria Loureiro da Silva.

Dissertação (Mestrado em Modelagem Computacional e Tecnologia Industrial) – Programa de Pós-Graduação, Centro Universitário SENAI CIMATEC, Salvador, 2022.

Inclui referências.

1. Fusão de sensores. 2. Robô subaquático. 3. ROS. 4. Sonar. 5. Inertial navigation system. 6. PointCloud2. I. Centro Universitário SENAI CIMATEC. II. Silva, Valéria Loureiro da. III. Título.

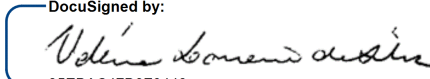
CDD 629.045

## CENTRO UNIVERSITÁRIO SENAI CIMATEC

**Mestrado Acadêmico em Modelagem Computacional e Tecnologia Industrial**

A Banca Examinadora, constituída pelos professores abaixo listados, aprova a Defesa de Mestrado, intitulada “**FUSÃO DE SENSORES PARA LOCALIZAÇÃO: DESENVOLVIMENTO DE MODELOS PARA UM VEÍCULO ROBÓTICO SUBAQUÁTICO**” apresentada no dia 11 de março de 2022, como parte dos requisitos necessários para a obtenção do Título de Mestre em Modelagem Computacional e Tecnologia Industrial.

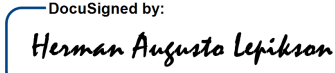
Orientadora:

DocuSigned by:  
  
95FBAC47B6F0449  
**Prof.ª Dr.ª Valéria Loureiro da Silva**  
SENAI CIMATEC

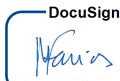
Membro Interno:

DocuSigned by:  
  
9F19B6195F7C4E8  
**Prof. Dr. Valter de Senna**  
SENAI CIMATEC

Membro Interno:

DocuSigned by:  
  
031471D7E0CA484...  
**Prof. Dr. Herman Augusto Lepikson**  
SENAI CIMATEC

Membro Externo:

DocuSigned by:  
  
E5EFC6B69A6E4B8...  
**Prof. Dr. Paulo César Machado de Abreu Farias**  
UFBA

## **AGRADECIMENTOS**

Agradeço, primeiramente, a Deus, pela força e energia para a conclusão deste trabalho.

À minha professora e orientadora Prof<sup>a</sup>. Dr<sup>a</sup>. Valéria Loureiro da Silva, pela atenção, paciência e todo o conhecimento passado na orientação.

A meus pais, pelo apoio emocional e incentivo ao estudo, durante toda a minha vida.

A todos os meus amigos e colegas do SENAI CIMATEC, pelo apoio e incentivo, especialmente a todos que trabalharam no projeto MCCR.

Ao SENAI CIMATEC, pela estrutura de laboratório, pelos equipamentos e pela bolsa de pesquisa, essenciais à realização da pesquisa.

À EMBRAPPII e à ANP pelo financiamento do projeto MCCR.

## RESUMO

O uso de veículos móveis autônomos e teleoperados cresce em várias áreas com o objetivo de automatizar diversos tipos de atividades. Com isso, a localização através de sensores é um dos elementos mais importantes que permitem que o robô possa identificar suas posições e realize atividades de forma mais precisa. Em um ambiente subaquático são utilizados diversos tipos de sensores como *Inertial Measurement Unit* ou *Inertial Navigation System* e a técnica de odometria do giro de motores ou rodas, especialmente quando as características do ambiente e situação não permitem o uso de dados globais. Na construção deste trabalho, foram estudados alguns desses sensores e técnicas para localização, principalmente, de forma simulada, utilizando *framework* para robótica ROS e simulador 3D *Gazebo* para construção do ambiente. Para isso, foram realizados experimentos iniciais com o veículo Husky, variando parâmetros, movimento e características no ambiente que permitissem avaliar a eficiência dos algoritmos de fusão de sensores. Além disso, foi desenvolvido um modelo de INS ("*Inertial Navigation System*") para testes no veículo MCCR (*MEC Combi Crawler Robot*), veículo robótico para inspeção de casco de navios, simulado no *Gazebo* em um tanque de testes, assim como um modelo para conversão de dado de sonares para nuvem de pontos do tipo *PointCloud2*. Após esses testes, foi demonstrado que foi possível utilizar esses modelos desenvolvidos com os algoritmos de fusão de sensores (Kalman estendido e *uncented* e de partículas) com o robô MCCR para redução do erro. Os resultados realçaram a importância de uma configuração adequada dos parâmetros dos algoritmos de fusão e variabilidade do impacto do escorregamento das rodas nos dados de odometria. Assim, uma otimização efetiva dos parâmetros de fusão para o robô MCCR será possível após a caracterização do sistema de odometria e validação de seu modelo.

Palavras-chave: fusão de sensores; robô subaquático; ROS; sonar; inertial navigation system; PointCloud2.

## ABSTRACT

Autonomous and teleoperated mobile vehicles use grows in several areas, aiming different activities for localization, and estimate position and orientation are important data that allow the robot to identify and perform it more accurately. In an underwater environment, using sensors, such as Inertial Measurement Unit, Inertial Navigation System, and wheels odometry, are essential when the environment does not allow the use of global data. In this work, sensors and techniques for localization were studied and simulated, using a ROS robotics *framework* and a 3D Gazebo simulator. After that, experiments with the Husky vehicle were performed, varying parameters, movement, and environment characteristics that allowed to evaluate the efficiency of the sensor fusion algorithms. In addition, an INS (Inertial Navigation System) model was developed for testing in the MCCR vehicle (MEC Combi Crawler Robot), a robotic vehicle for ship hull inspection, simulated in the Gazebo in a test tank, as well as a model for converting sonar data to PointCloud2. Subsequently, these models were used with sensor fusion algorithms (particles filters, extended and unscented Kalman filter) for error reduction in the simulated MCCR robot. The results highlighted the importance of properly configuring the fusion model parameters and the variability of the slippage impact on the odometry data. Therefore, an effective optimization of the fusion parameters for the MCCR robot will be possible after the odometry system characterization and validation of its model.

Keywords: sensor fusion; underwater robot; ROS; sonar; inertial navigation system; PointCloud2.

## LISTA DE QUADROS

Quadro 1 – Características para o mapa de testes com o MCCR .....	39
Quadro 2 – Dados e parâmetros para simulação dos sonares, baseados nos sensores utilizados no MCCR .....	42
Quadro 3 – Dados do INS utilizados neste projeto .....	46
Quadro 4 – Algoritmos, sensores e variáveis para fusão no experimento H1 .....	48
Quadro 5 – Algoritmos, sensores e variáveis para fusão no experimento H2 .....	51
Quadro 6 – Algoritmos, sensores e variáveis para fusão no experimento H3 .....	56
Quadro 7 – Definição das características dos mapas, ambientes e percursos .....	58
Quadro 8 – Erro máximo obtido pela odometria e por cada filtro em cada um dos percursos .....	63
Quadro 9 – Algoritmos, sensores, variáveis e parâmetros para fusão para experimento M1 .....	77
Quadro 10 – Valores de variância da mensagem de odometria utilizados em cada experimento. Experimento M1 com MCCR .....	78
Quadro 11 – Experimentos M2 com MCCR e a variação dos valores dos parâmetros para o AMCL .....	84
Quadro 12 – Resultados de experimentos M2 com erro na posição de 120 m.....	85



## LISTAS DE FIGURAS

Figura 1 - Esquemático de veículo robótico com rodas do .....	17
Figura 2 – Representação dos eixos 3D do veículo e seus sensores em relação ao eixo de referência inicial.....	18
Figura 3 – INS Rovins Nano IxBlue .....	19
Figura 4 – No visualizador Rviz, as linhas ou pontos vermelhos representam a informação do ambiente como percebido pelo laser. Um mapa pré-existente também é mostrado, sendo representado pelas cores cinza e preta. Preto representa as barreiras, cinza claro ambiente conhecido e cinza escuro, ambiente desconhecido.....	20
Figura 5 – Diagrama genérico para estimação de estados com predição e atualização a partir de dados medidos .....	23
Figura 6 – Aproximação Gaussiana do Filtro de Kalman .....	24
Figura 7 – Transformada unscented com pontos sigma .....	25
Figura 8 – Localização AMCL com comparação dos dados de laserscan (vermelho) e paredes (preto) .....	26
Figura 9 – Exemplo de ligação entre nós e tópicos no RQT Console no ROS.....	27
Figura 10 – Imagem de um console do RQT.....	28
Figura 11 – Imagem do robô Husky em ambiente para testes .....	29
Figura 12 – Robô (a) construído através de polígonos e junções. Veículo (b) Husky com suas peças identificadas por cores diferentes.....	30
Figura 13 – Husky, veículo robótico no ambiente simulado do Gazebo.....	30
Figura 14 – Pacotes que permitem a comunicação entre o ROS e o Gazebo .....	31
Figura 15 – MEC Combi Crawler Robot (MCCR) .....	35
Figura 16 – Tanque para testes de robótica, localizado no DFKI. Contém capacidade de 3,5 milhões de litros de água salgada.....	37
Figura 17 – Tanque placa com diversos obstáculos simulados.....	38
Figura 18 – Visão da parte inferior de um FPSO com detalhes de algumas partes do navio .....	39
Figura 19 – Representação do percurso do MCCR nos experimentos em vermelho e azul. Pontos brancos são obstáculos.....	39

Figura 20 – Mapa utilizado para estimar localização com AMCL. Traçados pretos e cinzas são considerados como barreiras para o algoritmo .....	40
Figura 21 – Formas de leitura de dados dos sonares a) Emissão de ondas acústicas em diferentes direções com um único feixe por leitura. b) Ângulo de emissão da onda. ....	41
Figura 22 – Exemplo de um AUV com um forward looking sonar, identificando estrutura submarina em simulação por imagem, a partir de ondas sonoras .....	41
Figura 23. Representação de imagem fornecida pela simulação.....	43
Figura 24 – Algoritmo para conversão de dados do sonar em PointCloud2 no eixo XYZ. As imagens representam os dados do sonar no cálculo .....	44
Figura 25 – Diagrama de bloco da modelagem desenvolvida para o sensor. Saída do valor do INS no diagrama vermelho .....	47
Figura 26 – Erro comparando a posição original do robô e os resultados obtidos por EKF (Vermelho), UKF (Azul), Odometria de Rodas (Verde) e AMCL (Roxo) .....	50
Figura 27 – Ambiente de simulação com o veículo robótico terrestre simulado no momento do impacto .....	52
Figura 28 – Gráfico de erro para odometria de Rodas (verde), GPS (azul) e EKF (vermelho) no eixo X para valor de covariância da matriz R, $\partial x^2$ igual a 0,2 .....	53
Figura 29 – Gráfico de erro para odometria de rodas (verde), GPS (azul) e EKF (vermelho) no eixo X para valor de covariância da matriz R, $\partial x^2$ igual a 0,02 .....	54
Figura 30 – Gráfico de erro para odometria de rodas (verde), GPS (azul) e EKF (vermelho) no eixo X para valor de covariância da matriz R, $\partial x^2$ igual a 0,002 .....	55
Figura 31 – Caminhos percorridos para os mapas H3.1 (a) e H3.2 (b). As linhas coloridas mostram o ground truth (linha preta), a localização estimada usando EKF (vermelho), UKF (roxo), AMCL (marrom) e odometria de rodas (verde) .....	60
Figura 32 – Ambiente estruturado (caixas brancas), caminho percorrido do robô (Ground Truth – Linha Preta) e localização estimada usando EKF (vermelho), UKF (Roxo), AMCL (Marrom e Laranja) e odometria de rodas (Verde) para o Mapa 1(a) e Mapa 2(b). ....	62
Figura 33 – Representação do MCCR com sonar no tanque no Gazebo com objetos para comparação. As linhas finas nas cores azul, vermelha e verde representam o eixo inicial do mapa. ....	64

Figura 34 – Visualização dos dados do sonar forward-looking no Rviz. a) Representação original pré-processamento. b) Representação pós-processamento no tipo de mensagem PointCloud2. Vermelho é uma representação do valor de intensidade zero. Roxo, maior intensidade .....	66
Figura 35 – Visualização dos dados do sonar mecânico de feixe único no Rviz. a) Representação original pré-processamento. b) Representação pós-processamento no tipo de mensagem PointCloud2. Vermelho é uma representação do valor de intensidade zero. Roxo, maior intensidade .....	68
Figura 36 – Elo de corrente similar utilizada no experimento .....	69
Figura 37 – Tanque utilizado para testes, simulando algumas estruturas do veículo MCCR .....	70
Figura 38 – Imagem original do sonar em PointCloud2 com a presença de eco e ruído....	71
Figura 39 – Visualização do objeto na posição 1 dentro do tanque pelo sonar, convertido em PointCloud2 (a) e por uma câmera mergulhada (b) .....	72
Figura 40 – Visualização do objeto na posição 2 dentro do tanque pelo sonar, convertido em PointCloud2 (a), e por uma câmera mergulhada (b) .....	73
Figura 41 – Visualização do objeto na Posição 3 dentro do tanque pelo sonar, convertido em PointCloud2 (a), e por uma câmera mergulhada (b) .....	74
Figura 42 – Visualização do objeto na posição 4 dentro do tanque pelo sonar, convertido em PointCloud2 (a), e por uma câmera mergulhada (b) .....	75
Figura 43 – Erro de posição modelado para sistema inercial de navegação. Linha em azul representa o possível erro do modelo, de acordo com a distância percorrida	76
Figura 44 – Erro associado à estimação de posição no eixo X para 3 rodadas do experimento M1.1. a) Rodada 1. b) Rodada 2. C) Rodada 3 .....	80
Figura 45 – Erro associado à estimação de posição no eixo X para 3 rodadas do experimento M1.2. a) Rodada 1. b) Rodada 2. C) Rodada 3 .....	82
Figura 46 – Erro associado à estimação de posição no eixo X do experimento M2.3. A linha verde representa o ponto de 120m . a) Gráfico com escala do erro maior. b) Gráfico com escala do erro menor. ....	86
Figura 47 – Erro associado à estimação de posição no eixo X do experimento M2.2. A linha verde representa o ponto de 120m. a) Gráfico com escala do erro maior. b) Gráfico com escala do erro menor. ....	87

Figura 48 – Erro associado à estimação de posição no eixo X do experimento M2.1. A linha verde representa o ponto de 120m. a) Gráfico com escala do erro maior. b) Gráfico com escala do erro menor. .... 88

## LISTAS DE SIGLAS E ABREVIATURAS

AMCL – Adaptive Monte Carlo Localization

ANP – Agência Nacional de Petróleo

AUV – Autonomous Underwater Vehicle

DFKI – Deutsches Forschungszentrum für Künstliche Intelligenz (German Research Center for Artificial Intelligence)

EKF – Extended Kalman Filter

FLS – Forward Looking Sonar

FPSO – Floating Production Storage and Offloading

GPS – Global Positioning System

GPU – Graphics Processing Units

IMU – Inertial Measurement System

INS – Inertial Navigation System

LBL – Long baseline acoustic positioning system

LiDAR – Light Detection and Ranging

MCCR – MEC Combi Crawler Robot

MSIS – Mechanical Scanning Imaging Sonar

PD&I – Pesquisa, Desenvolvimento e Inovação

PPGMCTI – Pós-graduação em Modelagem Computacional e Tecnologia Industrial

ROS – Robot Operating System

RQT – ROS Qt-based *framework*

SLAM – Simultaneous Localization and Mapping

UGV - Unmanned Ground Vehicle

UKF – Unscented Kalman Filter

USBL – Ultra-short baseline acoustic positioning system

XML – Extensible Markup Language

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	13
1.1 OBJETIVO.....	15
<b>2 REVISÃO DE LITERATURA</b> .....	16
2.1 SENSORES PARA LOCALIZAÇÃO .....	16
<b>2.1.1 Odometria de Rodas</b> .....	16
<b>2.1.2 IMU</b> .....	17
<b>2.1.3 INS</b> .....	18
<b>2.1.4 LaserScan</b> .....	19
<b>2.1.5 GPS</b> .....	20
2.2 FUSÃO DE SENSORES.....	21
<b>2.2.1 Filtros de Kalman</b> .....	21
<b>2.2.1.1 Filtro de Kalman estendido (Extended Kalman Filter ou EKF)</b> .....	24
<b>2.2.1.2 Filtro de Kalman unscented (Unscented Kalman Filter ou UKF)</b> .....	24
<b>2.2.2 Adaptive Monte Carlo Localization – Filtro de partículas</b> .....	25
2.3 ROS.....	26
<b>2.3.1 RQT Console</b> .....	28
2.4 GAZEBO.....	28
<b>3 MATERIAIS E MODELOS</b> .....	32
3.1 VEÍCULOS UTILIZADOS NOS EXPERIMENTOS .....	34
<b>3.1.1 Husky</b> .....	34
<b>3.1.2 MCCR</b> .....	35
3.3 MODELOS DESENVOLVIDOS PARA MCCR .....	36
<b>3.3.1 Criação do tanque para ambiente simulado de teste</b> .....	36
<b>3.3.2 Sonares</b> .....	40
<b>3.3.2.1 Conversão de dados de sonar para PointCloud2</b> .....	42
<b>3.3.3 INS</b> .....	45
<b>4 EXPERIMENTOS COM O HUSKY</b> .....	48
4.1 EXPERIMENTO H1: ALGORITMOS DE FUSÃO E AMBIENTE DE SIMULAÇÃO .....	48
<b>4.1.1 Resultados – Experimento H1</b> .....	49
4.2 EXPERIMENTO H2: CONFIGURAÇÃO DOS ALGORITMOS DE FUSÃO.....	51
<b>4.2.1 Resultados – Experimento H2</b> .....	52
4.3 EXPERIMENTO H3: INFLUÊNCIA DO AMBIENTE COM FILTRO DE PARTÍCULAS COM LIDAR .....	56
<b>4.3.1 Resultados – Experimento H3</b> .....	59

<b>5 EXPERIMENTOS COM O MCCR</b> .....	64
5.1 TESTES DE VERIFICAÇÃO DOS MODELOS DESENVOLVIDO PARA MCCR .....	64
<b>5.1.1 Conversão de dados de sonar para <i>PointCloud2</i></b> .....	64
<b>5.1.1.1 Testes em ambiente simulado</b> .....	64
5.1.1.1.1 Resultados .....	65
5.1.1.2 Testes em ambiente com imagens de um sensor real .....	68
5.1.1.2.1 Resultados .....	71
<b>5.1.2 Testes de modelo para INS</b> .....	76
<b>5.1.2.1 Resultados</b> .....	76
5.2 EXPERIMENTOS COM ALGORITMOS DE FUSÃO COM O MCCR .....	77
<b>5.2.1 Experimento M1: Utilização do modelo de INS com os algoritmos de filtros de Kalman</b> .....	77
<b>5.2.1.1 Resultados - Experimento M1</b> .....	78
<b>5.2.3 Experimento M2: Utilização dos dados do sonar para localização com o algoritmo de filtro de partículas AMCL</b> .....	83
<b>5.2.3.1 Resultados - Experimento M2</b> .....	84
<b>6 CONCLUSÃO</b> .....	90
REFERÊNCIAS .....	93

## 1 INTRODUÇÃO

O emprego de veículos móveis autônomos e teleoperados cresce em diversos tipos de aplicações, como a necessidade da realização de tarefas de inspeção em áreas não acessíveis e automatização de transportes de cargas. Adicionalmente, houve a evolução dos sistemas robóticos visando entender à necessidade de realizar uma localização com menos erros em diversos tipos de ambientes cada vez mais complexos, utilizando para tanto vários sensores medindo sua posição, velocidade, aceleração e orientação de forma absoluta ou relativa (KLANČAR *et al.*, 2017; MENNA *et al.*, 2017; PEI; KLEEMAN, 2017; STENTZ; HEBERT, 1995).

Em um ambiente subaquático, ondas eletromagnéticas apresentam maior atenuação da sua potência comparado a transmissão no ar. Como consequência, um Sistema de Localização Global (GPS), normalmente utilizado para aplicações externas, não funciona. Por causa disso, o uso de sensores inerciais acoplados ao robô podem ser ferramentas úteis para localização em ambiente subaquático, utilizando modelos matemáticos para identificar posição através da medição de outras grandezas como velocidade, orientação e aceleração (DONOVAN, 2012; SIEGWART; NOURBAKHS, 2011).

A odometria é um desses métodos, contendo medidores ligados diretamente às rodas ou ao eixo do motor do veículo, calculando a posição através do seu giro, orientação e velocidades. Há também o sensor conhecido como IMU (*Inertial Measurement Unit*) ou Unidade de Medição Inercial, contendo acelerômetros (aceleração) e giroscópios (orientação). No entanto, esses instrumentos podem sofrer de acúmulo de erro de posição e orientação, dependendo das manobras, da distância percorrida e do ambiente no qual está inserido (BARSHAN; DURRANT-WHYTE, 1995; KLANČAR *et al.*, 2017).

Outro equipamento utilizado para localização em ambiente subaquático é conhecido como *Inertial Navigation System* (INS), um sistema independente que identifica a posição de um veículo através de acelerômetros, giroscópios e medidores de velocidade associados. Esse dispositivo tem características semelhantes a um IMU, porém o INS contém um sistema embarcado que permite a fusão dos dados de todos os sensores ligados através de algoritmos proprietários otimizados para a aplicação. Esse equipamento, dependendo da sua configuração e calibração, gera dados com erro menor comparado ao



seus anteriormente citados, porém o erro ainda dependente da distância percorrida (LEE *et al.*, 2009; LI *et al.*, 2016).

Por esse motivo, algoritmos para fusão de sensores adicionais se tornam essenciais para reduzir o ruído e estimar um valor ótimo de localização com os dados obtidos. Esses algoritmos são os filtros de Kalman estendido e *unscented*, que utilizam dados com incerteza associada de vários sensores para estimar o valor de posição como variável de saída, a partir da previsão, comparação e atualização das variáveis das equações de um sistema inercial conhecido. Sendo assim, valores de posição e velocidade de um sensor com ruído intrínseco são comparados com um modelo e outros dados de aceleração, velocidade e posição vindos de um segundo ou terceiro sensor, obtendo uma estimativa de localização com erro menor (LEE *et al.*, 2009; LI *et al.*, 2016).

Adicionalmente, existem outros algoritmos que, através de informações do ambiente, como barreiras e objetos, permitem estimar a localização de sistemas robóticos. Para isso, é necessário utilizar sensores que permitam encontrar essas estruturas em variados tipos de ambientes, mais e menos estruturados. Para um ambiente fora da água é comum a utilização do LiDAR ou dispositivos que utilizam escaneamento a *laser*. No entanto, neste trabalho, houve a necessidade de realizar a estimação da localização em ambientes subaquáticos com precisão, onde há grandes desafios usando esse tipo de tecnologia, como a pouca quantidade de elementos no ambiente; logo, foi utilizado um sistema que contém um sonar de multifeixe e um de feixe único mecânico para reconhecer o ambiente, com capacidade de visualização do ambiente utilizando ondas sonoras. Além disso, foi desenvolvido um software que permite converter dados do sonar simulado para tipo *PointCloud2* e de *LaserScan*, podendo utilizar essa informação como entrada para o algoritmo AMCL.

Neste trabalho, esses algoritmos são utilizados com o *framework* ROS (*Robot Operating System*), que permite desenvolver, simular e programar algoritmos diversos, dados de sensores da área de robótica de um modo otimizado e com código aberto. Para a simulação de um ambiente 3D que represente a física e o mundo real foi utilizado o *Gazebo*, um software aberto que permite criar modelos de veículos, mapas e criar uma ponte entre o espaço onde o robô está inserido e os algoritmos já desenvolvidos, permitindo assim simular a mecânica real e o software a ser embarcado (GAZEBO, 2019; OPEN SOURCE ROBOTICS, 2019).

O veículo principal a ser investigado nesta dissertação é o MCCR (*MEC Combi Crawler Robot*), atualmente em desenvolvimento pelo SENAI CIMATEC e outras empresas parceiras, na realização de missões de limpeza e inspeção em navios de produção de petróleo. Para a realização dessas missões, esse veículo necessita de um sistema de localização adequado às dificuldades do ambiente submarino onde as missões serão realizadas.

Esta dissertação visa explorar, por simulação, sensores e técnicas de fusão de sensores mais comuns e sua adequação para a localização de um veículo submarino sobre rodas para inspeção industrial em ambiente limitado, desenvolvendo os modelos necessários.

## 1.1 OBJETIVO

O objetivo deste trabalho é desenvolver os modelos necessários dos sensores e ambiente submarino para a simulação do sistema de localização do robô MCCR que permitam a investigação dos algoritmos de fusão de sensores.

Os objetivos específicos são:

- Desenvolver um ambiente de simulação no Gazebo para o veículo Husky para investigar algoritmos de fusão de sensores disponíveis no ROS.
- Desenvolver ambiente de simulação submarino (tanque) para testes do MCCR no *Gazebo*.
- Desenvolver os modelos necessários para a simulação dos sensores do MCCR utilizando o ROS:
  - Sistema Inercial de Navegação (INS);
  - Conversão de dados de entrada de sonar simulado em mensagem do tipo *PointCloud2*.
- Testar os modelos desenvolvidos e seu funcionamento com os algoritmos de fusão de sensores no ROS.

## 2 REVISÃO DE LITERATURA

### 2.1 SENSORES PARA LOCALIZAÇÃO

Normalmente é possível classificar os sensores que geram dois tipos de dados para localização, relativos ou absolutos, que determinam como o erro pode se comportar em relação ao tempo, à distância percorrida ou à velocidade do veículo. De acordo com Zekavat e Buehrer (2011), um sistema com dados de localização absoluto ou global usa uma referência física externa com posição conhecida, sendo o GPS o exemplo mais conhecido. Já os sistemas com dados relativos utilizam informações do próprio movimento respectivo a um ponto fixo estático inicial, tendo como exemplos a odometria de rodas ou cálculo de posição através de acelerômetros e giroscópios.

Neste capítulo, alguns desses equipamentos e seus modelos serão apresentados com o objetivo de entender o comportamento do erro e a possibilidade de correção desse erro através dos algoritmos de alguma fusão de sensores.

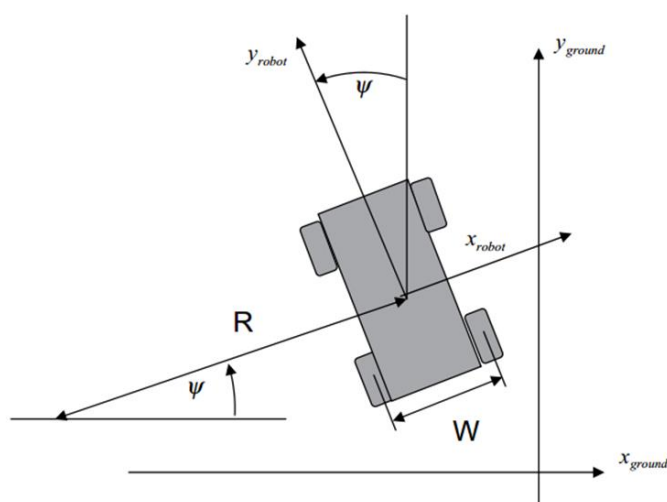
#### 2.1.1 Odometria de Rodas

Odometria de rodas é um dos métodos de localização em veículos móveis, que no caso diferencial, com rodas à direita e à esquerda da estrutura, permite estimar velocidade do giro de um motor ou rodas através de um *encoder*. Dessa forma, é possível estimar velocidade, orientação e conseqüentemente sua posição através de um modelo do sistema (JHA; KUMAR, 2015; PEI; KLEEMAN, 2017).

Esse método de localização pode ser considerado confiável quando não há influência do ambiente onde está inserido, porém, fenômenos de escorregamento, atrito ou ruídos produzem erros ao decorrer do percurso ou movimento gerado pelo robô. Esses fenômenos podem ser modelados, mas nem sempre representam completamente a realidade, causando um erro que aumenta a partir da distância percorrida e do tempo (CHENG; WANG, 2003; COOK, 2011). Por exemplo, um veículo pode utilizar um modelo diferencial de controle quando possui 2 pares de rodas que se movimentam independentemente.

Por exemplo, o robô com controle de movimentação diferencial apresentado na Figura 1 permite movimentos em duas direções, para frente e para trás, mais um giro em torno do próprio eixo, como demonstrado por Cook (2011). No entanto, podem ocorrer escorregamentos em relação à superfície ou falta de sincronia entre as rodas. Logo, como esse sistema de localização depende somente das suas referências internas, não é possível identificar se ele está se movimentando pelo caminho predeterminado.

Figura 1 - Esquemático de veículo robótico com rodas do modelo diferencial



Fonte: Klančar *et al.* (2017)

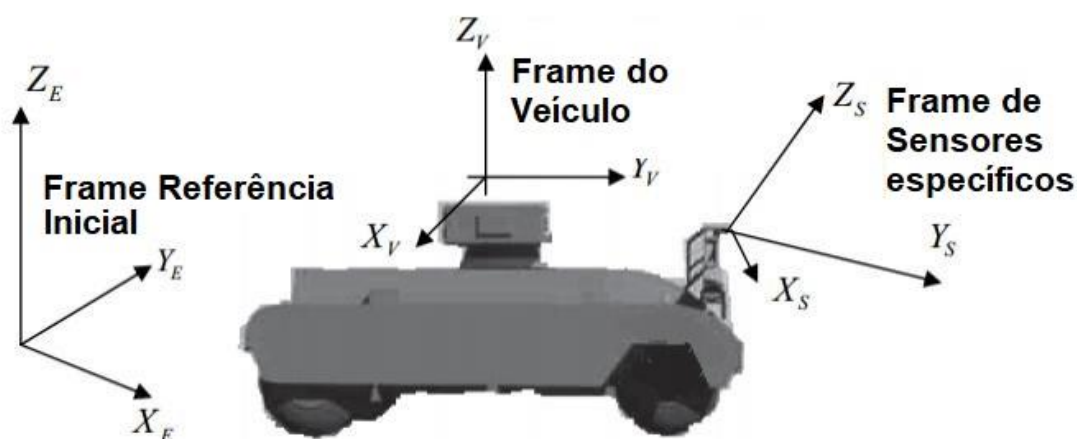
Dessa forma, para obter uma localização que se adeque aos requisitos de projeto, é necessário fundir dados de outros sensores que permitem identificar o aumento do erro causado por esses fenômenos não detectados pelo sistema de odometria (CHENG; WANG, 2003).

