UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

NICOLAS DALMEDICO

SENSOR DE ESCANEAMENTO OMNIDIRECIONAL E MAPEAMENTO POR JANELA DESLIZANTE

DISSERTAÇÃO

CURITIBA

2018

NICOLAS DALMEDICO

SENSOR DE ESCANEAMENTO OMNIDIRECIONAL E MAPEAMENTO POR JANELA DESLIZANTE

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Mestre em Ciências" – Área de Concentração: Engenharia de Automação e Sistemas.

Orientador: André Schneider de Oliveira

Co-orientadora: Lucia Valeria Ramos De Arruda

CURITIBA

2018

Dados Internacionais de Catalogação na Publicação

D148s 2018

Dalmedico, Nicolas
Sensor de escaneamento omnidirecional e mapeamento por janela deslizante / Nicolas Dalmedico.-- 2018.
89 f.: il.; 30 cm.

Disponível também via World Wide Web.
Texto em português com resumo em inglês.
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Curitiba. Área de Concentração: Engenharia de Automação e Sistemas, 2018.
Bibliografia: f. 85-89.

1. Robôs móveis. 2. Sensoriamento remoto. 3. Robôs autônomos. 4. Detectores - Desenvolvimento. 5. Mapeamento digital. 6. Imagem tridimensional. 7. Processamento de imagens. 8. Visão de robô. 9. Serviços baseados em localização. 10. Sistemas de computação em grade. 11. Métodos de simulação. 12. Engenharia elétrica - Dissertações. I. Oliveira, André Schneider de, orient. II. Arruda, Lúcia Valéria Ramos de, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. IV. Título.

CDD: Ed. 23 -- 621.3

Biblioteca Central do Câmpus Curitiba – UTFPR
Bibliotecária: Luiza Aquemi Matsumoto CRB-9/794

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

TERMO DE APROVAÇÃO DE DISSERTAÇÃO Nº 807

A Dissertação de Mestrado intitulada **"Sensor de Escaneamento Omnidirecional e Mapeamento por Janela Deslizante"** defendida em sessão pública pelo(a) candidato(a) **Nicolas Dalmedico**, no dia 03 de outubro de 2018, foi julgada para a obtenção do título de Mestre em Ciências, área de concentração Engenharia de Automação e Sistemas, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial.

BANCA EXAMINADORA:

Prof(a). Dr(a). André Schneider de Oliveira - Presidente (UTFPR)
Prof(a). Dr(a). João Alberto Fabro - (UTFPR)
Prof(a). Dr(a). Eduardo Todt - (UFPR)

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 03 de outubro de 2018.

AGRADECIMENTOS

Gostaria de agradecer a todos que colaboraram direta e indiretamente para a realização do projeto:

- Agradeço aos Professores Drs. Flávio Neves Júnior, Lucia Valéria Ramos de Arruda, André Schneider de Oliveira e Daniel Rodrigues Pipa pela orientação geral, disponibilização do ambiente de trabalho e de todos os materiais e equipamentos necessários para a realização do projeto.
- Agradeço à equipe do laboratório LASCA pelo extensivo apoio no projeto e pelo auxílio nas dúvidas durante o desenvolvimento.
- Agradeço a minha família e amigos, por acreditarem em mim e me apoiarem durante a atividade acadêmica.
- Agradeço o apoio financeiro da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis ANP, da Financiadora de Estudos e Projetos FINEP, do Ministério da Ciência, Tecnologia, Inovações e Comunicações MCTIC por meio do Programa de Recursos Humanos da ANP para o Setor Petróleo e Gás PRH-ANP, da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior CAPES e do Programa de Formação de Recursos Humanos da PETROBRAS PRH10-UTFPR.

RESUMO

DALMEDICO, Nicolas. SENSOR DE ESCANEAMENTO OMNIDIRECIONAL E MAPEAMENTO POR JANELA DESLIZANTE. 89 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

Sensores LIDAR são comumente utilizados para realizar mapeamento 2D em robótica. Alguns projetos dão um passo a frente e giram o sensor em frente ao robô para formar uma nuvem de pontos com a qual é possível aplicar técnicas de mapeamento 3D. Este trabalho tenta ir além e criar um método de mapeamento especializado em que sensores LIDAR são rotacionados, escaneando o ambiente com o auxílio de um motor e tendo como resultado uma superfície que represente com acurácia as imediações do robô móvel. Uma superfície base vertical em frente é modelada para representar os pontos lidos (mapa local) e os dados representados nesta superfície são utilizados para montar um mapa de elevação capaz de representar chão (mapa global) e teto de um ambiente. O mapa resultante é compatível com métodos de SLAM e mapeamento existentes (mapa de ocupação). Vários obstáculos de desenvolvimento foram encontrados na tentativa de se combater os problemas que acompanham o método de escaneamento, como utilização de compensação de movimento do robô para combater a baixa frequência de atualização de dados inerente dos sensores LIDAR. Um protótipo foi desenvolvido para validar o método de mapeamento. Ensaios de navegação utilizando o novo mapa gerado são apresentados no final deste trabalho.

Palavras-chave: LIDAR em rotação, mapeamento 3D, mapeamento não-tradicional, robôs móveis

ABSTRACT

DALMEDICO, Nicolas. OMNIDIRECTIONAL SCANNING SENSOR AND SLIDING BAY WINDOW MAPPING. 89 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

LIDAR sensors are commonly used in robotics to perform 2D mapping. Some projects go beyond that and spin the sensor in front of the robot to build a point cloud with which it's possible to apply 3D mapping techniques. This work tries to improve upon this idea and create a novel mapping strategy by rotating LIDAR for acquisition, scanning the environment and building a surface that represents the mobile robot's neighborhood with accuracy. A base vertical grid in front of the robot is created to represent the acquired points (local map) and this grid is used to build two elevation maps, one representing the floor (global map) and other representing the ceiling. The resulting map is compatible with common SLAM and mapping methods (occupation map). Several difficulties were faced when dealing with problems intrinsic from the scanning strategy, as using movement compensation to tackle the low update frequency inherent to LIDAR sensors. A prototype was developed to validate the mapping strategy. Navigation experiments were carried out using the new mapping strategy and the results are presented on this work.

Keywords: Rotating LIDAR, 3D mapping, non-traditional mapping, mobile robots

LISTA DE FIGURAS

FIGURA 1 – Exemplo de funcionamento de um sensor de captura por retrodifusão (como os sensores LIDAR). Retirado de (POULIN, 2014). 21
FIGURA 2 – Sensor Dimetix EDS-C. 22
FIGURA 3 – 3a: Sensor Hokuyo URG-04LX. 3b: Funcionamento de um sensor *laser scan*. Retirado de (SIEGWART et al., 2011). 22
FIGURA 4 – Câmeras 3D comumente utilizadas na área de robótica móvel. 4a: Microsoft Kinect 1.0. 4b: MESA Swiss Ranger 4000. 23
FIGURA 5 – Sensor omnidirecional por espelho hiperbólico e projeção de laser, retirado de (JIA et al., 2015). 5a: Protótipo omnidirecional utilizando espelho hiperbólico e lasers projetados. 5b: Imagem capturada pelo espelho com a os pontos do laser projetados. 25
FIGURA 6 – Métodos de representação 3D. 6a: Octomap, um dos métodos de representação 3D mais utilizados em robótica. 6b: Grid map, método 2.5D de representação por elevação. Retirado de (FANKHAUSER et al., 2014). 26
FIGURA 7 – Métodos de representação 3D utilizando cor. 7a: Uma nuvem de pontos com informação de cor. Retirado de (CHEN et al., 2014). 7b: Mapa octomap utilizando informação de cor. Retirado de (ENDRES et al., 2012). 27
FIGURA 8 – Métodos de mapeamento e SLAM em robótica. 8a: SLAM com mapa Costmap2D retirado de (SELBY, 2016). 8b: SLAM utilizando mapeamento 3D, retirado de (ENGEL et al., 2014). 28
FIGURA 9 – Métodos de mapeamento não-tradicionais. 9a: Mapa de superfície e planejamento de trajetória a partir de imagens. Retirado de (BELTER; SKRZYPCZYNSKI, 2013). 9b: Método de representação de mapa utilizando contorno de planos compartilhado entre múltiplos robôs. Retirado de (NIETO-GRANDA et al., 2014). 30
FIGURA 10 – Estrutura do protótipo do sensor desenvolvido. 32
FIGURA 11 – 11a: Perfil de alumínio 2020 utilizado. 11b: Perfil de alumínio 2020 no software SketchUp. 34
FIGURA 12 – 12a: Modelo da peça que abraça o motor. 12b: Peça que abraça o motor impressa e montada no projeto. 36
FIGURA 13 – 13a: Modelo da peça que fixa a extremidade de atuação do motor. 13b: Peça que fixa o motor impressa e montada no projeto. 36
FIGURA 14 – 14a: Modelo da peça que prende o Hokuyo no eixo do motor. 14b: Peça impressa e montada no projeto sustentando o Hokuyo ao eixo do motor. . 37
FIGURA 15 – 15a: Modelo da peça que suporta a câmera e o espelho hiperbólico. 15b: Peça da câmera após impressa no projeto. 38
FIGURA 16 – Tela do software EPOS Studio, utilizado na calibração do motor. 41
FIGURA 17 – Exemplo de uma triangularização Delaunay, retirada de (SCHINDELIN, 2008). 44
FIGURA 18 – Exemplo de um grid construído pelo pacote 'grid map'. 44
FIGURA 19 – Janela deslizante inicializada sem dados assinalados. 46

FIGURA 20 – Nós de processamento do algoritmo de mapeamento, robô de ensaio, motor e sensor. Em azul, nós de pacotes necessários para o funcionamento do protótipo. Em preto, nós do pacote do robô móvel utilizado. Em vermelho, nós do pacote desenvolvido para o método de mapeamento proposto. 48
FIGURA 21 – Árvore de transformadas resultante do robô e dos mapas. 49
FIGURA 22 – Diagrama de blocos com a estrutura geral do software de mapeamento. .. 50
FIGURA 23 – Diagrama de blocos com a estrutura geral do software construção do mapa local. 51
FIGURA 24 – Mapas locais (frente, esquerda e direita) e globais (chão e teto) inicializados e limpos. 52
FIGURA 25 – 25a: Representação das distâncias (em laranja) medidas para grids (em verde) de 10 células em relação ao círculo de raio 'd' (em azul). 'c' representa a distância que é armazenada no vetor (uma para cada célula). 25b: Vetor final 'C' de distâncias entre células do mapa e um círculo de raio 'd'. 56
FIGURA 26 – 26a: Representação dos ângulos coletados para grids de 10 células. 'ϕ' é o ângulo coletado para cada célula e armazenado no vetor. 26b: Vetor final 'ϕ' com os ângulos entre os dois lados opostos de cada célula e o centro do robô. 57
FIGURA 27 – Diagrama de blocos com a estrutura geral do software construção do mapa global. 58
FIGURA 28 – Mapas locais e globais do método omnidirecional. Mapa local de chão em azul e de teto em vermelho. Mapa global de chão em amarelo e de teto em rosa. 60
FIGURA 29 – 29a: Primeira cena de ensaio virtual, apenas uma sala fechada com formas geométricas de fácil identificação. 29b: Segunda cena de ensaio virtual, um lugar aberto com pessoas e objetos complexos. 65
FIGURA 30 – Primeira cena do ensaio virtual com resolução do mapa local igual a 5cm (30a), 10cm (30b) e 20cm (30c). O mapa global resultante é semelhante para todas as cenas. 66
FIGURA 31 – Segunda cena do ensaio virtual com resolução do mapa local igual a 5cm (31a), 10cm (31b) e 20cm (31c). O mapa global resultante é semelhante para todas as cenas. É possível perceber que vários detalhes do ambiente são perdidos comparando as cenas 31a e 31c. 66
FIGURA 32 – Mapa global do segundo ambiente com limite de teto em 60cm (32a) onde o chão sob a mesa é considerado navegável, e em 1 metro (32b) onde a mesa é considerada um obstáculo. A posição da mesa está marcada em azul nas imagens. 66
FIGURA 33 – 33a: Cena real utilizada para o primeiro ensaio. 33b: Cena real utilizada para o segundo ensaio. 67
FIGURA 34 – Primeira cena do ensaio real com resolução do mapa local igual a 5cm (34a), 10cm (34b) e 20cm (34c). O mapa global resultante é semelhante para todas as cenas. 67
FIGURA 35 – Segunda cena do ensaio real com resolução do mapa local igual a 5cm (35a), 10cm (35b) e 20cm (35c). 67
FIGURA 36 – Mapa global do primeiro ambiente real com limite de teto em 60cm (36a) onde o chão sob as bancadas é considerado navegável, e em 1 metro (36b) as bancadas são consideradas um obstáculo. 68

FIGURA 37 – Cenas utilizadas nos ensaios de rotação. 37a: Cena virtual. 37b: Cena real. 69
FIGURA 38 – Mapas virtuais iniciais antes da rotação. 38a: Mapa local. 38b: Mapa global. 69
FIGURA 39 – Mapas virtuais após rotação com método de aproximação. 39a: Mapa local. 39b: Mapa global. 70
FIGURA 40 – Mapas virtuais após rotação com método de transformação sem correção de erro. 40a: Mapa local. 40b: Mapa global. 70
FIGURA 41 – Mapas virtuais após rotação com método de transformação utilizando correção de erro. 41a: Mapa local. 41b: Mapa global. 71
FIGURA 42 – Mapas reais escaneados antes da rotação. 42a: Mapa local. 42b: Mapa global. 72
FIGURA 43 – 43a e 43b: Mapas local e global após rotação utilizando compensação por aproximação. 43c e 43d: Mapas local e global após rotação utilizando compensação por transformação sem correção de erro. 43e e 43f: Mapas local e global após rotação utilizando compensação por transformação com correção de erro. 73
FIGURA 44 – Ambientes onde os ensaios de navegação foram realizados. 44a: Ambiente real. 44b: Ambiente virtual. 74
FIGURA 45 – Mapas globais resultantes da navegação em ambiente virtual utilizando compensação de rotação por aproximação. 45a: Antes da rotação final. 45b: Após a rotação final. 75
FIGURA 46 – Mapas globais resultantes da navegação em ambiente virtual utilizando compensação de rotação por transformação sem utilizar correção de erro. 46a: Antes da rotação final. 46b: Após a rotação final. 75
FIGURA 47 – Mapas globais resultantes da navegação em ambiente virtual utilizando compensação de rotação por transformação utilizando correção de erro. 47a: Antes da rotação final. 47b: Após a rotação final. 76
FIGURA 48 – Mapas globais criados a partir da navegação em ambiente real utilizando método de aproximações. 48a: Antes da rotação final. 48b: Após a rotação final. 77
FIGURA 49 – Mapas globais criados a partir da navegação em ambiente real utilizando compensação de rotação por transformação sem aplicar correção de erro. 49a: Antes da rotação final. 49b: Após a rotação final. 77
FIGURA 50 – Mapas globais criados a partir da navegação em ambiente real utilizando compensação de rotação por transformação aplicando correção de erro. 50a: Antes da rotação final. 50b: Após a rotação final. 78
FIGURA 51– Ensaio de reconstrução realizado com o sensor posicionado horizontalmente. 51a: cena e robô virtuais utilizados no ensaio, 51b: Mapa local reconstruído a partir do movimento de inclinação do sensor, 51c: Mapa global construído a partir do mapa local. 79
FIGURA 52 – Mapas locais dos ensaios de navegação com o sensor na horizontal varrendo sentido cima/baixo. 52a: Mapa local após movimento de 1 metro para frente. 52b: Mapa local após rotação de 45 graus a esquerda. 80
FIGURA 53 – Ensaio realizado para o método omnidirecional de escaneamento. 53a: cena e robô virtuais utilizados no ensaio, 53b: Mapa local reconstruído a partir método omnidirecional de mapeamento, 53c: Mapa global construído a partir do mapa local. 80
FIGURA 54 – Capturas realizadas pela câmera RGB fixada no protótipo do sensor durante

ensaio de navegação em ambiente real. 54a: Começo do ensaio, 54b: Metade do trajeto, 54c: Final da navegação. 81
FIGURA 55 – Algoritmo de detecção utilizando YOLO, uma rede neural treinada para detecção de objetos e pessoas. 55a: Detecção de pessoas. 55b: Detecção de obejtos. 82

LISTA DE TABELAS

TABELA 1 – Lista de materiais que compõem o sensor. 32
TABELA 2 – Tabela de componentes do projeto e suas dimensões 35
TABELA 3 – Tabela de componentes elétricos e alimentação necessária. 38
TABELA 4 – Memória ocupada em duas situações pelo mapa local com grids fixos em 3m x 3m. A memória informada inclui os três grids que formam o mapa local. 59
TABELA 5 – Memória ocupada pelo mapa global fixado em 20m x 20m. 59

LISTA DE SIGLAS

SLAM	*Simultaneous Localization and Mapping* - Localização e Mapeamento Simultâneos
GPS	*Global Position System* - Sistema Posicionamento Global
RGB	*Red Green Blue* - Vermelho, Verde e Azul, os três canais de cor utilizados para representação de imagem
RGB-D	*Red Green Blue Distance* - Representação dados 3D (distância) junto com dados RGB (cor)
LIDAR	*Light Detection and Ranging* - Tecnologia de btenção de distância através da luz
ROS	*Robot Operating System* - Plataforma de *software* sobre o qual é desenvolvido a maioria dos sistemas robóticos descritos neste trabalho
GPU	*Graphics Processing Unit* - Unidade de processamento gráfico
DVS	*Dynamic Vision Sensor* - Sensor de visão dinâmica
IMU	*Inertial Measurement Unit* - Unidade de medição inercial, sensor que informa dados como aceleração, inclinação e velocidade
CAD	*Computer Aided Design* - Design auxiliado por computador, formato de representação de modelos 3D muito utilizado na engenharia
USB	*Universal Serial Bus* - Barramento Serial Universal
DC	*Direct Current* - Corrente contínua
CPU	*Central Processing Unit* - Unidade de processamento central
CUDA	*Compute Unified Device Architecture* - Interface de programação de aplicações orientadas a processamento paralelo (GPU)
STL	*Standard Template Library* - Biblioteca Padrão de Modelos, formato de representação de modelos 2D e 3D
CAN	*Controller Area Network* - Rede de Área de Controle, tecnologia de envio e recebimento de mensagens de controle em redes locais
URDF	*Unified Robot Description Format* - Formato unificado de Descrição de Robôs
YAML	*YAML Ain't Markup Language* (acrônimo recursivo) - Formato de serialização de dados legíveis
Rviz	*ROS Visualization* - Visualizador gráfico padrão de mensagens do ROS
LASCA	Laboratório de Automação e Sistemas de Controle Avançado - Laboratório onde este trabalho foi desenvolvido
V-Rep	*Virtual Robot Experimentation Platform* - Plataforma de Representação Visual de Robôs
YOLO	*You Only Look Once* - Algoritmo de detecção de objetos baseado em *deep learning*
HD	*High Definition* - Alta Definição

SUMÁRIO

1 INTRODUÇÃO 14
1.1 MOTIVAÇÃO 15
1.2 OBJETIVOS 16
1.2.1 Objetivos Específicos 17
1.3 ESTRUTURA 18
2 ESTADO DA ARTE 19
2.1 ROS - ROBOT OPERATING SYSTEM 19
2.2 FONTES DE PERCEPÇÃO DE DISTÂNCIA 21
2.3 SENSORIAMENTO TRIDIMENSIONAL 24
2.4 MÉTODOS DE MAPEAMENTO DE AMBIENTES 26
3 PROTÓTIPO DO SENSOR 31
3.1 PROJETO MECÂNICO 33
3.2 PROJETO ELÉTRICO E ELETRÔNICO 38
3.3 MOVIMENTOS 40
4 MAPEAMENTO 43
4.1 SOFTWARE 46
4.1.1 Mapeamento por janela deslizante 47
4.1.2 Mapa omnidirecional 60
4.2 PARÂMETROS 61
5 ENSAIOS E RESULTADOS 64
5.1 RECONSTRUÇÃO DE AMBIENTE 65
5.2 COMPENSAÇÃO DE ROTAÇÃO 68
5.3 NAVEGAÇÃO 72
5.4 ENSAIOS ADICIONAIS 78
6 CONCLUSÃO 83
REFERÊNCIAS 85

1 INTRODUÇÃO

Em robótica, os sistemas de sensoriamento são responsáveis por fornecer aos robôs dados sobre o ambiente em que estão inseridos. A capacidade de um robô móvel se autolocalizar e navegar em um determinado ambiente depende da qualidade de seus sensores, que introduzem as habilidades de perceber obstáculos, objetos, pessoas, variações na elevação, entre outras. Nesse contexto, sensores que são aplicados para extração de informações do ambiente em que o robô está inserido são denominados como *fontes de percepção*.

A capacidade autônoma de navegação de um robô depende da qualidade e quantidade de informações extraídas do ambiente de execução da tarefa. Robôs autônomos podem realizar um planejamento de trajetória mais preciso quando dispõem de mais informações sobre o ambiente. Fontes de percepção de profundidade tridimensional permitem ao robô identificar a distância entre si mesmo e obstáculos no ambiente. Fontes de percepção puramente visuais permitem identificar pontos de interesse para melhorar sua odometria, identificar pessoas e objetos, ou até mesmo seguir uma trajetória pré-determinada. A utilização de dados de ambos os tipos de percepção permite ao robô até mesmo classificar objetos para aprendizado. Isto pode ser visto em (GUPTA et al., 2017), onde os autores utilizam fontes de percepção e 3D para formar um mapa junto a um sistema cognitivo, de modo que é possível ordenar 'vá até a porta' ao robô e ele procurará a porta mais próxima e navegará até ela.

Segundo (SIEGWART et al., 2004), um robô móvel deve ser capaz de dizer onde está localizado, onde está seu objetivo e como alcançá-lo. Para isto, se deve seguir o ciclo '*see-think-act*' ou visualizar, pensar e agir. Este ciclo inicia com a informação crua dos sensores do robô, que coletam dados do ambiente para que informações sejam extraídas dele (*see*), no exemplo de um robô autônomo isto significa coletar informações do ambiente e construir um mapa para permitir a localização. Em seguida o robô deve ser capaz de processar as informações (*think*) de sua localização e do mapa construído para gerar uma rota ou um plano de ação até o seu objetivo. Após isso, o robô pode utilizar seus atuadores para se locomover ou alcançar seu objetivo (*act*). Este ciclo então é reiniciado com um novo objetivo.

A maioria das fontes de percepção disponíveis se baseiam em câmeras com um campo de visão limitado, dependendo fortemente do movimento do robô, que necessita estar direcionado ao objeto ou obstáculo a ser reconhecido (SCHMID et al., 2013). A concentração das fontes de percepção na parte dianteira de robôs móveis é a estratégia mais utilizada, porém isso deixa o robô vulnerável quando em ambientes dinâmicos ou em situações onde o robô está preso e ele necessita manobrar, dando ré. Outra estratégia empregada é dispor os sensores em uma base rotativa independente do movimento do robô, comum em robôs humanoides (BEIRA et al., 2006), porém isto ainda possui a limitação de não poder olhar para dois lados ao mesmo tempo. Devido a estas limitações, existe a necessidade de uma fonte de percepção tridimensional ampla que não dependa de outras fontes de percepção e que permita ao robô identificar mudanças no ambiente provenientes de qualquer direção.

Na área de robôs móveis, sensores são utilizados principalmente para realizar localização e mapeamento simultâneos (SLAM). A técnica de SLAM consiste em construir um mapa do ambiente a partir dos dados dos sensores espaciais do robô e identificar o próprio robô dentro deste mapa a medida que este se move. A dificuldade do SLAM se encontra na hora de unir as aquisições das diferentes fontes de percepção, uma vez que o erro de odometria do robô é transmitido ao mapa (a não ser que alguma estratégia de correção seja aplicada) e a própria odometria do robô depende do mapa para ser exatamente calculada (a não ser que o robô possua algum método baseado em GPS). Para minimizar erros na hora da montagem do mapa do ambiente, técnicas de identificação de pontos de interesse, filtros de Kalman (CHEN, 2012) ou métodos de Monte Carlo (ERDOGAN et al., 2012) podem ser aplicados. Esses métodos dependem da quantidade de dados disponíveis no mapa e da distribuição dos novos dados coletados para otimizar a localização, portanto, um sensor tridimensional pode acelerar a curva de otimização desses algoritmos fornecendo dados de todo o ambiente antes de sensores mais precisos realizarem a leitura.

1.1 MOTIVAÇÃO

Na área de sensores e visão computacional, muitos pesquisadores desenvolveram diferentes tipos de fontes de percepção 3D. Em (STEINBRÜCKER et al., 2014) é desenvolvido um algoritmo de mapeamento através da geração de superfície a partir de nuvem de pontos (*point cloud*) e dados RGB. Em (HORNUNG et al., 2013) é descrito o Octomap, uma abordagem para construção de mapas 3D de modo dinâmico a partir de qualquer sensor espacial. Em (HANDA et al., 2014) os autores implementam um algoritmo de fusão RGB-D e odometria para um Microsoft Kinect segurado à mão para reconstrução de ambientes internos. Em

(BELBACHIR et al., 2014) os autores descrevem um sensor RGB-D utilizando câmeras RGB estereoscópicas para fazer aquisição 3D, as câmeras são rotacionadas em 10 revoluções por segundo permitindo uma aquisição panorâmica. Na literatura, não foram encontrados trabalhos que realizassem uma aquisição omnidirecional 3D e um mapeamento orientado a *laser scan* que permitisse fusão com dados RGB.

A utilização de um sensor *laser scan* é preferida pois câmeras 3D existentes são caras e dificultam a leitura omnidirecional. Tornar possível um mapeamento 3D capaz de reconstruir ambientes utilizando componentes de baixo custo é um dos objetivos deste trabalho. Um sensor *laser scan* como o utilizado no protótipo descrito nesta dissertação (Hokuyo URG-04LX) pode ser encontrado por 1000$ enquanto que câmeras 3D com o mesmo alcance e acurácia que os pontos coletados pelo *laser scan* são encontradas a partir de 6000$, exemplos são a Swiss Ranger 4000 e a PMD CamCube 2.0. Câmeras 3D mais baratas porém com menor precisão podem ser encontradas por 250$, como a Microsoft Kinect v1.0 ou a Intel Real Sense, porém não possuem precisão ou alcance que se comparem as outras, prejudicando a acurácia e versatilidade do método proposto.

