CENTRO UNIVERSITÁRIO FEI

FLÁVIO BARBOSA VIEIRA

PARÂMETROS DE SENSORES EM ROBÓTICA MÓVEL PARA EXPLORAÇÃO AUTÔNOMA DE AMBIENTES INTERNOS

São Bernardo do Campo - SP
2020

FLÁVIO BARBOSA VIEIRA

PARÂMETROS DE SENSORES EM ROBÓTICA MÓVEL PARA EXPLORAÇÃO AUTÔNOMA DE AMBIENTES INTERNOS

Dissertação de Mestrado, apresentada ao Centro Universitário da FEI para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Plinio Thomaz Aquino Junior.

São Bernardo do Campo - SP
2020

Vieira, Flávio Barbosa.
Parâmetros de sensores em robótica móvel para exploração autônoma de ambientes internos. / Flávio Barbosa Vieira. São Bernardo do Campo, 2020.
104 f. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Plinio Thomaz Aquino Junior .

1. SLAM. 2. Fusão de Sensores. 3. Robô de Serviço. 4. Exploração. 5. Autônomo. I. Aquino Junior , Plinio Thomaz, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Flavio Barbosa Vieira **Matrícula:** 117305-3

Título do Trabalho: Parâmetros de sensores em robótica móvel para exploração autônoma de ambientes internos.

Área de Concentração: Inteligência Artificial Aplicada à Automação e Robótica

Orientador: Prof. Dr. Plinio Thomaz Aquino Junior

Data da realização da defesa: 01/10/2020

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

_____

_____

_____

_____

_____

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Plinio Thomaz Aquino Junior	Ass.: _____
Prof. Dr. Flavio Tonidandel	Ass.: _____
Prof. Dr. João Alberto Fabro	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Dedico este texto a minha esposa Marcia que me apoia em todos os momentos e decisões da minha vida.

AGRADECIMENTOS

Agradeço a todos os professores e funcionários da FEI que deram todo o suporte para que fosse possível a realização deste trabalho. Agradeço também à CAPES que viabilizou o meu ingresso no mestrado, contribuindo fortemente para concretização do meu sonho.

RESUMO

A área de robótica móvel tem se desenvolvido de maneira notável nos últimos anos, diversos pesquisadores são motivados pela crescente demanda por essa tecnologia e pela infinidade de aplicações. Competições de robótica fomentam novos desafios ao considerar diferentes cenários de aplicações para a robótica de serviço, como por exemplo a RoboCup@Home que determina regras para que robôs autônomos e inteligentes sejam avaliados durante a execução de tarefas em cenários domésticos ou públicos. O presente trabalho concentra-se na resolução do problema de exploração de ambientes internos residenciais desconhecidos. Para fazer isso, o robô deve coletar informações externas e internas através de sensores, para fundir estes dados e interpretá-los de maneira eficiente, possibilitando se localizar através de algoritmos probabilísticos, mapeando o ambiente simultaneamente, e navegar pelo ambiente mapeado evitando colisões. O trabalho estuda e testa a configuração de sensores de distância (lasers, sonares e câmeras) e técnicas de exploração, disponíveis e compartilhados na comunidade ROS, para garantir que o robô consiga explorar o ambiente residencial completamente buscando a otimização do tempo necessário, da distância percorrida e da rotação realizada. Os pacotes de exploração testados são: *explore-lite*, *RRT-exploration* e *cam-exploration*. A variação de sensores foi determinante para entender as vantagens e desvantagens de utilizar o laser Lidar e as câmeras de profundidade em diferentes combinações. Sendo assim, os resultados mostram que o aumento do número de sensores não melhora os desempenhos na exploração em todas as condições. O trabalho conclui que tanto o *explore-lite* quanto o *RRT-exploration* possuem bons desempenhos em todas as condições propostas e indica as melhores montagens de sensores para cada pacote. Dessa forma, criou-se o pacote para implementação da exploração autônoma no Robô HERA.

Palavras-chave: SLAM, Fusão de Sensores, Robô de Serviço, Exploração, Autônomo

ABSTRACT

.

The area of mobile robotics has developed remarkably in recent years, many researchers are motivated by the growing demand for this technology and the variety of applications. Robotics competitions foster new challenges when considering diverse application scenarios for service robotics, such as RoboCup@Home, which sets rules for autonomous and intelligent robots to be evaluated while performing tasks in domestic or public scenarios. The present work focuses on solving the problem of exploring unknown residential indoor environments. To do this, the robot must collect external and internal information through sensors, to fuse this data and interpret it efficiently, making it possible to locate itself through probabilistic algorithms, simultaneously mapping the environment, and navigate the mapped environment avoiding collisions. The work studies and tests the configuration of distance sensors (lasers, sonar and cameras) and exploration techniques, available and shared in the ROS community, to ensure that the robot is able to fully explore the residential environment in order to optimize the necessary time, distance traveled and the rotation performed. The tested exploration packages are: *explore-lite*, *RRT-exploration* and *cam-exploration*. The variation of sensors was crucial to understand the advantages and disadvantages of using the Lidar laser and depth cameras in different combinations. Thus, the results show that the increase in the number of sensors does not improve performance in exploration in all conditions. The work concludes that both *explore-lite* and *RRT-exploration* perform well in all proposed conditions and indicate the best sensor assemblies for each package. Thus, a package was created for the implementation of autonomous exploration in the HERA Robot.

Keywords: SLAM, Sensor Fusion, Service Robot, Exploration, Autonomous

LISTA DE ILUSTRAÇÕES

Ilustração 1 – Diagrama com a composição das tarefas necessárias para realização da exploração 18
Ilustração 2 – Gráfico que mostra o crescimento da demanda por robôs domésticos de serviço desde 2017 e estimativas até o fim do ano de 2019 e 2020 a 2022. 20
Ilustração 3 – Robô HERA, da equipe RoboFEI para competições @Home. 20
Ilustração 4 – Etapas do processo de identificação do estado s da i-ésima célula C_i utilizando o modelo do sensor e o teorema probabilístico de Bayes. 33
Ilustração 5 – Reconstrução de um ambiente de escritório em 3D utilizando técnica de fusão de elementos de superfície com pontos obtidos por um dispositivo manual giratório de laser LiDAR. 34
Ilustração 6 – Representação topológica de um ambiente utilizando GVG. 35
Ilustração 7 – Representação puramente topológica de um ambiente após sua completa exploração. 36
Ilustração 8 – Ilustração do funcionamento do algoritmo MCL, onde os pontos pretos representam as partículas: a) Início da operação com as partículas distribuídas aleatoriamente; b) após se deslocar por alguns metros o algoritmo começa a aglomerar partículas em grupos separados após atribuição dos pesos; c) depois de um certo tempo de operação o algoritmo tende a aglomerar as partículas na posição correta do robô com uma margem de erro relativamente pequena. 42
Ilustração 9 – A imagem mostra o resultado de uma simulação da aplicação do algoritmo Dijkstra para planejamento global do percurso. Os pontos O e G representam os pontos inicial e final, respectivamente. Os grafos conectados por v_i e $v_(i+1)$ representam os obstáculos de maneira simplificada. As setas mostram o percurso escolhido pelo robô até o ponto objetivo. 44
Ilustração 10 – O mapa em questão é a representação do ambiente fragmentado em grades, no qual o quadrado azul representa o nó inicial, os azuis claros os nós possíveis em formato de grades, os pretos são os obstáculos, as numerações são os custos atrelados, o quadrado verde é o nó objetivo e o caminho em vermelho é o percurso ótimo escolhido para o caso específico. 45
Ilustração 11 – Representação da simulação da aplicação do algoritmo DWA para planejamento local da trajetória. 47
Ilustração 12 – Correção da trajetória utilizando o algoritmo Eband. 48

Ilustração 13 – Exemplo da configuração de camadas de *costmap*. A camada *Static* é a representação do mapa estático, a camada *Obstacles* é a representação da leitura dos sensores de distância quando encontram-se obstáculos, a camada *Inflation* é a camada que projeta a representação das outras camadas, na qual as outras são infladas para representar a configuração espacial do robô, e por fim, a camada *Master* que une todas a informações obtidas em todas as camadas separadas para representação final do *costmap2D*. . 50
Ilustração 14 – a) Mostra um MGO, onde os espaços em branco são as áreas conhecidas (espaços abertos), os pontos menores são as áreas desconhecidas (espaços desconhecidos) e os pontos maiores são os limites conhecidos. b) Destaque dos limites que separam os espaços abertos dos espaços desconhecidos. c) Destaque das regiões conhecidas como fronteiras candidatas à exploração. 53
Ilustração 15 – Passos da exploração de um escritório, utilizando FBE. 53
Ilustração 16 – a) Representação dos quatro pontos já explorados e das fronteiras (*free edges*). b) Criação dos pontos candidatos à exploração q a partir do ponto q_4 com visibilidade para as regiões desconhecidas. c) Representação de $A(q)$ (potencial ganho de informação) do ponto q selecionado. d) seleção do ponto NVB que teve melhor pontuação $g(q)$ 55
Ilustração 17 – Representação do ambiente com MGO e Mapeamento topológico (TTM do inglês *Thining Topological Map*). 56
Ilustração 18 – EBE utilizando a representação em GVG 58
Ilustração 19 – Resultado experimental da aplicação das técnicas *Frontier Based Exploration*, *Next Best View Exploration*, *Thining Based Topological Exploration* e *EKF Based Exploration* na exploração de um ambiente desconhecido. . 60
Ilustração 20 – a) Representação da lateral do HERA no simulador Gazebo, indicando as posições dos sensores lasers e das câmeras Xtion e Kinect. b) Posição dos sensores ultrassônicos a partir da vista superior. 63
Ilustração 21 – Arquitetura do tratamento dos dados provenientes dos sensores para aplicação do pacote *gmapping* para a realização da tarefa do SLAM. 65
Ilustração 22 – Arquitetura do algoritmo *explore-lite*. 66
Ilustração 23 – Visão do funcionamento e princípios do *explore-lite* através do *rviz*. . . . 67
Ilustração 24 – Visão macro da estrutura de funcionamento do pacote *RRT-exploration*. . 68
Ilustração 25 – Representação da criação das árvores de exploração *RRT* global e local para busca das fronteiras, sendo que em vermelho é a representação da detecção local e em azul da detecção global, e ambos estão em um MGO. 68

Ilustração 26 – Arenas utilizadas para os testes com a marcação "ABC"que indicam os pontos a partir dos quais o robô inicia a exploração. a) Arena 1: disposição mais simples, sem obstáculos pequenos e passagens relativamente largas entre cômodos. b) Arena 2: simulação de uma residência comum com passagens entre cômodos equivalentes ao tamanho de portas padrão e móveis pequenos espalhados pelo caminho. 71
Ilustração 27 – Disposição do ambiente residencial (arena 2) visto pelo Gazebo com o robô no ponto inicial "A". 72
Ilustração 28 – Configurações dos sensores de distância nos testes realizados. Os sonares estão inclusos em todas as configurações para a navegação. a) Montagem mais básica, com o laser frontal para SLAM e navegação e a câmera Xtion para navegação. b) Montagem com os lasers frontal e traseiro em conjunto para SLAM e a câmera Xtion em conjunto com o laser frontal para navegação. c) Montagem com o laser frontal e a câmera Kinect apontada para trás para o SLAM e o laser frontal em conjunto com a câmera Xtion para a navegação. d) Montagem com a câmera Xtion para o SLAM e navegação e o laser traseiro para SLAM. 75
Ilustração 29 – Representação do robô utilizando o sonar para identificar a distância até o obstáculo no ambiente. 77
Ilustração 30 – Gráficos com os resultados da aplicação do pacote *explore-lite* nas 3 configurações propostas. Com a distância em metros, a rotação em radianos e o tempo em minutos. 80
Ilustração 31 – Gráficos com os resultados da aplicação do pacote *RRT-exploration* nas 3 configurações propostas. Com a distância em metros, a rotação em radianos e o tempo em minutos. 82
Ilustração 32 – Gráficos com os resultados da aplicação do pacote *cam-exploration* nas 3 configurações propostas. Com a distância em metros, a rotação em radianos e o tempo em minutos. 83
Ilustração 33 – Mapas criados com o pacote *gmapping* após a atuação dos pacotes de exploração: a) representação da arena 1 e b) representação da arena 2. . . 83
Ilustração 34 – Ponto indicando local na arena 2 que impossibilitou a exploração autônoma utilizando a configuração de sensores 4 com o Xtion frontal e o laser traseiro realizando o SLAM. 84
Ilustração 35 – Gráficos com os resultados da aplicação do pacote *explore-lite* com 4 diferentes montagens de sensores na arena 1 e com 3 montagens de sensores na arena 2. Com a distância em metros, a rotação em radianos e o tempo em minutos. 86

Ilustração 36 – Gráficos com os resultados da aplicação do pacote *RRT-exploration* com 4 diferentes montagens de sensores na arena 1 e com 3 montagens de sensores na arena 2. Com a distância em metros, a rotação em radianos e o tempo em minutos. 87
Ilustração 37 – Vista superior no Gazebo do robô HERA, com os sensores laser frontal e traseiro ativos, ilustrando o alcance do ângulo de abertura para aplicação no mapeamento. 88
Ilustração 38 – Gráficos com os resultados da aplicação do pacote *cam-exploration* com 3 diferentes montagens de sensores na arena 1. Com a distância em metros, a rotação em radianos e o tempo em minutos. 89
Ilustração 39 – Percursos realizados dado um ponto inicial P até o ponto final F. a) Teste utilizando o mapa criado e se localizando através do *AMCL*; b) teste utilizando SLAM através do *gmapping* com a porta de entrada fechada; c) teste utilizando SLAM através do *gmapping* com a porta de entrada aberta. 94

LISTA DE TABELAS

Tabela 1 – Características básicas dos sensores RGB-D, ultrassônico e a laser. 25
Tabela 2 – Taxonomia de Problemas de localização em robótica. 29
Tabela 3 – Quantidade de testes planejados para cada configuração diferente, variando-se tanto os parâmetros dos pacotes de exploração quanto a disposição dos sensores. A quantidade final é por pacote de exploração por arena. 74
Tabela 4 – *explore-lite*: Variação da configuração dos parâmetros do pacote de exploração. 79
Tabela 5 – *RRT-exploration*: variação da configuração dos parâmetros do pacote de exploração. 81
Tabela 6 – *cam-exploration*: Variação da configuração dos parâmetros do pacote de exploração. 81
Tabela 7 – Resumo das melhores configurações dos parâmetros dos pacotes de exploração para cada pacote. 84
Tabela 8 – Resumo da configuração dos sensores com o *explore-lite* para os testes nas duas arenas. 85
Tabela 9 – Resumo da configuração dos sensores com o *RRT-exploration* para os testes nas 2 arenas. 85
Tabela 10 – Resumo da configuração dos sensores com o *cam-exploration* para os testes nas duas arenas. 88
Tabela 11 – Resumo das quantidades de testes planejados em relação aos testes realizados. 89
Tabela 12 – Resultados das distâncias percorridas por pacote de exploração para as 2 arenas disponíveis em metros 90
Tabela 13 – Resultados das rotações realizadas por pacote de exploração para as 2 arenas disponíveis em radianos 91
Tabela 14 – Resultados dos tempos de execução por pacote de exploração para as 2 arenas disponíveis em minutos 91

LISTA DE ALGORITMOS

Algoritmo 1 – Pseudo-código que aplica o filtro de Bayes, de acordo com Thrun, Burgard e Fox (2005). 26
Algoritmo 2 – Pseudo-código que aplica a Localização de Markov, de acordo com Thrun, Burgard e Fox (2005). 30
Algoritmo 3 – Pseudo-código da aplicação do filtro de Partículas da localização de Monte Carlo, de acordo com Thrun, Burgard e Fox (2005). 41
Algoritmo 4 – Pseudo-código da aplicação do algoritmo *RRT* para navegação (LAVALLE, 1998) 49

LISTA DE ABREVIATURAS

AMCL *Adaptive Monte Carlo Localization* (Localização Adaptativa de Monte Carlo).

EBE *Ear Based Exploration*.

EKF *Extended Kalman Filter* (Filtro de Kalman Estendido).

FBE *Frontier Based Exploration* (Exploração Baseada em Fronteiras).

FEI Centro Universitário da FEI.

GVG Grafo de Voronoi Generalizado.

HERA *Home Environment Robot Assistant* (Assitente Robótico em Ambientes Domésticos).

KF *Kalman Filter* (Filtro de Kalman).

MGO Mapas de Grades de Ocupação.

RBPF *Rao-Blackwellized Particle Filter* (Filtro de Partículas Rao-Blackwellized).

RRT *Rapdly-exploring Random Tree* (Árvore de Rápida Exploração Aleatória).

SLAM *Simultaneous Localization and Mapping* (Localização e Mapeamento Simultâneos).

TBTE *Thining Based Topological Exploration*.

SUMÁRIO

1 Introdução . . . 17
1.1 Contextualização . . . 19
1.2 Contribuição . . . 20
1.3 Objetivos . . . 21
1.3.1 Objetivo Geral . . . 21
1.3.2 Objetivos Específicos . . . 21
1.4 Organização do Texto . . . 22
2 Fundamentação Teórica . . . 23
2.1 Sensoriamento . . . 23
2.1.1 Definições . . . 23
2.1.2 Discussão . . . 24
2.2 Algoritmos Probabilísticos em Robótica . . . 26
2.2.1 Filtro de Bayes . . . 26
2.2.2 Tipos de Filtros . . . 27
2.3 Localização . . . 27
2.3.1 Taxonomia de Problemas de Localização . . . 28
2.3.2 Localização de Markov . . . 29
2.4 SLAM . . . 30
2.5 Mapeamento . . . 31
2.5.1 Mapeamento em 2D - Mapas de Gade de Ocupação . . . 32
2.5.2 Mapeamento em 3D - Fusão de Elementos de Superfície e *Visual* SLAM . . . 33
2.5.3 Mapeamento Topológico . . . 35
2.6 Fusão de Sensores para Localização e Mapeamento . . . 37
2.6.1 Filtro de Kalman (KF) . . . 38
2.6.2 Filtro de Kalman Estendido (EKF) . . . 39
2.6.3 Filtros de Partículas . . . 41
2.7 Navegação . . . 43
2.7.1 A* e Djikstra: Planejamento Global . . . 44
2.7.2 DWA e Eband: Planejamento Local . . . 46
2.7.3 Planejamento de trajetória com *RRT* . . . 48
2.7.4 *costmap2D* . . . 49
3 Exploração Autônoma . . . 52
3.1 Técnicas de Exploração . . . 52
3.1.1 Exploração Baseada em Fronteiras . . . 52
3.1.2 Exploração Baseada em Ganhos . . . 52
3.1.3 Exploração com Mapeamento Topológico . . . 56
3.1.4 Abordagens Diversas . . . 58

4 Metodologia 61
4.1 Robocup e Plataforma 61
4.1.1 Regras ROBOCUP@HOME 61
4.1.2 Plataforma 62
4.2 ROS (*Robot Operating System*) 63
4.3 Pacotes Auxiliares 64
4.4 Algoritmos de Exploração 65
4.4.1 *explore-lite* 65
4.4.2 *RRT-exploration* 67
4.4.3 *cam-exploration* 70
4.5 Estratégias de avaliação 71
5 Experimentos 76
5.1 Sensores para o SLAM 77
5.2 Resultados experimentais 78
5.2.1 Etapa de Calibragem 79
5.2.2 Etapa de Avaliação dos Sensores 84
5.2.3 Resumo dos Resultados Obtidos 89
5.2.4 Vantagens da Exploração Autônoma 93
6 Conclusão e Trabalhos Futuros 96
6.1 Trabalhos Futuros 97
REFERÊNCIAS 98

1 INTRODUÇÃO

A área de robótica móvel tem se desenvolvido de maneira notável nos últimos anos, principalmente na resolução do problema de localização e mapeamento simultâneos (SLAM - *Simultaneous Localization and Mapping*) (TOROSLU; DOĞAN, 2018) (MAITY; SAHA; BHOWMICK, 2017) (BROSSARD; BONNABEL; BARRAU, 2018). Diversos pesquisadores são motivados pela crescente demanda por essa tecnologia e pela infinidade de aplicações, como tarefas em ambientes domésticos (TSUN et al., 2016), transportes autônomos de passageiros (Y. CHEN et al., 2018), em ambientes de alta periculosidade para os humanos (WANG et al., 2017), em ambientes de difícil acesso (LOSCH et al., 2018) ou na agricultura (CHRISTIANSEN, 2018).

Da mesma forma que seres vivos utilizam os sentidos para entenderem o ambiente no qual estão inseridos para tomarem decisões, os robôs autônomos precisam coletar os dados do ambiente através de sensores para resolverem diversos tipos de tarefas. Com os dados obtidos, os robôs podem realizar diversas tarefas em um ambiente qualquer, entretanto quando o robô executa ações ele necessita deslocar-se no espaço, e em muitas aplicações é necessário que consiga mapear e se localizar no ambiente. No entanto, é necessário que as leituras dos sensores sejam filtradas antes de serem utilizadas em algoritmos. Uma das técnicas mais utilizadas para realização da fusão dos dados coletados para localização e mapeamento é o Filtro de Kalman (KF - do inglês *Kalman Filter*) (AGUNBIADE; ZUVA, 2018), que basicamente une os dados obtidos em cada unidade de tempo ponderando as incertezas de cada tipo de entrada para obter a localização. O Filtro de Kalman Estendido (EKF - do inglês Extended Kalman Filter) é derivado do KF, entretanto este é adaptado para lidar com sistemas não lineares (J. PARK; SONG, 2015), o que normalmente ocorre em situações reais, devido a desvios nas leituras de sensores e descontinuidades. Outras técnicas para fusão de sensores são vistas na literatura, destacando-se os algoritmos de Redes Neurais (SUNDERHAUF et al., 2016), de Deep Learning (BOHEZ et al., 2017), o filtro *Unscended Kalman Filter* (UKF) (BROSSARD; BONNABEL; BARRAU, 2018), os algoritmos utilizando Lógica Fuzzy (KAM; XIAOXUN ZHU; KALATA, 1997), entre outros.

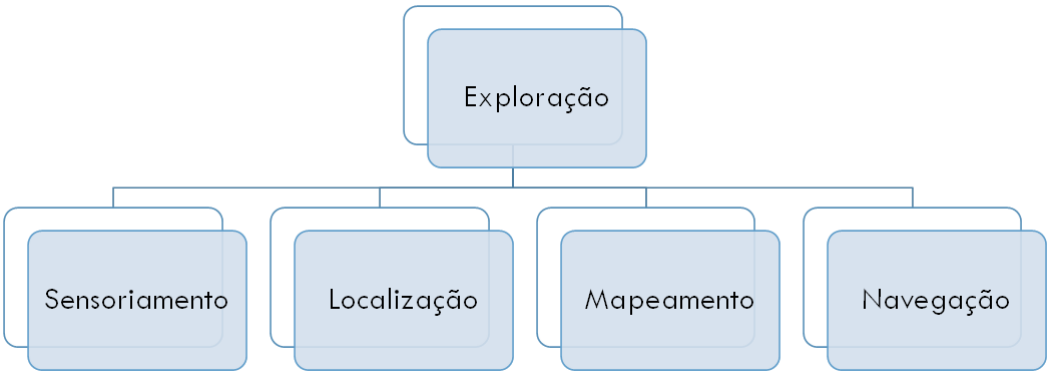
Com os dados tratados após a fusão de sensores, os mapas que servirão de referência para localização do robô são criados e podem ser em 2D ou em 3D dependendo da aplicação. Para realização do mapeamento em 2D, observa-se na literatura a frequente utilização do método de Mapeamento de Grades de Ocupação (MGO), que consiste em dividir o ambiente em células, obter a posição do robô no espaço e sua distância até o obstáculo, demarcando os limites no mapa ao preencher as células que condizem com a posição de obstáculos. Inicialmente a distância até as delimitações do ambiente era obtida estritamente através de sensores ultrassônicos (ELFES, 1989), no entanto, com a necessidade de maior precisão, outros sensores foram se desenvolvendo e passou-se a utilizar com maior frequência os sensores a laser como principais e os sonares como complementares (J. PARK; SONG, 2015) (RAVANKAR et al., 2017) (GAO

et al., 2018). Para o mapeamento em 3D o uso de câmeras (MAITY; SAHA; BHOWMICK, 2017) (MUR-ARTAL; TARDÓS, 2017) e sensores de infravermelhos (JIANHAO; MEIQIN; WEIHUA, 2015) são os mais comuns na literatura, entretanto, também é possível a realização do mapeamento em 3D através dos sensores a laser (C. PARK et al., 2018). Tanto o mapeamento 2D quanto o 3D necessitam que haja a fusão dos dados coletados na construção dos pontos das superfícies e limites do ambiente conforme é realizada sua exploração.

Para se localizar e mapear um ambiente um robô móvel necessariamente irá se mover de um ponto a outro em algum momento, portanto, é necessário que essa tarefa seja feita com segurança desviando-se de obstáculos. Para isso, as técnicas de navegação que são encontradas na literatura consistem principalmente em criar um planejamento da melhor trajetória com base nos dados obtidos dos sensores de distância e de sua localização relativa no mapa. Um conjunto de pacotes bastante utilizado para esse fim nos dias atuais é conhecido como *navigation*, desenvolvido por Marder-Eppstein et al. (2010), como mostrado no estudo de Conner e Willis (2017).

Para tomar decisões sobre quais serão os pontos a serem visitados durante a tarefa de exploração de ambientes desconhecidos é necessária a aplicação de técnicas para que a atividade seja completamente autônoma. Logo, uma técnica largamente utilizada é a Exploração Baseada na Fronteira (FBE do inglês) (SCHULTZ et al., 1999) (GAO et al., 2018) que consiste em explorar as áreas que não possuem limites conhecidos, outra técnica relevante é a *Next Best View Exploration* (NBVE), a qual baseia-se em estimar e selecionar qual seria o ponto no mapa que teria mais chances de obter maior área de exploração (GONZÁLEZ-BAÑOS; LATOMBE, 2002). Também é comum encontrar técnicas que utilizam o mapeamento topológico, através da criação de nós em estruturas de grafos com pesos para tomada de decisão em conjunto com MGO, para definir tanto a localização do robô quanto as coordenadas dos nós e dos próximos passos, como por exemplo a técnica denominada *Ear Based Exploration* (EBE) (ZHANG et al., 2014).

Figura 1 – Diagrama com a composição das tarefas necessárias para realização da exploração



Fonte: Autor.

Para que um robô móvel realize alguma atividade específica em qualquer tipo de ambiente, principalmente atividades que exijam alto grau de confiabilidade, é necessário que se localize e saiba navegar até determinado destino com precisão. A união de todos os conceitos citados envolvem a realização do trabalho proposto, como mostra o diagrama da figura 1. Muitos trabalhos já foram desenvolvidos a respeito do tema, todavia, diversos desafios ainda precisam ser superados para que as aplicações em situações cotidianas sejam cada vez mais comuns, como por exemplo a questão da segurança de operação com humanos, a eficiência durante a exploração do ambiente, a capacidade de se adaptar em ambientes dinâmicos, o custo de processamento computacional, a imprecisão na leitura dos sensores, etc. (CHRISTIANSEN, 2018) (Y. CHEN et al., 2018) (AGUNBIADE; ZUVA, 2018) (J. PARK; SONG, 2015) (JIANHAO; MEIQIN; WEIHUA, 2015) (GAO et al., 2018) (SUN et al., 2017). Todos esses desafios são envolvidos na exploração de ambientes que são desconhecidos. Os ambientes desconhecidos são aqueles que o robô nunca navegou ou ambientes que se transformam com o passar do tempo. Desta forma, esta pesquisa concentra-se na área de exploração autônoma de ambientes estáticos e desconhecidos.

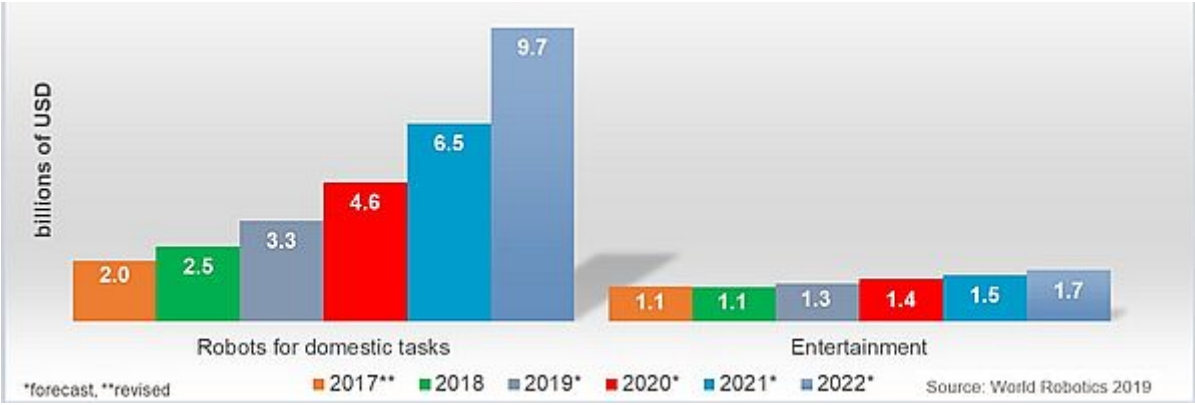
1.1 CONTEXTUALIZAÇÃO

Devido à grande importância da robótica nos últimos anos e seu potencial de crescimento, principalmente para uso doméstico e interno (figura 2), existe uma competição mundial chamada RoboCup. Fundada em 1997, seu intuito é fomentar as pesquisas relacionadas a robótica e inteligência artificial no mundo. Nesta competição, existem diversas modalidades de robótica, dentre elas destaca-se a modalidade @Home, foco do presente trabalho, que coloca a prova robôs de serviço em ambientes domésticos para realização de tarefas específicas. Para esta modalidade, foi criado um regulamento (MATAMOROS et al., 2019) que possui diversos critérios e regras para que a competição possa simular situações reais em cômodos distintos para que os robôs consigam desempenhar atividades propostas da melhor forma possível, sempre buscando minimizar a necessidade da intervenção humana.

A equipe de robótica da FEI, a RoboFEI, participa anualmente das competições. O robô utilizado para a modalidade @Home, chamado HERA (*Home Environment Robot Assistant*) (Figura 3), possui grandes avanços no reconhecimento de objetos, localização e manipulação (AQUINO JUNIOR et al., 2019). A estrutura do HERA é de um modelo de robô chamado *Peoplebot*, projetado para interagir com humanos.

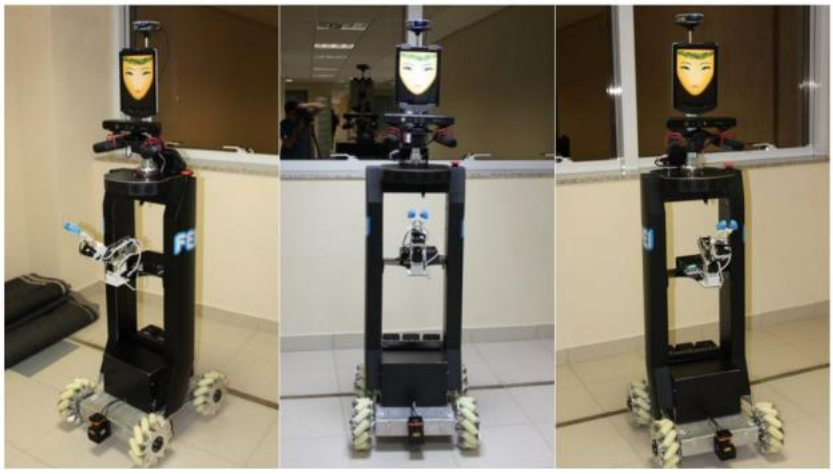
Durante as competições, e em tarefas normais, os robôs precisam se deslocar, evitar contato com objetos fixos e móveis, com humanos e com as paredes. Visando diminuir tanto o tempo necessário quanto a trajetória total percorrida durante o deslocamento entre dois pontos na realização de alguma tarefa, é extremamente importante que o robô se localize com boa precisão e mapeie o ambiente desconhecido em seu sistema durante as tarefas realizadas ou durante a exploração.