### 2.1.2 IMU

*Inertial Measurement Unit* ou “Unidade inercial de medição” é um dispositivo que utiliza acelerômetros, giroscópios e, às vezes, magnetômetros em seu sistema com o objetivo de medir a aceleração e orientação de forma absoluta. No entanto, o IMU não permite identificar posição de modo absoluto, necessitando usar os dados de aceleração junto a outros sensores através da fusão do GPS e da odometria. Dessa forma, é possível utilizá-lo para estimar a localização do veículo, através do reconhecimento dos seus

movimentos, reduzindo o erro gerado por interferências do ambiente, principalmente relacionadas a movimentos bruscos (COOK, 2011; KLANČAR *et al.*, 2017). Esse dispositivo permite relacionar os “frames” ou eixos 3D do veículo e uma orientação global do ambiente, como é possível visualizar no exemplo da Figura 2, contribuindo para a estimativa da localização.

Figura 2 – Representação dos eixos 3D do veículo e seus sensores em relação ao eixo de referência inicial



Fonte: Cook (2011)

### 2.1.3 INS

O sistema Inercial de Navegação ou INS contém um IMU e outros sensores internos, que permitem acoplar diversos sensores para medir outras variáveis, como velocidade absoluta e profundidade quando utilizado submerso. Esses sistemas fazem uso da técnica de *dead reckoning*, onde a posição e orientação são estimadas a partir de dados de aceleração e movimentação relativos a uma posição inicial definida (KLANČAR *et al.*, 2017). Semelhante ao IMU, o INS contém também acelerômetros, giroscópios e magnetômetros, porém os dados obtidos desses sensores são processados internamente de forma a estimar valores de posição.

Apesar de ser bastante utilizado como uma alternativa para localização quando não há informações advindas de sensores para a localização global, o INS ainda sofre a influência da acumulação do erro sistemático, escorregamento ou má calibração, fazendo o erro médio na posição e orientação crescer com o aumento da distância percorrida. Um dos modelos, presentes em diversas aplicações, é o sistema inercial de navegação *Rovins*

*Nano*, fornecido pelo fabricante *IxBlue*, visualizado na Figura 3 (IXBLUE, 2020; ZHANG *et al.*, 2009). Esse modelo foi adquirido para o robô MCCR e será avaliado nesta dissertação.

Figura 3 – INS *Rovins Nano IxBlue*

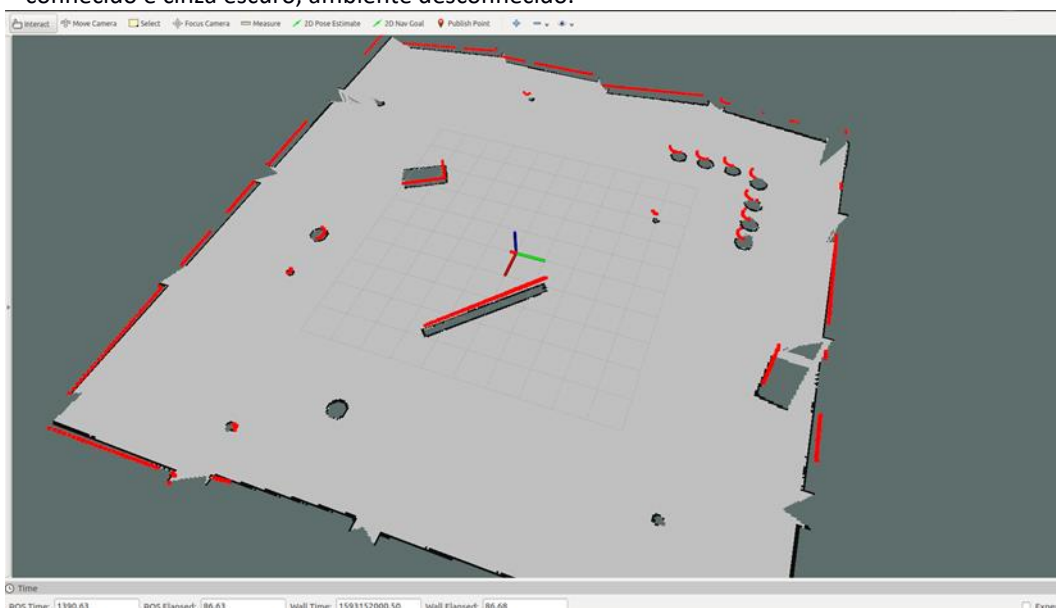


Fonte: IXBLUE (2020)

#### 2.1.4 LaserScan

A localização de veículo autônomo relativo a um mapa fixo é um outro método utilizado que permite identificar posição de uma forma mais precisa em ambientes internos com a ajuda de sensores que percebem o ambiente ao seu redor. O equipamento mais utilizado para essa atividade de reconhecimento 2D ou 3D do ambiente é o *LiDAR*, acoplado ao veículo, que usa escaneamento de feixes de *laser* (*LaserScan*) para identificar a distância do robô a alguns objetos, como apresentado na Figura 4. Através do uso desse sensor, um mapa pré-existente e de alguns algoritmos como o filtro de partículas, é possível criar uma estimativa absoluta de localização no ambiente inserido, obtendo resultados mais precisos que com os outros apresentados anterior. No entanto, em ambientes externos com pouca informação, sensores ou um mapa menos definido podem causar um aumento de incertezas. Por isso, é interesse não descartar todos os dados de posição adquiridos por outros sensores (CHONG *et al.*, 2013).

Figura 4 – No visualizador Rviz, as linhas ou pontos vermelhos representam a informação do ambiente como percebido pelo laser. Um mapa pré-existente também é mostrado, sendo representado pelas cores cinza e preta. Preto representa as barreiras, cinza claro ambiente conhecido e cinza escuro, ambiente desconhecido.



Fonte: autoria própria

Outra solução utilizada em alguns casos em que não existe mapa pré-existente, é a utilização de um algoritmo de localização e mapeamento simultâneo, conhecido como SLAM, não necessitando de um mapa pré-existente. Porém, essa solução ainda depende de referências do ambiente e de outros sensores para encontrar um frame inicial de referência (HASHIKAWA; MORIOKA, 2011; ZHANG *et al.*, 2014).

### 2.1.5 GPS

O GPS é um sensor que fornece dados em tempo real com posição absoluta, orientação, velocidade e altitude através de conexão com antenas ou satélites espalhados pelo globo. Isso é realizado a partir de um método de triangulação do tempo de resposta entre as antenas de referência. No entanto, uma das limitações do GPS é o não funcionamento em ambientes internos, principalmente quando há objetos ou paredes que impedem a comunicação ou em ambientes submarinos (COOK, 2011; ROSE *et al.*, 2014).

Para superar essas limitações, uma nova percepção absoluta pode ser definida através de uma rede de sensores no ambiente interno ou de um sistema de percepção do ambiente. Dessa forma, é possível reduzir o erro associado aos sensores e solucionar problemas, como o uso de veículos em ambientes fechados com a presença de pessoas

(SIEGWART, ROLAND; NOURBAKHSH, ILLAH R.; SCARAMUZZA, 2011; SIEGWART; NOURBAKHSH, 2011).

## 2.2 FUSÃO DE SENSORES

Localização para veículos autônomos significa adquirir dados de posição e orientação em relação a um sistema de coordenadas baseado num modelo e sensores para localização (CHENG; WANG, 2003).

Sensores desempenham um papel fundamental na determinação da localização automática de um determinado veículo, cada um com as suas incertezas e ruídos associados. Como discutido anteriormente, é possível estimar a posição e orientação do robô através da odometria de rodas, com medidores que identificam o seu giro e velocidade, e sensores IMU (*Inertial Measurement Unit*) ou Unidade de Medição Inercial, com acelerômetros (aceleração) e giroscópios (orientação). Cada sensor utilizado tem suas especificidades e incertezas associadas, definidas através de suas características físicas, do ambiente e como é utilizado em determinadas situações. Algoritmos de fusão de sensores são utilizados para reduzir o ruído das informações obtidas e estimar um valor ótimo de localização, a partir de dados obtidos de duas ou mais fontes (LEE *et al.*, 2009; ZHANG *et al.*, 2014).

Esta dissertação aborda a fusão de sensores utilizados comercialmente para atividades terrestres e submarinas. Três algoritmos são abordados, dois deles baseados no modelo de filtro de Kalman (o filtro Kalman estendido e o *unscented*) e o filtro de partículas (AMCL).

### 2.2.1 Filtros de Kalman

*Kalman filter* ou filtro de Kalman é conhecido como um dos mais importantes métodos de estimação ótima de variáveis para sistemas lineares, utilizando teorias dos filtros Bayesianos e etapas de previsão de modelos e atualização dos estados como apresentado na Figura 5. Utilizando, por exemplo, valores de posição e velocidade de um sensor com ruído intrínseco são comparados com um modelo e outros dados de



aceleração, velocidade e posição vindos de um segundo ou terceiro sensor, obtendo uma informação mais precisa (LEE *et al.*, 2009; LI *et al.*, 2016).

Para encontrar um valor estimado de posição e orientação a partir de determinadas variáveis inerciais, um modelo do sistema de equações pode ser definido a partir de vetores e matrizes que se relacionam, como demonstrado nas equações (1) e (2). (KLANČAR *et al.*, 2017; SIEGWART, ROLAND; NOURBAKHS, ILLAH R.; SCARAMUZZA, 2011; SIEGWART; NOURBAKHS, 2011; THRUN, 2000).

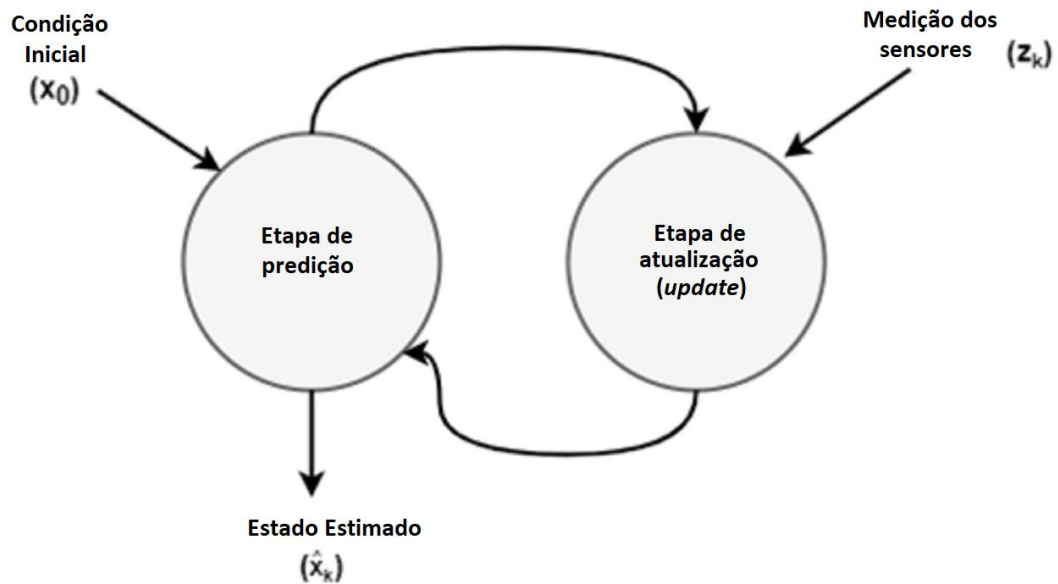
$$x_k = Ax_{k-1} + Bu_k + w_k \quad (1)$$

$$z_k = Cx_k + v_k + \delta_t \quad (2)$$

Sendo,  $x_k$  o vetor de estados atual,  $x_{k-1}$  o vetor de estados anterior,  $A$  a matriz de constantes para dimensionar estados a partir do estado anterior  $x_{k-1}$ ,  $u_k$  - vetor de comandos de controle,  $B$  - matriz de constantes para dimensionar estados, a partir do comando de controle  $u_k$ ,  $w_k$  - vetor de ruído de estados,  $z_k$  - vetor com valores observados da medição,  $v_k$  - vetor de ruído de medição e  $C$  - matriz de observação constante para dimensionar os valores medidos a partir do estado atual e  $\delta_t$  - ruído de medição dos sensores.

A partir desse modelo e suas covariâncias, é possível medir e prever valores estimados dos estados  $x_k$ , equação 3, iterativamente. A saída estimada depende de um ganho de Kalman  $G_k$ , que identifica qual sensor é mais confiável, baseado na covariância da medição  $R$  de cada sensor apresentado na equação 4 e  $P_k$ , como uma matriz de covariância intrínseca do processo.  $R$  é a matriz essencial na definição dos pesos de importância de cada sensor, com suas covariâncias  $\partial_x$  de posição e  $\partial_{vel}$  de velocidade, para definição da matriz de ganho de Kalman  $G_k$ , equação 5, sendo o foco das simulações avaliadas (THRUN, 2000).

Figura 5 – Diagrama genérico para estimação de estados com predição e atualização a partir de dados medidos



Fonte: adaptado de LABBE (2019)

$$\begin{matrix} x_k \\ z_k \end{matrix} = \begin{bmatrix} \textit{dist\~{a}ncia} \\ \textit{velocidade} \\ \textit{acelera\~{c}\~{a}o} \\ \textit{orienta\~{c}\~{a}o} \end{bmatrix} \quad (3)$$

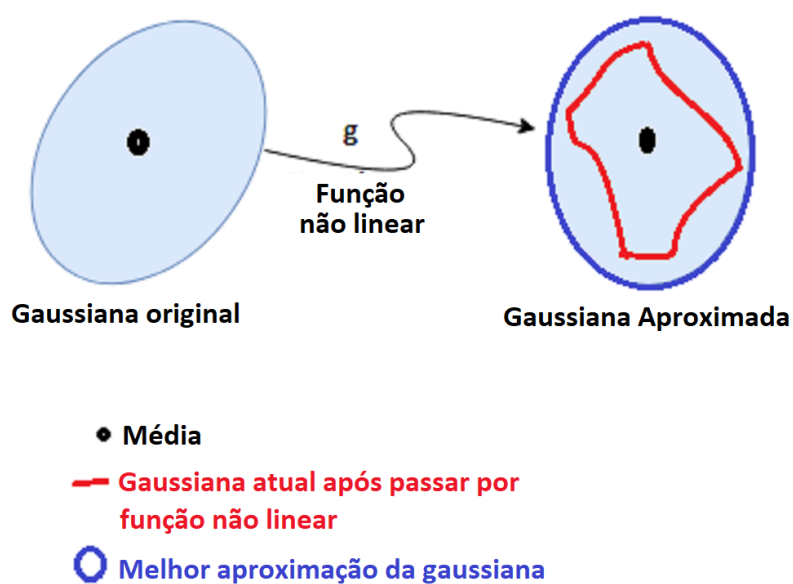
$$R = \begin{bmatrix} \partial_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \partial_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \partial_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \partial_{velx}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \partial_{vely}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \partial_{velz}^2 \end{bmatrix} \quad (4)$$

$$G_k = P_k C^T (C P_k C^T + R)^{-1} \quad (5)$$

### 2.2.1.1 Filtro de Kalman estendido (Extended Kalman Filter ou EKF)

O filtro de Kalman simples contém algumas limitações que o impedem de ser utilizado em diversas aplicações práticas, como por exemplo: somente pode ser aplicado em sistemas lineares, e o ruído a ser reduzido necessita ser gaussiano. Para sistemas não lineares, o filtro de Kalman estendido pode ser aplicado ao utilizar técnicas de linearização de primeira ordem como expansão de Taylor na estimação das equações de estados de todo o sistema como na Figura 6, permitindo encontrar valores mais próximos do real em relação a um ponto de referência inicial. Adicionalmente, essas equações precisam estar próximas de um sistema localmente linear e contínuo para um resultado mais eficiente (KLANČAR *et al.*, 2017; LI *et al.*, 2016).

Figura 6 – Aproximação Gaussiana do Filtro de Kalman



Fonte: adaptado de Chadha (2018)

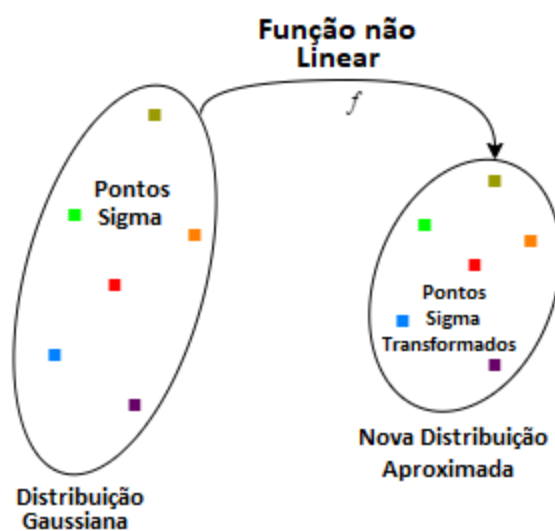
### 2.2.1.2 Filtro de Kalman unscented (Unscented Kalman Filter ou UKF)

Um outro método para lidar com sistemas não lineares é o filtro de Kalman *unscented*, que utiliza *unscented transform* com a premissa de selecionar pontos amostrados (sigma) de acordo com a média e covariância de uma distribuição aproximadamente gaussiana, como apresentado na Figura 7. A aplicação da transformada possibilita calcular distribuição probabilística aproximada para obtenção de uma nova

matriz de estados e ganhos. Esses novos dados servirão para predição e atualização do filtro de Kalman (CHADHA, 2018; LI *et al.*, 2016; S.J. JULIER; J.K. UHLMANN, 2004; UYULAN; ERGUZEL; ARSLAN, 2018).

Esse método permite encontrar resultados mais precisos em sistemas de ordem maiores, por ser mais próximo das características estatísticas do sistema, além de ter um menor custo computacional por não necessitar realizar o cálculo derivativo da matriz Jacobiana. Por exemplo, o escorregamento das rodas pode gerar um comportamento não linear do modelo diferencial, logo é interessante utilizar essa técnica para linearização dos dados para serem aplicados ao filtro de Kalman (CHADHA, 2018; LI *et al.*, 2016).

Figura 7 – Transformada *unscented* com pontos sigma



Fonte: autoria própria

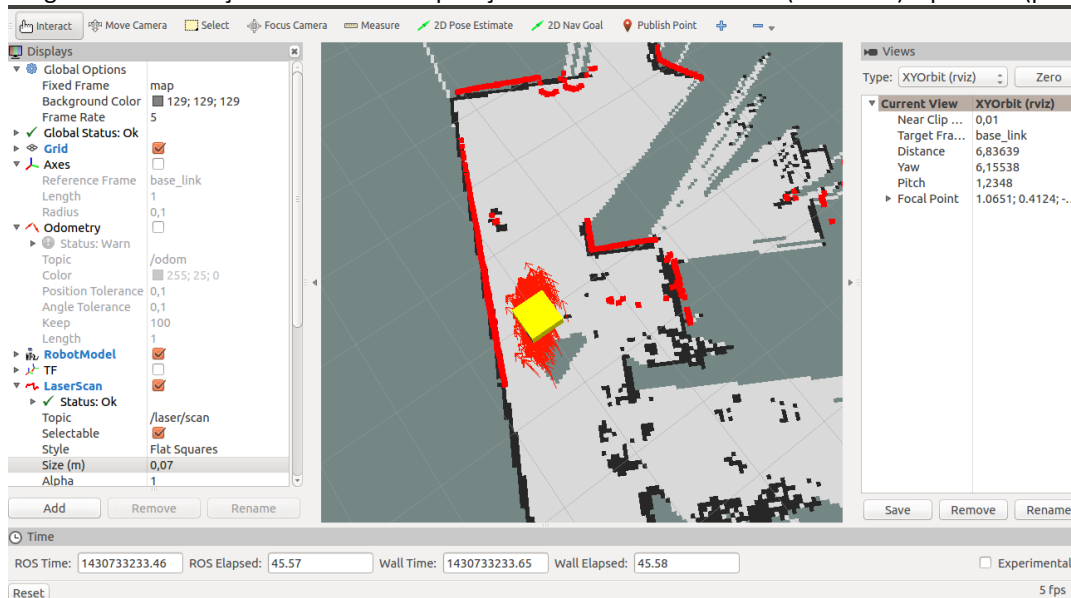
### 2.2.2 Adaptive Monte Carlo Localization – Filtro de partículas

O *Adaptive Monte Carlo Localization* (AMCL) é um algoritmo que utiliza o filtro de partículas para realizar cruzamento de dados coletados por um sensor de percepção, como, por exemplo, o *laserscan*, com os dados de odometria de rodas e um mapa pré-carregado para calcular a distância entre eles e os recursos detectados. Os dados de *laserscan* utilizadas são normalmente originários de modelos de LiDAR (ZHANG; ZAPATA; LÉPINAY, 2012). Como ilustrado na Figura 8, as partículas geradas são alocadas aleatoriamente e atualizadas iterativamente, estimando a posição e a orientação do robô (THRUN, 2000).

O algoritmo analisa os dados de reflexão da luz laser para identificar a posição de barreiras no ambiente e confronta com os elementos presentes no mapa utilizado como

por exemplo, paredes e objetos. A partir disso, há uma atualização da posição e atribuição de pesos às partículas. Essa combinação ou *match* com o mapa permite corrigir a posição do veículo baseado no ambiente e um *frame* de referência inicial, normalmente localizado por outro sensor, como odometria. Adicionalmente, o algoritmo precisa que as superfícies e barreiras permaneçam constantes, de modo que permitam realizar a comparação entre os mapas e os dados do *laser*. Quando isso não é possível, o cálculo pode ser prejudicado em boa parte dos casos (CHONG *et al.*, 2013).

Figura 8 – Localização AMCL com comparação dos dados de *laserscan* (vermelho) e paredes (preto)



Fonte: ROS ANSWER - FELIXWATZLAWIK (2015)

## 2.3 ROS

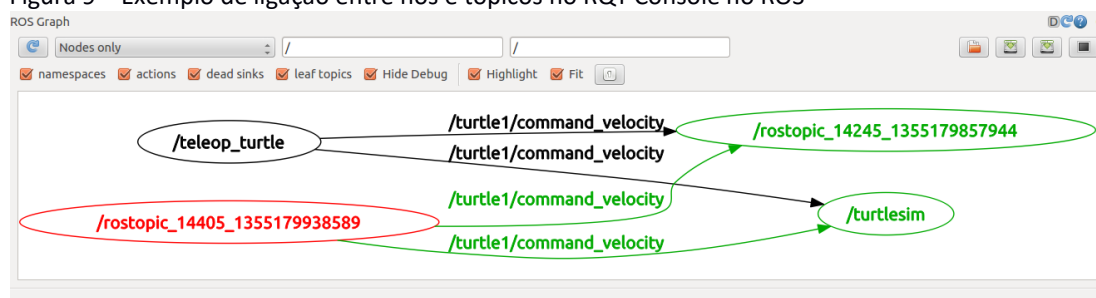
*Robot Operating System*, ou somente ROS, é um *framework* para robótica, desenvolvido pela *Open Robotics*, que possibilita gerenciar pacotes, códigos executáveis, bibliotecas, ferramentas e rede de mensagens de uma forma robusta para ser utilizado em plataformas, como o Linux, porém mantendo uma forma simplificada e aberta de programação. (OPEN SOURCE ROBOTICS, 2019).

Um sistema para robótica sempre foi um desafio devido às diversas variáveis envolvidas, como o grande número de atividades que um robô pode realizar. Cada um era desenvolvido com um objetivo individual e por causa disso havia uma dificuldade de integração entre as diversas bibliotecas e aplicações. Por causa disso, o ROS começou a ser

desenvolvido de forma colaborativa em diversas universidades e empresas espalhadas pelo mundo, havendo constantes atualizações, ferramentas, aplicações e bibliotecas diversas. Como exemplo, existem pacotes com código para o controle dos motores de um veículo terrestre sobre rodas, usam dados de sensores para localização, permitem identificar obstáculos através de câmeras e realizam um controle de segurança de equipamentos elétricos e sensores. Devido a sua larga distribuição aberta, o ROS está sendo utilizado e aprimorado por diversas pessoas e empresas ao redor do mundo (OPEN SOURCE ROBOTICS, 2019).

O ROS utiliza um sistema de nós, tópicos, mensagens e serviços que permitem gerenciar uma rede de comunicação entre os diversos dados obtidos de uma simulação ou de um veículo real. Como exemplo, é possível utilizar um nó para obter dados de um sensor, ligá-los a um tópico que será utilizado na forma de uma mensagem específica como um conjunto de caracteres ou números. Esse tópico pode ser utilizado por outro nó, permitindo que seus dados sejam processados e utilizados para criar um outro tópico que realizará essas ações. Um exemplo de rede que liga tópicos e nós pode ser visualizado na Figura 9. A estrutura do ROS utiliza um agrupamento de pacotes com códigos executáveis e bibliotecas, podendo utilizar como linguagens C++ e Python para desenvolvimento e XML e outras linguagens de marcação para configuração de parâmetros e execução contínua dos nós, como, por exemplo, arquivos de *launch* para executar sequência de nós e configurar parâmetros (OPEN SOURCE ROBOTICS, 2018).

Figura 9 – Exemplo de ligação entre nós e tópicos no RQT Console no ROS

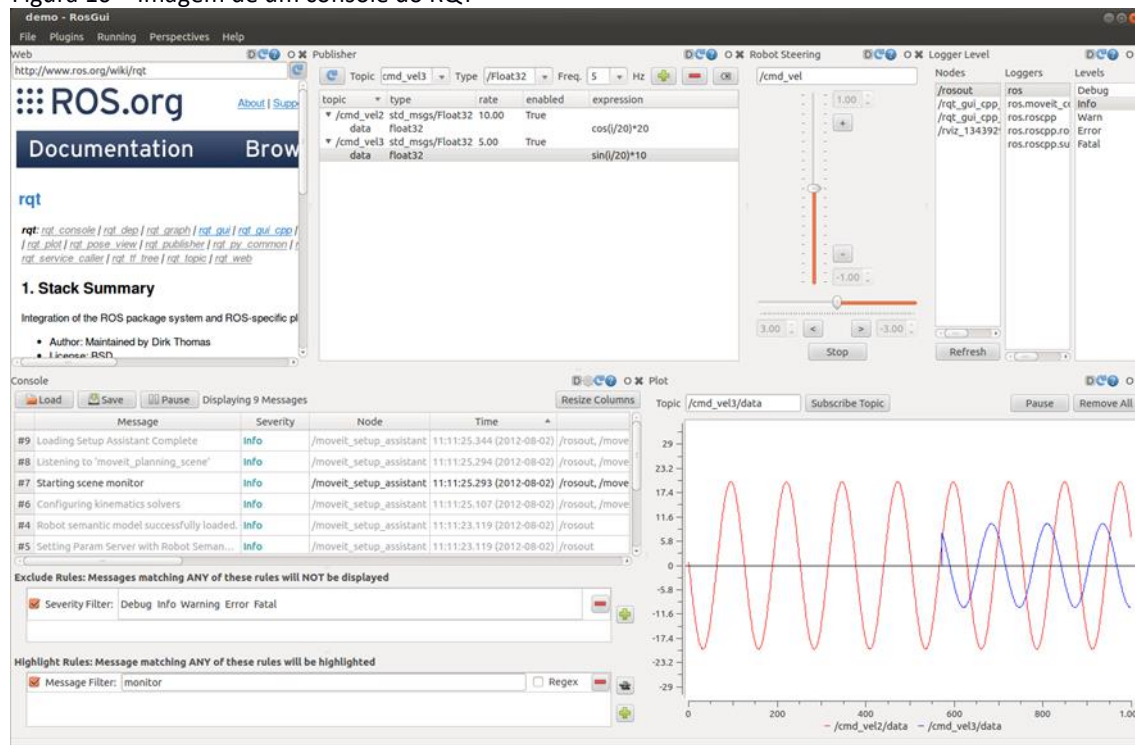


Fonte: OPEN SOURCE ROBOTIC FUNDATION (2017)

### 2.3.1 RQT Console

O console RQT é uma ferramenta desenvolvida para ROS que permite a visualização e a interação com dados dos nós, através de diversos *plugins* e módulos. Os *plugins* mais comuns servem para a publicação e visualização numérica e gráfica de dados adquiridos através de mensagens, tópicos, acesso a dados e visualização da rede de gerenciamento entre nós, como apresentado na Figura 10 (OPEN SOURCE ROBOTIC FOUNDATION, 2016).