A proposta deste trabalho é desenvolver um método de mapeamento 3D, através de fontes de percepção com baixa influência de luminosidade e baixo custo, através de varredura tridimensional a laser. O protótipo de sensor proposto realiza escaneamento 3D em tempo real coletando pontos do ambiente e assinalando-os no mapa. O protótipo é omnidirecional, conseguindo aquisições em qualquer direção. O método de mapeamento é orientado à navegação utilizando este tipo de sensor (escaneamento).

Tanto o protótipo construído quanto o método de mapeamento proposto suportam a coleta e assinalação de dados RGB, porém esta etapa não foi implementada e nenhum processamento de imagem ou coleta RGB estão incluídos neste trabalho.

1.2 OBJETIVOS

Esta dissertação tem por objetivo desenvolver um sensor laser omnidirecional para a melhoria da precisão de mapeamento para navegação e permitindo mapeamento omnidirecional. O mapeamento para navegação é feito utilizando um método de janela deslizante, e deve ser capaz de realizar leitura e reconstrução do ambiente tridimensional na direção do movimento do robô, a fim de construir um mapa 3D fiel do ambiente, e também podendo auxiliar outros sensores, funcionando como um sensor secundário. O mapeamento omnidirecional tem como objetivo realizar a percepção do ambiente sem pontos cegos, este método de mapeamento é

proposto porém não é o foco principal do trabalho. Para realizar estes objetivos é utilizada uma estratégia de mapeamento tridimensional que executa ambas as funções do sensor, tendo suporte à inserção de dados extras para auxiliar a visualização. Para atingir o objetivo de ser omnidirecional, é necessária a rotação de um sensor, uma vez que todos os sensores 3D disponíveis para o projeto têm campo de visão limitado.

Este trabalho foca em realizar duas contribuições: a construção de um protótipo para escaneamento omnidirecional utilizando um sensor *laser scan* rotativo, e um mapeamento 3D não-tradicional deslocando três grids verticais junto ao robô para reconstrução de ambientes. Estes três grids representam o mapa local do robô e é a partir deles que o mapa global é construído. O mapa global possui informações de obstáculos no chão, a partir do qual um mapa de ocupação é gerado, e do teto, representando obstáculos acima do robô que permitem o robô navegar por baixo. O objetivo de registrar todas estas informações é para que futuramente um registro de imagem seja feito permitindo uma reconstrução fidedigna. Uma versão omnidirecional do método proposto também é detalhada neste trabalho.

1.2.1 OBJETIVOS ESPECÍFICOS

- Desenvolver um sensor com capacidade de operar 360 graus através de uma plataforma rotativa. O sensor será um *laser scanner* que obterá pontos em uma esfera em volta do robô a medida que for rotacionando. A orientação do giro e velocidade de rotação deverão ser definidas para balancear número de pontos coletados com velocidade de varredura.

- Criar um método de mapeamento 3D que seja capaz de atualizar o mapa a cada conjunto de pontos novos coletados em tempo real. O mapa deve ser feito no estilo superfície 3D permitindo reconhecer vãos verticais e horizontais.

- Propor um método de mapeamento para suporte à varredura feita por sensores 3D rotacionando. Além disso, deve representar os dados de forma que seja possível aplicar ambas as estratégias propostas, omnidirecional e janela deslizante. O método de mapeamento também deve permitir informações adicionais no mapa como cor ou posição de objetos dinâmicos.

- Realizar experimentos em ambiente virtual e em um robô real, para comprovar a eficiência do sensor e do método de mapeamento proposto.

1.3 ESTRUTURA

Este documento é organizado em seis capítulos. O primeiro trata da apresentação do tema, motivação e objetivos da dissertação. O capítulo 2 discute o estado da arte e conceitos principais de percepção e mapeamento tridimensional. O capítulo 3 apresenta o sensor 3D desenvolvido para a varredura omnidirecional do ambiente, detalhando o projeto mecânico, elétrico e eletrônico bem como a calibração e configuração do motor acoplado no protótipo. O capítulo seguinte (capítulo 4) descreve o método de mapeamento idealizado e suas restrições. O capítulo 5 detalha os ensaios realizados para validação do projeto. Ensaios virtuais e reais são conduzidos utilizando variações de parâmetros e métodos de mapeamento. Por fim, a conclusão do projeto é apresentada no capítulo 6.

2 ESTADO DA ARTE

Este capítulo trata da introdução de ideias necessárias para o compreendimento do trabalho, da apresentação das principais fontes de percepção de ambientes e da discussão do estado da arte de mapeamento tridimensional por robôs móveis.

2.1 ROS - ROBOT OPERATING SYSTEM

O *Robot Operating System* 1.0, comumente chamado de ROS, é uma coleção de ferramentas, bibliotecas e funções para o desenvolvimento de *software* focado no funcionamento de robôs. Seu objetivo é simplificar a tarefa de criar comportamentos complexos em robôs, através de sua variedade de ferramentas de programação e comunicação, e plataformas de simulação robóticas (ROS.ORG, 2016).

O ROS foi criado com o intuito de incentivar o desenvolvimento de *software* voltado à robótica de modo colaborativo, permitindo que diversos grupos e laboratórios possam colaborar no mesmo projeto. No ROS, cada componente capaz de processamento (como por exemplo um computador ou um periférico) é tratado como um 'nó'. Estes nós são conectados uns aos outros através de 'tópicos' formando uma árvore computacional. Os tópicos comunicam entre si enviando e recebendo de 'mensagens'. Esta estrutura forma a base principal do programa, integrando todos os componentes do robô sendo desenvolvido.

As mensagens podem assumir desde formatos simples, tal como um *float* ou uma *string*, até formatos complexos, tal como uma estrutura que contém vários campos. Nós podem 'publicar' as mensagens em tópicos para transmitir dados, e qualquer nó 'inscrito' neste tópico terá acesso a informação transmitida, permitindo um modo simples e rápido de transmissão de dados.

Para facilitar a criação de mensagens, o ROS possui formatos padrões de mensagens para os componentes mais utilizados, como por exemplo vetores 3D para movimentação, posição e orientação para representar a 'odometria' do robô, mensagens padrões para imagens

de câmeras, vetores de distância de lasers e outros componentes comumente encontrados em robôs.

Odometria é a estimação da localização do robô através de dados provenientes de sensores de movimento, posição das rodas ou do processamento de dados visuais, necessária em um robô móvel que precisa se localizar enquanto realiza 'mapeamento'. Mapeamento é o uso de sensores 2D ou 3D para construir mapas do local onde o robô se encontra, permitindo reconstruir o ambiente sobre o qual o robô pode navegar. A utilização de odometria enquanto o robô realiza a construção de mapas é chamada de 'localização e mapeamento simultâneos' (SLAM).

O ROS também conta com bibliotecas de conversão que eliminam tempo de programação através de ferramentas úteis, como por exemplo a biblioteca *transform* ('tf') que armazena a localização de cada sensor e atuador do robô para facilitar a conversão de entre diferentes 'referências' (*frames*). A definição de referências deve ser feita quando se desenvolve um sensor, método de navegação ou robô móvel, pois permite a utilização da 'tf' na construção da 'árvore de transformadas' (*tf tree*). Esta árvore contém as transformadas que definem a conversão entre os pontos de referência e é consultada sempre que a 'tf' é aplicada. A referência principal da árvore de transformadas comumente adotada é chamada de 'base link', o ponto central do robô móvel.

Além destas funcionalidade, o ROS também possui diversos *softwares* auxiliares de visualização e simulação. O visualizador de dados mais utilizado é chamado de 'Rviz'. Nele, é possível visualizar a grande maioria dos tópicos que podem ser representados em um espaço 3D ou 2D, como leituras de distância, mapas, posição do robô móvel ou imagem de câmeras. Para auxiliar no desenvolvimento de robôs, existem simuladores que criam um ambiente físico virtual, que permitem a definição de sensores, atuadores e modelos 3D do robô e do ambiente, integrados de modo a interagir similares à realidade. Os mais utilizados ambientes de simulação 3D para ROS são o 'Gazebo' (KOENIG; HOWARD, 2004) e o 'V-rep' (COPPELIA, 2018).

O ROS é uma ferramenta essencial para o desenvolvimento de robôs. Suas funcionalidades permitem fácil implementação do sistema em diversas camadas, desde o nível mais baixo, como a identificação dos dados de sensores do robô, até alto nível, como o planejamento de trajetória e construção de mapa. A estratégia de mapeamento e protótipo de sensor apresentados neste projeto são baseados no *framework* ROS e tiram vantagem de várias de suas funcionalidades. O protótipo de sensor construído serve como fonte de percepção a robôs móveis. Uma revisão sobre fontes de percepção 3D utilizadas em robôs móveis é apresentada na próxima seção.

2.2 FONTES DE PERCEPÇÃO DE DISTÂNCIA

A utilização de lasers para o reconhecimento de distâncias começou na década de 70 quando a tecnologia LIDAR (*Light Detection and Ranging*) foi criada, sendo largamente utilizada nos dias atuais. A tecnologia LIDAR é popular, pois permite a leitura, em um curto intervalo de tempo, de longas distâncias com precisão. Desde então diversos tipos de sensores utilizando a tecnologia LIDAR foram criados, com destaque para os *range finders*, sensores de apenas um ponto de aquisição mas com longo alcance e alta precisão, e os *laser scanners*, que rotacionam um espelho sobre um laser para a realização de várias aquisições por segundo e mapeamento de vários pontos em um feixe de 180 graus ou mais.

O princípio de funcionamento do LIDAR consiste em emitir um feixe de luz cuja frequência pode variar entre o ultravioleta, passando pela luz visível, até o infravermelho. A frequência luminosa determina a distância máxima de leitura e o tempo de aquisição. Quando este feixe atinge um objeto, a luz é refletida de modo difuso (retrodifusão). A luz refletida é captada novamente por um fotodetector (figura 1), e o tempo levado desde a emissão até a recepção adquirido pela diferença de fase do feixe emitido determina a distância do objeto.

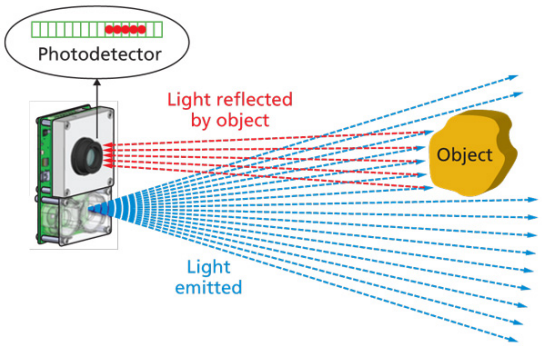
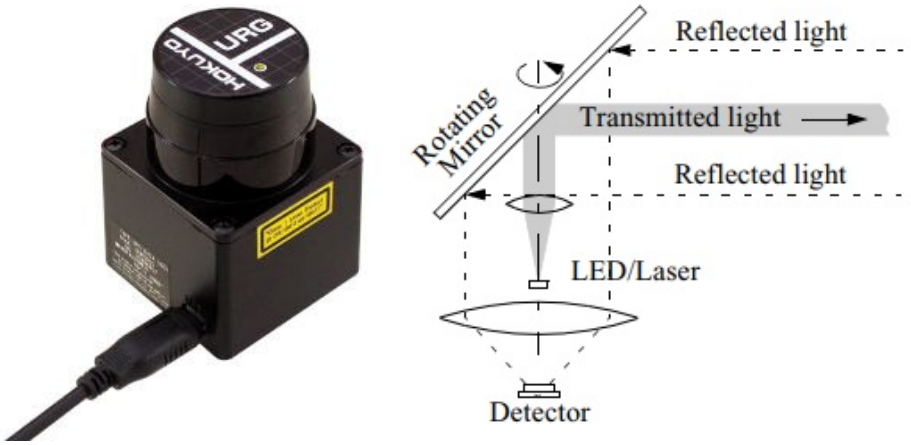


Figura 1: Exemplo de funcionamento de um sensor de captura por retrodifusão (como os sensores LIDAR). Retirado de (POULIN, 2014).

Range finders são sensores que medem apenas um ponto com frequência de aquisição baixa porém com grande precisão e longo alcance. Eles são utilizados em robótica para garantir a acurácia em uma fusão de sensores. Como a precisão deste sensor é muito alta comparada a outros sensores 3D, é comum utilizá-los como base para calibração antes ou durante a navegação do robô, como pode ser visto em (NGUYEN; REITMAYR, 2013). Um exemplo de sensor *range finder* é o Dimetix EDS-C, mostrado na figura 2, que possui apenas um ponto de aquisição, fazendo uma leitura por segundo com alcance de 30m e precisão de 3mm (DIMETIX, 2018).

Figura 2: Sensor Dimetix EDS-C.

Os *laser scanners* são frequentemente utilizados em robôs móveis para a realização de mapeamento e localização, pois proporcionam um modo fácil e rápido de aquisição de dados espaciais para geração de mapas 2D. *Frameworks* de operação de robôs, como o ROS (*Robot Operating System*), já possuem interfaces de software (pacotes) prontos para instalação e configuração dos principais modelos de sensores. Como exemplo de *laser scanner* pode-se citar o mais utilizado na área de robótica, o Hokuyo URG-04LX (KNEIP et al., 2009), mostrado na figura 3a. Estes sensores utilizam apenas um emissor e receptor de feixe luminoso. A varredura ocorre através da rotação de um espelho inclinado 45 graus embaixo do laser, que reflete tanto a luz emitida quanto a sua retrodifusão (figura 3b). Durante o ciclo de rotação do espelho, várias aquisições de distância são feitas, que então são encapsuladas e transmitidas como uma aquisição *laser scan*. Estas aquisições por retrodifusão dependem das propriedades do material e do ângulo de incidência. Para o sensor URG-04LX o erro a partir da variação destes parâmetros de incidência não passa de 3% da leitura. Um estudo sobre estas características para sensores *laser scan* pode ser visto em (PARK et al., 2010).



(a) (b)

Figura 3: 3a: Sensor Hokuyo URG-04LX. 3b: Funcionamento de um sensor *laser scan*. Retirado de (SIEGWART et al., 2011).

Laser scan é um formato de representação de dados provenientes de sensores LIDAR. Foi especialmente criado para sensores que fazem leitura por escaneamento, acumulando

ordenadamente as medições de distância de uma revolução completa do espelho giratório. Junto com o vetor de distâncias, também estão encapsulados na mensagem dados de ângulo mínimo e máximo (começo e fim da varredura) e ângulo de passo (distância angular entre duas medidas consecutivas), permitindo representar cada ponto lido em um espaço tridimensional. Movimentar um sensor LIDAR, seja por inclinação ou rotação, permite um escaneamento em três dimensões, organizando as várias leituras *laser scans* como linhas de uma matriz convertida em uma nuvem de pontos. Um exemplo deste tipo de escaneamento pode ser encontrado em (KLIMENTJEW et al., 2010) onde o autor utiliza o Hokuyo inclinando em movimento cima/baixo (pan tilt) e duas câmeras RGB para escanear e reconstruir ambientes com cor.

Outros tipo de sensor que pode ser utilizados para aquisição espacial são as câmeras 3D, como por exemplo o Microsoft Kinect (figura 4a), uma câmera que coleta informações de alcance em 480x640 pixels utilizando tecnologia de luz estruturada (PAGLIARI; PINTO, 2015), ou a Swiss Ranger 4000 (figura 4b), uma câmera que utiliza tecnologia tempo de voo para gerar imagens de distância em 176x144 pixels (IMAGING, 2010).

(a) (b)

Figura 4: Câmeras 3D comumente utilizadas na área de robótica móvel. 4a: Microsoft Kinect 1.0. 4b: MESA Swiss Ranger 4000.

Câmeras 3D como as citadas anteriormente utilizam a representação em nuvem de pontos como formato padrão de encapsulamento das informações coletadas. Nuvens de pontos são mensagens que contém todos os pontos representados em [x, y, z] coletados em um vetor de tamanho linha x coluna da resolução da câmera. As informações de representação como resolução do sensor também estão incluídas na mensagem.

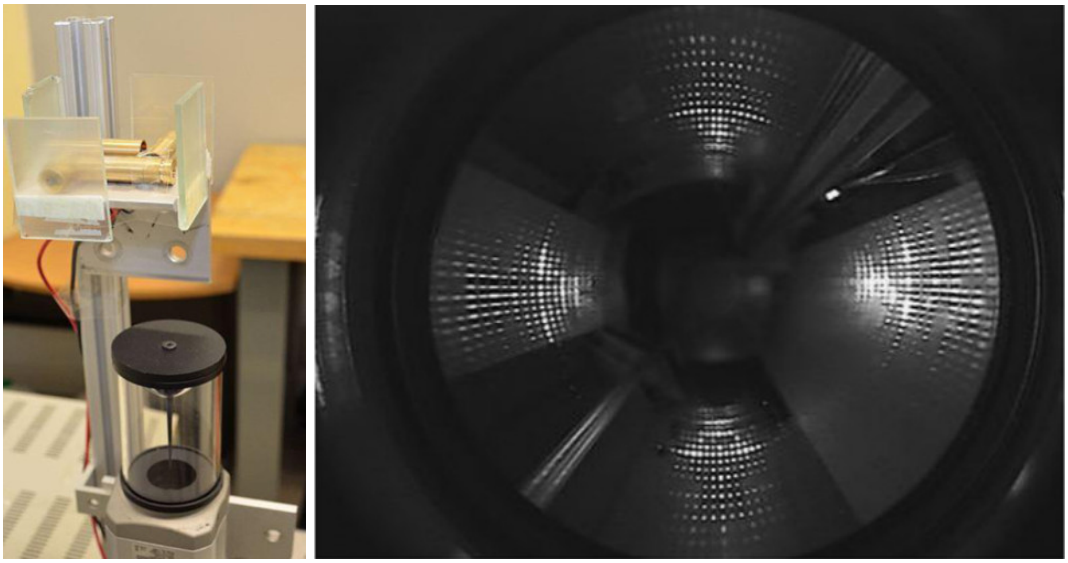
Apesar de mapas 2D serem um formato de representação já bem definido, mapas 3D ainda são tópicos de discussão, uma vez que existem inúmeras maneiras de se representar um mapa 3D e diversos algoritmos já foram propostos para gerá-los de forma rápida e eficiente. Mapas 3D são gerados a partir de nuvens de pontos (várias aquisições de distâncias), que são provenientes de medidas de alcance feitas por um sensor LIDAR ou uma câmera 3D.

2.3 SENSORIAMENTO TRIDIMENSIONAL

O desenvolvimento de novos métodos de sensoriamento atualmente são baseados no uso de uma fonte de percepção já existente em um arranjo inovador, por exemplo a utilização de múltiplas câmeras RGB para extração de pontos 3D (MEILLAND et al., 2015), espelhos em conjunto com câmeras para atingir omnidirecionalidade (LIN; BAJCSY, 2003) ou rotação de sensores para aumentar o campo de visão resultante (NEMOTO et al., 2007).

Associado ao arranjo de sensores, são desenvolvidos novos algoritmos e métodos que permitam a construção de mapas baseados em uma grande quantidade de dados. Um sensor omnidirecional é qualquer sensor que consiga fazer leituras em 360 graus completos em duas ou três dimensões, independente se a cobertura for total para cada aquisição ou se forem aquisições sequenciais até atingir a cobertura completa. Na área de sensores 3D omnidirecionais, poucos trabalhos recentes são encontrados. Um dos primeiros trabalhos utilizando espelhos para capturas omnidirecionais pode ser encontrado em (LIN; BAJCSY, 2003) onde o autor utiliza câmeras catadióptricas panorâmicas para obter uma aquisição estéreo do ambiente, e com isso identificar obstáculos perto do robô. Em (HOUSHIAR et al., 2015) encontra-se o processo reverso. Considerando diversas aquisições de imagens panorâmicas e aquisições de nuvens de pontos do ambiente, os autores elencam as melhores projeções avaliando sua precisão durante o processo de registro baseado em características chave. Em (ZHANG et al., 2012) os autores desenvolvem um sensor 3D omnidirecional através de uma câmera, um projetor e dois espelhos hiperbólicos. O processo de calibração do sensor também é descrito. Em (JIA et al., 2015) os autores desenvolvem um sensor 3D omnidirecional utilizando um espelho hiperbólico, uma câmera e lasers de infravermelho, que juntos geram uma matriz de pontos infravermelhos cujo deslocamento permite o cálculo da distância similar a um sensor de luz estruturada, imagens na figura 5. Em (UKIDA et al., 2008) os autores utilizam dois espelhos hiperbólicos, duas câmeras e quatro projetores para coletar dados de distância em todas as direções, os projetores criam um padrão na superfície do ambiente que são reconhecidos pelas câmeras em seus espelhos, utilizando método estéreo para identificar a distância a cada projeção. Outro sensor utilizando uma câmera catadióptrica, um espelho hiperbólico e lasers de infravermelho pode ser visto em (XU et al., 2016), onde os autores utilizam tanto as câmeras estéreo quanto os lasers para conseguir uma medição omnidirecional bastante precisa de distância.

Neste contexto, o objetivo do trabalho descrito nesta dissertação é utilizar o conceito de espelhos hiperbólicos como ferramenta para aquisição de imagens panorâmicas, porém a aquisição 3D é feita unicamente pelo sensor LIDAR rotacionando (*rotating actuated LIDAR*).



(a) (b)

Figura 5: Sensor omnidirecional por espelho hiperbólico e projeção de laser, retirado de (JIA et al., 2015). 5a: Protótipo omnidirecional utilizando espelho hiperbólico e lasers projetados. 5b: Imagem capturada pelo espelho com a os pontos do laser projetados.

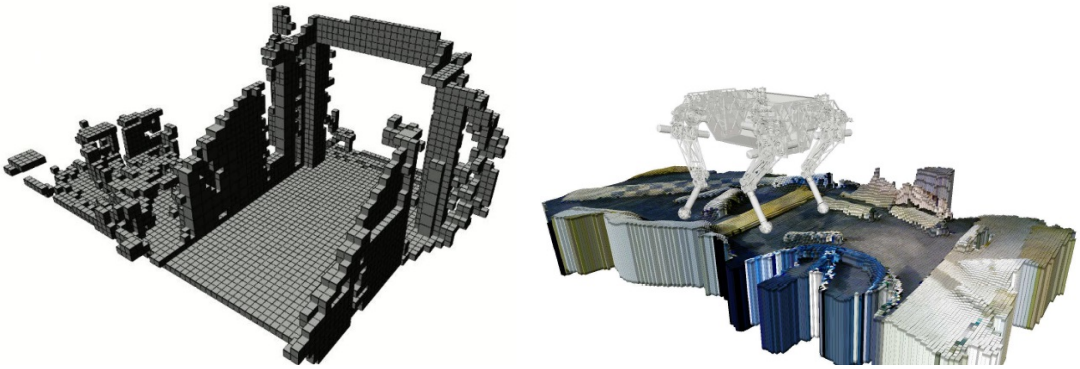
No âmbito de sensores LIDAR como ferramenta de escaneamento em três dimensões (através de rotação) pode-se citar o trabalho de (ALISMAIL et al., 2014) onde um algoritmo de registro e ajuste de leituras (*scan matching*) de um LIDAR rotativo durante navegação é desenvolvido, levando em conta erros de odometria entre aquisições e atingindo um alto nível de acurácia após registro, principalmente para situações de trajetória de *loop* aberto (quando o robô não é capaz de reconhecer obstáculos por onde já passou). Em (BOSSE; ZLOT, 2009) também é desenvolvido um algoritmo de registro para LIDAR rotativo durante navegação. Em (CHOU; LIU, 2013) é desenvolvido um sistema de mapeamento, utilizando também um algoritmo próprio de ajuste de leituras, orientado a escaneamento com LIDAR. Em (KURNIANGGORO et al., 2015) é apresentado um processo de calibração de um sensor LIDAR rotacionando utilizando restrição de pontos coplanares. No trabalho de (MEN et al., 2011) é realizado o registro de cores em um nuvens de pontos coletadas através de um LIDAR rotacionando, porém o registro aplicado requer o robô parado. Em (NEMOTO et al., 2007) um sensor LIDAR é rotacionado 360 graus para escanear o ambiente e uma superfície é criada a partir da nuvem de pontos, uma diferença aplicada em relação ao ambiente vazio permite identificar entidades móveis como pessoas. Por fim, em (ZHANG; SINGH, 2014) é desenvolvido um algoritmo de odometria e mapeamento orientado a LIDAR rotacionando utilizando ajuste de características (*feature matching*).

O protótipo de sensor desenvolvido descrito nesta dissertação tem o objetivo de providenciar fontes de aquisição omnidirecional nos formatos de nuvem de pontos e imagem

(RGB). A nuvem de pontos é coletada através de um sensor LIDAR e sua omnidirecionalidade é atingida pela rotação deste sensor. As aquisições RGB são feitas através de uma câmera comum com o auxílio de um espelho hiperbólico para tornar a imagem panorâmica.

2.4 MÉTODOS DE MAPEAMENTO DE AMBIENTES

Um modo de se representar mapas 3D é através de uma malha tridimensional de cubos (chamados de voxel), onde um voxel é marcado como ocupado se existe um ponto (da nuvem de pontos) identificado dentro dele e desocupado se não há pontos, este método é chamado de Octomap (figura 6a). Outro método busca criar um mapa de superfície, onde uma malha 2D plana é descrita previamente, e cada vértice ou célula é atualizado com a altura dos pontos coletados (figura 6b). Este método também é chamado de 2.5D, pois a terceira dimensão é representada apenas por elevação, de modo que dois pontos com posição igual mas alturas diferentes não são corretamente representados, impedindo a representação de vãos verticais, ou do teto (no caso de ambientes fechados).



(a) (b)

Figura 6: Métodos de representação 3D. 6a: Octomap, um dos métodos de representação 3D mais utilizados em robótica. 6b: Grid map, método 2.5D de representação por elevação. Retirado de (FANKHAUSER et al., 2014).