Figura 2 – Gráfico que mostra o crescimento da demanda por robôs domésticos de serviço desde 2017 e estimativas até o fim do ano de 2019 e 2020 a 2022.



Fonte: IFR, 2019.

Figura 3 – Robô HERA, da equipe RoboFEI para competições @Home.



Fonte: AQUINO JUNIOR, P. et al., 2019, p. 68.

Baseado no contexto das competições robóticas e robótica de serviço, esse trabalho propõe o estudo de técnicas de exploração autônoma, visando a eficiência, uso de sensores para segurança e a capacidade de explorar um ambiente de serviço completamente.

1.2 CONTRIBUIÇÃO

O problema de localização e mapeamento simultâneos em conjunto com exploração autônoma tem sido amplamente estudado pela comunidade científica. Diversas soluções distintas com variações têm sido propostas nos últimos anos (apresentadas no capítulo 3), no entanto, ainda existem muitas oportunidades de desenvolvimento e otimização, principalmente com o crescimento da capacidade computacional de processamento de dados e com o desenvolvimento de ferramentas para programação.

Neste estudo serão utilizados algoritmos e pacotes desenvolvidos por outros pesquisadores. Serão propostas novas parametrizações e adaptações utilizando o sistema *ROS* (do inglês *Robot Operating System*). As aplicações no *framework* ROS tem aumentado nos últimos anos e a comunidade contribui com diversos tutoriais e bibliotecas para implementação em projetos de mapeamento e localização simultâneos, como por exemplo a documentação compartilhada pela comunidade ROS (ROS.ORG, 2018). O robô HERA utiliza este mesmo sistema para coordenar suas atividades e processar os dados obtidos pelos sensores. A pesquisa considera o uso de um conjunto de sensores para melhoria de precisão do resultado de exploração do ambiente. Como escopo, consideram-se os seguintes sensores: laser (frontal e traseiro), câmera e sonar, sendo que estes elementos são comparados na análise dos resultados.

O estudo realiza todos os testes em um ambiente simulado chamado Gazebo. Ao implementar e testar algoritmos em simulações, obtém-se a agilidade no desenvolvimento do projeto e redução de custos. Neste contexto, o trabalho contribuirá principalmente com uma revisão abrangente sobre o tema de exploração e aplicação prática de alguns dos principais conceitos relacionados à localização, ao mapeamento e à navegação. O trabalho pretende verificar e testar pacotes projetados para exploração autônoma, bem como a variação da configuração de sensores para otimização dos resultados, para que seja possível, por fim, a criação do pacote *HERA-exploration*.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

O presente projeto tem por objetivo geral simular e implementar técnicas de exploração de ambientes de serviço desconhecidos e variar a configuração de sensores buscando otimizar os desempenhos em relação ao tempo, distância percorrida e necessidades de rotação para explorar um ambiente completamente.

1.3.2 Objetivos Específicos

Para atingir o objetivo geral destacam-se os seguintes objetivos específicos:

a) Criar um ambiente simulado utilizando o Gazebo para realização dos testes de desempenho dos algoritmos de exploração autônoma.
b) Utilizar o Kinect e o Xtion (câmeras para mensurar a profundidade), o laser e o sonar para otimização da detecção de obstáculos e limites do ambiente para que não hajam colisões e o Mapa de Grade de Ocupação (MGO) possa ser criado por completo.
c) Implementar e avaliar três pacotes, desenvolvidos no sistema ROS, com estratégias distintas de exploração: *explore-lite*, *RRT-exploration* e *Cam-exploration*.

d) Realizar a comparação das estratégias de exploração, levando em conta as diferentes configurações possíveis dos sensores, tanto em relação ao tempo gasto e distância percorrida para realização da exploração quanto em relação a capacidade de explorar o ambiente completamente.
e) Ao descobrir a configuração com melhor desempenho nas simulações, criar o pacote *HERA-exploration* para possibilitar o uso no robô HERA.

1.4 ORGANIZAÇÃO DO TEXTO

A desenvolvimento da dissertação organiza-se em quatro capítulos. O capítulo 2 refere-se a fundamentação teórica dos principais conceitos utilizados neste trabalho. Inicialmente os conceitos e definições relacionados ao sensoriamento são apresentados na seção 2.1, em seguida os principais algoritmos probabilísticos (seção 2.2) são apresentados para darem suporte ao entendimento dos conceitos de localização e mapeamento, utilizados no projeto. A seção 2.3 traz as definições e principais problemas encontrados em relação à localização de robôs móveis. Na seção 2.5 é apresentado um resumo de técnicas de mapeamento 2D e 3D. Na seção 2.6 são abordadas técnicas para fusão de sensores. Por fim, na seção 2.7 há a descrição das principais técnicas usadas tanto na navegação do robô quanto nas técnicas associadas a navegação que servem como base para o desenvolvimento das técnicas de exploração indicadas.

O capítulo 3 aborda o estado da arte de problemas de exploração, onde são discutidos e apresentados diversos conceitos e trabalhos relacionados ao estudo proposto.

No capítulo 4 é feita a descrição da arquitetura da plataforma robótica HERA que é utilizada nos experimentos, discussão sobre as regras da Robocup, e uma introdução sobre o funcionamento do ROS. Após isso, o capítulo traz os algoritmos e pacotes que auxiliaram no tratamento de dados de sensores. Na ultima seção deste capítulo, é apresentado o funcionamento detalhado dos pacotes de exploração avaliados.

No capítulo 5 são detalhados todos os experimentos realizados para encontrar as técnicas e configurações ideais para um ambiente de serviço. A seção 5.1 traz a discussão sobre os desempenhos dos sensores de distância para aplicação do SLAM durante a exploração. Em seguida a seção 5.2 mostra os resultados obtidos. A subseção 5.2.1 apresenta a análise feita para a calibragem dos pacotes de exploração disponíveis e seus desempenhos para utilização da variação de sensores. A subseção 5.2.2 avalia os desempenhos das montagens dos sensores. A subseção 5.2.3 traz um resumo dos resultados obtidos em todas as condições de exploração propostas. O capítulo termina o estudo trazendo uma discussão e análise das vantagens de se ter uma exploração prévia do ambiente antes de realizar tarefas no ambiente, mostrados na subseção 5.2.4.

Por fim, o capítulo 6 traz a conclusão e os trabalhos futuros dentro do contexto de exploração autônoma.

2 FUNDAMENTAÇÃO TEÓRICA

O estudo proposto é sobre a atividade de exploração autônoma de robôs móveis para realização de tarefas em ambientes internos. Um robô móvel precisa de mecanismos de locomoção para se deslocar pelo ambiente. No entanto, existem diversos mecanismos possíveis para diferentes tipos de aplicações, e isso determina como será o design do robô (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). O modelo Peoplebot, o mesmo do HERA mostrado na figura 3, é projetado para terrenos planos e uniformes, portanto sua locomoção é feita através de rodas.

Para um robô móvel realizar a exploração autônoma é necessário que ele perceba o mundo a sua volta. Analogamente aos sentidos dos seres vivos, os robôs devem conter multissensores espalhados pela sua estrutura para atingir este objetivo. Os dados obtidos através dos sensores precisam ser interpretados para gerar informação útil e tão precisa quanto possível para as tomadas de decisões. Portanto, diversas técnicas de fusão de sensores são propostas na literatura ao longo dos anos.

As quatro principais tarefas que um robô deve ser capaz de implementar para explorar um ambiente são: a capacidade de se localizar, a capacidade de criar mapas baseados nesta localização relativa, a capacidade de navegar sem sofrer colisões e a estratégia para decisão do melhor destino para exploração. Portanto, o capítulo abordará os princípios básicos de percepção com sensores, de localização, de mapeamento, de navegação e de exploração.

2.1 SENSORIAMENTO

2.1.1 Definições

Uma das tarefas mais importantes de um robô autônomo é adquirir informação e conhecimento do ambiente a sua volta (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Um robô sem sensores para realizar tarefas, torna-se apenas uma máquina que repete o movimento continuamente sem qualquer possibilidade de realizar tarefas mais complexas (EVERETT, 1995). Contudo, existem muitos tipos de sensores disponíveis no mercado para os mais diversos tipos de aplicações. Siegwart, Nourbakhsh e Scaramuzza (2011) classificam os sensores em duas importantes subdivisões:

Proprioceptivos/ exteroceptivos, classificação quanto ao tipo de informação coletada.

a) Sensores Proprioceptivos: sensores que medem características internas do robô; por exemplo, a velocidade do motor, a voltagem da bateria.
b) Sensores Exteroceptivos: sensores que adquirem dados do ambiente no qual o robô se encontra; por exemplo, medidas de distância, amplitude do som, imagem de câmeras.

E passivo/ ativo, classificação quanto à energia.

a) Sensores passivos: são os sensores que captam a energia vinda do ambiente; por exemplo, termômetros, microfones, câmeras.
b) Sensores ativos: sensores que emitem energia e retornam a leitura para interpretar como o sinal interagiu com o ambiente; por exemplo, sensores ultrassônicos, lasers.

Os sensores apresentam características importantes, possíveis de serem quantificadas em laboratório, e devem ser levadas em conta, quando aplicável, durante o projeto para determinar quais sensores devem ser utilizados para alguma aplicação desejada. Siegwart, Nourbakhsh e Scaramuzza (2011) classificam essas características da seguinte forma:

a) Alcance: é o maior valor de entrada que um sensor consegue atingir em suas leituras.
b) Resolução: é a mínima diferença entre dois valores que podem ser detectado por um sensor.
c) Linearidade: refere-se ao tipo de comportamento do sinal de entrada durante a leitura de sensores.
d) Largura de banda ou frequência: é a medida na qual um sensor consegue fornecer leituras durante sua operação.
e) Sensividade: é a medida do grau de variação do sinal de saída conforme varia-se o sinal de entrada.
f) Erro: é a diferença entre o valor medido e o valor real. Seja o valor medido m e o valor real v, tem-se:

$$erro = m - v \tag{1}$$

g) Exatidão: é o grau de conformidade entre a medida do sensor e o valor real. Pode ser representado da seguinte forma:

$$exatidão = 1 - \frac{|erro|}{v} \tag{2}$$

h) Precisão: é a capacidade de um sensor trazer sempre a mesma medida quando forem feitas múltiplas medidas de um mesmo estado do ambiente. Seja σ o desvio padrão das medições, tem-se:

$$precisão = 1 - \frac{variação}{\sigma} \tag{3}$$

2.1.2 Discussão

Os tipos de sensores utilizados no projeto para mensurar a profundidade são exteroceptivos, pois são os utilizados para obter as medições das características do ambiente. Na configuração do robô HERA tem-se disponíveis as câmeras RGB-D Xtion e Microsoft Kinect, que são sensores ativos, sendo que o Kinect possui a configuração com a câmera IR (*infrared camera*), e também utilizam-se os sensores ativos ultrassônicos e a laser (AQUINO JUNIOR

Tabela 1 – Características básicas dos sensores RGB-D, ultrassônico e a laser.

Produto	**Sensor a Laser**	**Sensor Ultrassônico**	**câmera RGB-D**
Modelo	Hokuyo UTM-30LX	URM06	Kinect
Alcance nominal	0,1-30m	0,2-10m	0,8-3,5m
Alcance máximo	60m	-	-
Exatidão	0,1 a 10m - 30mm	-	-
Resolução	1mm	1cm	3mm
Alcance espacial nominal	-	-	640 x 480 (VGA)
Ângulo de detecção	270° horiz x 0° vert	15° horiz x 15° vert	57° horiz x 43° vert
Resolução angular	$0{,}25^o$	-	-
Frequência de operação	40Hz	49,5kHz	30Hz
Peso	370g	45g	325g

Fonte: (HOKUYO, 2008), (DRROBOT, 2019) e (ANDERSEN et al., 2012).

et al., 2019). A tabela 1 mostra o comparativo de características nominais padrão entre os tipos de sensores escolhidos na configuração do robô.

O comparativo mostra que o sensor a laser possui o melhor alcance e a resolução mais precisa, além de conseguir detectar a profundidade em um ângulo de 270° na horizontal. Essas características são muito superiores aos dos outros dois tipos de sensores, no entanto, pontos específicos não demonstrados na tabela 1 podem justificar a necessidade do uso dos três em conjunto. O sensor a laser escolhido possui a limitação de apenas conseguir varrer o ambiente na horizontal (em 2D), e isso é compensado com o uso da câmera RGB-D que possui um ângulo de detecção na vertical de 43°, principalmente para detecção de obstáculos de tamanhos diversos. Tanto o sensor a laser quanto o de câmera têm a desvantagem em relação aos efeitos negativos da alteração da luminosidade, a exatidão e a resolução do sensor a laser Hokuyo UTM-30LX pioram conforme aumenta-se a incidência de luz (HOKUYO, 2008), assim como o que ocorre nos testes realizados no estudo de Andersen et al. (2012). Logo, os sensores ultrassônicos podem resolver estas questões, pois não são afetados pela luminosidade e podem detectar objetos e obstáculos translúcidos, pois seu princípio depende da reflexão de ondas sonoras. No entanto, para suprir a limitação de seu alcance angular, utilizam-se várias unidades dispostas ao redor do robô. Porém, neste caso, sua principal função é para detectar e desviar de objetos não detectados pelo laser e pela câmera.

O sensor a laser, devido às suas capacidades, costuma ser escolhido com frequência para aplicação de técnicas de exploração e de localização e mapeamento simultâneos em ambientes internos. No entanto, na literatura existem soluções distintas para suprir algumas das limitações citadas. Por exemplo, ao se utilizar o sensor acoplado a um dispositivo que rotaciona 360° supri-se a limitação de apenas mapear em 2D, como mostra o trabalho de C. Park et al. (2018). Também há a alternativa de utilizar lasers que mapeiam em 3D de fábrica, como mostra a aplicação em minas de escavação no trabalho de Losch et al. (2018). Contudo, o custo financeiro e computacional destas soluções ainda é relativamente maior do que as soluções mais comuns com apenas laser em 2D, logo, optou-se por manter essa configuração em conjunto com os

outros dois tipos de sensores de profundidade mostrados na tabela 1 para cumprir o objetivo de simular a exploração do ambiente de maneira autônoma.

2.2 ALGORITMOS PROBABILÍSTICOS EM ROBÓTICA

Um dos principais usos da probabilidade em robótica serve para estimar o estado de um robô móvel com base em leituras de sensores. Como os sensores possuem ruídos de natureza estocástica, é necessário que sejam aplicados algoritmos probabilísticos e recursivos para minimizar as incertezas (THRUN; BURGARD; FOX, 2005).

O Filtro de Bayes é um dos principais algoritmos utilizados no contexto de localização robótica. Nesta seção, serão introduzidos os conceitos do Filtro de Bayes e a classificação dos filtros derivados.

2.2.1 Filtro de Bayes

O Filtro de Bayes é um algoritmo recursivo baseado na crença $bel(x_t)$ do estado x_t no tempo t. A crença apresenta o resultado na forma de probabilidade e seu cálculo é feito em cima do resultado de $bel(x_{t-1})$ no tempo $t-1$ (THRUN; BURGARD; FOX, 2005). O algoritmo 1 mostra o pseudo-código de implementação da regra de atualização do Filtro de Bayes.

Algoritmo 1 – Pseudo-código que aplica o filtro de Bayes, de acordo com Thrun, Burgard e Fox (2005).

1 **Entrada:** ($bel(x_{t-1})$,u_t,z_t)
2 for todo x_t fazer
3 $\quad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx$
4 $\quad bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$
5 $endfor$
6 **retorna** $bel(x_t)$

O algoritmo funciona em duas partes: predição (atualização de controle) e atualização da medição. Na etapa de predição é feito o cálculo da crença $\overline{bel}(x_t)$ depois da ação de controle u_t baseado no instante x_{t-1}, como mostra a linha 3 do pseudo-código 1. O resultado da equação de predição é a integral da probabilidade de ocorrer a transição do estado x_{t-1} dado um controle u_t para estado x_t multiplicado pela crença $bel(x_{t-1})$. Importante frisar que para aplicar este algoritmo é preciso levar em conta a condição de controle x_0 com $t = 0$. A linha 4 traz a etapa de atualização da medição; nessa etapa obtém-se o resultado da crença $bel(x_t)$, onde multiplica-se a probabilidade da medição z_t no estado posterior x_t, que até esse ponto é hipotético, pela crença $\overline{bel}(x_t)$. Essa multiplicação deve ser normalizada pelo fator η para obter a probabilidade. Por fim, o algoritmo retorna a crença $bel(x_t)$, que recursivamente será utilizado para calcular o próximo estado (THRUN; BURGARD; FOX, 2005).

Embora esse algoritmo não seja implementado diretamente neste trabalho por só ser indicado para estimativas simples, seus princípios são importantes para compreensão dos algoritmos mais complexos de localização, pois a maioria dos algoritmos citados neste trabalho são seus derivados.

2.2.2 Tipos de Filtros

Muitos filtros para definição do estado a partir das probabilidades são propostos na literatura para aplicação em robótica, e podem ser divididos em duas categorias: filtros Gaussianos e Não-paramétricos (THRUN; BURGARD; FOX, 2005).

a) **Filtros Gaussianos:** ultimamente os filtros Gaussianos são as derivações do filtro de Bayes mais utilizadas. Seus princípios baseiam-se que a crença é representada por uma distribuição normal multivariada. Uma característica importante de um filtro gaussiano é o fato de possuir apenas uma máxima local, ou seja, estes filtros são unimodais. Estes métodos são largamente utilizados para o problema de rastreamento, pois apresentam baixíssimas incertezas. Os métodos mais conhecidos, são: Filtro de Kalman (KF do Inglês Kalman Filter), Filtro de Kalman Estendido (EKF do inglês Extended Kalman Filter), Filtro de Informação e Filtro de Informação Estendido (THRUN; BURGARD; FOX, 2005).
b) **Filtros não-paramétricos:** Os filtros não-paramétricos são uma alternativa aos Filtros Gaussianos, diferentemente dos filtros Gaussianos, eles não utilizam uma função fixa (distribuição normal) para cálculos dos espaços de estado. Ao invés disso, utilizam um número finito de valores que correspondem aproximadamente a uma sub-região do Espaço de Estados, para estimar a crença a partir da obtenção das leituras dos sensores. Exemplos de métodos: Filtro de Partículas (Monte Carlo) e histogramas (THRUN; BURGARD; FOX, 2005).

Como é possível observar nesta seção, diversos tipos de algoritmos recursivos e probabilísticos são possíveis de serem utilizados em robótica, no entanto, o foco do trabalho é em relação à métodos probabilísticos aplicados à localização de robôs autônomos. Nas seções seguintes são detalhados os principais algoritmos que se enquadram na resolução destes problemas para serem aplicados em exploração autônoma.

2.3 LOCALIZAÇÃO

A capacidade de se localizar no ambiente é fundamental para um robô conseguir executar tarefas com eficiência. Na maioria das aplicações o mapa é a referência global, desta forma, o robô obtém as coordenadas em relação a este referencial (THRUN; BURGARD; FOX, 2005). O robô pode definir a localização das coordenadas de objetos de interesse em seu próprio sistema de referência, o que é indispensável para navegação. Dessa forma, é possível realizar

a transformação de coordenadas do robô apenas definindo os seguintes parâmetros expressos na equação 4, desde que o sistema de coordenadas do mapa seja o mesmo que o do robô (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

$$p = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{4}$$

Para atingir o objetivo de se localizar com precisão, é necessário tratar os ruídos intrínsecos às características dos sensores, pois não existe leitura 100% precisa em aplicações reais (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Para lidar com estas questões, técnicas probabilísticas são propostas na literatura visando minimizar os erros associados às imprecisões das medições.

O *aliasing*, ou sobreposição, também é um problema comum e um desafio associado à localização do robô. Este tipo de problema refere-se à pouca quantidade de informação obtida por um sensor para caracterizar um ambiente dificuldade de um robô diferenciar duas situações diferentes que acabam produzindo a mesma leitura, ou seja, um sensor a laser que só consegue medir a distância, não consegue diferenciar uma pessoa de um objeto inanimado (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Por isso, é necessário aplicar a fusão de sensores para unir o máximo de informações úteis conforme as necessidades de cada aplicação.

Um dos métodos mais simples relacionados à localização de um robô é o chamando *Dead Reckoning*, que basicamente utiliza o modelo dinâmico do robô juntamente com dados de sensores para obtenção de maior precisão (KLEEMAN, 1992). Porém, o método possui a desvantagem de acumular erros devido ao escorregamento, como por exemplo durante o esterçamento em uma curva.

Nesta seção, primeiro é abordada a taxonomia dos problemas de Localização, em seguida descreve-se a técnica de localização de Markov, que é o a base para o desenvolvimento do filtro de Kalman e seus derivados e, por fim, o problema de localização e mapeamento simultâneos é apresentado.

2.3.1 Taxonomia de Problemas de Localização

A partir do conhecimento dos tipos de problemas de localização existentes, é possível entender o grau de dificuldade de cada um e a técnica necessária para lidar com seus ruídos e particularidades (THRUN; BURGARD; FOX, 2005), portanto, está seção traz uma breve discussão a respeito.

Os problemas de localização podem ser agrupados quanto aos seguintes tipos (THRUN; BURGARD; FOX, 2005):

a) **Informação Disponível:** Problemas de localização podem ser classificados quanto ao tipo de informação disponível durante o início e durante a operação.

Tabela 2 – Taxonomia de Problemas de localização em robótica.

Informação Disponível	Tipos de Ambientes	Abordagem Passiva x Ativa	Quantidade de Robôs
Localização Global: essa é a situação na qual o robô não conhece sua localização inicial, é colocado aleatoriamente no ambiente e não possui mais o referencial para estimativa de erro.	**Ambientes estáticos:** são ambientes onde seus elementos não se alteram com o tempo.	**Localização ativa:** Nessa abordagem, o algoritmo de localização influencia diretamente no controle do robô.	**Localização de um único robô:** Nesse caso, o algoritmo deve localizar apenas um robô.
Rastreamento de posição: este é um caso de problema local, no qual a posição inicial é conhecida e diversas medições consecutivas são obtidas conforme ocorre a operação do robô móvel, que normalmente gera um ruído relativamente baixo, pois as incertezas dos sensores geralmente são associados à uma distribuição Gaussiana.	**Ambientes dinâmicos:** a grande maioria das aplicações reais em robótica deve lidar com esse tipo de ambiente. Nesses casos, existem elementos que mudam de posição conforme passa o tempo.	**Localização passiva:** Nessa abordagem, o robô é apenas observado durante a operação, ou seja, o algoritmo de localização não interfere nos comandos do robô.	**Localização de vários robôs:** Nesse caso, vários robôs são localizados. A complexidade computacional cresce conforme aumenta-se o número de robôs, no entanto, em grupo há a vantagem de poder localizar um robô em relação ao outro, desde que a posição relativa seja conhecida.
Problema do sequestro do robô: este é um problema clássico associado à robótica móvel, no qual o robô é "transportado" sem prévio aviso e possui a tarefa de se localizar novamente. Esta situação pode ocorrer quando o robô passa por uma falha em seus sensores e retoma o funcionamento após percorrer alguma distância. Dessa forma, os robôs devem ser capazes de se recuperarem em casos de falhas de sistemas, principalmente tratando-se de robôs autônomos.	-	-	-
Problema do localização e mapeamento simultâneos: conhecido como SLAM (do inglês *Simultaneous Localization and Mapping*), este problema é mais complexo do que os anteriores, pois o robô inicia a operação sem as informações do ambiente e de sua própria localização relativa.	-	-	-

Fonte: (THRUN; BURGARD; FOX, 2005).

b) **Tipos de Ambientes:** Existem dois tipos de ambientes que podem afetar o grau de dificuldade durante a tarefa de localização. Ambientes dinâmicos e estáticos.

c) **Tipos de Abordagens:** Outra característica importante a ser considerada é se o algoritmo de localização comanda ou não os movimentos do robô.

d) **Quantidade de Robôs:** Existem dois tipos de abordagens quanto a quantidade de robôs para realizar a localização. Robôs únicos ou vários robôs.

A tabela 2 mostra os principais tipos de implicações referentes aos grupos classificados nesta seção. Com relação ao projeto proposto tem-se que resolver o problema de SLAM, em ambientes potencialmente estáticos, com localização ativa e com um único robô.

2.3.2 Localização de Markov

A Localização de Markov é a aplicação direta do filtro de Bayes para o caso em que o robô se locomove, que endereça o problema de rastreamento do robô, de Localização Global

e de Sequestro do Robô. Seu algoritmo é derivado do pseudo-código 1 apresentado na seção 2.2.1. O pseudo-código 2 mostra a aplicação de Markov. Importante notar que o algoritmo exige que o mapa m seja dado como entrada. Da mesma forma que o filtro de Bayes, o algoritmo transforma a crença probabilística no tempo $t-1$ na crença no tempo t (THRUN; BURGARD; FOX, 2005).

Algoritmo 2 – Pseudo-código que aplica a Localização de Markov, de acordo com Thrun, Burgard e Fox (2005).

1 **Entrada:** $(bel(x_{t-1}),u_t,z_t,m)$
2 for todo x_t fazer
3 $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1},m)bel(x_{t-1})dx$
4 $bel(x_t) = \eta p(z_t|x_t,m)\overline{bel}(x_t)$
5 $endfor$
6 **retorna** $bel(x_t)$

Este tipo de localização necessita ser aplicado em conjunto com os Mapas de Grade de Ocupação, pois cada célula vai carregar uma possibilidade de o robô estar ou não em seu espaço. A Localização de Markov depende diretamente do robô estar em um mapa m conhecido, portanto, apesar de resolver quase todos os problemas de localização descritos nesta seção, não é possível realizar a exploração a partir do algoritmo 2 diretamente. O filtro de Kalman Estendido é um tipo especial da localização de Markov que pode ser utilizado no cálculo do mapeamento de ambientes desconhecidos (THRUN; BURGARD; FOX, 2005), apresentado na seção 2.6.2.

2.4 SLAM

O SLAM é o problema mais complexo relacionado à localização, como apresentado na tabela 2, pois neste caso o mapa do ambiente e as posições iniciais não estão disponíveis para o robô. As únicas informações disponíveis para o robô, são as medições z_t e os dados de controle u_t (THRUN; BURGARD; FOX, 2005). É basicamente o processo no qual o robô móvel deve criar o mapa ao mesmo tempo que se localiza dentro dele (H. F. DURRANT-WHYTE; BAILEY, 2006).

Na visão probabilística, segundo Thrun, Burgard e Fox (2005) existem duas formas de lidar com problemas de SLAM. O problema *Online SLAM* é resolvido baseando-se em uma posição momentânea para estimativa do estado posterior.

$$p(x_t, m|z_{1:t}, u_{1:t}) \tag{5}$$

Onde x_t é o a posição no tempo t, m é o mapa, e $z_{1:t}$ e $u_{1:t}$ são as medidas e os controles, respectivamente. Esta abordagem é incremental e os resultados, medidas e controles anteriores são descartados após serem processados (THRUN; BURGARD; FOX, 2005).

Outra abordagem de SLAM é o *full* SLAM que consiste em calcular o estado posterior baseado em todo o caminho percorrido $x_{1:t}$ em conjunto com o mapa, ao contrário da abordagem anterior que só utiliza a posição x_t.

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) \tag{6}$$

As duas abordagens tem o objetivo de encontrar o estado posterior, no entanto, o *full* SLAM se torna inviável em muitas aplicações pelo aumento da quantidade de dados que necessitam de processamento, portanto, para resolver o problema do SLAM, utiliza-se com maior frequência o *online* SLAM, no qual o método EKF SLAM é um dos mais aplicados segundo Thrun, Burgard e Fox (2005).

O estado da arte do SLAM, segundo Bailey e H. Durrant-Whyte (2006), foca em três áreas chave de pesquisa: a complexidade computacional, associação de dados (fusão de sensores) e representação do ambiente (mapeamento).

O pacote utilizado para realização do SLAM no projeto para aplicação em todas as técnicas de exploração chama-se *gmapping* [1] (GERKEY B., 2018). Este pacote foi escolhido por se apresentar como um método bastante popular, e já configurado no pacote padrão do robô HERA (PIMENTEL F., 2020). Esse pacote utiliza o SLAM baseado na leitura de lasers e utiliza o filtro de partículas Rao-Blackwellized. O RBPF (do inglês *Rao-Blackwellized particle filter*) é uma técnica bastante conhecida na literatura, consiste em criar partículas nas quais cada uma descreve uma trajetória em potencial, além de carregar consigo um mapa 2D simulando a ação tomada. O trabalho de Grisetti, Stachniss e Burgard (2007) foca em diminuir o número de partículas geradas e propõe uma abordagem que leva em conta não apenas as projeções de trajetórias, mas também as leituras mais recentes dos sensores. O resultado é um MGO com boa precisão e com relativa baixa complexidade computacional dependendo da granularidade do tamanho das células escolhida.

Este trabalho não foca em resolver estas questões diretamente relacionadas ao SLAM, mas sim foca em resolver o problema da exploração de ambiente desconhecidos, não obstante, isso só é possível quando o SLAM é implementado de maneira satisfatória. Para atingir este objetivo, leva-se em conta a necessidade de uma associação de dados eficiente, com o estudo de técnicas de fusão de sensores (seção 2.6), e uma representação do ambiente eficaz, com o estudo de técnicas de mapeamento tanto em 2D quanto em 3D (seção 2.5).

2.5 MAPEAMENTO

Diversas aplicações em robótica exigem que o ambiente seja representado em forma de mapas para que o robô consiga se localizar dentro dele. As representações através de mapas podem variar dependendo da aplicação escolhida. Dessa forma, Siegwart, Nourbakhsh e Scaramuzza (2011) destacam três principais aspectos que devem ser considerados durante a

[1] Disponível em: <http://wiki.ros.org/gmapping> Acesso em: 01 de dez. 2019

escolha da melhor representação: o mapa deve ser aproximadamente tão preciso quanto a precisão exigida para o robô atingir seus objetivos; a precisão do mapa e das características a serem representadas devem combinar com os tipos de dados que são obtidos pelos sensores; e a complexidade do mapa afeta diretamente o custo computacional para operação.

Existem dois tipos de representação possíveis: representação contínua e através de estratégias de decomposição. Nas representações contínuas o mapa busca representar precisamente as características do ambiente no espaço contínuo e, devido ao custo computacional, para robôs móveis as representações devem ser feitas em 2D. Nas representações através de estratégias de decomposição, o ambiente é segregado em partes menores, como por células. Uma das vantagens dessa representação é a diminuição do custo computacional, porém diminui-se a precisão da localização, e consequentemente, a capacidade de realizar tarefas mais precisas. Há também a dificuldade de transformar o ambiente em blocos muito pequenos, pois assim há uma necessidade de maior espaço na memória (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

O mapeamento é um dos pontos mais importantes durante a tarefa de SLAM. A partir de um mapeamento preciso o robô poderá se localizar no ambiente e realizar tarefas de maneira mais eficiente. No entanto, diversos métodos diferentes são encontrados na literatura, como por exemplo em mapeamentos em 2D com MGO (ELFES, 1989), em 3D com Fusão de Elementos de Superfícies por laser (C. PARK et al., 2018) ou por câmeras com o Visual SLAM (BERGEON et al., 2015), e, embora muitas aplicações utilizem MGO em conjunto com mapeamento topológico, há trabalhos que utilizam mapeamento puramente topológico, que não depende de grande precisão na localização no ambiente (CHENG; H. CHEN; LIU, 2015).

Nesta seção tem-se uma revisão de como funcionam os mapeamentos citados. A princípio o mapeamento em 2D será detalhado, em seguida uma revisão dos mapeamentos em 3D e, por fim, a explicação a respeito de mapeamentos topológicos.