Figura 10 – Imagem de um console do RQT



Fonte: OPEN SOURCE ROBOTIC FOUNDATION (2016)

### 2.4 GAZEBO

O *Gazebo* começou a ser desenvolvido em 2002 por um grupo de estudantes da *University of Southern California* com o objetivo de criar um software que simulasse veículos robóticos e ambientes abertos e fechados em variadas circunstâncias. Após alguns anos de desenvolvimento inicial junto à comunidade *open-source*, a *Open Robotics*, que já desenvolvia o ROS, tornou-se a principal mantenedora do projeto, apoiando financeiramente e, ao mesmo tempo, permitindo que a comunidade ativa e diversa continuasse a desenvolver para o *Gazebo*. Dessa forma, o *Gazebo* se tornou o principal simulador utilizado e bastante aceito pela comunidade ROS. Ele permite desenvolver

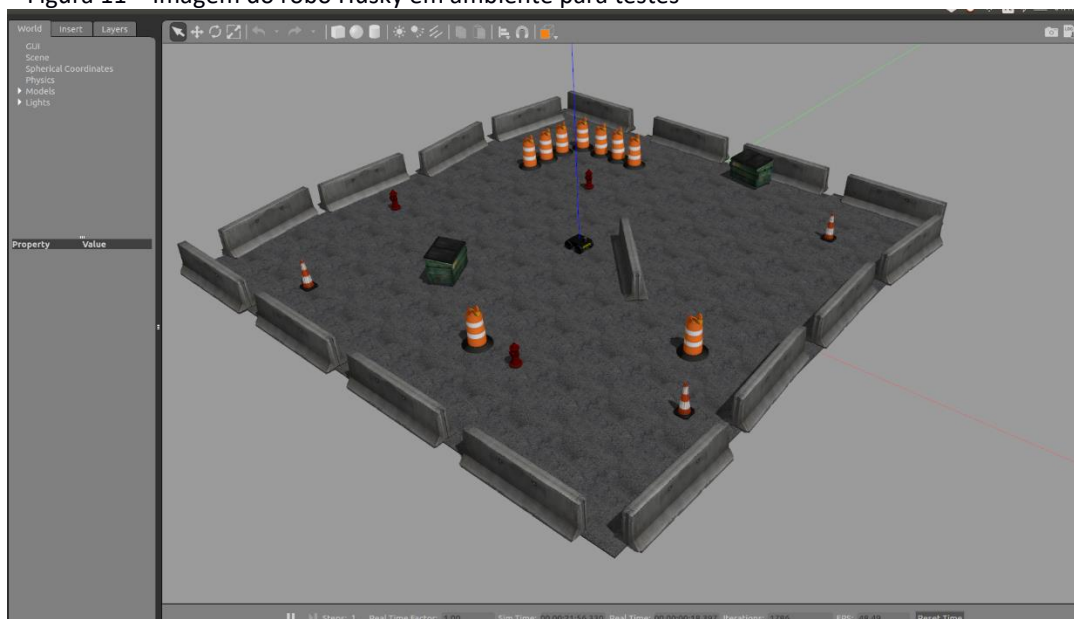
algoritmos, projetar robôs complexos, desenvolver testes, treinar sistemas de inteligência artificial em ambientes complexos internos e externos próximos da realidade. O simulador permite criar ambientes e veículos diversos através de um motor gráfico de alta qualidade e interface gráfica para a interação entre os objetos (GAZEBO, 2019; OPEN SOURCE ROBOTICS, 2019).

As principais funções do simulador são:

- criar ambientes com física predeterminada, adicionando objetos baseados em modelos 3D gerados anteriormente.
- desenvolvimento de *plugins* e bibliotecas para o veículo, adicionando sensores, sistemas de controle do veículo e do ambiente, como, por exemplo, simulação de câmeras e controle de rodas;
- desenvolver modelos de veículos ou importar modelos desenvolvidos por outros softwares de simulação 3D;
- fazer e codificar ligações entre as diversas partes de um sistema robótico, como movimentação de juntas e simulação de atuadores;
- realizar comandos através de linha de código.

A Figura 11 mostra a interface gráfica do *Gazebo* e do ambiente desenvolvido para testes. Esse modelo, por exemplo, contém um chão com atrito, diversos obstáculos e física predeterminada para um ambiente terrestre aberto.

Figura 11 – Imagem do robô Husky em ambiente para testes

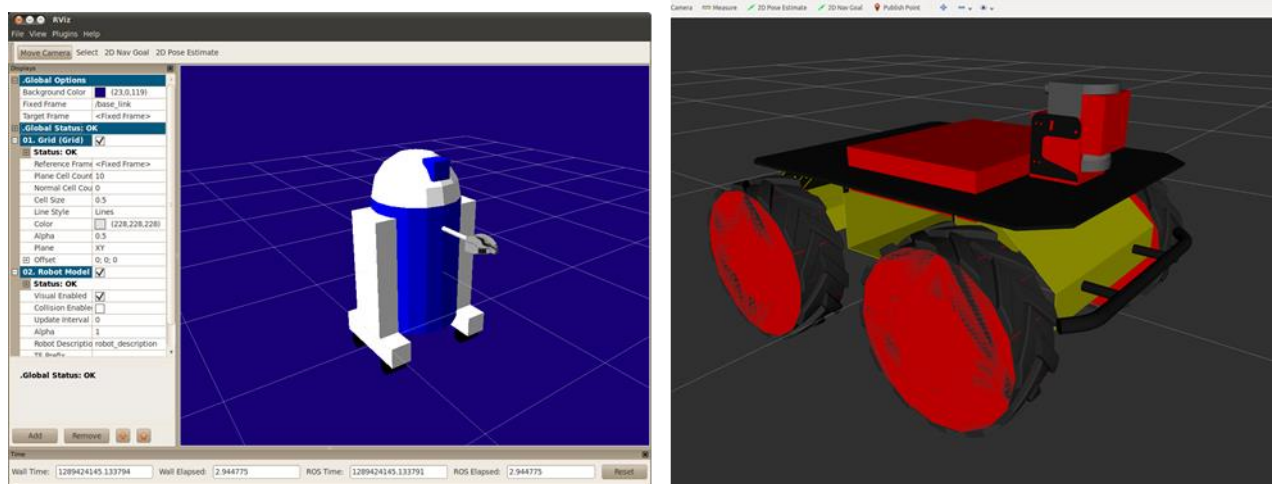


Fonte: Simulação *Gazebo*. autoria própria



Os veículos podem ser apresentados através de modelos que devem se comunicar através de juntas e atuadores, como o movimento de uma roda. Por exemplo, a Figura 12 apresenta o modelo 3D de um robô, construído através de polígonos e um veículo terrestre sobre rodas.

Figura 12 – Robô (a) construído através de polígonos e junções. Veículo (b) Husky com suas peças identificadas por cores diferentes.



Fonte: OPEN SOURCE ROBOTICS (2011)

Após a criação e a importação da estrutura, é possível visualizar e interagir com o modelo no ambiente de simulação, como ilustrado na Figura 13.

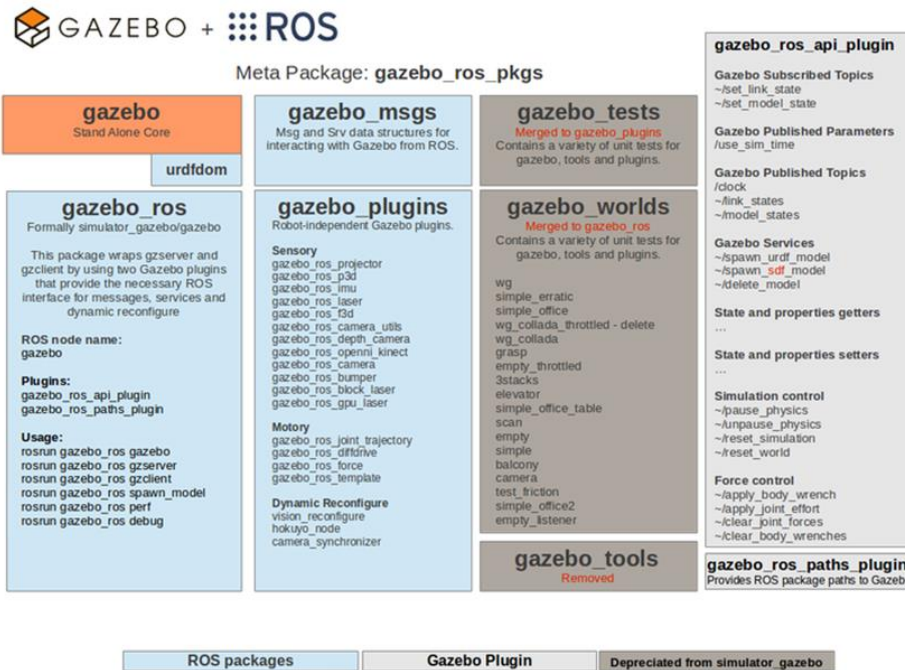
Figura 13 – Husky, veículo robótico no ambiente simulado do Gazebo



Fonte: Simulação Gazebo. autoria própria

Toda a comunicação entre o *Gazebo* e o ROS é realizada através de pacotes desenvolvidos especificamente, que criam uma ponte para a comunicação entre ambos os sistemas. Dessa forma, é possível obter dados da simulação do *Gazebo*, como de sensores, no ROS, permitindo utilizar essa informação e processá-la para posteriormente utilizá-la (GAZEBO, 2019; OPEN SOURCE ROBOTICS, 2019). A Figura 14 mostra alguns desses pacotes que permitem a comunicação entre os dois.

Figura 14 – Pacotes que permitem a comunicação entre o ROS e o *Gazebo*



Fonte: GAZEBO.SIM.ORG (2014)

### 3 MATERIAIS E MODELOS

Para esse trabalho, foram utilizados dois softwares principais, o framework ROS na distribuição *Kinetic e Gazebo* na versão 7.16.0, para criar o robô e o ambiente simulados, com *Ubuntu 16.04 Xenial* como sistema operacional principal de programação. Como apresentado anteriormente, o ROS é um framework de robótica que contém centenas de bibliotecas que permitem processar informações, simular e controlar sistemas. Ele também permite fazer a conexão com o simulador Gazebo, em que é possível montar robôs do zero com eixos e motores. Duas linguagens de programação são utilizadas, C++ e Python, para a criação de códigos executáveis e bibliotecas. XML também é utilizado para a criação de códigos com linguagem de marcação para a execução de cada nó separado e configurações. Para visualização gráfica dos dados, é utilizada a ferramenta RQT e o *plugin Multiplot* (KAESTNER, 2016).

Inicialmente, foram explorados os recursos do ROS e Gazebo para a simulação de um sistema de localização usando robôs disponíveis no ROS. Para isso foram desenvolvidos um ambiente de simulação no Gazebo para um veículo *Husky* e uma série de experimentos para o entendimento dos algoritmos de fusão de sensores inerciais de localização, um IMU (Inertial Measurement System) e dados de odometria de rodas. Para isso, foi utilizado o pacote *robot\_localization* (MOORE, 2018), uma coleção de bibliotecas que contém algoritmos de filtro de Kalman estendido, *unscented* e filtro de partículas (AMCL).

Nas etapas seguintes, foi necessário desenvolver alguns módulos que permitiriam realizar experimentos para um novo veículo a ser testado, o MCCR, em desenvolvimento pelo núcleo de robótica do SENAI CIMATEC. Como uma primeira etapa, foi desenvolvido um ambiente submarino no *Gazebo* para a simulação dos primeiros testes do robô, inspirando-se no tanque existente no DFKI, onde seriam executados os primeiros testes do projeto com odometria e outros sensores inerciais. A seguir foi desenvolvido um modelo do sensor inercial de navegação (INS) para robô submarino baseado no modelo comercial *Rovins Nano*, a partir dos parâmetros disponibilizados pelo fabricante *iXBlue*, disponibilizando uma mensagem do tipo odometria para o ROS. Esses modelos foram utilizados para criar uma referência de localização relativa, gerada a partir da estimação de localização a partir dos filtros de Kalman estendido e *unscented* no MCCR, semelhante ao

realizado com o Husky, porém variando parâmetros de covariância da mensagem do sensor e dos algoritmos.

Para uma terceira etapa, foi verificada a necessidade de uma abordagem que utilizasse o ambiente como referência para melhorar a estimativa da localização com sensores inerciais, devido ao seu erro crescente com a distância percorrida e possíveis escorregamentos ao lidar com obstáculos. Como o ambiente é submarino, a utilização de LiDAR ou *LaserScan* para o reconhecimento do ambiente apresenta limitações no alcance devido à absorção da luz pela água. Para superar essa dificuldade, sonares podem ser utilizados para o reconhecimento do ambiente e estimar localização, corrigindo os erros da fusão de sensores inerciais com os de odometria.

Para isso, foram utilizados 2 módulos simulados para o veículo MCCR, um para sonar *Forward Looking* de Multifeixe (FLS), produzindo imagens de uma área à frente do veículo a uma frequência de 0,5 segundo. Enquanto isso, o outro dispositivo é um sonar mecânico de feixe único na parte traseira do veículo, o qual contém um mecanismo que gira ao redor do seu próprio eixo, reconstruindo uma imagem após determinado tempo de giro. Eles simulam o funcionamento desses dois dispositivos e geram um tipo de mensagem específico definido no pacote de simulação de sonar, apresentado por Cerqueira *et al.* (2017) e uma imagem, que posteriormente são convertidos para formato *PointCloud2* por um algoritmo para a realização de experimentos para estimar a localização, utilizando o ambiente como referência. A motivação dessa atividade foi utilizar dados gerados pelo sonar em um formato compatível para ser utilizado como entrada para o algoritmo de AMCL nos experimentos. Dessa forma, foi possível estimar localização a partir de características do ambiente e posterior comparação com os algoritmos de filtro de Kalman.

Assim, para este trabalho, foram desenvolvidos os seguintes módulos e ambientes para a utilização em experimentos.

- Tanque de testes para instalação do MCCR
- Modelo de INS
- Algoritmo de conversão da mensagem de sonar para *PointCloud2*

## 3.1 VEÍCULOS UTILIZADOS NOS EXPERIMENTOS

### 3.1.1 Husky

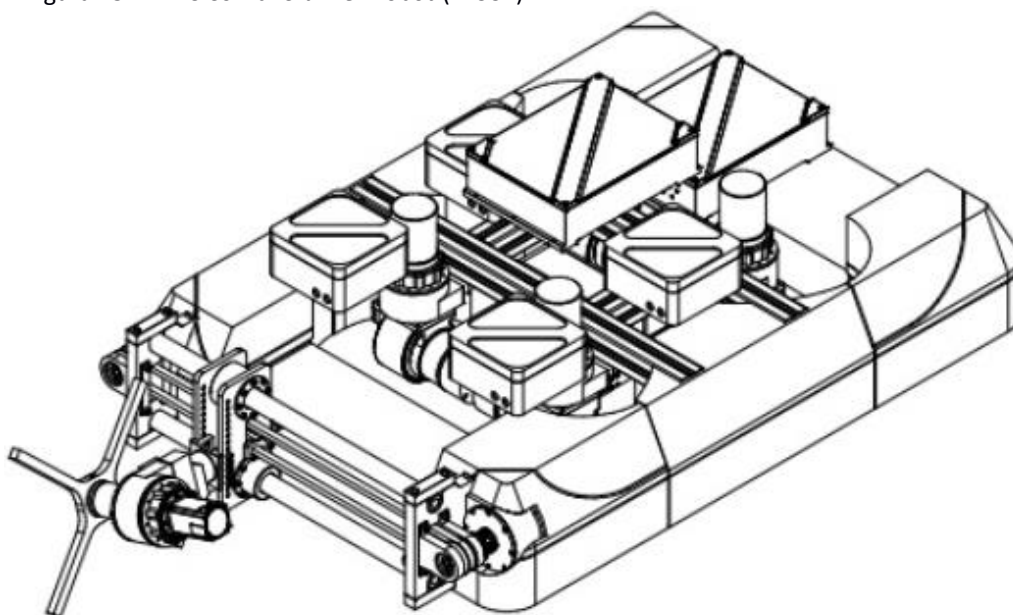
Um exemplo de robô que utiliza a técnica de odometria é o Husky, uma plataforma de desenvolvimento para robótica, no formato de um veículo terrestre não tripulado (UGV) com rodas configuradas no modelo direcional. Para esse trabalho, ele foi escolhido inicialmente devido à fácil aprendizagem para simulação no Gazebo, pelo suporte da comunidade *Open Source* do ROS e por ser um robô com alta capacidade de carga e de customização de dispositivos. Além dos sensores de odometria, o Husky contém também um IMU com acelerômetro, giroscópios de 3 eixos e magnetômetro, GPS e LiDAR 2D da série MS1xx (CLEAR PATH, 2018).

No caso do Husky, também foram utilizados algoritmos e referências do ambiente, como mapas, barreiras e pontos de referências combinados com o uso de câmeras e escaneamento por *laser* ou LiDAR. Adicionalmente, o veículo *Husky* foi escolhido devido à sua disponibilidade para simulação e por ter algumas características semelhantes ao MCCR, como o uso da técnica de odometria e a capacidade do Husky em usar sensores para localização. Dessa forma, a partir dos dados obtidos foi possível estudar o funcionamento dos algoritmos de fusão de sensores e compará-los nos experimentos descritos neste trabalho.

### 3.1.2 MCCR

Outro veículo utilizado neste trabalho foi o MCCR ou *MEC Combi Crawler Robot* (Figura 15), um robô *crawler* em atual desenvolvimento pelo SENAI CIMATEC, com objetivo de realizar missões para a limpeza e inspeção externa de navio-plataforma ou *FPSOs* para a identificação de defeitos do casco, das soldas e das superfícies por meio das técnicas MEC e inspeção visual, em ambiente submarino. Para realizar essas missões, o robô se acopla magneticamente no casco do navio, realiza a movimentação através de um conjunto de rodas e tem a capacidade de navegar remotamente, com velocidade e direcionamento controlados, usando sensores e um sistema de *feedback* visual a partir de uma câmera (SENAI CIMATEC, 2018).

Figura 15 – MEC Combi Crawler Robot (MCCR)



Fonte: Serviço Nacional de Aprendizagem Industrial e Innospection; LTD (2019)

Quanto ao sistema de localização submarina, ainda é possível utilizar odometria de rodas e sistemas de medidores inerciais, como IMU ou um sistema de INS, gerando uma posição e uma orientação relativas ao ponto inicial, visto que navegação e localização para veículos autônomos submarinos é desafiador devido à atenuação de sinais de rádio frequência e GPS (PAULL *et al.*, 2014). O MCCR contém quatro rodas que se movimentam de forma independente e todas as curvas são feitas de modo a manter o controle das rodas similar ao apresentado no Husky.

Uma possível solução seria utilizar sensores conhecidos como USBL e LBL, que utilizam ondas sonoras e um sistema de triangulação. No entanto, essa tecnologia é limitada e depende de uma instalação de *beacons* em pontos fixos do solo marinho ou em boias estacionárias. Logo, essa solução não poderia ser utilizada no MCCR devido à dificuldade de instalação desses dispositivos em um FPSO, visto que o solo marinho não é acessível em alguns pontos (PAULL *et al.*, 2014).

Outra possibilidade considerada na melhoria da estimação de localização, a partir de dados do ambiente, foi o uso de escaneamento a *laser*. Porém, essa abordagem também não foi considerada, devido à alta absorção da luz pela água, que limita o alcance de visualização, mesmo com a utilização de *lasers* com comprimentos de onda com menor absorção, como verde e azul (CASTILLÓN *et al.*, 2019). Esse problema é ainda agravado em águas rasas com alta turbidez, como pode ser o caso do MCCR durante a limpeza. Como o FPSO pode chegar a medir 250 m de comprimento e 50 m de largura, possui pouca iluminação e contém poucas informações do ambiente na área onde o robô realizará missões, o uso de *laser-scanners* não foi considerado adequado.

Dessa forma, o MCCR foi projetado para conter um sonar frontal multifeixe e um sonar mecânico de feixe único na parte traseira para reconhecimento e desvio de obstáculos para o sistema de navegação. Esses sonares podem ser considerados partes importantes para estimar localização, visto que esses dispositivos podem ser utilizados para uma visualização a uma longa distância, cobrindo parte da região da missão, servindo como parte importante desse trabalho.

### 3.3 MODELOS DESENVOLVIDOS PARA MCCR

#### 3.3.1 Criação do tanque para ambiente simulado de teste

O desenvolvimento desse trabalho com o MCCR foi realizado em dois ambientes simulados. O primeiro, para testes iniciais, foi um ambiente subaquático simulado no Gazebo emulando o tanque mostrado na Figura 16. O tanque possui 23 m comprimento, 19 m de largura e 8 m de altura e contém água salgada conforme o tanque situado no *Maritime Exploration Hall* localizado no *DFKI-Bremen (German Research Center for Artificial*

*Intelligence*), onde novas tecnologias da robótica e da indústria são testadas e demonstradas (DFKI, 2018).

Figura 16 – Tanque para testes de robótica, localizado no DFKI. Contém capacidade de 3,5 milhões de litros de água salgada



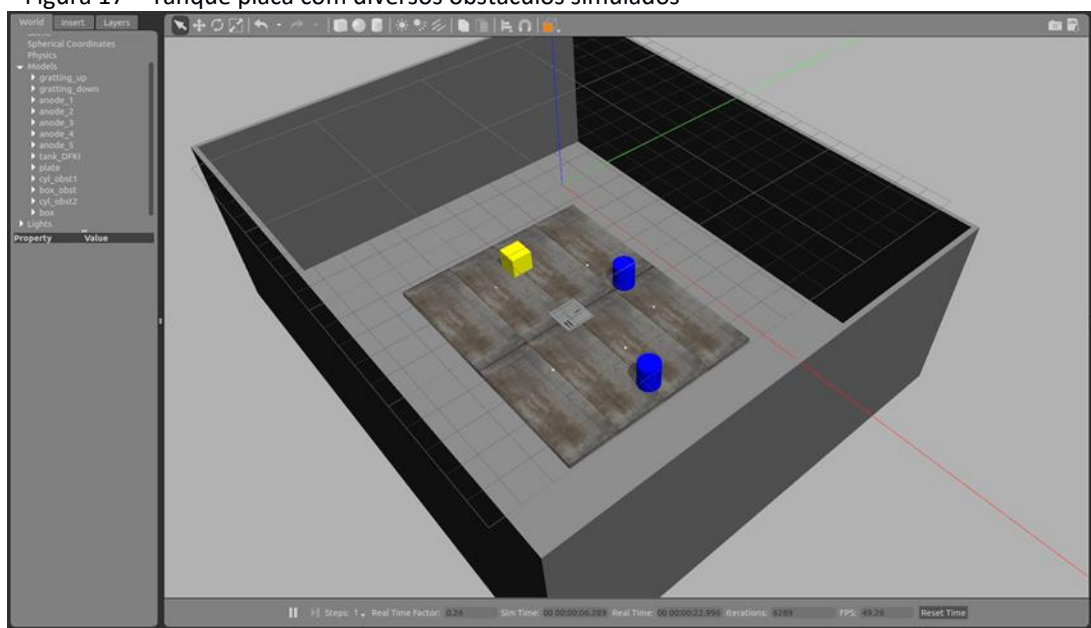
Fonte: DFKI (2018)

Essa estrutura foi desenvolvida e simulada no Gazebo junto à base de metal para realizar a inspeção de uma chapa de metal, como se o veículo estivesse apoiado na parte inferior de um navio FPSO. Essa placa tem o tamanho total de 12 m por 9,6 m e foi simulada e inserida no centro do tanque de modo horizontal e fixada de modo que o robô pudesse se movimentar, junto à presença de alguns obstáculos.

A estrutura foi desenvolvida no software *Blender* v2.83.3 com as dimensões definidas e inseridas no *Gazebo*. Também foram inseridos alguns obstáculos como caixas padrões e uma grade, simulando obstáculos com menor altura, permitindo a criação de um ambiente com barreiras, para possível reconhecimento de sensores, Figura 17.



Figura 17 – Tanque placa com diversos obstáculos simulados



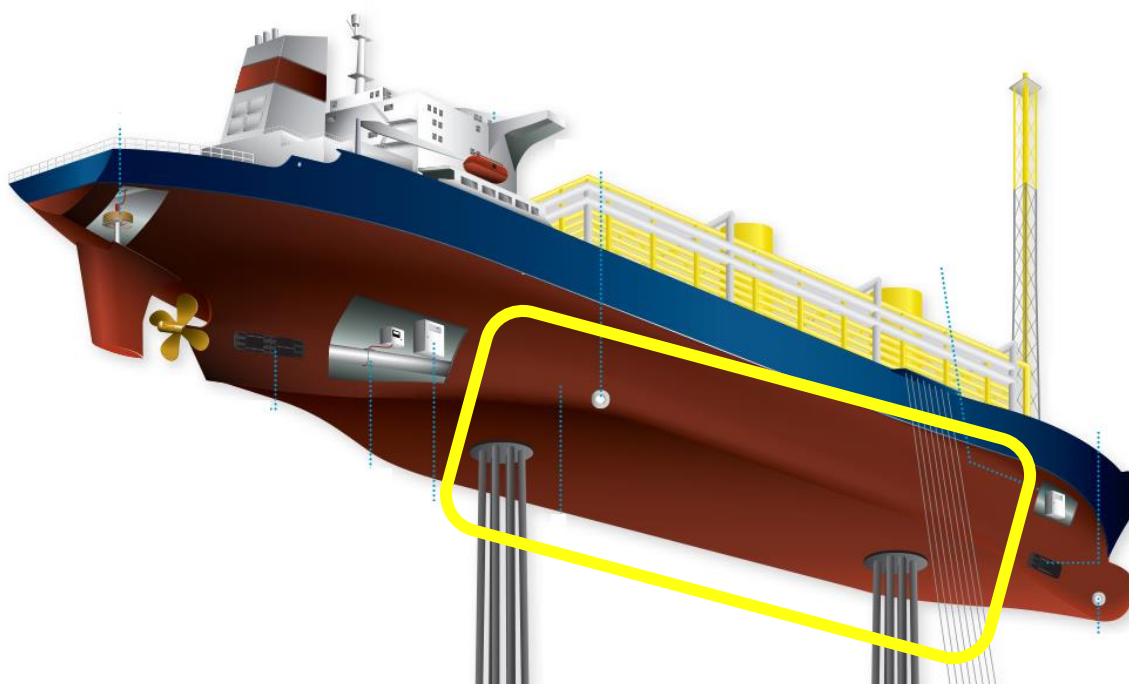
Fonte: autoria própria

O segundo ambiente desenvolvido e utilizado nos testes descritos na próxima seção foi criado pela necessidade de aumentar a distância percorrida pelo veículo. As missões do MCCR devem ocorrer na parte inferior e plana do casco do FPSO, como visualizado e marcado na Figura 18, por distâncias superiores a 100 m. Com esse novo ambiente, foi possível emular um ambiente para o MCCR mais próximo de uma operação em ambiente relevante (EVAC CLEANTECH SOLUTIONS ANYWHERE, 2012; MOREU *et al.*, 2015).

Adicionalmente, alguns obstáculos e uma placa com as mesmas dimensões de largura e comprimento do tanque foram adicionados ao ambiente, permitindo que o veículo alcançasse e identificasse as paredes, que representam o fim da borda do fundo do navio sendo uma parte importante para o reconhecimento do ambiente a partir do sonar.

As características e parâmetros do mapa apresentados são encontrados no Quadro 1 e visualizado nas Figura 19 e Figura 20.

Figura 18 – Visão da parte inferior de um FPSO com detalhes de algumas partes do navio



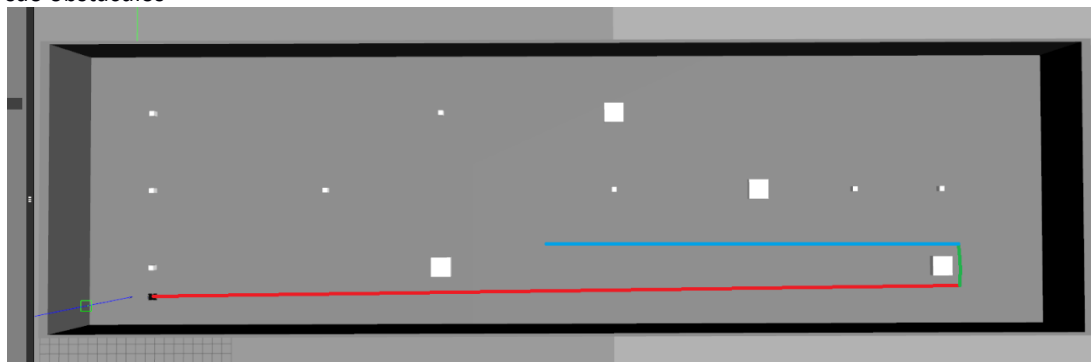
Fonte: adaptado de EVAC CLEANTECH SOLUTIONS ANYWHERE (2012)

Quadro 1 – Características para o mapa de testes com o MCCR

Mapas	Características		
	Tamanho	Configuração das caixas	Percurso do robô
Mapa M1	100 m × 30 m.	12 caixas localizadas ao decorrer do percurso. 2 tipos de caixas localizadas em pontos com espaçamento não definido,	Percurso com curvas ao fim do mapa.

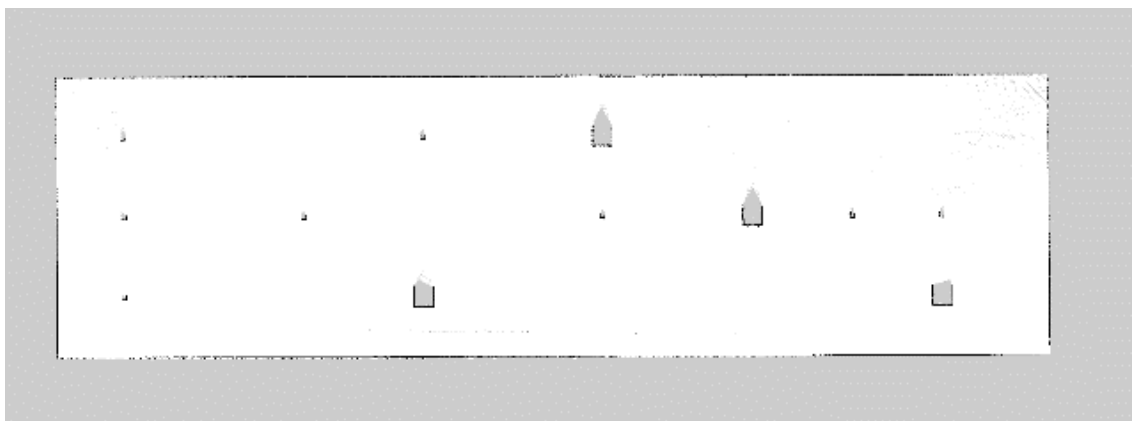
Fonte: autoria própria

Figura 19 – Representação do percurso do MCCR nos experimentos em vermelho e azul. Pontos brancos são obstáculos



Fonte: autoria própria

Figura 20 – Mapa utilizado para estimar localização com AMCL. Traçados pretos e cinzas são considerados como barreiras para o algoritmo



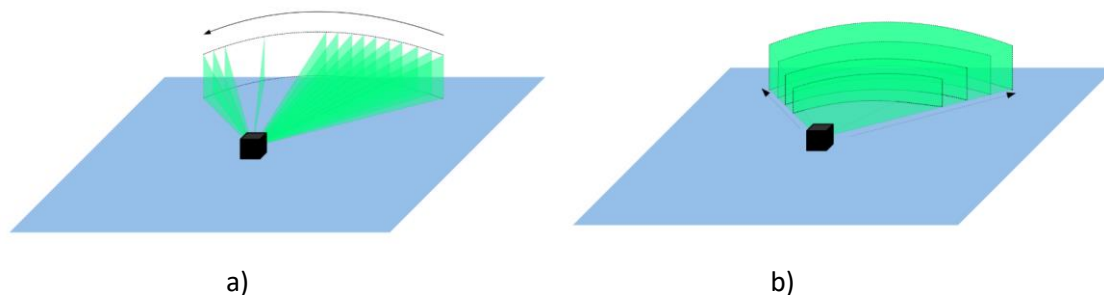
Fonte: autor própria

### 3.3.2 Sonares

O MCCR contém 2 sonares, o primeiro está localizado na parte frontal do robô denominado de *Forward-Looking Sonar (FLS)*, que gera imagens a partir de múltiplos feixes de ultrassom. O segundo é um sonar mecânico ou *mechanical scanning imaging sonar (MSIS)*, que tem a capacidade de gerar imagem a partir de um feixe único que rotaciona por determinado tempo, como visualizado na Figura 21 a). Esse dispositivo foi posicionado na parte traseira do veículo, realizando um giro de 210°, permitindo varrer parte da área ao redor, onde o FLS não pode. Para ambos os dispositivos, foi utilizado um sistema baseado em GPU com sonar simulado para aplicações em tempo real, junto ao Gazebo e ROS, desenvolvido e modelado por Cerqueira *et al.* (2017).