O registro de cores em uma nuvem de pontos é chamado de fusão RGB-D. A relação entre a imagem e a nuvem de pontos permite uma tradução direta entre um pixel da imagem e um ponto 3D correspondente, permitindo assim que algoritmos de processamento de imagens, computacionalmente mais rápidos, possam ser utilizados para detecção de características em uma nuvem de pontos, tornando mais eficientes processos como localização e mapeamento simultâneo (SLAM) ou reconstrução de sólidos. Como trabalhos nesta área pode-se citar (CHEN et al., 2014) onde é realizado localização e mapeamento RGB-D utilizando um sistema multi-robôs (figura 7a). Em (ENDRES et al., 2012) é desenvolvido um novo método de localização e mapeamento RGB-D, usando o Microsoft Kinect 1.0, o algoritmo identifica

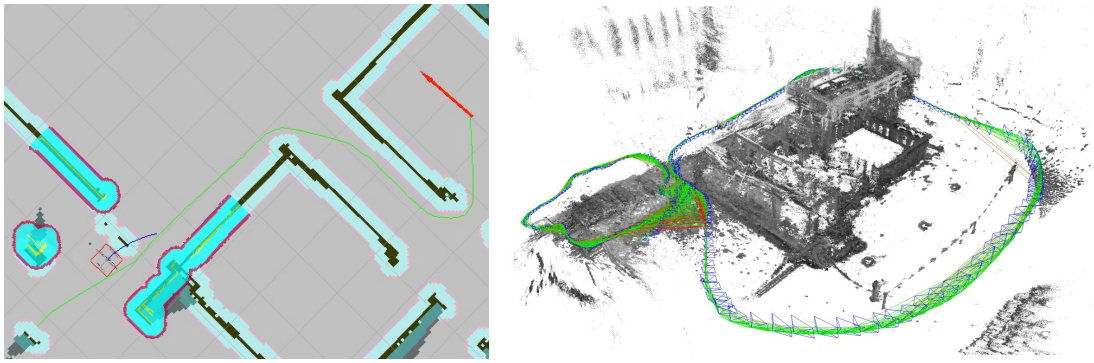
características na imagem, realiza encaixe da nuvem de pontos, estima a localização do robô, faz o registro da nuvem de pontos com a imagem e voxeliza utilizando o OctoMap (pacote de voxelização de nuvem de pontos, figura 7b) para criar um mapa de ocupação. Em (FANKHAUSER et al., 2014) é criado um mapa de elevação centrado no robô (mapa local), cada voxel do mapa possui característica de incerteza, permitindo que robôs quadrúpedes prefiram pisar em regiões com maior certeza sobre a elevação.

(a) (b)

Figura 7: Métodos de representação 3D utilizando cor. 7a: Uma nuvem de pontos com informação de cor. Retirado de (CHEN et al., 2014). 7b: Mapa octomap utilizando informação de cor. Retirado de (ENDRES et al., 2012).

Localização e mapeamento simultâneo (SLAM) é um problema clássico da área de robótica móvel. Sabe-se que é impossível solucioná-lo completamente e sempre será necessária a comparação com outros métodos para definir um limiar que caracterize uma técnica SLAM como 'boa o suficiente'. No entanto, existem diversas técnicas que propõem melhorar a precisão da localização no problema de SLAM, como o filtro de Kalman ou o método de Monte Carlo. Estes métodos possuem algo em comum: ficam mais precisos quando se têm mais informações sobre o ambiente (pontos de medição). A redundância aumenta a acurácia (através do processo de fusão de sensores) e permite uma identificação mais precisa de pontos de interesse. Por causa deste efeito, desenvolvedores de robôs móveis podem apostar não apenas na qualidade como também na quantidade dos sensores integrados à estrutura dos robôs. Um número maior de sensores diferentes também ajuda a cobrir pontos cegos do robô e permite a aplicação do SLAM em ambientes que desfavorecem certos tipos de sensores, como por exemplo lugares escuros ou com muito reflexo para uma determinada faixa do espectro luminoso.

Na área de SLAM (*Simultaneous Localization and Mapping*), tem-se a pesquisa feita em (FUENTES-PACHECO et al., 2015) que compara diversos métodos de SLAM visual (utilizando apenas imagens). Em (HENRY et al., 2014) os autores tentam desenvolver um



(a) (b)

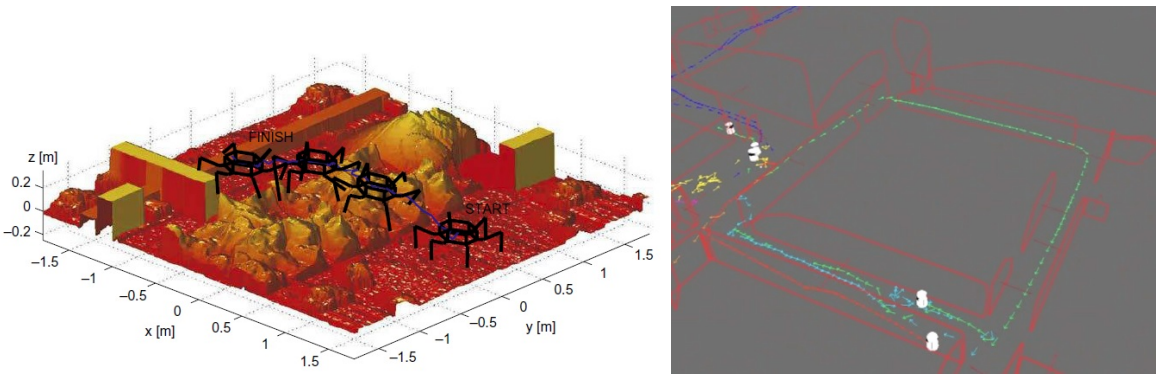
Figura 8: Métodos de mapeamento e SLAM em robótica. 8a: SLAM com mapa Costmap2D retirado de (SELBY, 2016). 8b: SLAM utilizando mapeamento 3D, retirado de (ENGEL et al., 2014).

método prático para SLAM RGB-D com um Microsoft Kinect 1.0 utilizando apenas métodos visuais ou espaciais, chegando à conclusão que para alcançar uma precisão similar a sensores no estado da arte é necessária uma otimização exaustiva. Outro trabalho envolvendo o Kinect pode ser encontrado em (LEE et al., 2012), onde os autores utilizam um processo de SLAM RGB-D com o auxílio de uma GPU para acelerar o processamento, alcançando 22Hz de taxa de processamento de aquisições. Em (MUR-ARTAL et al., 2015) os autores desenvolvem um método inovador de SLAM monocular com relocalização (*loop closure*), afirmando atingir processamento em tempo real para o cálculo de trajetória. Em (SCHERER; ZELL, 2013), um método SLAM RGB-D para veículos aéreos com processamento embarcado em tempo real é desenvolvido. Em (TAGUCHI et al., 2013), os autores desenvolvem um método SLAM RGB-D orientado a sensores portáteis de mão, utilizando pontos e planos como figuras primitivas para a construção do mapa. Por fim, em (WEIKERSDORFER et al., 2014) os autores utilizam uma Asus Xtion (RGB-D) junto com um sensor de visão dinâmica (DVS) para a geração do mapa, utilizando os eventos gerados pelo DVS para focar o processamento da informação 3D capturada pela câmera.

Desenvolver métodos de SLAM a partir do mapeamento proposto não é o foco do trabalho desta dissertação, porém é algo para se ter em mente no desenvolvimento de qualquer sistema de mapeamento. A construção de SLAM para o trabalho aqui descrito é uma das propostas para trabalhos futuros.

Na área de métodos de mapeamento não-tradicionais é possível encontrar vários trabalhos orientados a vários ambientes e métodos de mapeamento inovadores. No trabalho de (BELTER; SKRZYPCZYNSKI, 2013) os autores conseguem aplicar um algoritmo visual de localização em ambientes sem estrutura para construir e realizar navegação (figura 9a).

O algoritmo de mapeamento e rastreamento (sobre o sensor LIDAR e a câmera RGB) em conjunto com a IMU do robô permitir calcular a odometria, e assim possibilitando a navegação de um robô de seis pernas sobre um terreno acidentado. A navegação e computação apesar de bastante lenta consegue determinar a próxima posição para cada passo do robô através do mapa construído. Em (BORRMANN et al., 2014) os autores utilizam um *laser scanner* geográfico para capturar o interior de ambientes, fundindo os dados lidos com informação de cor proveniente de uma câmera RGB e dados de temperatura através de uma câmera térmica. Com esses dados um mapa de voxels é criado (o escaneamento dura entre 6 a 55 segundos com o robô parado dependendo da resolução dos voxels). O objetivo dos autores é possibilitar uma detecção rápida de falhas em sistemas de controle de temperatura de ambiente para uma construção com vários cômodos. No trabalho de (NIETO-GRANDA et al., 2014) os autores desenvolvem uma estratégia de coordenação de múltiplos robôs, casada com um novo método de mapeamento 3D utilizando contornos de planos (figura 9b). Neste trabalho, cada robô possui uma câmera 3D capaz de capturar nuvens de pontos. Através da nuvem de pontos coletada, planos espaciais são encontrados e seus contornos definidos por cada robô. Os robôs compartilham os planos espaciais encontrados ao invés de uma nuvem de pontos, tornando a comunicação mais eficiente. Além dos planos, um mapa de ocupação 2D é montado e compartilhado, permitindo com que, através da odometria de cada robô, cada um saiba a posição dos outros robôs da equipe. O algoritmo desenvolvido promove a exploração assinalando nos mapas de ocupação uma camada de 'fronteira' que representa os lugares que não foram explorados. Uma validação com nove robôs TurtleBots (robôs de baixo custo com *software* aberto) é feita para explorar o andar de uma universidade. Em (MAIER et al., 2012) é desenvolvido um método de mapeamento e navegação baseado em CAD. Orientado a ambientes conhecidos, um mapa construído em CAD é utilizado para definir um ambiente apenas com as partes estáticas. Então, os autores utilizam um robô humanoide equipado com um sensor Asus Xtion para navegar por este ambiente utilizando um planejamento de trajetória baseado no mapa CAD. O ambiente real, além dos obstáculos imóveis, possui alguns objetos e obstáculos móveis que não estavam representados no ambiente CAD. Um algoritmo de voxelização estilo Octomap é utilizado para fazer o registro de novos obstáculos lidos que estavam sem representação no mapa CAD, permitindo ao robô consertar sua trajetória desviando dos objetos espalhados pelo mapa.



(a) (b)

Figura 9: Métodos de mapeamento não-tradicionais. 9a: Mapa de superfície e planejamento de trajetória a partir de imagens. Retirado de (BELTER; SKRZYPCZYNSKI, 2013). 9b: Método de representação de mapa utilizando contorno de planos compartilhado entre múltiplos robôs. Retirado de (NIETO-GRANDA et al., 2014).

A descrição do desenvolvimento é iniciada na próxima seção começando pelo protótipo do sensor desenvolvido.

3 PROTÓTIPO DO SENSOR

O desenvolvimento do projeto envolve o projeto mecânico, elétrico, eletrônico e de software envolvido na construção do sensor omnidirecional, sendo dividido em protótipo do sensor e software de mapeamento. Esta seção detalha o desenvolvimento do protótipo.

Para a realização do mapeamento omnidirecional, utilizou-se um sensor LIDAR rotacionando em cima de uma plataforma. O sensor LIDAR foi escolhido pois é a classe de sensores com maior ângulo de visão. Apesar de coletar poucos dados por leitura (511 pontos por leitura) e ter uma frequência de aquisição baixa (10 aquisições por segundo), o ângulo de leitura na grande maioria dos sensores LIDAR é maior que uma câmera 3D de médio custo, seja ela de tempo de voo ou de luz estruturada. Como o objetivo do sensor é possibilitar um mapa omnidirecional do ambiente sem a necessidade de manobrar o robô, foi decidido que a frequência de atualização dos dados do mapa poderia ser sacrificada em favor de uma cobertura total do ambiente.

A plataforma do sensor LIDAR por sua vez deve girar a partir de um servomotor de corrente contínua, para permitir um movimento vai-vem em uma esfera completo. Outra opção para realizar a cobertura total do ambiente seria utilizar um conector USB rotativo para manter o giro, porém houve a preocupação com a perda de dados e robustez da solução que inviabilizou o método.

O protótipo inclui também uma câmera RGB. Para a aquisição omnidirecional foi decidido utilizar um espelho hiperbólico. Outra opção seria colocar a câmera afixada junto à plataforma do sensor 3D, que também foi inviabilizada pela perda de robustez.

Como o protótipo foi desenvolvido antes dos testes e desenvolvimento do software de mapeamento, não se sabia quanto processamento seria necessário, uma vez que a proposta incluía processamento de nuvem de pontos, registro em mapa e processamento de imagem. Para garantir que não fosse necessária a reconstrução do protótipo no futuro, foi decidido incluir também um kit de desenvolvimento com processamento GPU caso se fizesse preciso.

Além de todos os componentes previamente descritos, também estão incluídas fontes de alimentação que cobrem seus requisitos elétricos. Podem ser encontradas mais informações referentes ao projeto elétrico na seção 3.2. A lista total de componentes envolvidos no protótipo é encontrada na tabela 1. O protótipo completo pode ser visualizado na figura 10.

Componente	Localização
Sensor LIDAR Hokuyo URG-04LX	Motor
Servo motor DC *brushless* Maxonmotor EC-32	Motor
Câmera RGB Pixart Imaging Snake Eye	Câmera
Espelho hiperbólico GoPano	Câmera
GPU/CPU embarcado Nvidia Jetson Tegra K1	Base
Fonte DC Mean Well NES-15-5	Base
Fonte DC Mean Well S-201-24	Base
Controlador Maxonmotor EPOS2 24/5	Base
Perfis de alumínio	Estrutura
Peças suporte projetadas impressas 3D	Estrutura

Tabela 1: Lista de materiais que compõem o sensor.

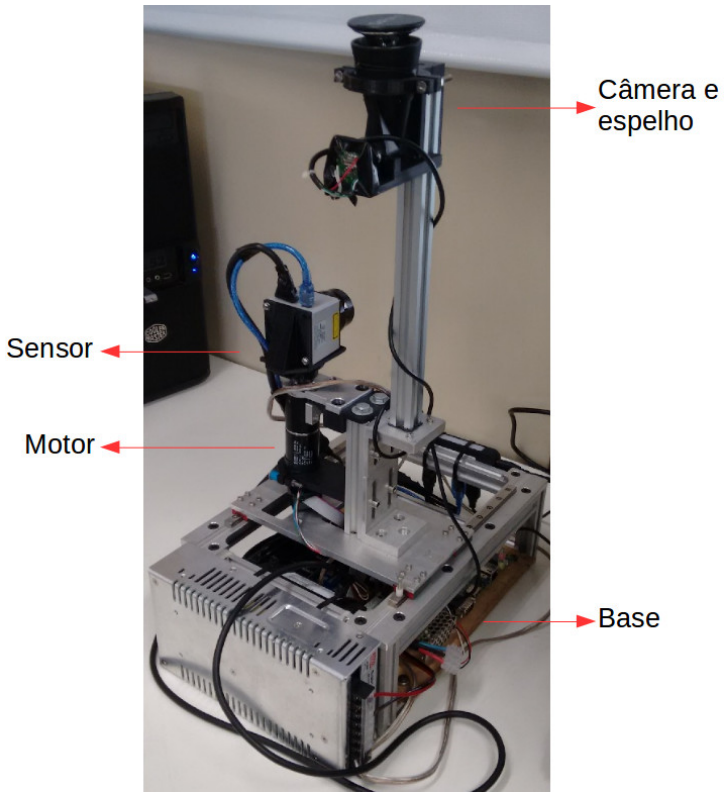


Figura 10: Estrutura do protótipo do sensor desenvolvido.

O sensor LIDAR Hokuyo URG-04LX foi escolhido como sensor principal pois é um sistema popular muito comum na área da robótica. O 04LX é um sensor simples com frequência de atualização baixa, apenas 10 aquisições de 511 pontos por segundo, o que permite validar a operação da estratégia de mapeamento para sensores limitados em frequência. Possui

uma margem de erro de aproximadamente 3%, e um alcance de 5 metros. Comparado a outros sensores LIDAR, como por exemplo o SICK LMS500 (SICK, 2018), o Hokuyo 04LX possui desempenho muito abaixo da média, sendo utilizado principalmente para algoritmos de localização e mapeamento 2D simultâneo em robôs pequenos.

O servo motor *brushless* Maxon Motor EC-32 é responsável por rotacionar o sensor. É um motor de alta potência e torque e larga faixa de velocidade de operação. O motor está equipado com uma caixa de redução de 79:1. Além disso, um codificador de alta resolução (3 canais, 500 contagens por giro) consegue informar ao controlador a posição precisa. O servo motor é conectado a um controlador EPOS2 24/5 calibrado no software EPOS Studio disponibilizado pela própria empresa (Maxon Motor). O servo motor, seus componentes e o controlador foram escolhidos pela disponibilidade e capacidade de cobrir os requisitos do projeto do protótipo.

A câmera RGB Snake Eye da Pixart Imaging foi acoplada ao protótipo para realizar a aquisição de imagens omnidirecionais. É uma câmera estilo webcam, de 480x640 pixels de resolução a 30 quadros por segundo, escolhida pela disponibilidade. O invólucro protetor da câmera foi removido para afixar ao espelho hiperbólico GoPano de modo que nenhuma luz adentre a câmera sem que venha diretamente pelo espelho.

Por fim, o kit de desenvolvimento Nvidia Tegra K1 (TK1) é uma placa de desenvolvimento de pequeno porte orientada a processamento CPU e GPU embarcado equipada com um processador ARM Cortex A15 e GPU Nvidia Kepler de 2GB e 192 núcleos CUDA. O kit também está presente no protótipo para caso fosse necessário realizar pré-processamento paralelo dos dados adquiridos pelo sensor e pela câmera.

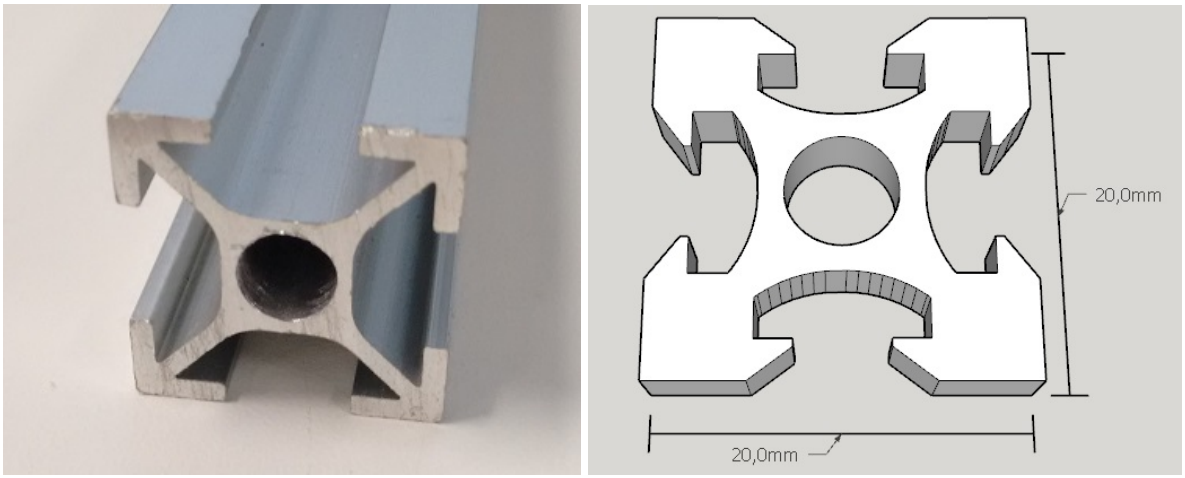
A próxima seção detalha o desenvolvimento do sensor omnidirecional, iniciando pela parte mecânica do protótipo.

3.1 PROJETO MECÂNICO

O sensor omnidirecional tem como componentes principais o sensor LIDAR Hokuyo URG-04LX, um motor *brushless* Maxon Motor EC-32 80 watts, uma câmera RGB modelo Snake Eye Webcam da Pixart Imaging e uma placa de desenvolvimento Nvidia Tegra K1 (TK1). O motor tem a função de girar o Hokuyo de modo a cobrir 360 graus de varredura. A câmera também precisa capturar imagens em 360 graus, porém por ser uma imagem sem requisitos de altas resoluções, existe a opção de se utilizar um espelho hiperbólico para espalhar a captura em 360 graus, removendo a necessidade de rotacioná-la. A TK1 precisa estar embarcada ao projeto

sem requisitos de posição.

Foi construída para montagem dos componentes uma estrutura de suporte utilizando perfis de alumínio. Perfis de alumínio são estruturas padrão de suporte com formato otimizado para encaixe de diversas peças, criados a partir da extrusão de uma seção transversal fixa. Deste modo, é possível ter um maior controle sobre o tamanho da peça desejada, mesmo após produção, através de ferramentas simples. Outro fator que influenciou nesta escolha foi o fato de alumínio ser um metal leve porém resistente, permitindo a criação de uma estrutura suporte que não se deforme nem seja muito pesada a ponto de dificultar o seu transporte. O perfil padrão utilizado foi 20x20 (milímetros), que também é o mais produzido. A seção transversal utilizada foi o padrão europeu, que pode ser visualizada na figura 11.



(a) (b)

Figura 11: 11a: Perfil de alumínio 2020 utilizado. 11b: Perfil de alumínio 2020 no software SketchUp.

Uma estrutura com os vértices de uma caixa de 30cm x 23cm foi montada utilizando os perfis de alumínio, capaz de comportar todos os componentes. A caixa possui apenas uma base de madeira para fixação dos componentes mas não é coberta dos lados para facilitar a manutenção e operação. Os componentes necessários para o projeto e suas dimensões físicas estão listados na tabela 2.

Na base da caixa estão fixados a placa de desenvolvimento Nvidia TK1, o controlador Maxon Motor EPOS2 24/5 e a fonte conversora Mean Well NES15-5 24V/5V. Nas laterais da caixa estão fixadas através de abraçadeiras a fonte 110 240V/24V Mean Well S-201-24 e a fonte de alimentação da placa de desenvolvimento Nvidia TK1. Em cima da caixa se encontra uma base pequena de alumínio afixada, com tamanho suficiente para fixação do motor, Hokuyo e câmera. Perpendicularmente a esta base estão afixados sobre ela uma estrutura de alumínio de tamanho similar ao motor. O motor se encontra preso a esta estrutura com o auxílio de duas

Componente	Modelo	Tamanho	Localização
Sensor LIDAR	Hokuyo URG-04LX	4.8cm x 4.8cm x 7cm	Motor
Servo motor	Maxonmotor EC-32	11.9cm x 3.2cm x 3.2cm	Motor
Câmera RGB	Pixart Imaging Snake Eye	2.8cm x 3cm x 2.6cm	Câmera
Espelho hiperbólico	GoPano	4cm x 4cm x 7.6cm	Câmera
GPU/CPU embarcado	Nvidia Jetson Tegra K1	12.7cm x 12.7cm x 3cm	Base
Fonte DC	Mean Well NES-15-5	7.9cm x 5.1cm x 2.8cm	Base
Fonte DC	Mean Well S-201-24	21.6cm x 11.5cm x 4.8cm	Base
Controlador	Maxonmotor EPOS2 24/5	10.5cm x 7.6cm x 2.3cm	Base

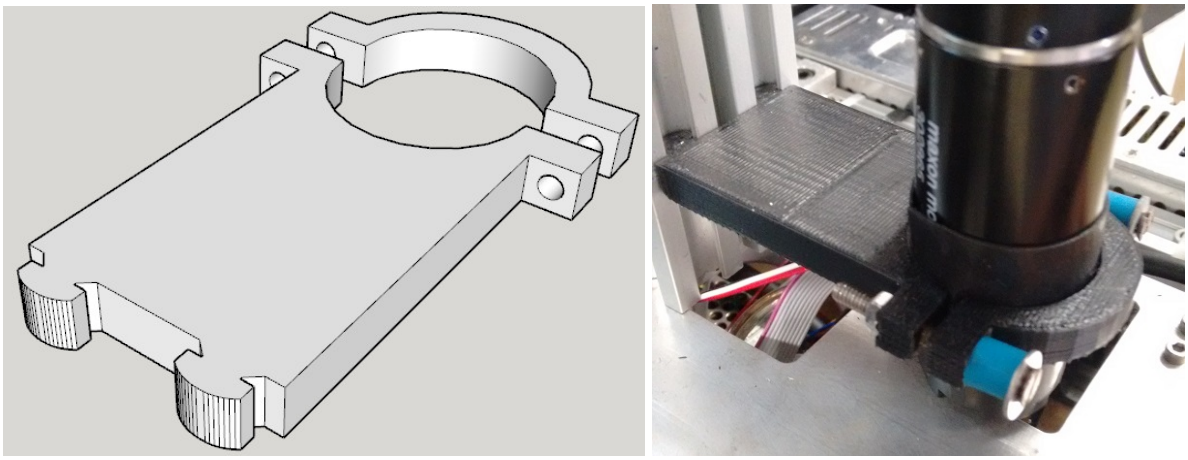
Tabela 2: Tabela de componentes do projeto e suas dimensões

peças de plástico modeladas especialmente para este projeto, criadas a partir de uma impressora 3D. O motor está orientado diretamente para cima, de modo a rotacionar horizontalmente. Acima da estrutura suporte do motor está presa mais uma haste de alumínio para segurar a câmera RGB com o espelho hiperbólico, a afixação também foi feita através de uma peça criada a partir de uma impressora 3D.

Durante a montagem do protótipo, vários componentes utilizados não possuíam um método simples de fixação. O motor possui um perfil cilíndrico sem nenhuma base com suporte para parafuso. A câmera RGB e o espelho hiperbólico também não possuem um método simples de fixação (parafusos ou encaixes). Uma peça customizada também era necessária para o acoplamento entre o sensor Hokuyo e o motor, uma vez que o sensor precisa rotacionar com o centro de massa diretamente sobre o eixo de rotação do motor para minimizar a força total no sensor.

Para solucionar estes problemas, as peças necessárias foram projetadas para serem impressas por uma impressora 3D. A modelagem foi realizada através do software SketchUp, distribuído pela Trimble Navigation. SketchUp é um software de modelagem 3D que contém diversas ferramentas orientadas a construção fácil de modelos 3D, capaz de suportar vários formatos de modelos tridimensionais. Para realizar a conversão para STL (formato de entrada da impressora 3D) se encontra disponível na Extension Warehouse do próprio Sketch Up uma extensão que permite a conversão.