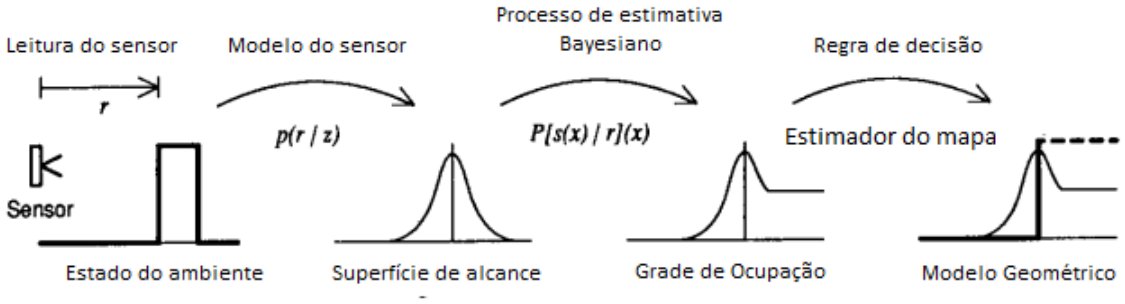
2.5.1 Mapeamento em 2D - Mapas de Gade de Ocupação

Um dos métodos mais empregados e que será utilizado no trabalho proposto é o Mapeamento por Grade de Ocupação (MGO). Um dos primeiros trabalhos a respeito do assunto foi proposto por Elfes (1989), que usa técnicas de mapeamento probabilísticos que transforma o ambiente em células, e cada célula recebe dados ruidosos de sonares ou de sensores a laser para definição de seu estado, 0 (vazio) ou 1 (ocupado).

Os sensores utilizados para este tipo de representação apresentam o modelo da incerteza de medição pela distribuição Gaussiana, dado pela equação 7, onde o r é a medição do sensor de distância, z é a distância real e σ é o desvio padrão.

$$p(r|z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-(r-z)^2}{2\sigma^2}\right] \tag{7}$$

Figura 4 – Etapas do processo de identificação do estado s da i-ésima célula C_i utilizando o modelo do sensor e o teorema probabilístico de Bayes.



Fonte: Autor "adaptado de" Elfes, 1989, p. 48.

Para definir a probabilidade de termos uma célula C_i ocupada ou não, usa-se a fórmula de atualização sequencial do teorema de Bayes, como mostra a equação 8.

$$p[s(C_i) = OCC|r_{t+1}] = \frac{p[r_{t+1}|s(C_i) = OCC]P[s(C_i) = OCC|\{r\}_t]}{\sum\limits_{s(C_i)} p[r_{t+1}|s(C_i)]P[s(C_i)|\{r\}_t]} \tag{8}$$

Onde OCC é o estado ocupado, $s(C_i)$ é o estado que se encontra a i-ésima célula, $\{r\}_t$ é o vetor com os dados obtidos dos sensores no tempo e r_{t+1} é o próximo dado a ser computado. A figura 4 demonstra como é feito o processo de medição, transformação, estimativa e definição do estado da célula.

2.5.2 Mapeamento em 3D - Fusão de Elementos de Superfície e *Visual* SLAM

O mapeamento em 3D tem demonstrado muitos avanços nos últimos anos, principalmente em relação ao *Visual* SLAM. Suas duas principais vertentes são relacionadas ao tratamento de imagens capturadas por câmera, uma com técnicas de *Deep Learning* para cálculo de profundidades, a outra refere-se à obtenção de características de quadros chaves selecionadas a partir de determinados critérios (MUR-ARTAL; TARDÓS, 2017) (MAITY; SAHA; BHOWMICK, 2017) (JIANHAO; MEIQIN; WEIHUA, 2015).

Para resolver o problema de *Visual* SLAM existe um método chamado rastreamento e mapeamento paralelos (PTAM do inglês *Parallel Tracking and Mapping*) (BLOESCH et al., 2018). As aplicações são para utilização de câmeras monoculares, contudo, na literatura é possível encontrar trabalhos que utilizam câmeras estéreas (S-PTAM) (PIRE et al., 2015). O princípio desta técnica é dividir o processamento em duas *threads* diferentes: na *thread* do rastreamento o robô realiza a combinação das características, cria novos pontos e estima a posição da câmera; e na *thread* do mapeamento são refinados os pontos de referência de vizinhança que compõem o mapa interativamente (SUNDERHAUF et al., 2016). Apesar do trabalho de S-

Figura 5 – Reconstrução de um ambiente de escritório em 3D utilizando técnica de fusão de elementos de superfície com pontos obtidos por um dispositivo manual giratório de laser LiDAR.

Fonte: PARK, C., 2018, p. 1208.

PTAM trazer resultados promissores para resolver o problema de SLAM em ambientes abertos, ainda há a possibilidade de implementar a exploração e a navegação autônomas em conjunto.

Além do *Visual* SLAM, estudos de técnicas de mapeamento em 3D com lasers também tem evoluído nos últimos anos. Existem dois tipos principais de mapeamento com lasers: com o sensor a laser 3D com visão em 360 graus, como no caso do robô utilizado em minas subterrâneas do trabalho de Losch et al. (2018), ou com sensor a laser 2D acoplado a um dispositivo que o rotaciona em 360 graus fornecendo a característica de três dimensões de maneira assíncrona. Essa técnica é conhecida como CT-SLAM (*Continuous-Time SLAM*) (C. PARK et al., 2018) (DUBÉ et al., 2016). As técnicas com Lasers possuem em comum a característica de criar nuvens de pontos e realizar a fusão da superfície conectando os pontos mais próximos gerados. A figura 5 mostra um ambiente de escritório reconstruído com CT-SLAM depois de uma varredura.

Tanto as técnicas de mapeamento 3D quanto as técnicas de mapeamento em 2D estudadas tem em comum a necessidade do fechamento de laço para um mapeamento mais preciso. Isso significa que os algoritmos vão realizando os mapeamentos, vão montando os mapas e ao mesmo tempo vão verificando se os pontos são os mesmos que foram registrados anteriormente. Quando pontos repetidos são encontrados, o robô recalcula o mapa e os erros associados à imprecisão das medidas dos sensores que foram se acumulando são reduzidos. Importante ressaltar que o fechamento de laço serve para ambos os tipos de mapeamento, embora na literatura seja mais citado quando relacionados a técnicas em 3D. O pacote *gmapping*, implementado no

projeto para o mapeamento em 2D, utiliza este recurso para otimização da qualidade do mapa criado.

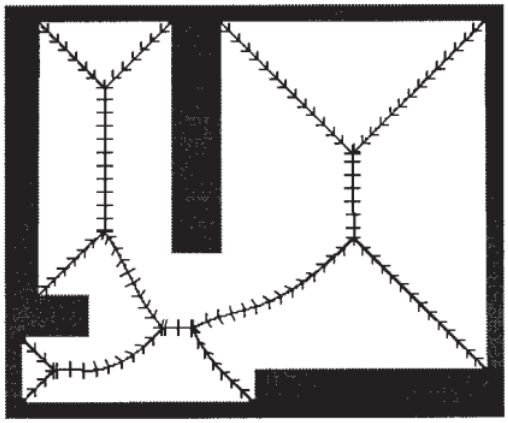
O mapeamento 3D no presente trabalho busca utilizar as imagens de uma câmera de profundidade (Kinect e Xtion) para gerar uma nuvem de pontos e projetar os objetos encontrados para duas dimensões. É importante citar todos os métodos de mapeamento, pois são alternativas para situações mais complexas que demandam maior detalhamento e consigam trabalhar com um custo computacional mais elevado. Como o escopo está no desenvolvimento da exploração, utiliza-se neste trabalho o mapeamento 2D para representação do ambiente.

2.5.3 Mapeamento Topológico

O mapeamento topológico pode ser utilizado de muitas formas, mas as principais envolvem a característica da criação de grafos com nós conectados para representação do ambiente e/ou geração de caminhos que conectam dois pontos de interesse. Essas duas estratégias citadas podem servir tanto para exploração quanto para navegação.

Uma das principais formas de implementação do grafo é através de criação um esqueleto no MGO, como mostra a figura 6. Neste caso, o objetivo da implementação foi utilizar o Grafo de Voronoi Generalizado GVG para definir pontos de exploração do ambiente (CHOSET; BURDICK, 2000). Este grafo é caracterizado por ser composto por vários nós equidistantes dos obstáculos em n dimensões e é criado com o objetivo de gerar nós para que o robô os visite conforme o ambiente é explorado. Wurm, Stachniss e Burgard (2008) e Kwon e Song (2006) também seguem esta estratégia de mapeamento, nos quais o primeiro foca em uma aplicação de exploração com multi-robôs e o segundo foca na criação do grafo, desbastando os espaços vazios até sobrarem apenas linhas, e na tomada de decisão, conforme aumenta-se a probabilidade de encontrar um nó final.

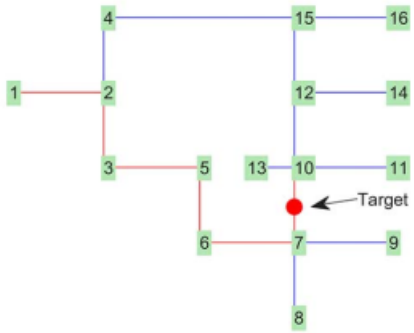
Figura 6 – Representação topológica de um ambiente utilizando GVG.



Fonte: Choset; Burdick, 2000, p.101.

Outro tipo de estratégia para aplicação de mapeamentos topológicos é através de classificações semânticas, como o trabalho feito por Cheng, H. Chen e Liu (2015). Neste caso, o

Figura 7 – Representação puramente topológica de um ambiente após sua completa exploração.



Fonte: Cheng; Chen, H.; Liu, 2015, p. 737.

ambiente todo é representado por um grafo. A figura 7 mostra o mapeamento topológico completo de um ambiente após a exploração. Para realizar este mapeamento o projeto utiliza uma plataforma robótica com um sensor Kinect para medição do ambiente interno a sua frente e, com essa informação obtida, classifica os tipos de corredores, através do método de Bayes e uma base de dados com modelos pré-classificados para que o mapeamento seja feito. Os números mostrados na figura 7 são a representação destes corredores identificados. O mapeamento é finalizado após o robô percorrer livremente o ambiente até achar o fechamento de laço a partir da identificação de pontos em comum. O Kinect não é tão preciso, por isso não é necessária a precisão absoluta para localização, pois o mapa é topológico e não leva em conta medidas exatas, mas sim as classificações e localização relativa do robô no ambiente.

Vários algoritmos foram desenvolvidos e adaptados para navegação. Neste tipo de mapeamento leva-se em conta a criação de um MGO ou já tem-se um mapa estático carregado para que os nós sejam criados e conectados nos espaços livres de obstáculos. Guruji, Agarwal e Parsediya (2016) levam em conta a utilização do algoritmo A* para busca do menor caminho, criando um grafo da menor distância entre dois pontos desviando de eventuais obstáculos. Neste trabalho, leva-se em conta a aplicação da heurística do algoritmo A* convencional apenas nas regiões de possíveis conflitos, buscando a otimização do tempo necessário para o processamento da avaliação dos nós. Com uma ideia muito similar, tem-se o algoritmo analítico Dijkstra aplicado para encontrar busca do menor percurso, como mostra o trabalho de Wang, Yu e Yuan (2011).

Outro algoritmo empregado na criação de grafos para encontrar o menor caminho é o RRT (do inglês **Rapidly-exploring random trees**). Resumidamente, este algoritmo baseia-se na criação de pontos ou nós aleatórios no no mapa a partir da localização do robô que conectam-se aos seus vizinhos mais próximos. Estes nós são criados e conectados até que seja encontrada uma conexão com o ponto objetivo, seu resultado é como se fosse uma árvore com ramificações. Como as conexões só são feitas entre espaços livres do mapa, encontra-se um caminho livre de obstáculos para que o robô atinja o objetivo (LAVALLE, 1998).

Neste trabalho utiliza-se o mapeamento topológico principalmente para obtenção da distância necessária até os possíveis destinos de exploração. Usando essa métrica, tem-se o algoritmo Dijkstra através do pacote *navfn* para o pacote *explore-lite*, *RRT* para o pacote *RRT-Exploration* e A* para o pacote *Cam Exploration*. Estes algoritmos de navegação e implementações na exploração serão discutidos com maior profundidade na seção 2.7 e no capítulo 3.

2.6 FUSÃO DE SENSORES PARA LOCALIZAÇÃO E MAPEAMENTO

Os sensores são os componentes do robô responsáveis pela percepção do mundo e fornecimento de dados para realização dos mais diversos procedimentos, no entanto, sensores isoladamente possuem, em muitos casos, baixa eficácia, devido principalmente aos seus graus de incerteza e ruídos das leituras. Portanto, para tarefas mais complexas, que envolvam muitos parâmetros e fatores, como o SLAM, é necessário que esses dados obtidos pelos sensores sejam tratados e processados para que o robô possa utilizar como entrada para seus procedimentos. Para isso ser possível é necessário que seja feita a fusão dos sensores utilizando filtros específicos, de modo que haja a complementação e/ou a compensação dos dados (KAM; XIAOXUN ZHU; KALATA, 1997).

Quando se trata da fusão de sensores com características diferentes, Thrun, Burgard e Fox (2005) definem que há duas formas de lidar com o problema. No primeiro caso o filtro de Bayes e suas derivações para SLAM são aplicados normalmente com os modelos dos sensores apropriados para cada tipo, no entanto, se um tipo detectar um obstáculo e o outro não, haverá conflitos, o que prejudicará a construção do mapa. No segundo caso, para cada tipo de sensor é gerado um mapa diferente, desta forma os mapas gerados se combinarão e o pior caso será considerado, ou seja, se uma célula estiver ocupada em qualquer um dos mapas criados, o mapa final, após fundirem as informações de todos os mapas, mostrará a célula como se estivesse ocupada.

No contexto da exploração autônoma, os algoritmos e métodos de fusão de sensores são na maioria dos casos utilizados para obtenção mais precisa da localização com os dados de odometria, portanto é importante o aprofundamento e o entendimento das técnicas mais comuns para escolher as que melhor se encaixam nas condições do problema proposto.

Logo, por ser um assunto fundamental para aplicação deste projeto em situações reais, esta seção trará alguns dos principais métodos pesquisados na literatura que contribuíram para elaboração da proposta de pesquisa: Filtro de Kalman, Filtro de Kalman Estendido, Localização Adaptativa de Monte Carlo (AMCL - do inglês*Adaptive Monte Carlo Localization*).

2.6.1 Filtro de Kalman (KF)

O Filtro de Kalman (KF) é um método matemático que visa utilizar dados adquiridos ao longo do tempo com o intuito de estimar o estado atual, ou seja, a posição do robô no ambiente. Por exemplo, a partir de sensores de distância para estimativa da posição, com seus respectivos graus de incertezas. Como dito anteriormente, essa técnica é derivada da localização de Markov. A vantagem é conseguir convergir para um resultado mais preciso do que se fosse estimado por qualquer sensor isoladamente.

A técnica é recursiva (*Online* SLAM), não necessita de todo o histórico de estados para estimativa do estado atual, apenas o estado anterior e as medidas atuais, podendo ser utilizada em tempo real. Funciona em um processo de duas etapas. Na etapa de predição o algoritmo estima as variáveis do estado atual com suas respectivas incertezas. Com a obtenção das medidas pelos sensores tem-se a etapa de atualização, que utiliza a média ponderada para corrigir os valores das estimativas da etapa anterior. A média ponderada segue o racional que da maior peso para as medições e variáveis que têm as menores incertezas (WELCH; BISHOP, 1994).

O objetivo do KF é obter o estado $x \in \Re^n$ do processo controlado de tempo discreto que é governado pela equação linear e estocástica (WELCH; BISHOP, 1994):

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{9}$$

Com a medida z $\in \Re^m$:

$$z_k = Hx_k + v_k \tag{10}$$

Onde A é a matriz nxn que representa o estado anterior $k-1$ e B é a matriz nxl relacionada ao controle de entrada $u \in \Re^l$. As variáveis que representam os ruídos dos processos e das medições são representados por w_k e v_k respectivamente, as quais são independentes entre si, assumidamente possuem a distribuição de probabilidade normal e aleatória com média zero.

$$p(w)\ N(0, Q), \tag{11}$$

$$p(v)\ N(0, R). \tag{12}$$

As equações 11 e 12 representam as relações das matrizes de covariância dos ruídos dos processos (Q) e das medidas (R). Ambas as matrizes podem variar a cada instante de tempo ou podem ser constantes dependendo da aplicação.

A matriz A da equação 9 possui dimensão n x n e relaciona o estado anterior $k-1$ com o estado atual k. A matriz B da equação 9 relaciona o controle de entrada opcional $u \in \Re^l$ e possui dimensão n x l. A matriz H da equação 10 de dimensão n x m relaciona o estado das medidas atual z_k com o estado anterior. Todas as matrizes podem variar a cada passo no tempo ou serem consideradas constantes.

O algoritmo para aplicação com dados discretos no tempo se divide em dois grupos de equações distintas: as equações que definem a atualização do estado e as que definem a correção do estado. As equações de atualização fazem uma projeção no tempo do estado e da matriz de covariância dos erros para que obter a estimativa primária. A estimativa primária é utilizada na etapa de correção a partir de sua combinação com as medidas obtidas z_k.

As equações 13 e 14 representam o estágio de atualização que traz a estimativa primária.

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_k \tag{13}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{14}$$

Pela equação 13 é possível observar a projeção da estimativa de $\hat{x}_k$ (estado atual) com base em $hatx_{k-1}$ (estado anterior), nota-se que as matrizes A e B são as mesmas da equação 9, Q vem de 11 e P é a matriz de covariância dos erros. As equações 15, 16 e 17 representam a etapa de correção.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{15}$$

$$\hat{x}_k = A\hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{16}$$

$$P_k = (I - K_k H)P_k^- \tag{17}$$

Após completada a etapa de estimativa prioritária para realizar a correção a tarefa inicial, é calculado o ganho de Kalman, K_k. Para calcular a equação 16 é necessário obter as medições z_k para enfim determinar a correção da estimativa primária. Por fim, a equação 17 calcula a covariância P_k.

Estes são os princípios básicos do KF, no entanto Welch e Bishop (1994) e Thrun, Burgard e Fox (2005) detalham mais profundamente as questões que envolvem as estimativas dos parâmetros, as origens das equações de covariância e explicações mais detalhadas dos principais conceitos e tipos de aplicações.

2.6.2 Filtro de Kalman Estendido (EKF)

O KF é largamente utilizado em diversos estudos, porém, há uma limitação para calcular equações que não são lineares. Portanto é proposto o Filtro de Kalman estendido (EKF), que atende principalmente às necessidades de aplicações reais de auto localização no âmbito de robótica móvel a partir de vários sensores.

A principal tarefa é resolver o problema da não linearidade, logo, para linearizar o modelo através do EKF, é aplicada a expansão de primeira ordem de Taylor (AGUNBIADE; ZUVA, 2018). Da mesma forma que no caso do KF, é necessário estimar o vetor de estados x

$\in \Re^n$. O modelo agora é governado pela equação diferencial não linear estocástica (18), e as medições $z \in \Re^m$ são dadas pela equação 19.

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{18}$$

$$z_k = h(x_k, v_k) \tag{19}$$

Da mesma forma que no caso do KF, v_k e u_k representam os ruídos do processo e das medições. As funções não lineares f e h representam as relações entre os instantes de tempo k e $k-1$ e entre o estado x_k e as medições z_k, respectivamente.

A divisão entre as etapas de estimativa do estado e aplicação da correção são análogas ao KF, portanto as equações 20 e 21 representam a etapa de estimativa primária e as equações 22, 23 e 24 representam a etapa de correção (WELCH; BISHOP, 1994).

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{20}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Qk - 1 W_k^T \tag{21}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{22}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-,0)) \tag{23}$$

$$P_k = (I - K_k H_k) P_k^- \tag{24}$$

A sequência para realização dos cálculos é a mesma descrita na seção do KF, no entanto as matrizes A, W, H, V e W que relacionam os estados, a estimativa, as medições e os erros são agora Jacobianas, ou seja, são matrizes de derivadas parciais de primeira ordem para realização das transformações não lineares e também são variáveis a cada instante de tempo k. As matrizes R_k e Q_k representam a covariância do ruído das medições e do processo respectivamente, e também variam a cada instante de tempo.

A viabilidade da utilização desta técnica ainda é atual, todavia, em comparação com o KF original, seu uso possui um custo computacional significativamente maior (AGUNBIADE; ZUVA, 2018). Este filtro também é muito utilizado em aplicações de *Online* SLAM, no entanto, em muitos casos há a necessidade de se trabalhar com pontos de referência no mapa para atualização da localização, e se estes pontos forem muito ambíguos, o mapeamento se torna menos eficiente (THRUN; BURGARD; FOX, 2005).

O pacote avaliado para implementação no contexto deste trabalho foi o *robot_localization* desenvolvido por Moore e Stouch (2014). Este pacote calcula a posição do robô com base em

vários sensores. No estudo de sua implementação, Moore e Stouch (2014) avaliam a combinação de sensores na eficiência do fechamento de laço (diferença entre o ponto final e inicial de um circuito). Neste caso, combinaram-se *encoders* com IMU (do inglês *Inertial Measurement Unit*) e com GPS (do inglês *Global Position System*). Os resultados demonstraram um erro de fechamento de laço menor com a combinação dos três, no entanto, no presente trabalho o robô é designado primordialmente para tarefas em ambientes fechados e por isso o GPS não pode ser utilizado. A pesquisa mostrou que ao utilizar a combinação de *encoders* e um IMU também é possível obter resultados satisfatórios. Uma saída para melhorar a precisão é unir os dados de odometria obtidos pelo pacote *robot_localization* com filtros de partículas, como discutido na seção 2.6.3 quando o mapa já é conhecido. A aplicação deste pacote só será útil nos testes em situações reais para diminuir os erros provenientes dos sensores e escorregamento. Para o caso simulado do projeto, o Gazebo fornece dados de odometria que não necessitam filtros para aplicação posterior junto aos algoritmos de exploração e SLAM.

2.6.3 Filtros de Partículas

Um dos filtros de Partículas mais utilizados na literatura é o método de localização de Monte Carlo (MCL do inglês *Monte Carlo localization*) (THRUN et al., 2001). O MCL é um algoritmo probabilístico que obtém a crença bel do estado através de partículas devidamente ponderadas. Este algoritmo serve para resolver os problemas da Localização Local e da Localização Global (THRUN; BURGARD; FOX, 2005).

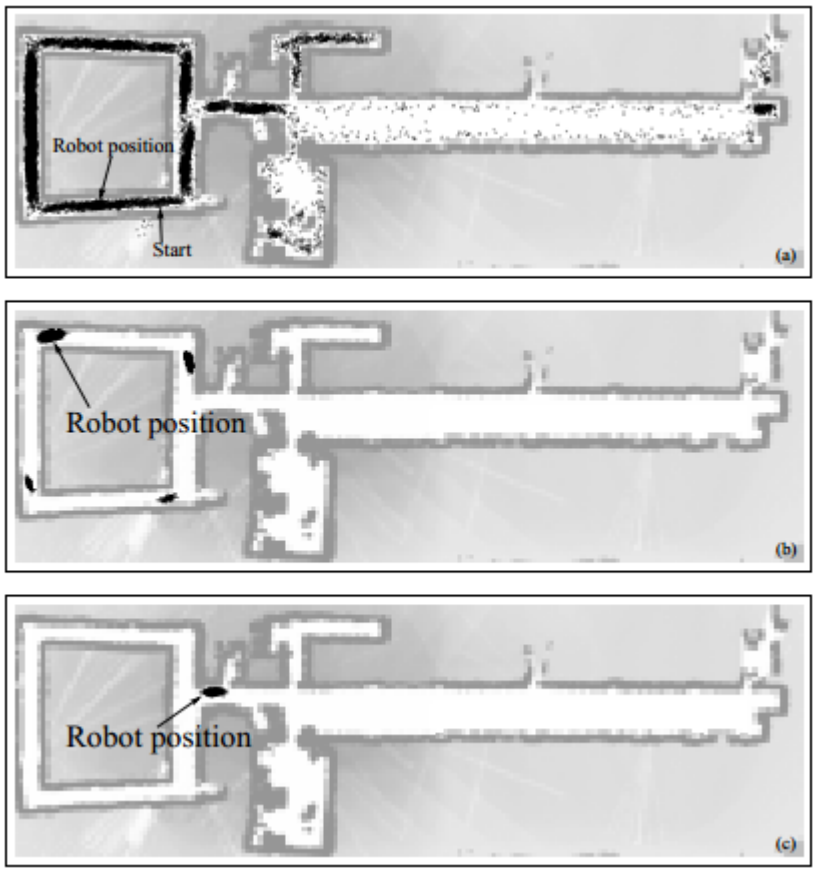
O algoritmo possui variações com heurísticas para diferentes tipos de aplicações (THRUN et al., 2001), porém, seu princípio básico se dá através do pseudo-código demonstrado pelo Algoritmo 3.

Algoritmo 3 – Pseudo-código da aplicação do filtro de Partículas da localização de Monte Carlo, de acordo com Thrun, Burgard e Fox (2005).

1 **Entrada:** $(\chi_{t-1}, u_t, z_t, m)$
2 $\overline{\chi_t} = \chi_t = \emptyset$
3 $for\ m = 1$ até M fazer
4 $\quad x_t^{[m]} =$ **modelo_de_movimento_por_amostragem** $(u_t, x_{t-1}^{[m]})$
5 $\quad w_t^{[m]} =$ **modelo_de_medição** $(z_t, x_t^{[m]}, m)$
6 $\quad \overline{\chi_t} = \overline{\chi_t} + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7 $endfor$
8 $for\ m = 1$ até M fazer
9 $\quad$ desenhar i com probabilidade $\propto w_t^{[i]}$
10 $\quad$ adicionar $x_t^{[m]}$ a χ_t
11 $endfor$
12 **retorna** χ_t

χ_t representa a crença $bel(x_t)$ dividido em M partículas como $\chi_t = \{x_t^{[1]}, x_t^{[2]}, x_t^{[3]} ... x_t^{[M]}\}$. Da linha 3 até a linha 7 todas as partículas e seus pesos são calculados. Na linha 4 a partícula

Figura 8 – Ilustração do funcionamento do algoritmo MCL, onde os pontos pretos representam as partículas: a) Início da operação com as partículas distribuídas aleatoriamente; b) após se deslocar por alguns metros o algoritmo começa a aglomerar partículas em grupos separados após atribuição dos pesos; c) depois de um certo tempo de operação o algoritmo tende a aglomerar as partículas na posição correta do robô com uma margem de erro relativamente pequena.



Fonte: Thrun; Burgard; Fox, 2005, p. 203.

$x_t^{[m]}$ é calculada com base na função que traz o modelo de movimento do robô em função da ação de controle u_t realizado no período t e do estado da partícula $x_{t-1}^{[m]}$ no tempo $t-1$. O modelo de movimento do robô depende do tipo de configuração, dinâmica e modelo do robô, por isso é descrito de forma genérica. A linha 5, por sua vez, calcula o peso da partícula $w_t^{[m]}$ através do modelo de medição dos sensores envolvidos na aferição da posição real em função da própria partícula $x_t^{[m]}$ e da medição z_t obtidas no cálculo anterior e através dos dados dos sensores respectivamente (THRUN; BURGARD; FOX, 2005).

O principal aspecto que garante a eficiência deste algoritmo são as linhas 8 até a 11, as quais são responsáveis pela nova amostragem com base nos pesos $w_t^{[m]}$ obtidos na linha 5. Dessa forma, conforme as medições vão ocorrendo, as partículas que inicialmente foram distribuídas aleatoriamente vão se aglomerando na posição do robô, como mostra a Figura 8 (THRUN; BURGARD; FOX, 2005).

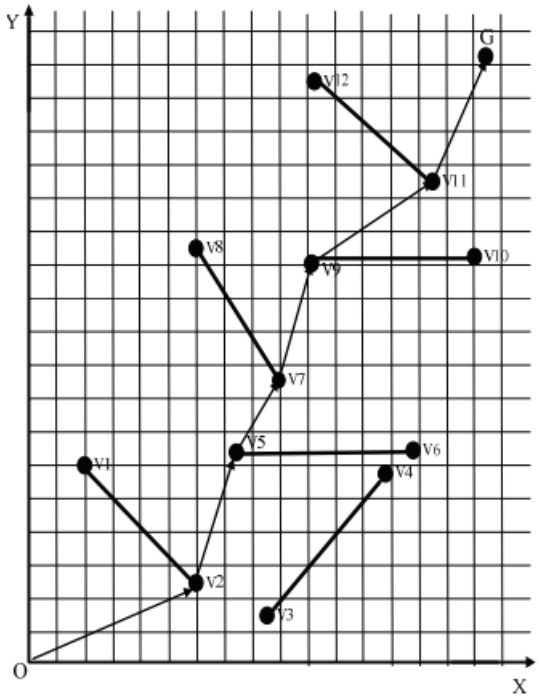
Um parâmetro importante a ser discutido é a quantidade M de partículas que devem ser escolhida para atingir a melhor precisão com o menor custo computacional possível. Thrun, Burgard e Fox (2005) mostra soluções adaptativas para definir o número de partículas conforme a necessidade de maior precisão na localização. A principal vantagem de MCL em relação ao EKF é a possibilidade de filtrar todos os tipos de comportamento de ruídos de sensores, não só as distribuições Gaussianas como no caso do EKF. Tendo um mapa pronto como referência após a tarefa de exploração é possível utilizar o princípio do MCL para obter uma localização mais estável, da mesma forma que foi feito no trabalho de Lee et al. (2017), o qual utilizou-se o nó AMCL do conjunto de pacotes *navigation* no ROS, contido em ROS.org (2019), que se trata de uma versão adaptativa do MCL, o mesmo que será implementado neste trabalho. Como citado na seção anterior é importante ressaltar que ambas as técnicas, EKF e AMCL, podem ser utilizadas em conjunto para obter uma precisão maior quando há um mapa de referência dado como entrada, todavia, dependendo no número de sensores utilizados e a quantidade de partículas geradas, o custo computacional pode aumentar consideravelmente tornando a combinação inviável.

2.7 NAVEGAÇÃO

A exploração de ambientes desconhecidos só pode ocorrer se o robô móvel se deslocar pelo ambiente. Para isso, depende de uma estratégia eficiente de navegação, na qual o robô deve ser capaz de desviar de obstáculos e se locomover até o objetivo. A estratégia de navegação depende da capacidade do robô de perceber o ambiente através de sensores, se localizar e ter pelo menos uma representação parcial do mapa para que consiga identificar sua posição em relação ao objetivo. Essas capacidades devem ser combinadas para que o robô consiga planejar a trajetória e enviar comandos para que o deslocamento ocorra (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). O planejamento de percurso pode ser dividido em duas abordagens diferentes: planejamento global e planejamento local. Entre eles, o planejamento global refere-se ao planejamento baseando-se nas informações estáticas do ambiente e o planejamento local planeja o percurso com base nas informações obtidas conforme ocorre a navegação, como em situações dinâmicas (WANG; YU; YUAN, 2011).

Diversos algoritmos para realização do planejamento são propostos na literatura, e que inclusive serão utilizados nos pacotes associados a esse projeto. Para otimização das aplicações dos algoritmos de navegação, em muitos casos utiliza-se uma representação do ambiente que permite o algoritmo obter informações semânticas, separando o mapa em camadas para realização de cálculos. Essa representação é descrita na subseção 2.7.4.

Figura 9 – A imagem mostra o resultado de uma simulação da aplicação do algoritmo Dijkstra para planejamento global do percurso. Os pontos O e G representam os pontos inicial e final, respectivamente. Os grafos conectados por v_i e $v_(i+1)$ representam os obstáculos de maneira simplificada. As setas mostram o percurso escolhido pelo robô até o ponto objetivo.



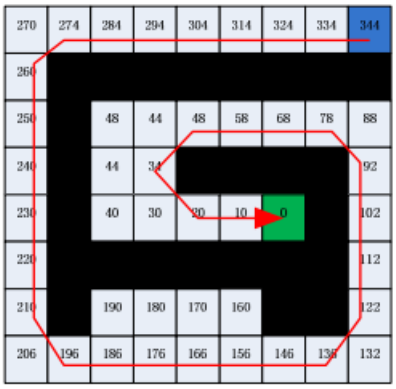
Fonte: Wang; Yu; Yuan, 2011, p. 1069.