Esse modelo utiliza um sistema que simula pulsos das ondas ultrassônicas, que podem ser caracterizados em formato de *beams* ou feixes, com o formato de um cone com uma distância máxima de alcance, raio e *beamwidth* (ângulo vertical e horizontal) Figura 21 a) igual à da especificação do sonar. Dessa forma, é possível emitir feixes em regiões determinadas, em um ângulo específico e obter valores de intensidade e distância, baseados no tempo em que essa onda retorna ao dispositivo. A partir desses dados, é possível uma amostragem em intervalos regulares, que são chamados de *bins*, permitindo identificar a distância entre o sonar e algum objeto (Cerqueira *et al.*, 2017), como mostrado na Figura 21 b).

Figura 21 – Formas de leitura de dados dos sonares a) Emissão de ondas acústicas em diferentes direções com um único feixe por leitura. b) Ângulo de emissão da onda.

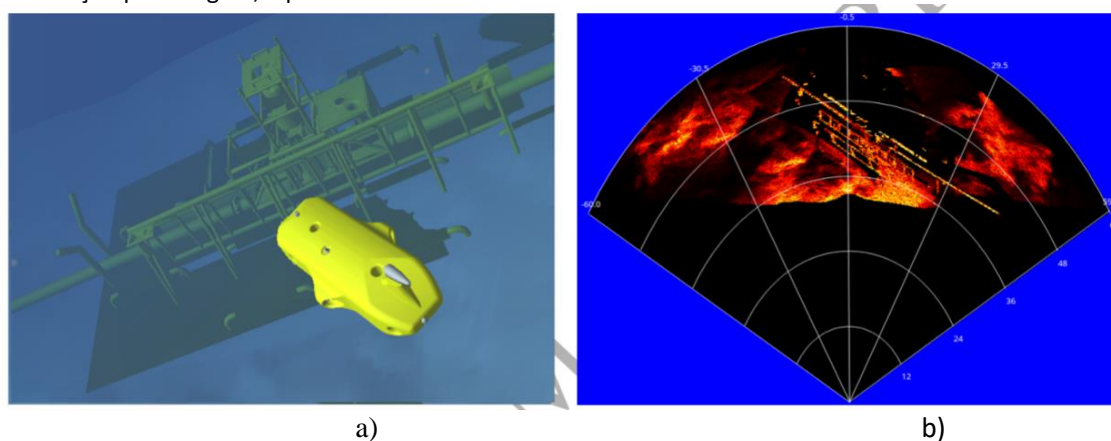


Fonte: Cerqueira *et al.* (2017)

Um dos fatores importantes analisados neste trabalho é a intensidade com que a onda retorna, sendo possível verificar se houve uma reflexão completa ou espalhamento. Dessa forma é possível interpretar que quando há um retorno com valores máximos de intensidade, uma reflexão completa aconteceu, e que há uma barreira na direção de emissão do pulso. Quando essa onda não dá sinal de retorno ou tem baixo valor de intensidade, pode ser considerada como um espaço vazio. Adicionalmente, quando o valor de intensidade é abaixo do valor máximo, é possível interpretar que houve o espalhamento da onda sonora ou absorção parcial da onda pelas barreiras, fazendo a onda retornar com menor amplitude.

Esses foram os princípios básicos para encontrar as barreiras para comparação e criação de uma imagem no software, como mostrado na Figura 22. Os parâmetros dos sonares utilizados são apresentados no Quadro 2.

Figura 22 – Exemplo de um AUV com um *forward looking* sonar, identificando estrutura submarina em simulação por imagem, a partir de ondas sonoras



Fonte: Cerqueira *et al.* (2017)

Quadro 2 – Dados e parâmetros para simulação dos sonares, baseados nos sensores utilizados no MCCR

<b>Tipo</b>	<b><i>Mechanical scanning sonar (MSIS)</i></b>	<b><i>Forward-looking Sonar (FLS)</i></b>
<b>Modelo</b>	Tritech Micron Sonar (TRITECH, 2020)	M900-2250-S 130/45-Mk2 (TELEDYNE MARINE, 2021)
<b>Número de beams</b>	1	768
<b>Resolução</b>	1,3 cm	7,5 mm
<b>Distância máxima utilizada</b>	30 m (alcance máximo de 75 m)	30 m (alcance máximo de 100 m)
<b>Taxa de atualização</b>	2 Hz	25 Hz
<b><i>Beam width</i></b>	3° (horizontal) 35° (vertical)	1° (horizontal) 12° (vertical)
<b>Campo de visualização</b>	Giro de 180°	130°

Fonte: autoria própria

### **3.3.2.1 Conversão de dados de sonar para PointCloud2**

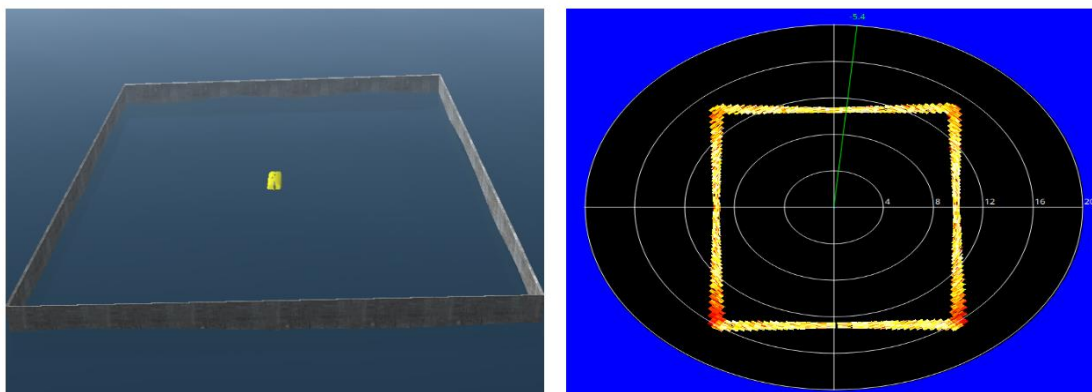
Um dos objetivos desse trabalho *foi* permitir a avaliação do filtro de partículas AMCL, obtendo informações do ambiente onde ele está inserido. O veículo MCCR contém tanto sonares frontais quanto traseiros, havendo necessidade de simular esses sonares como ferramenta para localização e adaptar os dados de entrada deles para utilização no algoritmo de estimação da localização. Como os dados obtidos dos sonares não são interpretados por determinados algoritmos, foi necessário converter esses sinais do ambiente gerados pelo sonar para uma imagem em pontos 3D que pudesse ser utilizada

no algoritmo AMCL. Para isso, os dados do sonar foram gravados, convertidos e filtrados, obtendo uma imagem do objeto em movimento.

Neste trabalho, foi utilizado o algoritmo AMCL, encontrado no pacote de biblioteca para ROS “robot\_localization” (MOORE, 2022), que permitiu analisar o ambiente e compará-lo com o mapa da área ou mapa de ocupação, para estimar a localização no eixo X e Y. Esse algoritmo necessita que a mensagem de entrada seja do tipo *LaserScan* ou *PointCloud2*, e esses dados não são fornecidos diretamente pelo modelo que simulava os sonares no ROS, necessitando de um *script* adicional que convertesse uma mensagem de sonar para uma nuvem de pontos do tipo *PointCloud2* para servir como entrada. Esse tipo de mensagem contém a representação de diversos pontos organizados em três dimensões e valores de intensidade, podendo ser utilizado para mostrar barreiras e a formação de objetos (RUSU; COUSINS, 2011).

O *script* foi desenvolvido utilizando a plataforma ROS, a linguagem Python e bibliotecas *numpy* (HARRIS *et al.*, 2020) e *scipy* (JONES *et al.*, 2001), para processamento de matrizes. Como entrada foram utilizados dados a partir de uma simulação de sonar de imagem no Gazebo (CERQUEIRA *et al.*, 2017).

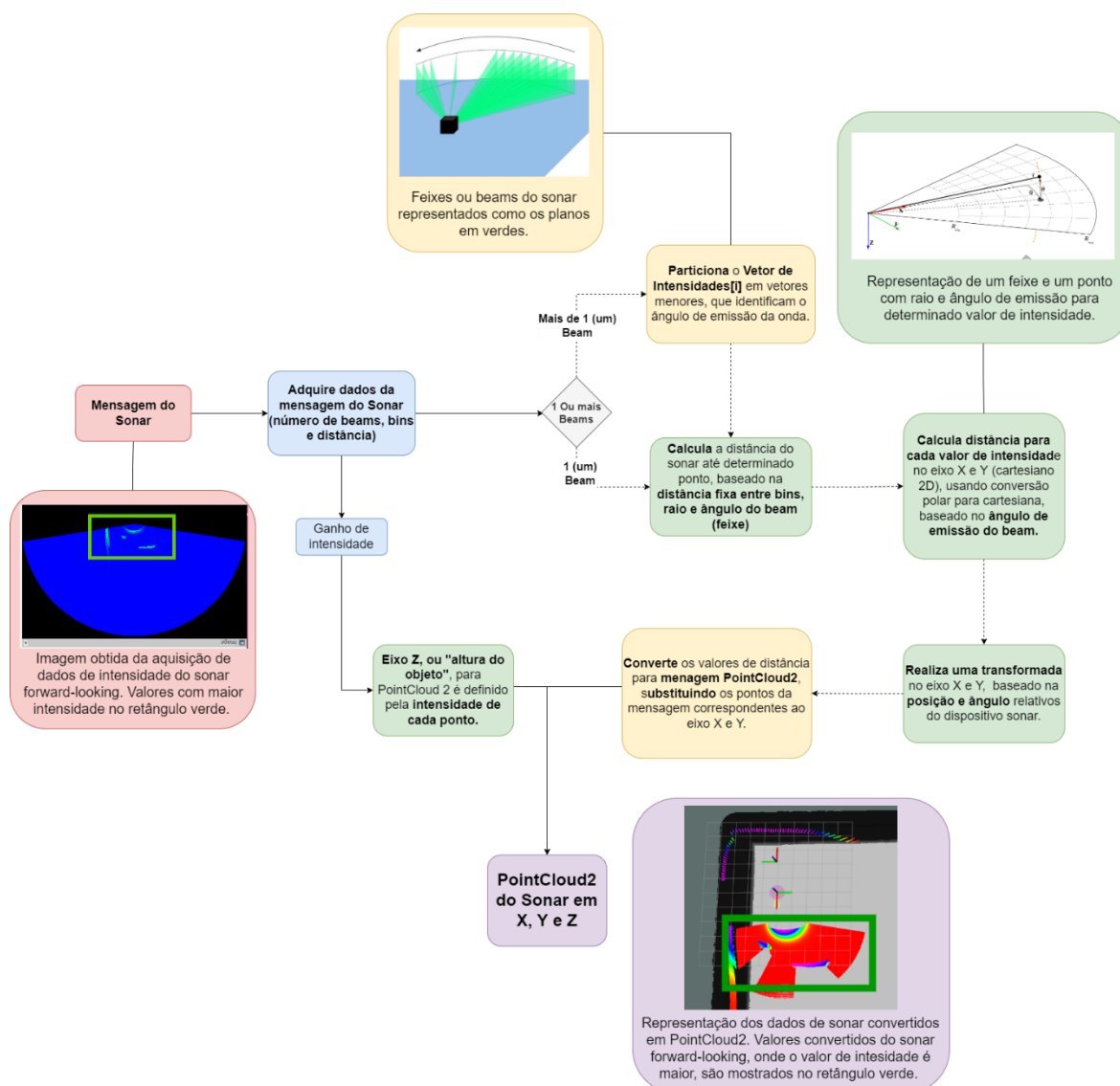
Figura 23. Representação de imagem fornecida pela simulação



Fonte: Cerqueira *et al.* (2017)

De forma geral, o algoritmo usa dados de um vetor de intensidade, valores de número de *beams* e distância entre *bins* para criar uma imagem única em *PointCloud2* no ambiente 3D, e a cada iteração é renovado, seguindo a metodologia indicada na Figura 24. Esse algoritmo utilizou as mensagens originadas do sonar multifeixe e do sonar mecânico de feixe único e foi inserido na simulação do veículo MCCR.

Figura 24 – Algoritmo para conversão de dados do sonar em *PointCloud2* no eixo XYZ. As imagens representam os dados do sonar no cálculo



Fonte: Autoria própria com imagens adaptadas de Cerqueira *et al.* (2017)

De forma mais detalhada, o algoritmo segue as seguintes etapas nessa sequência:

1. Adquirir dados da mensagem do sonar;
  - a. Adquirir valores de intensidade, obtidos como um vetor único e longo;
  - b. Adquirir os parâmetros de número de *bins* ( $b$ ) e *beams* ( $B$ );
  - c. Adquirir dados de resolução e de distância máxima e mínima da mensagem;
2. Particionar o vetor de intensidades em vetores menores de mesmo tamanho, baseado na quantidade de *beams* ( $B$ ), caso seja um sonar *multibeam*;
3. Calcular a distância do sonar até determinado ponto, baseado na quantidade e distância entre os *bins*, utilizando valores de raio ( $R$ ) e ângulo de emissão ( $\alpha$ );

4. Calcular a distância para cada ponto, onde há um valor de intensidade, transformando os valores de raio (R) e ângulo de emissão ( $\alpha$ ) em posição em coordenadas cartesianas em X e Y;
5. Realiza uma transformada no eixo X e Y, baseado na posição e ângulo relativos ao veículo;
6. Converter os valores de distância para mensagem do tipo *PointCloud2*.
  - a. Substituir, na mensagem, os pontos correspondentes ao eixo X e Y.
  - b. Valores de intensidade são multiplicados por um ganho (k), e os pontos correspondentes ao Z no eixo cartesiano são inseridos baseados nessa multiplicação, definindo a altura do objeto.
7. Remover dados de *PointCloud2* que correspondem a intensidade 0 ou próximo, restando somente informações das barreiras.
8. Transformar *PointCloud2* para mensagem de *LaserScan* a ser inserida como entrada no algoritmo AMCL, utilizando algoritmo "*pointcloud\_to\_laserscan*" (BOVBEL, 2015).  
Etapa necessária para um dos experimentos a seguir.

Essa rotina pode ser utilizada tanto para o sonar *forward-looking* multifeixe, quanto para o sonar mecânico de feixe único. No entanto, para o sonar mecânico (MSIS), é preciso uma etapa posterior da criação dos pontos, que permite acumular esses pontos e formar uma nova mensagem de *PointCloud2*. Essa nova mensagem contém vários pontos, baseados na posição e na orientação nas quais o sonar mecânico estava naquele momento, possibilitando a visualização na barreira. Adicionalmente, quando o sonar faz uma volta completa e retorna ao mesmo ângulo, esses pontos são substituídos por novos. Essa etapa só ocorre nesse sonar, pois é preciso um longo tempo de giro para identificação de vários pontos do ambiente (CHO *et al.*, 2015).

### 3.3.3 INS

Para as aplicações em ambiente submarino, o INS é um equipamento que estima posição de um veículo de um modo mais eficiente que a odometria quando não há dados globais, apresentando baixo erro médio acumulado. O MCCR utilizou o *ROVINS Nano* do fabricante *IxBlue* (IXBLUE, 2020), cujos dados são apresentados no Quadro 3. Dessa forma,



foi desenvolvido um algoritmo para simular esse equipamento acoplado a um veículo robótico simulado no ROS e Gazebo, respeitando a precisão do dispositivo.

Quadro 3 – Dados do INS utilizados neste projeto

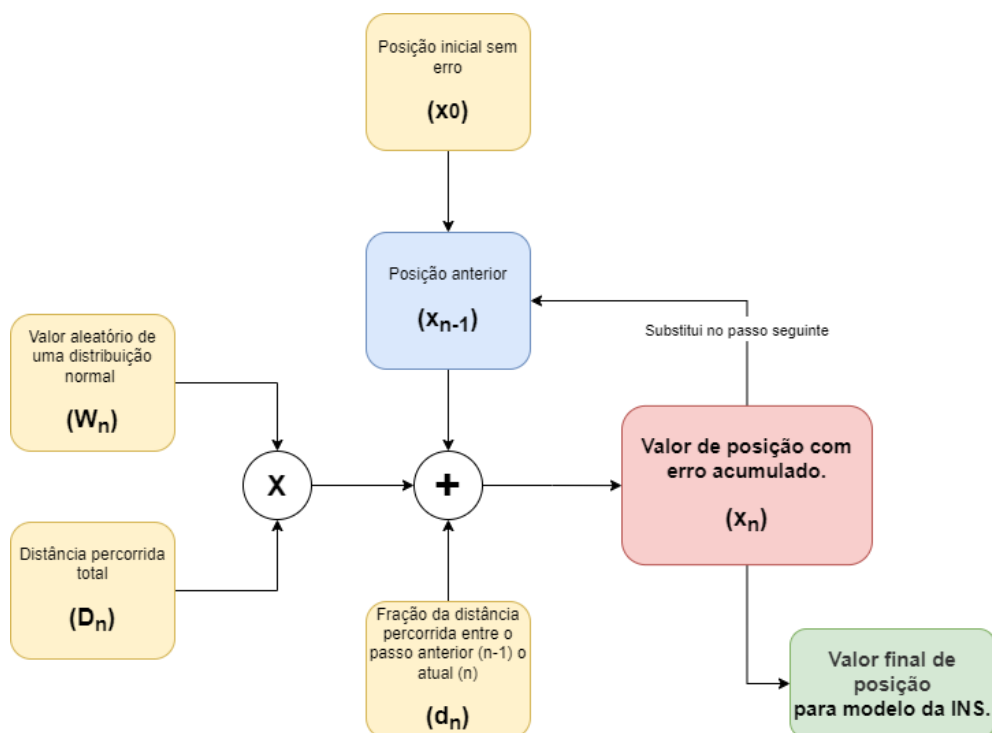
<b>Sensor</b>	<b>Erro esperado eixo X e Y</b>	<b>Erro esperado <i>Roll e Pitch</i></b>
INS <i>iXblue – Rovins Nano</i>	0,04% da distância percorrida	0,05 graus

Fonte: IxBlue (2020)

Para manter as características do próprio dispositivo, o INS foi modelado de modo que erro gerado a partir da distância percorrida, fosse incluída. Para isso, foram utilizadas algumas informações da folha de dados do sensor, apresentadas no Quadro 3 (IXBLUE, 2020) e um algoritmo, que é demonstrado a seguir na Figura 25 e a na equação 6. As variáveis utilizadas foram as seguintes:

- $x_0$  - Posição inicial sem erros definida pelo ambiente simulado (*ground truth*);
- $D_n$  - Distância percorrida total em determinado espaço de tempo fixo ( $t$ ) ou passo ( $n$ );
- $d_n$  - Fração da distância percorrida entre o passo  $n - 1$  e  $n$ ;
- $W_n$  - É um número aleatório de uma distribuição normal (podendo ser negativo), baseado na variância do dispositivo, a ser multiplicado pela distância percorrida  $D_n$  no passo  $n$ ;
- $x_n$  - Valor estimado de posição, medido pelo INS, com erro inserido no passo  $n$ .

Figura 25 – Diagrama de bloco da modelagem desenvolvida para o sensor. Saída do valor do INS no diagrama vermelho



Fonte: autoria própria

$$x_n = x_{n-1} + d_n + W_n D_n \quad (6)$$

De forma resumida, para o cálculo do erro acumulado, a distância percorrida em determinado tempo é multiplicada por um valor que se refere à precisão do INS e somada ao valor de posição real do sensor no momento inicial ( $x_0$ ). Após o primeiro cálculo, a posição estimada do modelo do sensor é somada ao erro calculado, como ( $x_1 = x_0 + W_1 \cdot D_1 + d_1$ ). Logo, ao decorrer dos *loops*, esse erro será sempre incrementado, quão mais distante o veículo percorra.

## 4 EXPERIMENTOS COM O HUSKY

Para os primeiros experimentos, o veículo *Husky* foi escolhido devido à sua fácil simulação, fácil disponibilização na internet e por ter características similares ao MCCR, como modularidade de sensores. Dessa forma, a partir dos dados obtidos, foi possível estudar o funcionamento dos algoritmos de fusão de sensores e compará-los nos experimentos apresentados neste trabalho.

Para os testes, foram utilizados um ambiente 3D simulado e um mapa 2D do tutorial inicial do *Husky no Gazebo*, onde um ambiente interno de uma sala foi criado com vários obstáculos distribuídos, como apresentado na Figura 11.

### 4.1 EXPERIMENTO H1: ALGORITMOS DE FUSÃO E AMBIENTE DE SIMULAÇÃO

O experimento H1 foi realizado no ambiente da Figura 11, no qual foram realizados movimentos que permitissem a comparação entre valores de localização sem erro, ou *ground truth*, e os valores estimados para três algoritmos de fusão de sensores com as configurações-padrão disponibilizadas pelo tutorial (CLEARPATH, 2015). Foram eles o filtro de Kalman estendido, filtro de Kalman *unscented* e filtro de partículas, parte do pacote "*robot\_localization*". Os sensores fundidos em cada algoritmo e as variáveis selecionadas para estimar a posição são apresentados no Quadro 4.

Quadro 4 – Algoritmos, sensores e variáveis para fusão no experimento H1

<b>Algoritmos – Experimento H1</b>	<b>EKF</b>	<b>UKF</b>	<b>Filtro de partículas</b>
<b>Sensores para fusão</b>	- IMU; - Odometria de rodas	- IMU; - Odometria de rodas	- IMU; - Odometria de rodas - <i>LiDAR</i>
<b>Variáveis selecionadas para estimar posição</b>	- Velocidade linear; - Orientação; - Aceleração;	- Velocidade linear; - Orientação; - Aceleração;	- Velocidade linear; - Orientação; - Aceleração; - Distância entre o robô e barreiras (Dados de <i>LaserScan</i> )

Fonte: autoria própria

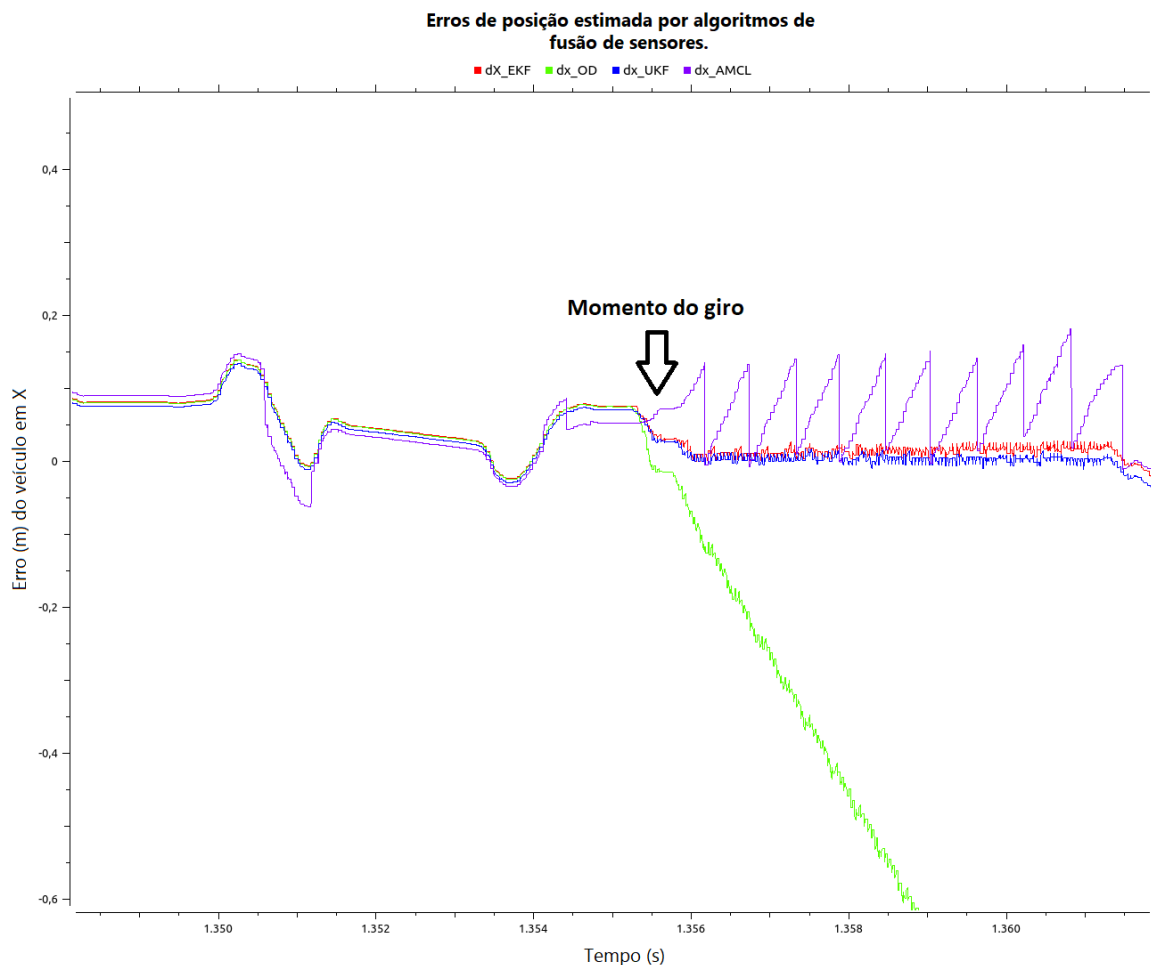
Nesse experimento, o veículo faz um percurso retilíneo de 10 m e, após isso, um giro no próprio eixo durante 20 segundos, controlado por controle remoto. A diferença entre a posição estimada e a *ground truth* foi calculada e os resultados foram analisados de forma gráfica no eixo do movimento. Dessa forma, foi possível avaliar o erro associado a cada filtro, como funcionam e a necessidade de configuração do algoritmo, dos sensores e do ambiente. Na sequência, essa informação é disponibilizada em forma de gráficos para comparação.

Foi esperado que o erro associado à odometria de rodas, como único sensor, crescesse com a distância devido a possíveis escorregamentos. Esse experimento permitiu verificar se os dados fundidos do IMU identificam esse escorregamento e diminuem o erro associado.

#### **4.1.1 Resultados – Experimento H1**

A Figura 26 apresenta um gráfico com o erro de localização durante o percurso percorrido pelo Husky. Esse percurso foi retilíneo até aproximadamente metade do tempo quando um giro foi realizado. Esse gráfico demonstra que o erro de odometria de rodas (linha verde) e o estimado pelos algoritmos são bem próximos, com pequenas variações até o movimento de giro ocorrer. Mas a partir do ponto mostrado no gráfico da Figura 26, o erro acumulado da odometria cresceu devido à percepção errada da orientação. No entanto, os valores estimados pelos filtros (linha vermelha e azul) tiveram erro bastante reduzido, próximo de zero, pois os acelerômetros e giroscópios presentes no IMU identificaram a orientação do robô de forma mais precisa, como esperado.

Figura 26 – Erro comparando a posição original do robô e os resultados obtidos por EKF (Vermelho), UKF (Azul), Odometria de Rodas (Verde) e AMCL (Roxo)



Fonte: autoria própria

O resultado obtido pelo algoritmo de filtro de partículas (AMCL), linha roxa, segue a odometria até o momento do giro quando passa a apresentar um formato de dente serra. Esse comportamento se deve ao tempo de resposta do algoritmo de AMCL que depende do número de partículas utilizadas.

## 4.2 EXPERIMENTO H2: CONFIGURAÇÃO DOS ALGORITMOS DE FUSÃO

O segundo teste foi realizado com o objetivo de verificar e avaliar a eficiência dos filtros de Kalman ao ocorrer uma mudança abrupta na movimentação do robô que pudesse ocasionar o crescimento do erro súbito em algum dos sensores, como no caso de um escorregamento ou batida, assim como identificar as melhores configurações dos algoritmos para obter melhores resultados nessa situação, utilizando o pacote “*robot\_localization*” com os algoritmos e sensores indicados no Quadro 5.

Para isso, foi utilizado o mesmo ambiente e sensores do experimento H1. Porém, o veículo fez um movimento retilíneo por controle remoto em linha reta durante 20 m, saindo do centro do mapa até as bordas, onde encontrou uma barreira. O choque com a barreira causou um escorregamento das rodas, que deviam continuar a girar após o impacto, como apresentado na Figura 27.

Também foram avaliadas três configurações da matriz de covariância dos filtros de Kalman disponibilizados pelo “*robot\_localization*”, com três experimentos alternando valores de covariância  $\sigma_x^2$  na matriz R entre “0,2”, “0,02” e “0,002”. Dessa forma, os dados dos sensores foram obtidos e fusionados, obtendo um valor estimado de localização no eixo X, comparando o crescimento do erro em relação ao *ground truth* e apresentando-os em um gráfico.

Quadro 5 – Algoritmos, sensores e variáveis para fusão no experimento H2

<b>Algoritmos – Experimento H2</b>	<b>EKF</b>
<b>Sensores fundidos</b>	- IMU; - Odometria de rodas; - GPS.
<b>Variáveis selecionadas para estimar posição</b>	- Velocidade linear; - Orientação; - Aceleração; - Posição (GPS).

