UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Filipe Querubim Bordon
Douglas Dellacruci Goulart

Sistema de mapeamento robótico para corpos móveis com redundância sensorial

São Carlos

2020

Filipe Querubim Bordon
Douglas Dellacruci Goulart

Sistema de mapeamento robótico para corpos móveis com redundância sensorial

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Marco Henrique Terra

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Dellacruci Goulart, Douglas
D646s Sistema de mapeamento robótico para corpos móveis com redundância sensorial / Douglas Dellacruci Goulart; orientador Marco Henrique Terra. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Mapeamento Robótico. 2. ROS. 3. Kinect. 4. Lidar. 5. SLAM. 6. Movimentação Autônoma. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Querubim Bordon, Filipe
Q483s Sistema de mapeamento robótico para corpos móveis com redundância sensorial / Filipe Querubim Bordon; orientador Marco Henrique Terra. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Mapeamento Robótico. 2. ROS. 3. Kinect. 4. Lidar. 5. SLAM. 6. Movimentação autônoma. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Douglas Dellacruci Goulart

Título: "Sistema de mapeamento robótico para corpos móveis com redundância sensorial"

Trabalho de Conclusão de Curso defendido e aprovado em 15/06/2020,

com NOTA 9,0 (nove , zero), pela Comissão Julgadora:

Prof. Titular Marco Henrique Terra - Orientador - SEL/EESC/USP

Mestre Kenny Anderson Queiroz Caldas - Doutorando - SEL/EESC/USP

Prof. Dr. Roberto Santos Inoue - UFSCar

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

FOLHA DE APROVAÇÃO

Nome: Filipe Querubim Bordon

Título: "Sistema de mapeamento robótico para corpos móveis com redundância sensorial"

Trabalho de Conclusão de Curso defendido e aprovado em 15/06/2020,

com NOTA__9,0____(nove , zero), pela Comissão Julgadora:

Prof. Titular Marco Henrique Terra - Orientador - SEL/EESC/USP

Mestre Kenny Anderson Queiroz Caldas - Doutorando - SEL/EESC/USP

Prof. Dr. Roberto Santos Inoue - UFSCar

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

AGRADECIMENTOS

Gostaríamos de agradecer aos nossos respectivos familiares pelo suporte durante o desenvolvimento do trabalho, aos nossos amigos de faculdade: Aurélio, João Artur, Pedro Marchi, Nicholas, Gabriel Baptista, que estiveram presente nos altos e baixos dessa jornada chamada graduação, e principalmente, ao nosso orientador Marco Henrique Terra que iniciou a jornada conosco nos projetos que enfim culminariam na conclusão deste trabalho.

"O maior bem do Homem é uma mente inquieta."
Isaac Asimov

RESUMO

BORDON, F. e GOULART, D. **Sistema de mapeamento robótico para corpos móveis com redundância sensorial**. 2020. 82p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Mapeamento e localização simultâneas é uma técnica aplicada em robótica na qual se utiliza de sensores para se obter conhecimento do ambiente e da trajetória realizada por um robô móvel autônomo. Esta técnica também chamada de *SLAM* (sigla provinda de seu nome em inglês) e é comumente utilizada pelos entusiastas de robótica para diversas aplicações. Uma das técnicas de *SLAM* utilizada neste trabalho é a desenvolvida pela equipe Hector da Universidade Técnica de *Darmstadt*, que provê uma solução aplicada a apenas um sensor de medição de distância *laser* de linha garantindo uma solução rápida, eficiente e com pouca utilização de processamento de máquina (sendo premiada diversas vezes em torneios de robótica). Também utiliza-se aplicações que não necessitam apenas de sensores *laser* de linha para a realização do *SLAM*, uma dessas abordagens é a técnica utilizando o *RTAB-Map* que adiciona sensores como o *KINECT* e odômetros ao processo. Este trabalho visa a abordagem dessas aplicações, onde se utiliza mais de um sensor para realizar tais técnicas, utilizando-se da redundância de sensores e assim fornecendo uma visão de quais abordagens adotadas possam possuir um efeito vantajoso ou desvantajoso para possíveis aplicações futuras que tenha como objetivo a utilização de diversos sensores para *SLAM* aplicado à robótica. Este trabalho apresenta a utilização do mapeamento com a técnica desenvolvida pelo time *Hector* extendida para mais de um sensor *laser* e também utilizando o *RTAB-Map* com dois sensores *laser* e *KINECT*.

Palavras-chave: Mapeamento Robótico. ROS. Kinect. Lidar. SLAM. Movimentação autônoma.

ABSTRACT

BORDON, F. e GOULART, D. **Robotic mapping system for mobile bodies with sensory redundancy**. 2020. 82p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Simultaneos localization and mapping is a technique applied in robotics which it uses sensors to gain knowledge of the environment and the trajectory performed by an autonomous mobile robot. This technique is also called by its acronym SLAM and is commonly used by robotics enthusiasts for various applications. One of the SLAM techniques used in this graduation project is developed by the team Hector of the Technical University Darmstadt wich demonstrates a solution applied to only one laser line sensor ensuring a fast and efficient solution with low data processing cycles (being awarded some times in robotics tournaments). Applications that do not only require laser line sensors to perform the SLAM are also used, one of these approaches is the technique RTAB-Map which adds sensors like kinect and odometers to the process. This graduation project aims to approach these applications, where more than one sensor is used to perform such techniques, using the redundancy of sensors and thus providing an insight into which approaches adopted may have a win-win or disadvantageous effect for possible future applications that aims to use various sensors for SLAM applied to robotics. This job presents the usage of mapping tecnic developed by the Hector team extended for more than one laser sensor and also using RTAB-Map with two laser sensor and a KINECT.

Keywords: Robotic Mapping, ROS, Kinect, Lidar, SLAM, Autonomous Movement.

LISTA DE FIGURAS

Figura 1 – Rotações *Roll*, *Pitch* e *Yaw*. 30
Figura 2 – Diferentes sistemas de coordenadas representadas em robótica. 31
Figura 3 – Pseudo código do algoritimo de *loop* fechado utilizado no RTAB-Map . 40
Figura 4 – Representação gráfica de locais, onde setas verticais são *links* de *loop* fechado e setas horizontais são *links* vizinhos. 41
Figura 5 – Modelo 3D do robô desenvolvido 44
Figura 6 – *Sensor Scanning Laser RangeFinder Smart - URG mini - UST-10LX (UUST003)* da *Hokuyo Automatic CO. LTD.* 46
Figura 7 – Sensor Microsoft KINECT v2 47
Figura 8 – Desenho feito no campo de modelagem de ambiente dentro do *gazebo*. . 48
Figura 9 – Visão tridimensional do ambiente desenhado pela ferramenta do *gazebo*. 49
Figura 10 – Árvore de transformações homogêneas para a execução correta do pacote *Hector_slam*. 50
Figura 11 – Mapa obtido através do *hector SLAM* com um sensor laser de linha apenas realizado no anfiteatro Armando Toshio Natsume EESC - USP 51
Figura 12 – Confontro do mapa da figura 11 com a plata baixa do anfiteatro. . . . 52
Figura 13 – Mapeamento do ambiente simples com um sensor *laser*. 53
Figura 14 – Mensagens de dois sensores *laser* de linha publicadas no mesmo tópico com o mesmo *frame* de referência. 55
Figura 15 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô parado. 56
Figura 16 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô parado aproximado. 57
Figura 17 – Seção das transformações homogêneas sendo executadas e correlacionadas com as mensagens lidas pelo *RViz*. 57
Figura 18 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô tendo se movimentado. 58
Figura 19 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô parado após movimentação. 58
Figura 20 – Representação gráfica dos dois sensores em conjuntos, cada um com sua transformação homogênea. 59

Figura 21 – Erros obtidos durante a execução do algorítimo de SLAM com dois sensores publicando em dois tópicos, mas cada um com *frame* de referência diferente. 60
Figura 22 – Mapeamento utilizando a abordagem de um tópico de publicação, mas cada mensagem possui seu *frame* de referência. 61
Figura 23 – Posição inicial do robô dirante o mapeamento utilizando a abordagem de um tópico de publicação, mas cada mensagem possui seu *frame* de referência. 61
Figura 24 – Modelo do cubo de aresta $2.5m$. 62
Figura 25 – Mapeamento do cubo com os contornos dos sensores. 62
Figura 26 – Sensor laser virtual resultante do *merger*. 63
Figura 27 – Sensor resultante comparado com os dois sensores. 63
Figura 28 – Mapeamento realizado com o sensor virtual. 64
Figura 29 – Execução de apenas um mapeamento para o sensor virtual com *angle increment* padrão. 65
Figura 30 – Execução de apenas um mapeamento para o sensor virtual com *angle increment* de 0.05. 65
Figura 31 – Sensor laser virtual resultante do *merger* com *angle increment* igual à 0.05. 66
Figura 32 – Sensor resultante com *angle increment* igual à 0.05 comparado com os dois sensores. 66
Figura 33 – Arvore de transformações executadas no processo de mapeamento com o sensor virtual. 67
Figura 34 – Mapeamento gerado durate uma movimentação brusca do robô. 69
Figura 35 – Mensagem de erro no terminal executando o *Hector_slam* durante uma movimentação brusca do robô. 70
Figura 36 – Velocidade mínima angular e linear em um movimento combinado entra as duas. 70
Figura 37 – Velocidade mínima em um movimento linear. 70
Figura 38 – Velocidade mínima em um movimento angular. 71
Figura 39 – Setup de configuração para modelo utilizando sensor *laser* de linha, *kinect v2* e a odometria presente no robô 72
Figura 40 – Ciclo inicial de mapeamento utilizando o *RTAB-Map* 72
Figura 41 – Mapeamento realizado após algumas movimentações do robô utilizando o *RTAB-Map* 73
Figura 42 – Ciclo inicial de mapeamento utilizando o *RTAB-Map* com 0.0025 de incremento angular. 74
Figura 43 – Comparação entre os dois primeiros ciclos de mapeamento com parâmetros de *angle increment* diferentes. 74

Figura 44 – Mapeamento realizado do ambiente simples utilizando o *RTAB-Map*. . 75

LISTA DE TABELAS

Tabela 1 – Especificações técnicas do UST-10LX. 46
Tabela 2 – Especificações do Kinect e do Kinect v2 47

LISTA DE ABREVIATURAS E SIGLAS

ROS	*Robot Operating System*
SLAM	*Simultaneous Localization And Mapping*
ToF	*Time of Flight*
RTAB-Map	*Real-Time Appearance-Based Mapping*
Lidar	*Light Detection And Ranging*
tf	*Track and Field*
WM	*Working Memory*
LTM	*Long-Term Memory*

SUMÁRIO

1 INTRODUÇÃO 25

2 TEORIA 27
2.1 ROBÓTICA E MOVIMENTAÇÃO 27
2.2 TRANSFORMAÇÕES HOMOGÊNEAS 28
2.3 SLAM 31
2.4 ROS 33
2.5 SIMULAÇÃO APLICADA À ROBÓTICA 34
2.6 SENSORES 35
2.6.1 LIDAR 35
2.6.2 KINECT 36
2.7 HECTOR SLAM 37
2.8 RTAB-MAP 39

3 MATERIAIS E MÉTODOS 43
3.1 AMBIENTE DE SIMULAÇÃO 43
3.1.1 MODELAGEM ROBÓTICA 43
3.1.2 LASER DE LINHA E MODELAGEM 45
3.1.3 SENSOR KINECT V2 47
3.1.4 MODELAGEM DO AMBIENTE SIMPLES 47
3.2 AMBIENTE DE VISUALIZAÇÃO 49
3.3 MAPEAMENTO UTILIZANDO SENSOR A LASER DE LINHA E PACOTE ROS PARA CORPOS MÓVEIS 49

4 RESULTADOS 53
4.1 MAPEAMENTO UTILIZANDO HECTOR_SLAM 53
4.2 INTEGRAÇÃO COM DOIS SENSORES *LASER* 54
4.2.1 ABORDAGEM DE MESMO *FRAME* E/OU TÓPICO DE MENSAGEM 54
4.2.2 ABORDAGEM DE FUSÃO DOS SENSORES *LASER* 63
4.2.3 CONDIÇÕES DE CONTORNO DO *HECTOR_SLAM* 68
4.3 MAPEAMENTO UTILIZANDO RTAB-MAP 71
4.4 CONTRIBUIÇÃO *OPEN SOURCE* 75

5 CONCLUSÃO 77

REFERÊNCIAS 79

1 INTRODUÇÃO

A necessidade de interação entre o ambiente e as novas tecnologias são de extrema importância para o desenvolvimento orgânico da nova indústria 4.0 e da era mundial envolvida na *internet* das coisas. A necessidade de interação permite que máquinas, cada vez mais precisamente e autonomamente possam tomar decisões e executar suas respectivas funções. Outro ponto importante desta interação é a comunicação das máquinas com os seres humanos, permitindo uma melhor comunicação entre eles. Essa comunicação feita através de trocas de informações, levantamento de bancos de dados e interpretações, permite que a sociedade possa ir fundo nessa nova jornada denominada de era digital.

Sensores são a ponte entre o ambiente humano e o ambiente digital, permitindo que os então chamados robôs possam "enxergar" à sua volta. Em termos de visão, sensores óticos possuem uma tecnologia tão estudada e difundida no meio de entusiastas à robótica, claro que, executando um paralelo entre um dos principais, se não, o principal sentido humano, a visão.

A aplicação dos sensores ópticos vem tornando-se cada vez mais comum na área de mobilidade e autonomia robótica devido ao crescente aumento na capacidade de processamento de imagens dos computadores. Outro fator que contribui para esse crescimento é o fato dos sensores ópticos estarem cada vez mais precisos, rápidos, baratos e menores. A combinação desses fatores resulta na aplicação e melhoria dos algoritmos de localização e mapeamento, melhorando constantemente a sua eficiência.

Na área de navegação e mapeamento robótico a confiabilidade e precisão de um dado são de suma importância assim como processar, interpretar e tomar as ações corretamente. Na área de carros autônomos, por exemplo, a confiabilidade e velocidade computacional pode evitar um acidente com respostas de velocidade sobre humana. Com esse intuito esse trabalho foi planejado visando abstrair os melhores recursos presentes em dois tipos de sensores e utilizar um para cobrir possíveis falhas do outro, gerando uma redundância sensorial.

O objetivo do projeto em questão é estudar as técnicas de mapeamento e localização utilizando um sensor *laser* de linha e um sensor 3D de profundidade gerando um dado mais confiável. Esse projeto aborda uma técnica de mapeamento utilizando o sensor de linha, explorando seus prós e contras. Também aborda outra técnica de mapeamento que utiliza além do sensor *laser* como parâmetro de referência. As técnicas envolvem entre trabalhar com sensores operando em conjunto executando a mesma funcionalidade, operando de maneira complementar na mesma função, ou também operando como instrumentos diferentes.

O estudo será através de simulações em ambiente controlados, na qual conclusões sobre os dois métodos adotados para a realização do mapeamento e localização simultâneos serão apresentadas. Em trabalhos anteriores realizados pelos co-autores deste trabalho já garante o funcionamento do mesmo através de experimentação, proporcionando assim, possíveis extensões do mesmo para aplicações diferentes do que se considera uma utilização comum.

2 TEORIA

2.1 ROBÓTICA E MOVIMENTAÇÃO

Um dos principais problemas encontrados nos dias atuais quando os assuntos envolvidos são inteligência artificial, robótica, autonomia de corpos móveis de modo geral é a capacidade do sistema inteligente conseguir criar uma representação espacial do ambiente em que este se movimenta, (LAKEMEYER; NEBEL, 2003).

Esta representação serve para que o corpo móvel consiga desenvolver um movimento autônomo mais próximo possível da movimentação humana, ou seja, sem colidir com nenhum obstáculo no caminho e também criar uma memória que reflete o caminho realizado pelo corpo móvel ao se movimentar neste espaço. Antigamente os mapeamentos eram feitos com base nas métricas das propriedades geométricas do ambiente de modo geral, procedimentos conhecidos como geo-mapeamento, (LAKEMEYER; NEBEL, 2003).

A tarefa de obtenção de mapas de ambientes internos é uma tarefa dura. Entretanto, a sua realização é bastante importante, pois um mapa bem elaborado traz precisão e robustez para um sistema de navegação, (Sun et al., 2018). O processo de mapeamento gera um mapa do ambiente, enquanto o processo de localização calcula a posição do corpo móvel em relação ao mapa, de acordo com os dados provindos pelos sensores utilizados, (Sun et al., 2018).

O problema de localização, em alguns casos, pode ser resolvido com sistemas de posicionamento externos ao corpo móvel, por exemplo sistema de triangulação com bases fixas e sistema de geolocalização global (*Global Positioning System* - GPS). Entretanto o primeiro implica em mudanças no ambiente e o segundo não é eficiente em ambientes internos, no subsolo ou em baixo da água. Nesses casos é necessário a utilização do conjunto de sensores como *laser*, *kinect* e odômetro presente nos corpos móveis, (Sun et al., 2018) e (GARCIA-FIDALGO; ORTIZ, 2015).

A união dos processos de mapeamento e localização gera um sistema de navegação suficientemente autônomo, capaz de realizar uma série de tarefas evitando colisões e obstáculos. Essa autonomia é importante para veículos autônomos que precisam ser capazes de operar sem intervenção humana.

Este conceito de mapeamento é muito aplicado em robótica, para poder estimar o movimento de corpos móveis de uma maneira mais exata e efetiva para sistemas inteligentes. Existe uma certa diferença quando se trata de mapeamento voltado para geografia e robótica. Um tem o objetivo de gerar mapas baseados em coordenadas globais, o mapa criado não se apega a todas as medidas feitas pelo sensor, e sim pelo espaço que se está sendo mapeado.

Em robótica, é necessário conhecer todas as medidas do sensor de forma a se estimar uma posição inicial e uma posição final de um corpo móvel utilizando-se de todas as medidas espaciais geradas pelo sensor, (LAKEMEYER; NEBEL, 2003).

É necessário também compreender o problema que permeia as técnicas de mapeamento em geral, é necessário compreender que o corpo móvel inicialmente não possui nenhuma informação do ambiente em que ele se encontra sem ser os dados da primeira leitura do sensor de distância. Quando este ainda não iniciou o movimento, bem como também se é possível que este corpo móvel consiga criar uma representação do ambiente em que ele se encontra apenas se movimentando por ele e coletando informações de distâncias entre o corpo móvel e os obstáculos, (LAKEMEYER; NEBEL, 2003) e (BAILEY; DURRANT-WHYTE, 2006).

Sabendo-se que o objetivo do mapeamento então é conhecer o ambiente em que o corpo móvel está se movimentando, e dessa forma criar uma representação espacial do mesmo, é necessário compreender que a ciência que discute essa estimativa de movimentação e de posição que o corpo móvel ocupa após realizar um movimento, baseando-se apenas nas informações espaciais anteriores, é uma ciência probabilística. Consiste em prever qual movimento foi feito baseado nos dados atuais em comparação com os dados antigos armazenados. E assim conseguir de forma simultânea construir um mapa do ambiente em que este se movimenta e saber sua localização de maneira exata, (BAILEY; DURRANT-WHYTE, 2006).

Apesar dos processos de mapeamento e localização poderem ser realizados como tarefas distintas, eles estão fortemente relacionados. Para construir o mapa de um ambiente é necessário saber a posição das estruturas e obstáculos presentes no local. Por outro lado, durante o processo de localização, a posição do corpo móvel é obtida em relação ao mapa gerado. Nesse caso, o mapa do local deve ser obtido antes de iniciar a navegação do corpo móvel, o que pode limitar a autonomia do veículo, gerando uma redundância, onde o mapeamento depende da localização e vice-versa. Afim de resolver essa redundância várias abordagens, que realizam as duas tarefas simultaneamente, foram propostas. Essas abordagens criam um mapa de um ambiente desconhecido e posicionam o corpo móvel dentro desse mapa. Essas técnicas são genericamente conhecidas como Localização e Mapeamento Simultâneo (*Simultaneous Localization and Mapping* - SLAM), (GARCIA-FIDALGO; ORTIZ, 2015).

2.2 TRANSFORMAÇÕES HOMOGÊNEAS

Em um sistema espacial a representação de objetos e suas posições são determinados por coordenadas espaciais, no espaço cartesiano se utiliza de três principais coordenadas, estas são X, Y e Z, tais coordenadas são imaginárias e só podem ser utilizadas caso um sistema de coordenadas espacial esteja definido, e também a unidade espacial esteja

delimitada, (SPONG; HUTCHINSON; VIDYASAGAR, 2020). Em alguns países que adotam o Sistema Internacional de Unidades, e a unidade para medidas espaciais é o metro (também conhecido como unidade de medida de distância), (GERMAN; WÖGER, 2005).

Para a representação de uma posição em um ambiente espacial, é necessário que um sistema de coordenadas seja delimitado, e para delimitá-lo é necessário que uma origem seja escolhida, e essa origem é a coordenada $0, 0, 0$ e a partir dela é possível representar os objetos por coordenadas espaciais, pois estarão em referência a esta origem, (CRAIG, 2005).