2.7.1 A* e Djikstra: Planejamento Global

Na literatura podemos encontrar diversos algoritmos de busca para calcular o menor caminho de um nó inicial de um grafo até o nó final (objetivo), logo, os principais são os algoritmos analíticos, representados por Dijkstra (WANG; YU; YUAN, 2011), e algoritmos heurísticos, representados por A* (WEITENG et al., 2013). Esses dois casos são comumente utilizados para o planejamento global de percurso.

O algoritmo de Dijkstra é relativamente mais simples de se implementar no contexto do planejamento de percurso do que o A*. Wang, Yu e Yuan (2011) descrevem a construção do algoritmo da seguinte forma: O primeiro passo é identificar os obstáculos em um ambiente e transformá-los em grafos conectando os pontos que representam seus extremos. Em seguida, traça-se uma linha entre o ponto de origem e o ponto objetivo. A partir disso, avalia-se se há intersecções entre a linha traçada e os obstáculos. Quando houver intersecção, conecta-se a posição atual do robô aos dois extremos da representação do obstáculo. Após isso, seleciona-se o vértice com a distância mais curta e o traçado é feito a partir do vértice selecionado até o ponto final. Quando não houver intersecções entre a linha até o objetivo e os obstáculos, o caminho está definido até o final. A figura 9 ilustra uma simulação da aplicação do algoritmo para planejamento global do percurso de um robô.

Figura 10 – O mapa em questão é a representação do ambiente fragmentado em grades, no qual o quadrado azul representa o nó inicial, os azuis claros os nós possíveis em formato de grades, os pretos são os obstáculos, as numerações são os custos atrelados, o quadrado verde é o nó objetivo e o caminho em vermelho é o percurso ótimo escolhido para o caso específico.



Fonte: Weiteng et al., 2013, p. 163.

O algoritmo A* implementado para planejamento do menor percurso entre o nó inicial e nó objetivo utiliza, em grande parte das aplicações, a representação do ambiente em grades, nas quais cada grade é um nó com um custo a ser atribuído. A ideia principal é calcular e comparar a distância ou o custo entre o nó inicial e o nó possível, e a distância entre o ponto atual até o nó objetivo. Com o custo calculado de cada nó possível, escolhe-se o de menor valor, repetindo esse processo sucessivamente até que o nó atual torna-se o nó objetivo (WEITENG et al., 2013). O Algoritmo A* tradicional é governado pela seguinte equação:

$$F(n) = G(n) + H(n) \tag{25}$$

Onde $F(n)$ é o resultado da equação, $G(n)$ define o custo de mover-se do nó inicial até o nó possível, normalmente utilizando a distância geométrica entre as grades, e $H(n)$ é o custo de mover-se do ponto atual até o ponto final (WEITENG et al., 2013). Weiteng et al. (2013) e Guruji, Agarwal e Parsediya (2016) citam que a principal limitação do algoritmo é a elevada complexidade computacional, pois quando o ambiente é separado em grades, é necessário o cálculo de todos os nós ou grades para definição do caminho ótimo, e quanto maior for o tamanho e a complexidade do ambiente maior será o custo para calcular todos os nós do percurso. Para isso ambos propõem abordagens para que não haja o cálculo geral do ambiente a cada passo do algoritmo. No caso do trabalho de Guruji, Agarwal e Parsediya (2016), é proposta uma abordagem que utiliza a linha do nó inicial até nó objetivo como caminho até aparecer um obstáculo, para que apenas nesses nós ao redor do obstáculo seja aplicada a equação 25, bastante similar com a estratégia apresentada no trabalho de Wang, Yu e Yuan (2011). Enquanto no trabalho de Weiteng et al. (2013), apresenta-se uma abordagem focada em um planejamento de percurso local, no qual calcula-se apenas os nós das vizinhanças do nó em que ele se encontra, não mais todo o ambiente de uma vez. A figura 10 mostra um exemplo de aplicação do A*.

Os princípios dos dois algoritmos citados ainda são muito utilizados no contexto de definição de percurso, pois seus resultados são satisfatórios e suas implementações são relativamente simples, principalmente no ROS, que possui o conjunto de pacotes denominado *navigation*[2] (ROS.ORG, 2019) que foi implementada no robô HERA durante as simulações e implementações. O pacote do *navfn*[3] (KONOLIGE; MARDER-EPPSTEIN, 2014) , pertencente ao *navigation*, permite a utilização de ambos os algoritmos para definição do planejamento global do menor percurso entre dois pontos no mapa.

2.7.2 DWA e Eband: Planejamento Local

Para o planejamento local da trajetória, um dos métodos utilizados neste projeto chama-se DWA (do inglês *Dynamic Window Approach*). Esse método é implementado nos pacotes *dwa_local_planner*[4](criado por Marder-Eppstein (2014)) e *base_local_planner* (criado por Marder-Eppstein E. e Perko E. (2019)), também contidos no conjunto de pacotes *Navigation* para navegação. O princípio básico é proposto e descrito por Fox, Burgard e Thrun (1997). O algoritmo segue dois grandes passos para implementação: definição das restrições do espaço de busca e depois a implementação da equação de otimização e cálculo de custos das trajetórias.

Para Definir as restrições do espaço de busca, tem-se:

a) **Trajetórias circulares**: o método DWA apenas admite trajetórias circulares para os cálculos, determinadas por pares (v,ω), que representa a velocidade translacional e rotacional, respectivamente.
b) **Velocidades admissíveis**: o algoritmo apenas permite calcular trajetórias que não causem uma colisão com um obstáculo, sendo assim, todas as trajetórias que culminariam em colisão são descartadas.
c) **Janela Dinâmica**: a janela dinâmica restringe a velocidade admissível dentro da capacidade do robô atingir em um curto intervalo de tempo com base na aceleração limitada do robô.

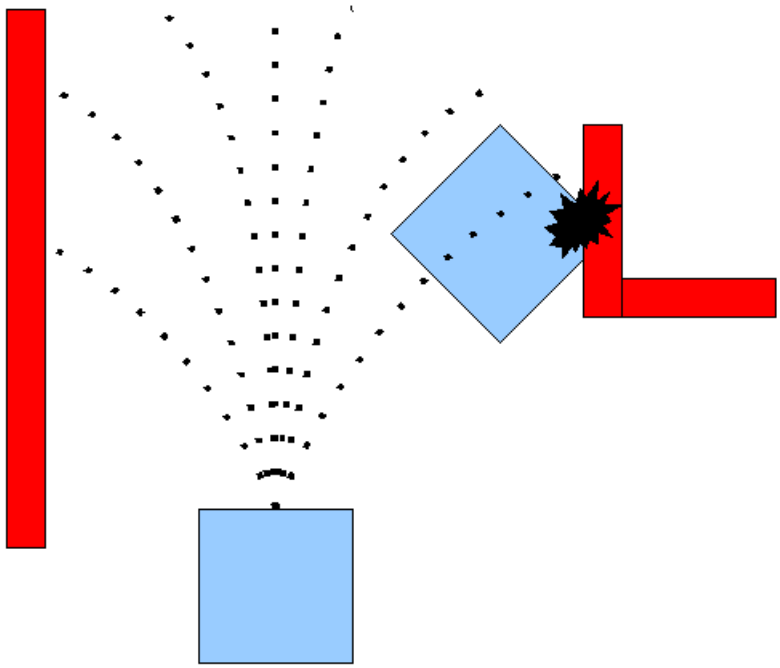
Com essas restrições definidas, tem-se a equação de otimização:

$$G(v,\omega) = \sigma(\alpha.alvo(v,\omega) + \beta.distancia(v,\omega) + \gamma.velocidade(v,\omega)) \tag{26}$$

Nesta equação G representa o custo da trajetória dado (v,ω) de entrada. A função $alvo$ mede o progresso em direção ao ponto final definido, e torna-se máximo quando a trajetória atinge o alvo pré-definido. A função $distancia$ mede a distância entre o robô durante a trajetória e os obstáculos, quando mais próximo o robô andar de um obstáculo, maior será este valor. Por fim, a função $velocidade$ calcula a velocidade para frente do robô, sempre dando maior importância para trajetórias menos agudas. Nota-se que as letras gregas α, β e γ formam

[2]Disponível em: <http://wiki.ros.org/navigation> Acesso em: 20 de jun. 2019
[3]Disponível em: <http://wiki.ros.org/navfn> Acesso em: 01 de ago. 2019
[4]Disponível em: <http://wiki.ros.org/dwa_local_planner> Acesso em: 01 de nov. 2019

Figura 11 – Representação da simulação da aplicação do algoritmo DWA para planejamento local da trajetória.



Fonte: Marder-Eppstein, 2014.

a ponderação das funções, e σ é responsável pela suavização da ponderação dos três componentes, que resulta em um espaçamento maior entre a trajetória do robô e os obstáculos (FOX; BURGARD; THRUN, 1997).

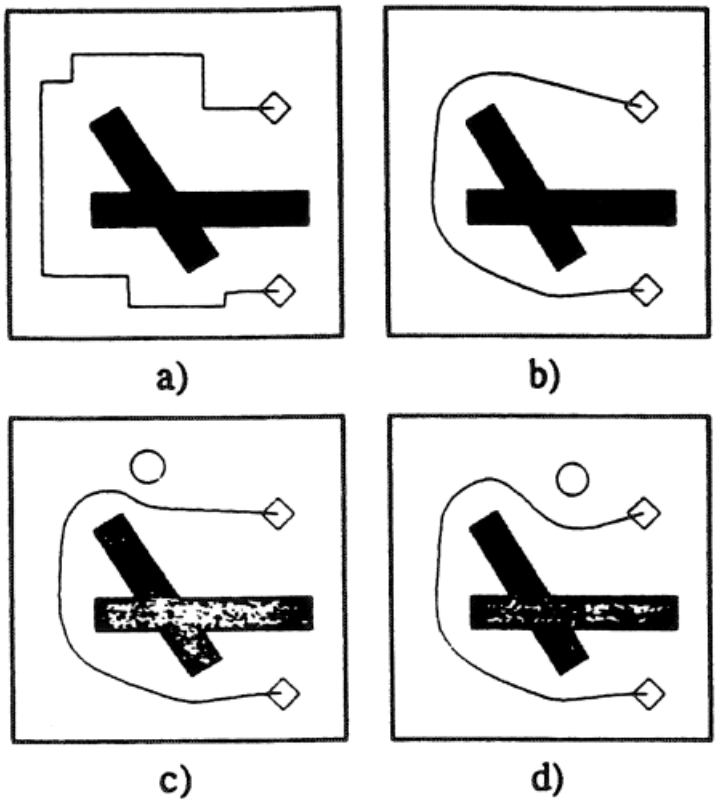
A figura 11 mostra um exemplo de uma simulação feita pelo robô para definição da melhor trajetória. Neste exemplo, foi definida uma amostra com sete combinações de velocidades translacionais e rotacionais, onde uma delas é descartada, pois há uma colisão com a o obstáculo a direita. Assim, essas outras trajetórias são calculadas com seus respectivos custos e escolhe-se a que obtém o maior valor na equação 26.

Eband (do inglês *Elastic Band*) é outro algoritmo muito utilizado para planejamento local de trajetórias na robótica móvel e também esta contido no conjunto de pacotes *navigation* através do pacote *eband_local_planner* [5] (UTEXAS-BWI, 2018). Proposto inicialmente por Quinlan e Khatib (1993), este algoritmo permite a flexibilização de trajetórias definidas conforme dados são obtidos por sensores, como mostra a sequência da figura 12. Nesta figura, tem-se os obstáculos fixos pintados de preto, o obstáculo móvel representado pelo círculo branco e os losangos brancos representado os pontos final e inicial. Na configuração inicial (figura 12a)), tem-se a trajetória definida pelo algoritmo representadas por linhas retas com descontinuidades, em seguida, na figura 12b) são aplicados os filtros para suavização das curvas, e, por fim, as figuras 12c) e 12d) mostram a adaptação da trajetória conforme introduz-se um obstáculo móvel no ambiente. Apesar dessa característica de adaptação da trajetória, este algoritmo só consegue se adaptar quando não há uma variação muito brusca. Como por exemplo

[5] Disponível em: <http://wiki.ros.org/eband_local_planner> Acesso em: 01 de out. 2019

quando fecha-se uma porta, mesmo que tenha uma outra alternativa em outro cômodo, o robô não será capaz de encontrar a solução diretamente através dessa técnica (QUINLAN; KHATIB, 1993).

Figura 12 – Correção da trajetória utilizando o algoritmo Eband.



Fonte: Quinlan; Khatib, 1993, p. 2.

O robô HERA utiliza o pacote *base_local_planner*[6] como padrão para o planejamento local, todavia, no projeto selecionou-se o pacote *eband_local_planner* para implementação. Como o foco deste trabalho é na implementação de técnicas de exploração, não serão avaliadas as diferentes técnicas de navegação para atingir o objetivo, visto que ao usar o mesmo pacote para navegação em conjunto com os pacotes de exploração, é possível obter o mesmo parâmetro para comparação, evitando distorções causadas pela possível diferenciação de desempenho durante a navegação.

2.7.3 Planejamento de trajetória com *RRT*

O algoritmo *RRT* (do inglês *Rapidly-exploring Random Trees*), citado na subseção 2.5.3, é um importante método de planejamento de trajetórias encontrado na literatura. A tarefa de planejamento de trajetória é normalmente uma busca de um caminho contínuo em um espaço métrico X a partir da posição inicial, x_{ini}, até a posição final, x_{fim}, ambos contidos em X (LAVALLE, 1998). Em um ambiente real, obstáculos delimitam os locais que não podem

[6]Disponível em: <http://wiki.ros.org/base_local_planner> Acesso em: 01 de out. 2019

ser acessados pelo robô, no caso, representa-se como X_{obs} o espaço inacessível, que deve ser evitado durante a geração de pontos aleatórios e suas conexões. Ou seja, X é composto por X_{obs} e X_{livre}, e todos os cálculos serão realizados no espaço acessível X_{livre}.

Dado K como sendo os nós ou vértices criados, x_{ini} como sendo o estado inicial,tem-se o pseudo-algoritmo:

Algoritmo 4 – Pseudo-código da aplicação do algoritmo *RRT* para navegação (LAVALLE, 1998)

1 **Entrada:** x_{ini},K,Δt
2 trajetoria.inicio(x_{ini})
3 **for** $k = 1$ **to** K **do**
4 x_{ale} <= FUNCAO-ALEATORIA();
5 x_{prox} <= VIZINHO-PROXIMO(x_{ale}, trajetoria)
6 u <= SELECIONAR-ENTRADA(x_{ale}, x_{prox})
7 x_{novo} <= NOVO-ESTADO(x_{prox},u, x_{prox}, Δt)
8 trajetoria.ADICIONAR-VERTICE(x_{novo})
9 trajetoria.ADICIONAR-ARESTA(x_{novo},x_{ale},u)
10 **retorna** trajetoria

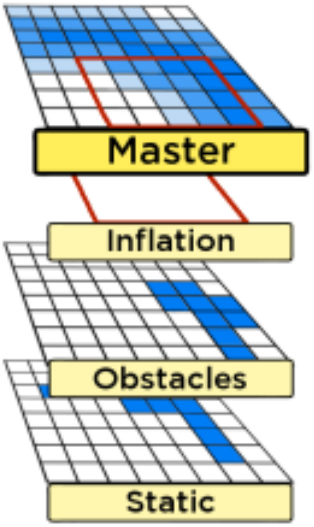
O primeiro vértice da trajetória é $x_{ini} \in X_{livre}$. A cada interação da função aleatória, adiciona-se um estado aleatório x_{ale} selecionado de X. O passo 5 encontra o vértice mais próximo x_{prox} em relação a distância. No passo 6 obtém-se u que minimiza distância do x_{prox} para o x_{ale}, garantindo que esteja no espaço X_{livre}. A detecção de colisões não está explicita, mas pode ser incorporado juntamente com o passo 6. O $NOVO - ESTADO$ é acionado para cada entrada para calcular o potencial novo estado x_{novo}. x_{novo} é obtido aplicando-se u e adicionado como vértice da trajetória, e então uma aresta conecta x_{novo} ao x_{prox} atribuindo-se u à conexão (LAVALLE, 1998).

Este algoritmo é capaz de construir uma trajetóŕia livre de obstáculos do ponto inicial até o ponto final. Sendo assim o algoritmo pode ser utilizado tanto para o planejamento local quanto o global, dependendo da aplicação. No presente trabalho utiliza-se o algoritmo de exploração *RRT-Exploration*, que usa os princípios deste método para encontrar a fronteira mais próxima, obtendo esse valor através dos cálculos dos planejamentos global e local. As características principais desse método estão descritas na subseção 4.4.2.

2.7.4 *costmap2D*

As características do mapa que são utilizadas como referência para os algoritmos descritos nesta seção fazerem os planejamentos de percurso e identificação de obstáculos compõem o *costmap*. O *costmap* pode ser representado em 2D ou em 3D e é construído tanto para representar o mapa global quanto do local. A forma mais simples do *costmap* é um MGO, na qual as grades ocupadas representam as regiões que não deve ser visitadas pelo robô para evitar que haja colisões (LU; HERSHBERGER; SMART, 2014).

Figura 13 – Exemplo da configuração de camadas de *costmap*. A camada *Static* é a representação do mapa estático, a camada *Obstacles* é a representação da leitura dos sensores de distância quando encontram-se obstáculos, a camada *Inflation* é a camada que projeta a representação das outras camadas, na qual as outras são infladas para representar a configuração espacial do robô, e por fim, a camada *Master* que une todas a informações obtidas em todas as camadas separadas para representação final do *costmap2D*.



Fonte: Lu; Hershberger; Smart, 2014, p. 4.

Muitas técnicas de navegação baseiam-se nas informações contidas em um único mapa estático para planejamento do menor percurso evitando obstáculos, todavia, quando há um aumento da complexidade e dinamismo do ambiente essa abordagem é pouco eficiente, pois assim, por exemplo, não é possível distinguir uma pessoa ou um obstáculo móvel. Para resolver esse problema, Lu, Hershberger e Smart (2014) propõem uma abordagem na qual o ambiente se divide em várias camadas de *costmap*. Essas camadas possuem informações semânticas para diferenciar o tipo de informação que o algoritmo irá receber como entrada para tomada de decisão.

A figura 13 ilustra como seria a representação de um *costmap* separado em camadas com informações semânticas do ambiente. No caso do exemplo citado, tem-se 4 camadas: *Static*, *Obstacles*, *Inflation* e *Master*. A camada *Static* é a representação do mapa estático, como por exemplo quando se tem um mapa em 2D previamente desenhado ou uma planta dada como entrada no robô. A camada *Obstacles* é a representação da leitura dos sensores de distância quando encontram-se obstáculos, como por exemplo quando um laser detecta um móvel que não esta previsto na planta de alguma sala. A camada *Inflation* é a camada que projeta a representação das outras camadas, na qual as outras são infladas para representar a configuração espacial do robô, e por fim, tem-se a camada *Master* que une todas a informações obtidas em todas as camadas separadas para representação final do *costmap2D*.

Como já dito nesta seção, o robô HERA utiliza o conjunto de pacotes *navigation*, que contém entre outros pacotes, o pacote *Costmap2D* [7] criado por Marder-Eppstein (2018). Este pacote permite utilizar por padrão as camadas mostradas na figura 13, no entanto, o autor deixa em aberto a possibilidade da criação de outras camadas conforme as necessidades de projeto, como por exemplo implementar uma camada para identificar pessoas em um ambiente (*proxemic_layer* [8]).

[7]Disponível em: <http://wiki.ros.org/costmap_2d> Acesso em: 01 de out. 2019
[8]Disponível em: <http://wiki.ros.org/social_navigation_layers> Acesso em: 01 de out. 2019

3 EXPLORAÇÃO AUTÔNOMA

O tema principal do projeto é a exploração autônoma com variação de sensores em ambientes internos. Portanto, este capítulo aborda trabalhos importantes relacionados ao assunto dentro do contexto em questão.

A primeira seção aborda as principais classificações das técnicas de exploração encontradas na literatura, alguns trabalhos implementados e desafios encontrados.

3.1 TÉCNICAS DE EXPLORAÇÃO

Existem diversas técnicas de exploração autônoma na literatura, portanto, é possível encontrar características que podem ser divididas da seguinte forma: exploração baseada em fronteiras, exploração através de ganhos, exploração com mapeamento topológico e exploração com multi-robôs. Normalmente, encontram-se combinações dessas características em um mesmo pacote, como é possível notar nesta seção.

Nesta seção, a princípio analisa-se a exploração baseada em fronteiras (o tipo mais comum de abordagem), em seguida a exploração através de ganhos, depois com o mapeamento topológico e, por fim, abordagens diversas com a combinação das anteriores, ou possibilidades com variação de sensores.

3.1.1 Exploração Baseada em Fronteiras

O método FBE (do inglês *Frontier Based Exploration*), é um dos primeiros caminhos escolhidos para o estudo de exploração autônoma, é bastante encontrado na literatura e possui diversas variações com aprimoramentos e abordagens distintas. Esse método foi introduzido primeiramente por Yamauchi (1997). É importante descrever seu conceito, pois a maioria das técnicas mais modernas, inclusive as implementadas neste trabalho, utilizam seus princípios.

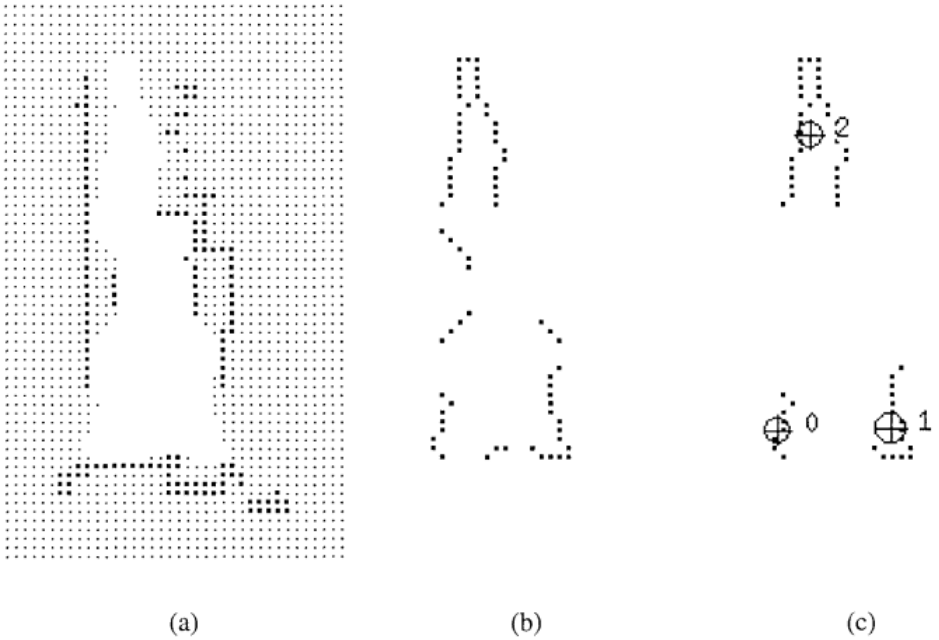
O primeiro passo para entendimento dessa abordagem é definir o conceito de fronteira. Yamauchi (1997) define como sendo o limite entre o espaço aberto e espaço desconhecido. O conceito de fronteiras pode ser ilustrado como mostra a figura 14.

A técnica possui um conceito simples, no qual, com ajuda do MGO, as fronteiras são delimitadas e definidas como candidatas a serem visitadas. Com as fronteiras definidas, o robô irá decidir a fronteira desconhecida mais próxima para explorar e assim sucessivamente, como mostra a figura 15 (YAMAUCHI, 1997).

3.1.2 Exploração Baseada em Ganhos

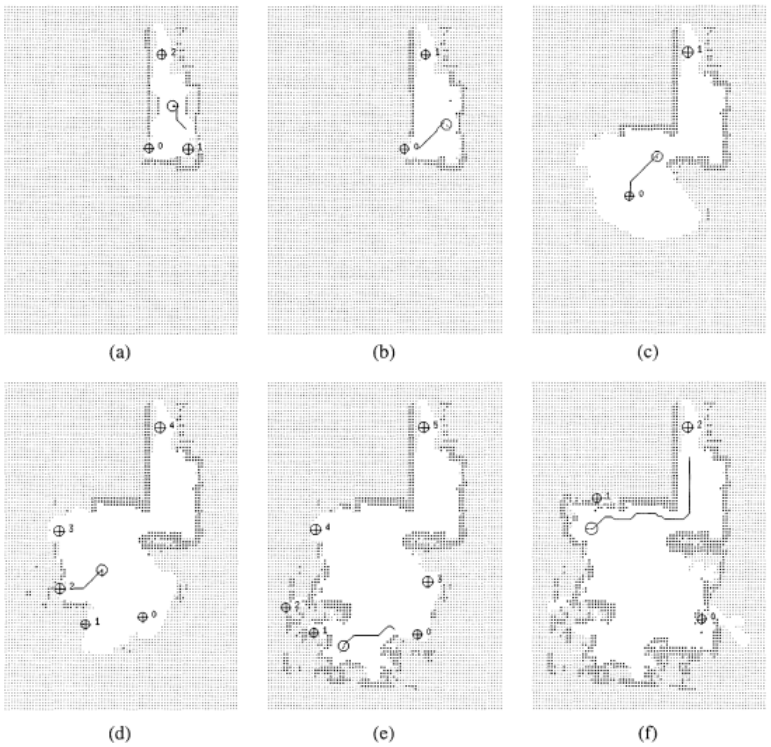
A partir do momento que há a necessidade de aumentar a eficiência de exploração e definir critérios mais sofisticados para escolha de novos destinos, introduz-se o conceito de ganho.

Figura 14 – a) Mostra um MGO, onde os espaços em branco são as áreas conhecidas (espaços abertos), os pontos menores são as áreas desconhecidas (espaços desconhecidos) e os pontos maiores são os limites conhecidos. b) Destaque dos limites que separam os espaços abertos dos espaços desconhecidos. c) Destaque das regiões conhecidas como fronteiras candidatas à exploração.



Fonte: Yamaguchi, 1997, p. 148.

Figura 15 – Passos da exploração de um escritório, utilizando FBE.



Fonte: Yamaguchi, 1997, p. 149.

Normalmente, os ganhos são atribuídos a candidatos a serem visitados e com isso escolhe-se o que tem o maior valor calculado, conforme a definição de parâmetros e estratégias pré-definidas.

O algoritmo NBVE (do inglês *Next Best View Exploration*) é utilizado por González-Baños e Latombe (2002) para realizar a exploração de ambientes desconhecidos. Este algoritmo, assim como os outros de exploração, auxilia na decisão do próximo passo do robô para obter a maior quantidade de informação possível acerca das regiões desconhecidas, ou seja, regiões que não possuem limites conhecidos (*free edges*), analogamente à definição de fronteira apresentada na subseção 3.1.1.

O algoritmo primeiro cria um mapa parcial em MGO na posição q_0 com os dados dos sensores. A cada interação o mapa é atualizado com as novas informações obtidas na posição q_k. O principal objetivo do algoritmo é obter a próxima posição sensorial q_{k+1}. Para chegar nessa posição, algumas etapas devem ser respeitadas: primeiro é necessário criar uma amostra de candidatos $N_amostra$ aleatórios com uma quantidade calculável de candidatos que tem visibilidade para as fronteiras; em seguida, para todo $q \in N_amostra$ computar o comprimento das curvas sólidas que são visíveis a partir de q_k e verificar os pontos que estão dentro de um limite mínimo, os que não tiverem são descartados; por fim utiliza-se um planejador de trajetória (que pode ser qualquer um) e os pontos q que não tiverem um caminho definido sem conflitos com obstáculos a partir de q_k são mantidos, os demais descartados (GONZÁLEZ-BAÑOS; LATOMBE, 2002).

A partir de um número calculável de candidatos $q \in N_amostra$, é aplicada a equação 27 que dará a pontuação de cada candidato q para possível escolha.

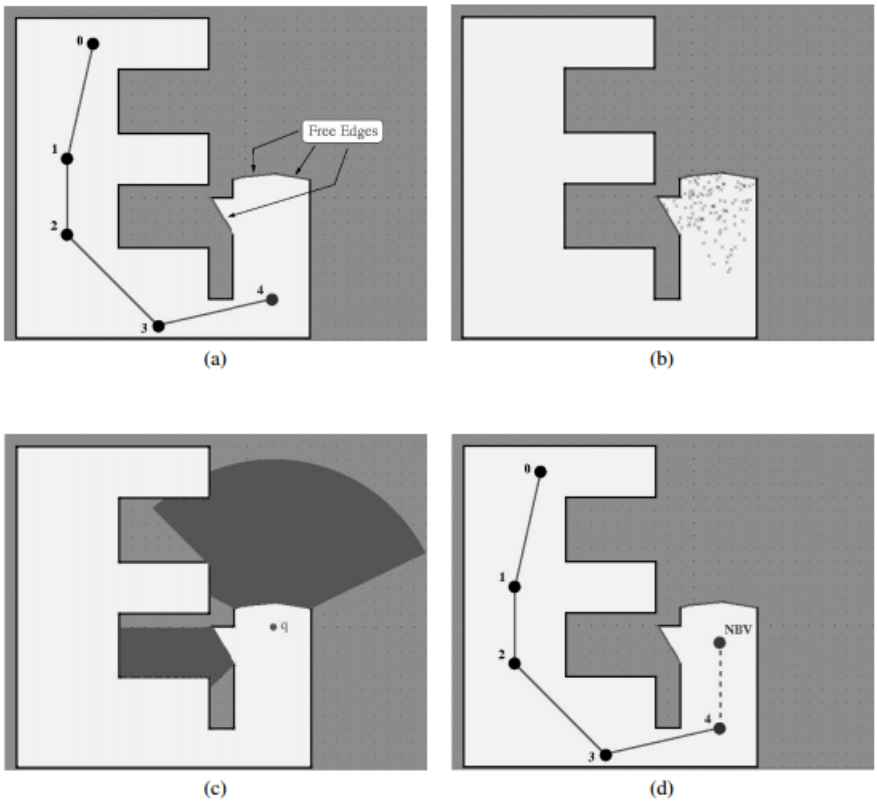
$$g(q) = A(q)\exp(-\lambda L(q)) \tag{27}$$

Onde $L(q)$ é o comprimento do caminho gerado para navegação entre os pontos, $A(q)$ é a área da região visível não explorada e λ é uma constante que serve como peso para ajustar o ganho de informação em relação ao caminho a ser percorrido. Quanto maior for o λ maior será o peso da distância a ser percorrida para encontrar o melhor candidato (GONZÁLEZ-BAÑOS; LATOMBE, 2002).

Para ilustrar melhor como é feita a seleção do próximo candidato a visitação, tem-se a figura 16.

Existem outros métodos que utilizam o ganho da informação, como mostra o trabalho de Stachniss, Grisetti e Burgard (2005). Neste estudo foi implementado o método *Information Gain Based Exploration*, o qual utiliza em sua aplicação o algoritmo RBPF. O princípio fundamental deste método é baseado em tomar a decisão pela seleção da partícula que minimiza as incertezas dos sensores, buscando um fechamento de laço ou o caminho mais curto, ao mesmo tempo que fornece maior informação do ambiente. Tudo isso é calculado e pode ser chamado de entropia, ou seja, a partícula que apresentar a maior variação de entropia conterá a trajetória a ser escolhida. O método de exploração depende da capacidade e eficiência da técnica RBPF aplicada, que é diretamente proporcional ao número de partículas criados. Entretanto, quanto

Figura 16 – a) Representação dos quatro pontos já explorados e das fronteiras (*free edges*). b) Criação dos pontos candidatos à exploração q a partir do ponto q_4 com visibilidade para as regiões desconhecidas. c) Representação de $A(q)$ (potencial ganho de informação) do ponto q selecionado. d) seleção do ponto NVB que teve melhor pontuação $g(q)$



Fonte: González-Baños; Latombe, 2002, p. 835.

maior for a quantidade de partículas, maior será a complexidade e o custo computacional, dificultando a decisão do número de partículas ótimo a fim de obter o equilíbrio entre a capacidade computacional e a eficiência desejada (AGUNBIADE; ZUVA, 2018).