Fonte: autoria própria

Figura 27 – Ambiente de simulação com o veículo robótico terrestre simulado no momento do impacto

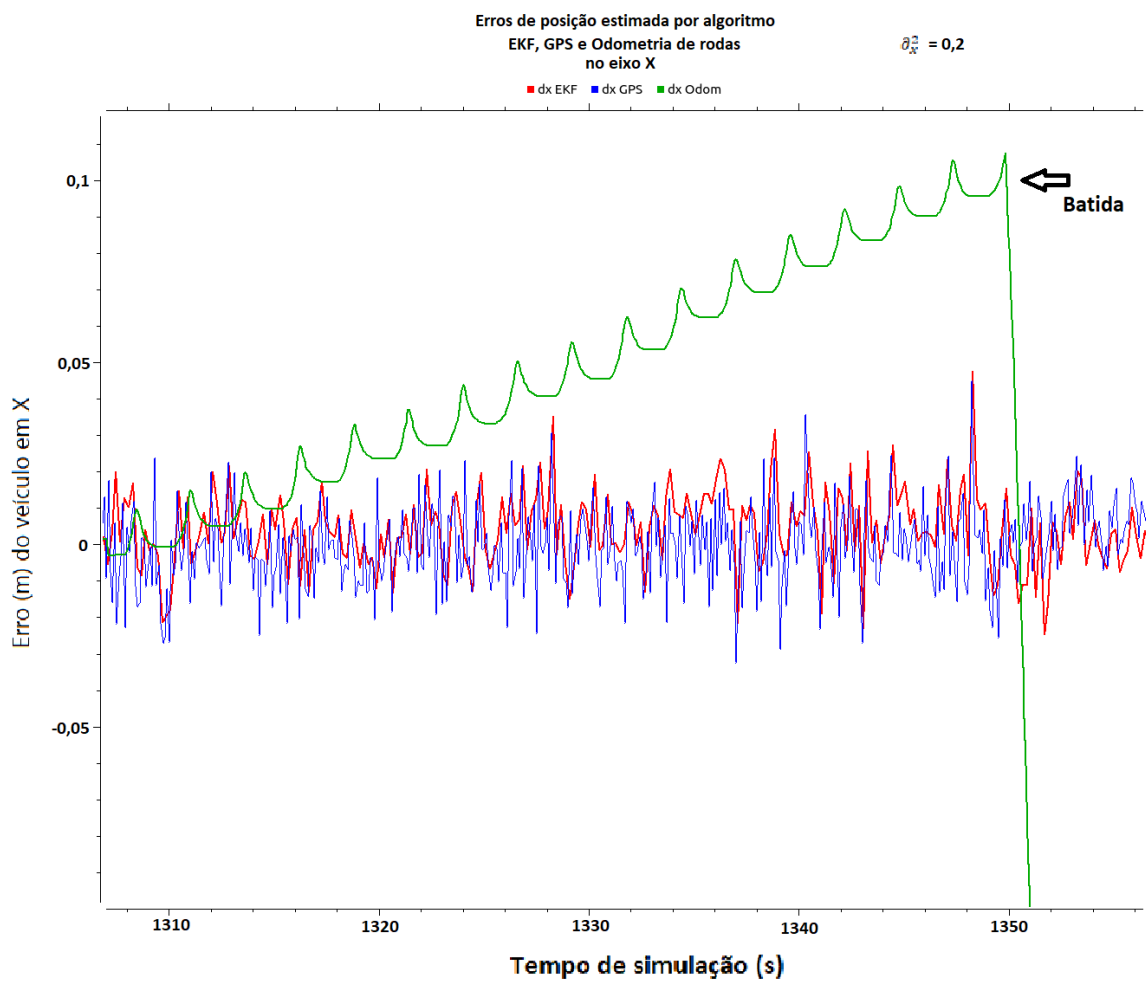


Fonte: Simulação Gazebo. Autoria própria

#### 4.2.1 Resultados – Experimento H2

Os erros simulados para 3 valores distintos da matriz de covariância no eixo X são apresentados nas Figura 28, Figura 29 e Figura 30. A linha verde correspondente ao erro da odometria de rodas cresce linearmente com o tempo devido ao escorregamento. As oscilações em torno da linha de tendência são devido ao atraso na saída dos dados obtidos pelo GPS e o cálculo do erro, similar ao ocorrido em H1 com o AMCL. A linha vermelha corresponde ao erro do resultado obtido com a fusão entre odometria de rodas, IMU e GPS com o filtro de Kalman estendido. Essa curva segue os valores do GPS com uma pequena variação devido à fusão de dados de odometria e do IMU. Quando o choque ocorreu com a parede e o erro da odometria cresceu, a fusão deveria dar maior peso ao GPS. A rapidez com que isso é feito depende do valor de covariância utilizado.

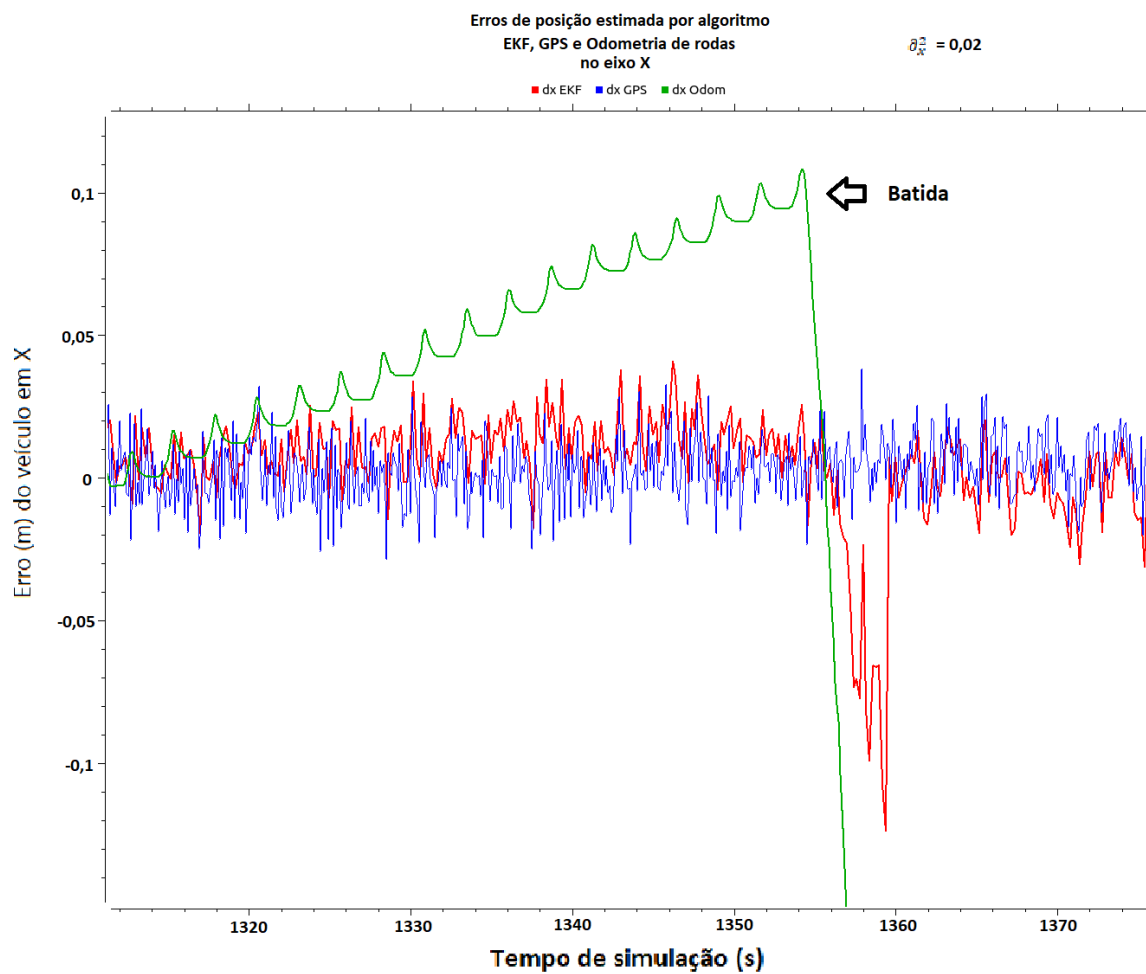
Figura 28 – Gráfico de erro para odometria de Rodas (verde), GPS (azul) e EKF (vermelho) no eixo X para valor de covariância da matriz R,  $\sigma_x^2$  igual a 0,2



Fonte: autoria própria

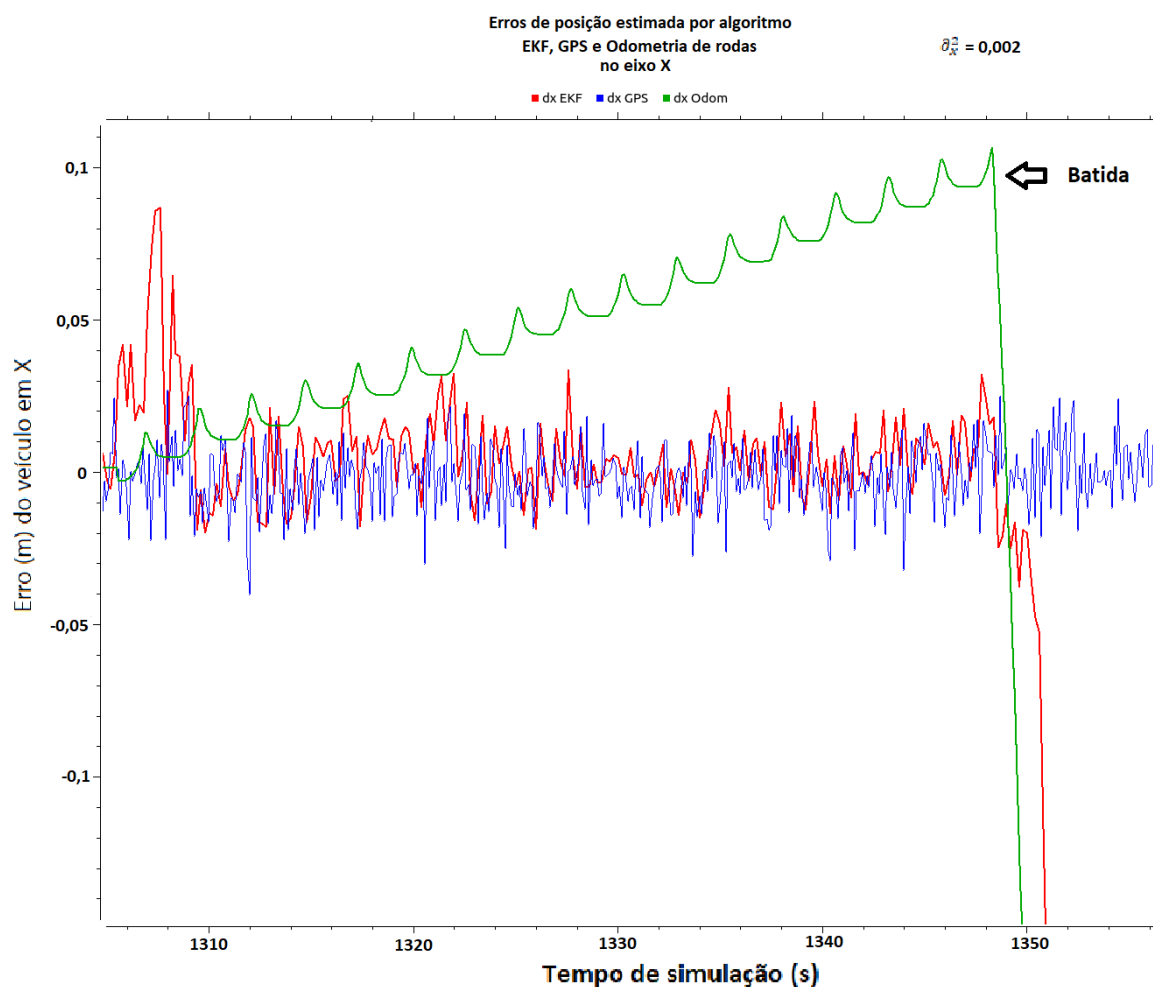


Figura 29 – Gráfico de erro para odometria de rodas (verde), GPS (azul) e EKF (vermelho) no eixo X para valor de covariância da matriz R,  $\sigma_x^2$  igual a 0,02



Fonte: autoria própria

Figura 30 – Gráfico de erro para odometria de rodas (verde), GPS (azul e EKF (vermelho) no eixo X para valor de covariância da matriz R,  $\sigma_x^2$  igual a 0,002



Fonte: autoria própria

Os resultados da Figura 28 mostram que ao utilizar valor de covariância do modelo  $\sigma_x^2$  igual a 0,2, os dados para os sensores fusionados com o algoritmo de Kalman se mantiveram próximos aos obtidos com o GPS, mesmo após o choque. Nesse caso o algoritmo reconheceu rapidamente a medição errônea da odometria e reduziu o ganho de Kalman representativo da odometria de rodas.

Na segunda simulação, apresentada na Figura 29, o valor de covariância do modelo foi reduzido em 10x para 0,02. Neste caso os dados fusionados demoraram alguns segundos após o choque para voltar ao valor próximo ao obtido com o GPS. A identificação da medição errônea e redução do ganho de Kalman representativo da odometria foi feito de maneira mais lenta, causando um pico de erro no momento do choque.

Na terceira simulação, mostrada na Figura 30, o valor de variância do modelo,  $\partial_x^2$ , foi reduzido ainda mais para 0,0002, um valor bem menor comparado aos anteriores. Nesse caso os resultados fusionados após o choque divergiram totalmente do GPS, e o erro não chegou a se recuperar dentro do período simulado. Essa reação provém da lenta convergência do ganho de Kalman, ao existir uma mudança brusca das medições.

Dessa forma, é possível perceber que o algoritmo de fusão de sensores consegue compensar o erro da mensagem de odometria desde que a covariância do modelo EKF,  $\partial_x^2$ , seja suficientemente grande. Quando ele é muito pequeno, a resposta do algoritmo fica lenta, até o ponto onde os dados do GPS são ignorados e o algoritmo não consegue mais estimar a posição após o escorregamento de rodas causada pela batida. Adicionalmente, os valores do erro do GPS com ruído adicionado se mantiveram constantes sem muitas variações.

#### 4.3 EXPERIMENTO H3: INFLUÊNCIA DO AMBIENTE COM FILTRO DE PARTÍCULAS COM LIDAR

No experimento H3, a fusão com filtro de partículas também foi avaliada com o veículo *Husky*, conforme descrito no Quadro 6. Para tanto, novos ambientes foram desenvolvidos com o objetivo de simular uma área maior não estruturada com obstáculos similares. Dessa forma, foi possível verificar a eficiência dos algoritmos de filtro de partículas em ambiente com diferentes objetos, além de verificar a eficiência dos filtros de Kalman após longas distâncias percorridas e curvas que causam escorregamento de rodas.



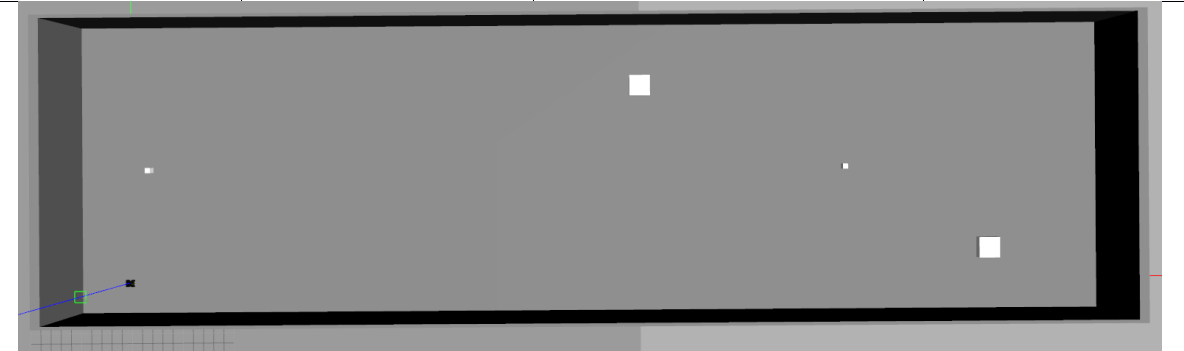
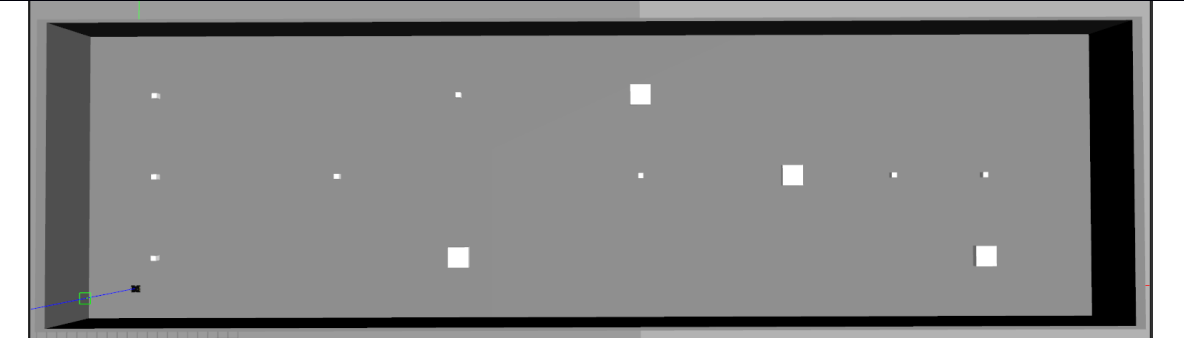
Quadro 6 – Algoritmos, sensores e variáveis para fusão no experimento H3

<b>Algoritmos – Experimento H3</b>	<b>EKF</b>	<b>UKF</b>	<b>Filtro de partículas</b>
<b>Sensores fundidos</b>	- IMU; - Odometria de rodas	- IMU; - Odometria de rodas	- IMU; - Odometria de rodas - LiDAR
<b>Variáveis selecionadas para estimar posição</b>	- Velocidade linear; - Orientação; - Aceleração;	- Velocidade linear; - Orientação; - Aceleração;	- Velocidade linear; - Orientação; - Aceleração; - Distância entre o robô e barreiras (Dados de <i>LaserScan</i> )

Fonte: autoria própria

A estrutura do ambiente foi definida pela localização e número de dois tipos de caixas, uma maior (2 m × 2 m) e uma menor (0.5 m × 0.5 m). Quatro grandes mapas 2D e 3D foram construídos a partir do software de simulação *Gazebo*. Todos esses mapas foram baseados em ambientes grandes e menos estruturados, como num galpão industrial, com variações no número de caixas presentes no ambiente, conforme apresentado no Quadro 7.

Quadro 7 – Definição das características dos mapas, ambientes e percursos

Mapas	Características		
	Tamanho	Configuração das caixas	Percurso do robô
Mapa H3.1	400m × 20 m	15 caixas localizadas ao decorrer do percurso com 2 tipos de tamanhos diferentes. Mapa menos estruturado.	Percurso retilíneo
			
Mapa H3.2	400 m × 20 m	19 caixas largas localizadas ao decorrer do percurso. Mapa mais estruturado.	Percurso retilíneo
			
Mapa H3.3	100 m × 30 m	4 caixas localizadas ao decorrer do percurso. 2 tipos de tamanhos diferentes. Mapa menos estruturado.	Percurso com curvas ao fim do mapa.
			
Mapa H3.4	100 m × 30 m.	12 caixas localizadas ao decorrer do percurso. 2 tipos de tamanhos diferentes. Mapa mais estruturado.	Percurso com curvas ao fim do mapa.
			

Fonte: autoria própria

Os mapas H3.1 e H3.2 foram construídos para avaliar a eficiência do filtro de partículas em um ambiente com paredes e objetos parecidos em quantidades diferentes, permitindo avaliar a eficiência quando o ambiente estava mais vazio ou não, no caso em

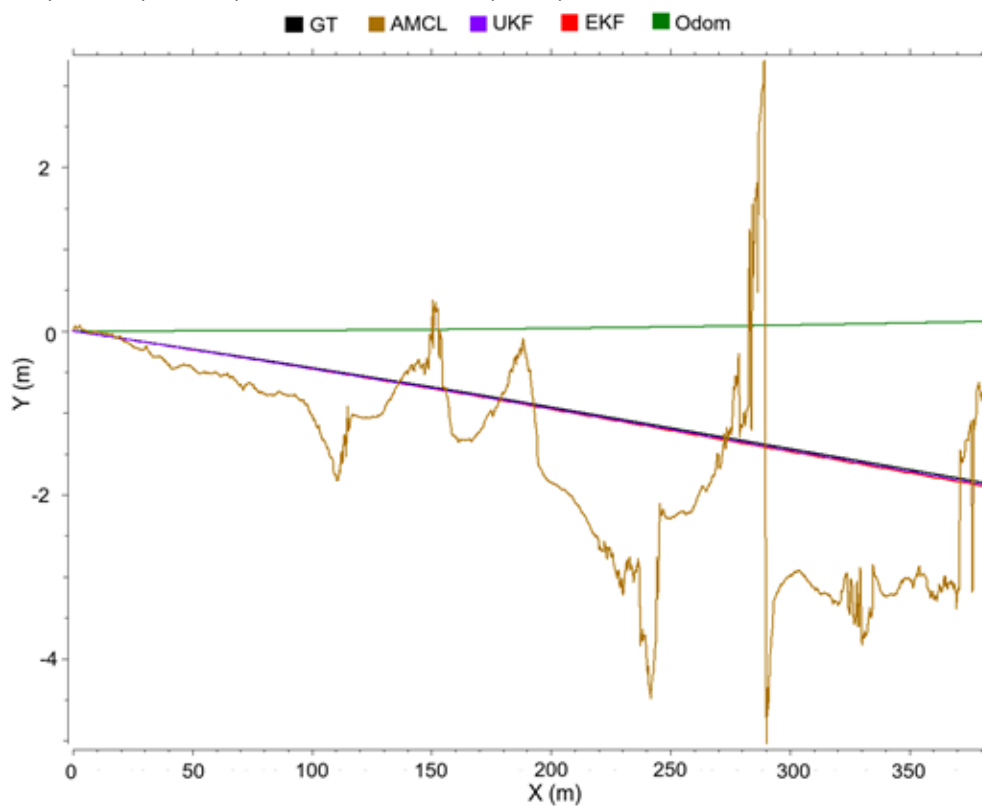
que o robô percorreu um caminho retilíneo. Com os mapas H3.3 e H3.4 foi adicionado o efeito de curvas que causam escorregamento. Nesses dois últimos casos, o robô fez um movimento contornando obstáculos, possibilitando a avaliação da eficiência de localização com filtro de partículas com a quantidade de obstáculos e dos filtros de Kalman na presença de escorregamento em grandes percursos.

O número máximo e mínimo de partículas do algoritmo AMCL foram definidos para os valores padrões entre 500 e 2000. O parâmetro *range* máximo para o AMCL foi definido em 30 m, tamanho aproximado da largura do tanque. Uma comparação entre o *ground truth* e a posição estimada é fornecida graficamente para cada um dos filtros testados e comparada com o caso em que apenas a odometria foi utilizada, permitindo demonstrar a eficiência de cada um dos algoritmos de fusão em um determinado ambiente.

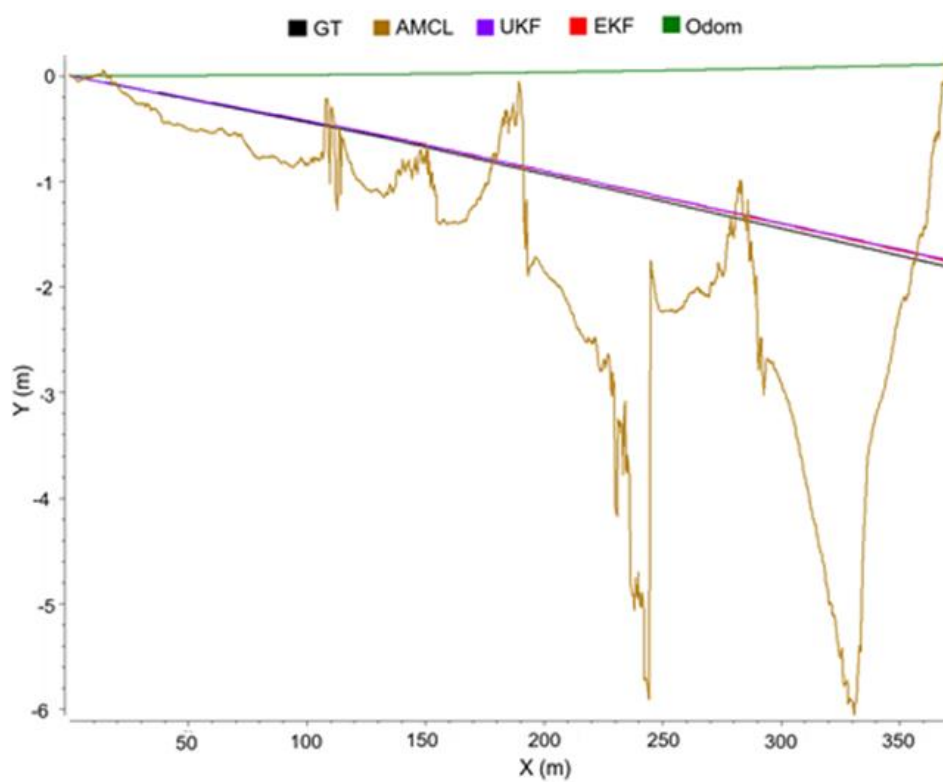
#### 4.3.1 Resultados – Experimento H3

No caso em que o robô percorreu um longo caminho retilíneo, em que não foram feitas curvas, o impacto do deslizamento das rodas durante as curvas pôde ser visualizado na simulação dos Mapas H3.1 e H3.2. Os caminhos percorridos para esses mapas são mostrados nas Figura 31 (a) e (b). As curvas verdes mostram a localização prevista apenas pela odometria. E como observado em ambos os casos, há um crescimento do erro inicial que continua a acumular linearmente com a distância percorrida. O uso de EKF e UKF permitiu detectar adequadamente a localização em ambos os casos e a posição prevista seguiu o *ground truth*, resultando em um erro baixo para esses filtros. A estimativa da AMCL não foi satisfatória no mapa menos estruturado, mostrado na Figura 31 (a), quando não havia caixas dentro da faixa de varredura do LiDAR. Aumentar a estrutura do ambiente adicionando caixas não foi suficiente para melhorar o desempenho do AMCL. Conforme visualizado na Figura 31 (b), o algoritmo AMCL não conseguiu identificar uma posição correta, pois não pôde distinguir pequenas alterações no posicionamento das caixas ou pelo fato de todas as caixas serem iguais em agrupamentos semelhantes.

Figura 31 – Caminhos percorridos para os mapas H3.1 (a) e H3.2 (b). As linhas coloridas mostram o *ground truth* (linha preta), a localização estimada usando EKF (vermelho), UKF (roxo), AMCL (marrom) e odometria de rodas (verde)



(a)



(b)

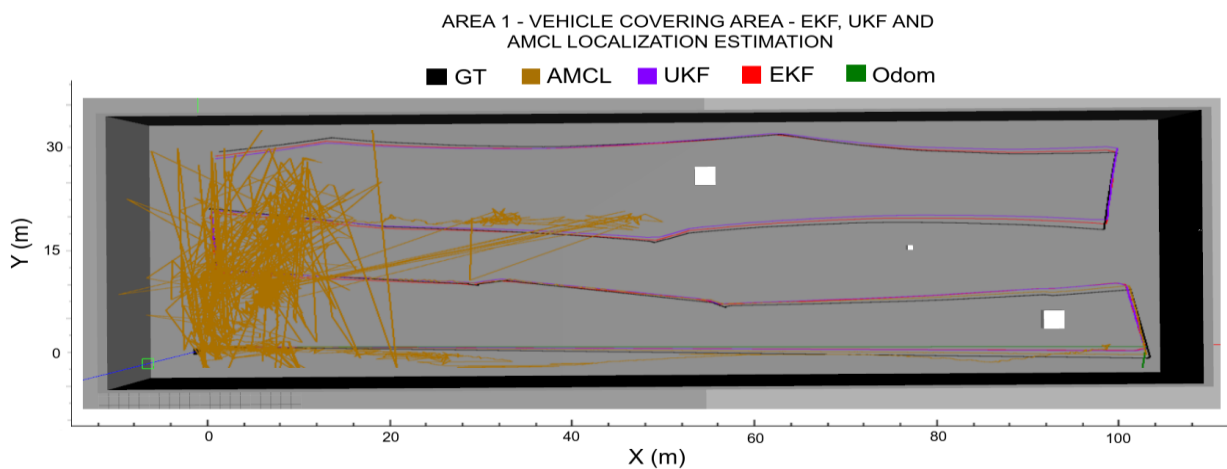
Fonte: autoria própria

A Figura 32 (a) e (b) mostra o caminho percorrido e estimativa de posição nos mapas H3.3 e H3.4 de 100 m × 30 m, com as caixas que definem a estrutura do ambiente. O robô seguiu um caminho de uma parede à outra, fazendo curvas de 180° graus quando chegou numa das paredes. A linha preta mostra o *ground truth* e a posição estimada para cada um dos casos investigados é mostrada pelas linhas coloridas. Embora os caminhos planejados sejam semelhantes, os caminhos tomados não foram idênticos devido a alterações no início do percurso e derrapagem durante as curvas. É também possível perceber que o erro da posição estimada apenas pela odometria (linhas verdes) para ambos os mapas após a primeira volta cresceu, ficando fora de escala dos outros. No entanto, com os filtros de Kalman, o erro de posição foi corrigido mesmo após diversos escorregamentos.