Dessa maneira é possível então correlacionar sistemas de coordenadas espaciais, criando uma relação de coordenadas entre suas origens, sendo assim um objeto descrito em um sistema de coordenadas pode ser descrito em outro sistema, sendo que suas origens possam ser descritas cada uma no sistema de coordenadas alheio, (SPONG; HUTCHINSON; VIDYASAGAR, 2020). A operação que descreve a orientação de um sistema de coordenadas em relação a outro é chamada de transformação homogênea. Esta transformação pode conter tanto rotações dentro de um sistema, quanto deslocamentos lineares, (SPONG; HUTCHINSON; VIDYASAGAR, 2020). Uma transformação pode ser representado pela Equação (2.1), onde para se obter a nova coordenada de um ponto em um sistema de coordenadas $\{1\}$ para um sistema de coordenadas $\{0\}$ é dada pela coordenada do ponto pré multiplicada pela matriz de transformação homogênea.

$$
{}^{0}_{1}T = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1}
$$

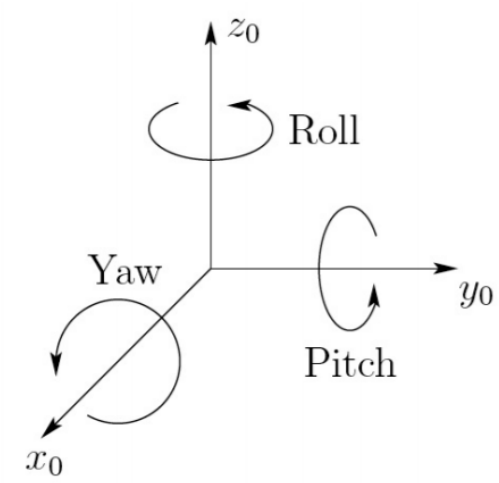
sendo que: o vetor $n_x, n_y, n_z, 0$ representa a direção de x_1; o vetor $s_x, s_y, s_z, 0$ representa a direção de y_1; o vetor $a_x, a_y, a_z, 0$ representa a direção de z_1; o vetor $d_x, d_y, d_z, 1$ representa a origem de $\{1\}$ em relação a $\{0\}$, (SPONG; HUTCHINSON; VIDYASAGAR, 2020).

Algumas rotações especiais são comumente utilizadas para desenvolvimentos com aplicações em robótica, que são as rotações conhecidas como $roll(\phi)$, $pitch(\theta)$ e $yaw(\psi)$, estas rotações são características cada uma em um eixo fixo, dadas respectivamente pelos eixos $X - Y - Z$, (SPONG; HUTCHINSON; VIDYASAGAR, 2020). A representação gráfica dessas rotações pode ser vista na Figura 1.

As matrizes de rotação $roll(\phi)$, $pitch(\theta)$ e $yaw(\psi)$ estão descritas nas Equações (2.2), (2.3) e (2.4).

$$
R_Z(\phi) = \begin{bmatrix} cos(\phi) & -sen(\phi) & 0 \\ sen(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}
$$

Figura 1 – Rotações *Roll*, *Pitch* e *Yaw*.



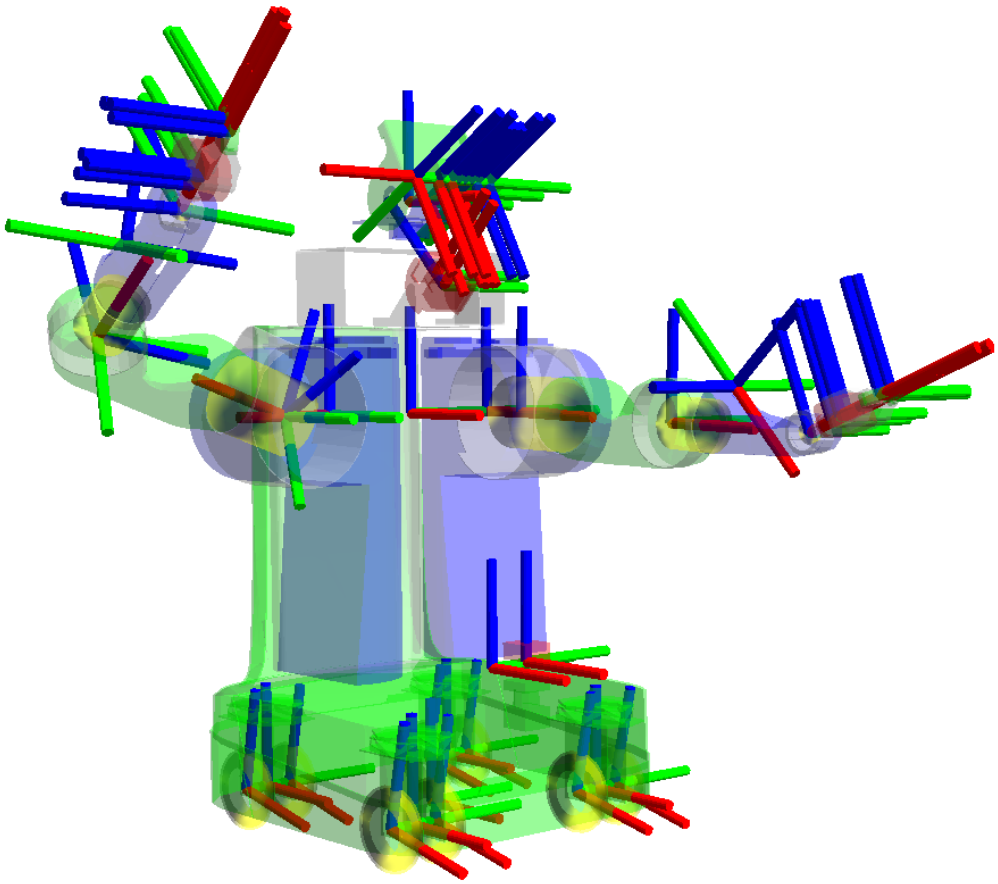
Fonte: Spong, Hutchinson e Vidyasagar (2020)

$$R_Y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sen(\theta) \\ 0 & 1 & 0 \\ -sen(\theta) & 0 & cos(\theta) \end{bmatrix}, \tag{2.3}$$

$$R_X(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\psi) & -sen(\psi) \\ 0 & sen(\psi) & cos(\psi) \end{bmatrix}. \tag{2.4}$$

Em robótica, essas transformações são responsáveis por manter unidas as partes de um robô quando representado digitalmente. Também é responsável por correlacionar os dados obtidos através de sensores em relação ao espaço em que o robô se encontra, adotando como origem o corpo do próprio robô, também comumente denominado de chassis., (CRAIG, 2005). Um exemplo dessa referência gráfica, pode ser vista na Figura 2.

Figura 2 – Diferentes sistemas de coordenadas representadas em robótica.



Fonte: 3D transformations (2017)

2.3 SLAM

O *SLAM* é um método de solução para problemas de localização e mapeamento que teve a sua origem em 1986 na *IEEE Robotics and Automation Conference*, realizada em San Fracisco, Califórnia. Nesse tempo os métodos probabilísticos estavam começando a ser aplicados em robótica e inteligencias artificiais. O *SLAM* é um método em que um corpo móvel seja capaz de gerar o mapa de um ambiente e ao mesmo tempo usar esse mapa para obter sua atual localização. Tanto a posição dos pontos de referência quanto a trajetória do corpo móvel são estimados online, sem a necessidade de um conhecimento prévio da sua localização, (GARCIA-FIDALGO; ORTIZ, 2015).

Um exemplo de *SLAM* é compreender que um ser humano é capaz de entrar em um ambiente desconhecido, criar uma memória de como este ambiente é, de onde ele veio e onde ele está neste ambiente. Considerando o *SLAM* ideal, em seu funcionamento mais completo é de que o corpo móvel, consiga se localizar dentro de um ambiente ao mesmo tempo em que cria uma memória de como este ambiente é (mapeamento). Se este é alcançado, pode-se afirmar que um robô ou corpo móvel é totalmente autônomo,

(BAILEY; DURRANT-WHYTE, 2006).

Embora técnicas de *SLAM* estejam bastante avançadas hoje em dia, sua aplicação ainda tem sido um desafio para os estudiosos da área, que tentam a cada momento torná-las ainda mais eficientes. Suas aplicações têm sido feitas nos mais diversos sistemas autônomos: robôs que auxiliam deficientes visuais, carros autônomos, cadeiras de rodas autônomas, aviões autônomos ou remotamente pilotados entre outros que são facilmente encontrados na mídia.

SLAM é implementado em geral através de algoritimos complexos. Possui processamento intenso quando o objetivo é apresentar um resultado preciso. Trata-se de uma classe de algoritimos que analisa o ambiente a fim de criar marcos estacionários que são características peculiares do ambiente. Partindo deste princípio, o objetivo é conseguir localizar o corpo móvel em relação a estes marcos com base nas estimativas de movimentação, (BAILEY; DURRANT-WHYTE, 2006).

O principal problema probabilístico do *SLAM* consiste em um sistema recursivo em que ocorre um processo contínuo de predição e correção utilizando o teorema de Bayes para constantemente diminuir o erro da medida anterior e predizer a próxima medida, (DURRANT-WHYTE; BAILEY, 2006).

As soluções para o *SLAM* probabilístico envolvem encontrar reapresentações apropriadas para o modelo de observação e o modelo de movimento do corpo móvel. Uma das soluções do *SLAM* permite ao corpo móvel entrar em um ambiente desconhecido, criar uma memória sobre o ambiente e traçar sua trajetória dentro desse ambiente. Caso essa solução possa ser implementada de maneira totalmente simultânea pode-se dizer que essa é uma solução ideal, gerando um corpo móvel totalmente autônomo, (DURRANT-WHYTE; BAILEY, 2006).

Devido a complexidade do problema proposto pelo *SLAM* é necessário desenvolver cada vez mais as técnicas de otimização de processamento computacional sem comprometer os resultados gerados, (BAILEY; DURRANT-WHYTE, 2006).

Necessário compreender também que o sistema de mapeamento também deve prever a movimentação do corpo móvel para que este não colida com nenhum obstáculo durante o trajeto que criará o mapa, de forma a não prejudicar o processo de mapeamento em si, bem como apresentar um controle apropriado de velocidade e direção do corpo móvel para evitar colisões, (HORICHI et al., 2016) e (OCANDO et al., 2017).

Dessa forma um sistema de controle do corpo móvel também é necessário, mesmo que este sistema sirva apenas para alerta, ou medidas drásticas em caso de uma movimentação errada do operador do corpo móvel demandar mais processamento, o que pode influenciar no desempenho do algoritmo de *SLAM*.

2.4 ROS

O sistema operacional de robôs, ou do inglês *Robot Operating System* (ROS), é uma coleção de *frameworks* para aplicações robóticas. Esse sistema operacional é um projeto de código aberto que suporta o desenvolvimento e aplicação de diversos componentes. Tal sistema fornece mecanismos de abstração de hardware, *drivers* de dispositivos, bibliotecas, ferramentas de visualização, pacotes gerenciamento de ferramentas, etc. Atualmente diversos fabricantes de dispositivos e sensores desenvolvem bibliotecas para o ROS afim de facilitar a programação de aplicações, (BAILEY; DURRANT-WHYTE, 2006) e (QUIGLEY et al., 2009).

Por ser um ferramenta didática, e possuir uma certa rastreabilidade em todas as ferramentas, funções e bibliotecas disponíveis para o uso geral, o ROS mostra-se uma ferramenta excelente para a aplicação de robótica e desenvolvimento de algoritimos voltados para o mesmo. Somando também ao fato de na literatura existir diversos projetos que utilizam o ROS como ferramenta principal para o desenvolvimentos de projetos aplicados a robótica (vide a referência utilizada: (QUIGLEY et al., 2009), (KOHLBRECHER et al., 2011), (OCANDO et al., 2017), (KöSEOğLU; ÇELIK; PEKTAş, 2017) e (ABOUT ROS, 2017).

O ROS trabalha conectando nós para as atividades computacionais e tópicos para a comunicação de mensagens, (ROS NODE, 2018) e (ROS TOPIC, 2019). O nó é um processo que executa algum tipo de computação, seja ele através de um *plugin* ou de um programa em *C++*, *C*, *Matlab* ou *Python* e se comunicam com outros nós através de tópicos, publicando em, ou se inscrevendo em, (ROS NODE, 2018). O uso de diferentes nós durante uma aplicação provoca uma melhoria no sistema, pois evita um erro no sistema como um todo, visto que este opera em partes separadas, e também reduz a complexidade dos códigos necessários, (ROS NODE, 2018). Na maioria dos casos os nós trabalham com *frames*, que são recursos gráficos, muito utilizados em robótica, (ROS NODE, 2018). É importante ressaltar que os nós são únicos, e para que uma mesma instância de um nó possa ser executado ao mesmo tempo, é necessário remapeá-lo no núcleo do ROS, (ROS NODE, 2018).

Como dito anteriormente os nós utilizam os tópicos para comunicação interna, em forma publicação e inscrição nos mesmos. Dessa forma, mantém-se anônima a comunicação entre nós, por exemplo: um nó publica em um tópico específico e está inscrito (lê mensagens) de outro tópico específico, sem ter conhecimento de quem esta inscrito e publicando nos tópicos respectivamente, (ROS TOPIC, 2019). Os tópicos executam comunicações unidirecionais, isto é, apenas permitem leitura ou escrita de um nó específico, importante ressaltar que é possível que um nó escreva e outro nó leia o mesmo tópico, mas um nó não pode ler e escrever no mesmo tópico. Para isso é necessário a utilização de serviços, (ROS TOPIC, 2019).

O ROS oferece diversas soluções de algorítimo para o *SLAM*, algumas delas são: (SANTOS; PORTUGAL; ROCHA, 2013)

- HectorSLAM: Combina o sistema de *SLAM* 2D baseado em correspondência robusta de varredura e técnicas de navegação 3D utilizando ferramentas de detecção inercial. Os autores focaram na estimação da movimentação em tempo real do robô, utilizando uma alta taxa de atualização e um baixo ruído na medição da distancia e os modernos sensores laser para medidas de distância. Esse sistema não utiliza dados de odometria, permitindo sua implementação em robôs aéreos, como quadricópteros autônomos ou robôs terrestres operando em terrenos irregulares, (SANTOS; PORTUGAL; ROCHA, 2013).

- Gmapping: Um dos pacotes mais utilizados em *SLAM* para robótica ao redor do mundo. É um algorítimo *SLAM* baseado em laser conforme descrito em, (GRISETTI et al., 2007).

- *KartoSLAM*: É uma abordagem *SLAM* baseada em gráfico desenvolvida pela *SRI International's Karto Robotics* que tem sido estendido ao ROS utilizando decomposição de matriz altamente otimizada e não-quantitativa de Cholesky. Um sistema de *SLAM* baseado em gráfico gera um mapa através da média dos gráficos, (SANTOS; PORTUGAL; ROCHA, 2013).

- *CoreSLAM*: É um pacote desenvolvido para o sistema de 200 linhas chamado *tinySLAM*, que foi criado com o objetivo de ser de fácil entendimento e com mínima perda de desempenho, (Steux; Hamzaoui, 2010).

2.5 SIMULAÇÃO APLICADA À ROBÓTICA

Mapeamento é um processo que junta coleta de informação provinda de sensores e transcreve essa informação sobre o ambiente, como obstáculos e posição, (FUENTES-PACHECO; RUIZ-ASCENCIO; RENDÓN-MANCHA, 2015).

Algumas aplicações de *SLAM* deveriam ser cuidadosamente testados em ambientes controlados, a fim de cobrir diversos problemas possíveis, através da modelagem é possível criar modelos de ambientes devidamente calculados, e com os parâmetros já conhecidos dos sensores da atualidade e suas características precisamente medidas e catalogadas, é possível obter modelagens extremamente próximas às reais, (LAVRENOV; ZAKIEV, 2017).

Com a utilização de ambientes controlados para simulação é possível verificar e promover diversos fatores que contribuem na validação do processo de localização e mapeamento, mas não só preso a isso, contribuir em todo o processo de desenvolvimento de aplicações voltadas à robótica móvel, (Femmam, 2017).

A modelagem garante uma representação matemática controlada de um ambiente para uma aplicação específica, assim é possível supor e forçar situações desfavoráveis à uma aplicação e verificar qual seu comportamento diante de tal fato. Também é possível garantir que as condições de contorno de um ambiente são satisfeitas, a fim de conseguir retirar o maior proveito de uma aplicação, antes desta ser colocada em execução real, (Femmam, 2017).

Existem diversos ambientes que interagem entre si, e conversam diretamente um com o outro. Partindo do proposto utilizando o *ROS* para aplicação, a ferramenta de modelagem escolhida foi o *gazebo*, este por possuir integração direta e fácil com o ambiente de desenvolvimento em robótica (*ROS*), (OPEN SOURCE ROBOTICS FOUNDATION, 2014). Também por possuir uma interface amigável, voltada a futuros desenvolvedores de aplicações aplicadas à robótica móvel, (Roberts et al., 2003).

2.6 SENSORES

O projeto consiste da fusão de sensores (dois sensores *laser* e um *KINECT*), a importância de cada sensor está explicitada nas linhas abaixo, a fim de esclarecer quais suas importâncias no momento atual, correlacionado à robótica móvel e à era da indústria digital em questão.

A era atual de indústria 4.0, digitalização e decisões provenientes de análise de dados, compõe uma parcela de extrema importância no desenvolvimento de aplicações voltada para robótica em geral. Sensores inteligentes são um mecanismo para a aproximação e flexibilização da interpretação de dados e assim permitir uma tomada de decisão ou um processamento mais simples e eficiente, (ANDO, 1996). Um pré-processamento pode ser a diferença entre uma aplicação completa de uma aplicação rasa e inconclusiva, (ANDO, 1997).

2.6.1 LIDAR

Primeiramente, foi escolhido o *Scanning Laser Range Finder Smart - URG mini - UST-10LX (UUST003)*. Atualmente com os computadores capazes de executar cada vez mais cálculos por segundo possibilitou que sensores a laser (comumente conhecidos como LIDAR do inglês *Light Detection and Ranging*) que possuem capacidades de medidas de distância extremamente precisas, bem como também uma alta taxa de aquisição de dados, facilitou os algorítmos de *SLAM* se tornarem cada vez mais precisos em suas estimativas, (AGRAWAL et al., 2017).

Os algoritmos de *SLAM* utilizados atualmente possuem uma resposta muito promissora quando utilizados com sensores que possuam uma taxa de captura de dados muito maior que a velocidade do corpo móvel que está sendo utilizado para o mapeamento, tornando assim mais fácil a determinação da localização do robô no ambiente, pois o número

de marcos estacionários é maior para a estimativa da posição do corpo móvel, (HORICHI et al., 2016). Quando a varredura do sensor é muito superior que a velocidade do corpo móvel, durante a varredura, a velocidade de movimentação pode ser então desprezada na medição. Quando o contrário acontece, é necessário que o algoritimo leve em consideração a velocidade do corpo móvel para que não ocorra um mapeamento incoerente, mostrando assim ser necessário uma informação externa da velocidade do corpo móvel quando o mapeamento é realizado, (HORICHI et al., 2016).

Para se garantir que a velocidade de amostragem seja superior a velocidade do corpo móvel, foi escolhido um sensor mais simples, que é um sensor laser de linha, que possui o tempo de medição bem menor do que um sensor comum de nuvem de pontos (*point cloud*) como uma câmera estéreo, por exemplo. O sensor *UST-10LX* escolhido, possui uma velocidade de varredura de $25ms$, e nessa varredura são medidos 1081 pontos num ângulo de 270°, (ROS COMPONENTS, 2016).

2.6.2 KINECT

A primeira versão do sensor, desenvolvido pela Microsoft, era um sensor de profundidade de baixo custo para imagens rápidas e de alta qualidade. Seu lançamento causou um grande impacto na robótica, com diversas aplicações utilizando ele como base do projeto. Entretanto a tecnologia do sensor utilizada na primeira versão era sensível a determinadas condições de incidência solar, tornando o sensor limitado a algumas aplicações internas. A segunda geração do sensor (Kinect v2) chegou ao mercado em 2013 e está disponível aos pesquisadores desde julho de 2014. Essa versão do sensor se baseia na tecnologia de tempo de voo (*Time of Flight*- ToF) o que permite ao sensor uma maior resolução e um maior campo de visão, (Fankhauser et al., 2015).