Apesar de serem as mais utilizadas na literatura, as abordagens com ganhos e fronteiras podem apresentar algumas limitações durante suas execuções, como por exemplo: o ambiente não ser completamente coberto pelo robô; o robô andar em círculos em regiões com mínimo local; e áreas já visitadas sendo revisitadas. Todos estes pontos fazem diminuir a eficiência da exploração.

Um exemplo de solução para essas questões é proposta por Gao et al. (2018). A abordagem proposta realiza uma melhoria do método FBE através da implementação de um ganho para os pontos de fronteira atribuindo maior importância para rotação do robô na projeção do trajeto planejado até determinado nó. O método cria nós e grafos (mapeamento topológico) para manter a localização das regiões exploradas e marca os pontos que já foram visitados mantendo na memória os nós que ficaram para trás para posteriormente serem visitados. Os resultados provam que a velocidade de exploração em ambientes mais complexos diminui de

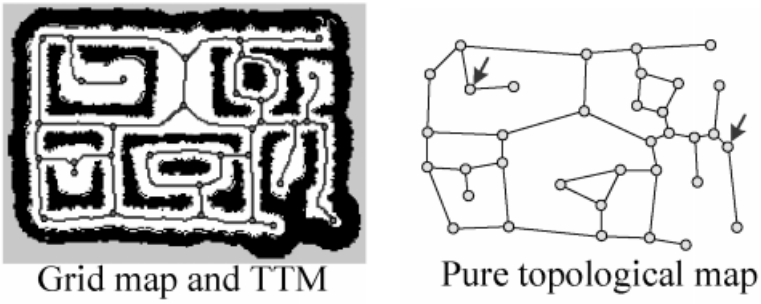
maneira significativa, porém ainda há dificuldade no trabalho apresentado em definir um sistema de coordenadas global estável para que a revisitação de nós que ficaram para trás seja feita de maneira mais eficiente.

3.1.3 Exploração com Mapeamento Topológico

O mapeamento topológico é essencial para muitos algoritmos de navegação e exploração mais recentes, principalmente para definição de trajetórias e estimativas de possíveis trajetórias para definição de um ponto de interesse. Ao realizar o mapeamento topológico é possível criar grafos com nós conectados entre si e atribuir características e pesos a pontos específicos do espaço em conjunto com mapas de grade de ocupação, como mostra os principais trabalhos citados nesta seção.

No contexto da exploração através do mapeamento topológico, Kwon e Song (2006) propõem um método chamado *Thining-based Topological Exploration* (TBTE). Este método cria grafos em conjunto com MGO acompanhando os limites do ambiente mapeado, desbastando os espaços vazios até sobrarem apenas linhas, como mostra a figura 17. Os nós vão sendo criados conforme há a exploração do ambiente. As tomadas de decisão do robô são feitas em cima da probabilidade de encontrar um nó final (nó ao qual não haverá mais de uma ligação) e na distância do robô para esse nó, ou seja, quanto maior for a probabilidade de ser um nó final e mais perto o robô estiver deste nó, menor será o custo do nó e maior será a chance de ser escolhido para fins de navegação. Conforme ocorre a tomada de decisão e a exploração segue, a posição relativa do robô é obtida pelo EKF e os lasers vão delimitando os contornos do mapa (KWON; SONG, 2006).

Figura 17 – Representação do ambiente com MGO e Mapeamento topológico (TTM do inglês *Thining Topological Map*).



Fonte: Kwon; Song, 2006, p. 345.

As vantagens apuradas no decorrer dos experimentos de Kwon e Song (2006) são: Foi atestada a eficiência do método para exploração de ambientes desconhecidos em locais sem ambientes dinâmicos e não necessita um algoritmo extra de planejamento de percurso; e a possibilidade de posição se mostrou útil para seleção de próximo nó a ser seguido e faz com

que o robô não precise ir diretamente a todos os nós para completar a exploração do ambiente. Suas maiores limitações são em ambientes muito abertos e com objetos no caminho do sensor.

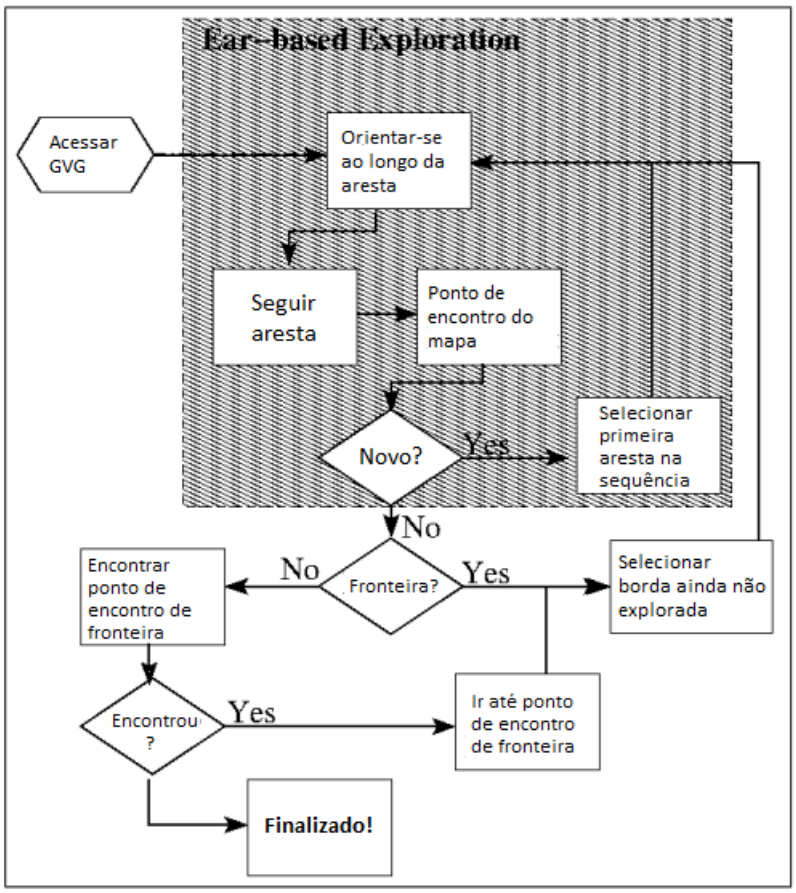
Outras abordagens variam a abordagem de SLAM para atingir o objetivo da exploração autônoma. Um exemplo é a utilização do algoritmo FastSLAM, que é uma extensão do RBPF, como mostra o trabalho de Stachniss, Hahnel e Burgard (2004). A abordagem proposta usa mapeamento topológico, FastSLAM e FBE para representar o ambiente, criar as partículas no espaço e realizar a exploração de ambientes desconhecidos. Conforme ocorre a exploração do ambiente, o robô tende a perder e acumular os ruídos dos sensores de posição, com isso, os autores propõem que o robô busque um nó já visitado anteriormente, que permitirá o fechamento de laço e, consequentemente, a redução das incertezas. Para isso o nó tem que ser acessível a partir da posição da partícula em que o robô se encontra e a distância até esse fechamento de laço deve ser menor do que a distância a ser percorrida caso o robô fizesse o caminho voltando por todos os nós visitados. A principal dificuldade desta técnica é a incapacidade de se recuperar de divergências, necessitando o reinício do processo (STACHNISS; HAHNEL; BURGARD, 2004).

O trabalho de Zhang et al. (2014) é mais recente e propõe uma abordagem do problema do fechamento de laço, bastante similar à abordagem apresentada por Stachniss, Hahnel e Burgard (2004). A representação topológica é como a utilizada nos trabalhos de Choset e Burdick (2000) e Kwon e Song (2006), utiliza-se o GVG em conjunto com a representação métrica de MGO. A estratégia fundamental deste algoritmo é a decisão entre ir até um nó para exploração ou para relocalização. Depois de completar um passo de exploração ou relocalização, o robô decide se explora um novo território com probabilidade p_{exp} ou se relocalizar com probabilidade 1 - p_{exp}, além disso quando o robô atinge um acúmulo de incertezas que ultrapassa um limite pré-definido, o robô vai até um ponto conhecido para se relocalizar, analogamente ao que é feito no trabalho de Stachniss, Hahnel e Burgard (2004).

Resumidamente, o diagrama da figura 18 mostra como é feita a tomada de decisão para seleção do próximo ponto em relação à exploração em conjunto com a representação topológica do ambiente com GVG.

Na parte inicial do processo, o robô deve atingir uma posição na aresta do GVG. Após se posicionar corretamente no meio entre dois obstáculos, o robô deve rotacionar até estar alinhado com a aresta GVG que ele se encontra. Em seguida o robô navega pela aresta se controlando tentando se manter equidistante de dois obstáculos, e quando três ou mais obstáculos são detectados, o robô busca algum ponto de encontro, e faz isso diminuindo-se a velocidade para maior precisão. Ao encontrar este ponto de encontro, o robô mapeia e adiciona esse nó ao grafo. Usando o racional de escolher o ponto de exploração ou o ponto para relocalização, através da EBE, o robô pode selecionar a próxima aresta que irá seguir (ZHANG et al., 2014). A exploração termina quando não existem mais nós sem terem sido visitados.

Figura 18 – EBE utilizando a representação em GVG



Fonte: Autor "adaptado de"Zhang et al., 2014, p. 3083.

3.1.4 Abordagens Diversas

O trabalho que serviu como base para a pesquisa foi feito por J. Park e Song (2015) que trata muitos aspectos relacionados ao SLAM. Eles propõem a realização da fusão de sensores com uma técnica baseada no EKF para localização, criação do mapa usando MGO e uma estratégia de exploração baseada na fusão dos sensores *EKF Based Exploration* comparando-a com outras três conhecidas na literatura: *Frontier Based Exploration* (FBE), *Next Best View Exploration* (NBVE) e *Thining Based Topological Exploration* (TBTE).

Para realização do mapeamento e comparação de técnicas é necessário levar em conta a eficiência da exploração. Segundo (KWON; SONG, 2006), a eficiência refere-se a quão rápido um robô consegue mapear o ambiente completamente percorrendo a menor distância possível. Sendo assim, A estratégia de exploração do método proposto por J. Park e Song (2015) consiste em receber informações do ambiente através dos sensores e gerar nós nas fronteiras com regiões desconhecidas, em forma de grafos, como possíveis candidatos para visitação do robô. A estes nós são atribuídos pesos a partir de racionais específicos que resultarão em um ganho para cada nó.

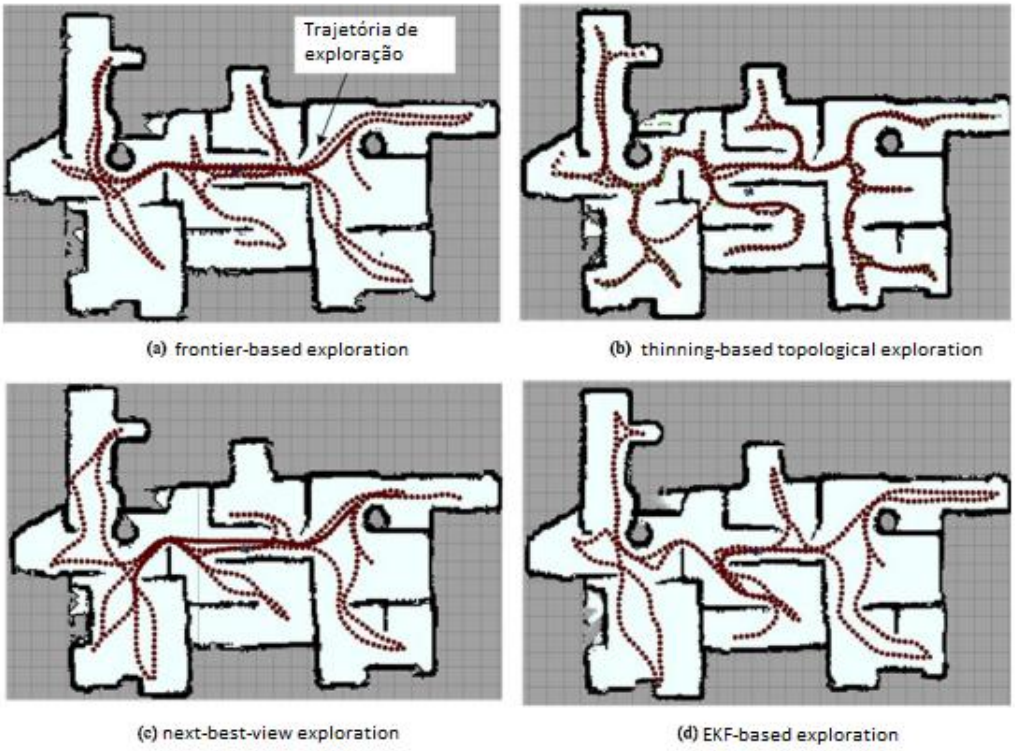
O estudo demonstra alguns métodos para calcular os pesos dos nós, o primeiro deles é o do peso relacionado à informação do ambiente, que consiste na quantidade de informação que é possível encontrar em um dado nó, ou seja, quanto maior a entropia, maior será o peso de seu respectivo nó. O peso da localização é feito baseando-se na quantidade de objetos conhecidos, identificados pela câmera e localizados no mapa, que possivelmente serão encontrados no caminho, portanto quanto mais referências o robô tiver, maior será a precisão do mapeamento, o que confere um peso maior para determinado nó. Além disso, essa informação de objetos de referência fixos são utilizados com o EKF para realizar a fusão de sensores e corrigir a localização do robô. Por fim, o peso relacionado ao deslocamento diz respeito às distancias entre os nós e a quantidade de mudança de direções que o robô deveria fazer para atingir determinado objetivo, ou seja, quanto mais simples o caminho, maior o peso desse nó durante a seleção. Com os três cálculos dos pesos para cada nó obtidos, é proposto um racional que leva em conta os três parâmetros e a imprecisão da localização do respectivo nó e, assim, é obtido cada ganho para escolha do próximo destino (J. PARK; SONG, 2015).

Os resultados atingidos por J. Park e Song (2015) mostram que o método proposto é mais eficiente que os métodos usados como comparação em ambientes complexos, devido principalmente às escolhas dos racionais dos nós gerados, e possui um resultado similar para ambientes mais simples. A figura 19 mostra o percurso resultante da aplicação das técnicas em um ambiente complexo.

Muitos trabalhos propõem melhorias baseando-se principalmente nas aplicações do robô. Existem outras abordagens na literatura, no entanto, apresentou-se as que mais se adequaram ao problema descrito no presente trabalho, ou seja, ambientes internos e planos, robôs de serviço, com apenas um robô em operação e fusão de sensores. Sendo assim, observa-se que as abordagens de exploração autônomas são muito complexas e envolvem diversas outras tarefas que podem variar e impactar significativamente a eficiência do objetivo final.

Os métodos FBE e NBVE, por exemplo, também podem ser utilizados variando-se os tipos de sensores que fornecem dados para exploração, portanto, destacam-se no contexto deste trabalho dois estudos distintos. O estudo de Jianhao, Meiqin e Weihua (2015) utiliza uma câmera RGB-D (Kinect) para extração de pontos de interesse com SURF (do inglês *Speeded Up Robust Features* para o mapeamento 3D e utiliza o método NBVE para exploração. No entanto, apesar de conseguir realizar o mapeamento de forma robusta em tempo real, os autores ainda buscam aprimoramentos para aplicação em ambientes mais complexos e dinâmicos. Já o estudo de Sun et al. (2017), propõe um algoritmo para criação de MGO unindo os sensores laser LIDAR e Kinect para implementar o método de exploração FBE. Esta configuração permite mapear os limites tanto na direção horizontal (com o LIDAR) quanto na direção vertical (com o Kinect), pois apesar do laser ser mais preciso, a informação do Kinect possibilita identificar obstáculos em todas as direções. O principal problema identificado pelos autores é lidar com a atualização do mapa quando um obstáculo que estava em determinado ponto desaparece.

Figura 19 – Resultado experimental da aplicação das técnicas *Frontier Based Exploration*, *Next Best View Exploration*, *Thining Based Topological Exploration* e *EKF Based Exploration* na exploração de um ambiente desconhecido.



Fonte: Autor "adaptado de"Park, J.; Song, 2015, p. 81.

O capítulo 4 aborda os princípios das técnicas e pacotes implementados no projeto, são eles: *explore-lite*, *RRT-exploration* e *cam-exploration*. Os três são pacotes implementados no *ROS* e foram adaptados para funcionar no HERA para executar exploração em ambientes internos.

Essas técnicas foram escolhidas por terem sido umas das poucas específicas implementadas e divulgadas na comunidade de sistema ROS. O *explore-lite* usa o princípio mais simples de exploração, que é o baseado em fronteiras e ganhos simples, o *RRT-exploration* é uma técnica que utiliza árvores de busca *RRT* em formato de mapeamento topológico, e o pacote *cam-exploration* utiliza exploração com fronteiras em conjunto com ganhos e algumas alternativas para sair de situações que impeçam a continuidade da tarefa. Dessa forma, os principais conceitos de exploração estarão contemplados.

A seção 4.4 apresenta as descrições dos algoritmos e pacotes utilizados para testes no projeto.

4 METODOLOGIA

A metodologia do projeto pretende avaliar três pacotes implementados no ROS para fins de exploração autônoma em ambientes internos: o *explore-lite*, o *RRT-exploration* e o *cam-exploration*. Os critérios de avaliação serão similares aos utilizados nos trabalhos de J. Park e Song (2015) e Pimentel (2016): a distância total percorrida, a necessidade de rotacionar, o tempo gasto para completar a exploração e a capacidade em cobrir a exploração do mapa por completo.

O capítulo é organizado da seguinte forma: a seção 4.1 apresenta a configuração geral da plataforma robótica utilizada e os desafios impostos pela Robocup. Na seção 4.2 é apresentado o conceito básico do *framework* ROS. Na seção 4.3 são apresentados os pacotes utilizados na manipulação e fusão de sensores. Na seção 4.4 são detalhados os pacotes de exploração avaliados, e por fim, na seção 4.5 descreve-se como são feitos os testes e as configurações possíveis avaliadas nos experimentos.

4.1 ROBOCUP E PLATAFORMA

Para realização dos testes e implementação dos algoritmos será utilizada a plataforma robótica HERA em ambiente simulado, mostrada na figura 3. O ambiente será simulado no Gazebo onde serão realizados todos os testes relacionados à implementação das configurações de sensores, dos pacotes auxiliares e dos pacotes de exploração, SLAM e navegação.

4.1.1 Regras ROBOCUP@HOME

O trabalho proposto visa superar desafios impostos dentro do contexto da Robocup, por isso nesta subseção estão descritos os itens mais importantes que podem influenciar na exploração autônoma em ambientes internos totalmente embasados no livro de regras de Matamoros et al. (2019).

Como em uma aplicação real em ambientes internos, a competição traz os seguintes cenários: quarto, sala de jantar, sala de estar e cozinha (MATAMOROS et al., 2019). No trabalho será considerado um ambiente semelhante ao de uma residência nas arenas propostas. Nesse contexto, o livro de regras define as paredes como fixas, que não se modificam durante a competição, contudo não teremos problemas em relação a isso, pois em um ambiente normal as paredes não costumam se movimentar, da mesma forma com os pisos, no entanto, as portas podem estar ou não abertas, e isso não pode afetar o desempenho do robô. A aparência das paredes, portas ou objetos em competições e ambientes reais poderão ser translúcidas, portanto, com os sensores ultrassônicos seria possível minimizar esse tipo de problema.

No livro de regras, as seguintes mobílias poderão ser encontradas em um ambiente de sala de estar: cama, sofá, mesa pequena, mesa de jantar com duas cadeiras, dois cestos de lixo,

um armário aberto ou uma mesa pequena com televisão e controle remoto, um armário com gavetas e uma estante ou prateleira com alguns livros dentro. O ambiente simulado proposto contará com mesas, estantes e latas de lixo.

Na competição os móveis maiores e menores podem sofrer alterações de posição, portanto é necessário que seja feita a exploração de maneira contínua e a navegação deve conseguir identificar possíveis alterações. No entanto, neste projeto, o ambiente será considerado estático.

As competições envolvem reconhecimento e manipulação de objetos, no entanto, a manipulação está fora do escopo do trabalho proposto. O reconhecimento será feito a partir do Kinect e da Xtion, no entanto não haverá diferenciação entre eles, ou seja, tudo que é detectado pela união dos sensores é tratado como um obstáculo que deve ser evitado no planejamento das trajetórias durante a navegação.

4.1.2 Plataforma

O robô HERA possui uma configuração que permite a navegação em ambientes internos domésticos.

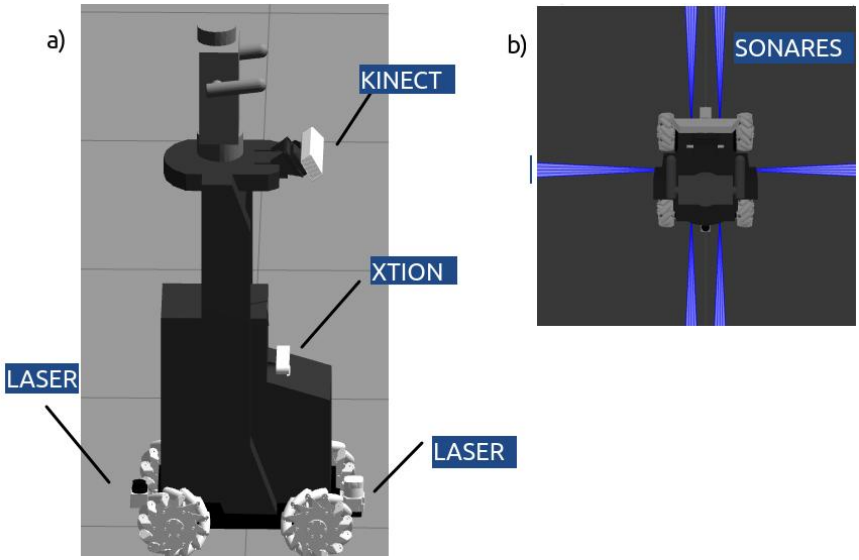
As escolhas dos sensores foram feitas baseando-se nos que mais foram utilizados nos trabalhos recentes no contexto do SLAM e nos disponíveis na configuração atual do robô HERA[1] (PIMENTEL F., 2020). A figura 20 indica as posições dos principais sensores utilizados no robô. Na parte frontal, tem-se o Kinect inclinado na parte superior, o Xtion apontado para frente acoplado ao torso e o laser acoplado à base. Na parte inferior traseira, tem-se o laser acoplado na base. Pela vista superior é possível mostrar a posição dos sonares ao redor da base. A disposição dos sonares foi para que fosse simulada a substituição dos lasers para realização do SLAM e para detecção de obstáculos que não poderiam ser captados pelos lasers ou câmeras. A disposição das câmeras permite que o robô consiga captar obstáculos em todas as direções na parte frontal do robô. Os lasers são utilizados principalmente para realização do SLAM.

Para realização dos testes, os sensores são testados e combinados para avaliação dos desempenhos e configurações mínimas para atuação em um ambiente residencial tanto para realização do SLAM quanto para navegação e exploração, como apresentado na com mais detalhes na seção 4.5.

Para utilizar os sensores em conjunto, alguns pacotes devem ser implementados. Na seção 4.3 estão apresentados os pacotes utilizados para unir as características de cada sensor para implementação dos pacotes de SLAM, navegação e exploração.

[1]Disponível em: <https://gitlab.com/fpimentel/hera> Acesso em: 01 de abr. 2020

Figura 20 – a) Representação da lateral do HERA no simulador Gazebo, indicando as posições dos sensores lasers e das câmeras Xtion e Kinect. b) Posição dos sensores ultrassônicos a partir da vista superior.



Fonte: Autor.

4.2 ROS (*ROBOT OPERATING SYSTEM*)

Muitos algoritmos foram estudados e avaliados para implementação neste projeto. Três pacotes relacionados à exploração foram escolhidos para testes, sendo que todos seguem a abordagem de exploração através de fronteiras e ganhos. Neste estudo, foram escolhidas as técnicas que permitissem a utilização das mesmas abordagens de SLAM e navegação, e que já possuem implementação no ROS. Os pacotes escolhidos foram: *explore-lite*, *RRT-exploration* e *cam-exploration*.

O sistema ROS tem sido cada vez mais utilizado nos trabalhos relacionados à robótica e possui uma plataforma *open-source*. Sua estrutura básica é formada blocos de executáveis conhecidos como nós, que se comunicam entre si por tópicos, mensagens e serviços dependendo das aplicações. É possível a implementação com comunicação entre múltiplos computadores, que permite facilmente o uso de robôs em paralelo, processamento de algoritmos remotamente e gerenciamento de pacotes (ROS.ORG, 2018).

Sua escolha foi feita pelo fato de ROS ter compatibilidade com o Python e C++, linguagens utilizadas nos algoritmos implementados, e por ser possível a implementação em conjunto com o Gazebo para realização de simulações. Portanto, é o sistema que conectará todos os aspectos dos experimentos que serão realizados.

4.3 PACOTES AUXILIARES

Quando necessita-se da união de sensores com características diferentes para determinadas aplicações, alguns tratamentos devem ser feitos, principalmente pelo fato de que sensores distintos geram mensagens distintas, conhecidas como *sensor_msgs*. As mensagens utilizadas no projeto e geradas no *framework* ROS pelos sensores são:

a) ***IMU***: Mensagens referidas ao sensor IMU. São formadas pela obtenção da variação da orientação, da velocidade e aceleração angulares. É utilizada principalmente junto ao pacote *robot-localization* que implementa o EKF;
b) ***LaserScan***: São as mensagens geradas pelo sensor a laser e é caracterizado por emitir pontos de distância paralelamente ao plano referencial escolhido do robô. Essas mensagens são utilizadas principalmente pelo pacote *gmapping* para aplicação do SLAM, e o referencial para esta aplicação deve ser o plano x-y;
c) ***PointCloud2***: Mensagens geradas pelo Kinect e pelo Xtion. São representadas como uma nuvem de pontos que indicam a profundidade relativa de cada um e podem ser utilizadas para detecção de obstáculos;
d) ***Range***: Mensagens características de sonares para detecção de obstáculos a distância.

O primeiro pacote a ser implementado é o que possui a ferramenta que converte as mensagens *PointCloud2* em mensagens *LaserScan*. Essa conversão deve ser feita para que o robô possa utilizar dados de câmeras RGB-D com o pacote de SLAM *gmapping*. O nome do pacote é *nodelet* e permite a utilização de vários algoritmos funcionando ao mesmo tempo. O nó que faz essa conversão chama-se *depthimage_to_laserscan*[2] e foi desenvolvido por Rockey, C. (2018).

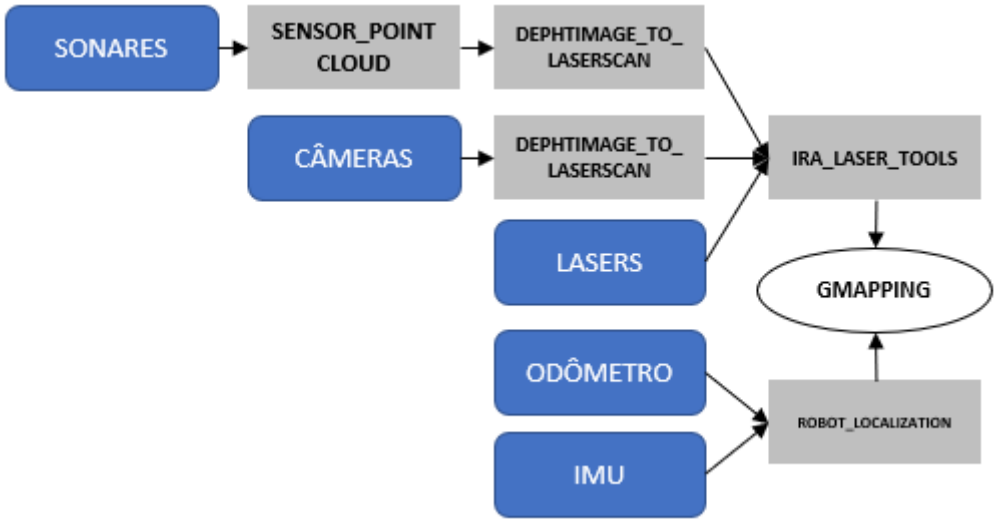
O segundo pacote implementado é o que transforma mensagens *Range* dos sonares em *PointCloud2*. O pacote desenvolvido por Lim, E. (2017) chama-se *sensor_pointcloud*[3] e é aplicado como uma etapa para utilização do sonar no *gmapping*. Na etapa final, para que seja possível esta implementação, utiliza-se pacote *depthimage_to_laserscan* para que as mensagens dos sonares transformadas em *PointCloud2* se transformem em *LaserScan*. Na seção 5.1 há uma discussão a respeito destas transformações no uso destes sensores para a aplicação no SLAM.

O último pacote utilizado para transformar os dados obtidos dos sensores de distância para a aplicação do pacote *gmapping* é o *ira_laser_tools*. Desenvolvido por iralabdisco (2014), este pacote une mensagens no formato de *LaserScan* em um único tópico. Esta etapa é necessária, pois o pacote *gmapping* só permite receber mensagens de um tópico por vez.

Para utilização em ambientes reais o pacote *robot-localization* é o recomendado para utilizar mensagens de odometria do robô em conjunto com mensagens *IMU* para reduzir o erro acumulado na localização. Sua base é o EKF como apresentado na subseção 2.6.2.

[2]Disponível em: <http://wiki.ros.org/depthimage_to_laserscan> Acesso em: 01 de abr. 2020
[3]Disponível em: <https://github.com/eliotlim/sensor_pointcloud> Acesso em: 01 de mar. 2020

Figura 21 – Arquitetura do tratamento dos dados provenientes dos sensores para aplicação do pacote *gmapping* para a realização da tarefa do SLAM.



Fonte: Autor.

A arquitetura simplificada que possibilitaria a utilização de todos os sensores de distância em conjunto para o SLAM está apresentada na figura 21. Os sensores são representados pelos retângulos azuis com cantos arredondados, os pacotes auxiliares são representados pelos retângulos cinzas e o pacote *gmapping* recebe os dados após os tratamentos no balão branco com contorno preto escrito "GMAPPING". Para simulação, as mudanças na arquitetura são: a utilização da odometria do robô virtual diretamente como entrada do *gmapping*, sem necessidade de passar pelo EKF; e os sensores de distância são variados e alternados conforme as configurações propostas para testes no projeto.

4.4 ALGORITMOS DE EXPLORAÇÃO

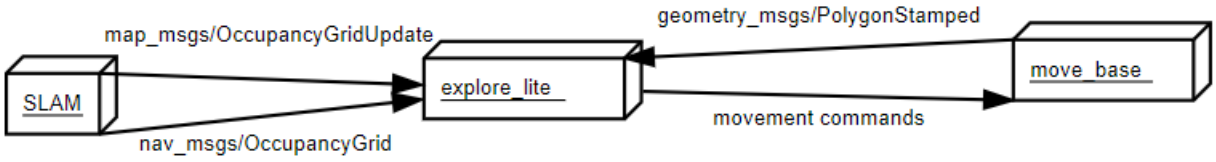
Esta seção detalha os algoritmos que são utilizados nesta pesquisa para a realização da exploração autônoma. Os pacotes descritos são: *explore-lite*, *RRT-exploration* e *cam-exploration*.

4.4.1 *explore-lite*

O primeiro algoritmo implementado chama-se *explore-lite*[4]. Este algoritmo utiliza a abordagem de exploração através de fronteiras com ganhos. Seu princípio básico consiste em identificar as fronteiras através do MGO, calcular as distâncias para as fronteiras candidatas através do cálculo do planejamento global de trajetória, selecionar a candidata mais próxima e, por fim, acionar o algoritmo *move-base*, que pertence ao conjunto de pacotes *navigation*, para os planejamentos local e global da trajetória.