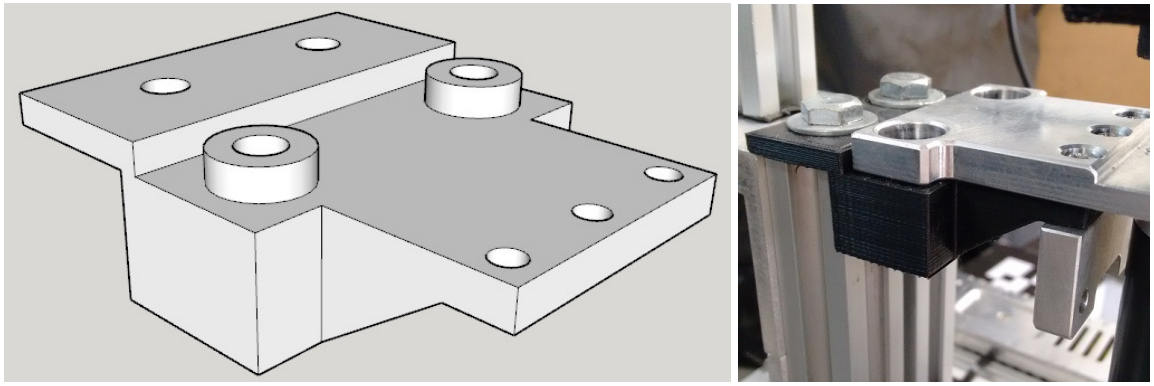
Para aproveitar as características modulares dos perfis de alumínio, a peça utilizada para segurar a base do motor para impedir vibrações e garantir o alinhamento vertical foi modelada com encaixes padrões. A peça segura o motor (que possui perfil cilíndrico) abraçando-o e prendendo com dois parafusos. Uma imagem da peça encontra-se na figura 12.



(a) (b)

Figura 12: 12a: Modelo da peça que abraça o motor. 12b: Peça que abraça o motor impressa e montada no projeto.

O motor possui uma peça de alumínio que prende em sua extremidade de atuação, de modo a facilitar sua fixação. Porém as dimensões desta peça não são orientadas a fixação em um perfil de alumínio. Portanto surgiu a necessidade de outra peça que permitisse a fixação das hastes de alumínio na peça que segura o motor. A peça modelada possui encaixes para ambos os pontos de fixação assim como buracos para a utilização de parafusos. A peça pode ser visualizada na figura 13.

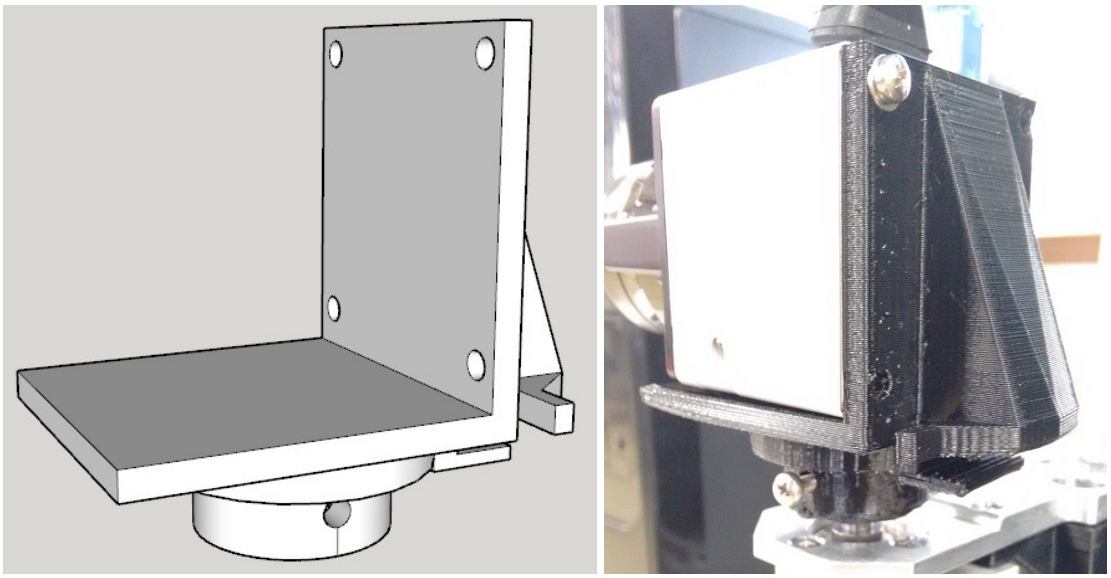


(a) (b)

Figura 13: 13a: Modelo da peça que fixa a extremidade de atuação do motor. 13b: Peça que fixa o motor impressa e montada no projeto.

O eixo do motor possui perfil cilíndrico exceto por uma face plana para fixação. O sensor deve estar montado em cima do eixo do motor, porém não há nenhum encaixe entre eles. Para resolver este problema, uma peça foi modelada de modo a prender o Hokuyo ao eixo do motor. A peça possui suporte para as duas orientações do sensor e espaço para prender os cabos de transmissão e alimentação. Na base da peça foi feito um buraco de parafuso para permitir fixação à face plana do eixo do motor. Como não é possível criar uma rosca interna na peça,

um buraco foi feito na sua base para a inserção de uma porca, que foi posicionada de modo a alinhá-la com o parafuso. O buraco da porca foi fechado com plástico derretido para que ela não se mova. Deste modo o parafuso consegue prender com segurança ao eixo do motor sem deslizar. A peça pode ser visualizada na figura 14.

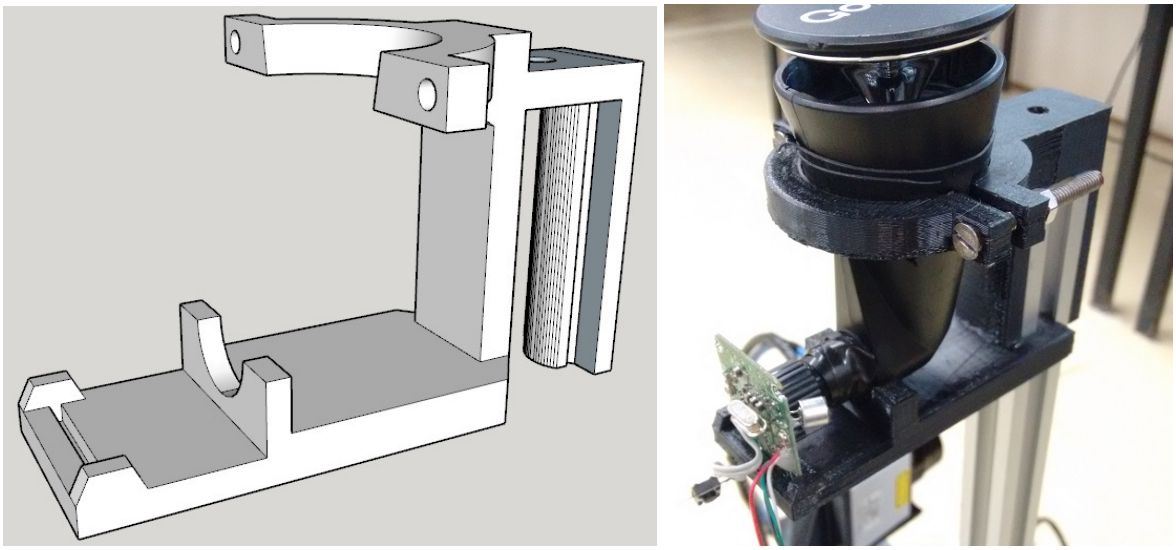


(a) (b)

Figura 14: 14a: Modelo da peça que prende o Hokuyo no eixo do motor. 14b: Peça impressa e montada no projeto sustentando o Hokuyo ao eixo do motor.

A câmera e o espelho hiperbólico também não possuem nenhum método para suporte. Uma peça foi modelada para fixá-los à haste de alumínio suporte. A peça possui um encaixe no formato da câmera numa distância calculada para ficar exatamente apontada para o espelho. O espelho possui um perfil em cone, o que dificulta a fixação. A solução foi fazer a peça de modo que abrace o sensor com o auxílio de parafusos para prendê-lo de forma a não vibrar ou escorregar. A peça encaixa diretamente no perfil de alumínio. Uma imagem da peça pode ser encontrada na figura 15.

Com o auxílio das peças projetadas foi possível afixar todos os componentes que formam o sensor omnidirecional, do modo apresentado na figura 10. Porém, antes de fixar o sensor Hokuyo e seus cabos no motor, foi necessário garantir que o motor não vá além do curso permitido pelos cabos e acabe danificando o sensor. Para isso alguns ensaios preliminares foram feitos sem os cabos conectados ao sensor. O motor foi calibrado segundo suas especificações disponíveis em (MAXONMOTOR, 2018a) utilizando o software EPOS Studio. O motor foi configurado para controle por ângulo enquanto um pacote ROS foi desenvolvido para enviar os comandos ao motor. Dentro do pacote foram limitados o ângulo mínimo e ângulo máximo baseados nas medições feitas durante os ensaios. Uma descrição mais aprofundada sobre o



(a) (b)

Figura 15: 15a: Modelo da peça que suporta a câmera e o espelho hiperbólico. 15b: Peça da câmera após impressa no projeto.

software de controle de motor é encontrada na seção 3.3.

3.2 PROJETO ELÉTRICO E ELETRÔNICO

Para o funcionamento do protótipo do sensor, um projeto elétrico foi necessário. A estrutura elétrica do sensor é muito simples, com poucos periféricos precisando de alimentação fora da faixa dos 100-240V, porém alguns periféricos precisam de uma alimentação segura sem interrupção para garantir o funcionamento e evitar danos aos componentes.

Como o protótipo tem apenas o objetivo de realizar ensaios e testes para validação do método de mapeamento, não houve preocupação com o superdimensionamento do projeto elétrico nem em reduzir o espaço ocupado pelos componentes. Os componentes que requerem alimentação bem como a potência respectiva necessária estão descritos na tabela 3.

Componente	Tensão	Corrente	Potência	Localização
Sensor LIDAR Hokuyo URG-04LX	5V	500mA	2.5W	Motor
GPU/CPU embarcado Nvidia Jetson Tegra K1	12V	1.5A	18W	Base
Controlador Maxonmotor EPOS2 24/5	15V	5.5A max	80W	Base

Tabela 3: Tabela de componentes elétricos e alimentação necessária.

O Hokuyo URG-04LX, apesar dos 5V de tensão de entrada, não pode ser alimentado apenas por conexão USB, uma vez que sua porta USB 2.0 possui potência máxima de 1.5W, sendo necessário utilizar os pinos de sua entrada serial para alimentação. Por este motivo, para

a alimentação do sensor uma fonte Mean Well NES-15-5 foi escolhida. Esta fonte possui tensão de entrada de 100-240V, tensão de saída de 5V e é capaz de suprir 15W de potência, suficiente para alimentar o sensor.

O controlador Maxonmotor EPOS2 24/5 fornece a potência necessária ao motor Maxonmotor EC-32, e requer uma alimentação constante a fim de não causar danos ao motor. Para dar conta da alta potência exigida, foi utilizada uma fonte Mean Well S-201-24 regulada para 15V. Esta fonte possui tensão de entrada 100-240V e uma tensão de saída regulável entre 10-30V e capacidade de prover 200W de potência. A regulação em 15V implica uma corrente máxima de aproximadamente 13A, suficiente para manter o motor em uma alimentação estável e segura. O kit de desenvolvimento Nvidia TK1 possui uma fonte de alimentação própria com entrada 100-240V.

O projeto eletrônico envolve realizar a conexão dos componentes do sensor a fim de garantir a comunicação entre eles. Os componentes a serem interfaceados são os mesmos do projeto elétrico.

O controlador EPOS2 24/5 é uma unidade de controle de posição digital, além de transmitir ao motor a energia necessária, interpreta comandos dados pelo computador através da porta USB e realiza o controle de posição do motor de acordo. Como o controlador é orientado ao motor e codificador, as portas e cabos já estavam disponíveis. Do motor, além dos cabos de alimentação, saem também cabos do sensor Hall e do codificador, responsáveis pelos dados necessários para o controle do motor. Para o funcionamento correto do motor, este deve ser calibrado. A etapa de calibração é discutida na seção sobre movimentos do sensor (seção 3.3).

Dois barramentos de dados são necessários para a conexão do sensor Hokuyo. O cabo USB que é conectado ao computador, e a porta serial, que neste projeto serve apenas para prover a alimentação, não havendo troca de dados. Para que a troca de dados entre estes dois componentes seja eficiente, as informações devem ser imediatamente encaminhadas pelo *framework* ROS, no qual é desenvolvido todo o método de mapeamento. Por serem componentes comuns na área da robótica, ambos possuem pacotes prontos desenvolvidos pela comunidade orientado diretamente ao interfaceamento e padronização dos dados recebidos e enviados.

Sensores Hokuyo contam com duas opções de pacotes à sua disposição para a interpretação dos dados lidos pelos sensores, o 'URG node' e o 'Hokuyo node'. Ambos possuem o mesmo objetivo: procurar a porta USB onde o sensor Hokuyo está conectado, estabelecer conexão, receber os dados do sensor e informá-los através de um tópico ROS. A diferença entre os pacotes são as versões do protocolo de comunicação que cada um suporta.

No caso do sensor utilizado no projeto (Hokuyo URG-04LX) ambos os pacotes são capazes de realizar o interfaceamento.

O controlador EPOS2 24/5 possui interface USB e CAN (MAXONMOTOR, 2018b). A interface CAN possui uma velocidade de transmissão de 1Mbit/s até 40 metros de comprimento de barramento. A interface USB possui velocidade máxima de 48Mbit/s até 5 metros. Como não há interesse em utilizar cabos para longas distâncias no protótipo, a comunicação USB foi escolhida. Para a interface USB ROS do controlador, existe um pacote mantido pela comunidade. O pacote 'EPOS hardware' realiza a integração entre controlador e computador, possuindo a biblioteca de comandos EPOS integrada dando toda a funcionalidade necessária.

O kit de desenvolvimento Nvidia Tegra K1 também foi preparado para uso sendo integrado ao protótipo. Para que seja possível o uso de ROS com uso da GPU da placa de desenvolvimento é necessário realizar uma formatação do sistema operacional utilizando o software JetPack disponibilizado pela própria Nvidia, instalação do ROS e dos pacotes necessários do projeto, e outras configurações adicionais. Todo o detalhamento deste processo é descrito em um capítulo de livro publicado como um dos resultados dessa dissertação (DALMEDICO et al., 2019).

3.3 MOVIMENTOS

O sensor realiza a leitura omnidirecional do ambiente, utilizando o movimento giratório de um sensor LIDAR para realizar um escaneamento tridimensional. O sensor pode rotacionar em torno de 3 eixos: '*roll*' (rolamento), '*pitch*' (arfagem) e '*yaw*' (guinada). Sabendo que o sensor possui um modo de leitura horizontal linear (511 pontos distribuídos em um arco de 240 graus em volta do sensor) girar o sensor no eixo de rolamento geraria um domo na frente do robô que não seria omnidirecional. Rotacionar em torno do eixo de arfagem permite uma leitura omnidirecional porém a quantidade de informação adquirida não seria a mesma para todos os ângulos devido ao limite de 511 pontos. Rotacionar o sensor no eixo de guinada com o sensor de lado (tombado 90 graus) faz com que o motor defina o ângulo de aquisição, e o espalhamento do sensor fique na vertical, escaneando o ambiente com a rotação do motor de modo igual para qualquer ângulo em volta do robô.

O protótipo do sensor se baseia na rotação do sensor LIDAR. Como a estabilidade da conexão com o sensor foi valorizada, decidiu-se por não empregar um conector USB rotativo. Por este motivo, a base rotativa deve fazer um movimento vai-vem para não enrolar os cabos do

sensor. Qualquer movimento além do esperado pode danificar o sensor e os cabos que levam, e em casos extremos pode danificar também o motor e seu controlador.

O primeiro passo para garantir que o motor funcione corretamente é a configuração e calibração. A configuração do motor é feita no software EPOS Studio (figura 16), disponibilizado pela Maxonmotor. Os parâmetros e *datasheets* do motor (código 118889), do codificador (código 110514) e da caixa de redução (código 110514) utilizados podem ser consultados em (MAXON, 2018). Estes parâmetros de funcionamento precisam ser informados ao software EPOS Studio, que realiza testes e faz a calibração do motor. O perfil de atuação é armazenado dentro do controlador e o motor estará pronto para ser utilizado.

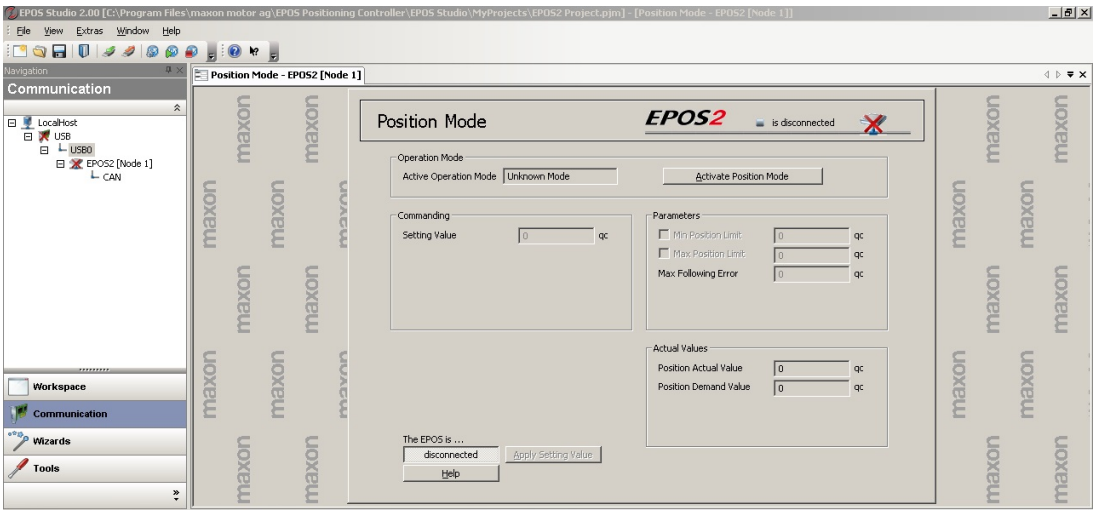


Figura 16: Tela do software EPOS Studio, utilizado na calibração do motor.

O pacote ROS que realiza comunicação com o servo motor, o EPOS Hardware, também precisa de parâmetros para que funcione corretamente. O pacote possui dois arquivos de configuração com extensões URDF e YAML que são carregados durante a inicialização do pacote. O arquivo URDF informa as transmissões do motor. No caso do projeto, os únicos parâmetros a serem alterados no arquivo URDF são a magnitude da redução e o nome do motor que corresponde ao nome informado no arquivo YAML. A magnitude da redução representa o passo que será informado quando a posição do motor for consultada. Para simplificar cálculos futuros, foi decidido que o motor deveria reportar sua posição em graus. Para o cálculo do valor a ser carregado como redução, toma-se o número total de passos por giro do motor sem redução, 2000, e multiplica-se pela redução dada pela caixa de redução do motor, que é 79, totalizando 158000 passos por giro completo. Como é desejado o valor em graus, divide-se o número de passos por giro por 360, resultando em 438.8889, número de passos que representam um grau. É este valor que deve ser informado como magnitude de redução para que o pacote reporte em graus. No arquivo YAML, basta carregar os parâmetros dados pelo *datasheet* do motor e

seus componentes, os mesmos utilizados para calibração do motor. Além destes parâmetros o arquivo YAML pede também o código serial do motor. Este serial é encontrado utilizando um executável do próprio pacote EPOS Hardware, chamado de '*list devices*'. Este executável procura motores conectados em portas USB do computador e reporta o código serial e porta USB de cada controlador EPOS encontrado.

O pacote Epos Hardware possui suporte aos perfis de velocidade e posição dos motores Maxonmotor. Neste trabalho o perfil de posição é utilizado uma vez que é utilizado movimento vai-vem. O pacote permite definir a velocidade máxima para o perfil de posição, bem como aceleração e de desaceleração (expressos em passos, desconsiderando a redução). Para os testes foi definido uma velocidade máxima de 300 passos por segundo, com aceleração e desaceleração de 1000 passos por segundo, o que representa aproximadamente 6.67 segundos para uma volta completa.

Além destes parâmetros, um pacote ROS foi criado para definir como o motor se comporta. O algoritmo em si é discutido em mais detalhes na seção 4. O movimento do motor depende do método de mapeamento sendo utilizado (omnidirecional ou janela deslizante). No caso do mapeamento omnidirecional, o motor faz um movimento vai-vem girando o sensor entre dois ângulos definidos em parâmetros de configuração do algoritmo de mapeamento, tipicamente zero e 360 graus. Para o método de janela deslizante, o algoritmo define o ângulo máximo e mínimo alcançado pela representação da janela, e alterna entre os dois ângulos. A próxima seção detalha melhor todo o algoritmo desenvolvido para os métodos de mapeamento.

4 MAPEAMENTO

Esta seção detalha o desenvolvimento do algoritmo de mapeamento para o sensor proposto, com o objetivo de proporcionar um método de registro de informação tridimensional enquanto tem-se em mente a redução de memória ocupada. O algoritmo também visa permitir que o sensor desenvolvido crie um mapa independente, pelo método omnidirecional de escaneamento e pelo método de janela deslizante. O escaneamento omnidirecional tem o objetivo de cobrir pontos cegos de outros sensores existentes no robô, e é indicado quando o robô possui outros sensores além do LIDAR rotacional. O método de janela deslizante é indicado quando o robô possui apenas o sensor LIDAR ou quando os outros sensores disponíveis possuem um campo de visão limitado e se faz necessário uma visão periférica, definida por se concentrar nas áreas laterais do robô e ter uma frequência de atualização menor que a visão principal.

Inicialmente, foi utilizada uma abordagem de lista dupla de estruturas para a representação das leituras e da superfície gerada. A proposta era guardar os pontos lidos pelo sensor separados por aquisição e entre duas aquisições criar uma estrutura 'mapa' a partir de uma triangularização Delaunay (CAO et al., 2014) que representasse a superfície resultante. Junto a isso, haveria também um método para navegar pela lista de estruturas, podendo retornar uma nuvem de pontos a partir das estruturas de leitura, ou uma superfície a partir caso dos triângulos gerados. A representação seria feita com 'markers' (marcadores), formas geométricas primitivas disponíveis para uso no 'rviz' (visualizador gráfico padrão de mensagens do ROS). Vários problemas foram encontrados com essa abordagem, porém o maior foi a memória ocupada. Quanto mais pontos fossem lidos, mais triângulos seriam gerados e isso eventualmente inviabilizaria o método. Limitar a quantidade de pontos poderia remover o aspecto omnidirecional ou dificultar o encaixe de novas leituras. A solução foi encontrar uma outra abordagem para otimizar um mapeamento por varredura.

A solução encontrada foi desenvolver o mapa baseado em vários 'grids' (grade) de elevação. Neste caso, uma grade representaria um plano dividido em intervalos discretos, a nomenclatura 'grid' será utilizada no restante do documento para essa definição. Para a

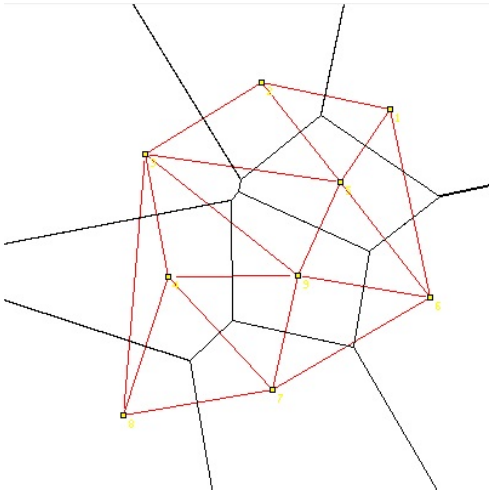


Figura 17: Exemplo de uma triangularização Delaunay, retirada de (SCHINDELIN, 2008).

representação dentro do ROS foi utilizado o pacote 'Grid Map' (FANKHAUSER; HUTTER, 2016). O pacote Grid Map cria um novo tipo de mensagem homônima do ROS, usada para representar superfícies orientadas à construção de mapas. Os grids (figura 18) são interpretados como uma matriz, onde cada valor representa a elevação do mapa referente à posição da matriz. Existe suporte para adição de 'layers' (camadas) no mapa para representação de dados secundários, como cor (RGB), confiabilidade ou erro. O pacote inclui várias ferramentas de suporte ao Grid Map, como visualização no Rviz (visualizador gráfico padrão de mensagens do ROS), conversão para mapa de ocupação (formato padrão de representação de mapas 2D) e filtros.

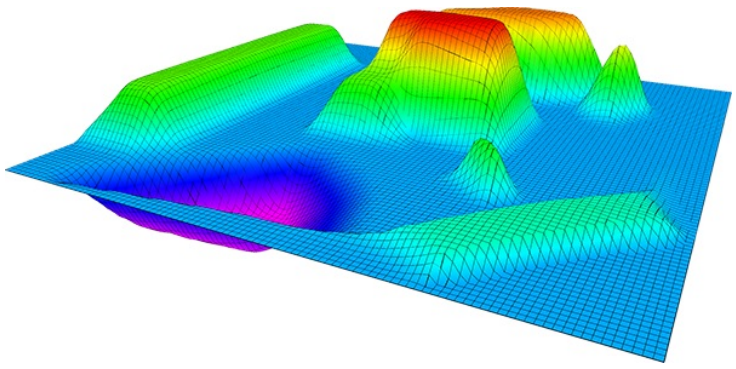


Figura 18: Exemplo de um grid construído pelo pacote 'grid map'.