Enquanto a primeira versão do sensor Kinect utilizava apenas medição de distância por luz infravermelha estruturada, a segunda versão do sensor utiliza o conceito de ToF para a medição de profundidade. Nesse método o sinal de medição é comandado por um pulso de *clock* de alta frequência. Esse sinal dispara uma série de três diodos de laser, que brilham simultaneamente através de difusores, iluminando toda a cena com pulsos curtos de luz infravermelha. Esses pulsos de luz refletem nos objetos da cena e retornam ao sensor de infravermelho. Esse sensor de profundidade é capaz de medir cada *pixel* diferente, armazenando as informações em dois acumuladores. O sinal de *clock* controla qual acumulador está ativo e registrando o pulso de infravermelho que retorna ao sensor. Durante o período em que o pulso de luz está retornando para o sensor, o sinal de *clock* dobra sua frequência registrando a chegada da luz, enviando uma porção de luz ao primeiro acumulador e uma outra porção para o segundo acumulador. Comparando a porção de luz recebida por cada um dos acumuladores é possível inferir uma distância para cada *pixel*. Como distâncias maiores podem ser registradas por frequências de *clocks* menores,

o sensor trabalha emitindo frequências distintas através da modulação da frequência do sinal, (Butkiewicz, 2014).

2.7 HECTOR SLAM

O sistema de *SLAM* escolhido para a aplicação foi o desenvolvido pelo time Hector da Universidade Técnica de Darmstadt, de autoria de Stefan Kohlbrecher and Oskar von Stryk e Johannes Meyer and Uwe Klingauf, (KOHLBRECHER et al., 2011). O sistema é um pacote *open source* desenvolvido pelo time Hector e disponibilizado diretamente pela enciclopédia do ROS, (KOHLBRECHE; MEYER, 2014). Sua utilização é dada nos parâmetros de configuração de seus arquivos *.launch* que determinam os *frames*, tópicos e nós que serão utilizados para o *SLAM*.

De uma maneira generalizada, o sistema de mapeamento e localização utilizado consiste em um processamento de medidas de sensores laser, e com eles o mapa é criado através do algoritimo de *scan matching*. Com o mapa criado, filtros de navegação (filtros de Kalman estendidos) são aplicados para a estimação a posição do corpo móvel e assim calcular sua trajetória, caracterizando sua localização dentro do mapa criado anteriormente, (KOHLBRECHER et al., 2011) e (OLSON, 2009).

A conversão dos dados do sensor laser para a nuvem de pontos é feita transcrevendo os dados de medição em uma matriz onde as distâncias observadas são tratadas como os pontos finais dentro dessa matriz, (KOHLBRECHER et al., 2011). Partindo da nuvem de pontos, um processamento inicial é feito, removendo os pontos que representam uma falsa medida ou uma medida tida como erro, (KOHLBRECHER et al., 2011).

A natureza discreta dos mapas de grade, que são é uma mapa de pontos onde apenas são representados as máximas distâncias obtida pelo sensor laser limita a precisão que pode ser alcançada e também não permite o cálculo direto de valores interpolados ou derivativos. Para tal é utilizado um esquema de interpolação que permite à célula da *sub grid* a precisão necessária. Para tal a filtragem bilinear é empregada tanto para estimar as probabilidades de ocupação quanto os derivativos. Intuitivamente, o valores da célula do *grid map* pode ser vistos como amostras de um distribuição de probabilidade contínua. A interpolação então é feita utilizando os pontos mais próximos do ponto interpolado, ou seja: P_{00}, P_{01}, P_{10} e P_{11}, onde P é o ponto e o número da sua coordenada a partir do ponto interpolado. Nota-se que o ponto interpolado é um ponto contínuo em todo o *map grid*, (KOHLBRECHER et al., 2011). Assim, valem as seguintes Equações:

$$M(P_m) \approx \frac{y - y_0}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{11}) + \frac{x_1 - x}{x_1 - x_0}M(P_{01})\right) +$$

$$\frac{y_1 - y}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{10}) + \frac{x_1 - x}{x_1 - x_0}M(P_{00})\right), \tag{2.5}$$

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y - y_0}{y_1 - y_0}(M(P_{11}) - M(P_{01})) + \frac{y_1 - y}{y_1 - y_0}(M(P_{10}) - M(P_{00})),$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x - x_0}{x_1 - x_0}(M(P_{11}) - M(P_{01})) + \frac{x_1 - x}{x_1 - x_0}(M(P_{10}) - M(P_{00})). \tag{2.6}$$

Sendo M o mapa sendo gerado.

Com o mapa de grade completo agora é realizado o *scan matching*, que utilizada um novo mapa encontrado com o processo de filtragem e interpolação para o alinhamento com um mapa já existente. A abordagem consiste em aplicar o método de *Gauss-Newton* nos feixes de limite do mapa de grade atual no mapa já existente, assim alinhando a nova imagem ao mapa existente e adicionando o que for necessário, montando assim o mapa do ambiente encontrado, (KOHLBRECHER et al., 2011).

Devido à atual precisão dos sensores laser e de suas capacidades de medidas, atualmente é visto em alguns sistemas que a precisão da odometria estimada a partir dos dados laser (mapa de grade) tem sido de melhor qualidade que de dados provenientes de unidades de medida inerciais provinda de unidades inerciais, (KOHLBRECHER et al., 2011).

Para a estimação da posição do corpo móvel, bem como também a trajetória do mesmo é necessário um filtro estimador. Dessa forma é escolhido é o filtro de *Kalman* estendido, (KOHLBRECHER et al., 2011). O filtro é empregado no modelo do sistema de posição quando se têm os resultados provenientes da unidade inercial, ou do valor de posição estimado utilizando o mapa de grades obtido anteriormente que são dados pelas Equações (2.7), (2.8) e (2.9):

$$\dot{\Omega} = E_\Omega \omega, \tag{2.7}$$

$$\dot{P} = v, \tag{2.8}$$

$$\dot{v} = R_\Omega * a + g, \tag{2.9}$$

onde Ω representa os ângulos de Euler, P a posição, v a velocidade, a é a aceleração, R_Ω é a matriz cosseno de direção e E_Ω mapeia as taxas de variação dos ângulos de Euler, (KOHLBRECHER et al., 2011).

O filtro de *Kalman* estendido é aplicado no modelo acima, principalmente para as Equações 2.7 e 2.9. Nota-se que a Equação do sistema não é linear devido aos termos dos ângulo de Euler nas matrizes E_Ω e R_Ω, portanto, um filtro não linear deve ser utilizado. As medidas inerciais são considerados como entradas de sistema conhecidas. A atualização

de velocidade e posição é uma integração pura das acelerações medidas e o sistema se tornaria instável sem a realimentação adicional através das atualizações de medição. A contramedidas habituais para evitar que a estimativa do estado cresça indefinidamente quando não há medições disponíveis são pseudo atualizações de velocidade zero, logo que a variância atinge um certo limiar e estabilidade não podem ser assegurados de outra forma, (KOHLBRECHER et al., 2011).

Dessa forma o mapa é criado e externado para o *RViz* (ambiente de visualização tridimensional), bem como também a posição e trajetória do corpo móvel durante a execução do sistema de *SLAM*, (HERSHBERGER; GOSSOW; FAUST, 2018) e (KOHLBRECHER et al., 2011).

2.8 RTAB-MAP

O algorítimo desenvolvido na Universidade de Sherbrooke em Quebec no Canadá, tem por objetivo implementar um algorítimo de detecção por ciclos fechados (em inglês, *loop closure detection*) utilizando abordagem de gerenciamento de memória, limitando o tamanho do mapa para que o ciclo fechado sempre se mantenha dentro do tempo pré fixado, atendendo assim aos requisitos on-line para mapeamento de ambiente a longo prazo e em larga escala. (LABBÉ; MICHAUD, 2019)

O algorítimo de RTAB-Map (mapeamento baseado em aparência em tempo real ,do inglês: Real-Time Appearance-Based Mapping) se tornou uma biblioteca em C++ multi plataforma e um pacote para ROS. O algorítimo tem as seguintes premissas, (LABBÉ; MICHAUD, 2019):

- Processamento *Online*: a saída do algorítimo *SLAM* deve ser limitado ao máximo atraso entre amostras do sensor em questão.
- Odometria robusta e com baixo desvio: embora o laço de repetição fechado seja capaz de corrigir a maior parte dos desvios de odometria, nos cenários reais o robô dificilmente é capaz se localizar corretamente no mapa, seja porque está explorando novos ambientes ou porque há falta de detalhes do cenário. O algorítimo utiliza das medições dos sensores para inferir a odometria corrigindo possíveis desvios de medição.
- Localização robusta: a abordagem *SLAM* deve ser capaz de reconhecer quanto revisita um local passado (para o laço de repetição fechado de detecção) para corrigir o mapa.
- Geração prática de mapa e exploração: as principais abordagens de navegação são baseadas em grades de ocupação, logo, é benéfico o desenvolvimento de abordagem de

SLAM que pode fornecer um mapa de grade 3D ou 2D pronta para isso, facilitando a integração.

O pseudo código do algorítimo de detecção por laço fechado, utilizado pelo RTAB-Map, está representado na Figura 3.

Figura 3 – Pseudo código do algoritimo de *loop* fechado utilizado no RTAB-Map

Algorithm 1 RTAB-Map

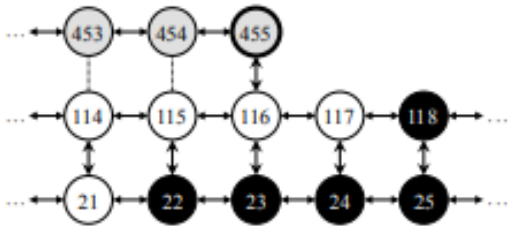
1: $time \leftarrow$ TIMENOW() ▷ TIMENOW() returns current time
2: $I_t \leftarrow$ acquired image
3: $L_t \leftarrow$ LOCATIONCREATION(I_t)
4: **if** z_t (of L_t) is a bad signature (using T_{bad}) **then**
5: Delete L_t
6: **else**
7: Insert L_t into STM, adding a neighbor link with L_{t-1}
8: Weight Update of L_t in STM (using $T_{\text{similarity}}$)
9: **if** STM's size reached its limit (T_{STM}) **then**
10: Move oldest location of STM to WM
11: **end if**
12: $p(S_t|L^t) \leftarrow$Bayesian Filter Update in WM with L_t
13: Loop Closure Hypothesis Selection ($S_t = i$)
14: **if** $S_t = i$ is accepted (using T_{loop}) **then**
15: Add loop closure link between L_t and L_i
16: **end if**
17: Join $trash$'s thread ▷ Thread started in TRANSFER()
18: RETRIEVAL(L_i) ▷ LTM → WM
19: $pTime \leftarrow$ TIMENOW() $- time$ ▷ Processing time
20: **if** $pTime > T_{\text{time}}$ **then**
21: TRANSFER() ▷ WM → LTM
22: **end if**
23: **end if**

Fonte: Labbe e Michaud (2013)

O pacote de mapeamento por aparência utiliza um laço de repetição fechado de detecção com e um número limitado de amostras para garantir a condição de medição em tempo real associando as amostras obtidas com o mapa global quando conveniente. Quando o número de amostras no mapa torna o tempo de processamento para encontrar correspondências maior que um limite de tempo, o algorítimo transfere locais com menor probabilidade de causar detecção de fechamento de *loop* da WM (memoria de trabalho, do inglês: *Working Memory*) do robô para LTM, para que eles não participem da detecção de laço de repetição fechados. No entanto, se um laço de repetição fechado for detectado, os locais vizinhos podem ser recuperados e trazidos de volta ao WM para serem considerados em futuras detecções de *loops* fechados, (LABBE; MICHAUD, 2013).

A Figura 4 ilustra o funcionamento do sistema de maneira gráfica os locais por onde o robô passou após três percursos na mesma região. Cada um dos locais é representado por um índice de acordo com o tempo em que foi gerado e um peso, os locais são vinculados em um gráfico de *loops* ou vizinhos fechados. Essas conexões representam locais próximos uns aos outros de acordo com o tempo e espaço de navegação do robô, (LABBE; MICHAUD, 2013).

Figura 4 – Representação gráfica de locais, onde setas verticais são *links* de *loop* fechado e setas horizontais são *links* vizinhos.



Fonte: Labbe e Michaud (2013)

Apesar do pacote RTAB-Map fornecer recursos de mapeamento 3D, no trabalho em questão optou-se por tratar apenas da abordagem 2D, uma vez que o mapeamento 2D já fornece uma informação suficiente para a movimentação de robôs móveis utilizado no projeto em questão. O mapeamento 3D é bastante custoso computacionalmente uma vez que para realizar o mapeamento é necessário processar e armazenar muito mais pontos em memória e tudo isso sem acrescentar muitas analises relevantes ao mapeamento, (GOULART; TERRA, 2020). Tendo isso em vista, esse trabalho focou em se aproveitar das técnicas fornecidas pelo pacote do RTAB-Map de redundância sensorial.

3 MATERIAIS E MÉTODOS

Essa sessão irá tratar dos programas, modelos e pacotes utilizados no desenvolvimento desse projeto. Bem como também mostrará o ponto de partida proveniente dos dois trabalhos de iniciação científica que foram produzidos anteriormente a este.

3.1 AMBIENTE DE SIMULAÇÃO

Para que fosse possível desenvolver o projeto foi utilizado um ambiente simulado proporcionado pela integração do *ROS* com o *Gazebo*. O *Gazebo* é um programa *open source* com um ambiente simulado voltado para modelos robóticos. O ambiente tridimensional gerado pelo programa reproduz nos modelos robóticos a ação das leis da física. No Gazebo também é possível realizar a simulação de sensores como o que está sendo utilizado nesse projeto. No programa também é possível gerar mapas para que o modelo robótico possa interagir com os objetos e paredes presentes no ambiente. (KOENIG; HOWARD, 2004)

A modelagem robótica dentro do Gazebo é representada por modelos, corpos e junções. Um modelo é qualquer objeto que possua uma representação física. Os modelos são composições de ao menos um corpo rígido e talvez junções e sensores. Os corpos são representações dos blocos básicos de um modelo. Sua representação física normalmente possui uma forma geométrica como cubos, esferas, cilindros, planos e linhas. Cada corpo possui diversos aspectos físicos tais como: massa, atrito, momento de inercia, cor, etc. Já as junções fornecem mecanismos para conectar os corpos juntos para formar relações cinemáticas e dinâmicas. As junções também podem ser utilizadas como motor. Quando uma força é aplicada a uma junção, o atrito entre o corpo conectado e os outros corpos gera o movimento. (KOENIG; HOWARD, 2004)

3.1.1 MODELAGEM ROBÓTICA

Para realizar os experimentos em um ambiente simulado foi necessário desenvolver um modelo robótico móvel capaz de suportar todos os sensores desejados no projeto. O modelo escolhido foi um modelo genérico com duas rodas e um controle diferencial. O modelo 3D do robô pode ser visto na Figura 5.

Vale ressaltar que esse modelo robótico é bastante conhecido e utilizado em conjunto com o Gazebo, entretanto muitos dos modelos disponibilizados possuem parâmetros de massa e inércia diferentes dos utilizados nesse projeto, sendo necessária uma nova modelagem baseada nesse molde.

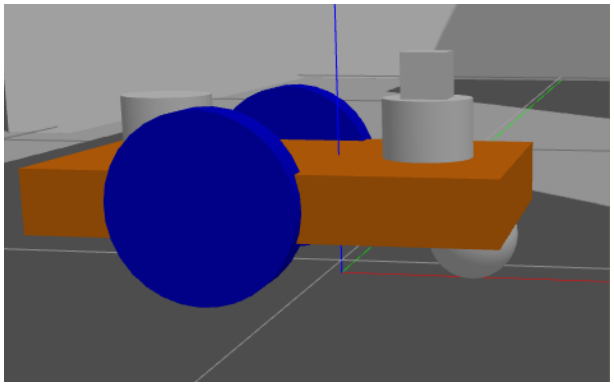


Figura 5 – Modelo 3D do robô desenvolvido

O chassi do robô foi modelado em um formato de paralelepípedo com dimensões de 0.5m x 0.3m x 0.07m e massa de 5kg. Para o cálculo do momento de inércia foi utilizada a seguinte Equação ((3.1)). (WEISSTEIN, 2009)

$$I = \begin{bmatrix} \frac{1}{12}m(h^2+d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2+d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(h^2+h^2), \end{bmatrix} \tag{3.1}$$

onde m é a massa; h é a altura; w é a largura; d é a profundidade, e obtém-se a Equação (3.2).

$$I = \begin{bmatrix} 0.0395416666667 & 0 & 0 \\ 0 & 0.106208333333 & 0 \\ 0 & 0 & 0.141666667 \end{bmatrix}. \tag{3.2}$$

Já para as rodas, que possuem um formato cilíndrico, a matriz de momento de inércia assume a Equação (3.3).(WEISSTEIN, 2009)

$$I = \begin{bmatrix} \frac{1}{12}m(3r^2+l^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+l^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix}, \tag{3.3}$$

onde o raio da roda é $0.1m$; a largura da roda é $0.04m$ e a massa $0.2m$. Obtém-se, então, a Equação (3.4) para o momento de inércia das rodas.

$$I = \begin{bmatrix} 0.000526666666667 & 0 & 0 \\ 0 & 0.000526666666667 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \tag{3.4}$$

Após modelar as formas geométricas do robô, foi necessário definir as junções. Foram estabelecidas 3 junções, sendo uma delas estática para fixar o sensor (LIDAR, ou

KINECT, ou ambos) ao chassi do robô e as outras duas junções foram utilizadas como motor.

Ainda na modelagem do modelo robótico, foi necessário especificar os parâmetros do sensor Kinect v2. Os parâmetros foram definidos conforme o sensor real presentes na tabela (2). Além disso também foi necessário especificar os parâmetros necessários para os sensores a laser, seguindo as mesmas especificações presentes na tabela (1). Os sensores laser foram modelados com um formato cilíndrico, enquanto o sensor *KINECT* foi modelado com um formato cúbico.

Para finalizar, a modelagem foi estabelecido o movimento diferencial das rodas do robô, sendo gerado um tópico de controle da velocidade e um de odometria para checar se a velocidade está sendo corretamente aplicada. Caso o tópico de odometria não corresponda ao tópico de velocidade, pode indicar que o robô está preso em alguma parede devido a um erro no algorítimo de navegação.

Todos os modelos criados, com exceção do corpo do chassi, sofrem uma mudança de sistema de coordenadas para que todos os outros modelos sejam representados com relação ao chassi. Já no caso do sensor *KINECT* é necessário realizar mais uma mudança em seu sistema de coordenadas para a representação no programa *RViz*, uma vez que o sensor possui uma representação voltada para o eixo-Z, entretanto a frente do sensor está presente no eixo-X.

3.1.2 LASER DE LINHA E MODELAGEM

Partindo dos resultados dos projetos desenvolvidos anteriormente utilizando sensores reais, foi necessário uma modelagem do sensor a ser adicionado no ambiente de simulação a fim de representar o mais fiel possível do que foi feito anteriormente.

A modelagem foi feita utilizando os parâmetros técnicos do sensor laser utilizado anteriormente: o *Sensor Scanning Laser RangeFinder Smart - URG mini - UST-10LX (UUST003)* que pode ser visto na Figura 6 abaixo (ROS COMPONENTS, 2016). A Hokuyo, empresa especializada em fabricação de sensores ópticos e dispositivos de automação com ampla aplicação em engenharia (HOKUYO AUTOMATIC CO., LTD, 2014).

Os valores técnicos do sensor podem ser vistos na Tabela 1 abaixo. Tais valores foram utilizados para a modelagem do sensor de maneira fiel ao real.

Figura 6 – *Sensor Scanning Laser RangeFinder Smart - URG mini - UST-10LX (UUST003)* da *Hokuyo Automatic CO. LTD.*

Fonte: ROS Components (2016)

Tabela 1 – Especificações técnicas do UST-10LX.

Características	Valores
Nº Modelo	UST-10LX
Precisão	+-40 mm
Interface	Ethernet
Iluminação ambiente	15.000 lux
Temperatura ambiente / umidade	-10 to +50°C
Peso	130 g
Tensão	10 to 30 V
Resolução angular	0.25°
Alcance	270º
IP	IP65
Ambiente	Indoors
Distancia	0.06 - 10 m
Frequência de varredura	40 Hz

Partindo do princípio que o robô pesa 5kg e o sensor possui um peso de 130g, é plausível afirmar que o peso do modelo do sensor está incorporado no peso total do modelo do robô, fazendo com que os parâmetros principais para serem levados em consideração na modelagem são as: *Scan frequency*; *Distance*; *Range*; e *Resolution angle*.

3.1.3 SENSOR KINECT V2

O sensor Kinect V2, presente na Figura 7, teve suas especificações técnicas obtidas, juntamente com a versão anterior do sensor para fins de comparação. As especificações dos dois sensores estão presentes na Tabela 2. (ASHLEY, 2014)(Meng Dong et al., 2016)

Figura 7 – Sensor Microsoft KINECT v2

Fonte: Kinect for Windows Team (2014)

Tabela 2 – Especificações do Kinect e do Kinect v2

Feature	Kinect	Kinect v2
Color Camera	640 x 480 @30 fps	1920 x 1080 @30 fps
Depth Camera	320 x 240	512 x 424
Max Depth Distance	4.5 M	8 M
Min Depth Distance	40 cm in near mode	50 cm
Depth Horizontal Field of View	57 degrees	70 degrees
Depth Vertical Field of View	43 degrees	60 degrees
Tilt Motor	yes	no
Skeleton Joints Defined	20 joints	25 joints
Full Skeletons Tracked	2	6
USB Standard	2.0	3.0

O sensor pesa 1.5kg e esse peso foi incorporado a massa do robô afim de facilitar a modelagem robótica, uma vez que a modelagem em si não é o foco o projeto, e sim os resultados de mapeamento.