[4]Disponível em: <http://wiki.ros.org/explore_lite> Acesso em: 01 de fev. 2020

Figura 22 – Arquitetura do algoritmo *explore-lite*.



Fonte: Horner J., 2017.

Este pacote foi desenvolvido como parte da tese de Hörner (2016). Seu trabalho foca na exploração de ambientes através de multi-robôs, todavia, o pacote criado serve para exploração no geral e é de fácil implementação. Seu princípio básico segue a estrutura mostrada na figura 22. Nesta estrutura à esquerda tem-se o bloco representando o nó SLAM, que pode ser composto por quaisquer pacotes, como o *gmapping* utilizado no trabalho, desde que possam gerar dois tipos de mensagem:

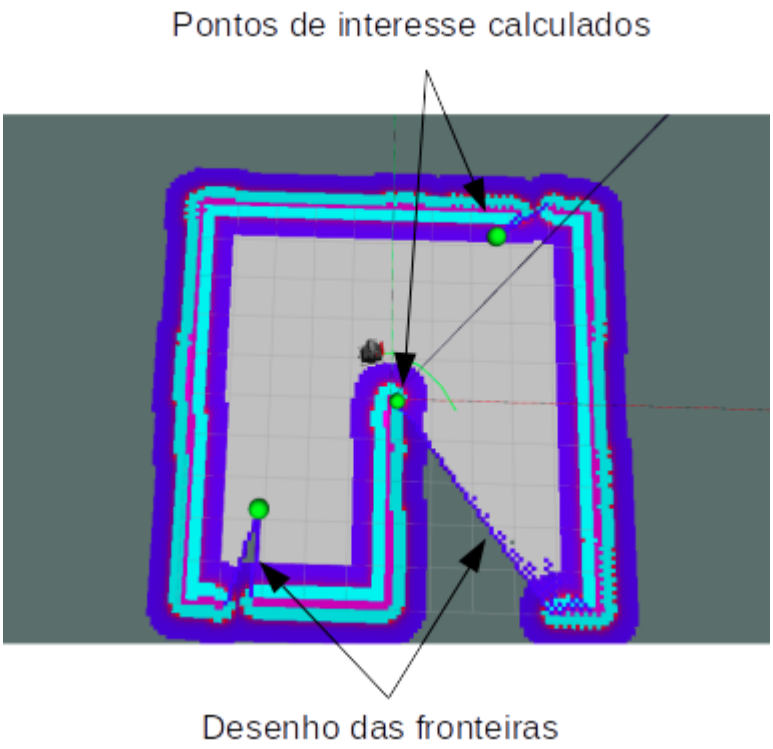
a) *map_msgs*: são mensagens que possuem informações sobre o mapeamento, no caso as mensagens que precisam ser enviadas são as atualizações do mapeamento do ambiente no formato de grade de ocupação;
b) *nav_msgs*: são mensagens para fins de navegação, no caso, envia-se a informação do *costmap* para avaliação do algoritmo *explore-lite*.

O nó *explore-lite* recebe essas duas mensagens e faz a avaliação das fronteiras e dos candidatos. O algoritmo interage com o pacote de navegação *move-base* no nó à direita. Na interação, dois tipos de mensagens são trocados:

a) *movements comands*: com o cálculo das fronteiras, comandos de movimentos, indicando o próximo objetivo no mapa, são enviados para o algoritmo *move-base* que fará os planejamentos global e local até o ponto de interesse;
b) *geometry_msgs*: são mensagens que contém formas, pontos ou linhas, e na aplicação serve para trazer as fronteiras para os cálculos do *explore-lite*.

O ponto principal do algoritmo de exploração é identificar as fronteiras possíveis e avaliar a melhor a ser escolhida para visitação. O algoritmo recebe as fronteiras e atribui ganhos a elas, através do cálculo da distância e do tamanho da fronteira. O cálculo da distância é feito utilizando o planejador de trajetória global clássico, o *navfn* pertencente ao *navigation*, como projeção do percurso e distância a ser percorrida para todos os possíveis pontos (HÖRNER, 2016). É importante salientar que é possível modificar os parâmetros para melhor adequação e ajuste do comportamento do funcionamento do robô, sendo que os principais são: tamanho mínimo de uma fronteira e proporção do ganho entre tamanho da fronteira e distância até ela. A figura 23 mostra um exemplo prático da utilização do algoritmo através do *rviz*. Na imagem, tem-se um ambiente simples com a parte mais clara representado a região do ambiente já explorada, nos contornos desta região estão representados os mapas MGO e *costmap2D* e indicados

Figura 23 – Visão do funcionamento e princípios do *explore-lite* através do *rviz*.



Fonte: Autor.

pelas setas estão os pontos de interesse que serão calculados (círculos verdes) e o contorno das fronteiras. Os pontos de interesse são sempre desenhados no local mais próximo ao robô.

O cálculo do ganho é governado pela seguinte equação:

$$Ganho = 1/(PotentialScale * DistMin - GainScale * TamFront) \tag{28}$$

Na equação 28 os termos *PotentialScale* e *GainScale* são as constantes que podem ser escolhidas no momento de aplicar o algoritmo. Basicamente, essa relação mostra que quanto maior for a fronteira ($TamFront$) e menor for a distância ($DistMin$) até ela, maior será o ganho e, consequentemente, terá mais chances de ser escolhida como próximo destino.

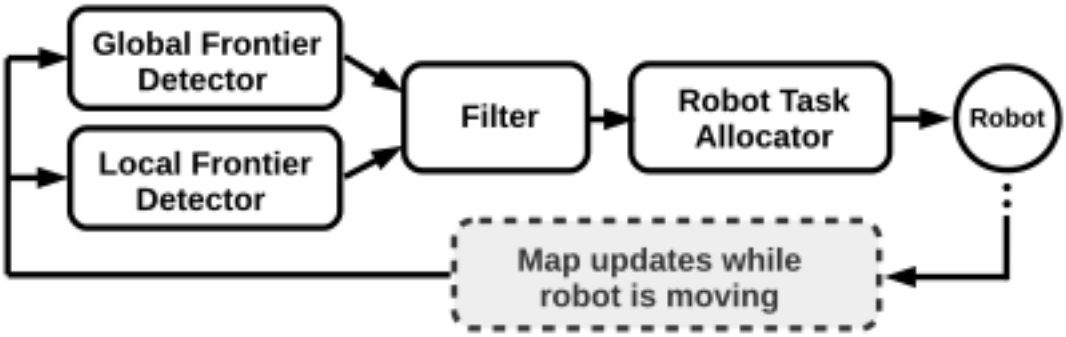
4.4.2 *RRT-exploration*

O pacote de exploração *RRT-exploration*[5] é desenvolvido por hasauino (2014) e implementado no sistema ROS. Baseia-se em encontrar as fronteiras através do método *RRT*, seguindo os princípios descritos na subseção 2.7.3. A grande diferença entre o proposto por Umari e Mukhopadhyay (2017) e o descrito por Lavalle (1998) é que a árvore de busca não monta a trajetória de um estado inicial para um estado final, mas sim, busca pontos em um

[5]Disponível em: <https://github.com/hasauino/rrt_exploration> Acesso em: 01 de fev. 2020

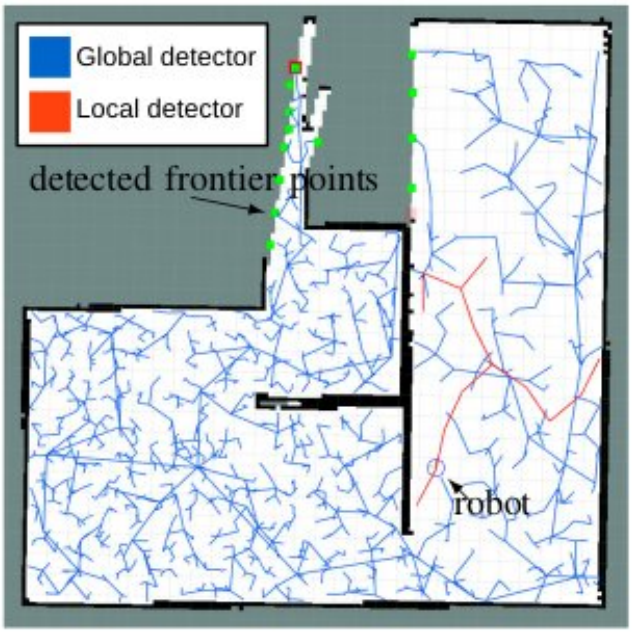
MGO que estão em espaços desconhecidos para encontrar as fronteiras. Para realizar essa tarefa, o diagrama apresentado na figura 24 ilustra o funcionamento do pacote.

Figura 24 – Visão macro da estrutura de funcionamento do pacote *RRT-exploration*.



Fonte: Umari; Mukohhpadhyay, 2017, p. 1397.

Figura 25 – Representação da criação das árvores de exploração *RRT* global e local para busca das fronteiras, sendo que em vermelho é a representação da detecção local e em azul da detecção global, e ambos estão em um MGO.



Fonte: Umari; Mukohhpadhyay, 2017, p. 1398.

As primeiras entradas do algoritmo são o mapa e a posição relativa do robô, no caso um MGO. Os pontos-chave do método são os detectores global e local de fronteiras. Sendo assim, árvores de busca são criadas a partir da posição do robô. A detecção local de fronteiras aplica o *RRT* para encontrar a fronteira mais próxima, no entanto, é uma trajetória por vez que chega até um obstáculo ou atinge uma fronteira, como mostrado em vermelho na figura 25. Ao atingir o ponto de fronteira, envia-se as coordenadas para o filtro. No caso da detecção global de fronteiras, as árvores vão sendo criadas e expandidas a partir do ponto inicial indicado no

início do funcionamento do pacote. Conforme o tempo passa as árvores expandidas vão criando ramificações cada vez menores, como mostrado em azul na figura 25 e os pontos de fronteira encontrados também são enviados para o filtro. Importante destacar que as árvores apenas são expandidas no interior de áreas conhecidas do mapa.

Todos os pontos enviados pelos detectores ao filtro são avaliados e os pontos antigos ou muito próximos são eliminados, deixando uma disposição conforme aparece em quadrados verdes na figura 25. Estes pontos filtrados são enviados para o alocador para que sejam atribuídos os pesos para escolha do melhor próximo destino para o robô.

O próximo destino ideal segue a seguinte métrica para escolha:

a) **Custo da navegação** (N): o custo de navegação é definido pela distância entre a posição atual do robô e o ponto da fronteira, essa distância é obtida simplesmente pela norma da diferença entre os dois pontos, para diminuir os custos computacionais;

b) **Ganho de informação** (I): o ganho de informação é medido a partir da quantidade de informação possível de ser obtida com a visita do robô. O cálculo é feito com base na quantidade de células desconhecidas dentro de um raio determinado pelo usuário.

c) **Peso final** (R): as equações 29 e 30 são as que governam a obtenção do peso final de todos os pontos de fronteira x_{fp} para a escolha de qual será o próximo destino, dado o ponto da posição atual do robô x_r.

$$R(x_{fp}) = \lambda.h(x_{fp},x_r)I(x_{fp}) - N(x_{fp}) \tag{29}$$

$$h(x_{fp},x_r) = \begin{cases} 1 & \text{se } \|x_r - x_(fp)\| > \text{h}_{rad} \\ \text{h}_{ganho} & h_{ganho} > 1 \end{cases} \tag{30}$$

λ é a constante que da o peso para a primeira parte da equação 29 e é definida pelo usuário do pacote. h é obtido pela equação 30 que refere-se ao ganho por histerese. Seu resultado é igual a 1 se o x_{fp} está fora de um determinado raio h_{raio} a partir da localização x_r, sendo que h_{raio} também é dado pelo usuário através da experiência em dada aplicação. já no segundo caso, onde $h(x_{fp},x_r) = h_{ganho}, h_{ganho} > 1$, é quando o ponto encontra-se dentro de determinado raio h_{raio}. Importante notar que h_{ganho} deve ser maior que 1, pois assim evita-se sobreposição e enviesa o robô a explorar regiões em sua proximidade. O resultado de R de todos os pontos são comparados e destina-se ao robô as coordenadas do ponto de maior valor.

Ao obter o ponto de objetivo, o robô utiliza um algoritmo de navegação para chegar até o destino. No caso, tanto no trabalho de Noreen, Khan e Habib (2016) quanto no presente trabalho, utiliza-se o *move-base* para navegação.

O pacote pode ser utilizado para um único robô ou múltiplos robôs dependendo da aplicação. Logo, para a presente proposta, apenas o robô HERA será considerado para implementação.

4.4.3 *cam-exploration*

O último pacote para exploração utilizado para os testes chama-se *cam-exploration*[6]. Morenza-Cinos et al. (2017) desenvolveram esse pacote com o intuito de criar uma exploração baseada em fronteiras para realização do mapeamento em 3D através de nuvens de pontos obtidos por câmeras RGB-D, pelo algoritmo *rtabmap_ros* (LABBE M., 2014), além disso, o algoritmo de exploração também permite o funcionamento do pacote utilizando uma representação em 2D do ambiente.

Da mesma forma que no caso do *explore-lite*, utiliza-se dois pacotes complementares para SLAM e navegação, logo também são implementados para a exploração em 2D o *gmapping* para o SLAM e o *move-base* para a navegação. Apesar dos princípios serem bem similares aos mostrados na subseção 4.4.1, há algumas particularidades que podem ser úteis na otimização da operação de exploração.

A primeira diferença introduzida é a capacidade do robô de realizar o replanejamento do objetivo quando se deparar com situações nas quais o robô ficaria preso, são elas:

a) ***not_moving***: nesta condição o robô está preso ou parado em um ponto, sendo assim, após passar um tempo sem se movimentar, envia-se um novo objetivo para o robô seguir a navegação;
b) ***too_much_time_near_goal***: quando o robô passa muito tempo próximo ao objetivo enviado mas não o alcança, envia-se um novo objetivo;
c) ***isolated_goal***: quando o objetivo não é próximo a nenhuma fronteira, envia-se outro objetivo.

Estes parâmetros teoricamente permitem que o robô não fique preso em ciclo sem conseguir sair de determinada situação e consiga completar sua tarefa adaptando-se aos possíveis imprevistos.

Para o cálculo do ponto de objetivo, também atribuem-se ganhos para os candidatos a próximo objetivo. Os parâmetros são:

a) ***Tamanho máximo***: atribui um ganho maior conforme o comprimento da fronteira for maior;
b) ***Mínima distância Euclidiana***: neste parâmetro, utiliza-se apenas a distância entre os pontos inicial, onde se encontra o robô, e final, fronteira que está sendo avaliada, não considerando os obstáculos. Quanto mais perto o ponto de fronteira estiver, maior será seu ganho.

[6]Disponível em: <https://github.com/UbiCALab/cam_exploration> Acesso em: 01 de abr. 2020

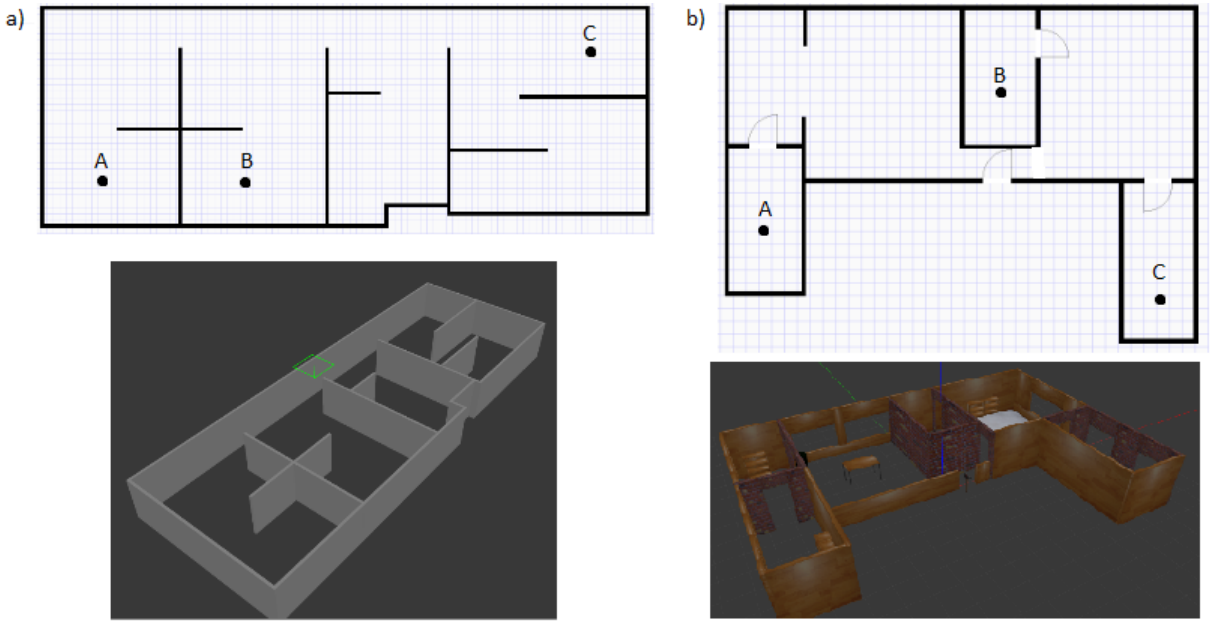
c) ***Distância mínima A****: aqui utiliza-se a distância a ser percorrida pelo robô levando em conta o planejamento global de trajetória, desviando de obstáculos, logo, quanto menor for essa distância, maior será o ganho atribuído ao ponto.

No pacote é possível alterar os pesos destes parâmetros conforme a necessidade de projeto. O ponto médio da fronteira é o escolhido como referência na avaliação de determinada fronteira (MORENZA-CINOS et al., 2017), diferente de como é feito no *explore-lite*. O próximo destino é escolhido quando a soma dos ganhos é superior às demais.

4.5 ESTRATÉGIAS DE AVALIAÇÃO

Para avaliar as estratégias de exploração e configuração de sensores a metodologia se desenvolve em dois cenários distintos: no primeiro cenário tem-se a arena 1 (figura 26.a)), que representa um ambiente residencial com passagens relativamente mais largas do que portas entre cômodos e sem móveis; no segundo cenário tem-se a arena 2 (figura 26.b)) que simula uma residência real com vários cômodos, móveis, obstáculos e passagens estreitas. Esta arena residencial foi obtida através do pacote *turtlebot3*. Nos dois cenários os pacotes foram calibrados e testados com diferentes configurações de sensores para obter-se a melhor estratégia de exploração para a aplicação proposta.

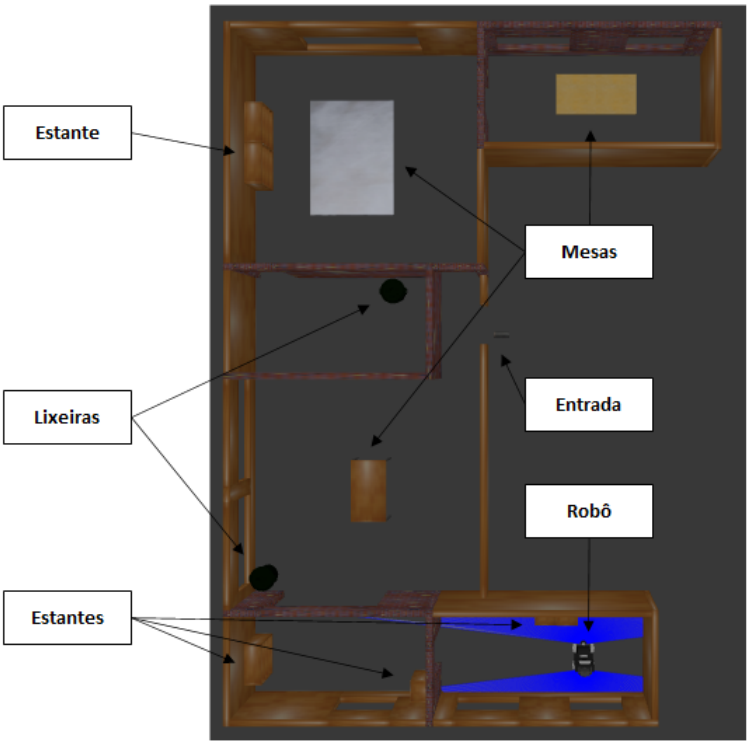
Figura 26 – Arenas utilizadas para os testes com a marcação "ABC"que indicam os pontos a partir dos quais o robô inicia a exploração. a) Arena 1: disposição mais simples, sem obstáculos pequenos e passagens relativamente largas entre cômodos. b) Arena 2: simulação de uma residência comum com passagens entre cômodos equivalentes ao tamanho de portas padrão e móveis pequenos espalhados pelo caminho.



Fonte: Autor.

Na figura 26 indica-se, em ambas as arenas, os pontos de início das explorações representados pelas letras "A","B" e "C". O objetivo é observar o comportamento dos pacotes e configurações saindo de pontos de partida distintos. A figura 27 mostra a disposição dos obstáculos que estarão no caminho do robô durante a exploração na arena 2.

Figura 27 – Disposição do ambiente residencial (arena 2) visto pelo Gazebo com o robô no ponto inicial "A".



Fonte: Autor.

A primeira avaliação feita é em relação a possibilidade de usar os sensores ultrassônicos, o laser e as câmeras isoladamente no pacote *gmapping* para SLAM para a definição das configurações que são utilizadas nos testes. Sendo assim, a seção 5.1 traz o detalhamento da decisão que define o sensor laser para o SLAM como base para os testes realizados neste trabalho.

No planejamento foram previstas seis configurações diferentes para cada pacote da seguinte forma: as 3 primeiras tem o objetivo de calibrar os pacotes com a montagem predefinida dos sensores, variando-se os parâmetros relacionados aos pesos para os ganhos atrelados aos tamanhos de fronteiras em relação aos pesos das distâncias até as fronteiras; as outras três configurações são feitas variando-se a montagem de sensores para o SLAM, fixando-se a calibragem dos pacotes.

A calibragem referida nas 3 primeiras configurações é feita com a montagem mínima necessária de sensores para o robô conseguir realizar a exploração completamente, como mostra a figura 28a). Importante ressaltar que para navegar com segurança em um ambiente com mesas e obstáculos translúcidos, no caso da montagem do robô HERA, pelo menos uma das câmeras RGB-D disponíveis (Kinect ou Xtion) precisa estar ativa para que não hajam colisões no caso

das mesas, e os sensores ultrassônicos precisam estar ativos no caso de haverem obstáculos translúcidos.

A combinação dos sensores é apresentada na figura 28. Uma abordagem parecida foi utilizada no trabalho de Sun et al. (2017), no qual os sensores a laser e o Kinect foram utilizados em conjunto e depois separados para obtenção do resultado mais satisfatório. No presente trabalho os sensores variam conforme as possibilidades apresentadas pelo robô HERA na configuração atual. Apesar de não estarem destacados na figura 28, os sonares são utilizados para navegação em todas as configurações. As alturas dos sensores foram definidas respeitando as dimensões do robô, a altura máxima dos obstáculos e janelas das arenas. Portanto, as diferentes montagens dos sensores estão definidas da seguinte forma:

a) Montagem 1: a figura 28.a) apresenta disposição mais básica, utilizada para calibragem dos pacotes de exploração com o laser frontal para SLAM e navegação e a câmera Xtion para navegação;
b) Montagem 2: a figura 28.b) apresenta a disposição com os lasers frontal e traseiro em conjunto para o SLAM e a câmera Xtion em conjunto com o laser frontal para navegação;
c) Montagem 3: a figura 28.c): apresenta a disposição com o laser frontal e a câmera Kinect apontada para trás para o SLAM e o laser frontal em conjunto com a câmera Xtion para a navegação;
d) Montagem 4: a figura 28.d) apresenta a disposição com a câmera Xtion para o SLAM e navegação e o laser traseiro para SLAM.

O planejamento inicial para cada pacote em cada arena foi feito conforme apresentado na tabela 3. A tabela mostra o total de 90 testes, entretanto, como têm-se disponíveis 3 pacotes e 2 arenas diferentes, o total de testes planejados a princípio foi de 540 testes. O "parâmetro fixo" apresentado na tabela 3 refere-se à configuração dos parâmetros dos pacotes de exploração com os melhores resultados relacionados ao tempo, distância e rotação realizadas. Cada pacote de exploração possui seu "parâmetro fixo", conforme está apresentado na subseção 5.2.1.

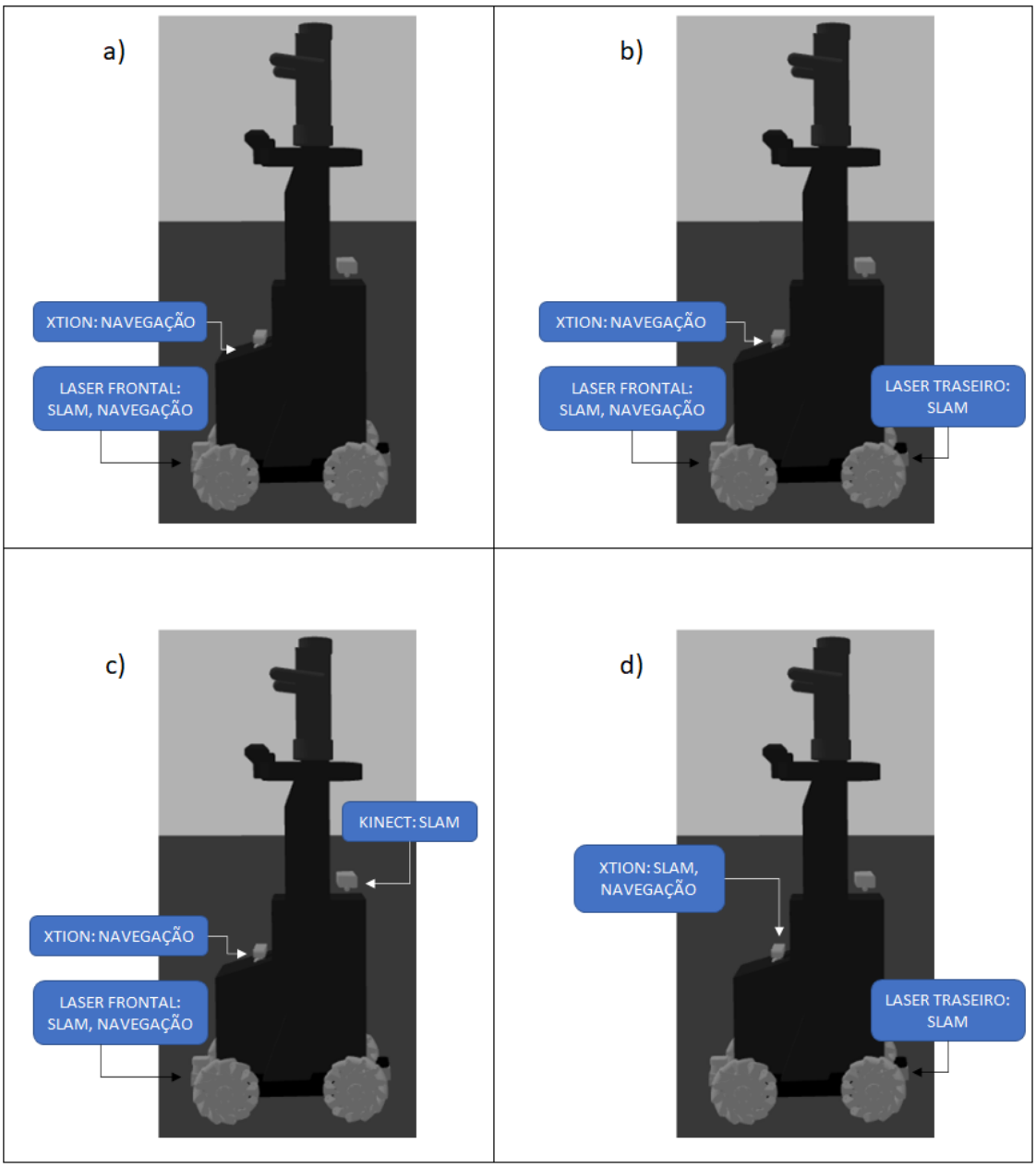
Com os testes realizados, a avaliação das configurações dos pacotes e dos sensores é feita com base nos seguintes critérios: tempo de exploração, distância percorrida e rotação realizada. É considerado uma falha quando o robô não consegue explorar o ambiente completamente com determinada configuração. Na seção 5.2 os resultados e discussões a respeito dos desempenhos são apresentados com maiores detalhes.

Tabela 3 – Quantidade de testes planejados para cada configuração diferente, variando-se tanto os parâmetros dos pacotes de exploração quanto a disposição dos sensores. A quantidade final é por pacote de exploração por arena.

Nº DA CONFIG.	CONFIGURAÇÕES	TESTES POR PONTOS DE PARTIDA			
		A	B	C	TOTAL
1	Exploração: parâmetros 1 Sensores: montagem 1	5	5	5	15
2	Exploração: parâmetros 2 Sensores: montagem 1	5	5	5	15
3	Exploração: parâmetros 3 Sensores: montagem 1	5	5	5	15
4	Exploração: parâmetros fixos Sensores: montagem 2	5	5	5	15
5	Exploração: parâmetros fixos Sensores: montagem 3	5	5	5	15
6	Exploração: parâmetros fixos Sensores: montagem 4	5	5	5	15
TOTAL DE TESTES POR PACOTE POR ARENA:					90

Fonte: Autor.

Figura 28 – Configurações dos sensores de distância nos testes realizados. Os sonares estão inclusos em todas as configurações para a navegação. a) Montagem mais básica, com o laser frontal para SLAM e navegação e a câmera Xtion para navegação. b) Montagem com os lasers frontal e traseiro em conjunto para SLAM e a câmera Xtion em conjunto com o laser frontal para navegação. c) Montagem com o laser frontal e a câmera Kinect apontada para trás para o SLAM e o laser frontal em conjunto com a câmera Xtion para a navegação. d) Montagem com a câmera Xtion para o SLAM e navegação e o laser traseiro para SLAM.



Fonte: Autor.

5 EXPERIMENTOS

Este capítulo traz a avaliação da metodologia proposta e a discussão dos resultados obtidos dos pacotes de exploração e das combinações de sensores. O objetivo principal do capítulo é expor os resultados e indicar a melhor estratégia de exploração visando reduzir a necessidade de deslocamento e rotação do robô, tempo de exploração e garantir a exploração completa do ambiente.

Esses critérios de eficiência foram escolhidos, pois estão conectados aos desafios de realizar atividades em ambientes domésticos ou de serviços. A questão da exploração no menor tempo está ligada diretamente à segurança e à busca incessante do mercado por soluções que otimizam o tempo, pois o tempo está diretamente ligado ao custo, seja energético ou financeiro. Quando um robô não conhece o ambiente, a referência que ele tem para se localizar são o mapa que ele está criando e sua odometria. Apesar de não serem avaliados neste trabalho os efeitos dos erros acumulativos na aplicação do SLAM, sabe-se que quanto maior for o deslocamento executado pelo robô até terminar a exploração maior será o erro acumulado da odometria no processo, principalmente durante o movimento de rotação, no qual o escorregamento se torna mais acentuado. Ou seja, ao minimizar a necessidade de navegação durante o SLAM, possibilita-se melhorar a precisão do mapa quando aplicado em situações reais. Sendo assim, apenas com um mapa completo e confiável é possível atribuir comandos e planejamentos mais efetivos para serviços diversos.

O software escolhido para realização das simulações foi o Gazebo que possui compatibilidade com o ROS. Todas as configurações e testes foram feitos no *framework* ROS e foi criado o pacote *HERA-exploration* com todos os parâmetros e ambientes selecionados para testes e replicação. As simulações foram feitas em um computador 64-bits, sistema operacional Ubuntu 18.04, processador Intel Core i5 e 8 GB de memória RAM.

Na simulação o robô apresenta um deslocamento planar, pode andar em linha reta ou girar em torno do próprio eixo. Como descrito nas seções anteriores, a navegação é feita através do pacote *move-base*. Os três pacotes de exploração testados são: *explore-lite*, *RRT-exploration* e *cam-exploration*.