Uma estratégia comum em mapeamento 2D é empregar dois mapas: um mapa local em alta resolução, que acompanha o robô e possui informação precisa de seus arredores; e um mapa global fixo que registra as informações do mapa local. Deste modo o robô possui uma visão mais detalhada do ambiente e pode realizar manobras minuciosas sem precisar sacrificar memória para armazenar um ambiente inteiro em grande nível de detalhes. Essa estratégia é utilizada nos métodos de mapeamento 3D propostos neste trabalho com uma diferença: a

utilização de dois mapas globais, um representando o chão onde o robô móvel navega, e outro representando o teto. Essa divisão entre mapa de chão e teto permite ao robô armazenar dados que não influenciariam na navegação porém ainda navegar por debaixo de obstáculos. Idealmente o mapa de teto é formado logo acima do robô, onde o usuário informa a altura como parâmetro (os parâmetros de entrada do algoritmo de mapeamento serão discutidos na seção 4.2). A formação do mapa de teto é importante para criar um mapa 3D mais fidedigno.

Para o mapa local proposto foi criado um novo método chamado mapeamento por janela deslizante. O mapa local se trata de três grids dispostos em pé a uma distância fixa do robô distribuídos lado a lado. Os grids laterais estão dispostos em um ângulo (definido pelo usuário) em relação ao grid central, podendo variar entre zero e 90 graus. Isso permite um controle sobre a configuração da janela deslizante, permitindo que ela represente melhor os lados do robô com um ângulo próximo a 90 graus em casos de ambientes fechados, ou permitir uma grande abertura e representação de obstáculos distantes com um ângulo próximo de zero. Uma imagem da disposição dos mapas pode ser vista na figura 19. A medida que o sensor realiza o escaneamento (vertical ou horizontal, independente do movimento) os pontos lidos que estão no alcance do mapa são convertidos e assinalados nos grids, quando o robô se move, os pontos representados são compensados para manter a informação de obstáculos registrados consistente. A conversão é necessária pois os pontos vêm de uma nuvem de pontos, cuja representação é dada em coordenadas cartesianas (x,y,z) enquanto os mapas apenas possuem 'elevação', portanto a distância dos pontos até cada mapa deve ser calculada um a um. Para reduzir o custo computacional, um filtro é feito quando a nuvem de pontos é recebida, reduzindo a quantidade de pontos baseado na resolução do mapa, aproveitando-se da característica sequencial da leitura LIDAR. O filtro elimina vizinhos que estejam mais próximos que uma resolução (verificando nas três dimensões), tornando o cálculo e atribuição subsequentes mais eficiente. Para levar em conta o deslocamento do robô, não existe método simples para realizar a transformação dos pontos do mapa, uma vez que existem apenas os dados de elevação e posição da matriz do grid. A transformação foi um problema a parte cuja solução não foi completamente encontrada. O processo de resolução do problema do movimento do mapa é descrito com mais detalhes na seção 4.1.1.

A técnica de mapeamento faz um uso intenso de uma das funcionalidades do ROS: a árvore de transformadas ('tf tree'). Cada tópico a ser publicado deve informar a sua referência e a distância entre cada referência é armazenada pelo usuário no tópico '/tf', com estes parâmetros a transformação se torna possível.

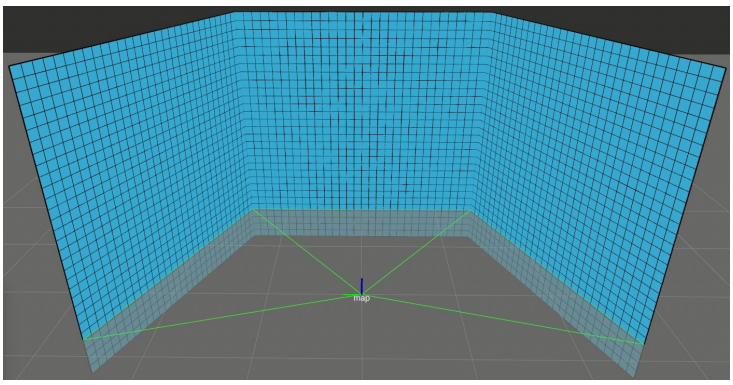


Figura 19: Janela deslizante inicializada sem dados assinalados.

O algoritmo de mapeamento possui como entrada uma nuvem de pontos, a 'tf' do robô e os parâmetros de configuração. A nuvem de pontos é proveniente do sensor LIDAR, que publica um tópico do tipo '*Laser Scan*', então convertido para o devido formato (*Point Cloud*). A 'tf' representa a árvore de transformadas do robô, com a localização de cada sensor e atuador que compõem o robô e também alguns pontos de referência importantes. Mais informações sobre a 'tf' são encontradas na seção 4.1. A árvore de transformadas do sistema pode ser visualizada na figura 21. Os parâmetros de configuração são informados pelo usuário através de um arquivo YAML na pasta 'config' dentro do pacote proposto. Este arquivo possui todas as configurações necessárias para definir a operação do motor de varredura de escaneamento (se necessário) e o tamanho, resolução e disposição dos mapas locais e globais (mais sobre na seção 4.2). Como saída do algoritmo têm-se os tópicos que compõem os mapas locais e globais, bem como o mapa de ocupação a partir do mapa global, e o comando para o movimento do motor.

O algoritmo de mapeamento possui várias configurações e é composto por vários arquivos. Eles serão explanados individualmente na próxima seção.

4.1 SOFTWARE

O software de mapeamento foi programado como um pacote ROS (*Robot Operating System*) e usa várias de suas funcionalidades, como a árvore de transformadas e o sistema de mensagens. O pacote, que recebeu o nome de 'scan map', é organizado como uma grande pasta que contém os arquivos de cada nó ('nós' são como são chamados os processos executados pelo ROS). Os nós que serão executados são configurados no arquivo '*launch*' (formato de arquivo padrão do ROS para configurar nós e parâmetros a serem iniciados). Três arquivos *launch* são disponibilizados no pacote desenvolvido: um com configurações para inicializar o mapa de escaneamento (varredura do sensor vertical, esquerda/direita) por janela deslizante, outro também para o mapa em janela deslizante porém configurado para motores

na horizontal (varredura cima/baixo), e um último *launch* para o método omnidirecional e varredura horizontal. O foco do projeto é o método de mapeamento por janela deslizante e varredura com sensor na vertical. Este método que está descrito nesta seção. As diferenças dos outros métodos em relação a esse estarão descritas na seção 4.1.2.

Como o pacote precisa de um *laser scan* de entrada e um motor ele depende destes componentes para operar. Para a execução em uma situação real, é necessário portanto dos pacotes 'hokuyo node' (para o sensor *laser scan*) e 'epos hardware' (para a comunicação com o motor). Em uma simulação (utilizando o software V-Rep, por exemplo) é necessário componentes similares simulados com as mesmas entradas e saídas que um sensor Laser Scan e um motor real.

4.1.1 MAPEAMENTO POR JANELA DESLIZANTE

O pacote possui oito nós que integram a técnica de mapeamento, dos quais sete foram desenvolvidos pelo autor. O oitavo nó vem do pacote 'grid map', um visualizador que permite converter o mapa global gerado pelo método de mapeamento proposto em mapa de ocupação, facilitando a integração com outros pacotes SLAM 2D. Uma imagem dos nós que compõem o método de mapeamento junto aos nós do Hokuyo, do motor e do robô utilizado para ensaios reais pode ser visualizada na figura 20.

Dos oito nós criados, quatro são publicadores de transformadas (na figura: 'sensor_tf', 'odom_map_tf', 'map_tf' e 'local_map_tf'), um para controle do motor (motor_sensor_64), um para conversão do *laser scan* de entrada para nuvem de pontos (scan_to_cloud), e dois para criação dos mapas locais e global ('scan_map_local' e 'scan_map_global'). A descrição do método de mapeamento começa pelos nós mais simples: os publicadores de transformadas.

Quatro nós publicadores de transformadas organizam os pontos de referência para o sensor Hokuyo, para o local inicial do robô móvel e para os mapas global e local. Estes publicadores são essenciais para a utilização da 'tf', uma ferramenta que mantém a relação entre os pontos de referência que integram o robô e permite a conversão por transformada entre eles.

O primeiro nó publica a transformada entre o sensor Hokuyo e 'base link'. Este nó toma como entrada a posição do motor e aplica sua rotação no eixo de arfagem (*yaw*) e uma rotação de 90 graus negativos no eixo de rolamento (*roll*) para a leitura vertical (com o sensor deitado). Além da rotação é adicionado também o deslocamento nas três dimensões entre o sensor e o centro do robô.

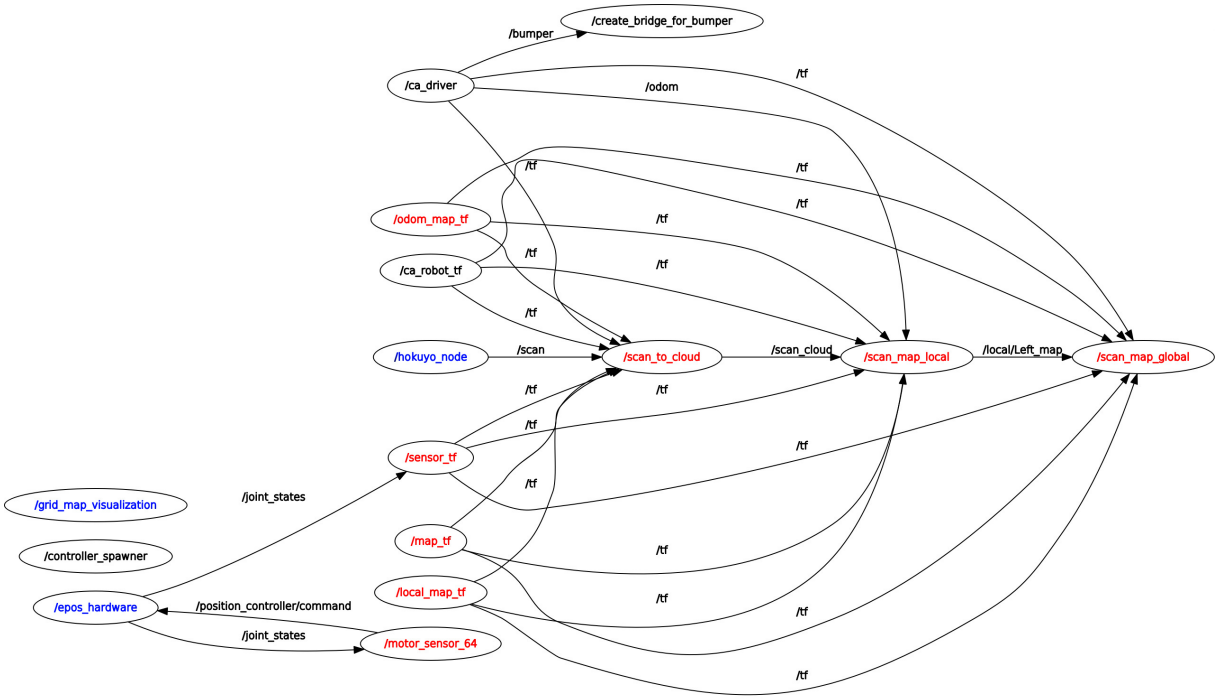


Figura 20: Nós de processamento do algoritmo de mapeamento, robô de ensaio, motor e sensor. Em azul, nós de pacotes necessários para o funcionamento do protótipo. Em preto, nós do pacote do robô móvel utilizado. Em vermelho, nós do pacote desenvolvido para o método de mapeamento proposto.

O segundo nó publica as transformadas do centro do mapa global, utilizando como parâmetro a referência 'odom' do robô. Essa referência é comumente adotada em robôs móveis para guardar a informação da posição inicial, em relação a qual a odometria é calculada. Essa referência é fixa e criada quando o robô é ligado. Por questão de organização a referência 'map' é criada com a mesma posição da referência 'odom'. Manter a referência do mapa separada da odometria permite a implementação da funcionalidade de carregar mapas no futuro (situação que faria com que as duas referências tivessem posições diferentes).

O terceiro nó define a posição do grid do teto do mapa global (referência 'ceiling') em relação ao centro do grid global de chão (referência 'map'). A mesma posição x e y da referência 'map' é utilizada, porém a altura da base do teto em relação ao chão é definida por um parâmetro inicial chamado 'ceiling_height' e rotacionado 180 graus, de modo que elevações positivas no mapa do teto representam um obstáculo mais próximo do robô.

O último nó de transformadas representa as posições centrais dos grids do mapa local. Este nó toma como parâmetros de entrada todas as definições de tamanho do mapa local, sendo estes o ângulo entre o mapa local da frente e os dois mapas laterais (sendo 90 graus representando os mapas laterais perpendiculares ao mapa da frente), o tamanho dos mapas (largura e altura), a distância entre o centro do mapa da frente e o robô e a elevação do centro do

mapa em relação a referência 'base link'. Baseando-se nestes parâmetros, o centro e orientação de cada mapa (frente, esquerda e direita) são calculados e as transformadas são publicadas. Os mapas são orientados em direção ao robô, com elevação positiva representando obstáculos mais perto do robô.

Com estas informações a árvore de transformadas fica completa, permitindo uma conversão de pontos entre várias referências. A árvore final de transformadas simplificada, sem incluir outras referências que não são usados pelo algoritmo de mapeamento, pode ser visualizada na figura 21.

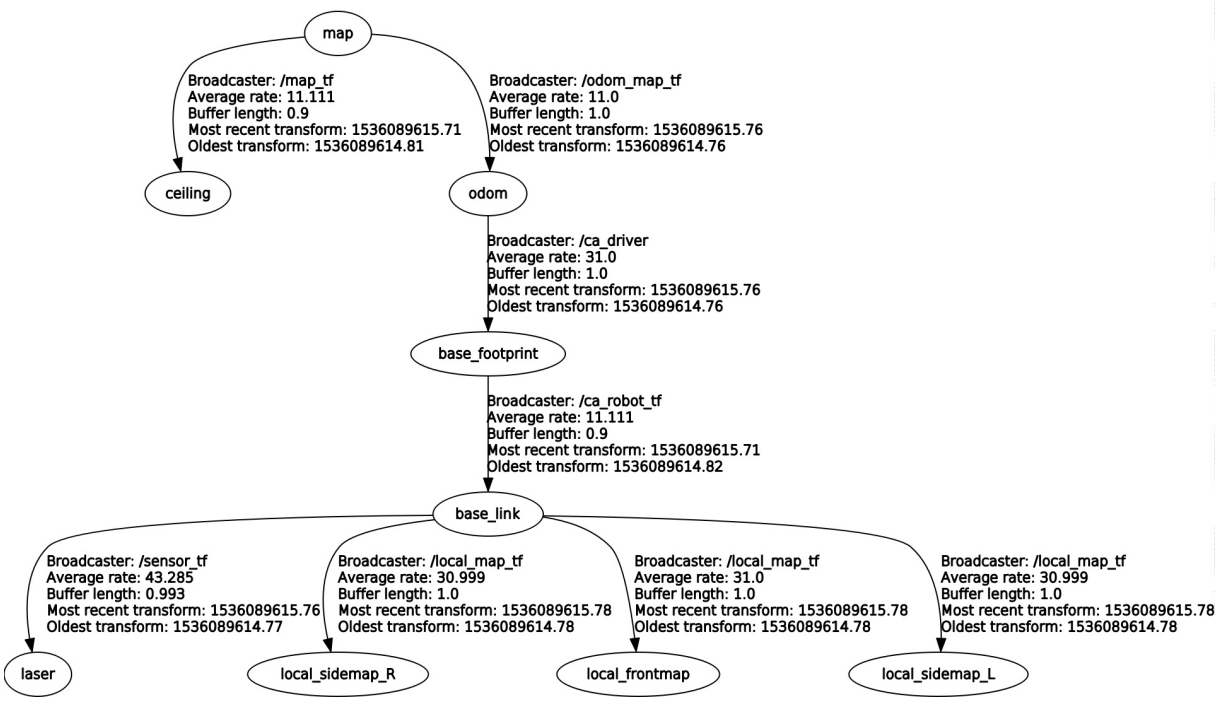


Figura 21: Árvore de transformadas resultante do robô e dos mapas.

A árvore de transformada é consultada por todos os outros processos pertencentes ao método. Sem algum dos quatro nós de transformadas a árvore fica incompleta e o processo de mapeamento entra em modo de espera até que a árvore esteja disponível novamente.

Os pacotes que formam a estrutura principal do processo de mapeamento têm o dever de girar o motor para que o sensor realize o escaneamento, realizar a conversão dos dados do sensor para uma nuvem de pontos e realizar o mapeamento local e global. Um diagrama de blocos representando o processo de mapeamento completo pode ser visualizado na figura 22.

O pacote que realiza os **movimentos do motor** tem como entrada a posição atual do motor e os parâmetros de configuração do arquivo YAML, listados na seção 4.2. Por causa da decisão de não se utilizar um conector USB rotativo para manter o giro, este algoritmo não permite giros completos, ao invés disso varia a posição do motor entre um ângulo máximo

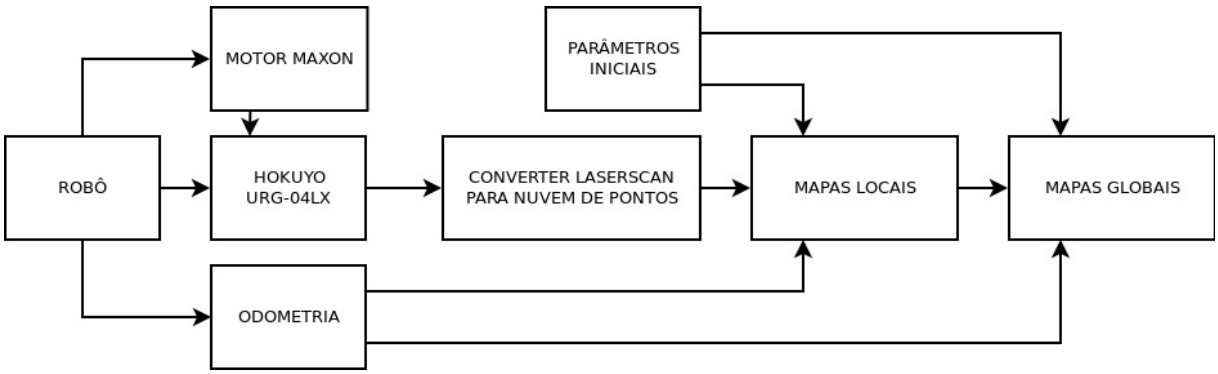


Figura 22: Diagrama de blocos com a estrutura geral do software de mapeamento.

e mínimo para realizar o escaneamento. Este algoritmo baseia-se no parâmetro '*mode*' para definir o modo de operação do motor Maxon Motor que gira o sensor Hokuyo. Este parâmetro permite ao usuário configurar o método de varredura em quatro modos diferentes: o método de escaneamento vertical, escaneamento horizontal, escaneamento omnidirecional e método de limites definidos. O método de escaneamento vertical configura uma varredura esquerda/direita intermitente e utiliza os parâmetros de configuração do mapa local para calcular o ângulo máximo e mínimo, fora dos quais o mapa local não representa a informação. Para o método horizontal o mesmo conceito é utilizado, porém utilizando a altura dos mapas é utilizada no cálculo pois este método é para varreduras cima/baixo. O método omnidirecional realiza giros completos e então alterna o sentido para mais um giro completo, repetindo este ciclo para que o sensor cubra todos os ângulos em volta do robô, este método foi feito para ser utilizado em conjunto com o método de mapeamento omnidirecional que difere do método de janela deslizante. Por fim, o método de limites definidos permite ao usuário informar dois parâmetros adicionais, um ângulo máximo e mínimo, para varreduras customizadas. O algoritmo permanece neste ciclo de varredura por toda a operação do pacote, a não ser que a comunicação com o motor caia, levando o algoritmo a não enviar mais comandos até que a comunicação seja restaurada.

As mensagens *laser scan* provenientes do sensor LIDAR através do pacote 'Hokuyo node' são recebidas pelo **conversor *laser scan* para nuvem de pontos** (*Point Cloud*). Este nó tem como entrada o tópico de *laser scan* e um parâmetro de configuração chamado 'scan frame' que determina a referência da nuvem de pontos de saída. Para a estratégia de mapeamento desenvolvida, a referência escolhida foi 'base link'. A conversão é necessária pois a mensagem *laser scan* informa apenas dados de distância em um vetor, bem como o ângulo máximo e mínimo. Baseando-se nestes parâmetros e tendo a referência da posição e orientação do sensor, é possível converter as medidas de distância em pontos 3D (xyz), que então são transformados para a referência informada por parâmetro. Estes dados são encapsulados na mensagem tipo

Point Cloud (nuvem de pontos), que é publicada para ser utilizada pelos outros nós do pacote. Nenhum filtro ou truncamento é realizado nesse processo.

A nuvem de pontos, que foi convertida a partir do *laser scan* e transformada para a referência do robô, serve de entrada para o algoritmo de construção do **mapa local**. O mapa local recebe a nuvem de pontos convertida, aplica um filtro de vizinhança e percorre os pontos assinalando-os no mapa. Um método de compensação de movimento do robô transforma o mapa de modo que obstáculos mantenham sua posição em relação ao robô. Um diagrama de blocos do método de construção do mapa local pode ser visualizado na figura 23.

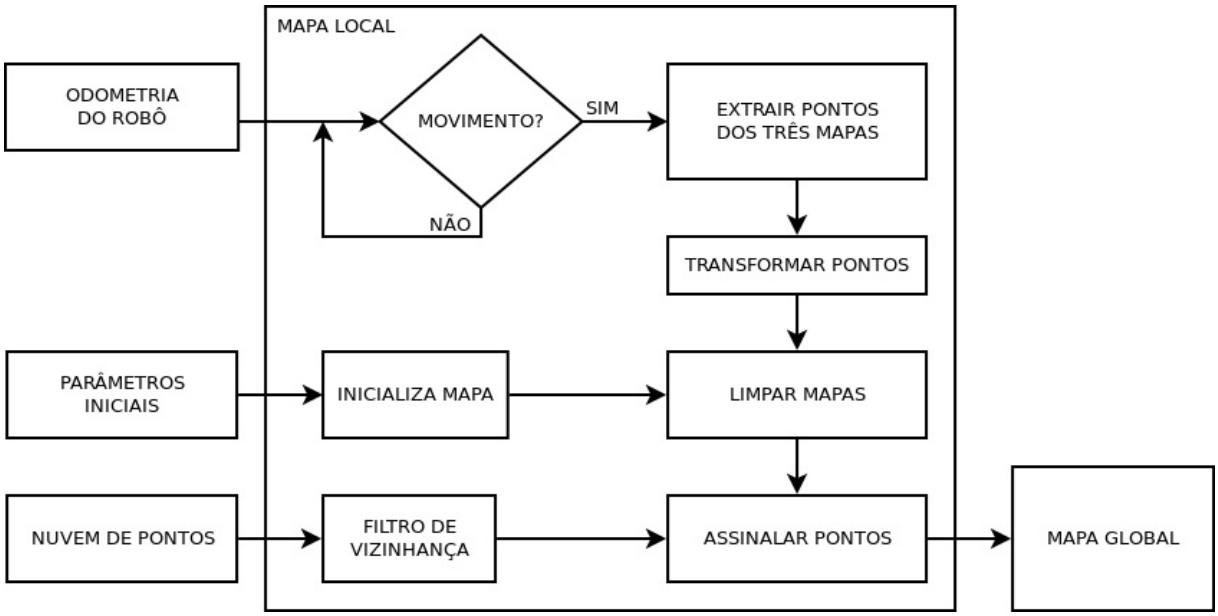


Figura 23: Diagrama de blocos com a estrutura geral do software construção do mapa local.

O mapa local é composto de três grids dispostos a uma distância 'map distance' do robô. Estes grids estão dispostos lado a lado, sendo um centrado no robô e dois grids laterais a um ângulo 'map angle' em relação ao grid central. Os grids possuem tamanho 'length' por 'width', e uma resolução 'map resolution' que define a quantidade de células retidas. Eles têm seu centro na mesma altura do robô, porém podem ser elevados 'map elevation' metros para tornar mais eficiente sua área de cobertura. Os parâmetros descritos são dados pelo usuário a partir de um arquivo YAML. Um apanhado de todos os parâmetros de configuração do algoritmo pode ser encontrado na seção 4.2. Uma disposição dos mapas limpos (sem dados assinalados) pode ser visto na figura 24.

Para a construção do mapa local, a nuvem de pontos do sensor (já dispostos em relação a 'base link', referência padrão do robô) é recebida e filtrada. O filtro utilizado é um **filtro de vizinhança** local que elimina pontos muito próximos, reduzindo o tamanho da nuvem de pontos de entrada. O raio do filtro é de meia resolução do mapa local, uma vez que pontos com

densidade maior que a resolução do mapa são redundantes. Isso diminui o tamanho da nuvem de pontos a ser processada permitindo uma iteração rápida. A nuvem filtrada é publicada como um tópico para facilitar a depuração.

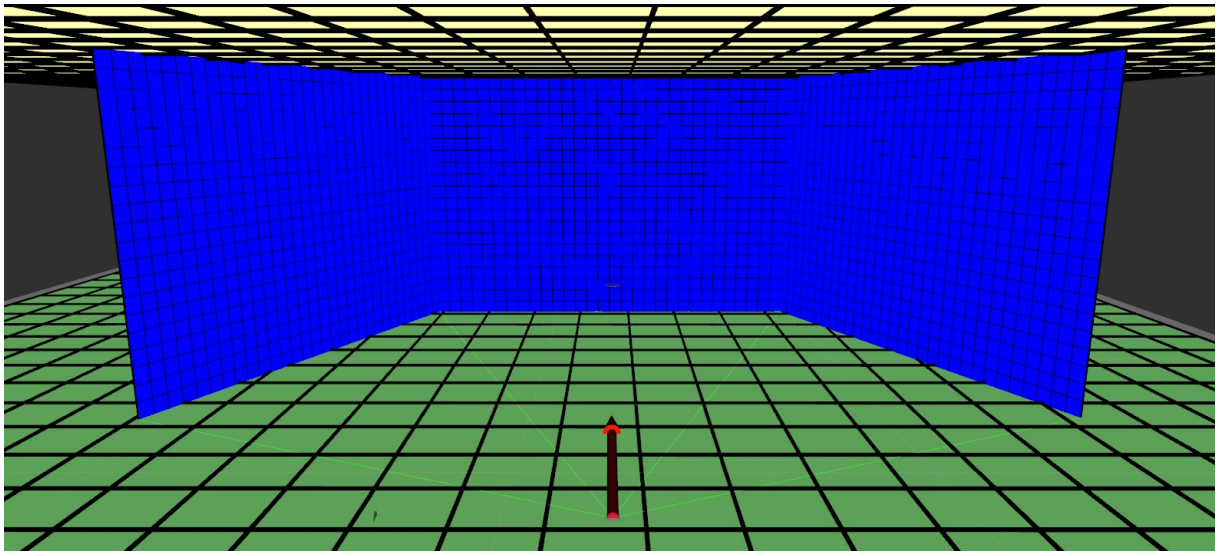


Figura 24: Mapas locais (frente, esquerda e direita) e globais (chão e teto) inicializados e limpos.