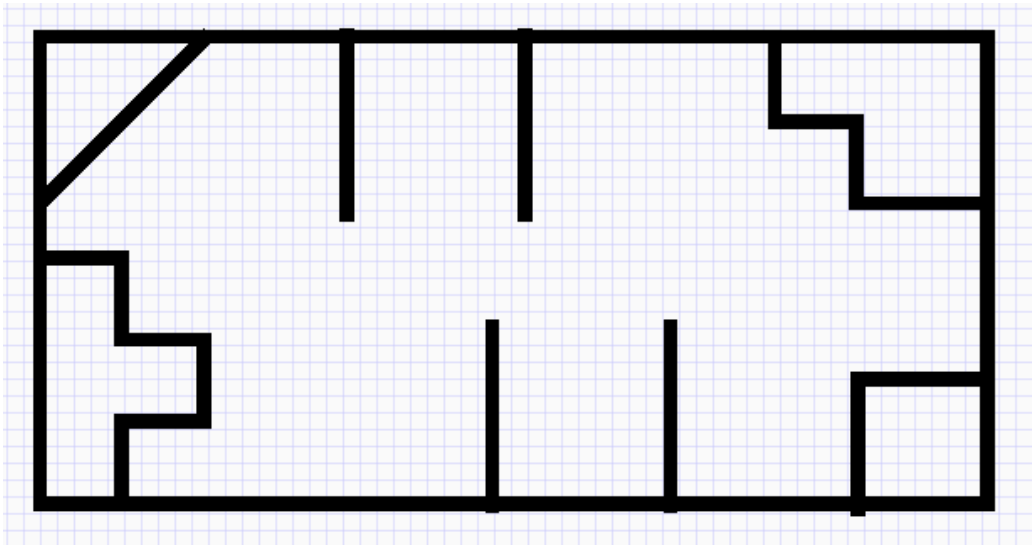
3.1.4 MODELAGEM DO AMBIENTE SIMPLES

Para a realização de algumas simulações, foi criado um ambiente com algumas características específicas para a orientação do robô durante o processo de mapeamento. Partindo do princípio que o alcance máximo do sensor laser é de $10m$ adotou-se que o ambiente teria no máximo $10m$ de comprimento e teria uma forma primariamente retangular.

Dessa forma um ambiente não muito extenso, mas com características que permitissem que o mapeamento seja possível de maneira rápida, dessa forma foi pensado em um

retângulo de $10m \times 5m$. Foram adicionados peculiaridades em cada ponta do retângulo para facilitar a visualização do mapa gerado pelo método de *SLAM*, e por identificar em qual momento o erro ocorreu. O modelo bidimensional pode ser visto na Figura 8.

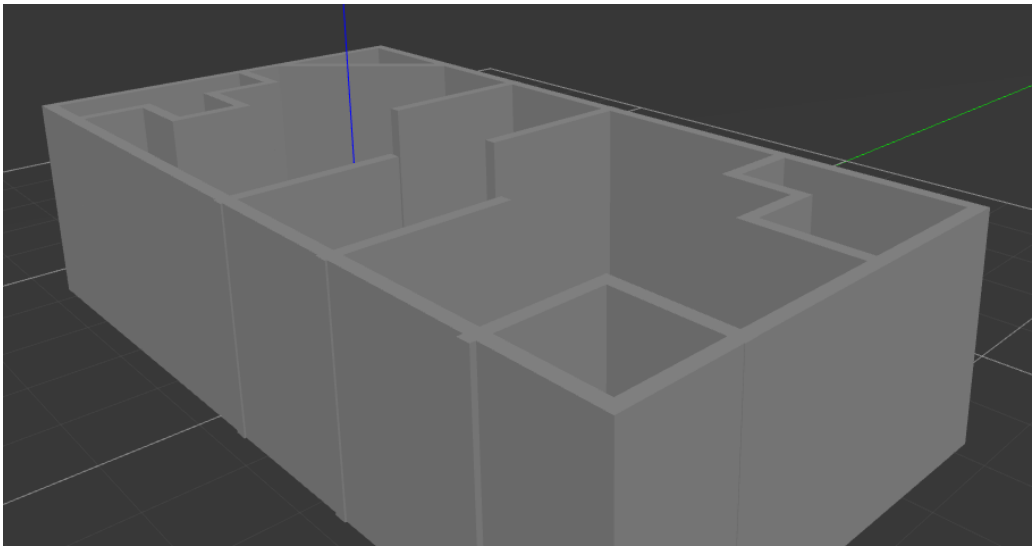
Figura 8 – Desenho feito no campo de modelagem de ambiente dentro do *gazebo*.



Fonte: Autor

Obstáculos no meio do retângulo foram adicionados para forçarem a movimentação do robô de maneira não linear, forçando assim a execução de curvas para poder obter imagens dos cantos. A visualização tridimensional do ambiente pode ser visto na Figura 9. As peculiaridades existem, não só para que seja possível verificar em qual posição o robô se encontra, mas também para facilitar o mapeamento caso o robô queira se movimentar de maneira linear, devido ao método de mapeamento realizar um processo de *match* onde imagens são encaixadas umas nas outras, é necessário que existam diferenças entre cada medição realizada após um movimento (KOHLBRECHER et al., 2011). Por exemplo se o robô se movimenta através de um corredor com duas paredes paralelas e os parâmetros de parede das pontas não podem ser visto, caso o robô se movimente de uma maneira linear paralelamente às paredes do corredor, o mapeamento são é possível, pois a imagem entre um mapeamento e outro não irá mudar com a movimentação do robô (KOHLBRECHER et al., 2011).

Figura 9 – Visão tridimensional do ambiente desenhado pela ferramenta do *gazebo*.



Fonte: Autor

Quando o robô é adicionado à simulação, este é carregado diretamente no centro do ambiente.

3.2 AMBIENTE DE VISUALIZAÇÃO

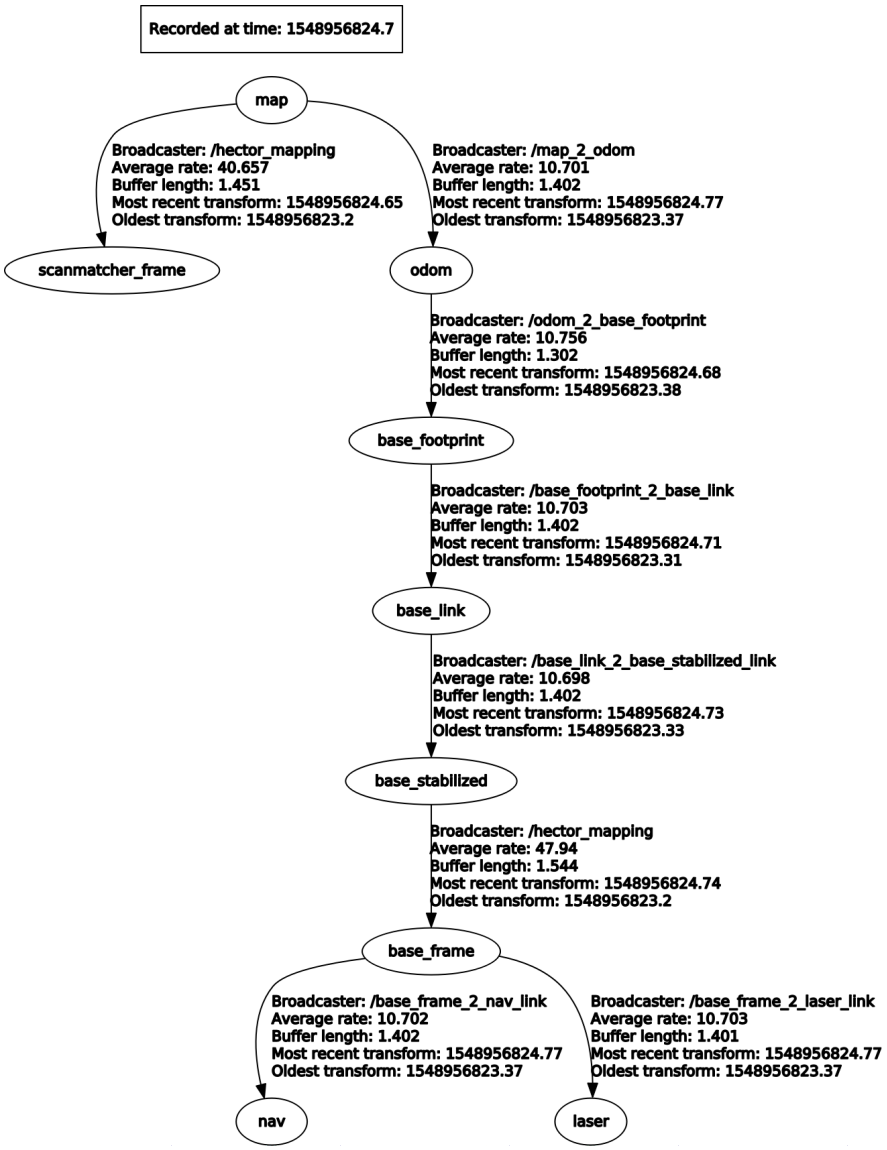
O *RViz* é um ambiente de visualização tridimensional que é implementado para operação dentro do *ROS*, este ambiente lê as mensagens que estão contidas dentro dos tópicos de comunicação entre os nós de aplicação, e através da transformação homogênea que pode ser aplicada entre dois *frameworks* de mensagens, o ambiente consegue interpretar e escrever essas mensagens, não só escrever, como também atualizá-las em tempo real.(HERSHBERGER; GOSSOW; FAUST, 2018)

3.3 MAPEAMENTO UTILIZANDO SENSOR A LASER DE LINHA E PACOTE ROS PARA CORPOS MÓVEIS

Esta seção trata sobre um dos trabalhos que serviu de parte da inspiração para o desenvolvimento deste trabalho em questão, este trabalho utilizou dados reais com aplicação experimental, aplicado nos anfiteatros da secretaria de engenharia elétrica da escola de engenharia de São Carlos. Trata-se do trabalho onde se aplica o método de *SLAM* descrito na seção 2.7.

O trabalho apresenta a configuração e utilização do pacote de *SLAM* dentro do ambiente ROS, aplicado o método de mapeamento com o *Sensor Scanning Laser RangeFinder Smart - URG mini - UST-10LX (UUST003)* (BORDON; TERRA, 2019). O trabalho mostrou a utilização do pacote *Hector_slam* no ambiente do ROS e suas limitações quanto à movimentação do corpo móvel durante a execução do mapeamento.

Figura 10 – Árvore de transformações homogêneas para a execução correta do pacote *Hector_slam*.



Fonte: Bordon e Terra (2019)

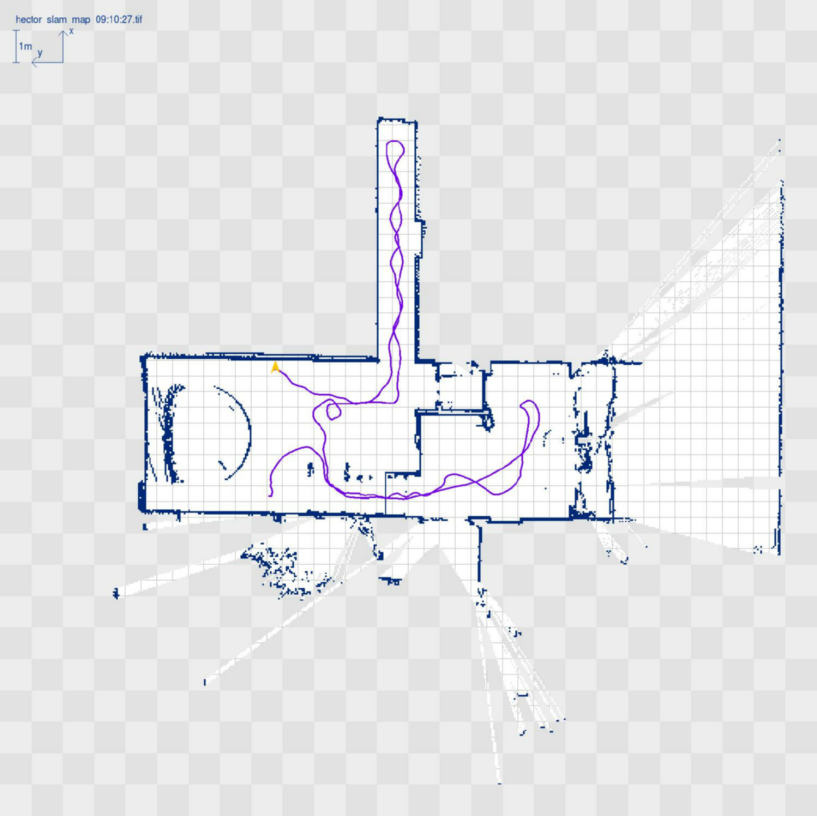
O trabalho mostra a configuração das mensagens de medidas a fim de correlacionar corretamente todas as informações de maneira correta durante a execução do método de *SLAM*. A sequência de como devem ser executadas as transformações homogêneas entre as mensagens dos tópicos e nós e como a sequência de *frames* é correlacionada entre si. A descrição da sequência de *SLAM* pode ser vista na Figura 10. A configuração dessa árvore de transformações garante que todos os processos que envolvem o pacote funcionem em conjunto, desde as funções de posicionamento e trajetória, e também junto com as funções de mapeamento e escrita de mapa dentro do ambiente do *RViz*.

O mapeamento obtido, quando confrontado com a planta baixa do prédio usado como referência, comprova o método de mapeamento. Outro método que comprovou a

validade do método de mapeamento foram as medidas realizadas nos ambientes no qual o sensor foi aplicado. Medido as distâncias entre paredes e sabendo que o método de *SLAM* utilizado exporta um mapa com escala real em metros foi possível obter o erro médio percentual da realização das medidas. Segundo o trabalho, o erro médio observado foi de: $4,2255\%$, várias medições no ambiente foram realizadas para a aferição da média de de referência.

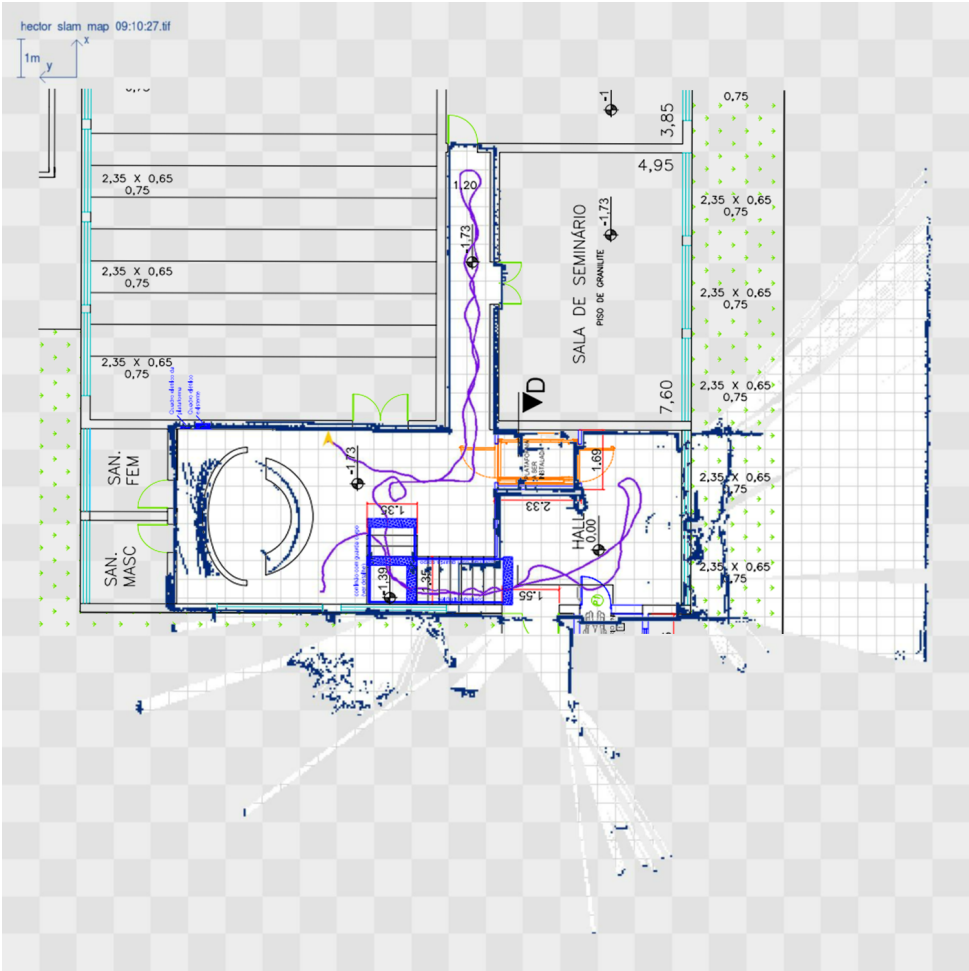
O resultado do mapa do *hall* de entrada do anfiteatro juntamente com o confronto com a plant baixa pode ser observado nas figs. 11 e 12

Figura 11 – Mapa obtido através do *hector SLAM* com um sensor laser de linha apenas realizado no anfiteatro Armando Toshio Natsume EESC - USP



Fonte: Bordon e Terra (2019)

Figura 12 – Confontro do mapa da figura 11 com a plata baixa do anfiteatro.



Fonte: Bordon e Terra (2019)

4 RESULTADOS

Os resultados juntam algumas abordagens que foram aplicadas com o pacote *Hector SLAM*, a fim de conseguir gerar redundância entre os sensores. As abordagens incluem entre acrescentar novas características ao método escolhido para mapeamento, bem como também melhorar as características individuais dos sensores, como amplitude de medição e como referência durante uma aplicação. Também foi aplicado a fusão sensorial durante a aplicação com o pacote de *SLAM RTAB-Map*

4.1 MAPEAMENTO UTILIZANDO HECTOR_SLAM

Utilizando-se das configuração encontradas em Bordon e Terra (2019) foi realizado um mapeamento do ambiente simples modelado na Seção 3.1.4 para realização de testes e comprovações das configuração encontradas para o funcionamento do sistema de simulação e mapeamento. Para tal, o resultado obtido do mapeamento do ambiente simples pode ser visto na Figura 13 que representa o funcionamento do pacote do *Hector_slam* para a simulação utilizando um sensor *laser* de linha no robô modelo (Figura 5).

Figura 13 – Mapeamento do ambiente simples com um sensor *laser*.

Fonte: Autor

Este mapeamento será utilizado como ferramenta de comparação para os mapeamentos realizados com a redundância de sensores.

4.2 INTEGRAÇÃO COM DOIS SENSORES *LASER*

Para a integração de dois sensores laser, primeiro é necessário compreender a forma que o *ROS* trata sua conexão entre tópicos, nós e *frames*, dessa forma é possível compreender quais técnicas foram utilizadas para poder utilizar dois sensores simultaneamente.

Tópicos são veículos responsáveis por transmitir mensagens através de nós durante a execução de um ou mais pacotes, dessa forma é de extrema importância compreender que um tópico é único. Um nó é um processo computacional, que executa um programa simples até executar uma cadeia de programas complexos, em suma, é um processo que executa algum tipo de computação. Assim como o tópico o nó é uma entidade única dentro do ROS. Embora uma sequência de nós possa ser executada junto para desenvolver uma única aplicação, por exemplo, para a visualização de uma imagem de um sensor laser são necessários alguns nós para que a informação seja vista na tela. São necessários pelo menos dois nós, um nó responsável por gerar/receber mensagens do sensor (onde receber pode envolver a conexão do sensor com a máquina ou não) e um nó responsável por mostrar essas informações na tela, como por exemplo uma aplicação gráfica descrita na Seção 3.2. Estas informações estão descritas de maneira completa na Seção 2.4 deste trabalho.

Sabendo que os tópicos e nós são únicos dentro do ROS, ou seja, cada um possui um nome específico e este não pode ser duplicado. Para o mapeamento simultâneo de dois sensores conectados à mesma máquina seria necessário abrir a raiz de toda a programação do pacote inicial de mapeamento *Hector_slam* e então conseguir alterar todos os nomes de tópicos e nós dentro do pacote em questão (extensas linhas de códigos e diversos programas, sem incluir em todo o adicional que deveria ser feito para poder criar então um novo pacote), mostrou-se um caminho sem fundamento a se seguir. Da mesma forma dois mapeamentos distintos, ocasiões diferentes, de um mesmo ambiente realizado por sensores diferentes, com caminhos seguidos diferentes, a fim de obter o mesmo resultado de mapeamento de um espaço previamente delimitado e conhecido, é tão eficiente quanto um mapeamento simultâneo de dois sensores diferentes.