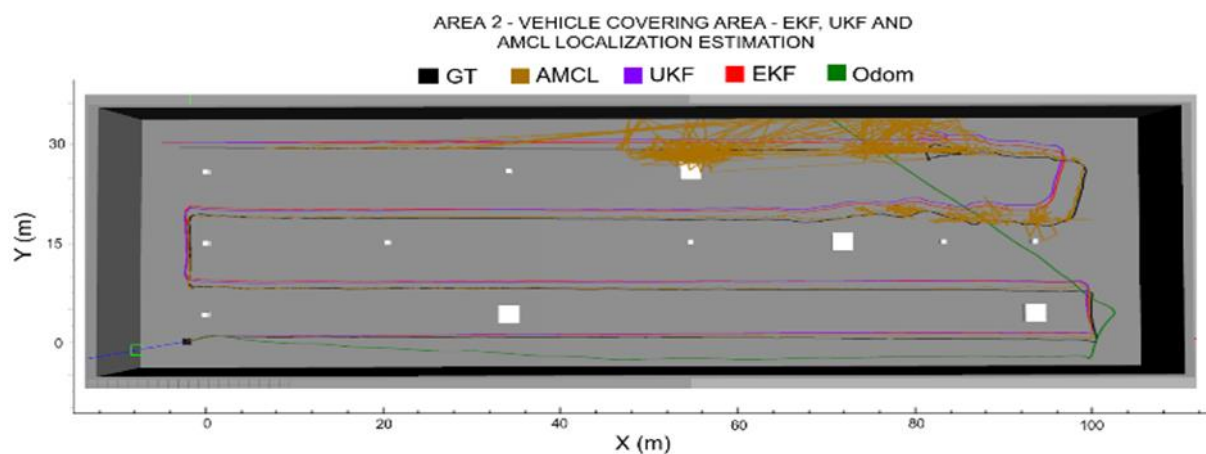
Os resultados obtidos pelo filtro AMCL para o mapa H3.3 (Figura 32 a) não foram satisfatórios, e a posição estimada apresentou grandes erros até que o robô atingisse uma área com objetos mais característicos, o que ocorreu apenas para  $x > 60\text{m}$ . Antes dessa distância, a varredura a *laser* não conseguiu perceber nenhum dos elementos presentes no mapa, apenas a parede do lado direito, e a AMCL não pôde determinar corretamente a localização do robô. Ao aumentar o número de caixas no Mapa H3.4 (Figura 32 b), o ambiente se tornou mais estruturado e o filtro AMCL mais eficaz. É observado que, nesse mapa, sempre há, pelo menos, uma caixa dentro do alcance da varredura a *laser*. O único momento em que não foi possível realizar a identificação foi na região do canto superior direito da Figura 21 (b), onde não havia objetos suficientes para que a AMCL identificasse a posição do robô através da comparação do mapa.



Figura 32 – Ambiente estruturado (caixas brancas), caminho percorrido do robô (*Ground Truth* – Linha Preta) e localização estimada usando EKF (vermelho), UKF (Roxo), AMCL (Marrom e Laranja) e odometria de rodas (Verde) para o Mapa 1(a) e Mapa 2(b).



(a)



(b)

Fonte: autoria própria

O Quadro 8 mostra o erro máximo nos eixos x e y para cada um dos casos simulados. Conforme apresentado, os ambientes grandes com número reduzido de obstáculos e estrutura não são adequados para o filtro AMCL.

Quadro 8 – Erro máximo obtido pela odometria e por cada filtro em cada um dos percursos

Erro no eixo x estimado	Odometria de rodas	Filtro de Kalman estendido	Filtro de Kalman <i>unscented</i>	AMCL
Mapa H3.3 (Com curvas)	Se mantém perdido	0.3 m	0.4 m	Se mantém perdido
Mapa H3.4 (Com curvas)	Se mantém perdido	1.6 m	1.8 m	Se mantém perdido
Mapa H3.1 (Retilíneo)	0.01 m	0.01 m	0.01 m	1.1 m
Mapa H3.4 (Retilíneo)	0.01 m	0.01 m	0.01 m	0.8 m

Fonte: autoria própria

Os resultados mostraram que ambos os filtros de Kalman obtiveram valores mais precisos quando os caminhos percorridos são retilíneos. Quando as curvas são introduzidas, os filtros de Kalman não conseguiram compensar totalmente os possíveis deslizamentos das rodas e o uso do filtro AMCL pode melhorar o desempenho, desde que o ambiente seja estruturado o suficiente, ou seja, o ambiente tenha elementos suficientes dentro do alcance do sensor de percepção e os elementos não sejam posicionados de forma repetitiva. Para os casos apresentados neste trabalho com recursos esparsos e semelhantes, a localização AMCL não é recomendada e a estimativa de posição pelos dois filtros de Kalman obtém resultados melhores.

## 5 EXPERIMENTOS COM O MCCR

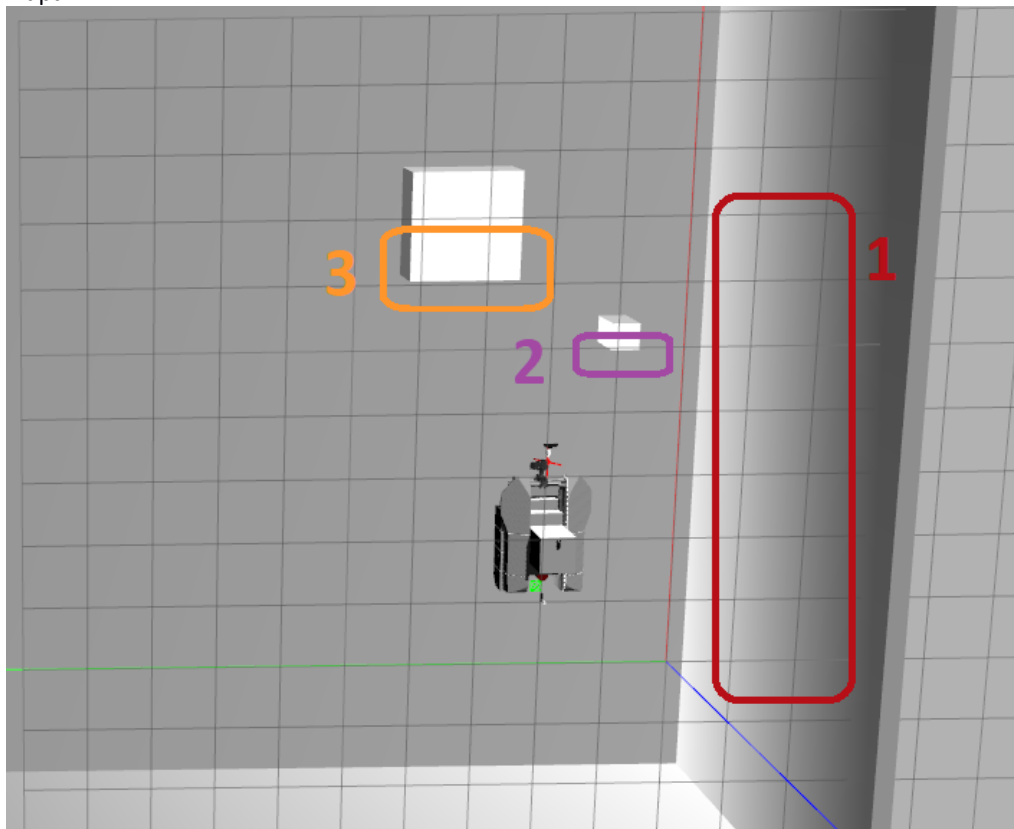
### 5.1 TESTES DE VERIFICAÇÃO DOS MODELOS DESENVOLVIDO PARA MCCR

#### 5.1.1 Conversão de dados de sonar para *PointCloud2*

##### 5.1.1.1 Testes em ambiente simulado

Para analisar o funcionamento da conversão dos dados dos sonares para *PointCloud2*, foram realizados alguns experimentos. No primeiro, o MCCR com sonares foi adicionado em um tanque com 2 objetos cúbicos de 50 cm de altura à frente dos sensores em um mapa de um tanque de 100 m por 30 m, como apresentado na Figura 19 e na Figura 33. Logo, a imagem no *Rviz*, contendo a mensagem do tipo *PointCloud2*, foi comparada à imagem original antes da conversão e processamento dos dados do sonar, analisando as posições e tamanho dos obstáculos.

Figura 33 – Representação do MCCR com sonar no tanque no Gazebo com objetos para comparação. As linhas finas nas cores azul, vermelha e verde representam o eixo inicial do mapa.



Fonte: autoria própria

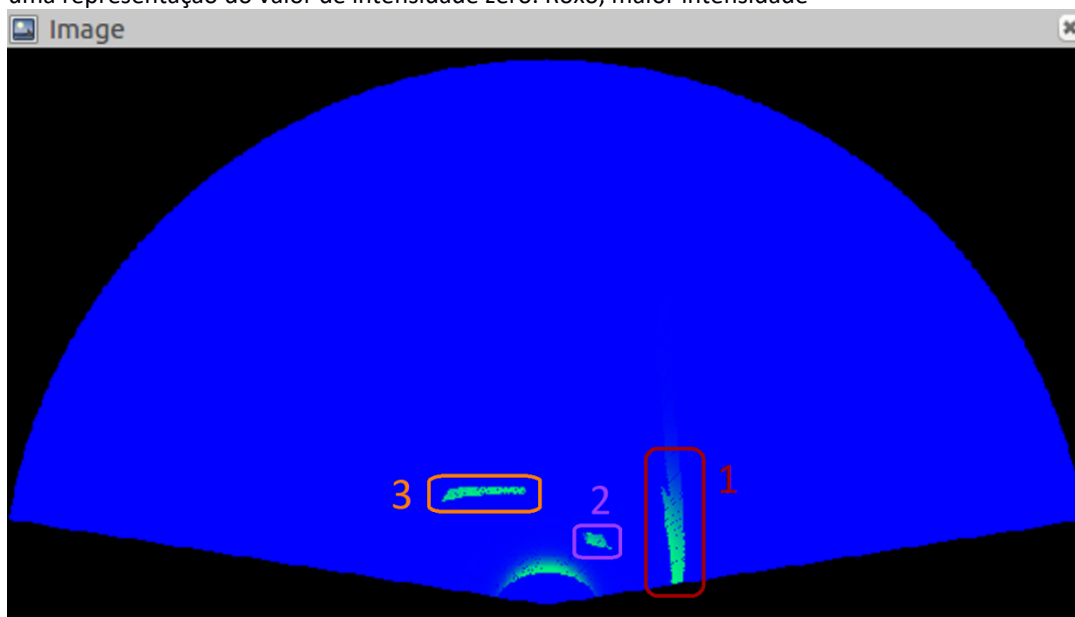
#### 5.1.1.1.1 RESULTADOS

A Figura 34 a) apresenta a imagem originária do modelo de sonar simulado do tipo *forward-looking*. Os pontos em azul claro mostram os valores de intensidade maior, mostrando onde há obstáculos, e azul escuro de intensidade menor. A Figura 34 b) mostra o resultado da conversão dos dados obtidos do sonar em *PointCloud2*. Os pontos com maior intensidade são apresentados com uma altura maior, valores maiores no eixo Z, e apresentam a coloração roxa. Os pontos com menor intensidade, onde a é altura próxima de zero, apresentam a coloração vermelha.

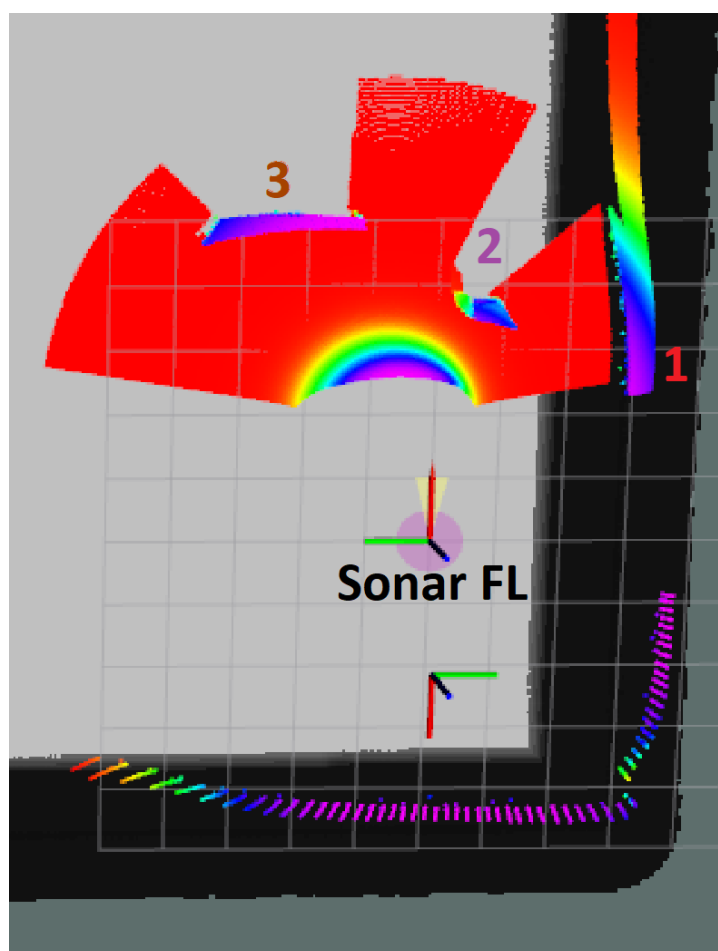
Ao comparar as imagens obtidas do sonar, quando em formato de *PointCloud2*, é possível numerar e comparar os objetos mostrados na Figura 34 a) e b) com a Figura 33. O objeto marcado no número 1 representa a parede à direita do veículo, número 2 a caixa de tamanho menor e número 3, a caixa de tamanho maior.

Outro ponto a ser analisado nas duas imagens é uma semicircunferência, logo à frente do sonar, que se mantém a todo momento, mesmo após a movimentação do veículo. Não é possível definir se é um erro dos equipamentos ou da estrutura do sonar, porém há a probabilidade de ser originário de “eco” da onda sonora no chão, devido à proximidade do sonar ao solo, retornando um valor de intensidade significativa.

Figura 34 – Visualização dos dados do sonar *forward-looking* no Rviz. a) Representação original pré-processamento. b) Representação pós-processamento no tipo de mensagem *PointCloud2*. Vermelho é uma representação do valor de intensidade zero. Roxo, maior intensidade



a)



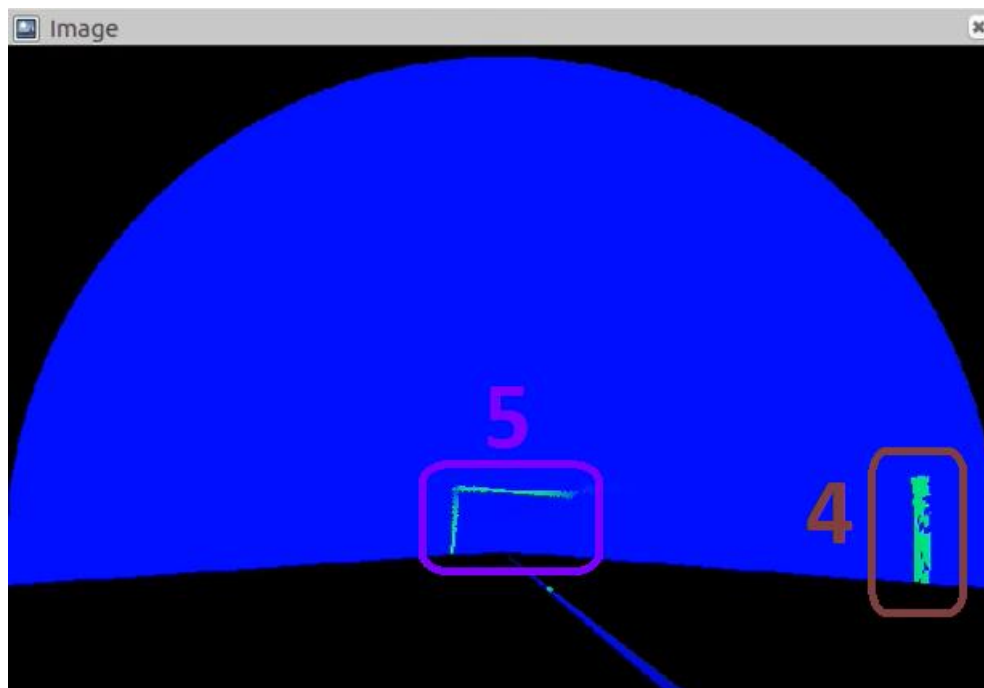
b)

Na Figura 35 a), é possível visualizar a imagem originária do sonar simulado do tipo *mecânico de feixe único*, descrito na seção 3.3.2. A Figura 35 b) mostra o resultado da conversão dos dados obtidos do sonar para PointCloud2. Como o sonar mecânico está virado para trás do veículo, a visualização da imagem em PointCloud2 é invertida.

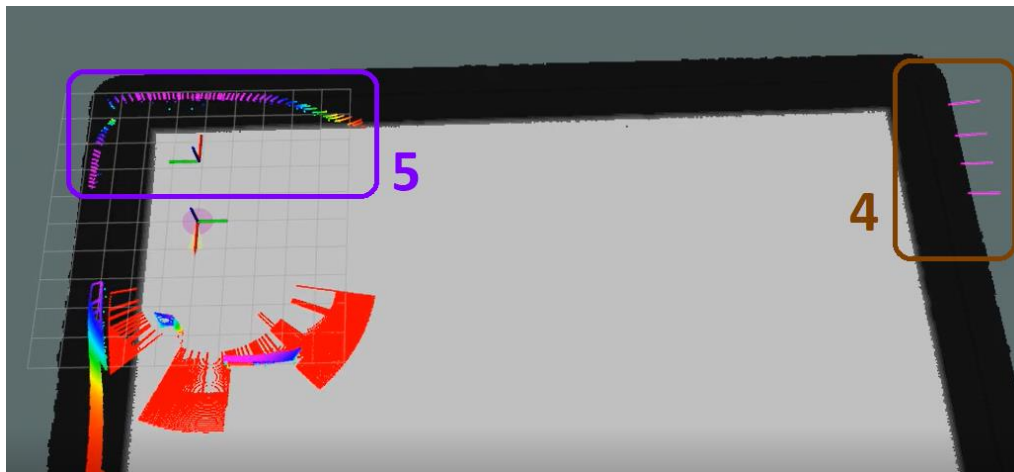
Na Figura 35, é possível visualizar alguns objetos. O número 4 seria a representação da parede à direita da imagem, enquanto o número 5 seria a parede que se encontra na parte traseira do veículo, podendo ser identificada pela presença de uma aresta da conexão entre as paredes. Além disso, os pontos da mensagem de PointCloud2 estão na mesma posição do início da parede, representado pelo mapa da Figura 35 b) na cor preta na parte superior e lateral da imagem, representado pelo número 4 e 5. Adicionalmente na Figura 34 b), com o sonar *forward-looking*, também é possível visualizar o limite da parede, no mapa em preto, como um obstáculo do sonar em *PointCloud2*.

Concluindo, a partir da simulação foi possível identificar e entender o formato dos dois objetos e da parede, separando-os por tamanho, forma e posição, a partir dos dados do *PointCloud2*.

Figura 35 – Visualização dos dados do sonar mecânico de feixe único no Rviz. a) Representação original pré-processamento. b) Representação pós-processamento no tipo de mensagem *PointCloud2*. Vermelho é uma representação do valor de intensidade zero. Roxo, maior intensidade



a)



b)

Fonte: autoria própria

#### 5.1.1.2 Testes em ambiente com imagens de um sensor real

Além dos testes simulados na seção anterior, para fins de validação dos sensores, o algoritmo de conversão para *PointCloud2* foi testado com o sonar *forwarding looking* da IxBlue, modelo M900-2250-130-Mk2 (TELEDYNE MARINE, 2021), em um tanque de 2 m de largura por 2 m de comprimento. Para isso foi realizado um teste em que um objeto de 8

cm de altura, similar ao visualizado na Figura 36, foi movimentado dentro do tanque apresentado na Figura 37, onde se encontravam o sonar e uma câmera para visualização do movimento. Os dados foram gravados e filtrados através do algoritmo desenvolvido. O objetivo desse experimento foi verificar que o sonar captura toda a movimentação do objeto e os resultados foram apresentados como imagens 3D de *PointCloud2*.

Foram analisadas duas situações: dados de *PointCloud2* sem e com a aplicação de um filtro passa-baixa. Os resultados foram comparados com a imagem obtida por uma câmera inserida dentro do tanque, apontando para a mesma direção que o sonar.

Nesse teste, o objeto realizou 3 movimentos a partir da posição inicial para verificar se era possível identificar sua posição quando na região central do tanque e no lado direito do tanque. As posições analisadas foram:

- Posição 0, o objeto inicia na parte direita e traseira do tanque;
- Posição 1, o objeto faz um movimento até o centro do tanque;
- Posição 2, o objeto faz um movimento até a parte direita e central do tanque;
- Posição 3, o objeto faz um movimento até a parte direita e traseira do tanque.

Posteriormente foram realizados também alguns testes com o sonar mecânico fora de ambiente de simulação, no tanque laboratorial de 2 m x 2 m. No entanto, devido à presença de ecos, não foi possível a visualização de objetos pelo sonar mecânico. Devido à pandemia, os testes com o sonar mecânico em ambiente laboratorial foram interrompidos.

Figura 36 – Elo de corrente similar utilizada no experimento



Fonte: ELASTOBOR (2019)



Figura 37 – Tanque utilizado para testes, simulando algumas estruturas do veículo MCCR



a)



b)

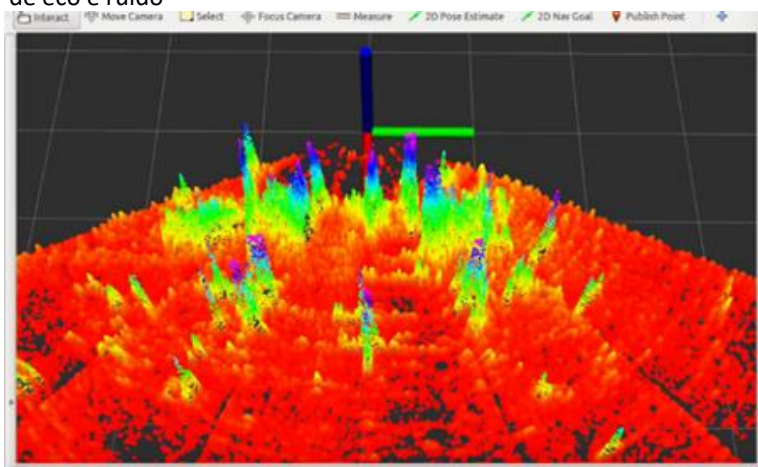
Fonte: autoria própria

#### 5.1.1.2.1 RESULTADOS

As figuras abaixo apresentam a sequência de um vídeo gravado, da posição 1 até 3 e o objeto é marcado por um retângulo de cor branca.

Durante a utilização do sonar, não foi possível visualizar o objeto, pela grande presença de ruído e eco das ondas sonoras, como visualizado na Figura 38, devido ao tamanho reduzido do tanque. Para melhorar a visualização, foi criada uma rotina que permitiu remover o eco da mensagem, antes da conversão para *PointCloud2*. Essa rotina gravou a primeira imagem do sonar com valores de intensidade, sem o objeto, e a partir disso subtraiu o valor de intensidade gravado com os valores atuais de intensidade nos mesmos pontos. Logo, os pontos que mantinham valores de intensidade constantes ficaram iguais a zero. Adicionalmente, para aumentar a velocidade de processamento e melhorar a visualização, os pontos com valor zero de intensidade não foram processados.

Figura 38 – Imagem original do sonar em *PointCloud2* com a presença de eco e ruído

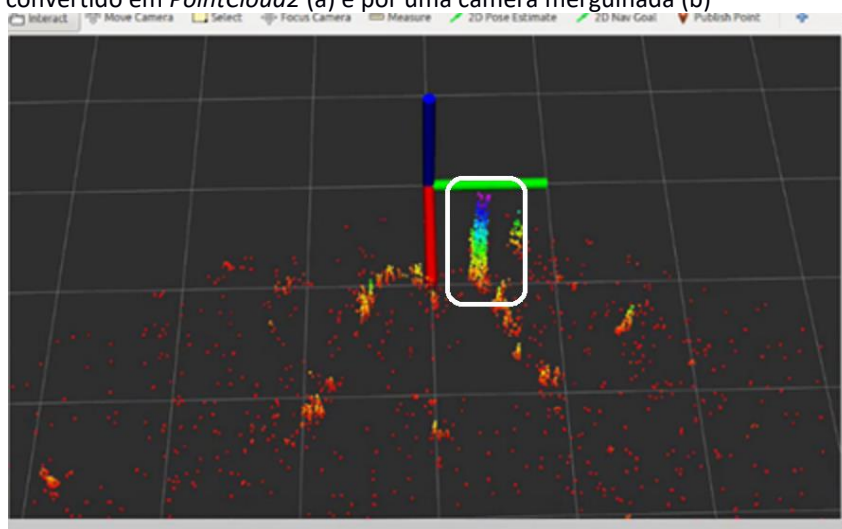


Fonte: autoria própria

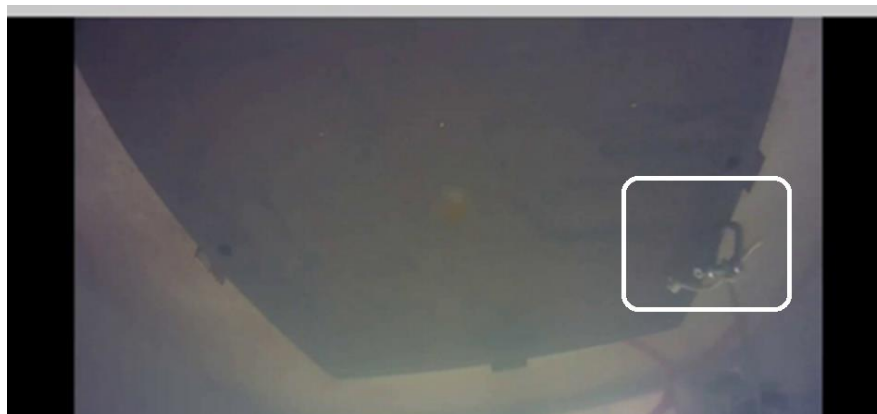
Após a implementação do filtro de eco, foi feito um vídeo do tanque com um objeto adicionado em diversos pontos, permitindo realizar a comparação da imagem da câmera com os dados obtidos pelo sonar. As Figura 39, Figura 40, Figura 41 e Figura 42 são parte de uma sequência de movimentos gravados direto da tela do computador e originários de um vídeo. Nessas mesmas figuras, é possível visualizar 3 cilindros que representam o eixo X, Y e Z e onde eles se cruzam é onde se localiza o dispositivo (sonar).

O objeto foi iniciado na posição traseira e direita, como visualizado pela câmera em Figura 39 b). Na Figura 39 a) é possível ver que há a presença de pontos mais altos e com valor de intensidade maior, esses se referem ao mesmo objeto visualizado pela câmera. Adicionalmente, o objeto se encontra em um ponto mais próximo da ponta do cilindro vermelho, indicando que ele está mais longe do sonar e à direita dele.

Figura 39 – Visualização do objeto na posição 1 dentro do tanque pelo sonar, convertido em *PointCloud2* (a) e por uma câmera mergulhada (b)



a)

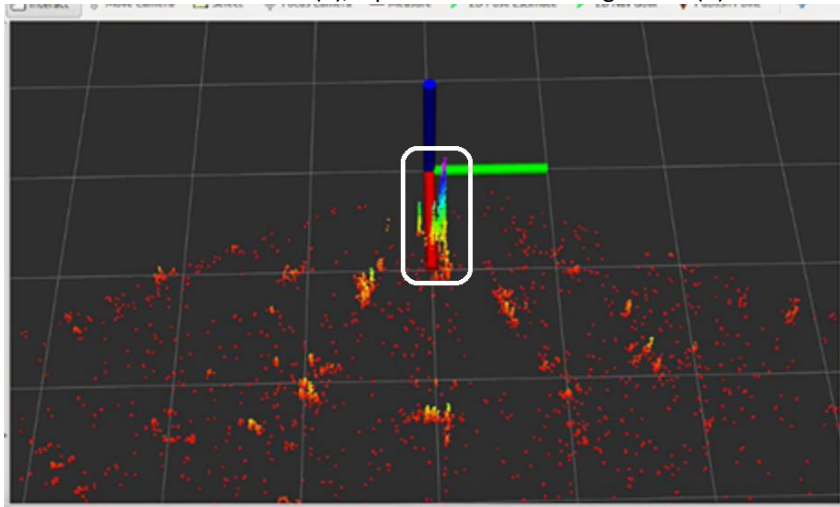


b)

Fonte: autoria própria

Na Figura 40, o objeto se movimentou até a posição 2, centro do tanque, como visualizado pela câmera em b). Ao comparar com a janela com *PointCloud2* em a), há uma grande presença de pontos mais altos (com maior intensidade); logo, com maior valor de intensidade. Adicionalmente, o objeto se encontra no centro do cilindro vermelho do eixo, indicando que está numa posição mais central e próxima do sonar.

Figura 40 – Visualização do objeto na posição 2 dentro do tanque pelo sonar, convertido em *PointCloud2* (a), e por uma câmera mergulhada (b)



a)

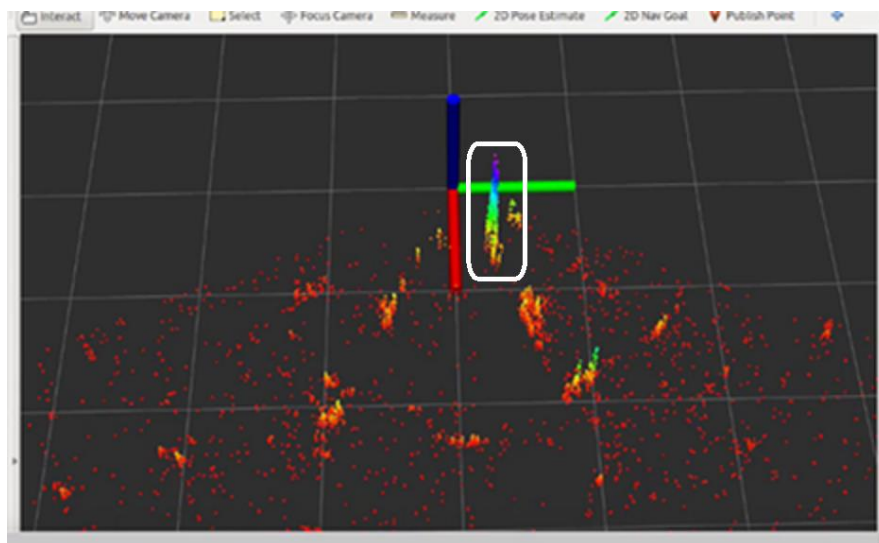


b)

Fonte: autoria própria

Na Figura 41, o objeto se movimentou até a posição 3, direita e centro do tanque, como visualizado pela câmera em b). Ao comparar com a janela com *PointCloud2* em a), há uma grande presença de pontos mais altos; logo, com maior valor de intensidade. Adicionalmente, o objeto se encontra no centro e à direita do cilindro vermelho do eixo, indicando que está numa posição mais central, próxima e à direita do sonar.