A nuvem de pontos filtrada é iterada e cada ponto contido é verificado para **assinalação**. O ângulo do ponto em relação ao eixo central do robô é calculado e utilizado para verificar em qual grid o ponto deve ser assinalado (esquerda, frente ou direita), podendo ser assinalado a dois grids ao mesmo tempo (redundante, assinalado por motivos estéticos). Após a definição do grid em que o ponto deve ser assinalado, a célula correspondente do grid é calculada (índices 'i' e 'j' da matriz do grid), bem como a distância do ponto ao grid, representando a elevação em relação ao plano descrito pelo grid. As informações coletadas por este procedimento ficam armazenadas na 'camada de elevação' do grid. Este cálculo traz o primeiro grande problema no método de representação de mapa local utilizado. Os grids são compostos de várias células equidistantes a uma resolução 'r' da outra, obrigando um arredondamento para a assinalação da elevação. Este arredondamento gera um erro nas duas dimensões do grid, altura e comprimento, cujo erro máximo é meia resolução. No momento da assinalação dos dados provenientes da nuvem de pontos este erro não chega a ser significante, porém em outras operações (transformação do mapa) este erro acumulará até a próxima leitura direta da nuvem de pontos (um período de leitura). Por este motivo, este erro de arredondamento é calculado e assinalado em outra matriz atrelada (camada) ao mapa, chamada 'camada de erro'. Apenas o erro lateral (no eixo horizontal) é armazenado uma vez que maior parte dos cálculos são feitos por causa do movimento lateral. O erro de altura no eixo vertical não é um problema pois nenhuma operação para o eixo vertical é realizada. Além da camada de erro, outra camada extra é criada chamada de 'camada de leitura', contendo apenas zeros e uns, discriminando

células que possuem um ponto assinalado das que não receberam conteúdo direto da nuvem de pontos. Essa camada é criada com o intuito de facilitar a iteração do grid para coletar apenas células com informação relevante. Com esses cálculos o mapa local consegue representar com acurácia um ambiente tridimensional com o robô imóvel.

O suporte aos movimentos do robô deve ser programado para permitir um mapeamento efetivo. Este foi o maior problema encontrado durante o desenvolvimento do método de mapeamento. Os dados do sensor são relativos à referência 'base link', ou seja, ao centro do robô. Isso torna a representação dos dados do sensor no mapa local muito simples. Entretanto, os obstáculos representados no mapa devem manter sua posição em relação ao mapa global quando o robô se move, porém o centro dos grids do mapa local possuem uma posição fixa em relação ao robô, significando que os dados representados no grid devem ser compensados por **transformação** ou **aproximação** quando o robô se move. As duas opções foram programadas e testadas, e possuem pontos a favor e contra seu uso.

Para o método por **transformação** dos pontos do mapa é necessário extrair os pontos do grid, representá-los em três dimensões, aplicar a transformada e reassinalar os pontos. Os dados representados nos três grids estão no formato [i, j, elevação] (método grid de representação), em relação ao respectivo grid. Para realizar uma transformação, é necessário converter todos os dados representados novamente para [x, y, z] (método cartesiano de representação) em relação ao centro do robô. Para isso, aplica-se uma operação para cada grid em um laço que percorre as células com dados representados, convertendo-os em pontos cartesianos. Os pontos recolhidos são armazenados em uma matriz 'n' por 3, onde 'n' é o número de pontos diferentes de zero assinalados nas matrizes e as três colunas representam as três coordenadas cartesianas. Essa matriz é multiplicada pela matriz de transformação para a rotação T definida na equação 1.

$$T = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

O ângulo θ é o passo de ângulo de rotação, definido pela diferença na orientação do robô atual em relação a orientação final da última transformação. Apenas a rotação no eixo de orientação lateral (guinada, ou '*yaw*') é levada em conta, não foram considerados ambientes não-planos neste trabalho. Além da rotação, o deslocamento também é levado em conta por um método similar. A posição do robô da última transformação é armazenada e utilizada para medir a variação de posição desde a última iteração. O deslocamento nos eixos x e y são

somados nos novos pontos transformados. Após isso, nos mapas limpos e sem dados, os pontos transformados são iterados e passam pelo mesmo método de assinalação ao mapa que o sensor.

Essa abordagem de compensação do movimento por transformação possui dois grandes problemas. O erro acumulado a cada transformação feita e a atribuição de dois pontos em uma célula. Toda vez que um ponto é assinalado no mapa ele precisa ser arredondado para a célula mais próxima, representando um erro de meia resolução do mapa. Para aliviar este problema, existem duas opções: armazenar o erro de arredondamento, ou manter os pontos da nuvem de pontos em outro formato preliminar a partir do qual o mapa seria gerado. A solução de armazenar pontos extras da nuvem foi descartada pois a memória ocupada pelo programa aumentaria bastante, mesmo que um limite de pontos armazenados fosse imposto. Portanto, decidiu-se por armazenar o erro de arredondamento em uma camada de cada grid. Camadas representam matrizes secundárias do mesmo tamanho e tipo da matriz de elevação, que podem ser utilizadas para definir cor, ocupação, erro ou qualquer outro dado que respeite a definição. Como definido anteriormente, apenas o erro lateral de arredondamento foi escolhido para ser armazenado. Este erro é adicionado novamente nos pontos extraídos dos mapas para tornar a transformação mais precisa. Entretanto, isso resolve apenas um dos problemas. Existe também uma complicação com os pontos transformados que serão *"reassinalados"*. Dependendo do ângulo de rotação, é possível que pontos de duas colunas dos mapas fiquem todos próximos de uma só coluna após a transformação resultando em uma coluna vazia sem dados assinalados. Não foi encontrada uma solução simples para este problema, levando a tentativa de se compensar os dados de obstáculos por aproximação.

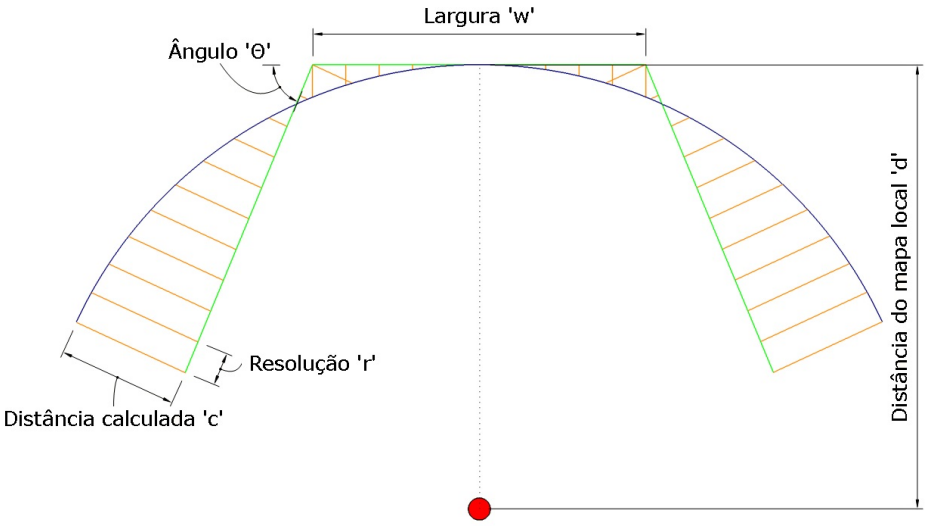
A correção dos dados do mapa por causa do movimento do robô utilizando **aproximações** foi motivada pela necessidade de manter a estrutura dos dados lidos, enquanto tenta-se corrigir os dados de modo que a posição dos obstáculos lidos seja constante. A abordagem tomada foi de utilizar um deslocamento de colunas (*shift*) a cada vez que o robô rotacionar um ângulo de passo pequeno pré-determinado (*step*). O ângulo de passo é calculado a partir do ângulo entre a célula diretamente a frente do robô, tomada como referência uma vez que na maioria das configurações ela está perto do ângulo médio de todas células. Utilizar apenas um ângulo de passo ao invés de calcular o ângulo apropriado para cada célula é uma aproximação, porém simplifica o programa imensamente uma vez que seria necessário múltiplas iterações sobre a matriz para determinar o deslocamento de cada célula e possivelmente seria necessário uma camada extra de fundo para armazenar dados que seriam sobrepostos (uma célula passando na frente da outra).

Com este algoritmo é possível manter a estrutura total do mapa local, perdendo apenas dados que seriam deslocados para fora do mapa. Os obstáculos registrados mantém sua posição em relação ao mapa global, porém apenas se não estiverem muito perto, muito longe ou trocarem de grid. A troca de grid é um problema em especial pois várias células existem onde os grids se encontram (grid lateral com grid central), porém a variação espacial dos pontos com o deslocamento de colunas é menor em relação à colunas no centro dos grids.

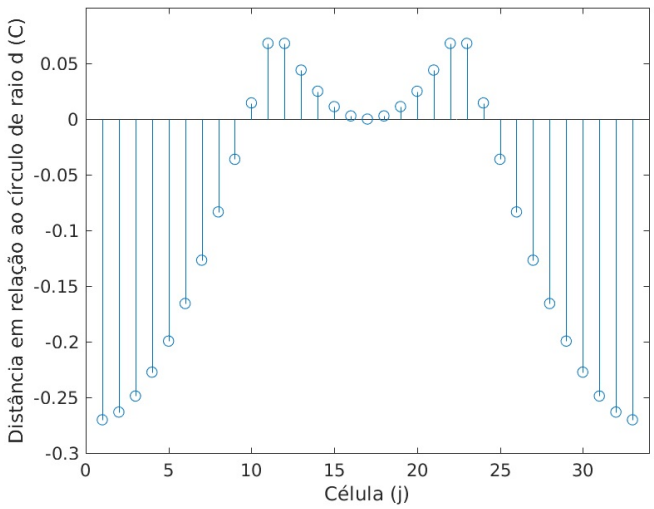
Numa tentativa de melhorar este método, uma compensação de magnitude de elevação registrada foi planejada como um modo de aproximar o deslocamento lateral a uma rotação, de modo que a medida em que colunas fossem deslocadas, as distâncias representadas deveriam ser compensadas de modo que a distância entre o obstáculo e o robô continuasse constante. Foi teorizado que esta aproximação deve depender de um valor multiplicativo (coeficiente) e um valor aditivo (constante). Como o ângulo de passo para rotação (e portanto para o deslocamento de colunas) é constante, valores exatos nunca serão atingidos, porém uma aproximação ainda é possível. A proposta foi coletar a distância entre cada célula e um círculo com raio igual a distância do grid central ao robô (o parâmetro 'map distance') formando um vetor de tamanho igual ao número de colunas dos três grids para utilizar na correção aditiva, da maneira como mostra a imagem da figura 25a, e coletar o ângulo formado por cada célula e o centro do robô para construir um vetor (de mesmo tamanho que o anterior) que permita uma correção por coeficientes, da mesma maneira como foi calculado o ângulo de passo, mostrado na figura 26a. A ideia foi criar os dois vetores (figuras 25b e 26b) com estes dados para ajustar a distância do obstáculo registrado no mapa a medida em que é deslocado lateralmente.

Os vetores são gerados durante a inicialização dos mapas locais e mantidos constantes durante a execução por serem baseados em parâmetros iniciais. A cada deslocamento, utiliza-se o valor da célula atual e acha-se a diferença em relação ao coeficiente correspondente a célula final, essa diferença entre os coeficientes de duas células adjacentes é utilizada na compensação. Como se trata de uma aproximação, a fórmula não foi deduzida a partir de uma propriedade geométrica ou trigonométrica, pois estes cálculos seriam muito dependentes do ângulo de passo que já é aproximado. A fórmula é apenas um compensação para deixar a operação de deslocamento menos grosseira. A fórmula final escolhida está descrita na equação 2.

$$M = \sum_{i}^{s} \sum_{j}^{3s} M(i,j) * (1 - \frac{(\phi(j) - \phi(j \pm 1))}{\phi(3s/2)}) + (C(j) - C(j \pm 1)) * \frac{d - M(i,j)}{d} \qquad (2)$$



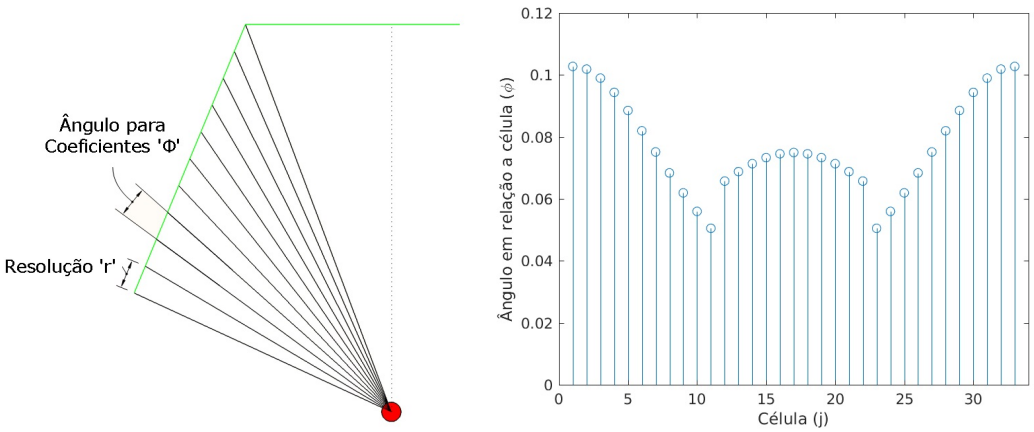
(a)



(b)

Figura 25: 25a: Representação das distâncias (em laranja) medidas para grids (em verde) de 10 células em relação ao círculo de raio 'd' (em azul). 'c' representa a distância que é armazenada no vetor (uma para cada célula). 25b: Vetor final 'C' de distâncias entre células do mapa e um círculo de raio 'd'.

Na equação, i e j são os iteradores de linha e coluna da matriz. A matriz M representa os três mapas locais concatenados, de tamanho s x $3s$ onde s é a quantidade de células do grid. O vetor ϕ representa o vetor de ângulos de cada célula (coeficientes) e tem tamanho $3s$, este vetor é ponderado pela sua mediana ($\phi(3s/2)$) que representa o ângulo da célula central do grid central (o mesmo valor que o ângulo de passo). O vetor C, também de tamanho $3s$, representa a distância entra a célula e um círculo de raio d (constantes aditivas) e é ponderado pela distância entre o obstáculo representado e o robô. d é a distância entre o robô e o mapa central. Um deslocamento para a direita do robô implica em um deslocamento à esquerda para o mapa,



(a) (b)

Figura 26: 26a: Representação dos ângulos coletados para grids de 10 células. 'ϕ' é o ângulo coletado para cada célula e armazenado no vetor. 26b: Vetor final 'ϕ' com os ângulos entre os dois lados opostos de cada célula e o centro do robô.

levando a escolher a operação de subtração na iteração do j. Da mesma forma, deslocar o mapa à direita por um movimento de rotação para a esquerda leva a uma adição no iterador j.

A operação compensa levemente a elevação assinalada e ajuda na integridade do mapa para rotações pequenas. Porém, a estratégia de utilizar aproximação por deslocamento, mesmo que lapidada, é grosseira demais para manter os dados representativos em rotações grandes. Outro problema desta estratégia é o erro acumulado. Por mais que para ângulos baixos os dados continuem precisos, ao longo do tempo o erro se acumula, uma vez que cada célula só recupera sua confiabilidade quando reassinalada, e para isso é necessário que um ponto da nuvem de pontos seja lido e assinalado a ela. Manter esta estratégia durante tempo indefinido exigiria um método de limpeza do mapa após um número fixo de iterações, porém para diferenciar dados acurados de pouco acurados seria necessário mais processamento, memória e tempo investido.

No fim, decidiu-se por manter a compensação por aproximação como uma opção secundária devido ao método por transformação ter um desempenho muito melhor em longos ensaios (apesar de ter um custo computacional maior). O método por aproximação funciona bem em ambientes fechados com obstáculos grandes (paredes ou portas) à média distância do robô porém é muito ineficiente para manter a estrutura de obstáculos médios ou pequenos devido a incapacidade de se manter a representação íntegra e fiel. Na seção de ensaios (seção 5) são realizados testes comparativos entre os dois métodos de compensação de movimento onde é possível visualizar melhor o comportamento de ambos.

O **mapa global** possui como entrada os três tópicos dos grids que descrevem o mapa local, a odometria do robô e os parâmetros iniciais, e possui como saída o mapa global

construído e um mapa de ocupação (convertido a partir do mapa global do chão graças ao pacote 'grid map visualization'). Um diagrama de blocos com a estrutura do algoritmo de construção do mapa global pode ser visto na figura 27.

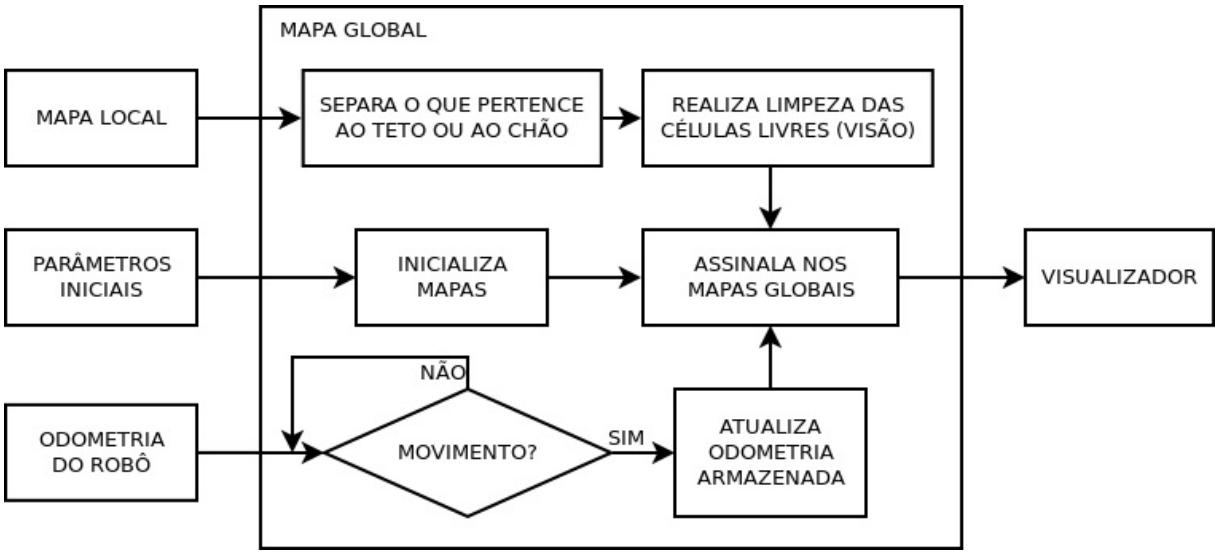


Figura 27: Diagrama de blocos com a estrutura geral do software construção do mapa global.

No momento em que um grid novo é recebido, utiliza-se a posição do mapa e a distância em relação ao chão para identificar a linha que representa o limiar sobre o que deve ser representado no mapa global do chão e no mapa global do teto, dividindo a matriz do mapa local em duas. As matrizes de cada grid do mapa local são iteradas e através da posição das células de cada obstáculo representado, da elevação representada nestas células, da atual posição e orientação do robô e dos parâmetros que definem o mapa local (elevação, distância em relação ao robô, tamanho dos grids e ângulo formado entre o grid central e os laterais) é possível calcular onde o obstáculo representado no mapa local está em relação ao mapa global.

A limpeza do mapa global para pontos não registrados é feita utilizando um iterador especial disponível pelo pacote 'grid map'. O iterador em polígono permite definir pontos no mapa que servem como arestas para a construção de um polígono, percorrendo apenas as células cujo centro está contido neste polígono. A limpeza é feita apenas no mapa do chão uma vez que o algoritmo atual está orientado para robôs terrestres. O valor máximo para cada coluna de cada grid é calculado, os pontos são extraídos e utilizados para definir o polígono (um por grid local). As células contidas no polígono são percorridas e zeradas antes da assinalação dos pontos recebidos. Isso efetivamente define um campo de visão para o robô limitado aos obstáculos representados nos mapas, limpando o mapa global no caso de objetos dinâmicos.

Cada grid criado pelo método de mapeamento (3 locais e 2 globais) é encapsulado em uma mensagem 'grid map' (definida pelo pacote grid map) e representa todos os dados

assinalados ao grid e suas camadas. O tamanho das mensagens é constante durante a execução, alocado previamente pelo respectivo algoritmo (local ou global). Os grids e suas camadas são matrizes de variáveis tipo ponto flutuante e possuem o mesmo tamanho que o número de células neles representado. A **memória** ocupada por este método de representação foi calculada para cada mensagem enquanto se variava a resolução do mapa. Os dados coletados se encontram nas tabelas 4 e 5. É válido notar que uma célula a mais é adicionada em cada dimensão do grid para facilitar cálculos de limite.

Resolução	Dimensão	Mapa local utilizando método de transformação	Mapa local utilizando método de aproximação
20 cm	16x16	10.2 KB	6.93 KB
10 cm	31x31	35.58 KB	23.85 KB
5 cm	61x61	135.18 KB	90.09 KB
2.5 cm	121x121	540.66 KB	368.64 KB

Tabela 4: Memória ocupada em duas situações pelo mapa local com grids fixos em 3m x 3m. A memória informada inclui os três grids que formam o mapa local.

Resolução	Dimensão	Mapa global
40 cm	51x51	21.18 KB
20 cm	101x101	81.98 KB
10 cm	201x201	356.36 KB
5 cm	401x401	1310.72 KB

Tabela 5: Memória ocupada pelo mapa global fixado em 20m x 20m.

As tabelas mostram que os mapas ocupam uma memória bastante baixa se comparada a outros métodos de mapeamento como o Octomap. Por exemplo em (MERRIAUX et al., 2018), onde o Octomap utilizando formato de representação Octree (armazenando o grid 3d em uma árvore de representação de obstáculos, que permite a utilização de resoluções menores em objetos complexos) é submetido a testes quando utilizado no mapeamento de ambientes com tubulação, ocupando 45.9 MB de memória para representar um ambiente complexo de 10m x 7m x 4m com 8cm de resolução mínima de voxel. Para aproximar as situações de mapeamento comparadas, é possível citar (WURM et al., 2010), onde o Octomap é usado em um ambiente interior pequenos (3.5m x 5m x 2m) com resolução de 5cm e o mapa final ocupa 1.38 MB de memória, um pouco mais que um mapa de 20m cúbicos com a mesma resolução utilizando o algoritmo de mapeamento por janela deslizante. Isso se deve ao grid ser um método de representação 2D que escala quadraticamente com o aumento de resolução, em contraste a proporção cúbica observada no grid 3D do Octomap.

4.1.2 MAPA OMNIDIRECIONAL

Adicionalmente ao método de mapeamento por janela deslizante, um mapeamento omnidirecional também foi criado utilizando o pacote 'grid map' e os conceitos de mapa de teto e mapa de chão. Porém, este método não possui um mapa local por janela deslizante, ao invés disso se aproxima mais de métodos clássicos de mapeamento ao utilizar um mapa local como um grid pequeno disposto horizontalmente no chão centrado ao robô. Como pode ser visto na figura 28.

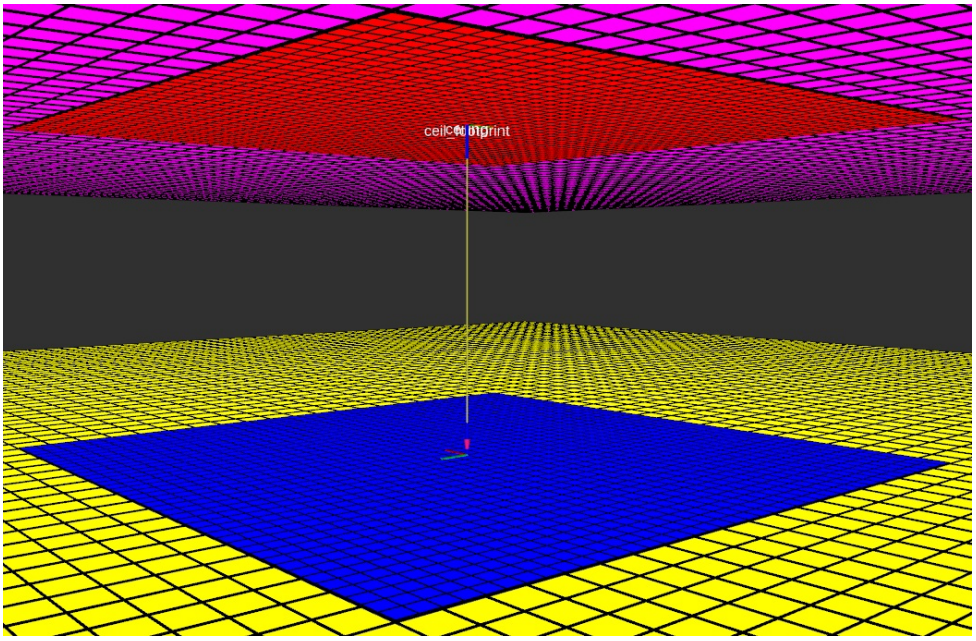


Figura 28: Mapas locais e globais do método omnidirecional. Mapa local de chão em azul e de teto em vermelho. Mapa global de chão em amarelo e de teto em rosa.

Este mapa é definido por quatro grids. Dois grids formam o mapa local (chão e teto) e dois formam o mapa global (novamente chão e teto). Estes grids são configuráveis por parâmetros exatamente como o método por janela deslizante. Três algoritmos diferem este método do outro. Um publicador de transformadas, o mapa local e o mapa global.