Dessa maneira a utilização de dois sensores de mesma função deveria ser atacada de outra maneira. Utilizou-se assim dois sensores, a fim de serem complementares, aumentando a quantidade de amostras a serem obtidas durante uma medição, aumentando o espectro de visão do sensor.

4.2.1 ABORDAGEM DE MESMO *FRAME* E/OU TÓPICO DE MENSAGEM

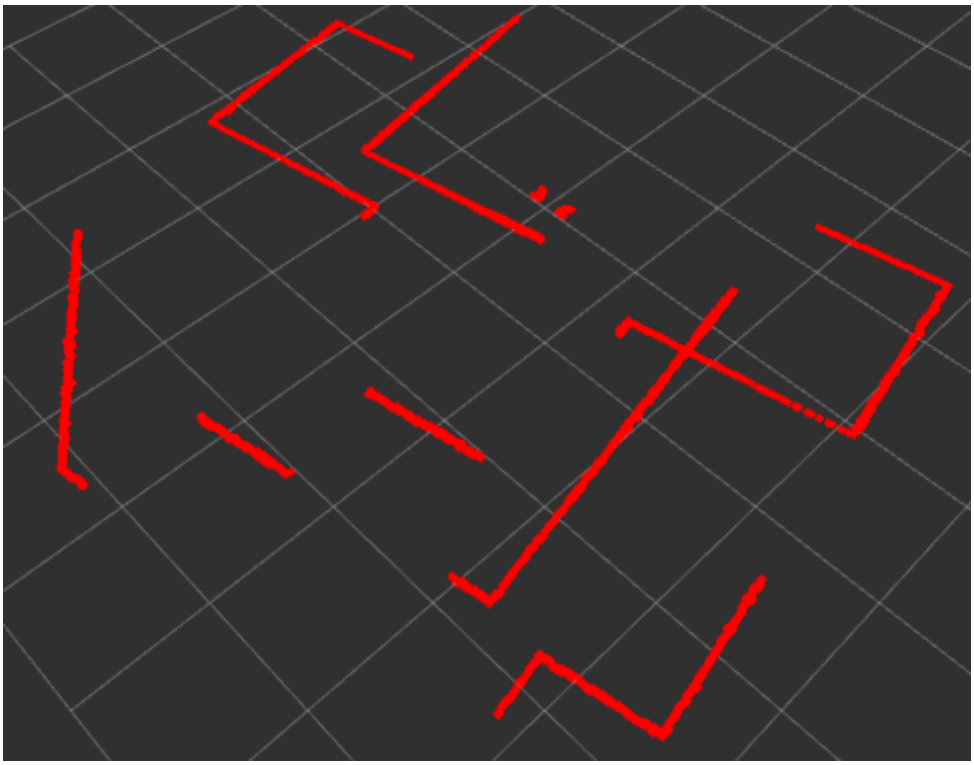
A primeira tentativa de integração com dois sensores laser para a execução do algorítimo de *SLAM* foi a tentativa de publicar as mensagens dos dois sensores no mesmo tópico, com o mesmo *frame* para poder identificar as mensagens e transformá-las em dados úteis para o plano cartesiano do *RViz*. Através dos *frames* é possível correlacionar as

mensagens que transitam entre os tópicos no ambiente gráfico.

Para a abordagem de mesmo *frame* de mensagem é necessário que as mensagens dos laser possuem o mesmo nome de *frame* para que quando a transformação homogênea seja feita, esta seja aplicada às mensagens dos dois sensores igualmente. As mensagens devem ser publicadas no mesmo tópico para que a execução do algorítimo de *SLAM* possa se inscrever nesse tópico e ler as mensagens (o método de *SLAM* utilizado aceita apenas um tópico de inscrição para a coleta das mensagens de distância).

Dentro do ambiente de simulação, com o ambiente simples utilizado, com um robô com dois sensores laser ambos virados de costas um contra o outro (o eixo de referência dos sensores estão dispostos de um ângulo de 180° entre si), dessa forma os sensores devem complementar os valores de medidas e assim expandir a área de medida coberta pelo sensor. Utilizando do recurso do *RViz* é possível visualizar as mensagens dos lasers dentro do plano cartesiano. O resultado simples dos sensores pode ser visto na Figura 14 abaixo.

Figura 14 – Mensagens de dois sensores *laser* de linha publicadas no mesmo tópico com o mesmo *frame* de referência.



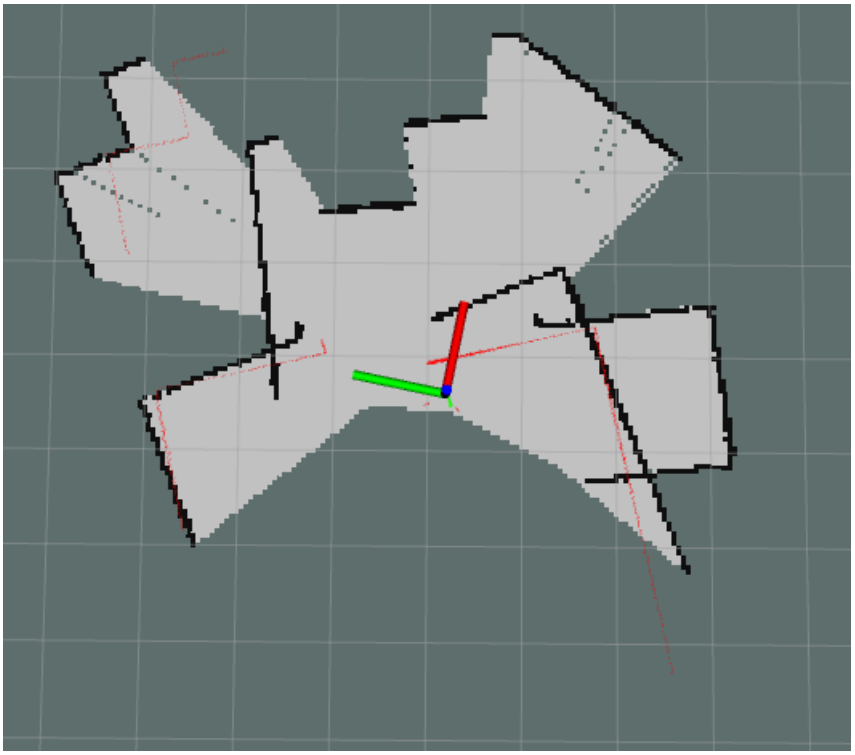
Fonte: Autor

Observando a Figura 14 é possível ver que as delimitações entre frente e trás se misturam, pois não é dito em nenhum momento que há uma diferença de posição entre os sensores, pois essa informação é especificada quando a transformação homogênea é feita através da função *tf static_transform_publisher*, quando este pacote é carregado o *frame* do mundo do ambiente de simulação (especificado por padrão como o *map*) é

correlacionado com as mensagens provenientes do tópico dos lasers, e como a transformação só informa um *offset* possível de X, Y, Z, *yaw*, *pitch* e *roll*.

Dessa forma é impossível informar a posição dos dois sensores caso elas sejam diferentes utilizando esta abordagem. Mesmo assim a técnica de *SLAM* para sensores *laser* de linha apresentada na Seção 3.3 para a tentativa de mapeamento não ser desperdiçada por decisões precipitadas. O resultado do mapeamento pode ser visto na Figura 15 e nota-se que o método de mapeamento não consegue permanecer estático, da mesma forma que a simulação tenta se comportar. A resposta dos sensores não se dá de maneira estática como é visto na Figura 14 durante a visualização das mensagens é possível perceber que a imagem oscila entre um sensor e outro, quando os *timestamps* das mensagens se coincidem, ambas são representadas juntas no ambiente de visualização. É possível verificar na Figura 16 que o robô parado se comporta como se estivesse em um movimento circular, pois há escrita na linha de posição do robô.

Figura 15 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô parado.

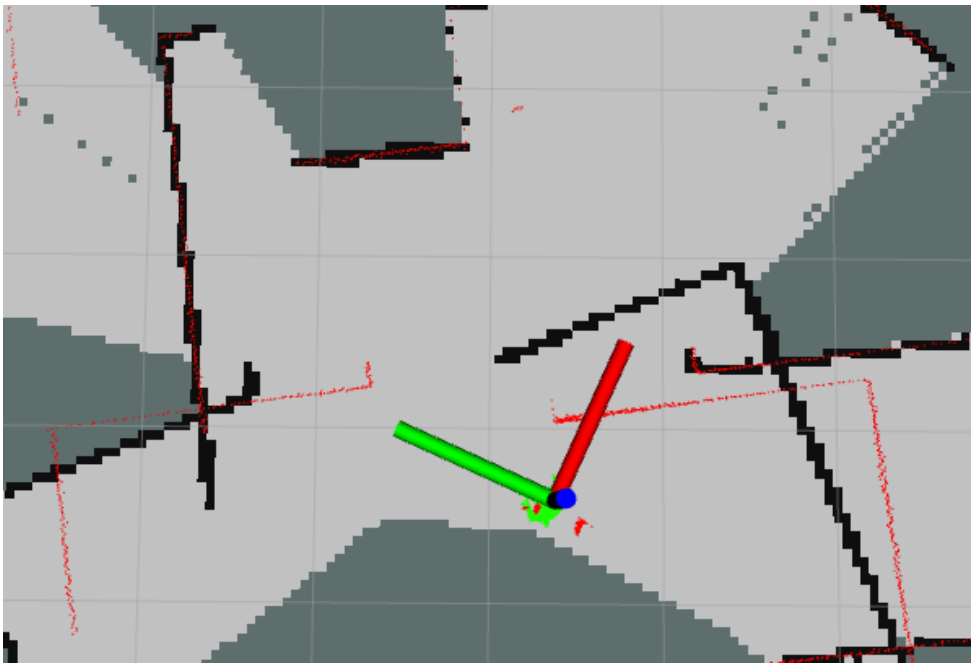


Fonte: Autor

Para certificar que o método de mapeamento estava em pleno funcionamento foi feita a verificação do comportamento das transformações homogêneas. Através do *RViz* é possível verificar se as transformações estão sendo executadas e conectadas. As correlações podem ser checadas na Figura 17.

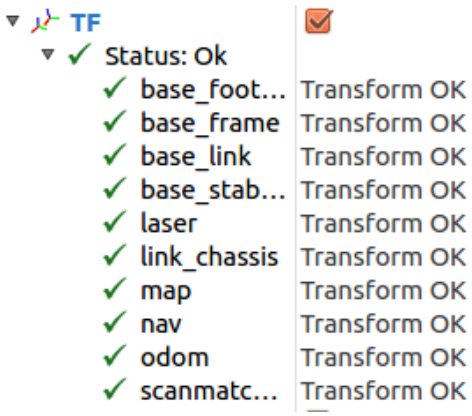
Realizando algumas movimentações com o robô através de comandos diretamente

Figura 16 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô parado aproximado.



Fonte: Autor

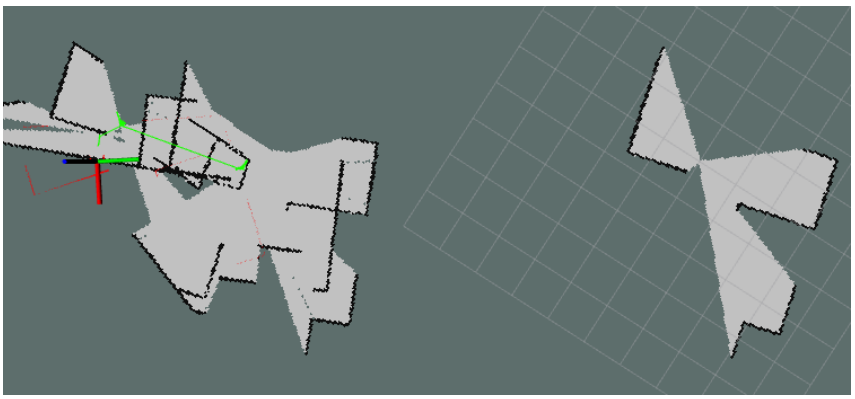
Figura 17 – Seção das transformações homogêneas sendo executadas e correlacionadas com as mensagens lidas pelo *RViz*.



Fonte: Autor

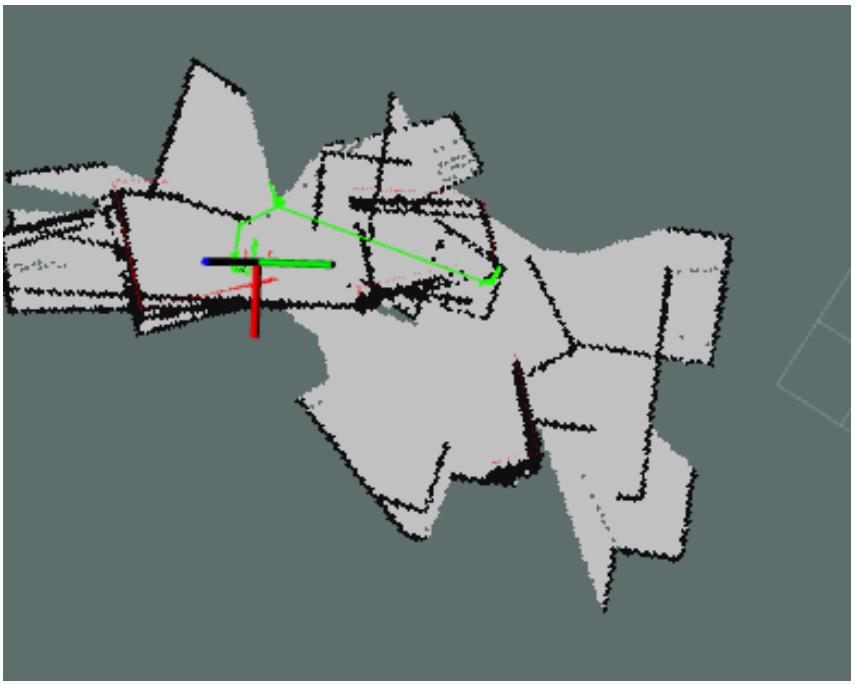
do terminal é possível observar que o mapeamento realizado pelo método com os dois sensores publicando informações no mesmo tópico com o mesmo *frame* de referência na Figura 18, após a movimentação mantendo o robô parado no lugar o algorítimo continua a escrever dados no mapa, pois a sobreposição de mensagens dos sensores laser simula como se este estivesse apresentando uma movimentação errática, como pode ser visto na Figura 19.

Figura 18 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô tendo se movimentado.



Fonte: Autor

Figura 19 – Tentativa de mapeamento utilizando o mesmo tópico de publicação e mesmo *frame* de referência para os dois sensores *laser* com o robô parado após movimentação.

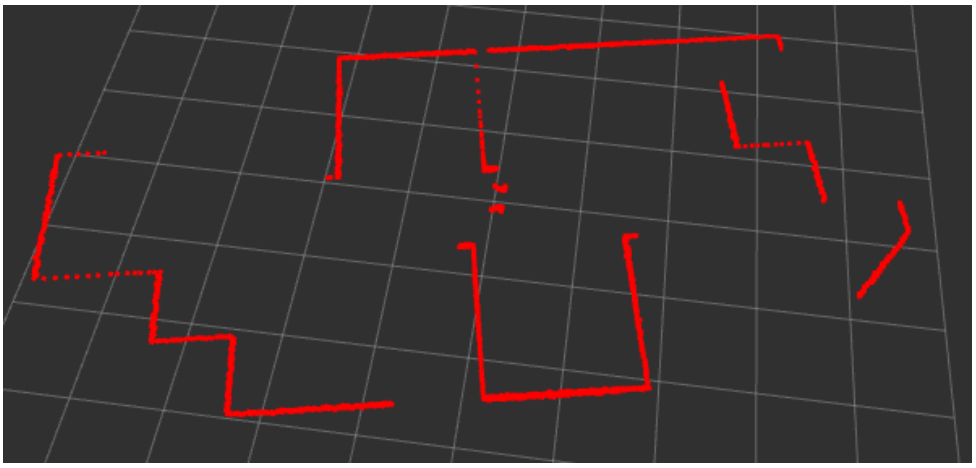


Fonte: Autor

Agora, fazendo com que os dois sensores publiquem cada um em seu tópico com seu respectivo *frame* de referência, e com uma transformação homogênea específica para cada conjunto de mensagens é possível perceber no *RViz* apenas com as mensagens dos dois sensores é possível ver o ambiente de maneira mais clara. É possível observar o ambiente

simples modelado através da representação gráfica das mensagens dos sensores, como pode ser visto na Figura 20.

Figura 20 – Representação gráfica dos dois sensores em conjuntos, cada um com sua transformação homogênea.



Fonte: Autor

Foi feita uma tentativa de colocar no arquivo de *launch* do *ROS* os dois tópicos de publicação de cada sensor laser, dessa forma não foi obtida uma resposta do algorítimo de *SLAM*, pois este só aceita inscrição em apenas um tópico de publicação de mensagens de um sensor *laser* de linha.

As linhas de configuração em linguagem XML dos arquivos podem ser vistam abaixo, primeiramente no arquivo de iniciação do pluguin de mapeamento e em seguida a transformação aplicada no arquivo de iniciação de todo o pacote do *Hector_slam*.

```
<arg name="scan_topic" default="scan1 scan2"/>

<node pkg="tf" type="static_transform_publisher"
    name="base_frame_2_laser_link_1" args="0.15 0 0.08 0 0 0
    /base_frame /laser1 100"/>
<node pkg="tf" type="static_transform_publisher"
    name="base_frame_2_laser_link_2" args="-0.15 0 0.08 3.1415 0 0
    /base_frame /laser2 100"/>
```

E dentro do modelo *.gazebo* do robô, o sensor laser acoplado segue o mesmo nome de tópico e *frame*, como pode ser visto nas linhas abaixo. Dentro desses arquivos está toda a modelagem do sensor descrita na Seção 3.1.2, e os parâmetros que são carregados a fim de simular o sensor real utilizado em um dos trabalhos que serviu de inspiração para este.

```
<gazebo reference="laser1">
    <sensor type="ray" name="head_hokuyo_sensor1">
```

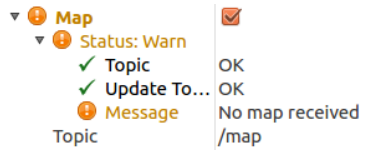
```
      ...
        <topicName>/scan1</topicName>
        <frameName>laser1</frameName>
      ...
</gazebo>

<gazebo reference="laser2">
  <sensor type="ray" name="head_hokuyo_sensor2">
      ...
        <topicName>/scan2</topicName>
        <frameName>laser2</frameName>
    ...
</gazebo>
```

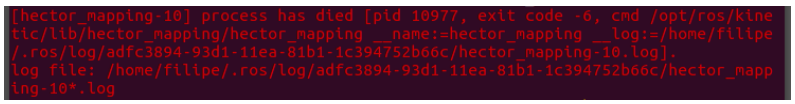
Os erros observados durante a execução do sensor laser podem ser vistos na Figura 21.

Figura 21 – Erros obtidos durante a execução do algorítimo de SLAM com dois sensores publicando em dois tópicos, mas cada um com *frame* de referência diferente.

(a) Erro de tf.



(b) Mensagem de erro do terminal.

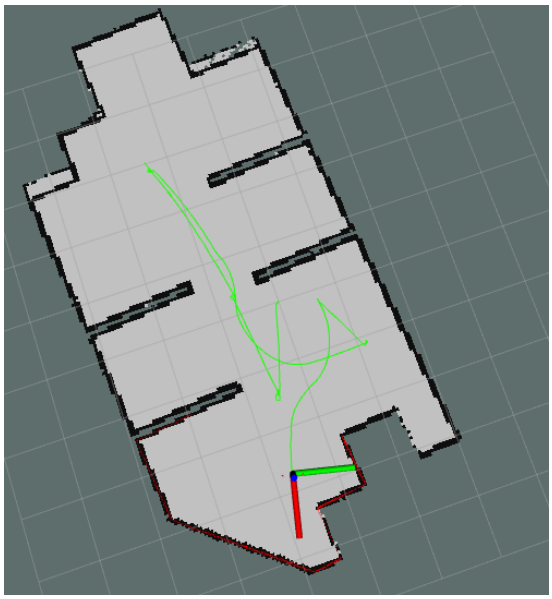


Fonte: Autor

Notando que o método de *SLAM* não conseguiu executar com a configuração em dois tópicos separados, as mensagens foram publicadas em um tópico, mas cada uma foi relacionada a um *frame* diferente dentro da simulação. A simulação foi executada para o mapeamento do ambiente simples modelado, os resultados encontrado foram um mapeamento similar ao apresentado no começo desta seção, utilizando o método de mapeamento aplicado em Bordon e Terra (2019) com os componentes modelados e aplicado ao ambiente simples, que pode ser visto na Figura 13.

O resultado do mapeamento obtido pode ser visto na Figura 22.