Figura 41 – Visualização do objeto na Posição 3 dentro do tanque pelo sonar, convertido em *PointCloud2* (a), e por uma câmera mergulhada (b)



a)

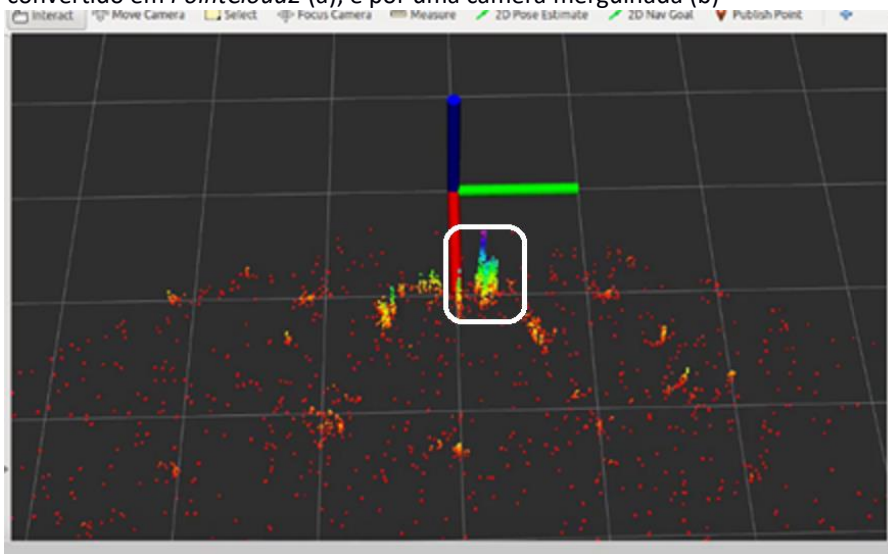


b)

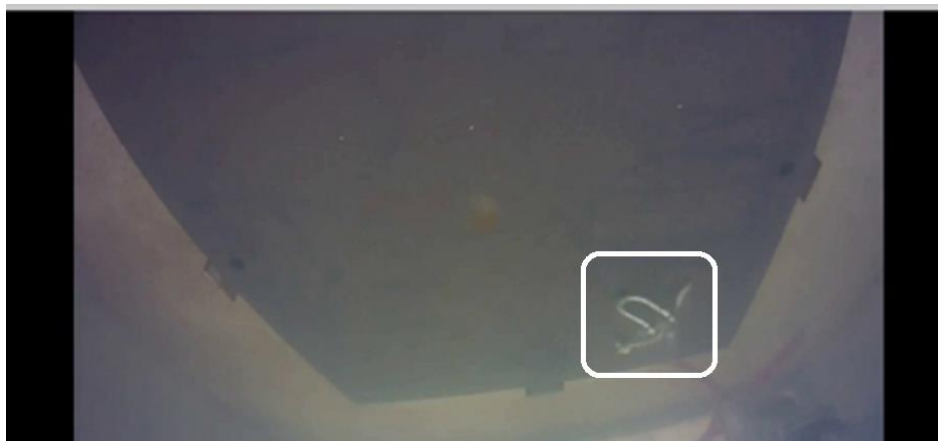
Fonte: autoria própria

Na Figura 42, o objeto se movimentou até a posição 4, direita e traseira do tanque, como visualizado pela câmera em b). Ao comparar com a janela com *PointCloud2* em a), há uma grande presença de pontos mais altos; logo, com maior valor de intensidade. Adicionalmente, o objeto se encontra na parte traseira e à direita do cilindro vermelho do eixo, indicando que está numa posição mais distante e à direita do sonar.

Figura 42 – Visualização do objeto na posição 4 dentro do tanque pelo sonar, convertido em *PointCloud2* (a), e por uma câmera mergulhada (b)



a)



b)

Fonte: autoria própria

Como conclusão, foi possível identificar a posição do objeto em relação ao sonar real, mesmo em movimento. Ao utilizá-lo no MCCR, seria possível identificar alguns objetos que estão inseridos no casco, evitando que o robô entre em contato e não atrapalhe o movimento. Dessa forma, foi verificado que essa ferramenta, com essa rotina, pode ser utilizada na identificação do ambiente para a localização e de grande importância para futuras aplicações de desvio de obstáculos.

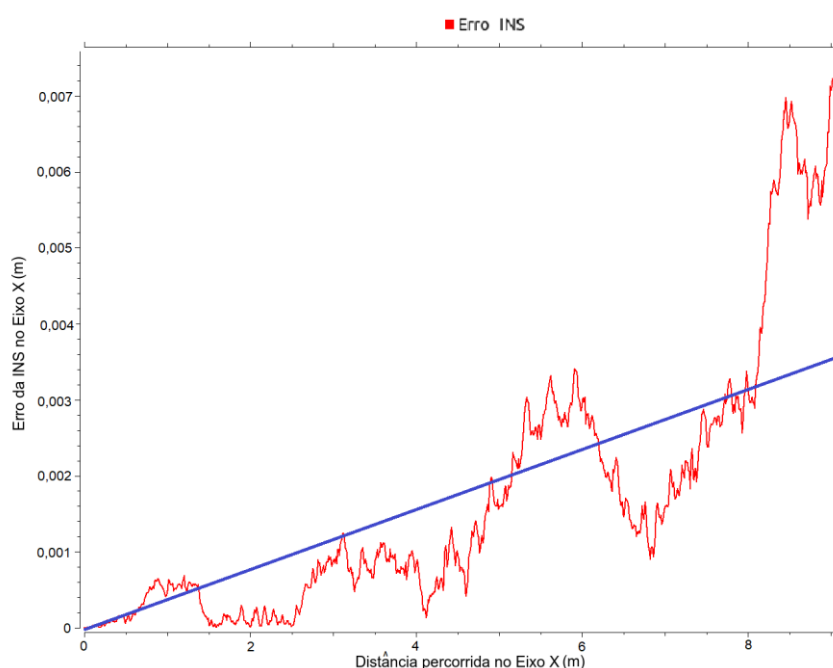
### 5.1.2 Testes de modelo para INS

Para a realização dos testes com o MCCR houve a necessidade de avaliar o uso do sensor inercial de navegação, que promete estimar localização com menor erro, conforme discutido anteriormente. Após as devidas configurações de parâmetros de erro definidos pelo *datasheet do iXBlue Rovins Nano*, foi desenvolvido um algoritmo para teste do INS em código executável no ROS, como apresentado na seção 3.3.3. O veículo *MCCR* fez um percurso em linha reta no eixo  $x$ , obtendo dados de posição sem erro do ambiente 3D do Gazebo e o mapa do experimento M1 sem a adição de obstáculos, Figura 19. Utilizando essas informações, um gráfico do erro associado por distância percorrida foi criado com o objetivo de verificar a curva do erro esperado, utilizando um script para cálculo do erro em determinado valor de distância percorrido pelo robô.

#### 5.1.2.1 Resultados

A Figura 43 mostra o erro de posição modelado para o INS. Esse erro cresce e decresce baseado na soma do ruído gaussiano, cuja média é de 0,04 % da distância percorrida pelo veículo, representada pela linha azul. Essa curva mantém uma tendência de crescimento em relação ao tamanho do percurso, como esperado.

Figura 43 – Erro de posição modelado para sistema inercial de navegação. Linha em azul representa o possível erro do modelo, de acordo com a distância percorrida



Fonte: autoria própria

## 5.2 EXPERIMENTOS COM ALGORITMOS DE FUSÃO COM O MCCR

Com os ambientes, robôs e módulos desenvolvidos, foi realizada uma série de experimentos para testar o uso dos modelos desenvolvidos com os algoritmos de filtro de Kalman e AMCL. O experimento M1 verificou a integração do modelo do INS com os filtros de Kalman. O experimento M2 verificou a integração dos dados dos sonares, utilizando os dados de *PointCloud2*, com o algoritmo AMCL. Os 2 experimentos foram realizados no tanque que simulou o fundo do navio apresentado em 3.3.1.

### 5.2.1 Experimento M1: Utilização do modelo de INS com os algoritmos de filtros de Kalman

No experimento M1 foi realizada a fusão dos dados de odometria com os obtidos pelo modelo do INS. Valores de localização sem erro, ou *ground truth*, e os valores estimados para 2 algoritmos de fusões de sensores, filtros de Kalman estendido e Kalman *unscented* foram capturados durante o movimento do robô no ambiente do tanque desenvolvido. Os sensores fundidos em cada algoritmo e as variáveis selecionadas para estimar a posição são apresentados no Quadro 9.

Quadro 9 – Algoritmos, sensores, variáveis e parâmetros para fusão para experimento M1

<b>Algoritmos – Experimento M1</b>	<b>EKF</b>	<b>UKF</b>
<b>Sensores fundidos</b>	- INS; - Odometria de rodas	- INS; - Odometria de rodas
<b>Variáveis selecionadas para estimar posição</b>	- Velocidade linear; - Orientação; - Aceleração;	- Velocidade linear; - Orientação; - Aceleração;
<b>Valores de covariância de processo para configuração dos algoritmos</b>	- Posição: 0,1 - Velocidade: 0,01	- Posição: 0,1 - Velocidade: 0,01
<b>Valores de covariância inicial para configuração dos algoritmos</b>	- Posição: $10^{-4}$ - Velocidade: $10^{-8}$	- Posição: $10^{-4}$ - Velocidade: $10^{-8}$

Fonte: autoria própria



Nesse experimento, o veículo realizou um trajeto em linha reta por 10 m, um giro no sentido anti-horário com aproximadamente 45° e outro no sentido horário também de 45°. Os giros ocorreram em 2 momentos, em torno de ~3 m e em torno de ~7 m da distância percorrida. Foram realizados 2 experimentos, variando os valores dos parâmetros da matriz de covariância de posição e velocidade da mensagem da odometria de rodas, indicado no Quadro 10, com o objetivo de verificar a influência da covariância.

Quadro 10 – Valores de variância da mensagem de odometria utilizados em cada experimento. Experimento M1 com MCCR

<b>Experimento M1 com MCCR.</b>		
<b>Nº</b>	<b>Inicial Cov. Pose em X</b>	<b>Inicial Cov. Velocidade em X</b>
M1.1	$1,0 \cdot 10^{-5}$	$1,0 \cdot 10^{-6}$
M1.2	$1,0 \cdot 10^{-25}$	$1,0 \cdot 10^{-26}$

Fonte: autoria própria

Como o percurso foi feito manualmente, havia uma incerteza no movimento e o escorregamento poderia variar de uma rodada para outra. Assim, cada condição analisada foi repetida três vezes para entendimento da variabilidade. A diferença entre a posição estimada e a *ground truth* ou erro foi calculada e os resultados foram analisados de forma gráfica no eixo do movimento.

### **5.2.1.1 Resultados - Experimento M1**

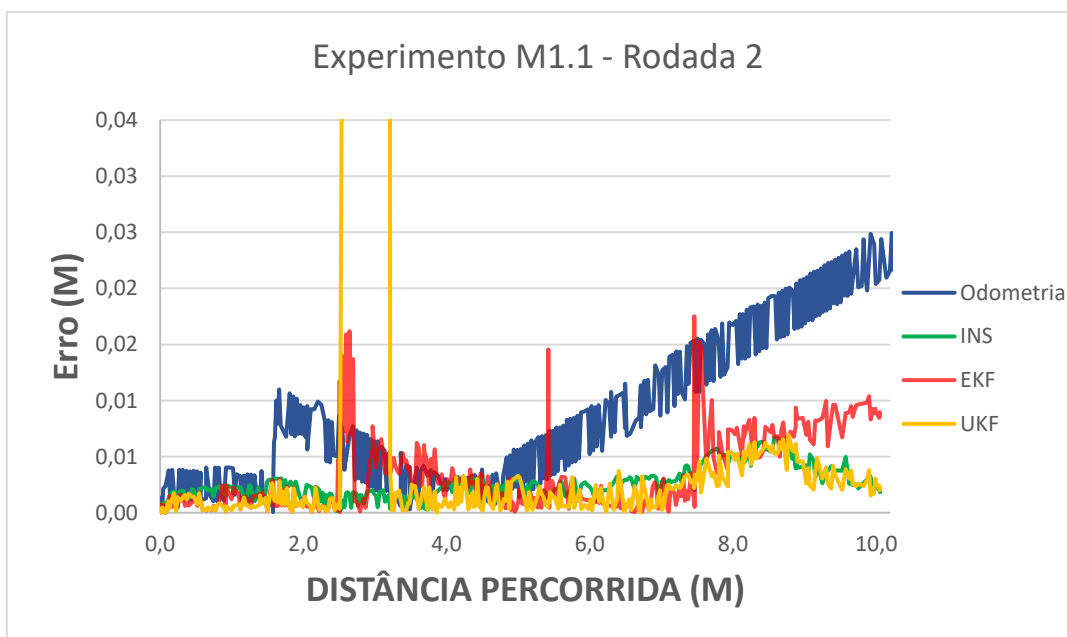
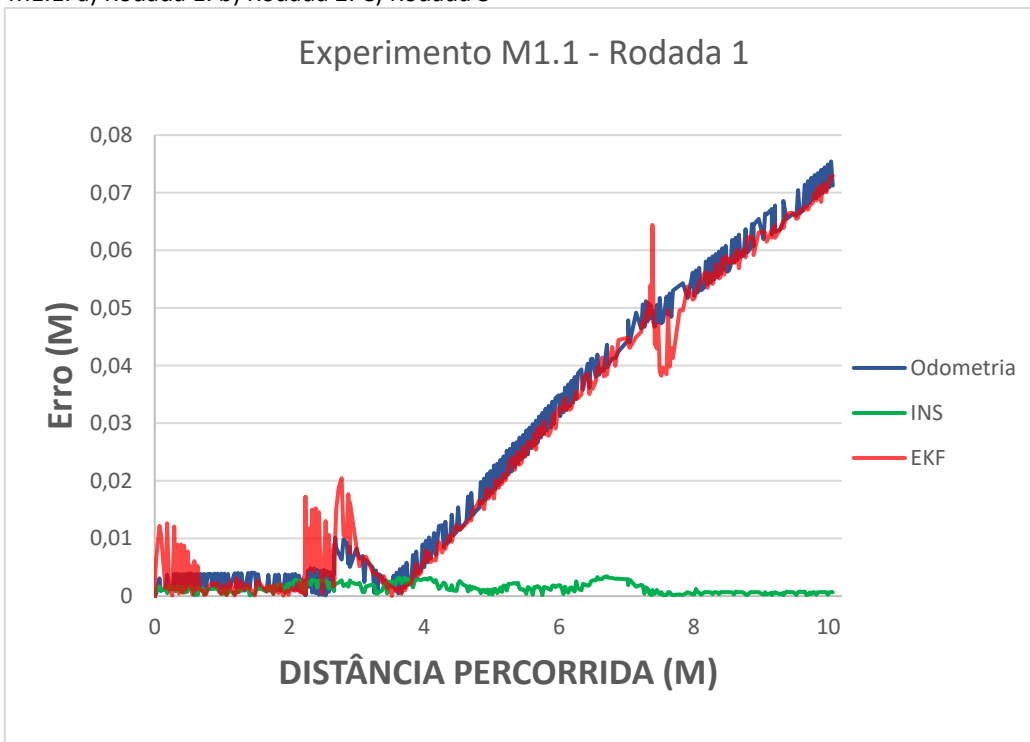
As Figura 44 e Figura 45 a), b) e c) mostram o erro obtido na estimação de posição no eixo X durante o percurso de 10 metros para o experimento M1.1 e M1.2 para as três a), b) e c) mostram o erro obtido na estimação de posição no eixo X durante o percurso de 10 metros para o experimento M1.1 e M1.2 para as três a), b) e c) mostram o erro obtido na estimação de posição no eixo X durante o percurso de 10 metros para o experimento M1.1 e M1.2 para as três rodadas efetuadas. As curvas de odometria refletem o escorregamento nos giros, como esperado, com um impacto maior no segundo giro. O efeito desse escorregamento na estimativa da posição varia significativamente de uma rodada para outra. Por exemplo, na rodada M1 a) a odometria perdeu completamente a posição, com aumento contínuo do erro, diferentemente do que ocorreu nas outras rodadas. O INS não sofreu tanto com as curvas, mesmo assim ainda pôde apresentar um

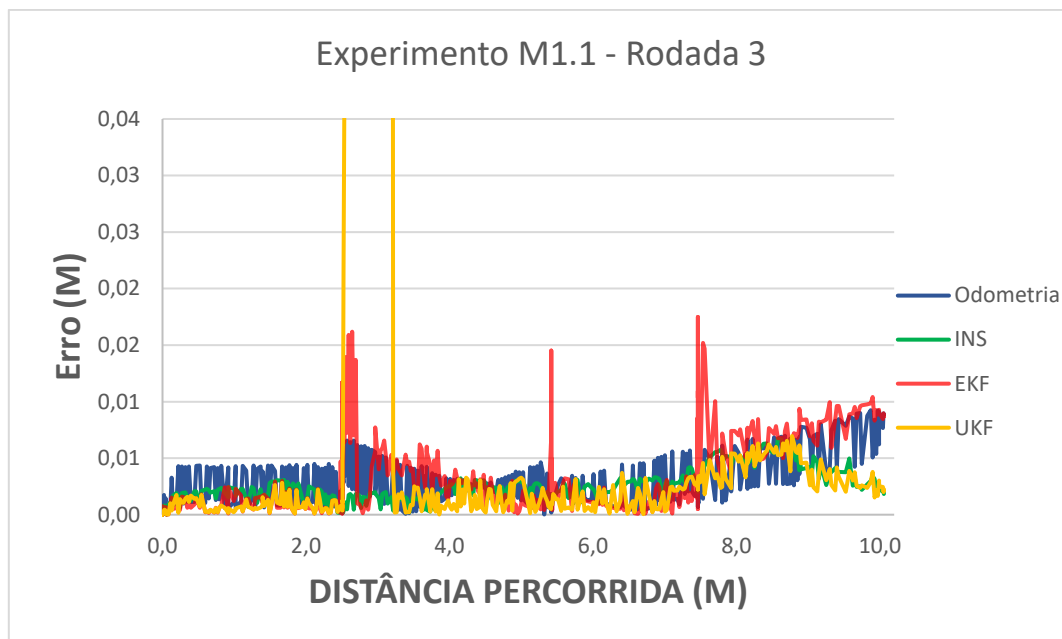
aumento no erro durante o segundo giro. Esse erro foi maior que o primeiro, pois o erro do INS aumentou com a distância percorrida (Figura 43) e o segundo giro ocorreu numa distância maior.

Ao verificar as curvas da fusão de sensores para o EKF nas 3 rodadas, estas seguiram os valores estimados de posição ou a tendência de crescimento da mensagem de odometria. Isso ocorreu provavelmente devido ao alto valor de covariância dado à mensagem de odometria. Ao comparar a rodada (a) com as outras, na qual a odometria manteve um erro crescente, o primeiro escorregamento já fez a curva de EKF seguir os valores de odometria. Isso possivelmente ocorreu em (a), devido a um escorregamento mais intenso, enquanto para (b) e (c) houve um impacto maior no segundo escorregamento.

Ao avaliar o UKF nas rodadas (b) e (c), as curvas seguiram o INS depois de um período de recuperação, gerando erros menores que o EKF ao final do percurso.

Figura 44 – Erro associado à estimação de posição no eixo X para 3 rodadas do experimento M1.1. a) Rodada 1. b) Rodada 2. C) Rodada 3





c)

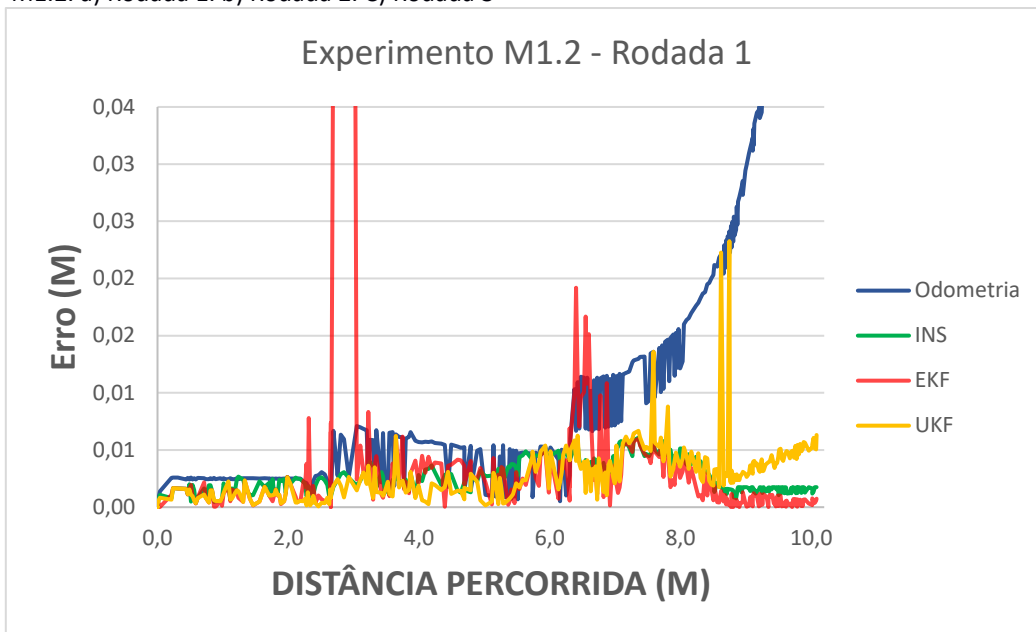
Fonte: autoria própria

Para o caso M1.2 apresentado na Figura 45, é possível verificar que em (a) e (b) o erro da estimativa de posição pela odometria cresceu após cada giro. Em (c), o escorregamento não foi tão acentuado e o crescimento do erro da estimativa da posição foi significativamente menor que nas outras rodadas. Nota-se que a variação da estimativa de posição pelo INS entre as três rodadas é muito menor do que para a odometria, porque ela foi menos afetada pelo escorregamento.

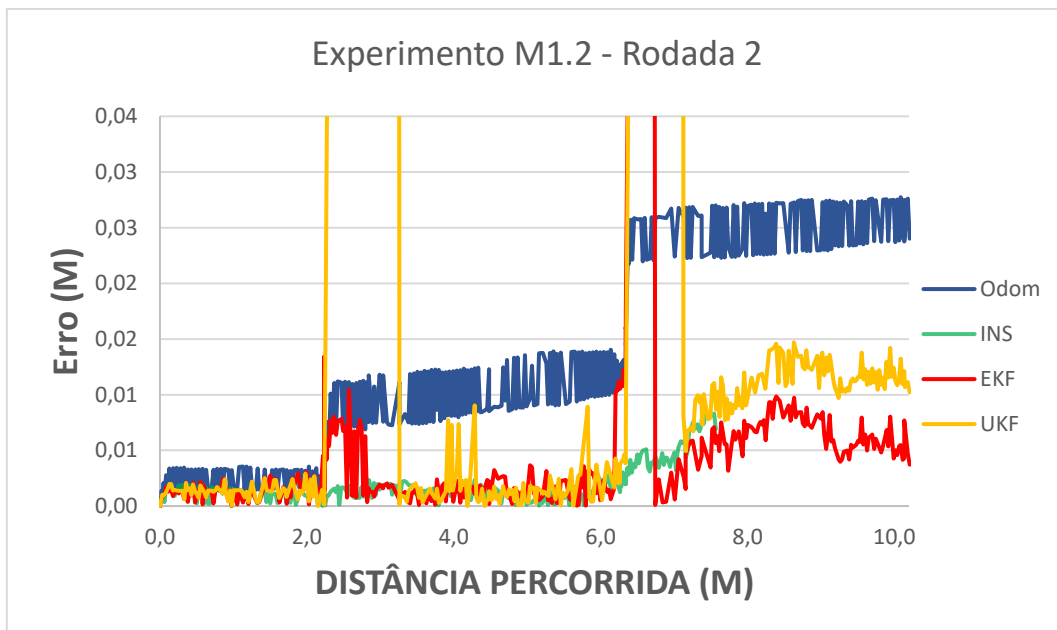
Nesses casos em que a covariância da odometria foi baixa, o valor estimado pelo algoritmo de fusão EKF conseguiu identificar os erros de escorregamento da odometria e após o período de recuperação seguiu a estimativa de posição do INS, nas rodadas (a) e (b). A odometria ainda ajudou na fusão e permitiu que o erro obtido com a EKF fosse menor que o do INS ao final do percurso. Nesses dois casos, a estimativa de posição pela EKF foi melhor do que para a UKF.

Ao avaliar a rodada (c), os escorregamentos não foram tão severos e os erros da odometria foram menores. Neste caso, o EKF reconheceu os escorregamentos, mas não melhorou o valor estimado de posição após o período de recuperação. Neste caso em particular, o UKF respondeu melhor que o EKF.

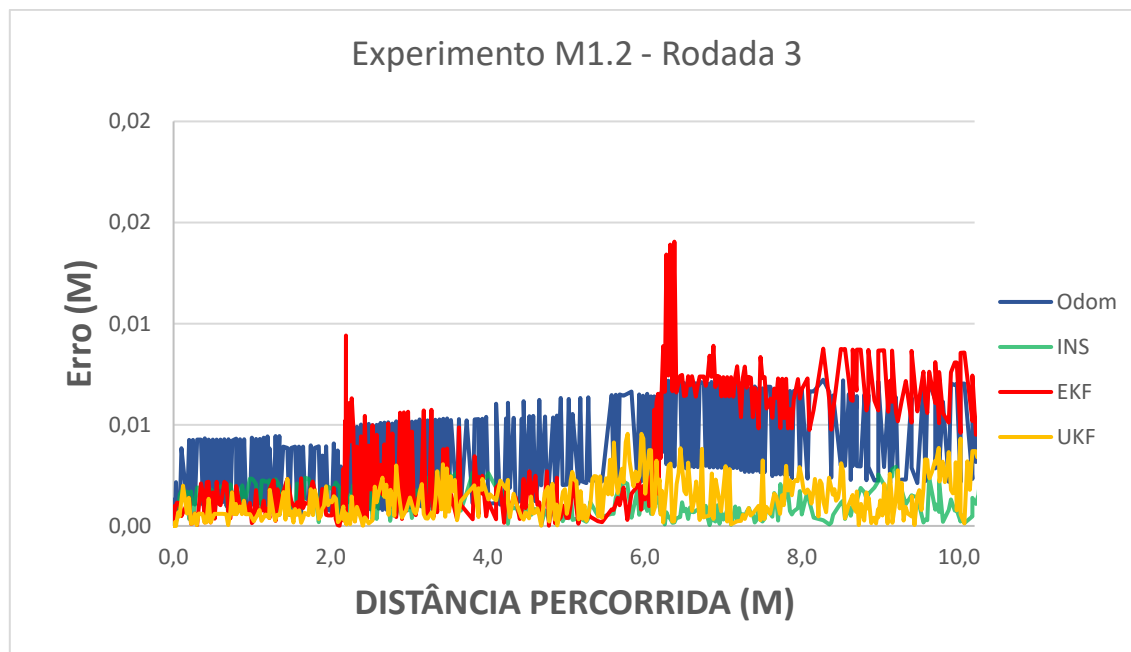
Figura 45 – Erro associado à estimação de posição no eixo X para 3 rodadas do experimento M1.2. a) Rodada 1. b) Rodada 2. C) Rodada 3



a)



b)



c)

Fonte: autoria própria

A variabilidade do escorregamento não permitiu verificar a influência da covariância da mensagem da odometria de forma sistemática. Apesar disso, essas simulações demonstraram que o modelo de INS pôde ser fusionado com os dados de odometria utilizando ambos os filtros de Kalman.

### 5.2.3 Experimento M2: Utilização dos dados do sonar para localização com o algoritmo de filtro de partículas AMCL

Para este experimento foi utilizado o sistema desenvolvido para conversão de dados de sonar para *PointCloud2*, sendo parte essencial do trabalho verificar seu funcionamento com o filtro de partículas AMCL do pacote *robot\_localization* do ROS. Os dados de entrada utilizados neste trabalho foram do sonar mecânico convertidos para *PointCloud2*, por manter características semelhantes para a visualização de um ambiente em 3 dimensões e pode ser utilizado como entrada para outros algoritmos, como de desvio de obstáculos. Posteriormente, essa mensagem foi convertida para o tipo *laser\_scan* do ROS, definindo um parâmetro de altura mínima, como uma barreira. Essa etapa foi necessária, pois o algoritmo AMCL do pacote *robot\_localization* utilizou o tipo de mensagem de *laser\_scan* como entrada.

Adicionalmente, foi utilizado um ambiente que simulou o fundo do navio de forma simplificada com blocos e paredes (bordas do navio), que puderam ser utilizados como referências visuais e ser comparados com o mapa apresentado na Figura 19 em 3.3.1.

O parâmetro de número de partículas foi variado de modo sequencial, começando com um número baixo e aumentando. Para o parâmetro de *laser-points* ou pontos de feixe de *laser* foi utilizado o valor padrão de 300 pontos máximos utilizados para o AMCL. O Quadro 11 resume os dados utilizados nos experimentos.