O novo publicador de transformadas substitui o publicador de transformadas do mapa local do método de mapeamento por janela deslizante. Ele é responsável por criar duas novas referências, a 'base footprint' e a 'ceiling footprint'. A referência 'base footprint' é comumente utilizada para representar a referência 'base link' sem elevação em relação ao chão e muitos robôs a utilizam por padrão para facilitar a configuração de mapeamento. A referência 'ceiling footprint' tem a mesma função porém na altura do teto. Estas referências são o ponto central dos dois novos mapas locais.

Quando uma nova leitura de *laser scan* transformada para nuvem de pontos chega até o mapa local do método omnidirecional, ela é filtrada e iterada para assinalação. A assinalação é dividida na altura definida pelo parâmetro 'ceiling threshold' que determina quais pontos pertencem ao chão e quais pertencem ao teto. Estas informações são convertidas de [x,y,z] dos pontos para [i,j, elevação] e assinaladas ao respectivo grid do mapa local.

A odometria é monitorada para responder aos movimentos do robô. A posição 'x' e 'y' do robô em relação ao centro do mapa global é constantemente atualizada, bem como sua rotação ('*yaw*' ou arfagem). A odometria é contabilizada toda vez que um ponto é assinalado. Como no método de janela deslizante, a variação da posição 'x' e 'y' do robô também causa uma compensação para que os obstáculos mantenham sua posição relativa ao mapa global. Um deslocamento de linhas ou colunas do mapa local de chão e teto é feito toda vez que o robô se mover a uma distância equivalente à resolução de uma célula em algum dos eixos. Isso é necessário pois o mapa local acompanha o robô, porém os obstáculos lidos são fixos em relação ao mapa global.

O mapa global recebe os tópicos do mapa local publicados, aplica o deslocamento da odometria, e assinala o ponto com altura equivalente no grid do mapa global respectivo (chão ou teto). O mapa global omnidirecional também é inicializado pelos mesmos parâmetros de configuração, e possui as mesmas saídas: um mapa de chão e um de teto. O mapa de chão, similarmente ao outro método de mapeamento, é convertido para mapa de ocupação pelo pacote 'grid map visualization'.

4.2 PARÂMETROS

Como já citado anteriormente, o pacote de mapeamento utiliza vários parâmetros de entrada para configuração do seu funcionamento. Isso foi feito para permitir uma troca rápida de valores de certas variáveis sem precisar compilar o projeto novamente, tornando o uso do pacote mais prático.

Todos os parâmetros de entrada com seus valores padrões serão listados abaixo junto com uma explanação de sua função, separados pelo nó que faz uso dos mesmos.

Conversor (/conversor):

- /scan_frame: base_link

  Contém a referência para a qual é transformada a nuvem de pontos resultante da leitura *laser scan*.

Mapa global (/global_map):

- /map_resolution: 0.2

  Valor em metros do tamanho de uma célula do mapa global.

- /ceiling_threshold: 1.0

  Valor em metros que define o limite de altura de obstáculos que serão considerados pertencentes ao teto, permitindo o robô passar por baixo (deve ser um pouco maior que a altura do robô).

- /ceiling_height: 2

  Valor em metros da altura base do teto.

- /width: 20

  Valor em metros da largura do mapa.

- /length: 20

  Valor em metros do comprimento do mapa.

Mapa local (/local_map):

- /map_distance: 4

  Distância em metros do grid da frente do mapa local em relação ao robô.

- /map_elevation: 1

  Elevação em metros do centro do mapa local em relação ao robô.

- /map_angle: 0.75

  Valor (entre 0 e 1) que multiplica $\pi/2$ e define o ângulo entre o grid da frente e o grid dos lados.

- /map_resolution: 0.10

  Valor em metros do tamanho de uma célula do mapa local.

- /width: 3

  Valor em metros da largura de cada grid do mapa local.

- /length: 3

  Valor em metros do comprimento de cada grid do mapa local.

- /error_compensation: true

  Habilita ou não a compensação do erro para o método de correção de rotação por transformada, criando uma camada de erro que armazena o erro de arredondamento, permitindo que situações de transformadas consecutivas (como rotacionar 90 graus rapidamente) sejam mais consistentes.

  Motor (/motor):

- /mode: V_WINDOW

  Parâmetro que define modo de operação do motor. V_WINDOW representa o modo de janela deslizante com sensor na vertical. H_WINDOW é similar porém com sensor na horizontal. OMNI define o mapeamento omnidirecional. SET permite definir limites angulares para o motor.

- /ang_max: 90

  Ângulo máximo (em graus) para o motor em modo de operação SET.

- /ang_min: -90

  Ângulo mínimo (em graus) para o motor em modo de operação SET.

Estes parâmetros ficam armazenados em um arquivo YAML junto ao pacote e são carregados toda vez que o pacote é inicializado, sem precisar de compilação. Isso fecha a seção de descrição do software do método de mapeamento, restando validar o método para diferentes situações para garantir seu funcionamento.

5 ENSAIOS E RESULTADOS

Vários ensaios foram feitos para validar as propostas dos algoritmos que integram o mapa. Os ensaios foram feitos em ambientes reais e virtuais para garantir a funcionalidade dos métodos de mapeamento propostos. Um total de 15 ensaios em ambiente real e 15 ensaios em ambiente virtual foram realizados com o método de mapeamento de janela deslizante. Ensaios adicionais apenas em ambiente virtual foram feitos com o sensor em posição horizontal escaneando verticalmente (inclinando cima/baixo) e no método de representação omnidirecional (sem janela deslizante).

Os ensaios em ambientes reais foram feitos utilizando o robô iRobot Roomba, que já dá suporte a vários requisitos da aplicação do software de mapeamento em robôs móveis como integração com o ROS, odometria, comando de velocidade e árvore interna de transformadas. Os ambientes reais escolhidos foram áreas dentro do laboratório LASCA da UTFPR onde o sensor foi desenvolvido. Por ser um ambiente com diversos obstáculos e pontos de referência permite a validação do método de mapeamento proposto.

Os ensaios em ambiente virtual foram feitos utilizando o software V-Rep. O V-Rep é um simulador 3D de robôs capaz de representar sistemas robóticos e ambientes de teste similares à realidade, possui integração completa com o ROS permitindo comunicar mensagens como um sistema robótico real. Um modelo simplificado de robô foi utilizado, concentrando os detalhes da simulação apenas no sensor e no motor de modo que se comportem semelhantemente a suas versões reais. O robô virtual responde aos comandos de velocidade do mesmo modo que o robô iRobot Roomba utilizado na validação real.

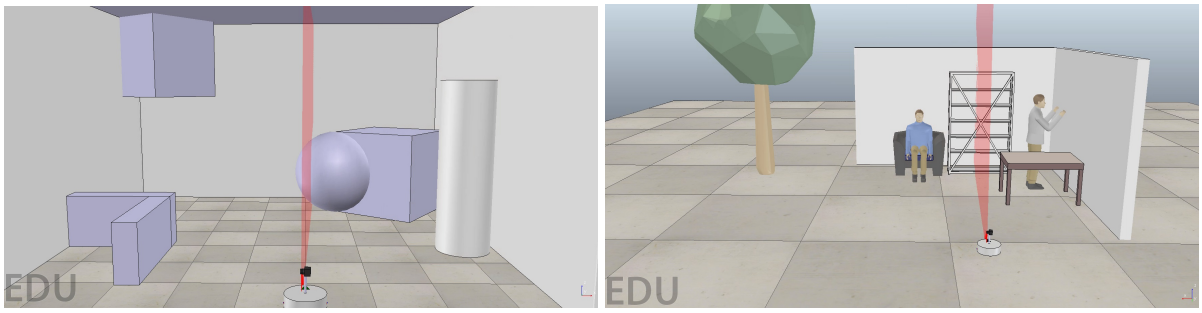
A maior diferença entre os robôs virtual e real é a precisão da odometria e a capacidade de obedecer precisamente os comandos de velocidade (o robô real, por causa do sensor e dos cabos presos a ele, faz trajetórias curvas quando ordenado a ir para frente).

Para validar todos os aspectos do método de mapeamento foram realizados três tipos de ensaios, reconstrução de ambiente (mapeamento parado, dois ambientes diferentes), avaliação da consistência do mapa na rotação (robô girando sobre o próprio eixo) e consistência do mapa

na navegação (robô em movimento). Para a reconstrução de ambiente, três níveis de resolução do mapa local foram avaliados (5cm, 10cm e 20cm) e dois níveis altura para o limite do mapa global de chão em relação ao teto (60cm e 1m). Para os ensaios de rotação e navegação, foram testados os diferentes métodos de compensação: transformada com correção de erro, transformadas sem correção de erro e aproximação. Todos os ensaios foram feitos em ambiente virtual e real. Para os ensaios reais, a distância entre o robô e mapa (parâmetro 'map distance') é de 2 metros, por causa dos ambientes fechados. Para os ensaios virtuais utilizou-se o valor padrão de 4 metros. Parâmetros que não têm seus valores citados no texto estão com seu valor padrão.

5.1 RECONSTRUÇÃO DE AMBIENTE

Os ensaios de reconstrução de ambiente envolveram dois ambientes virtuais e dois ambientes reais. Para os ensaios virtuais de reconstrução de ambiente, duas cenas no V-Rep foram construídas. Uma das cenas possui formas geométricas dispostas em um ambiente fechado (com teto), outra cena possui duas pessoas em um ambiente mais poluído para verificar a capacidade de representação de detalhes. As duas cenas podem ser visualizadas na figura 29.



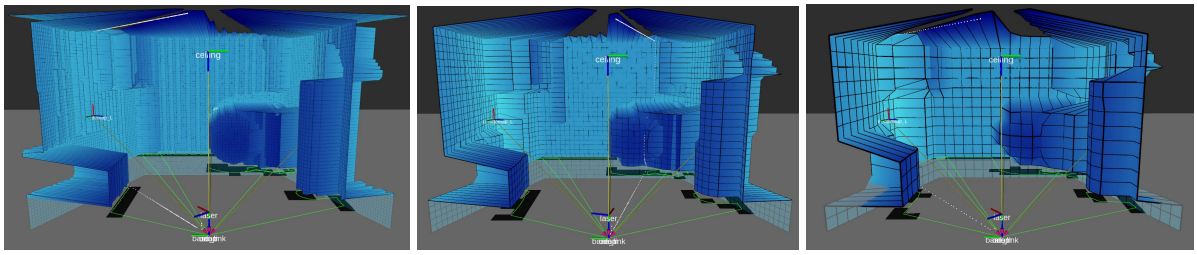
(a) (b)

Figura 29: 29a: Primeira cena de ensaio virtual, apenas uma sala fechada com formas geométricas de fácil identificação. 29b: Segunda cena de ensaio virtual, um lugar aberto com pessoas e objetos complexos.

Foram feitas reconstruções variando a resolução do mapa local entre 5cm, 10cm e 20cm. Para estes parâmetros a primeira cena resultou nos mapas obtidos na figura 30.

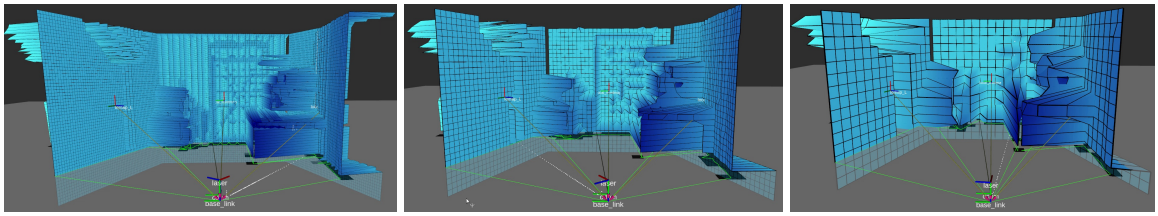
Para a segunda cena virtual, a mesma variação de parâmetros foi seguida. Os resultados podem ser visualizados na figura 31.

Para a avaliação do mapa global foi utilizada a segunda cena devido a uma particularidade, a mesa. A variação de limite entre chão e teto proposta para avaliar a capacidade do robô de classificar terreno navegável pode ser observada em como a mesa é representada no



(a) (b) (c)

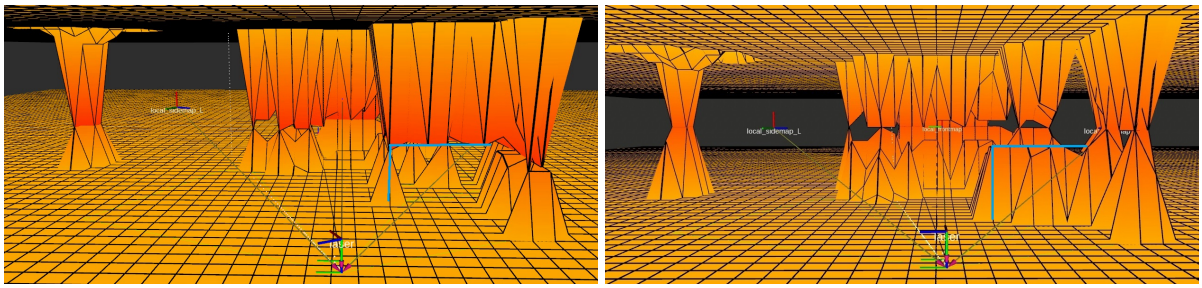
Figura 30: Primeira cena do ensaio virtual com resolução do mapa local igual a 5cm (30a), 10cm (30b) e 20cm (30c). O mapa global resultante é semelhante para todas as cenas.



(a) (b) (c)

Figura 31: Segunda cena do ensaio virtual com resolução do mapa local igual a 5cm (31a), 10cm (31b) e 20cm (31c). O mapa global resultante é semelhante para todas as cenas. É possível perceber que vários detalhes do ambiente são perdidos comparando as cenas 31a e 31c.

mapa global. A figura 32 evidencia o efeito da variação do limite. Como a mesa possui uma altura próxima de 1 metro, quando o mapa global está configurado com limite de teto a 60cm o chão abaixo da mesa é considerado navegável. No ensaio com limite de teto de 1 metro a mesa é representada como um objeto sólido.



(a) (b)

Figura 32: Mapa global do segundo ambiente com limite de teto em 60cm (32a) onde o chão sob a mesa é considerado navegável, e em 1 metro (32b) onde a mesa é considerada um obstáculo. A posição da mesa está marcada em azul nas imagens.

Para os ambientes reais, foram utilizados duas seções do laboratório LASCA onde o projeto foi desenvolvido. A figura 33 mostra o ambiente real que foi mapeado utilizando o protótipo do sensor construído preso ao iRobot Roomba.

Do mesmo modo que os ensaios virtuais, o parâmetro de resolução do mapa local foi

mapa global resultante ainda apresenta falhas perto da área final devido a erros acumulados nas transformações dos pontos.

Por fim, o método de mapeamento utilizando compensação por transformação e correção do erro de arredondamento é validado. Os mapas globais gerados por este método estão dispostos na figura 47.

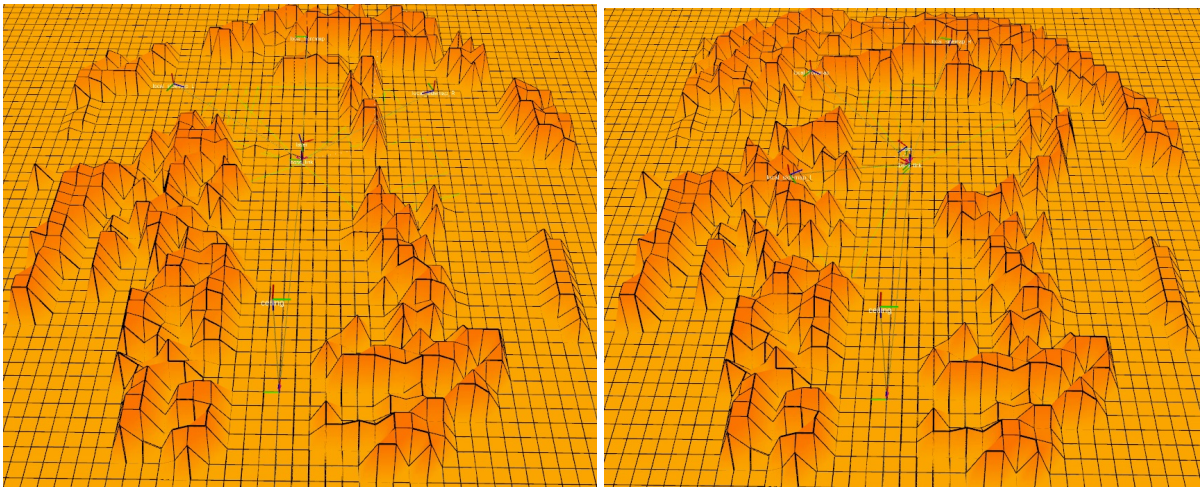
(a) (b)

Figura 47: Mapas globais resultantes da navegação em ambiente virtual utilizando compensação de rotação por transformação utilizando correção de erro. 47a: Antes da rotação final. 47b: Após a rotação final.

O desempenho deste método é superior em relação aos dois outros. O mapa global após a rotação final de 90 graus à esquerda quase não se difere do mapa antes da rotação, indicando este método como o melhor método de compensação de rotação e deslocamento.

Na validação do método de mapeamento em situação real, o sensor foi preso ao iRobot Roomba que percorreu o laboratório (figura 44a) e no fim faz uma curva de 90 graus à esquerda. Começando com o método de compensação por aproximação, os mapas construídos por este método podem ser vistos na figura 48.

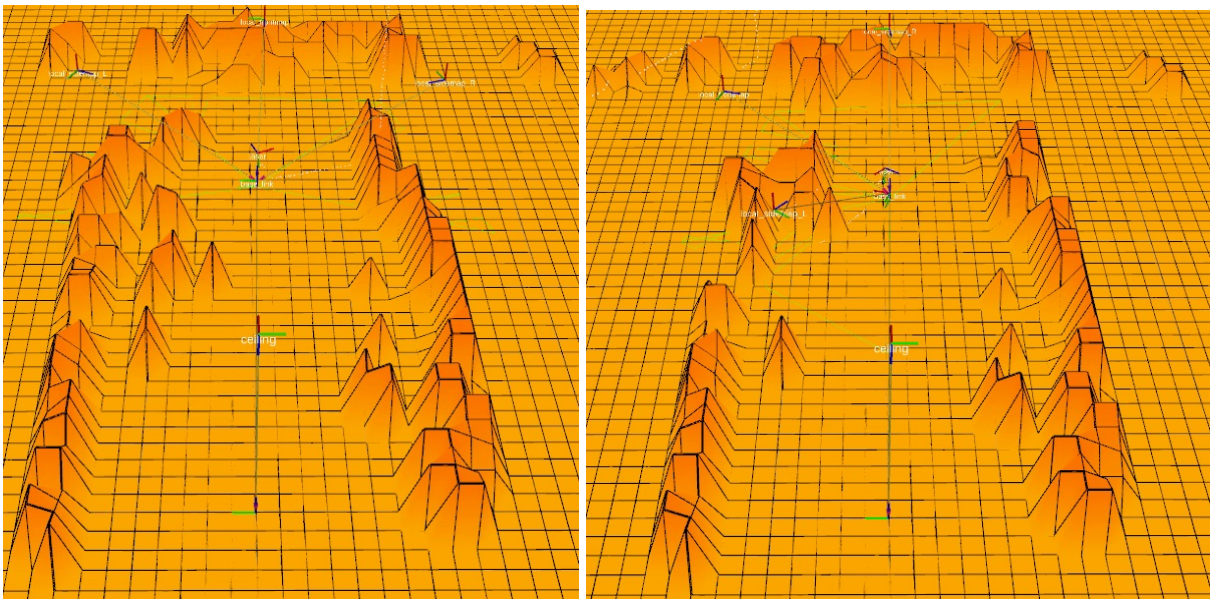
É possível perceber uma diferença grande entre a performance do método aproximado em situação virtual em relação à situação real. Muito mais células ocupadas foram registradas do que o que foi lido realmente pelo robô. A maior parte destas células são pontos assinalados muito longe do grid, onde nem mesmo a aproximação consegue dar coerência após rotações, levando o mapa global a assumir que é um obstáculo real. Muitas das células com valor alto correspondem a obstáculos que foram lidos anteriormente e aproximados com a navegação, porém o erro acumulado leva-os a serem representados longe do robô.



(a) (b)

Figura 48: Mapas globais criados a partir da navegação em ambiente real utilizando método de aproximações. 48a: Antes da rotação final. 48b: Após a rotação final.

Os mapas globais do ensaio real utilizando o método de transformação de pontos do mapa sem utilizar correção de erro estão exibidos na figura 49.

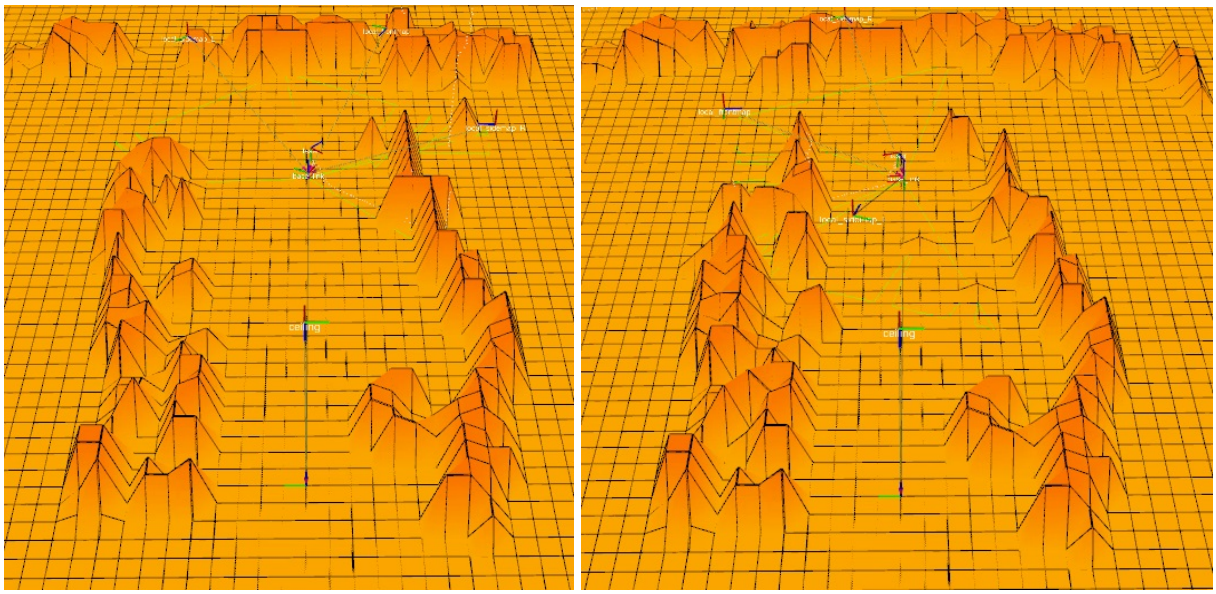


(a) (b)

Figura 49: Mapas globais criados a partir da navegação em ambiente real utilizando compensação de rotação por transformação sem aplicar correção de erro. 49a: Antes da rotação final. 49b: Após a rotação final.

O desempenho do método de transformação sem utilizar compensação de erro foi muito melhor em relação ao método de aproximação. A transformação mantém a estrutura geral do laboratório sem poluir o mapa.

O último ensaio feito para o método de mapeamento principal foi com o método de transformação com correção de erro. Os mapas resultantes deste ensaio podem ser visualizados na figura 50.



(a) (b)

Figura 50: Mapas globais criados a partir da navegação em ambiente real utilizando compensação de rotação por transformação aplicando correção de erro. 50a: Antes da rotação final. 50b: Após a rotação final.

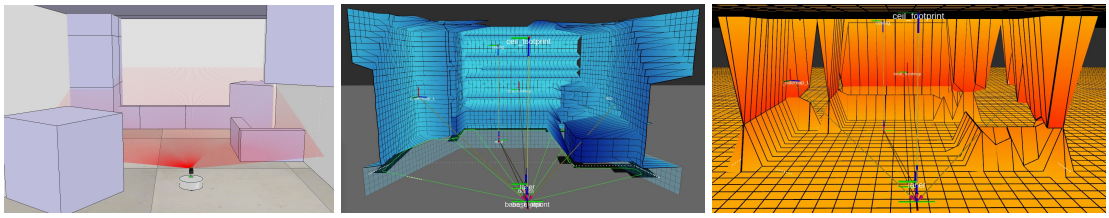
Em um ambiente fechado e com vários obstáculos como o laboratório, fica difícil comparar a performance dos métodos sem e com correção de erro. Porém, baseando-se nos outros ensaios feitos é possível afirmar que o método de compensação de rotação por transformação dos pontos utilizando correção de erro é o que melhor constrói o mapa global. Se o objetivo é construir apenas um mapa local fiel, o método sem compensação de erro deve ser utilizado.

5.4 ENSAIOS ADICIONAIS

Esta seção apresenta ensaios extras feitos com os métodos propostos, detalhando funcionalidades extras desenvolvidas para cobrir uma gama de situações adicional. Nesta seção serão apresentados testes de validação virtuais para dois outros métodos de mapeamento: escaneamento horizontal, com o sensor LIDAR se movendo sentido cima/baixo, e mapeamento omnidirecional, que abre mão da janela deslizante e utiliza um mapa local horizontal junto ao chão e outro junto ao teto para permitir a cobertura e representação total do ambiente. Além destes outros métodos, ensaios utilizando a câmera RGB também são conduzidos.

Utilizar o sensor na horizontal inclinando cima/baixo permite uma frequência de atualização maior para o mapa local, porém sacrifica a capacidade de obter a mesma quantidade de informação para qualquer direção. Na situação atual do método de mapeamento por janela deslizante, não é tirado proveito desta característica de escaneamento, assim uma estratégia de modelar atenção e permitir ao mapa local girar em torno do robô é um dos trabalhos futuros propostos.