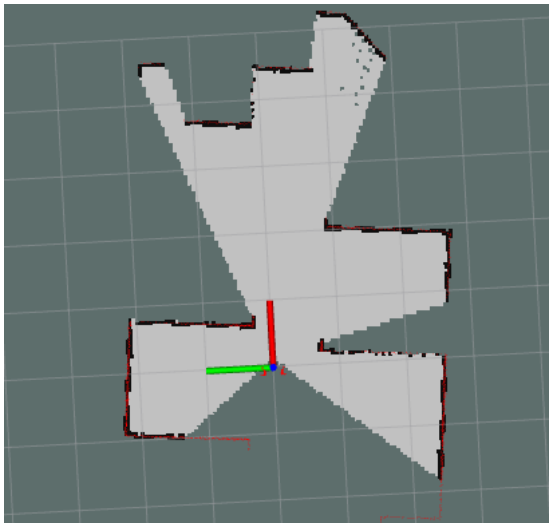
Figura 22 – Mapeamento utilizando a abordagem de um tópico de publicação, mas cada mensagem possui seu *frame* de referência.



Fonte: Autor

Apesar do mapeamento ser executado completamente, e o resultado ser similar ao visto na Figura 13, notou-se que durante a execução do método de *SLAM* o objeto de posição do sensor se mostrou instável, apresentando tremores apesar do robô estar estático, também notou-se que no primeiro ciclo de medida, onde são escritos no mapa os primeiros pontos vistos pelo sensor, apenas o sensor orientado da mesma maneira que o eixo de referência do robô estaria sendo utilizado para a realização do método de *SLAM*, como pode ser visto na Figura 23.

Figura 23 – Posição inicial do robô dirante o mapeamento utilizando a abordagem de um tópico de publicação, mas cada mensagem possui seu *frame* de referência.

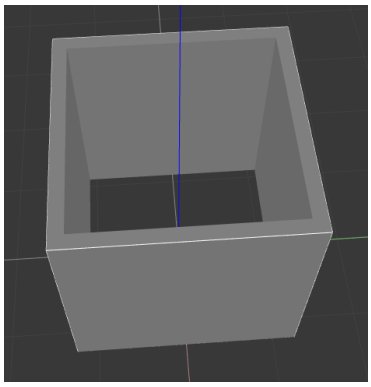


Fonte: Autor

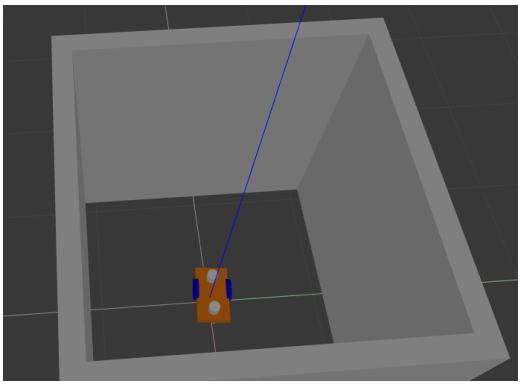
Para verificar se os dois sensores estavam sendo levados em conta durante o mapeamento, foi modelado um cubo de aresta de $2.5m$ onde o robô era colocado no meio a fim de verificar se todo o cubo era mapeado durante o robô estático na simulação. Dessa forma garante que o sensor em relação ao eixo de referência e o sensor defasado de 180° estavam sendo utilizados durante o mapeamento. O modelo pode ser visto na Figura 24 e o mapeamento na Figura 25.

Figura 24 – Modelo do cubo de aresta $2.5m$.

(a) Modelo do cubo.



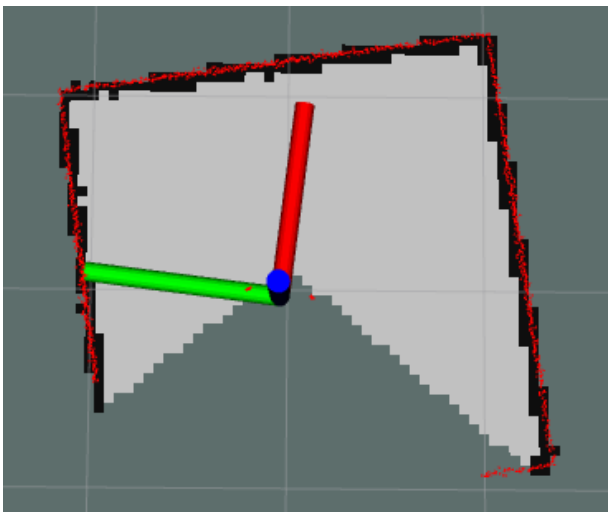
(b) Cubo com robô dentro.



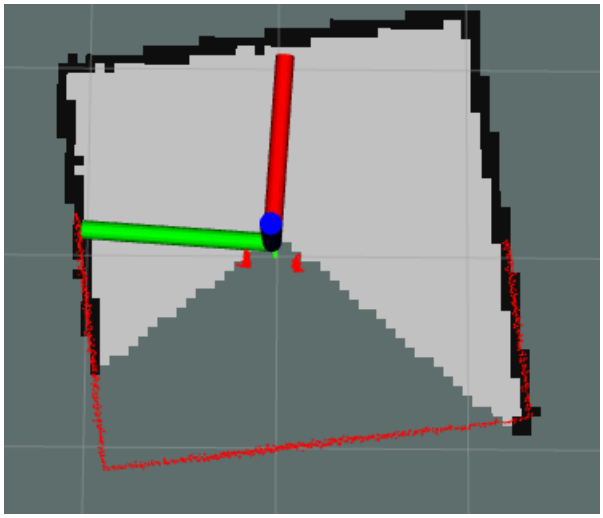
Fonte: Autor

Figura 25 – Mapeamento do cubo com os contornos dos sensores.

(a) Orientação de referencia.



(b) Defasamento de 180$circ$.



Fonte: Autor

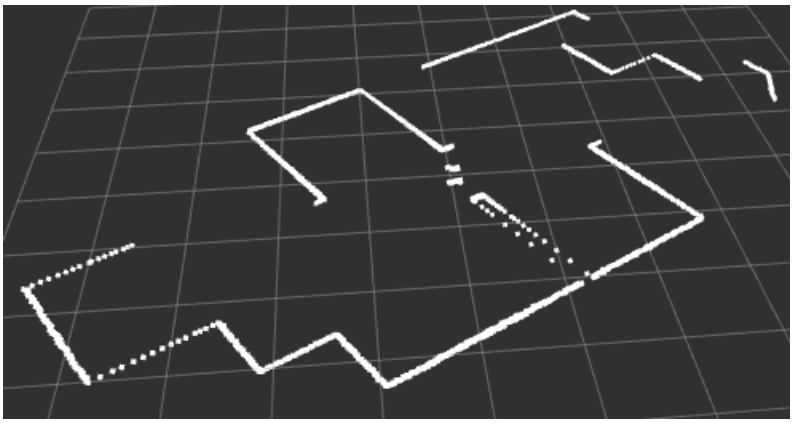
Mostrando desta forma que apenas o sensor de orientação voltada para a referência estava sendo utilizado durante o método de mapeamento, mesmo que na configuração os dois estivessem publicando suas mensagens no mesmo tópico.

4.2.2 ABORDAGEM DE FUSÃO DOS SENSORES *LASER*

Olhando a Figura 20 surge a ideia então de criar um terceiro sensor virtual, no qual, seria a fusão dos dois sensores, para a realização da fusão dos dois sensores o pacote para o *ROS ira_laser_tools* (COLOMBO,) foi utilizado. Este pacote contém duas funcionalidades, uma delas é o *laser scan multi merger* na qual executa a fusão de dois ou mais tópicos de mensagens de sensor *laser*, cada um com sua respectiva transformação homogênea em relação ao *frame* do *laser* digital, e também a função de transformar uma nuvem de pontos obtida de algum sensor que retorna esses valores em dados de um sensor *laser* de linha, criando assim um sensor *laser* virtual.

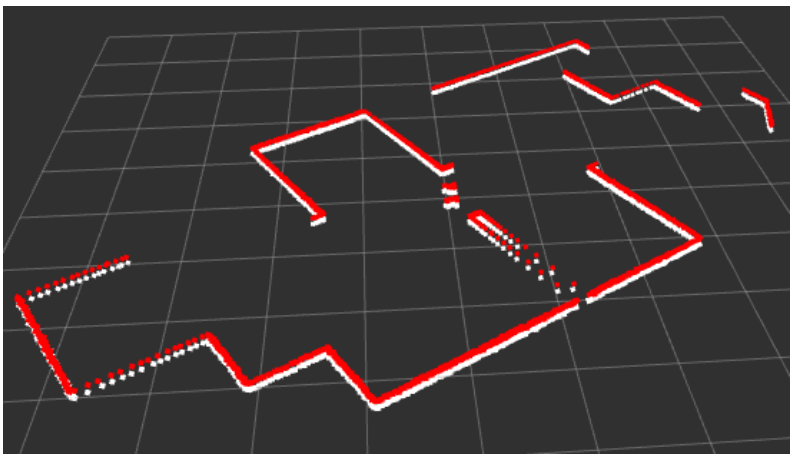
Para este trabalho e esta seção em questão foi utilizado o *merger* a fim de transformar os dois *lasers*, cada um com seu tópico e *frame* em apenas um tópico e um único *frame*. Com as devidas configurações do pacote do *ira_laser_tools* o resultado do sensor *laser* virtual visto dentro do *RViz* pode ser visto nas Figuras. 26 e 27.

Figura 26 – Sensor laser virtual resultante do *merger*.



Fonte: Autor

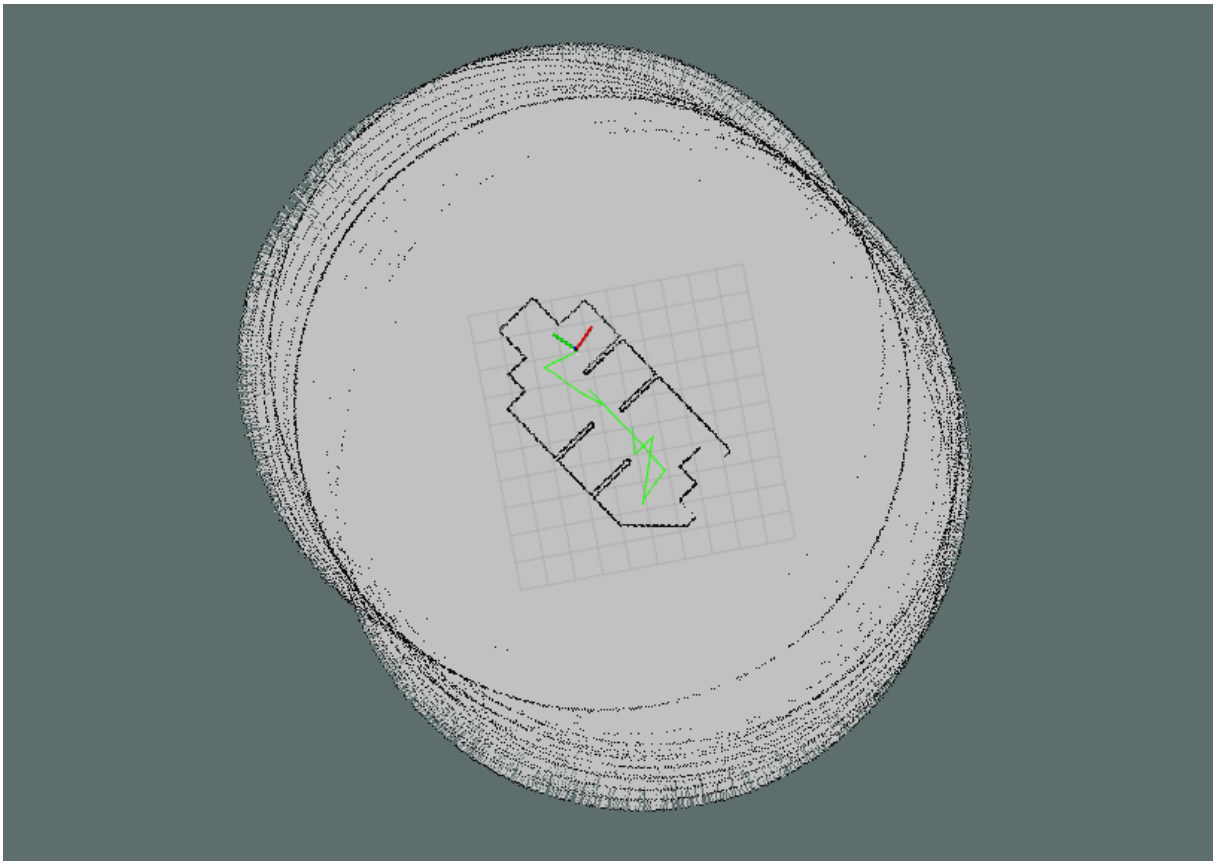
Figura 27 – Sensor resultante comparado com os dois sensores.



Fonte: Autor

Como o comportamento desse *laser* virtual é similar ao comportamento de um laser *comum* para o pacote de *SLAM*, foi então passado seu tópico para a inscrição de mensagens *laser* e feita a transformação com seu *frame* dessa forma o mapeamento foi realizado. Mas um problema foi encontrado, como pode ser visto na Figura 28.

Figura 28 – Mapeamento realizado com o sensor virtual.



Fonte: Autor

Notou-se que durante a execução do método de *SLAM* aplicado ao sensor virtual resultante da fusão dos outros dois sensores, alguns pontos eram considerados como valores na distância máxima do sensor com o valor de $10m$ (Tabela 1), não foi encontrado o porquê que essas mensagens apresentam esse erro, mas descobriu-se que alterando o valor de *angle increment* dentro do arquivo de configuração do pacote *ira_laser_tools* o número de erros diminuiu drasticamente, a comparação pode ser vista em uma medição para o valor de *angle increment* padrão, para o valor de *angle increment* encontrado realizando o método de tentativa e erro.

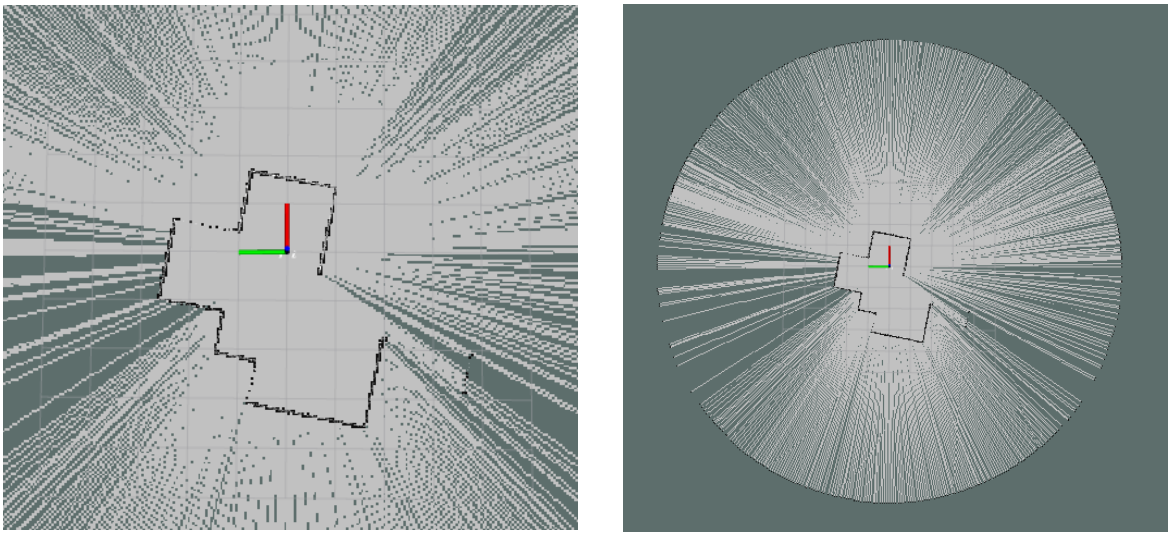
```
<param name="angle_increment" value="0.0029"/> <!--standart-->

<param name="angle_increment" value="0.05"/> <!--encontrado-->
```

As imagens de uma execução do método de mapeamento com o *angle increment* padrão e a quantidade de erros encontrada pode ser vista na Figura 29 e uma medição completa do ambiente simples com o *angle increment* com o valor de 0.05 na Figura 30

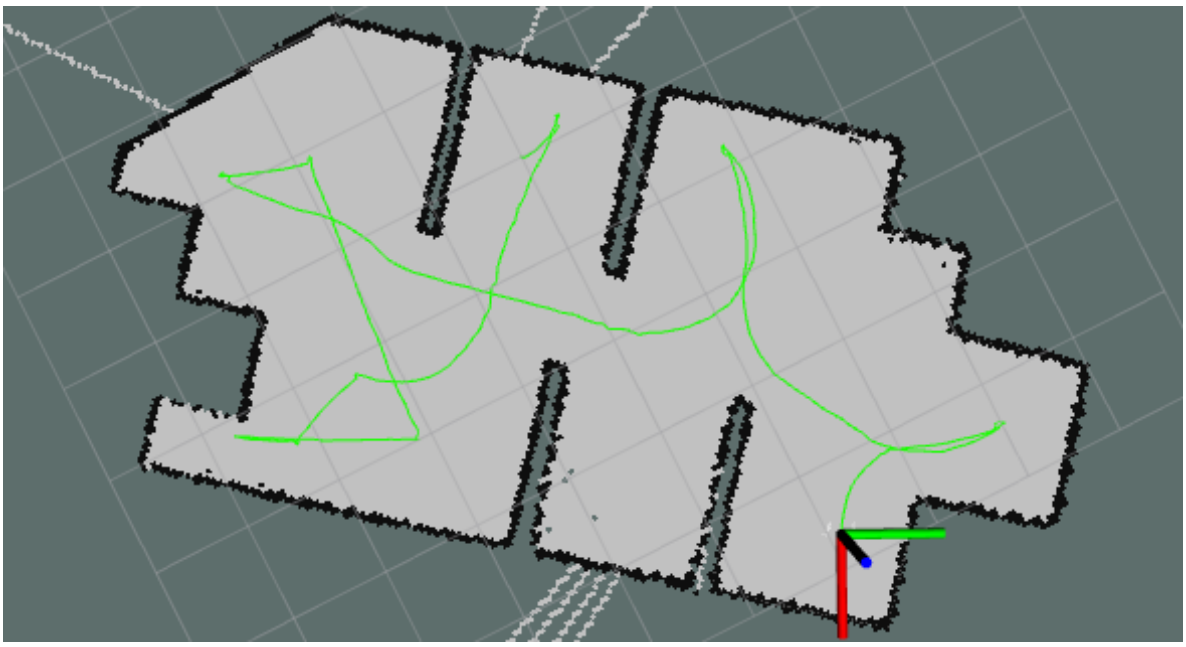
Figura 29 – Execução de apenas um mapeamento para o sensor virtual com *angle increment* padrão.

(a) *Zoom in.* (b) *Zoom out.*



Fonte: Autor

Figura 30 – Execução de apenas um mapeamento para o sensor virtual com *angle increment* de 0.05.

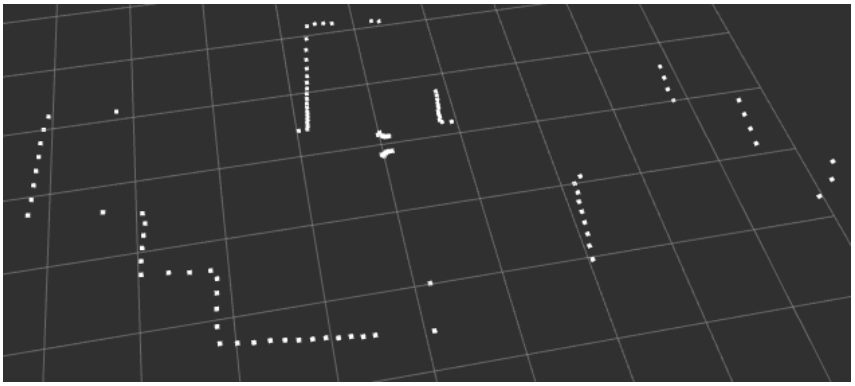


Fonte: Autor

É possível ver que a qualidade da medição apresentada na Figura 30 é muito superior que a apresentada na Figura 28 quando se é levado em conta os pontos fora da área delimitada, na Figura 30 é possível contar apenas 7 mensagens fora da área delimitada e na Figura 29 é quase impossível contar manualmente todas as mensagens de erro obtidas em apenas uma medição.