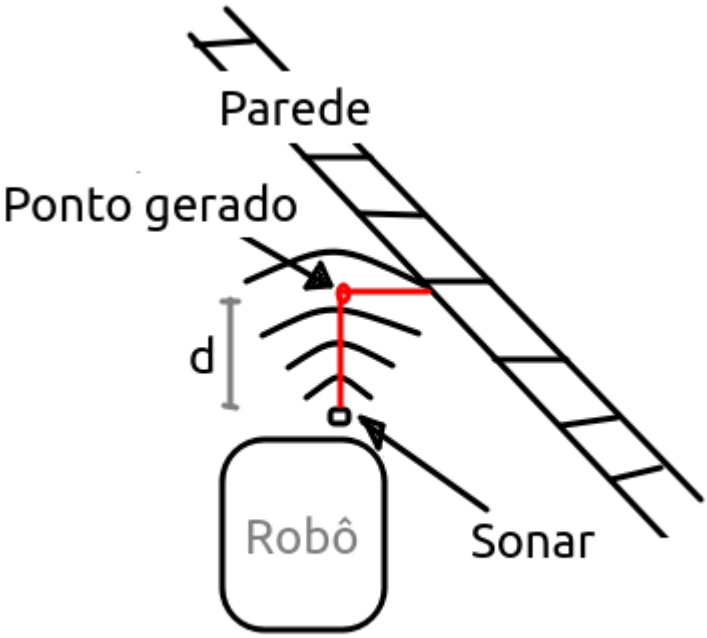
Neste capítulo, primeiro avalia-se os sensores para aplicação do SLAM na seção 5.1. Em seguida, a seção 5.2 explicita todos os resultados experimentais, tanto na etapa de calibragem dos pacotes de exploração (subseção 5.2.1), quanto na etapa de avaliação de sensores (subseção 5.2.2), para avaliação dos testes com os pacotes utilizando diferentes montagens de sensores. Em seguida, na subseção 5.2.3, discute-se a respeito do desempenho geral dos pacotes nas condições propostas. Por fim, na subseção 5.2.4, discute-se as vantagens de se fazer um mapeamento prévio do ambiente antes de se realizar alguma tarefa que envolva o deslocamento entre dois pontos no mapa.

5.1 SENSORES PARA O SLAM

Esta seção traz uma discussão a respeito da utilização dos sensores de distância para realização do SLAM durante a exploração de um ambiente desconhecido. Todos os testes realizados visam encontrar a maneira mais rápida de realizar uma exploração, buscando minimizar tanto os movimentos rotacionais quanto os translacionais, no entanto, o ambiente precisa ser representado de forma a ser possível utilizar o mapa posteriormente para se localizar e navegar em atividades futuras. Por isso, antes de escolher as combinações dos sensores para o SLAM e exploração, foi necessário fazer uma análise qualitativa para validar o uso dos sensores de distância individualmente.

O primeiro sensor avaliado no contexto da exploração, foi o sonar. Ao espalhar sonares ao redor do robô, a técnica *explore-lite* foi aplicada para observar a capacidade de completar a tarefa de maneira eficiente. Para sua aplicação no *gmapping*, como apresentado na seção 4.3, primeiro é necessário que seja feita a transformação da mensagem proveniente do sensor *Range* para *LaserScan*. O primeiro problema é que cada sonar representa um ponto de laser, enquanto que o laser utilizado possui 150 pontos distribuídos com 166 graus de alcance, conferindo uma abrangência muito maior durante a varredura, ou seja, para ter o mesmo desempenho utilizando este pacote, seria necessário haver mais sensores distribuídos ao redor do robô.

Figura 29 – Representação do robô utilizando o sonar para identificar a distância até o obstáculo no ambiente.



Fonte: Autor.

Outro problema identificado é a forma como a medida de distância obtida pelo sensor é interpretada pelo programa. Como apresentado na seção 2.1 o sonar emite um pulso de som

para o ambiente e calcula o tempo que demorar para retornar, no entanto, o cone de emissão do som irá retornar a distância no ponto que for atingido primeiro. A figura 29 mostra o comportamento do sensor. Pela imagem, é possível observar que a distância real entre o robô e a parede está sofrendo uma distorção. O "Ponto Gerado" está a uma distância d da posição do sensor, enquanto o dado correto seria se o ponto gerado estivesse encostado ou tão próximo quanto possível da parede neste exemplo. Entretanto, quanto maior for a distância entre o robô e o obstáculo, maior será a projeção do cone do sonar, que pode acarretar uma distorção cada vez maior. Portanto, para a aplicação de um mapeamento preciso, usar apenas o sonar, se mostrou uma estratégia ruim nas condições apresentadas, e por isso não é utilizado dessa forma no projeto. A alternativa para a o uso deste sensor é na navegação como entrada do *costmap* junto ao pacote *move-base* para planejamento local de trajetória. Todavia, é importante frisar que os sonares podem ser utilizados para o SLAM, sendo assim outras técnicas específicas devem ser utilizadas ao invés do pacote *gmapping*.

Os sensores Kinect e Xtion também foram avaliados com finalidade principal para realização do SLAM com o pacote *gmapping*. Da mesma forma que com os sonares, foi necessária a conversão do *PointCloud2* proveniente dos sensores de profundidade das câmeras em mensagens *laserScan*. Contudo, é possível observar suas limitações apenas analisando as características básicas desses sensores, que estão apresentadas na tabela 1. O alcance do Kinect é de 3,5m e seu ângulo de varredura horizontal é de apenas 57 graus, portanto, durante a exploração, é necessário que sejam feitos mais movimentos de rotação do robô, além de haver a limitação de ter que estar relativamente mais perto de um obstáculo para detectá-lo em comparação com o uso de lasers. Logo, como o desempenho durante o mapeamento exigiu muito mais movimentos de rotação, e consequentemente mais tempo e distorções no mapa criado com o SLAM, seu uso isoladamente foi descartado. Contudo, seu uso para o SLAM em conjunto com os lasers se mostrou útil para detectar os limites do mapa na altura do torso do robô ou completando o alcance na sua região traseira. O *PointCloud2* das câmeras se mostraram úteis para a navegação e detecção de móveis com estruturas esbeltas, como as mesas, que possuem pés que são apenas representados como pontos pelos lasers.

O único sensor que pôde ser utilizado isoladamente na atividade do SLAM para a exploração foi o sensor a laser, portanto os testes realizados nas seções posteriores deste capítulo mostram as vantagens e desvantagens de utilizar os lasers frontal e traseiro, e suas combinações com os sensores Kinect e Xtion em diferentes posições durante as explorações dos dois ambientes propostos.

5.2 RESULTADOS EXPERIMENTAIS

Conforme apresentado na seção 4.5, para avaliação dos pacotes e das configurações de sensores, são feitos testes em 2 arenas diferentes, em cada arena o robô sai de 3 pontos diferentes, representados pelos pontos "A", "B" e "C". Para cada arena são variadas 3 configurações

diferentes dos parâmetros dos pacotes utilizando a montagem mais básica (montagem 1: com o sensor frontal para o SLAM). Após a seleção dos parâmetros dos pacotes de exploração que possuem os melhores resultados, os parâmetros são fixados e então variam-se as disposições dos sensores. A figura 28 apresenta o resumo da variação da combinação dos sensores.

A subseção 5.2.1 apresenta a análise feita para a calibragem dos pacotes de exploração disponíveis e seus respectivos desempenhos para aplicação da variação de sensores. A subseção 5.2.2 avalia os desempenhos das montagens dos sensores. A subseção 5.2.3 traz um resumo dos resultados obtidos em todas as condições de exploração propostas. Por fim, na subseção 5.2.4 discute-se as vantagens de realizar a exploração do ambiente antes de realizar tarefas específicas.

5.2.1 Etapa de Calibragem

A montagem dos sensores para todos os testes durante a calibragem é mostrada na figura 28.a), com o sensor a laser frontal, a câmera Xtion e os sonares enviando dados para o pacote *move-base* para a navegação e o sensor a laser frontal enviando dados para o pacote *gmapping* para o SLAM.

Os testes foram realizados nas arenas apresentadas na figura 26. O primeiro pacote avaliado é o *explore-lite*. Os parâmetros do pacote de exploração estão apresentados na tabela 4. Os parâmetros *PotentialScale* e *GainScale* referem-se aos pesos dados à distância mínima e ao tamanho da fronteira, respectivamente, conforme apresentado na subseção 4.4.1. Os valores escolhidos variam da seguinte forma: na configuração 1 a importância das distâncias mínimas em relação aos tamanhos das fronteiras é igual a 3 e o peso da distância mínima é igual a 1; na configuração 2 os pesos da distância mínima e do tamanho da fronteira são iguais a 1; e na configuração 3 atribui-se o valor 3 para o peso da distância mínima e o 1 para o peso do tamanho da fronteira.

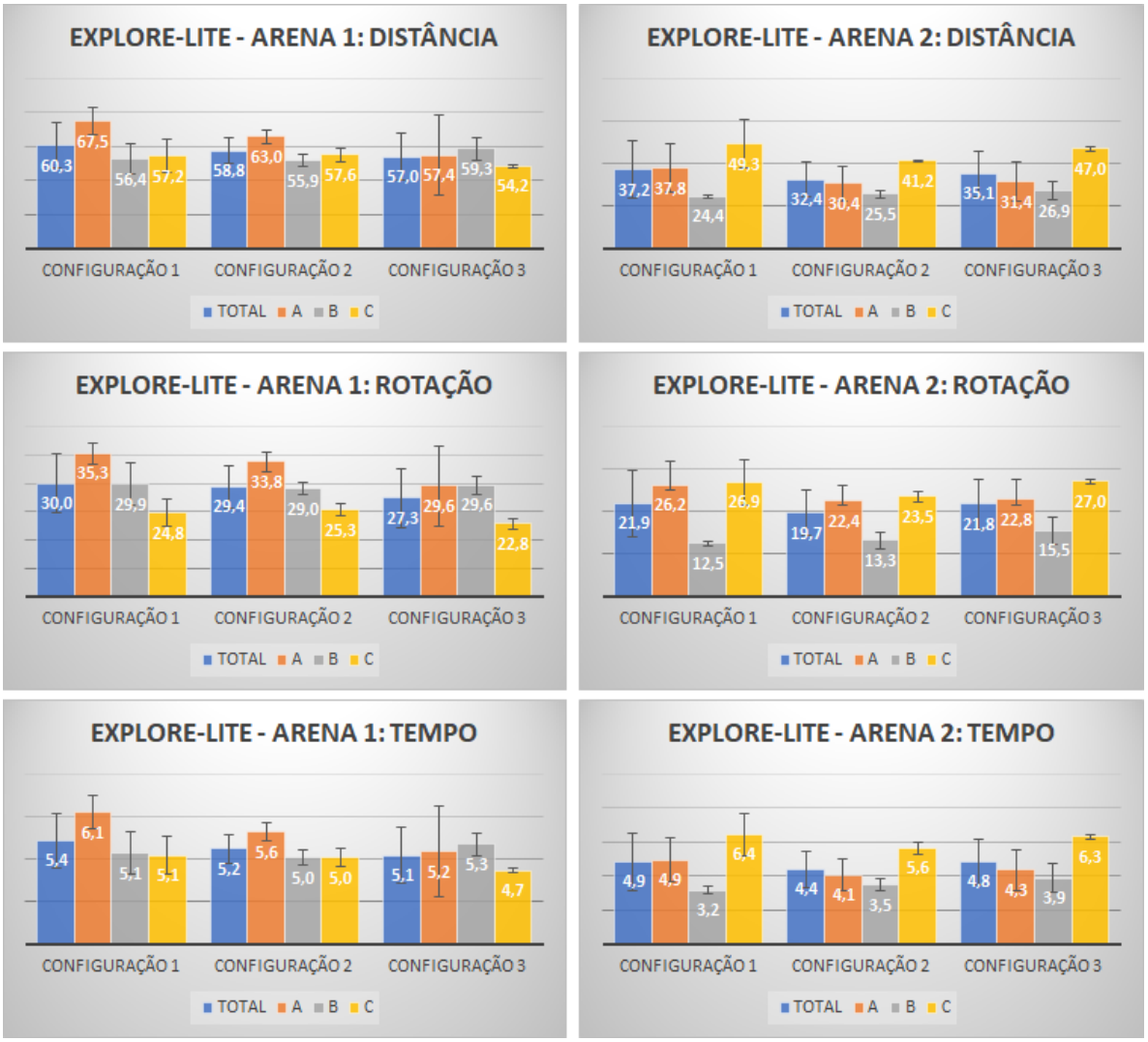
Tabela 4 – *explore-lite*: Variação da configuração dos parâmetros do pacote de exploração.

Nº da config.	Configurações	*PotentialScale*	*GainScale*
1	*explore-lite* : parâmetros 1	1/3	1
2	*explore-lite* : parâmetros 2	1	1
3	*explore-lite* : parâmetros 3	3	1

Fonte: Autor.

Nesta etapa de calibragem foram feitos 94 testes para o pacote *explore-lite*, sendo 4 testes interrompidos por falha durante a exploração e os 90 restantes divididos conforme a tabela 3. São 15 testes para cada configuração para cada uma das 2 arenas disponíveis. Os resultados com a média total de exploração e com as médias saindo de cada posição inicial estão apresentados na figura 30 com os respectivos desvios padrão. A distância está em metros, a rotação em radianos e o tempo em minutos.

Figura 30 – Gráficos com os resultados da aplicação do pacote *explore-lite* nas 3 configurações propostas. Com a distância em metros, a rotação em radianos e o tempo em minutos.



Fonte: Autor.

Os resultados mostram que a configuração 3 possui os melhores desempenhos em todos os critérios na arena 1 e a configuração 2 possui os melhores desempenhos na arena 2. A configuração dos parâmetros do pacote *explore-lite* escolhida para se tornar o "parâmetro fixo" durante a variação das montagens de sensores é a configuração 2. Essa escolha foi definida pelos seguintes critérios: a arena 2 possui maior similaridade com ambientes reais e, por isso possui um peso maior; os pontos de partida mostram que a configuração 2 supera o desempenho da configuração 3 saindo da posição "B" na arena 1 e em qualquer uma das 3 posições na arena 2, enquanto que a configuração só supera o desempenho da configuração 2 saindo de 2 posições diferentes na arena 1 e em nenhuma posição na arena 2; além destes pontos citados, os desvios padrão mostram que a configuração 2 no geral possui o comportamento mais estável entre todas as configurações.

Tabela 5 – *RRT-exploration*: variação da configuração dos parâmetros do pacote de exploração.

Nº da config.	Configurações	Multiplicador de Informação
1	*RRT-exploration* : parâmetros 1	3
2	*RRT-exploration* : parâmetros 2	1
3	*RRT-exploration* : parâmetros 3	1/3

Fonte: Autor.

O segundo pacote calibrado foi o *RRT-exploration*. Neste caso, variou-se o peso dado ao multiplicador de informação, conforme mostra a tabela 5. No pacote o nome dado ao parâmetro é *info-multiplier*. Neste caso só se utilizou este parâmetro, pois não há explicitamente para configuração deste pacote o peso da distância mínima, conforme apresentado na subseção 4.4.2. Sendo assim, tem-se a configuração 1 com o valor do multiplicador de informação igual a 3, depois, a configuração 2 com o multiplicador igual a 1 e, por fim, a configuração 3, com o multiplicador dividindo por 3.

Nesta etapa, foram feitos 98 testes com o pacote *RRT-exploration*, 8 testes foram interrompidos por falha de execução e outros 90 foram distribuídos seguindo a mesma lógica utilizada com o pacote *explore-lite*. Contempla-se os 3 pontos de partida para cada uma das 2 arenas para 3 configurações de parâmetros diferentes. A figura 30 mostra que a configuração 1 obteve os melhores resultados em todos os critérios analisados. Portanto, a configuração dos parâmetros do pacote *RRT-exploration* escolhida para se tornar o "parâmetro fixo"durante a variação das montagens dos sensores é a configuração 1.

O pacote *cam-exploration* foi o último calibrado. Os parâmetros disponíveis para variação, conforme apresentado na subseção 4.4.3, são: o peso da distância euclidiana e o peso da distância calculada pelo método A*. Dessa forma, a tabela 6 mostra como foram distribuídos os parâmetros para cada uma das 3 configurações propostas. Primeiro utiliza-se o peso da distância euclidiana igual a 1 e o peso da distância pelo A* como zero, em seguida utiliza-se tanto a distância euclidiana quanto a distância A* com o peso igual a 1 para ambos e, por fim, atribui-se o valor 1 para o peso da distância A* e zero para o peso da distância euclidiana.

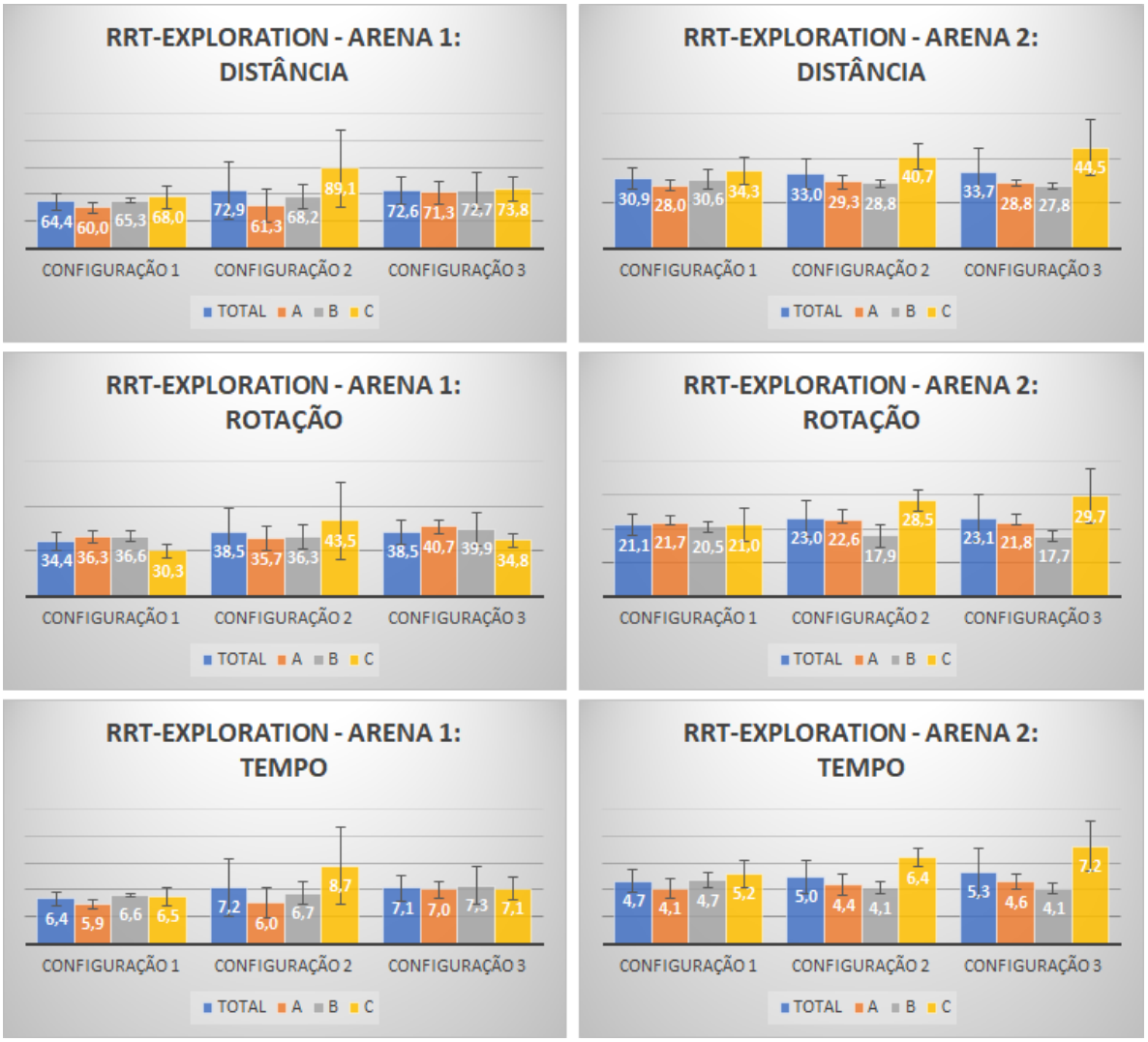
Tabela 6 – *cam-exploration*: Variação da configuração dos parâmetros do pacote de exploração.

Nº da config.	Configurações	Distância Euclidiana	Distância por A*
1	*cam-exploration* : parâmetros 1	1	0
2	*cam-exploration* : parâmetros 2	1	1
3	*cam-exploration* : parâmetros 3	0	1

Fonte: Autor.

Ao todo foram feitos 56 testes nesta etapa para o pacote *cam-exploration*. Sendo que, destes 56 testes, 11 foram interrompidos por conta de falhas na execução e os outros 45 foram distribuídos com 5 execuções por configuração de parâmetros para cada ponto de partida. Diferente dos outros dois pacotes, o pacote *cam-exploration* não foi capaz de completar nenhuma

Figura 31 – Gráficos com os resultados da aplicação do pacote *RRT-exploration* nas 3 configurações propostas. Com a distância em metros, a rotação em radianos e o tempo em minutos.



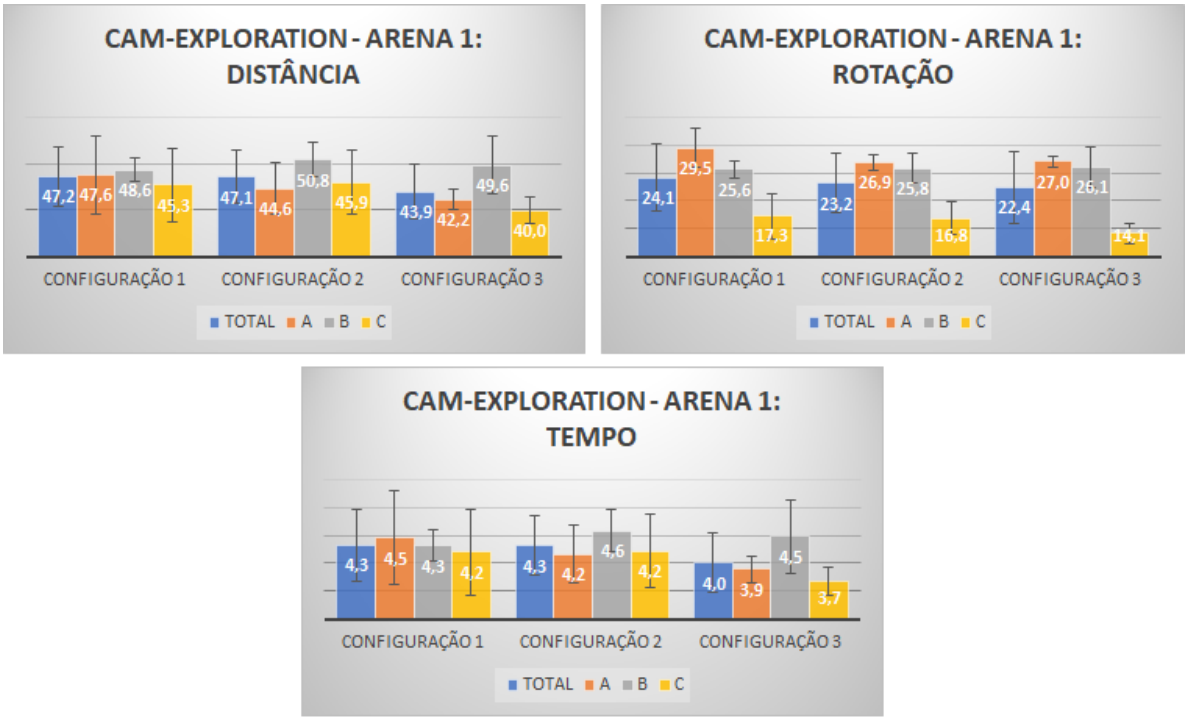
Fonte: Autor.

exploração na arena 2, pois este ambiente possui maior complexidade e obstáculos pelo caminho, além de possuir passagens relativamente mais estreitas do que na arena 1, que chegam próximas às dimensões do robô. A falha ocorreu pois, em algum momento durante a exploração, o algoritmo, que verifica se uma determinada fronteira é alcançável, não conseguia criar um caminho válido para enviar o objetivo para o *move-base* realizar o processo de navegação, fazendo com que o pacote parasse de funcionar.

A figura 32 mostra os resultados obtidos nos testes realizados na arena 1. A configuração 3 é a escolhida para ser o "parâmetro fixo" na variação das configurações dos sensores, pois obteve os melhores desempenhos em todos os critérios avaliados.

Apesar de admitir-se no contexto deste trabalho que o SLAM é um problema resolvido, a cobertura completa do ambiente foi importante para definir se a tarefa de exploração foi re-

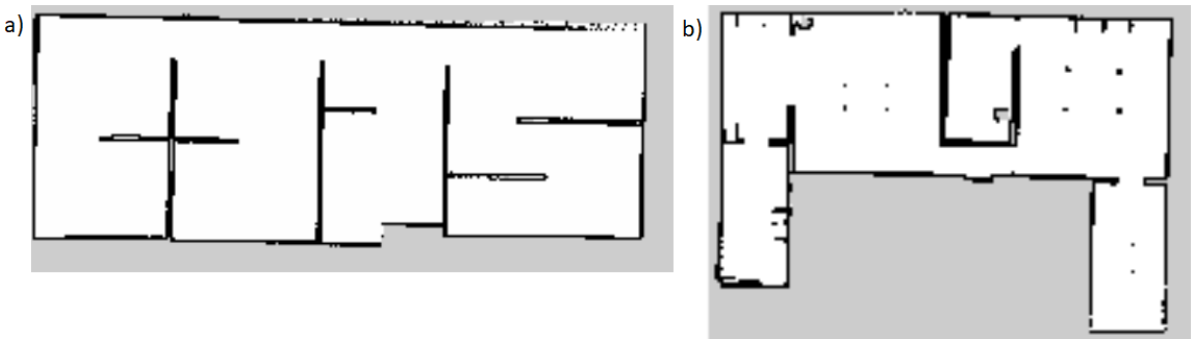
Figura 32 – Gráficos com os resultados da aplicação do pacote *cam-exploration* nas 3 configurações propostas. Com a distância em metros, a rotação em radianos e o tempo em minutos.



Fonte: Autor.

alizada por completo. A figura 33 mostra os mapas criados após a realização da exploração e ilustra exemplos de mapas que foram considerados como explorados completamente.

Figura 33 – Mapas criados com o pacote *gmapping* após a atuação dos pacotes de exploração: a) representação da arena 1 e b) representação da arena 2.



Fonte: Autor.

Com a calibragem dos parâmetros dos pacotes de exploração, foram definidos os "parâmetros fixos" para os testes das configurações dos sensores. A tabela 7 mostra o resumo das melhores configurações de cada pacote de exploração.

Tabela 7 – Resumo das melhores configurações dos parâmetros dos pacotes de exploração para cada pacote.

Pacote	Nº da config.	*Configuração*
explore-lite	2	PotentialScale = 1; GainScale = 1
RRT-exploration	1	Multiplicador de Informação = 3
cam-exploration	3	Dist Euclidiana = 0; Dist A* = 1

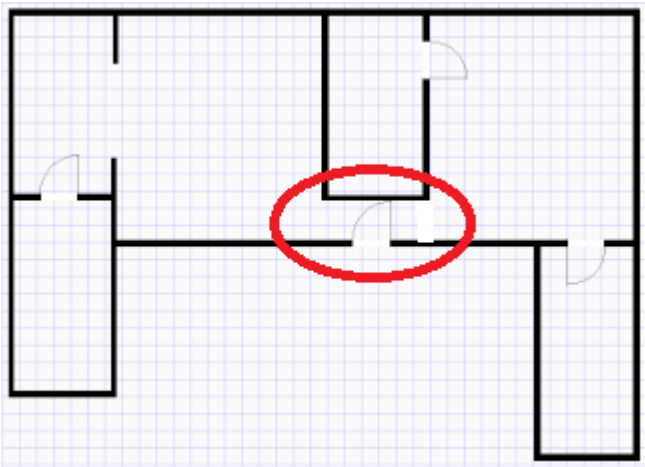
Fonte: Autor.

5.2.2 Etapa de Avaliação dos Sensores

Após calibrar os pacotes de exploração, definiram-se os "parâmetros fixos" que são utilizados nos testes com a variação da montagem dos sensores para cada pacote de exploração, conforme o resumo apresentado na tabela 7. Sendo assim, os testes foram feitos variando-se as montagens dos sensores conforme apresentado na figura 28.

A configuração 6 (montagem dos sensores 4) não pôde ser utilizada na arena 2 com nenhum pacote de exploração, pois não foi possível passar pelo ponto indicado na figura 34. Como o Xtion tem um alcance de apenas 3,5m de distância, o robô não adquire informação o suficiente para montar uma fronteira atingível na região indicada, pois o trecho é muito estreito e relativamente comprido. A diferença de quando utiliza-se o laser, é que seu maior alcance permite cobrir uma região inexplorada maior do mapa do próximo cômodo, possibilitando que o algoritmo de exploração encontre um caminho válido até a fronteira encontrada.

Figura 34 – Ponto indicando local na arena 2 que impossibilitou a exploração autônoma utilizando a configuração de sensores 4 com o Xtion frontal e o laser traseiro realizando o SLAM.



Fonte: Autor.

A primeira análise dos desempenhos foi feita com o pacote *explore-lite*. A tabela 8 mostra o resumo as configurações avaliadas e comparadas. A configuração dos parâmetros do *explore-lite* são: *PotentialScale* = 1 e *GainScale* = 1.

Tabela 8 – Resumo da configuração dos sensores com o *explore-lite* para os testes nas duas arenas.

Nº da config.	Configurações	SLAM	Navegação
2	Sensores: montagem 1	laser frontal	Xtion frontal; laser frontal
4	Sensores: montagem 2	laser frontal; laser traseiro	Xtion frontal; laser frontal
5	Sensores: montagem 3	laser frontal; Kinect traseiro	Xtion frontal; laser frontal
6	Sensores: montagem 4	Xtion frontal; laser traseiro	Xtion frontal

Fonte: Autor.

O total de testes realizados nesta etapa com o pacote *explore-lite* foram 81. Houveram 6 falhas durante a execução da exploração e os 75 restantes estão distribuídos da seguinte forma: na arena 1 todas as configurações de sensores obtiveram sucesso, logo tem-se 3 pontos de partida para 3 novas configurações com 5 testes para cada condição; e os outros 30 testes são feitos com as 2 montagens de sensores que possibilitaram a exploração saindo dos 3 pontos iniciais diferentes.

Os resultados estão apresentados na figura 30. No geral, para este pacote, o melhor desempenho foi obtido pela configuração de sensores 1, no entanto, os resultados obtidos pela configuração 5 (com a montagem de sensores 3) foram bem próximos aos da montagem de sensores 1, mostrando que também podem ser utilizados para este fim com um bom desempenho.

O mesmo procedimento foi feito com o pacote *RRT-exploration*, utilizou-se a melhor configuração dos parâmetros obtido pelos resultados apresentados na subseção 5.2.1. No caso, foi escolhido o parâmetro "multiplicador de informação" igual a 3. A tabela 9 mostra o resumo de como ficaram as configurações para os testes nesta etapa.

Tabela 9 – Resumo da configuração dos sensores com o *RRT-exploration* para os testes nas 2 arenas.