Um teste em ambiente virtual foi conduzido para validar a estratégia. O sensor posicionado na vertical calcula os limites de escaneamento para a rotação do motor, sendo ângulo zero o sensor de pé escaneando paralelo ao chão. Para realizar este escaneamento basta ajustar o parâmetro '*mode*' (referente ao motor) do pacote para o valor 'H_WINDOW'. As imagens exibidas na figura 51 representam o ambiente utilizado junto com o robô simulado com o sensor em posição horizontal, bem como a reconstrução deste ambiente escaneado horizontalmente.

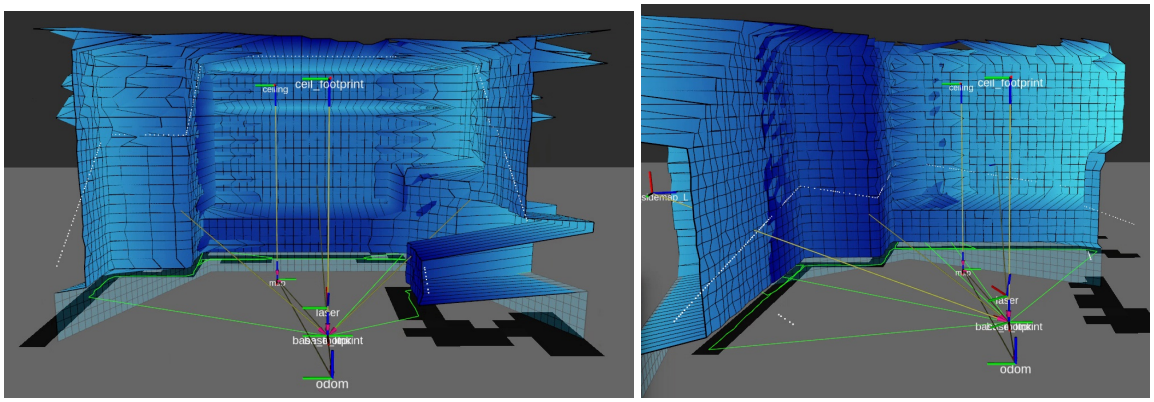


(a) (b) (c)

Figura 51: Ensaio de reconstrução realizado com o sensor posicionado horizontalmente. 51a: cena e robô virtuais utilizados no ensaio, 51b: Mapa local reconstruído a partir do movimento de inclinação do sensor, 51c: Mapa global construído a partir do mapa local.

Ensaios de movimento e rotação também foram feitos. O robô percorreu um metro a frente na cena virtual e então rotacionou 45 graus para a esquerda. Os mapas locais resultantes destes dois passos podem ser visualizados na figura 52. Estes ensaios foram feitos com o método de transformação utilizando correção de erro.

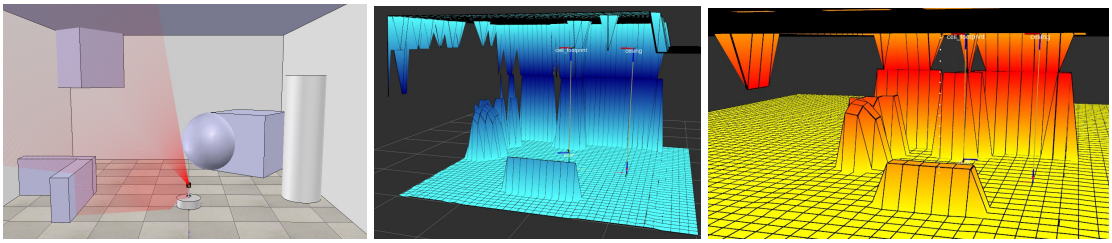
Para o método de escaneamento omnidirecional o robô é configurado com o parâmetro '*mode*' referente ao motor para o valor 'OMNI' fazendo com que o motor alterne entre os valores 0 e 360 graus para ângulo mínimo e máximo respectivamente. Além destas mudanças um novo aquivo '*launch*' deve ser utilizado para carregar as versões omnidirecionais do mapa local e global, bem como o publicador de transformada que cria as referências 'base footprint' e 'ceiling footprint'. Neste modo o sensor escaneia todos os arredores do robô móvel e registra qualquer obstáculo lido que esteja dentro do alcance do mapa local. Estes obstáculos são registrados no mapa diretamente como elevação uma vez que o mapa local agora está posicionado horizontalmente e centrado no robô. O mesmo vale para o mapa local de teto. A



(a) (b)

Figura 52: Mapas locais dos ensaios de navegação com o sensor na horizontal varrendo sentido cima/baixo. 52a: Mapa local após movimento de 1 metro para frente. 52b: Mapa local após rotação de 45 graus a esquerda.

partir destes dois novos grids, a versão omnidirecional do mapa global extrai estas informações de elevação e registra-as nos seus respectivos grids globais. Um mapa de ocupação é gerado a partir do mapa global de chão. Imagens que mostram o ambiente utilizado, a reconstrução do ambiente em mapa local, e o mapa global resultante podem ser visualizados na figura 53.

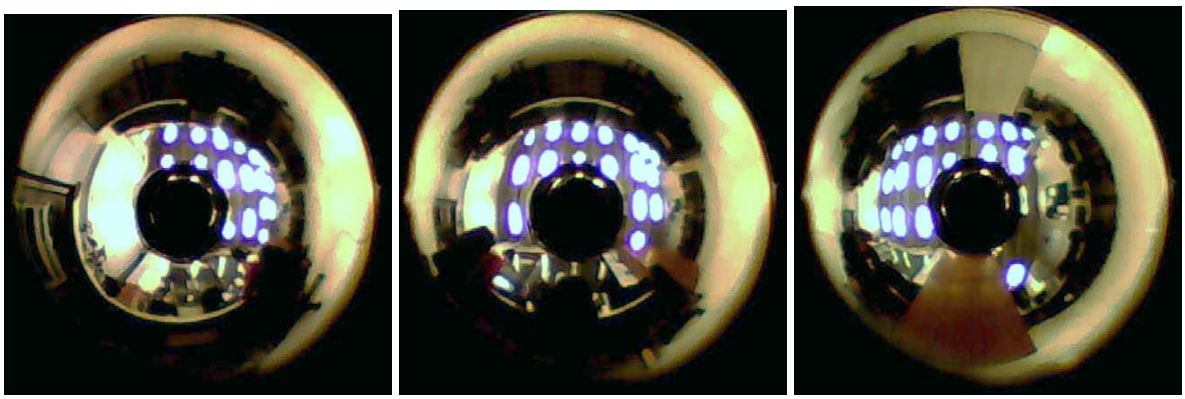


(a) (b) (c)

Figura 53: Ensaio realizado para o método omnidirecional de escaneamento. 53a: cena e robô virtuais utilizados no ensaio, 53b: Mapa local reconstruído a partir método omnidirecional de mapeamento, 53c: Mapa global construído a partir do mapa local.

O movimento do robô no caso omnidirecional é muito mais simples e requer apenas um deslocamento de colunas na horizontal ou vertical dependendo do movimento do robô (em x, y ou ambos) feito a cada vez que o robô andar uma resolução de célula local em uma dimensão.

Apesar da câmera RGB não ter sido utilizada no algoritmo de mapeamento, testes foram realizados em relação à viabilidade de se utilizá-la para adicionar dados de cor aos mapas. A câmera foi ligada durante os testes reais de navegação do método de mapeamento por janela deslizante e imagens foram capturadas no começo, meio e fim do ensaio. A figura 54 mostra as aquisições realizadas com a câmera Snake Eye da Pixart Imaging apontada para o espelho hiperbólico. Esta câmera possui 480x640 pixels de resolução.



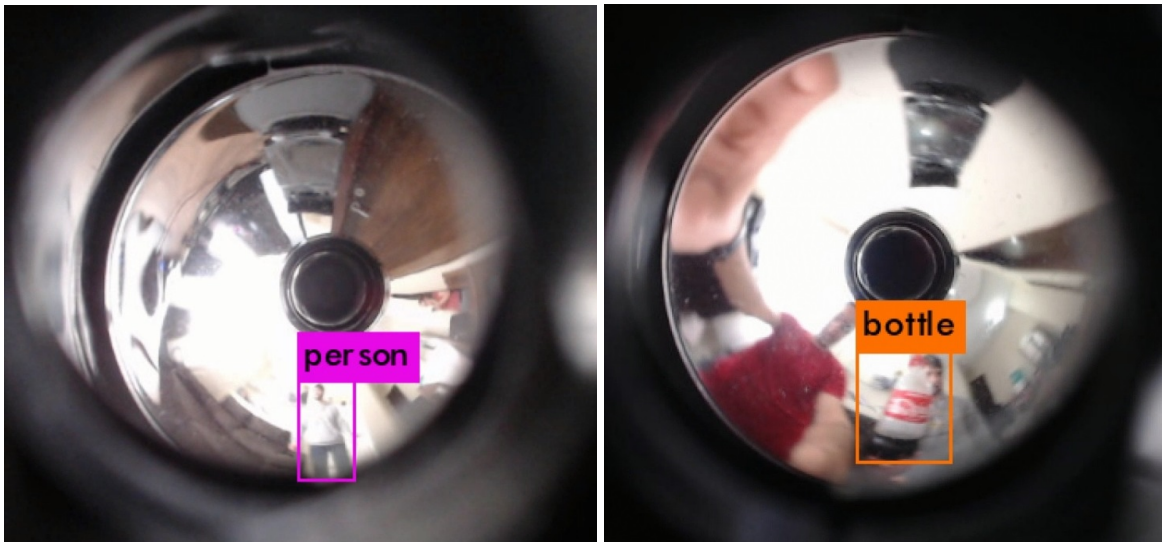
(a) (b) (c)

Figura 54: Capturas realizadas pela câmera RGB fixada no protótipo do sensor durante ensaio de navegação em ambiente real. 54a: Começo do ensaio, 54b: Metade do trajeto, 54c: Final da navegação.

Como é possível notar das imagens, a baixa resolução da câmera é agravada pela captura omnidirecional. O algoritmo de conversão de uma imagem omnidirecional para um panorama retangular, chamado de *dewarp* (NASCIMENTO, 2015), produziria por exemplo uma imagem de 200 pixels de altura (número de pixels de raio da aquisição omnidirecional), o que poderia impactar nos algoritmos de classificação.

Uma tentativa de se aplicar algoritmos de classificação utilizando *deep learning* foi feita utilizando as imagens omnidirecionais, mais precisamente o algoritmo de detecção de objetos YOLO (REDMON et al., 2016). A utilização da câmera original do projeto (Snake Eye) não produziu nenhuma detecção coerente devido a sua baixa resolução, porém ao utilizar uma câmera HD de resolução 1080x1920 pixels, a detecção foi possível. Estes resultados indicam que um algoritmo de *dewarp* aplicado nas imagens omnidirecionais permitem uma detecção e classificação omnidirecional. É válido apontar que mesmo depois da operação de *dewarp*, as imagens ainda ficam levemente distorcidas, e a detecção ainda não seria tão perfeita quanto numa imagem livre de distorções. Exemplos de imagens capturadas junto com o algoritmo de detecção podem ser visualizados na figura 55.

A rede YOLO é muito custosa computacionalmente, requerendo uma GPU para processamento. Nos ensaios foi utilizada uma Nvidia Titan X e a rede YOLO treinada com muitos objetos a serem reconhecidos (quanto mais objetos mais custosa é a identificação). Para o sensor desenvolvido, esta rede seria processada pela Nvidia Tegra K1 que está fixada na base do protótipo, porém a rede precisaria ser retreinada com menos objetos para reconhecimento para manter o processamento em tempo real. Neste ensaio, objetos só são detectados quando na posição de base da aquisição omnidirecional devido ao algoritmo YOLO ser treinado apenas para imagens em posição padrão (não-inclinadas), significando que pessoas de lado, invertidas



(a) (b)

Figura 55: Algoritmo de detecção utilizando YOLO, uma rede neural treinada para detecção de objetos e pessoas. 55a: Detecção de pessoas. 55b: Detecção de obejtos.

ou distorcidas na imagem não são reconhecidas. O ensaio indica que uma câmera HD pode ser a solução para a aplicação de reconhecimento de objetos, permitindo a criação de mapas com dados de objetos classificados e registrados no futuro.

6 CONCLUSÃO

O método de mapeamento por janela deslizante desenvolvido consegue representar ambientes com acurácia, sendo limitado principalmente pelo sensor, uma vez que o Hokuyo URG-04LX é um dos sensores LIDAR com menor frequência de aquisição disponíveis. É comum sensores LIDAR no mercado hoje atingirem 40Hz de frequência de aquisição, o que permitiria o protótipo de sensor desenvolvido atuar quatro vezes mais rápido, trivializando as estratégias empregadas de correção de rotação uma vez que o sensor poderia rapidamente escanear novamente o ambiente. Uma maior frequência de aquisição permitiria também utilizar mapas com maior resolução, diminuindo a resolução de célula para menos de 5cm. A representação em grids torna o mapa resultante muito versátil, pois ferramentas como transformação para nuvem de pontos, mapa de ocupação, atribuição de cor ou transformação para octomap estão disponíveis. A capacidade de reconstrução de ambientes do mapa local possui uma resolução similar a grande parte dos métodos de mapeamento 3D disponíveis e torna possível a utilização de algoritmos de identificação.

Como trabalhos futuros está a utilização da câmera omnidirecional para atribuir cor ao mapa. Adicionalmente, é possível considerar otimizar o mapa para trabalhar também com nuvem de pontos de sensores que já coletem nesta forma de representação, como o Microsoft Kinect. O método de mapeamento atual já funciona para qualquer nuvem de pontos de entradas porém está orientado e otimizado ao sensor proposto. Uma das ideias ocorridas durante o desenvolvimento deste sensor foi de utilizar algum algoritmo que simule atenção, onde o sensor desenvolvido atuaria como um sensor secundário e o mapa local rotacionasse para 'olhar' para pontos de interesse, que então poderiam ser escaneados com mais precisão, representados no mapa e classificados. É possível idealizar um procedimento de SLAM ou de aprimoramento de SLAM para esta estratégia de mapeamento, de modo a tornar o método atual ainda mais independente de outros sensores e dispositivos. Outra direção de pesquisa é a possibilidade de mapear superfícies com elevação diferenciada. O método atual consegue representar rampas porém os dados coletados com o robô inclinado não teriam coerência com o que foi lido pelo sensor, pois suporte a inclinação do robô não foi feito para este trabalho. Além de todas estas

ideias, há ainda a possibilidade de utilizar este método em conjunto com outro, tornando-o complementar a algum método existente como Costmap2D ou Octomap.

No geral, este trabalho abre um leque de possibilidades para aprimorar o método de mapeamento devido a sua estrutura de representação ser muito flexível e computacionalmente leve. O objetivo de reconstruir ambientes em três dimensões utilizando um sensor LIDAR giratório foi atingido, bem como utilizar essa reconstrução para montar um mapa que seja compreensível para a maioria das aplicações robóticas. Este trabalho serviu seu propósito de aprimorar o conhecimento sobre mapeamento e robótica e atingiu os objetivos propostos.

REFERÊNCIAS

ALISMAIL, H.; BAKER, L. D.; BROWNING, B. Continuous trajectory estimation for 3d slam from actuated lidar. In: IEEE. **Robotics and Automation (ICRA), 2014 IEEE International Conference on**. 2014. p. 6096–6101.

BEIRA, R. et al. Design of the robot-cub (icub) head. In: IEEE. **Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on**. 2006. p. 94–100.

BELBACHIR, A. N. et al. A novel hdr depth camera for real-time 3d 360 panoramic vision. In: IEEE. **Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on**. 2014. p. 425–432.

BELTER, D.; SKRZYPCZYNSKI, P. Precise self-localization of a walking robot on rough terrain using parallel tracking and mapping. **Industrial Robot: An International Journal**, Emerald Group Publishing Limited, v. 40, n. 3, p. 229–237, 2013.

BORRMANN, D. et al. A mobile robot based system for fully automated thermal 3d mapping. **Advanced Engineering Informatics**, Elsevier, v. 28, n. 4, p. 425–440, 2014.

BOSSE, M.; ZLOT, R. Continuous 3d scan-matching with a spinning 2d laser. In: IEEE. **Robotics and Automation, 2009. ICRA'09. IEEE International Conference on**. 2009. p. 4312–4319.

CAO, T.-T. et al. A gpu accelerated algorithm for 3d delaunay triangulation. In: ACM. **Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games**. 2014. p. 47–54.

CHEN, H. et al. Pose-graph based 3d map fusion with distributed robot system. In: IEEE. **Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on**. 2014. p. 1608–1613.

CHEN, S. Kalman filter for robot vision: a survey. **IEEE Transactions on Industrial Electronics**, IEEE, v. 59, n. 11, p. 4409–4420, 2012.

CHOU, Y.-S.; LIU, J.-S. A robotic indoor 3d mapping system using a 2d laser range finder mounted on a rotating four-bar linkage of a mobile platform. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 10, n. 1, p. 45, 2013.

COPPELIA. **V-Rep: Virtual Robot Experimentation Platform (Coppelia Robotics GmbH)**. 2018. Disponível em: <http://www.coppeliarobotics.com/>.

DALMEDICO, N. et al. Gpu and ros the use of general parallel processing architecture for robot perception. In: **Robot Operating System (ROS)**. : Springer, 2019. p. 407–448.

DIMETIX. **Manual EDS-C Laser Distance Sensor**. 2018. Disponível em: <https://dimetix.com/wp-content/uploads/2016/07/EDS-C-ManualV1.4.pdf>.

ENDRES, F. et al. An evaluation of the rgb-d slam system. In: IEEE. **Robotics and Automation (ICRA), 2012 IEEE International Conference on**. 2012. p. 1691–1696.

ENGEL, J.; SCHÖPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: SPRINGER. **European Conference on Computer Vision**. 2014. p. 834–849.

ERDOGAN, C.; PALURI, M.; DELLAERT, F. Planar segmentation of rgbd images using fast linear fitting and markov chain monte carlo. In: IEEE. **Computer and Robot Vision (CRV), 2012 Ninth Conference on**. 2012. p. 32–39.

FANKHAUSER, P. et al. Robot-centric elevation mapping with uncertainty estimates. In: **Mobile Service Robotics**. : World Scientific, 2014. p. 433–440.

FANKHAUSER, P.; HUTTER, M. A universal grid map library: Implementation and use case for rough terrain navigation. In: KOUBAA, A. (Ed.). **Robot Operating System (ROS) – The Complete Reference (Volume 1)**. Springer, 2016. cap. 5. ISBN 978-3-319-26052-5. Disponível em: <http://www.springer.com/de/book/9783319260525>.

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.

GUPTA, S. et al. Cognitive mapping and planning for visual navigation. **arXiv preprint arXiv:1702.03920**, v. 3, 2017.

HANDA, A. et al. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In: IEEE. **Robotics and automation (ICRA), 2014 IEEE international conference on**. 2014. p. 1524–1531.

HENRY, P. et al. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: SPRINGER. **Experimental robotics**. 2014. p. 477–491.

HORNUNG, A. et al. Octomap: An efficient probabilistic 3d mapping framework based on octrees. **Autonomous Robots**, Springer, v. 34, n. 3, p. 189–206, 2013.

HOUSHIAR, H. et al. A study of projections for key point based registration of panoramic terrestrial 3d laser scan. **Geo-spatial Information Science**, Taylor & Francis, v. 18, n. 1, p. 11–31, 2015.

IMAGING, M. **SR4000 data sheet**. Rev, 2010. Disponível em: <https://www.inf.u-szeged.hu/sites/default/files/ipcglab/docs/vision/SR4000_Manual.pdf>.

JIA, T. et al. 3d depth information extraction with omni-directional camera. **Information Processing Letters**, Elsevier, v. 115, n. 2, p. 285–291, 2015.

KLIMENTJEW, D.; HENDRICH, N.; ZHANG, J. Multi sensor fusion of camera and 3d laser range finder for object recognition. **Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI) (pp. 236-241)**, IEEE, 2010.

KNEIP, L. et al. Characterization of the compact hokuyo urg-04lx 2d laser range scanner. In: IEEE. **Robotics and Automation, 2009. ICRA'09. IEEE International Conference on**. 2009. p. 1447–1454.

KOENIG, N. P.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: CITESEER. **IROS**. 2004. v. 4, p. 2149–2154.

KURNIANGGORO, L.; HOANG, V.-D.; JO, K.-H. Calibration of a 2d laser scanner system and rotating platform using a point-plane constraint. **Computer Science and Information Systems**, v. 12, n. 1, p. 307–322, 2015.

LEE, D.; KIM, H.; MYUNG, H. Gpu-based real-time rgb-d 3d slam. In: IEEE. **Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on**. 2012. p. 46–48.

LIN, S.-S.; BAJCSY, R. High resolution catadioptric omni-directional stereo sensor for robot vision. In: IEEE. **Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on**. 2003. v. 2, p. 1694–1699.

MAIER, D.; HORNUNG, A.; BENNEWITZ, M. Real-time navigation in 3d environments based on depth camera data. In: IEEE. **Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on**. 2012. p. 692–697.

MAXON. **Maxon motors catalog page**. 2018. Disponível em: <https://www.maxonmotor.com/maxon/view/catalog/>.

MAXONMOTOR. **EC 32 Brushless 80Watt with Hall sensors**. 2018. Disponível em: <https://www.maxonmotor.com/maxon/view/product/motor/ecmotor/ec/ec32/118889>.

MAXONMOTOR. **EPOS2 24/5, Digital positioning controller, 5 A, 11 - 24 VDC**. 2018. Disponível em: <https://www.maxonmotor.com/maxon/view/product/control/Positionierung/367676>.

MEILLAND, M.; COMPORT, A. I.; RIVES, P. Dense omnidirectional rgb-d mapping of large-scale outdoor environments for real-time localization and autonomous navigation. **Journal of Field Robotics**, Wiley Online Library, v. 32, n. 4, p. 474–503, 2015.

MEN, H.; GEBRE, B.; POCHIRAJU, K. Color point cloud registration with 4d icp algorithm. In: IEEE. **Robotics and Automation (ICRA), 2011 IEEE International Conference on**. 2011. p. 1511–1516.

MERRIAUX, P. et al. Robust robot localization in a complex oil and gas industrial environment. **Journal of Field Robotics**, Wiley Online Library, v. 35, n. 2, p. 213–230, 2018.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. **IEEE Transactions on Robotics**, IEEE, v. 31, n. 5, p. 1147–1163, 2015.

NASCIMENTO, A. A. do. **Análise de Sistemas de Aquisição de Imagens Omnidirecionais para Navegação em Robótica Móvel**. Dissertação (mathesis) — Universidade Federal da Bahia, 2015. Disponível em: <http://www.ppgee.eng.ufba.br/teses/6ae3ca275b43cfcf8c406631c1437f07.pdf>.

NEMOTO, Z.; TAKEMURA, H.; MIZOGUCHI, H. Development of small-sized omni-directional laser range scanner and its application to 3d background difference. In: IEEE. **Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE**. 2007. p. 2284–2289.

NGUYEN, T.; REITMAYR, G. Calibrating setups with a single-point laser range finder and a camera. In: IEEE. **Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on**. 2013. p. 1801–1806.

NIETO-GRANDA, C.; III, J. G. R.; CHRISTENSEN, H. I. Coordination strategies for multi-robot exploration and mapping. **The International Journal of Robotics Research**, SAGE Publications Sage UK: London, England, v. 33, n. 4, p. 519–533, 2014.

PAGLIARI, D.; PINTO, L. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 15, n. 11, p. 27569–27589, 2015.

PARK, C.-S. et al. Characterization of the hokuyo ubg-04lx-f01 2d laser rangefinder. In: IEEE. **RO-MAN, 2010 IEEE**. 2010. p. 385–390.

POULIN, M. **LEDs and sensing technology enable new ranging applications**. 2014. Disponível em: <https://www.ledsmagazine.com/articles/print/volume-11/issue-6/features/developer-forum/leds-and-sensing-technology-enable-new-ranging-applications.html>.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 779–788.

ROS.ORG. **Why ROS? Core Components**. 2016. Disponível em: <http://www.ros.org/core-components/>.

SCHERER, S. A.; ZELL, A. Efficient onbard rgbd-slam for autonomous mavs. In: IEEE. **Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on**. 2013. p. 1062–1068.

SCHINDELIN, J. **Delaunay and Voronoi**. 2008. ImageJ. Disponível em: <https://imagej.net/Delaunay_Voronoi>.

SCHMID, K. et al. Stereo vision based indoor/outdoor navigation for flying robots. In: IEEE. **Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on**. 2013. p. 3955–3962.

SELBY, W. **2D Mapping & Navigation**. 2016. Disponível em: <https://www.wilselby.com/research/ros-integration/2d-mapping-navigation/>.

SICK. **2D LiDAR sensors LMS5xx**. 2018. Disponível em: <https://www.sick.com/de/en/detection-and-ranging-solutions/2d-lidar-sensors/lms5xx/lms500-20000s01/p/p476045>.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. Autonomous mobile robots. **Massachusetts Institute of Technology**, 2004.

SIEGWART, R. et al. **Introduction to autonomous mobile robots**. : MIT press, 2011.

STEINBRÜCKER, F.; STURM, J.; CREMERS, D. Volumetric 3d mapping in real-time on a cpu. In: IEEE. **Robotics and Automation (ICRA), 2014 IEEE International Conference on**. 2014. p. 2021–2028.

TAGUCHI, Y. et al. Point-plane slam for hand-held 3d sensors. In: IEEE. **Robotics and Automation (ICRA), 2013 IEEE International Conference on**. 2013. p. 5182–5189.

UKIDA, H. et al. Omni-directional 3d measurement by hyperbolic mirror cameras and pattern projection. In: IEEE. **Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE**. 2008. p. 365–370.

WEIKERSDORFER, D. et al. Event-based 3d slam with a depth-augmented dynamic vision sensor. In: IEEE. **Robotics and Automation (ICRA), 2014 IEEE International Conference on**. 2014. p. 359–364.

WURM, K. M. et al. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In: **Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation**. 2010. v. 2.

XU, J. et al. An omnidirectional 3d sensor with line laser scanning. **Optics and Lasers in Engineering**, Elsevier, v. 84, p. 96–104, 2016.

ZHANG, C. et al. Development of an omni-directional 3d camera for robot navigation. In: IEEE. **Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on**. 2012. p. 262–267.

ZHANG, J.; SINGH, S. Loam: Lidar odometry and mapping in real-time. In: **Robotics: Science and Systems**. 2014. v. 2.