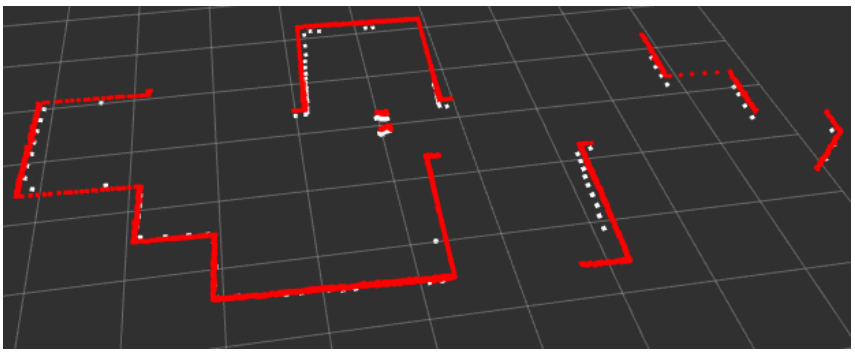
Mas então, por que não aumentar ainda mais o parâmetro de *angle increment*, pois este não se comporta linearmente, pois um teste foi realizado com *angle increment* de valor 0.06 e não houve redução dos pontos fora da delimitação. E conforme há um aumento do parâmetro de *angle increment* a instabilidade do método de *SLAM* em áreas que não foram mapeadas ainda é aumentada, pois com a redução de pontos, menos o mapa é preenchido durante a execução *SLAM* e com um preenchimento escasso os movimentos do robô não são acompanhados pelo sistema, pois não há um "encaixe" de nuvens de pontos (mapas) coerente (visitar seção 2.7, onde há a explicação do funcionamento do método de *SLAM* descrito), gerando assim, instabilidade na criação do mapa nos primeiros estágios. Tal fato pode ser visto nas figs. 31 e 32.

Figura 31 – Sensor laser virtual resultante do *merger* com *angle increment* igual à 0.05.



Fonte: Autor

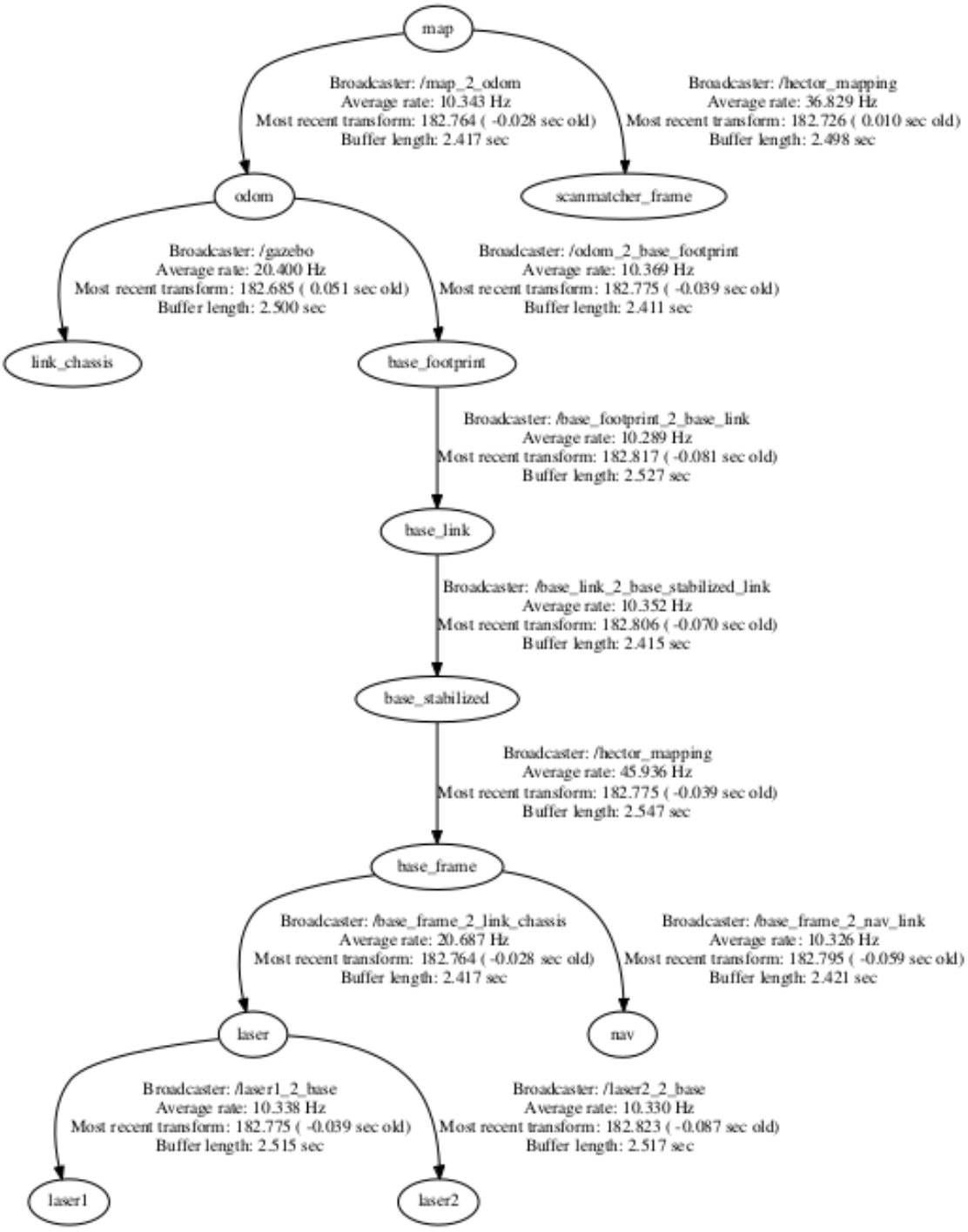
Figura 32 – Sensor resultante com *angle increment* igual à 0.05 comparado com os dois sensores.



Fonte: Autor

As transformações que são feitas e os *frames* envolvidos durante o processo de mapeamento com o *ira_laser_tools* pode ser visto na Figura 33 esta árvore dita todo o caminho das transformações homogêneas que são carregadas até a formação do mapa no topo da árvore, todas as transformações depois são referenciadas a este mapa direta ou indiretamente. O tópico responsável pela criação do mapa é o *scanmatcher*,

Figura 33 – Arvore de transformações executadas no processo de mapeamento com o sensor virtual.



Fonte: Autor

Apesar do simulador fornecer um tópico com mensagens de odometria, este é o tópico/*frame* na qual a odometria probabilística é gerada, é possível através do comando do terminal abaixo identificar o tópico responsável pela mensagem de odometria e ecoá-la no próprio terminal com o prefixo "*echo*" vinculado ao tópico em questão.

Da mesma maneira que pode ser visto na Figura 3.3 é possível ver uma diferença, pois o *frame* do *laser* digital se conecta aos *lasers* "reais", também é possível ver o frame responsável pelo chassi do robô modelado, denominado *link_chassis*. Esta referência é feita de maneira automática, por padrão do simulador *gazebo*, o mesmo poderia ser relacionado ao *frame base_link* sem alteração do resultado, é importante que este esteja relacionado à algum *frame* do mapeamento caso seja requerido que o modelo do robô apareça no ligar dos eixos do robô, ou até mesmo juntamente com os eixos. Para garantir que o *frame* de odometria utilizado não seja o gerado pelo simulador, é necessário utilizar a configuração no arquivo de inicialização do pacote de posição e trajetória dentro do pacote geral de *SLAM* esteja com as seguintes configurações de *frame*:

```
<launch>
  ...
  <arg name="base_frame" default="base_frame"/>
  <arg name="odom_frame" default="base_frame"/>
  ...
</launch>
```

Caso uma odometria que não seja a probabilística seja usada, é possível adicionar o *frame* padrão de odometria (é o nome padrão do *frame* caso algum odômetro esteja conectado ao ROS ou o *frame* que o próprio simulador gera) é só trocar o o nome do *odom_frame* para *odom* ao invés de *base_frame* (linhas de código apresentadas acima).

4.2.3 CONDIÇÕES DE CONTORNO DO *HECTOR_SLAM*

Como é sabido o método de *SLAM* proposto pelo *team Hector* pode trabalhar tanto com a odometria externa, quanto com a odometria interna (KOHLBRECHER et al., 2011).

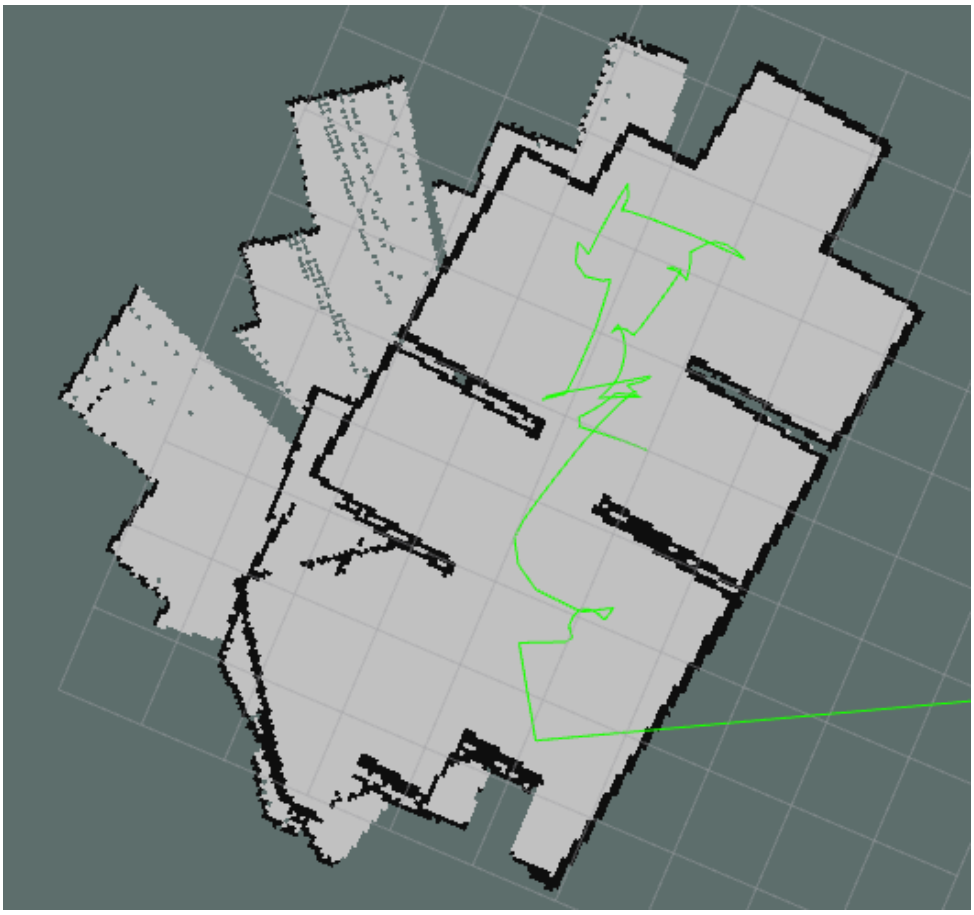
A odometria interna é uma componente obtida através de cálculos probabilísticos utilizando sistemas que trabalham com computação visual e eliminação dos erros das montagens das imagens a fim de formar um mapa conciso e determinar a posição do sensor e/ou corpo móvel (todo o detalhamento do processo e equações estão na Seção 2.7). É comprovado pelo próprio *team Hector* que a precisão do mapeamento é maior quando se utiliza a odometria probabilística (KOHLBRECHER et al., 2011). Dessa forma além de garantir uma precisão no mapeamento, há também uma redução nos custos de algum projeto de corpo móvel autônomo no futuro.

Mas essa odometria probabilística provoca uma condição de contorno dada pela velocidade com qual se movimenta o corpo móvel. Essa velocidade pode fazer com que o *frame* responsável por manter a posição do robô (*base_footprint*) se perca e escreva por cima do mapa existente criando inconsistências no mapa final.

É possível simular uma movimentação brusca no robô utilizando o recurso do *ROS* responsável por movimentar os *turtle bots* (nome dado aos modelos de movimentação que são executados dentro do *ROS*) o pacote em questão é o *teleop_twist_keyboard* onde é possível aumentar e diminuir tanto velocidade angular e linear, como também é possível executar movimentos angulares sentido horário e anti-horário, bem como também velocidade linear sentido positivamente orientado para o $eixo - X$ do robô e negativamente orientado para o $eixo - X$, bem como também a combinação dos dois movimentos.

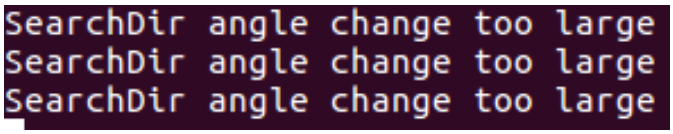
Um erro provocado pelo movimento brusco do robô pode ser percebido através da visualização do mapeamento, quando há a sobreposição incoerente de linhas de obstáculos dentro do mapa e também através do terminal que exporta mensagens de movimentos angulares muito extensos. Essas comprovações podem ser vistas nas Figuras. 34 e 35.

Figura 34 – Mapeamento gerado durate uma movimentação brusca do robô.



Fonte: Autor

Figura 35 – Mensagem de erro no terminal executando o *Hector_slam* durante uma movimentação brusca do robô.



Fonte: Autor

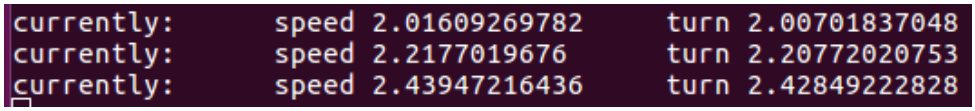
Importante ressaltar na observação da Figura 34 o fato que o objeto responsável por informar a posição do robô (os eixos coloridos) se perde no mapa, e o mesmo não pôde ser visto nem com o *zoom out* máximo da ferramenta *RViz*. Este fato é comum quando o método de *SLAM* se perde durante a medição, seja por um movimento brusco, ou por inconsistência de mensagens nos tópicos, como também pode ser visto nas figura 18.

Para a determinação destas velocidades foi-se aumentando as velocidade em 10% o valor da velocidade anterior (taxa de incremento mínimo do pacote de *teleop_twist_keyboard*) este incremento possui um comportamento similar ao de um juros composto. Para cada variedade de acréscimo:

- 10% de acréscimo em ambas velocidades linear e angular.
- 10% de acréscimo na velocidades linear.
- 10% de acréscimo na velocidade angular.

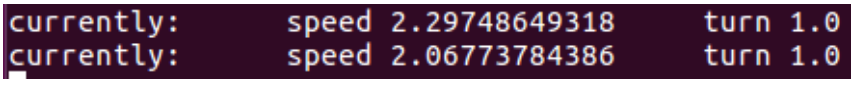
Para cada variedade três testes foram executados e o com menor valor de velocidade foi usado como o principal para este trabalho. Estes valores podem ser vistos nas figs. 36, 37 e 38 respectivamente igual a lista mencionada acima.

Figura 36 – Velocidade mínima angular e linear em um movimento combinado entra as duas.



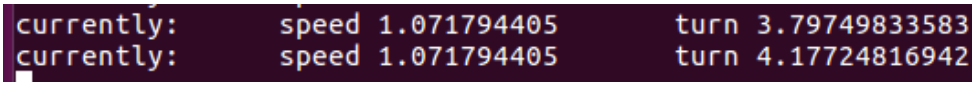
Fonte: Autor

Figura 37 – Velocidade mínima em um movimento linear.



Fonte: Autor

Figura 38 – Velocidade mínima em um movimento angular.



Fonte: Autor

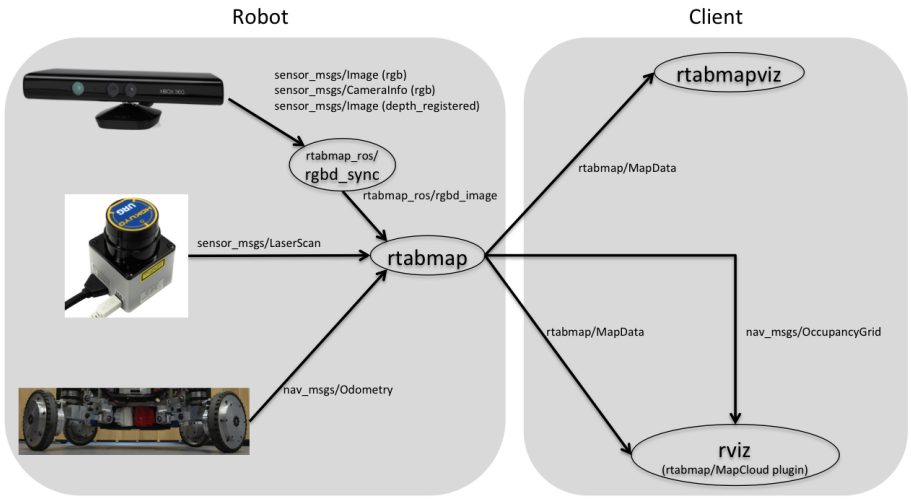
4.3 MAPEAMENTO UTILIZANDO RTAB-MAP

O método de mapeamento proposto pela equipe *Hector* apresentado no pacote *ROS: Hector_slam* utiliza apenas como referência um tópico de mensagem de sensores *laser* de linha e talvez um tópico de odometria (podendo ser real ou probabilística (virtual)). Conhecendo as condições de contorno que envolvem o método *Hector* de mapeamento, uma solução diferente foi procurada para se poder utilizar um sensor de visão tridimensional no processo.

Durante o trabalho executado por Goulart e Terra (2020), foi notado que o mapeamento tridimensional provou-se impraticável, pois o volume de dados gerado durante o processo é extenso, sendo estabelecido que mapeamento de ambientes tridimensionalmente se mostrou hábil em casos em que o conhecimento da variação de altura é necessário, para aplicações simples como robótica aplicada à ambientes fechados e mapeamento e localização simultâneos utilizados para conhecimento do ambiente o mapeamento bidimensional é suficiente, (GOULART; TERRA, 2020). Para este mapeamento utilizando um sensor como o *KINECT v2* é necessário que o mesmo seja transformado em um sensor laser de linha a partir das informações coletadas através da interpretação da nuvem de pontos gerada pela câmera de profundidade, e a partir desta conversão é possível aplicar a algum método de mapeamento bidimensional, como por exemplo o *gmapping* utilizado em Goulart e Terra (2020), ou *Hector_slam* como apresentado nas seções anteriores deste capítulo.

O *KINECT v2* é um sensor com diversos parâmetros de medições, e utilizar a nuvem de pontos gerada por sua visão tridimensional e transformá-la em um sensor *laser* de linha, tendo à disposição sensores *lasers* de linha reais, pode ser considerada uma subutilização do aparelho e desperdício dos dados levantados pelo sensor. Somando ao fato que a fusão de diversos sensores laser de linha provocam algumas características de instabilidade (Seção 4.2.2). A princípio foi-se pensado em transformar o *KINECT v2* em um odômetro, para incrementar e suprir as condições de contorno vistas na seção 4.2.3, durante a busca foi encontrado o pacote *RTAB-Map* que utiliza sensores *lasers* de linha fundidos e *KINECT v2* em conjunto para a realização do mapeamento e utilização simultâneos (configuração pode ser vista na Figura 39), utiliza-se do odômetro gerado pelo modelo do robô.

Figura 39 – Setup de configuração para modelo utilizando sensor *laser* de linha, *kinect v2* e a odometria presente no robô

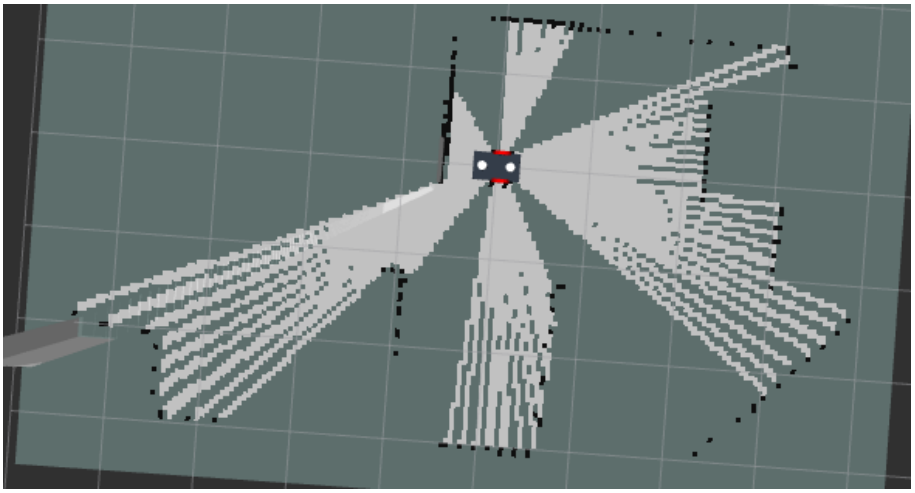


Fonte: Labbe (2020)

A aplicação do método de mapeamento apresentado pelo pacote *RTAB-Map* leva em conta a utilização de diferentes tópicos de medidas de diferentes tipos de sensores. Durante o desenvolvimento deste projeto, a utilização de dois sensores *laser* de linha sempre esteve incluída dentro das abordagens apresentadas nas seções anteriores deste capítulo.

Desta forma para o tópico do sensor laser utilizou-se o resultado obtido com o pacote *ira_laser_tools* na Seção 4.2.2. Utilizou-se inicialmente a configuração de *angle increment* de 0.05. Tal valor apresenta grandes espaços entre os pontos de medição do *laser* virtual (Figura 31), mesma característica que pode ser vista no primeiro ciclo de mapeamento do método (Figura 40).