Quadro 11 – Experimentos M2 com MCCR e a variação dos valores dos parâmetros para o AMCL

<b>Experimento M2 com MCCR.</b>		
<b>Nº</b>	<b>Número de partículas máximas</b>	<b>Número de pontos de feixe de <i>laser</i></b>
M2.1	500	300
M2.2	1000	300
M2.3	3000	300

Fonte: autoria própria

Tanto o parâmetro distância máxima da conversão do sonar quanto o *range* máximo para o AMCL foram definidos em 30 m, tamanho aproximado da largura do tanque. Isso significa que apenas os dados do sonar com distância <30 m foram utilizados, com os demais sendo descartados.

O percurso executado nestes experimentos foi mostrado na Figura 19, sendo retilíneo por ~80m, fazendo dois giros sequenciais de 90 graus para contornar uma caixa e voltando ~30-40m em movimento retilíneo em sentido oposto ao inicial.

### **5.2.3.1 Resultados - Experimento M2**

Os erros de estimativa da localização no eixo x são apresentados no Quadro 12, para a odometria e a estimativa para o AMCL, usando a mensagem do tipo PointCloud2 convertido em *LaserScan*, sendo (a) os valores obtidos ao fim do percurso de 120 m e (b) o erro máximo durante o percurso.

Nota-se que os erros de estimativa de odometria são bastante variáveis, com os valores comparáveis aos obtidos nos experimentos anteriores. Isso porque, apesar de a distância ser maior, o percurso é mais retilíneo e a maior contribuição ao erro ainda vem dos escorregamentos durante os dois giros para a volta de 180°. Em alguns experimentos,

o erro no percurso de volta compensou o erro do giro, reduzindo o erro de estimativa ao final do percurso, como exemplificado na Figura 46 b) e Figura 47 b).

Quadro 12 – Resultados de experimentos M2 com erro na posição de 120 m

<b>Experimentos 3 com MCCR</b>				
<b>Percurso de 120 m</b>				
Nº	Erro ao fim do percurso (m)		Parâmetros	
	ODOM	AMCL	Número máximo de partículas (AMCL)	Número de pontos de feixe de laser
<b>M2.1</b>	0,4179	2,7658	500	300
<b>M2.2</b>	0,0282	0,7589	1000	300
<b>M2.3</b>	0,04894	0,7431	3000	300

Fonte: autoria própria

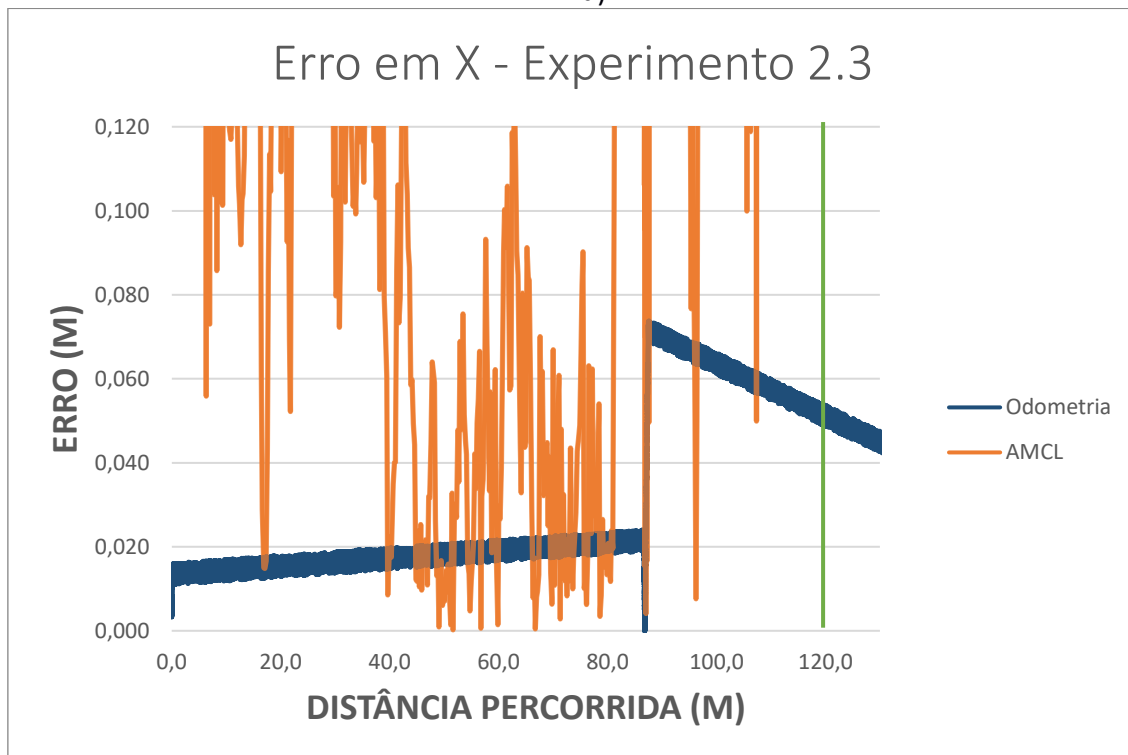
O melhor resultado para o AMCL foi o caso do experimento M2.3, cujo gráfico do erro de estimativa da posição é apresentado na Figura 46 a) com um zoom na Figura 46 b). É possível verificar que a curva dos dados para o AMCL se manteve com valores mais baixos até aproximadamente ~80 m, quando o veículo terminou a curva no último obstáculo. Como pode ser visto no mapa da Figura 19, há alguns objetos na região de visualização do sensor, somente a parede do tanque atrás. Neste momento, os valores de erro para o AMCL aumentam para valores de erro próximo de 1 m. Já na Figura 47, o erro da estimativa de posição para o AMCL oscila em torno de 1 m durante o percurso.



Figura 46 – Erro associado à estimação de posição no eixo X do experimento M2.3. A linha verde representa o ponto de 120m . a) Gráfico com escala do erro maior. b) Gráfico com escala do erro menor.



a)



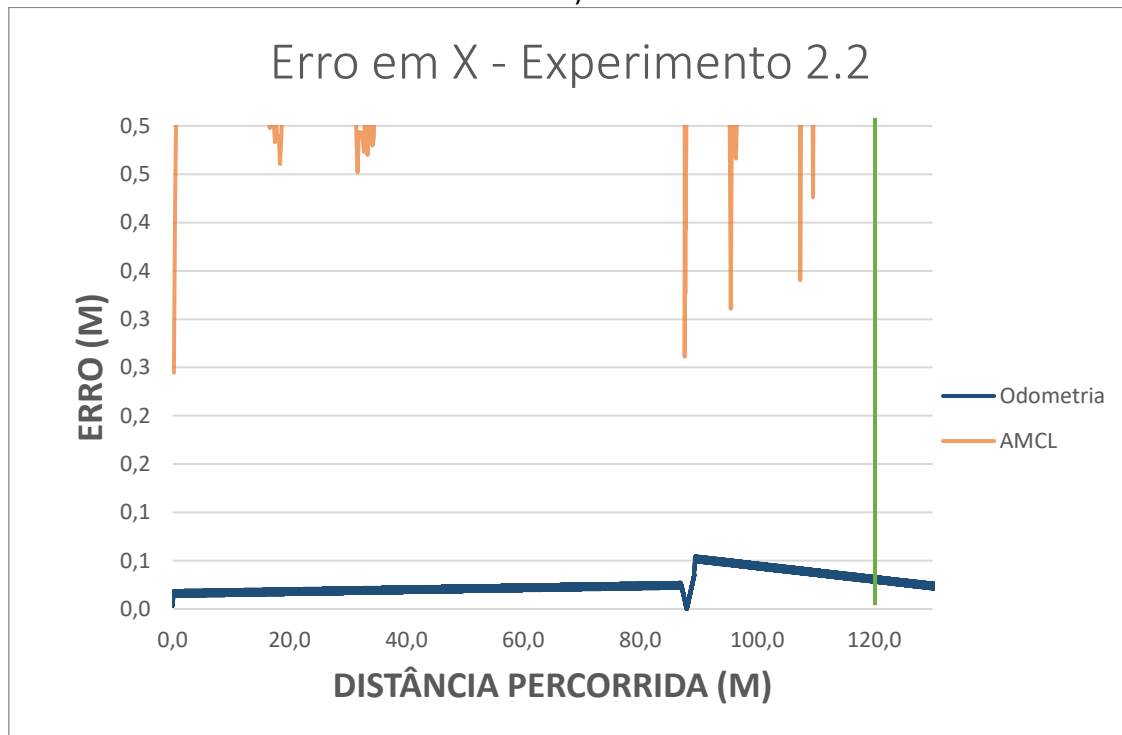
b)

Fonte: autoria própria

Figura 47 – Erro associado à estimação de posição no eixo X do experimento M2.2. A linha verde representa o ponto de 120m. a) Gráfico com escala do erro maior. b) Gráfico com escala do erro menor.



a)

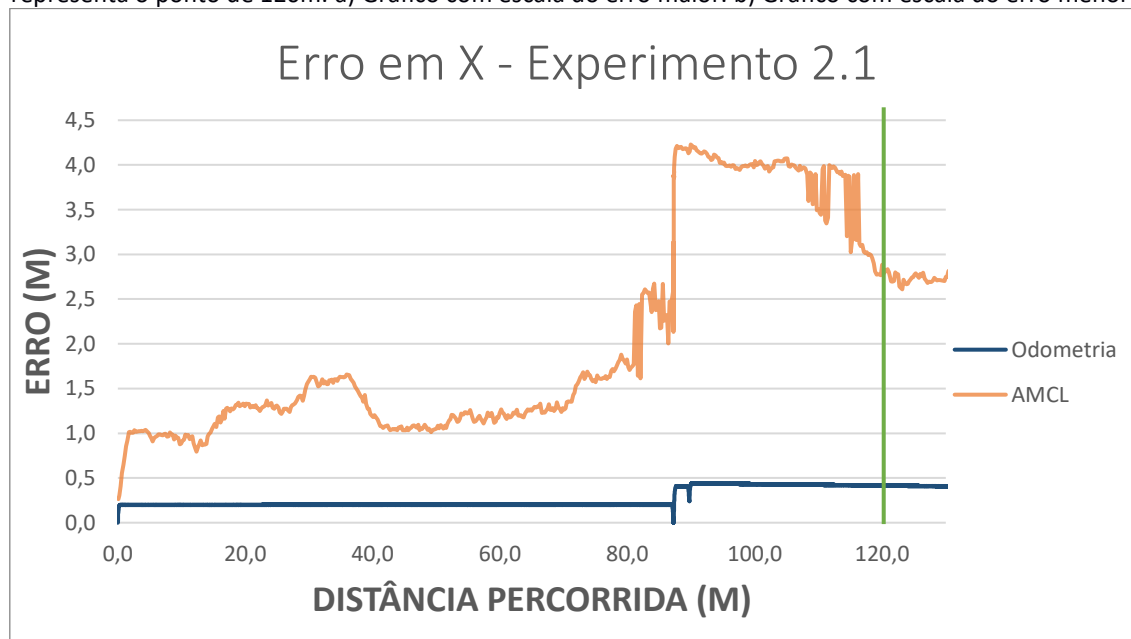


b)

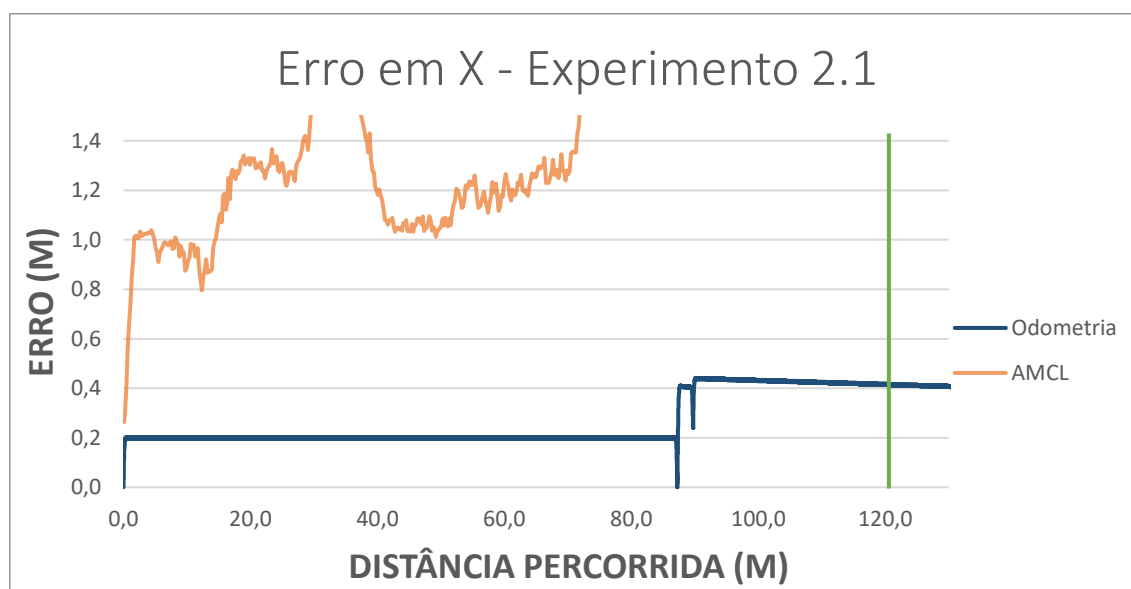
Fonte: autoria própria

A Figura 48 mostra o caso M2.1 onde o erro de odometria foi o maior e o número de partículas foi menor. Neste caso o AMCL apresentou também os maiores erros, mas não é claro qual desses fatores foi responsável por esse resultado.

Figura 48 – Erro associado à estimação de posição no eixo X do experimento M2.1. A linha verde representa o ponto de 120m. a) Gráfico com escala do erro maior. b) Gráfico com escala do erro menor.



a)



b)

Fonte: autoria própria

Os erros da estimativa de posição pelo AMCL obtidos não foram satisfatórios ao comparar com a odometria, ao utilizar os dados dessa forma. Um estudo mais aprofundado

é necessário para determinar se a baixa eficiência foi devido à falta de otimização de outros parâmetros ou uma limitação do ambiente menos estruturado, com pouca informação ou repetição de barreiras do fundo do FSPO. Se este último for o caso, os sonares ainda serão úteis na localização de obstáculos, mas não na localização do robô. Essa etapa demandaria mais tempo e rodadas adicionais, que foram impedidas devido à pandemia.

## 6 CONCLUSÃO

O SENAI CIMATEC está desenvolvendo um robô para limpeza e inspeção de plataformas de produção de petróleo do tipo FPSO, denominado de MCCR (*MEC Combi Crawler Robot*). Um dos grandes desafios desse robô é o sistema de localização adequado ao ambiente submarino.

Neste trabalho foram desenvolvidos modelos para a simulação do sistema de localização do MCCR e algoritmos para a realização de fusão de sensores com *framework* ROS no ambiente de simulação Gazebo.

Inicialmente, para a familiarização dos modelos de localização e algoritmos de fusão, foram estudados o funcionamento de sensores como odometria de rodas e o IMU do robô Husky. Para tanto, utilizou-se o software de simulação 3D, Gazebo, que permite desenvolver um ambiente virtual com robôs e testar aplicações. Todos os programas, bibliotecas e *frameworks* utilizados são abertos ao público, desenvolvidos por uma comunidade e mantidos pela empresa Open Robotics.

A simulação dos sensores do robô MCCR e seu uso com os algoritmos de fusão necessitaram do desenvolvimento de alguns modelos: um modelo do sensor INS, a partir das informações de erro da folha de dados e um sistema para conversão de dados de sonares para mensagem do tipo *PointCloud2*, a fim de permitir seu uso como mensagem do tipo *laserscan* junto ao algoritmo AMCL. O interesse no AMCL é o potencial de usar informações do ambiente para corrigir erros que crescem com a distância percorrida, que são característicos do INS e da odometria. Além disso, foi desenvolvido no Gazebo um tanque para representar o fundo do navio FPSO e permitir testes em ambiente submarino.

O funcionamento do modelo do INS foi verificado e as curvas de erro simuladas eram coerentes com o erro apresentado na folha de dados. A conversão dos dados de sonar para *PointCloud2* foi verificada em ambiente simulado e foi possível separar os objetos por tamanho e posição de forma qualitativa. Foi realizada também a comparação entre os dados obtidos com imagens de um objeto num tanque e imagens do sonar, com resultados satisfatórios.

Experimentos em simulação foram realizados em duas séries: H para testes com o robô Husky e M com o MCCR. A série H utilizou uma sala com diversos objetos para entender o funcionamento de cada um dos algoritmos a ser comparados. Em H1 e H2, foi

possível observar que os resultados obtidos da odometria de rodas geram erros que crescem de forma mais elevada, comparados aos obtidos pela fusão dos seus dados com o IMU, a partir do EKF. Também em H2, a importância da configuração correta dos parâmetros para o filtro de Kalman foi analisada para casos nos quais ocorreram um erro brusco em um dos sensores durante uma colisão com obstáculo.

Após esses experimentos, foram realizados os experimentos H3, que permitiram avaliar o crescimento do erro devido à distância percorrida e as características do ambiente, comparando os filtros de Kalman com o filtro de partícula AMCL. Foram criados vários ambientes fechados com diversos obstáculos espalhados, diferentes mapas com percursos retilíneos e rodeando objetos. Os filtros de Kalman apresentaram melhor resultado nesses experimentos e o AMCL foi sensível à quantidade de objetos no ambiente.

O experimento M1 mostrou que a variabilidade do escorregamento não permitiu verificar a influência da covariância da mensagem da odometria de forma sistemática. Apesar disso, essas simulações demonstraram que o modelo de INS pôde ser fusionado com os dados de odometria utilizando ambos os filtros de Kalman para a redução dos erros de estimativa de posição. Nesse experimento, os erros da fusão com UKF foram maiores que para o EKF na maioria dos casos.

Para o experimento M2, foram utilizados os dados do sonar mecânico em formato de mensagem em PointCloud2, posteriormente convertidos para o tipo de mensagem *laser scan*. Apesar dos erros obtidos na estimativa de posição pelo AMCL não terem sido satisfatórios, os experimentos demonstraram que os dados convertidos poderiam ser integrados ao algoritmo AMCL. Um estudo mais aprofundado é necessário para determinar se a baixa eficiência é devido à falta de otimização de outros parâmetros ou uma limitação do ambiente menos estruturado, com pouca informação ou repetição de barreiras do fundo do FSPO. Se esse último for o caso, os sonares ainda serão úteis na localização de obstáculos, mas não na localização do robô.

Este trabalho cumpriu com o objetivo principal de desenvolver modelos necessários para a simulação dos sensores do sistema de localização do MCCR e verificar seu funcionamento, permitindo estudo dos algoritmos de fusão. Além disso, esses modelos podem ser utilizados para simulações com novas aplicações. Como exemplo, a conversão

de dados para *PointCloud2* pode ser utilizada para melhorar a navegação e desvio de obstáculo de veículos robóticos.

Os experimentos simulados realçaram que quando o escorregamento das rodas durante as curvas é significativo, seu impacto é bastante variável. Isso indica a necessidade de caracterização da odometria do MCCR e a necessidade de validar seu modelo antes da otimização dos parâmetros dos algoritmos de fusão. Essas atividades foram paralisadas durante a pandemia, mas agora podem ser retomadas.

## REFERÊNCIAS

BARSHAN, B.; DURRANT-WHYTE, H. F. Inertial Navigation Systems for Mobile Robots. **IEEE Transactions on Robotics and Automation**, v. 11, n. 3, p. 328–342, 1995.

BOVBEL, P. **pointcloud\_to\_laserscan** - ROS Wiki, 2015. Disponível em: [http://wiki.ros.org/pointcloud\\_to\\_laserscan](http://wiki.ros.org/pointcloud_to_laserscan). Acesso em: 16 nov. 2021.

CASTILLÓN, M. *et al.* State of the Art of Underwater Active Optical 3D Scanners. **Sensors**, v. 19, n. 23, p. 5161, 25 nov. 2019.

CERQUEIRA, R. *et al.* A novel GPU-based sonar simulator for real-time applications. **Computers and Graphics (Pergamon)**, v. 68, p. 66–76, 2017.

CHADHA, H. S. **The Unscented Kalman Filter: Anything EKF can do I can do it better!**, 2018. Disponível em: <https://towardsdatascience.com/the-unscented-kalman-filter-anything-ekf-can-do-i-can-do-it-better-ce7c773cf88d>. Acesso em: 25 ago. 2020.

CHENG, L.; WANG, Y. Localization of the autonomous mobile robot based on sensor fusion. **IEEE International Symposium on Intelligent Control - Proceedings**, p. 822–826, 2003.

CHO, H. *et al.* Experimental results of rapid underwater object search based on forward-looking imaging sonar. **2015 IEEE Underwater Technology, UT 2015**, 2015.

CHONG, Z. J. *et al.* Synthetic 2D LIDAR for Precise Vehicle Localization in 3D Urban Environment. p. 1554–1559, 2013.

CLEARPATH, R. **Simulating HUSKY UGV**, 2015. Disponível em: <https://www.clearpathrobotics.com/assets/guides/melodic/husky/SimulatingHusky.html>. Acesso em: 3 jul. 2019

CLEARPATH, R. **HUSKY - UNMANNED GROUND VEHICLE**, 2018. Disponível em: <https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/>. Acesso em: 3 abr. 2019.

COOK, G. **Mobile Robots: Navigation, Control and Remote Sensing**. [s.l.] A JOHN WILEY & SONS, INC., 2011.

DFKI. **Maritime Exploration Hall - German Research Center for Artificial Intelligence (DFKI GmbH)**, 2018. Disponível em: <https://robotik.dfki-bremen.de/en/research/research-facilities/maritime-exploration-hall.html>. Acesso em: 15 ago. 2020.

DONOVAN, G. T. Position error correction for an autonomous underwater vehicle inertial navigation system (INS) using a particle filter. **IEEE Journal of Oceanic Engineering**, v. 37, n. 3, p. 431–445, 2012.



- ELASTOBOR. **Corrente plástica Elastobor com Click**. Disponível em: <<https://www.elastobor.com.br/corrente-plastica-elastobor-com-click-9mm-cinza/p>>. Acesso em: 5 nov. 2021.
- EVAC CLEANTECH SOLUTIONS ANYWHERE. **Soluções para FPSOs, FSOs e embarcações de apoio**, 2015. Disponível em: <https://evac.com/pt/offshore/supportvessels/>. Acesso em: 10 jan. 2022.
- GAZEBO. **Gazebo - Robot simulation made easy**, 2019. Disponível em: <http://gazebosim.org/>. Acesso em: 3 abr. 2020.
- GAZEBOSIM.ORG. **Gazebo Tutorial: ROS integration overview**, 2014. Disponível em: [http://gazebosim.org/tutorials?tut=ros\\_overview](http://gazebosim.org/tutorials?tut=ros_overview). Acesso em: 17 jun. 2020.
- HARRIS, C. R. *et al.* Array programming with NumPy. **Nature**, v. 585, n. 7825, p. 357–362, 2020.
- HASHIKAWA, F.; MORIOKA, K. Mobile robot navigation based on interactive SLAM with an intelligent space. **URAI 2011 - 2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence**, p. 788–789, 2011.
- IXBLUE. **Rovins Nano iXblue**, 2020. Disponível em: <https://www.ixblue.com/products/rovins-nano>. Acesso em: 16 abr. 2020.
- JHA, A.; KUMAR, M. Two wheels differential type odometry for mobile robots. **Proceedings - 2014 3rd International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2014**, 2015.
- JONES, E. *et al.* **SciPy: Open source scientific tools for Python**, 2001. Disponível em: <http://www.scipy.org/>. Acesso em: 31 ago. 2020.
- KAESTNER, R. **ROS Wiki - RQT Multiplot**, 2016. Disponível em: [http://wiki.ros.org/rqt\\_multiplot](http://wiki.ros.org/rqt_multiplot). Acesso em: 31 ago. 2020.
- KLANČAR, G. *et al.* **WHEELED MOBILE ROBOTICS**. [s.l.] Butterworth-Heinemann, 2017.
- LABBE, R. **GitHub - rlabbe/Kalman-and-Bayesian-Filters-in-Python**, 2019. Disponível em: <https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python>. Acesso em: 28 ago. 2020.
- LEE, D. *et al.* Sensor Fusion Localization System for Outdoor Mobile Robot. **2009 ICCAS-SICE**, p. 1384–1387, 2009.
- LI, Q. *et al.* Kalman filter and its application. **Proceedings - 8th International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2015**, n. 10, p. 74–77, 2016.
- MENNA, B. V. *et al.* GPS aided strapdown inertial navigation system for autonomous robotics applications. **2017 17th Workshop on Information Processing and Control, RPIC 2017**, v. 2017- Janua, p. 1–6, 2017.

MOORE, T. **robot\_localization - ROS Wiki**, 2018. Disponível em: [http://wiki.ros.org/robot\\_localization](http://wiki.ros.org/robot_localization). Acesso em: 8 nov. 2021.

MOREU, M. *et al.* Ultra-Deepwater steel riser systems hosted on fpsos offshore Brazil. **Proceedings of the Annual Offshore Technology Conference**, v. 4, n. 10, p. 3178–3195, 2015.

OPEN SOURCE ROBOTIC FOUNDATION. **ROS Wiki - RQT**, 2016. Disponível em: <http://wiki.ros.org/rqt>. Acesso em: 25 jun. 2020.

OPEN SOURCE ROBOTIC FOUNDATION. **ROS Wiki - ROS Tutorials Understanding Topics**, 2017. Disponível em: [http://wiki.ros.org/pt\\_BR/ROS/Tutorials/UnderstandingTopics](http://wiki.ros.org/pt_BR/ROS/Tutorials/UnderstandingTopics). Acesso em: 25 jun. 2020.

OPEN SOURCE ROBOTICS, F. **urdf/Tutorials/Building a Visual Robot Model with URDF from Scratch - ROS Wiki**, 2011. Disponível em: <http://wiki.ros.org/urdf/Tutorials/Building a Visual Robot Model with URDF from Scratch>. Acesso em: 24 jun. 2020.

OPEN SOURCE ROBOTICS, F. **ROS Wiki - Nodes**, 2018. Disponível em: <http://wiki.ros.org/Nodes>. Acesso em: 16 jun. 2020.

OPEN SOURCE ROBOTICS, F. **About ROS and Open Source Robotics**, 2019. Disponível em: <http://www.ros.org/about-ros/>. Acesso em: 3 abr. 2020.

PAULL, L. *et al.* AUV navigation and localization: A review. **IEEE Journal of Oceanic Engineering**, v. 39, n. 1, p. 131–149, 2014.

PEI, Y.; KLEEMAN, L. A novel odometry model for wheeled mobile robots incorporating linear acceleration. **2017 IEEE International Conference on Mechatronics and Automation, ICMA 2017**, p. 1396–1403, 2017.

ROS ANSWER - FELIXWATZLAWIK. **ROS Answer - AMCL Localization, Drifts when turning**, 2015. Disponível em: <https://answers.ros.org/question/208466/amcl-localization-drifts-when-turning/>. Acesso em: 30 jul. 2020.

ROSE, C. *et al.* An integrated vehicle navigation system utilizing lane-detection and lateral position estimation systems in difficult environments for GPS. **IEEE Transactions on Intelligent Transportation Systems**, v. 15, n. 6, p. 2615–2629, 2014.

RUSU, R. B.; COUSINS, S. **3D is here: Point Cloud Library (PCL)**. IEEE International Conference on Robotics and Automation (ICRA). **Anais [...]** Shanghai, China: IEEE, 2011

S. J. JULIER; J. K. UHLMANN. Unscented Filtering and Nonlinear Estimation. **Proceedings of the IEEE**, v. 92, n. 3, p. 401–422, 2004.

SENAI CIMATEC. **SENAI CIMATEC desenvolve solução para inspeção de tanques de carga em serviço de navios-plataforma**, 2019. Disponível em: <http://www.senaicimatec.com.br/noticias/senai-cimatec-desenvolve-solucao-para-inspecao-de-tanques-de-fpsos/>. Acesso em: 25 ago. 2020.

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL E INNOSPECTION; LTD. **Underwater inspection robot and embedded software**, 2019. Disponível em: <https://patents.google.com/patent/BR102019016938A2/en>. Acesso em: 25 ago. 2020.

SIEGWART, R.; NOURBAKHSI, I. Prediction of Position. **Introduction to autonomous mobile robots. MIT press.**, p. 159–230, 2011.

STENTZ, A.; HEBERT, M. A complete navigation system for goal acquisition in unknown environments. **Autonomous Robots**, v. 2, n. 2, p. 127–145, 1995.

TELEDYNE MARINE. **Teledyne Marine - NBLUEVIEW M900-2250-S 130/45-MK2o Title**, 2021. Disponível em: [http://www.teledynemarine.com/M900-2250-S\\_130-45-Mk2](http://www.teledynemarine.com/M900-2250-S_130-45-Mk2). Acesso em: 13 set. 2021.

THRUN, S. **Probabilistic Robotics**. p. 1999–2000, 2000.

TRITECH. **Tritech - Micron - Mechanical Scanning Sonar (Small ROV)**, 2020. Disponível em: <https://www.tritech.co.uk/product/small-rov-mechanical-sector-scanning-sonar-tritech-micron>. Acesso em: 13 set. 2021.

UYULAN, C.; ERGUZEL, T.; ARSLAN, E. Mobile robot localization via sensor fusion algorithms. **2017 Intelligent Systems Conference, IntelliSys 2017**, v. 2018-Janua, n. September, p. 955–960, 2018.

ZEKAVAT, S. A.; BUEHRER, M. **Handbook of position location : theory, practice and advances / by Reza Zekavat, Michael Buehrer**. Oxford: Wiley-Blackwell, 2011.

ZHANG, H. *et al.* Study on underwater synthetic navigation system based on inertial navigation system and multibeam bathymetric system. **ICEMI 2009 - Proceedings of 9th International Conference on Electronic Measurement and Instruments**, p. 3237–3241, 2009.

ZHANG, L.; ZAPATA, R.; LÉPINAY, P. Self-adaptive Monte Carlo localization for mobile robots using range finders. **Robotica**, v. 30, n. 2, p. 229–244, 2012.

ZHANG, T. *et al.* Sensor fusion for localization, mapping and navigation in an indoor environment. **2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2014 - 7th HNICEM 2014 Joint with 6th International Symposium on Computational Intelligence and Intelligent In**, n. November, p. 1–6, 2014.