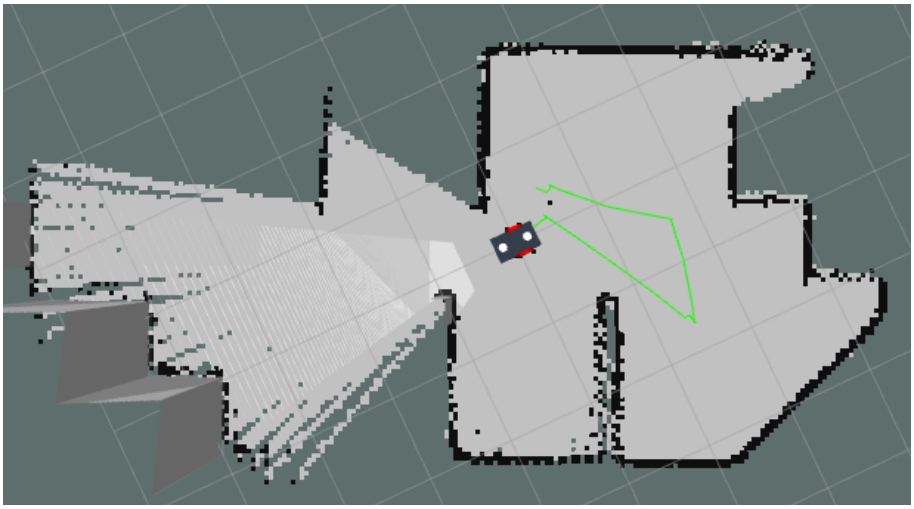
Figura 40 – Ciclo inicial de mapeamento utilizando o *RTAB-Map*



Fonte: Autor

Durante a movimentação do robô pelo ambiente simples, pode-se perceber que não havia acontecido nenhuma ocorrência de pontos sendo escritos na distância máxima do sensor. Esta movimentação persistiu por algum tempo e o resultado do mapeamento encontrado pode ser visto na Figura 41.

Figura 41 – Mapeamento realizado após algumas movimentações do robô utilizando o *RTAB-Map*

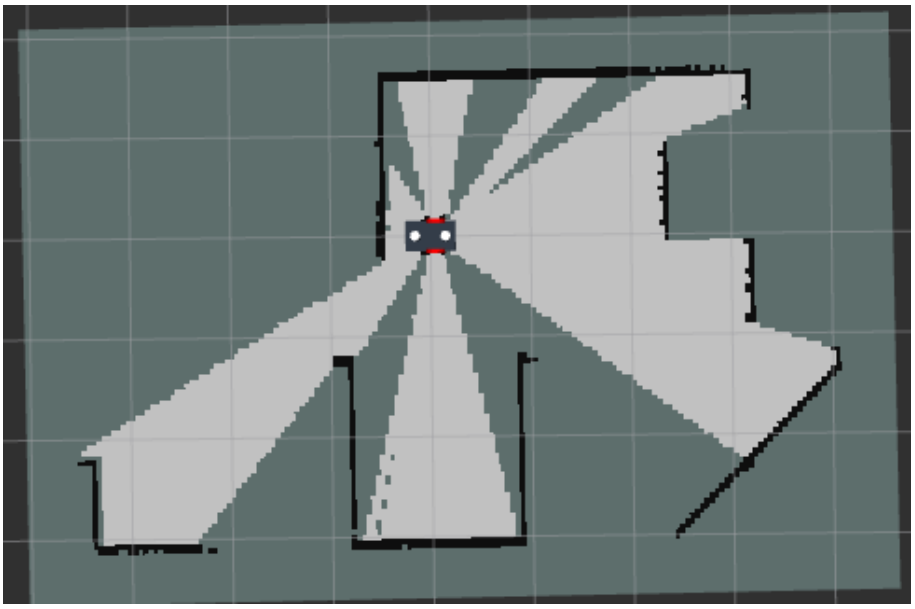


Fonte: Autor

Percebendo que a ocorrência de erros não era vista, partiu-se então para a tentativa de utilizar valores menores de *angle increment* para que o sensor possuísse uma característica linear, como se foi observado inicialmente na Figura 26. A alteração do parâmetro foi sendo feita a fim de observar até que ponto poderia se obter um mapeamento rápido, com preenchimentos consistentes sem que o volume de dados gerado pelo sensor sobrecarregasse a execução do método de mapeamento.

Encontrou-se um valor próximo ao utilizado como padrão de 0.0029 que foi o valor de 0.0025, visando corrigir as características vistas nas Figuras 40 e 41. Nota-se nessas imagens que o espaçamento entre as escritas das delimitações do ambiente durante o processo de mapeamento, e conforme a movimentação do robô é realizada para que a montagem do mapa possa prosseguir. Mesmo com alguns movimentos, percebe-se na Figura 41 que a qualidade do mapa não apresenta uma progressão aceitável, sendo necessário que diversos movimentos sejam repetidos em áreas mapeadas para garantir a cobertura de todo o ambiente, juntamente com tal fato é necessário que os movimentos sejam cadenciados para que seja possível que os pontos sejam escritos no mapa, limitando assim a cadência de movimento de acordo com a capacidade de processamento da máquina realizando o processo de mapeamento. Com o novo parâmetro de incremento angular o mapeamento para um ciclo de medição pode ser visto na Figura 42.

Figura 42 – Ciclo inicial de mapeamento utilizando o *RTAB-Map* com 0.0025 de incremento angular.

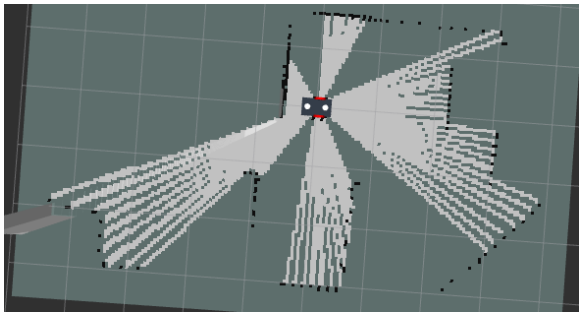


Fonte: Autor

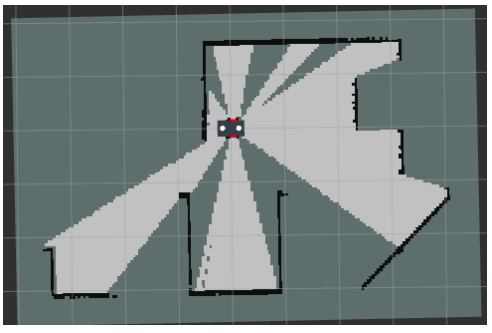
Comparando lado a lado as duas medições é possível perceber a diferença expressiva entre os dois parâmetros na Figura 43.

Figura 43 – Comparação entre os dois primeiros ciclos de mapeamento com parâmetros de *angle increment* diferentes.

(a) Incremento angular de 0.05.



(b) Incremento angular de 0.0025.



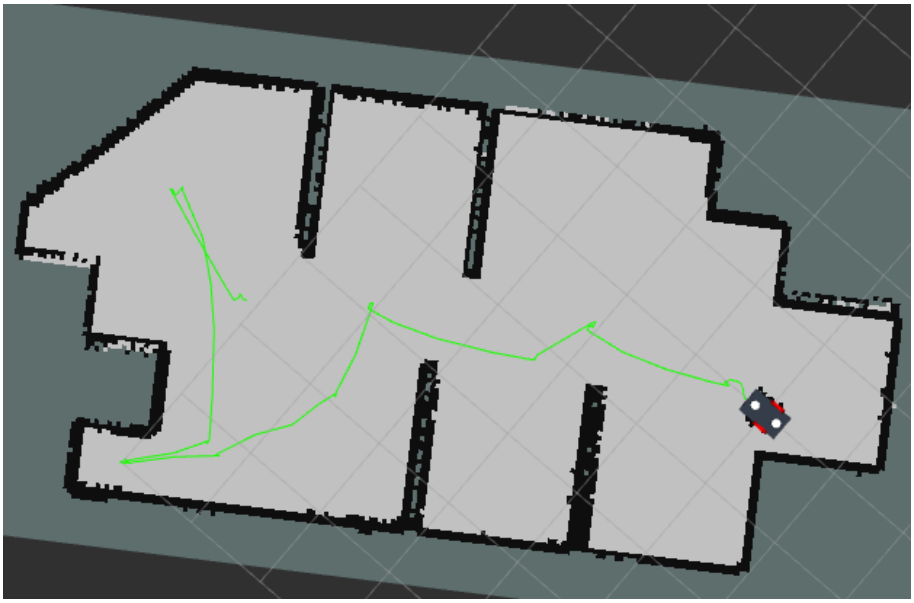
Fonte: Autor

O mapeamento completo do ambiente simples visto na Figura 44 e é possível comparar com a qualidade do mapeamento realizado com um sensor laser em operação total do pacote *Hector_slam* visto na Figura 13.

O preenchimento do mapa se dá consistentemente conforme o robô se movimenta controlado pelo usuário pelo ambiente simples modelado, tal fato é devido à quantidade de

espaço coberto pelo sensor laser virtual, e devido ao parâmetro que gerante que os pontos estejam conectados, e não separados.

Figura 44 – Mapeamento realizado do ambiente simples utilizando o *RTAB-Map*.



Fonte: Autor

Diferentemente do método apresentado pelo *team Hector* que utiliza diversos nós de processamento e diversos tópicos de comunicação verticais como pode ser visto na Figura 33, o método apresentado pelo *RTAB-Map* utiliza apenas um nó de processamento, e utiliza uma comunicação entre os tópicos de maneira horizontal, dependendo assim de uma capacidade de processamento mais potente.

4.4 CONTRIBUIÇÃO *OPEN SOURCE*

Com o intuito de facilitar os avanços desse projeto para futuros pesquisadores, todos os pacotes, programas e ambientes utilizados ao longo desse projeto foram disponibilizados de maneira pública através da plataforma do github. O repositório possui licença MIT de utilização e distribuição do ambientes desenvolvidos, desde que seja devidamente creditado. (BORDON; GOULART, 2020)

Todas as aplicações e pacotes utilizados foram conteinerizados afim de facilitar a integração em robôs reais. Entretanto o trabalho em questão não lida com problemas presentes em sensores em ambiente não simulado, tais como ruídos de medição ou *drift* de odometria.

5 CONCLUSÃO

Esse projeto se propôs a realizar o estudo e comparação de técnicas de mapeamento e localização utilizando sensores laser de linha e um sensor de de profundidade tridimensional visando melhorar a redundância e confiabilidade dos dados finais, realizando comparações entre se utilizar diversos arranjos de sensores explorando seus pontos fortes e fracos.

Ao longo do projeto foram desenvolvidas medições se utilizando de apenas um sensor de linha, utilizando o pacote do *Hector SLAM* para inferir os dados de mapeamento, localização e odometria do robô. Nessa sessão tivemos um bom resultado, explorando também as limitações do pacote como a não utilização de uma odometria mais confiável como a presente no próprio robô.

Em seguida, a fim de aumentar o alcance da visão e mapeamento do robô, foi instalado um segundo sensor de linha. Nessa etapa o Hector se demonstrou bastante limitado, uma vez que foi necessário reduzir muito a resolução e velocidade do robô para que fosse possível realizar um mapeamento se aproveitando da fusão entre os sensores.

Por fim, afim de solucionar o problema de odometria e também da baixa resolução do mapeamento desempenhado pelo sistema de mapeamento do Hector *SLAM*, utilizou-se o algorítimo *RTBA-Map* acrescentando um sensor *Kinect V2* e também a odometria presente no próprio modelo de simulação. A aplicação desse pacote resultou em um mapeamento mais preciso, se aproveitando dos melhores recursos presentes em cada um dos sensores utilizados. Dos sensores de linha foi extraída a máxima resolução presente no sensor, diminuindo os pontos cegos do robô. E o sensor *Kinect V2* teve uma função principal de sincronizar os dados de odometria com os dados vindos do sensor de linha, melhorando a qualidade tanto da odometria quanto do mapeamento, este procedimento utiliza diversos sensores iguais e diferentes, cumprindo com a premissa de redundância de sensores para medição de distância realizando um método de mapeamento e localização simultâneos em conjunto.

Ao fim do projeto foi possível estudar diversos métodos e sistemas utilizados em mapeamento e localização robótica, explorando os principais aspectos positivos presentes em cada um dos sensores utilizados e também buscando cobrir os pontos falhos entre eles. Todo o código, pacotes e ambiente utilizados no projeto foram compilados no repositório do *github* (BORDON; GOULART, 2020) afim de facilitar desenvolvimentos futuros.

REFERÊNCIAS

3D TRANSFORMATIONS. Open source Robotics Foundation. 2017. Disponível em: <http://wiki.ros.org/tf>. Acesso em: 26 fev. 2019.

ABOUT ROS. **About ROS**. Open Source Robotics Foundation. 2017. Disponível em: <http://www.ros.org/about-ros/>. Acesso em: 09 ago.2018.

AGRAWAL, P. et al. Pce-slam: A real-time simultaneous localization and mapping using lidar data. In: **2017 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.: s.n.], 2017. p. 1752–1757.

ANDO, S. Intelligent three-dimensional world sensor with eyes and ears. In: **Handbook of Sensors and Actuators**. [S.l.]: Elsevier, 1996. v. 3, p. 117–152.

ANDO, S. Intelligent sensor systems: integrating advanced automatism and optimality into sensors. In: IEEE. **Proceedings of International Solid State Sensors and Actuators Conference (Transducers' 97)**. [S.l.], 1997. v. 1, p. 291–294.

ASHLEY, J. **QUICK REFERENCE: KINECT 1 VS KINECT 2**. The Imaginative Universal. 2014. Disponível em: <https://www.stereolabs.com/zed/>. Acesso em: 30 abr. 2019.

BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (slam): part i. **IEEE Robotics Automation Magazine**, v. 13, n. 3, p. 108–117, Sept 2006. ISSN 1070-9932.

BORDON, F.; GOULART, D. **SLAM with Lidar and Kinect sensors**. 2020. Disponível em: <https://github.com/Filipe-Douglas-Slam/slam_lidar_kinect>.

BORDON, F. Q.; TERRA, M. H. Mapeamento utilizando sensor a laser de linha e pacote ros para corpos móveis. FAPESP, 2019.

Butkiewicz, T. Low-cost coastal mapping using kinect v2 time-of-flight cameras. In: **2014 Oceans - St. John's**. [S.l.: s.n.], 2014. p. 1–9. ISSN 0197-7385.

COLOMBO, P. **IRA LASER TOOLS**. iralabdisco. Disponível em: <https://github.com/iralabdisco/ira_laser_tools>.

CRAIG, J. J. **Introduction to Robotics**. [S.l.]: Pearson Prentice Hall, 2005. ISBN 0131236296.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. **IEEE robotics & automation magazine**, IEEE, v. 13, n. 2, p. 99–110, 2006.

Fankhauser, P. et al. Kinect v2 for mobile robot navigation: Evaluation and modeling. In: **2015 International Conference on Advanced Robotics (ICAR)**. [S.l.: s.n.], 2015. p. 388–394.

Femmam, S. Actuators: Modeling and analysis. In: _____. **Signals and Control Systems: Application for Home Health Monitoring**. [S.l.: s.n.], 2017. p. 117–159.

FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. **Artificial intelligence review**, Springer, v. 43, n. 1, p. 55–81, 2015.

GARCIA-FIDALGO, E.; ORTIZ, A. Vision-based topological mapping and localization methods: A survey. **Robotics and Autonomous Systems**, v. 64, p. 1 – 20, 2015. ISSN 0921-8890. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0921889014002619>.

GERMAN, S.; WÖGER, W. International system of units (si). **Van Nostrand's Scientific Encyclopedia**, Wiley Online Library, 2005.

GOULART, D. D.; TERRA, M. H. Mapeamento de ambientes indoor com sensor kinect 2 e sistema ros. FAPESP, 2020.

GRISETTI, G. et al. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE transactions on Robotics**, v. 23, n. 1, p. 34, 2007.

HERSHBERGER, D.; GOSSOW, D.; FAUST, J. **Rviz**. BSD, Creative Commons. 2018. Disponível em: <http://wiki.ros.org/rviz>. Acesso em: 08 ago. 2018.

HOKUYO AUTOMATIC CO., LTD. **Optical data transmission device, Photo sensor, Auto counter, and Automatic door**. HOKUYO AUTOMATIC CO., LTD. 2014. Disponível em: <https://www.hokuyo-aut.jp/>. Acesso em: 07 ago. 2018.

HORICHI, K. et al. Spot-based lidar profile estimation algorithm for mobile robots in motion. In: **2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)**. [S.l.: s.n.], 2016. p. 237–242.

KINECT FOR WINDOWS TEAM. **QUICK REFERENCE: KINECT 1 VS KINECT 2**. Microsoft. 2014. Disponível em: <https://blogs.msdn.microsoft.com/kinectforwindows/2014/06/05/pre-order-your-kinect-for-windows-v2-sensor-starting-today/>. Acesso em: 15 mar. 2019.

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. **2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)**. [S.l.], 2004. v. 3, p. 2149–2154.

KOHLBRECHE, S.; MEYER, J. **Hector SLAM - ROS**. 2014. Disponível em: <http://wiki.ros.org/hector_slam>. Acesso em: 25 fev. 2019.

KOHLBRECHER, S. et al. A flexible and scalable slam system with full 3d motion estimation. In: IEEE. **Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)**. [S.l.], 2011.

KöSEOğLU, M.; ÇELIK, O. M.; PEKTAş, O. Design of an autonomous mobile robot based on ros. In: **2017 International Artificial Intelligence and Data Processing Symposium (IDAP)**. [S.l.: s.n.], 2017. p. 1–5.

LABBE, M. **rtabmap_ros**. Open Source Robotics Foundation. 2020. Disponível em: <http://wiki.ros.org/rtabmap_ross>. Acesso em: 20 mai. 2020.

LABBE, M.; MICHAUD, F. Appearance-based loop closure detection for online large-scale and long-term operation. **IEEE Transactions on Robotics**, IEEE, v. 29, n. 3, p. 734–745, 2013.

LABBÉ, M.; MICHAUD, F. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. **Journal of Field Robotics**, Wiley Online Library, v. 36, n. 2, p. 416–446, 2019.

LAKEMEYER, G.; NEBEL, B. **Exploring artificial intelligence in the new millennium**. [S.l.]: Morgan Kaufmann, 2003.

LAVRENOV, R.; ZAKIEV, A. Tool for 3d gazebo map construction from arbitrary images and laser scans. In: IEEE. **2017 10th International Conference on Developments in eSystems Engineering (DeSE)**. [S.l.], 2017. p. 256–261.

Meng Dong et al. Uav flight controlling based on kinect for windows v2. In: **2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)**. [S.l.: s.n.], 2016. p. 735–739.

OCANDO, M. G. et al. Autonomous 2d slam and 3d mapping of an environment using a single 2d lidar and ros. In: **2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)**. [S.l.: s.n.], 2017. p. 1–6.

OLSON, E. B. Real-time correlative scan matching. In: IEEE. **2009 IEEE International Conference on Robotics and Automation**. [S.l.], 2009. p. 4387–4393.

OPEN SOURCE ROBOTICS FOUNDATION. **Tutorial: ROS integration overview**. Apache 2.00. 2014. Disponível em: <http://gazebosim.org/tutorials?tut=ros_overview>. Acesso em: 05 mai.2020.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software**. [S.l.], 2009. v. 3, n. 3.2, p. 5.

Roberts, D. et al. Constructing a gazebo: Supporting teamwork in a tightly coupled, distributed task in virtual reality. **Presence**, v. 12, n. 6, p. 644–657, 2003.

ROS COMPONENTS. **Hokuyo UST-10LX**. HOKUYO AUTOMATIC CO., LTD. 2016. Disponível em: <https://www.roscomponents.com/en/lidar-laser-scanner/85-ust-10lx.html>. Acesso em: 01 ago. 2018.

ROS NODE. **ROS Node**. Open Source Robotics Foundation. 2018. Disponível em: <http://wiki.ros.org/Nodes>. Acesso em: 25 fev. 2019.

ROS TOPIC. **ROS Topic**. Open Source Robotics Foundation. 2019. Disponível em: <http://wiki.ros.org/Topics>. Acesso em: 25 fev. 2019.

SANTOS, J. M.; PORTUGAL, D.; ROCHA, R. P. An evaluation of 2d slam techniques available in robot operating system. In: IEEE. **2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)**. [S.l.], 2013. p. 1–6.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot modeling and control**. [S.l.]: John Wiley & Sons, 2020.

Steux, B.; Hamzaoui, O. E. tinyslam: A slam algorithm in less than 200 lines c-language program. In: **2010 11th International Conference on Control Automation Robotics Vision**. [S.l.: s.n.], 2010. p. 1975–1979.

Sun, T. et al. Rss-based map construction for indoor localization. In: **2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)**. [S.l.: s.n.], 2018. p. 1–8. ISSN 2471-917X.

WEISSTEIN, E. W. Moment of inertia. Wolfram Research, Inc., 2009.