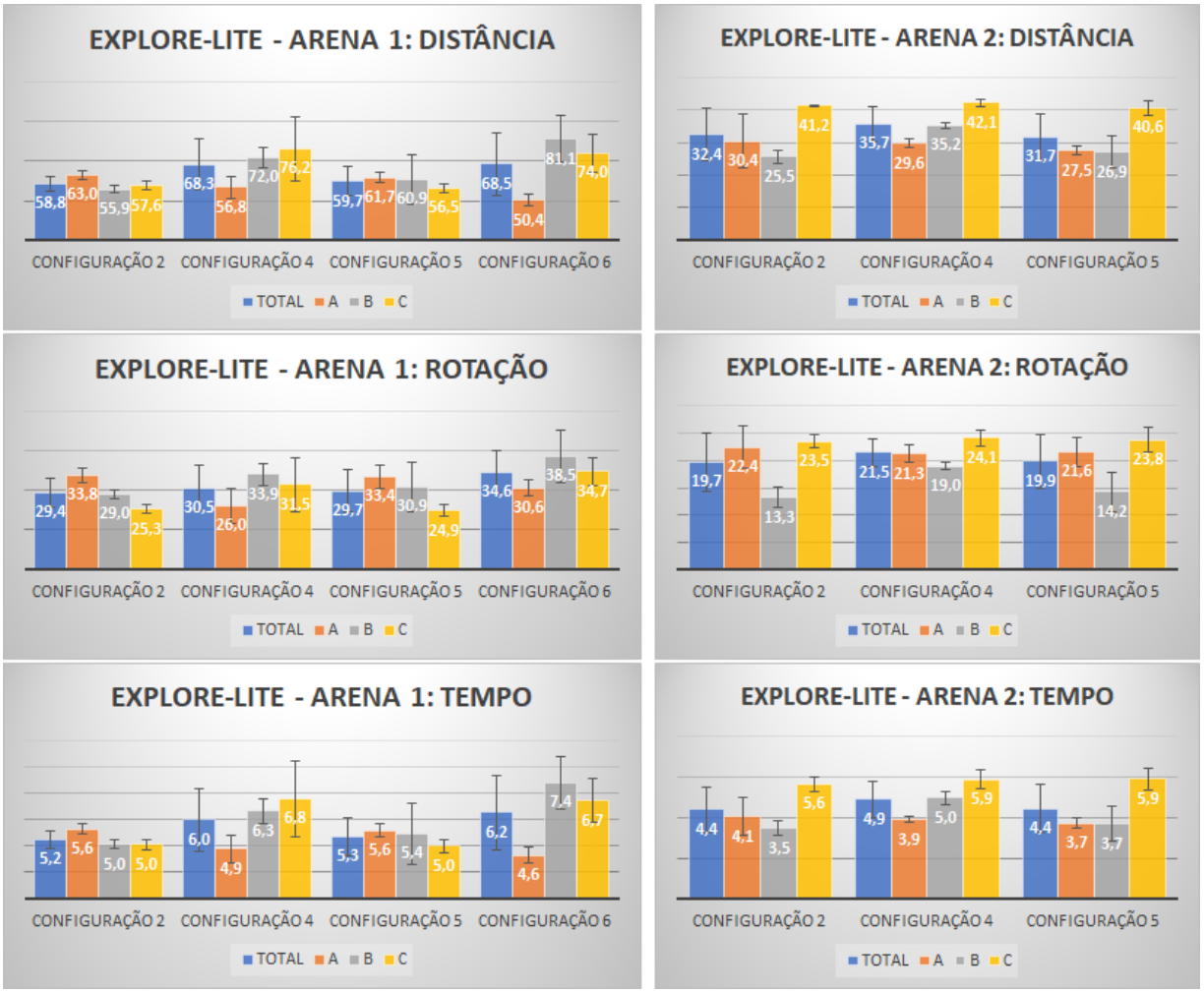
Nº da config.	Configurações	SLAM	Navegação
2	Sensores: montagem 1	laser frontal	Xtion frontal; laser frontal
4	Sensores: montagem 2	laser frontal; laser traseiro	Xtion frontal; laser frontal
5	Sensores: montagem 3	laser frontal; Kinect traseiro	Xtion frontal; laser frontal
6	Sensores: montagem 4	Xtion frontal; laser traseiro	Xtion frontal

Fonte: Autor.

Com o *RRT-exploration* foram realizados 75 testes nesta etapa. Não houve nenhuma falha na exploração completa do ambiente e os 75 testes estão distribuídos da mesma forma que foi feito com o *explore-lite*: na arena 1 todas as configurações de sensores obtiveram sucesso, logo tem-se 3 pontos de partida para 3 novas configurações com 5 testes para cada condição; e os outros 30 testes são feitos com as 2 configurações de sensores que possibilitaram a exploração saindo de 3 pontos iniciais diferentes.

A figura 36 mostra os resultados obtidos com as configurações de sensores apresentadas. Diferente do que ocorreu com o *explore-lite* as configurações de sensores obtiveram desempenhos diferentes dependendo da arena e dependendo do critério utilizado. Analisando o critério

Figura 35 – Gráficos com os resultados da aplicação do pacote *explore-lite* com 4 diferentes montagens de sensores na arena 1 e com 3 montagens de sensores na arena 2. Com a distância em metros, a rotação em radianos e o tempo em minutos.

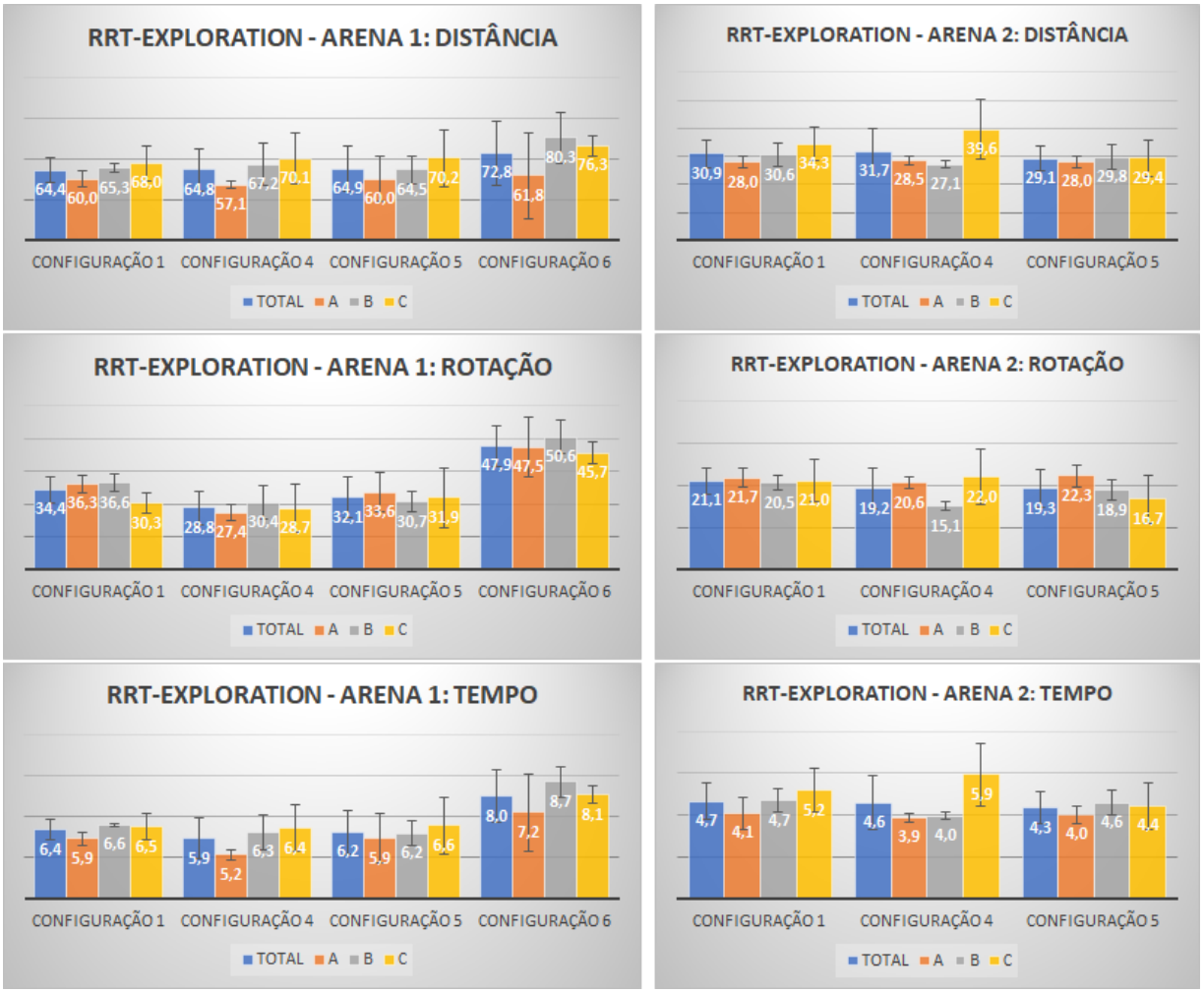


Fonte: Autor.

de distância percorrida, observa-se que a configuração 1 (montagem de sensores 2) foi melhor na arena 1 e intermediária na arena 2, enquanto que as configurações 4 (montagem de sensores 2) e 5 (montagem de sensores 3) alternaram os desempenhos nas arenas 1 e 2.

Quando o critério é a rotação realizada, há uma ampla vantagem da configuração 4 em relação as demais na arena 1 e uma pequena vantagem na arena 2 em relação à configuração 5. Isso ocorreu principalmente pelos ângulos de abertura horizontal serem os maiores entre todas as configurações propostas, ou seja, o robô não precisa ficar rotacionando para cobrir o mesmo espaço que com outras montagens de sensores seria necessário. A figura 37 ilustra o alcance dos lasers frontal e traseiro sendo utilizados em conjunto, com 166 graus na parte frontal e 166 graus na parte traseira. A configuração 5 na arena 2 obtém um resultado parecido com a configuração 4, pois os caminhos e os cômodos são mais estreitos que na arena 1, fazendo com que essa vantagem não se destacasse da mesma forma.

Figura 36 – Gráficos com os resultados da aplicação do pacote *RRT-exploration* com 4 diferentes montagens de sensores na arena 1 e com 3 montagens de sensores na arena 2. Com a distância em metros, a rotação em radianos e o tempo em minutos.



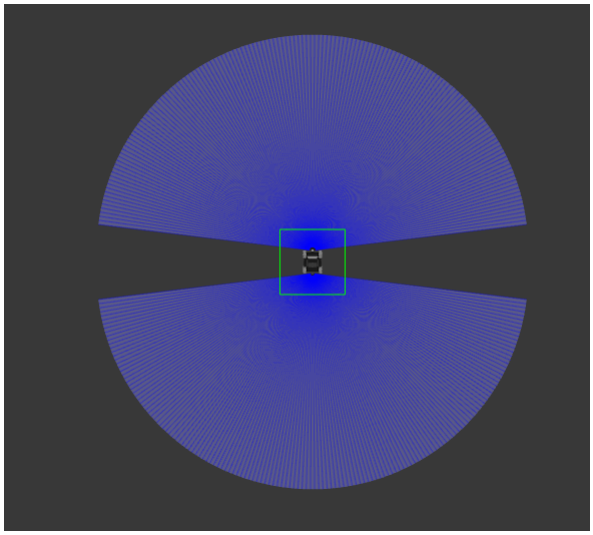
Fonte: Autor.

Em relação ao tempo de exploração, o comportamento é parecido com o resultado da rotação realizada. A configuração 4 é a melhor, seguida pela configuração 5 na arena 1 e melhor com a configuração 5 em relação às demais na arena 2. Os resultados mostram que a configuração 5 no geral foi melhor na arena 2 e a configuração 4 foi melhor na arena 1, no entanto, ambas podem ser utilizadas com boa performance dependendo da disponibilidade do robô. Apesar de não ser abordada a qualidade do mapa neste trabalho, sabe-se que a precisão do laser produz mapas mais precisos quando aplicado em ambientes reais, ou seja, se o critério fosse a qualidade do mapa criado, provavelmente a configuração 4 seria a escolhida para a aplicação.

A configuração 6 obteve os piores resultados em ambos os pacotes, e por isso foi descartada e não será utilizada no robô HERA.

A última etapa de variação de sensores, foi feita com o pacote *cam-exploration*. Para este pacote, o "parâmetro fixo" definido pelo melhor desempenho na etapa de calibragem é com

Figura 37 – Vista superior no Gazebo do robô HERA, com os sensores laser frontal e traseiro ativos, ilustrando o alcance do ângulo de abertura para aplicação no mapeamento.



Fonte: Autor.

o peso da distância euclidiana igual a zero e o peso da distância por A* igual a 1. Logo, a tabela 10 mostra o resumo das configurações que foram definidos para esta etapa.

Tabela 10 – Resumo da configuração dos sensores com o *cam-exploration* para os testes nas duas arenas.

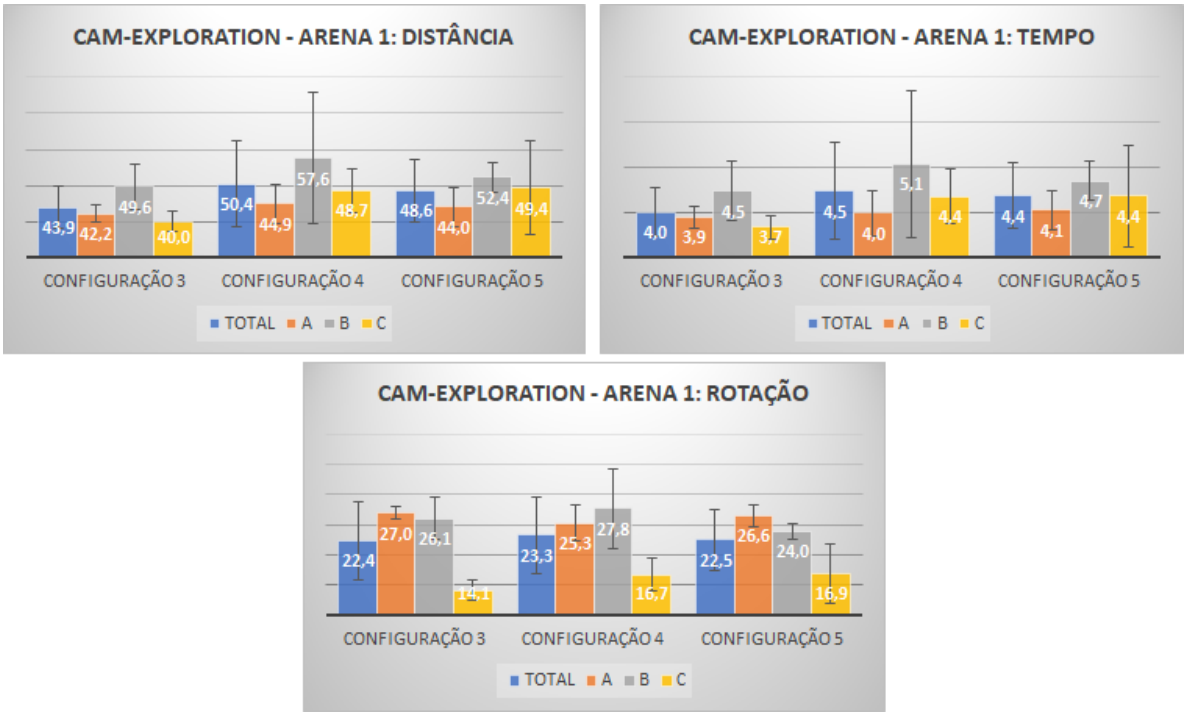
Nº da config.	Configurações	SLAM	Navegação
3	Sensores: montagem 1	laser frontal	Xtion frontal; laser frontal
4	Sensores: montagem 2	laser frontal; laser traseiro	Xtion frontal; laser frontal
5	Sensores: montagem 3	laser frontal; Kinect traseiro	Xtion frontal; laser frontal
6	Sensores: montagem 4	Xtion frontal; laser traseiro	Xtion frontal

Fonte: Autor.

Ao todo foram feitos 34 testes, sendo 4 testes interrompidos por falhas que impediram a exploração completa do ambiente. Os 30 restantes foram distribuídos nas configurações 4 e 5, saindo de cada uma das 3 posições iniciais da arena 1. Não foi possível completar nenhuma exploração com a configuração 6 em nenhuma das duas arenas e na arena 2 nenhuma configuração possibilitou a conclusão da exploração. A figura 38 apresenta os resultados obtidos.

Assim como no *explore-lite*, no geral a configuração de sensores que teve os melhores resultados foi a configuração inicial (montagem de sensores 1), apenas com o laser frontal atuando. Isso ocorre pois, com mais partes do mapa sendo descobertas simultaneamente, criam-se mais fronteiras para serem avaliadas, fazendo com que o robô faça escolhas que resultam na tomada de decisão que pode leva-lo a definir uma outra fronteira que atrapalha a sequência de terminar a exploração de determinado cômodo completamente. Ao ter essa interrupção, para completar o ambiente completamente, o robô acaba atravessando locais que já foram visitados em momentos anteriores, causando a ineficiência.

Figura 38 – Gráficos com os resultados da aplicação do pacote *cam-exploration* com 3 diferentes montagens de sensores na arena 1. Com a distância em metros, a rotação em radianos e o tempo em minutos.



Fonte: Autor.

Na subseção 5.2.3 é apresentado um resumo com as principais vantagens e desvantagens de cada pacote de exploração e da melhor montagem de sensores para exploração autônoma.

5.2.3 Resumo dos Resultados Obtidos

Os resultados das configurações são importantes para entender a melhor forma de atuar com o robô HERA em uma situação na qual seja necessário realizar uma exploração autônoma de um ambiente desconhecido. Inicialmente foi proposto um total de 540 simulações em todas as condições apresentadas na seção 4.5. No entanto, a tabela 11 mostra o rendimento de cada pacote em relação ao cumprimento dos objetivos.

Tabela 11 – Resumo das quantidades de testes planejados em relação aos testes realizados.

Quantidades	*explore-lite*	*RRT-exploration*	*cam-exploration*	Total
Testes planejados	180	180	180	540
Testes realizados	165	165	75	405
Arenas exploradas	2	2	1	5
Falhas na exploração	10	8	15	33
Falhas de configuração por arena	1	1	7	9

Fonte: Autor.

Os pacotes *explore-lite* e *RRT-exploration* tiveram um desempenho geral parecido, pois ambos conseguiram realizar 91,6% dos testes planejados (165 testes cada), conseguiram explorar as 2 arenas na maioria das configurações e só não conseguiram explorar a arena 2 com a configuração 6 (montagem de sensores 4). A quantidade de falhas nos testes foi menor no *RRT-exploration* em comparação às demais, mostrando que este pacote é o mais estável. Em relação ao desempenho geral, o pacote *cam-exploration* falhou em muitas situações. Apenas foi possível completar 41,6% dos testes planejados. Além disso, foi o pacote que mais teve falhas na execução, com o total de 15. Entretanto, o fato de não conseguir completar a exploração da arena 2 em nenhuma das configurações propostas, o pacote não é indicado para aplicações em condições reais.

Tabela 12 – Resultados das distâncias percorridas por pacote de exploração para as 2 arenas disponíveis em metros

ARENA 1							
Nº da config.	Montagem de sensores	*explore-lite*		*RRT-exploration*		*cam-exploration*	
		Média	σ	Média	σ	Média	σ
1	1	60,35	6,72	64,44	6,16	47,19	6,49
2	1	58,81	3,65	72,87	21,21	47,11	5,79
3	1	56,96	6,92	72,63	10,02	43,94	5,77
4	2	68,31	12,87	64,77	10,33	50,37	11,94
5	3	59,71	7,35	64,89	11,07	48,58	8,52
6	4	68,50	15,93	72,82	15,74	-	-
Melhor resultado =		56,96	**6,92**	64,44	**6,16**	43,94	**5,77**

ARENA 2							
Nº da config.	Montagem de sensores	*explore-lite*		*RRT-exploration*		*cam-exploration*	
		Média	σ	Média	σ	Média	σ
1	1	37,16	13,56	30,94	4,97	-	-
2	1	32,41	8,10	32,95	6,68	-	-
3	1	35,12	10,41	33,71	10,45	-	-
4	2	35,65	5,40	31,74	8,26	-	-
5	3	31,67	7,21	29,06	4,47	-	-
6	4	-	-	-	-	-	-
Melhor resultado =		31,67	**7,21**	29,06	**4,47**	-	-

Fonte: Autor.

A comparação entre os critérios propostos a respeito dos desempenhos dos pacotes e das configurações mostram as estratégias que seria mais adequadas para a aplicação em ambientes residenciais. Os desempenhos de cada configuração para cada pacote estão apresentados nas tabelas 12, 13 e 14. Nas tabelas têm-se os resultados médios de cada configuração do robô para cada pacote de exploração para cada critério de desempenho, sendo o desvio padrão de cada resultado apresentado na coluna com símbolo σ. Os melhores resultados para cada pacote estão destacados em vermelho nas tabelas. O primeiro ponto a ser observado é que o pacote *cam-exploration* possui os melhores resultados na arena 1 para todos os critérios. Sua forma de calcular a próxima fronteira é a mais eficiente entre as técnicas de exploração analisadas. No entanto, além de ser o mais instável, como mostrado na tabela 11, não é possível completar nenhuma exploração em um ambiente com maior complexidade (no caso, na arena 2). As

Tabela 13 – Resultados das rotações realizadas por pacote de exploração para as 2 arenas disponíveis em radianos

ARENA 1							
Nº da config.	Montagem de sensores	*explore-lite*		*RRT-exploration*		*cam-exploration*	
		Média	σ	Média	σ	Média	σ
1	1	29,99	5,17	34,39	3,94	24,13	6,07
2	1	29,35	3,81	38,46	10,62	23,18	5,21
3	1	27,34	5,12	38,49	5,36	22,38	6,46
4	2	30,48	5,66	28,84	5,12	23,25	6,43
5	3	29,75	5,31	32,09	6,14	22,48	5,12
6	4	34,60	5,31	47,90	6,25	-	-
Melhor resultado =		27,34	**5,12**	28,84	**5,12**	22,38	**6,46**

ARENA 2							
Nº da config.	Montagem de sensores	*explore-lite*		*RRT-exploration*		*cam-exploration*	
		Média	σ	Média	σ	Média	σ
1	1	21,86	8,00	21,10	3,11	-	-
2	1	19,71	5,32	23,01	5,36	-	-
3	1	21,77	5,85	23,08	6,88	-	-
4	2	21,46	2,48	19,22	4,81	-	-
5	3	19,87	5,03	19,28	4,36	-	-
6	4	-	-	-	-	-	-
Melhor resultado =		19,71	**5,32**	19,22	**4,81**	-	-

Fonte: Autor.

Tabela 14 – Resultados dos tempos de execução por pacote de exploração para as 2 arenas disponíveis em minutos

ARENA 1							
Nº da config.	Montagem de sensores	*explore-lite*		*RRT-exploration*		*cam-exploration*	
		Média	σ	Média	σ	Média	σ
1	1	5,44	0,64	6,35	0,50	4,33	0,64
2	1	5,24	0,34	7,16	2,08	4,32	0,53
3	1	5,09	0,65	7,13	0,94	4,01	0,53
4	2	5,98	1,19	5,95	0,97	4,48	1,07
5	3	5,34	0,70	6,24	1,08	4,38	0,74
6	4	6,25	1,41	8,00	1,29	-	-
Melhor resultado =		5,09	**0,65**	5,95	**0,97**	4,01	**0,53**

ARENA 2							
Nº da config.	Montagem de sensores	*explore-lite*		*RRT-exploration*		*cam-exploration*	
		Média	σ	Média	σ	Média	σ
1	1	4,85	1,69	4,65	0,88	-	-
2	1	4,39	1,10	4,96	1,22	-	-
3	1	4,84	1,34	5,29	1,76	-	-
4	2	4,91	0,89	4,60	1,28	-	-
5	3	4,44	1,20	4,35	0,75	-	-
6	4	-	-	-	-	-	-
Melhor resultado =		4,39	**1,10**	4,35	**0,75**	-	-

Fonte: Autor.

limitações na escolha de uma fronteira atingível fazem com que o pacote seja descartado na aplicação do robô HERA em ambientes residenciais.

Em todos os critérios, o *explore-lite* e o *RRT-exploration* tiveram resultados parecidos. Na arena 1, o melhor desempenho foi obtido com o pacote *explore-lite* e na arena 2 o melhor resultado foi obtido com o pacote *RRT-exploraion*. Os testes mostram que ambos podem ser utilizados com boa performance para a aplicação proposta. Entre estes dois pacotes, há uma característica que pode ser considerada uma vantagem ou uma desvantagem, que é o início da exploração. O pacote *RRT-exploration* exige que uma área seja delimitada no *rviz* para o robô começar a exploração, enquanto o *explore-lite* vai explorando enquanto houver fronteiras disponíveis para serem acessadas e não é necessária nenhuma interação inicial. Delimitar a área pode ser uma vantagem quando trata-se de uma área aberta, com acessos ao exterior de edifícios, no entanto, precisa da interação humana a cada nova porção do ambiente a ser selecionada. No caso dos testes feitos, os ambientes eram fechados, então selecionou-se uma larga região do ambiente para que não fosse preciso novas interações até completar a exploração dos 2 ambientes propostos. No caso do *explore-lite*, seria necessário que o ambiente de exploração fosse isolado fisicamente (como por exemplo com o fechamento das portas e acessos a locais indesejados ou inseguros), contudo além de iniciar a simulação, nenhuma outra interação é necessária para completar a exploração total do ambiente.

Em relação à montagem dos sensores, as tabelas 12, 13 e 14 mostram que os resultados variam muito com o pacote de exploração, com sua respectiva calibragem e com a diferença do ambiente. As montagens dos sensores são feitas conforme apresentado na figura 28 e são associados às configurações da seguinte forma: as configurações 1, 2 e 3 são feitas com a montagem de sensores 1; a configuração 4 é com a montagem de sensores 2; a configuração 5 é com a montagem de sensores 3; e a configuração 6 é com a montagem de sensores 4.

No geral, a montagem de sensores 1 obteve os melhores resultados com os pacotes *explore-lite* e *cam-exploration*, e as montagens 2 e 3 obtiveram os melhores resultados no geral com o pacote *RRT-exploration*. Essa diferença é explicada pela maneira que os pacotes encontram as fronteiras que serão calculadas. Os pacotes *explore-lite* e *cam-exploration* encontram as fronteiras disponíveis assim que o *gmapping* atualiza o mapa, avaliando o MGO gerado e identificando as células conhecidas que possuem vizinhança com células desconhecidas. Isso faz com que, no momento em que a região conhecida do mapa se expanda, as fronteiras já apareçam como objetivos em potencial, fazendo com que o robô mude o objetivo com mais frequência. Este efeito não ocorre com o *RRT-exploration*, pois como apresentado na seção 4.4.2, as fronteiras apenas são encontradas através da técnica do RRT. Por mais que a região do mapa se expanda rapidamente, o algoritmo que indica os pontos a serem avaliados, demora tempo o suficiente para que o robô consiga seguir seu curso normal de exploração, minimizando as chances de ter que passar muitas vezes pelo mesmo local já explorado para completar os pedaços que ficaram faltando de determinado cômodo.

Entre as montagens de sensores 2 e 3, não há grandes diferenças no comparativo dos desempenhos observados, contudo, a questão, que sairia do contexto do trabalho, seria em relação à qualidade do mapa criado durante o SLAM, pois os lasers em geral possuem uma precisão maior. Contudo, o custo de sensores a laser dessa natureza são muito maiores do que o custo de câmeras de profundidade, logo, se não há a disponibilidade de dois sensores a laser opostos, utilizar a câmera traz resultados satisfatórios, sem perdas de desempenho para a aplicação proposta.

Com todos estes dados analisados é possível definir que o pacote *cam-exploration* e a montagem de sensores 4, que possui o Xtion com o sensor a laser traseiro executando o SLAM e a navegação com o próprio Xtion e os sonares ao redor, não são indicados para o robô HERA na aplicação nos ambientes reais. Quando se tem disponível apenas a montagem de sensores 1 (com o laser frontal para o SLAM e uma câmera de profundidade e sonares para navegação), indica-se a utilização do pacote de exploração *explore-lite*. Quando forem possíveis as utilizações das montagens de sensores 2 (com os lasers posicionados em direções opostas para o SLAM) ou 3 (com uma câmera de profundidade na traseira do robô em conjunto com o laser frontal para o SLAM), o pacote mais indicado é o RRT-exploration.

5.2.4 Vantagens da Exploração Autônoma

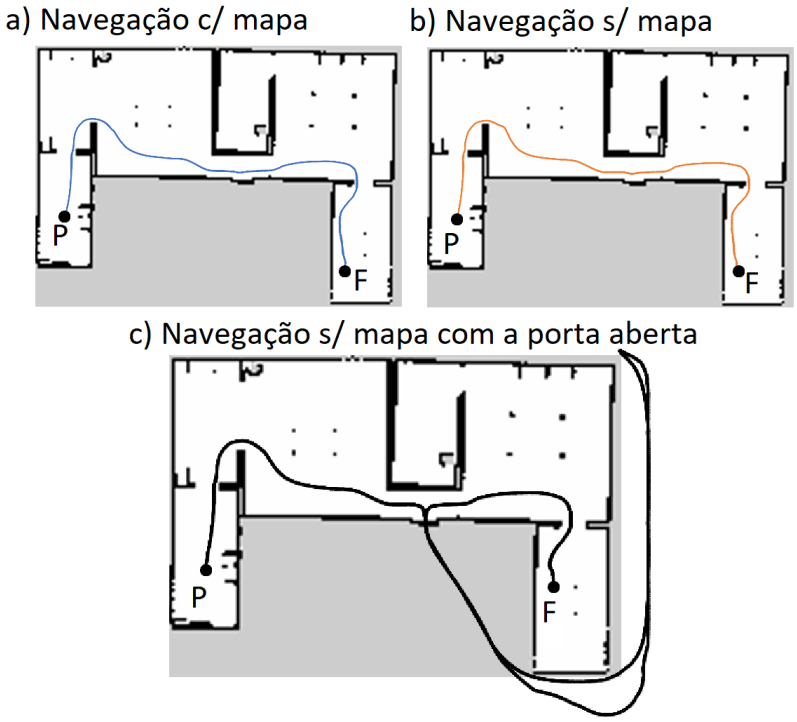
Os resultados mostram que é possível efetivar a exploração autônoma no robô HERA em ambientes residenciais, no entanto, alguns testes foram feitos para entender as vantagens de se utilizar uma exploração do ambiente fechado antes de começar a realizar tarefas no ambiente.

A partir do momento que um mapa fixo é dado como entrada de algum algoritmo que realiza tarefas em determinados ambientes, é possível determinar os limites alcançáveis pelo robô, evitado que este tente chegar até o exterior da residência caso a tarefa envie uma posição fora do mapa 2D e cause um acidente durante a operação. Com o mapa explorado e dado como entrada, o robô simplesmente não realizaria o deslocamento e aguardaria novas instruções caso fosse enviado para fora dos limites conhecidos. Caso não tenha um mapa, o robô poderia se localizar pelo SLAM e tentaria atingir o objetivo seguindo a posição das paredes, e caso a porta para o exterior estivesse aberta, o robô conseguiria sair.

Uma alternativa para o mapeamento pelo SLAM é utilizando pacotes como o *teleop-twist-keyboard* que permitem controlar o robô para realizar esta tarefa manualmente. O problema desse tipo de abordagem é que a interação humana também pode ser prejudicial por depender da habilidade do condutor, além de ser proibitiva para pessoas que têm dificuldades motoras.

Ao realizar a exploração autônoma do ambiente, com todos os cômodos permitidos acessíveis e cômodos proibidos inacessíveis, consegue-se delimitar com precisão o melhor caminho possível de se seguir durante a realização de alguma tarefa. Se houver algum tipo de

Figura 39 – Percursos realizados dado um ponto inicial P até o ponto final F. a) Teste utilizando o mapa criado e se localizando através do *AMCL*; b) teste utilizando SLAM através do *gmapping* com a porta de entrada fechada; c) teste utilizando SLAM através do *gmapping* com a porta de entrada aberta.



Fonte: Autor.

imprevisto pelo caminho, o *move-base* consegue fazer os ajustes e corrigir a trajetória através da implicação do *costmap2D* no o planejamento local da trajetória.

Alguns testes foram feitos para simular a navegação de um ponto inicial P até um ponto final F com as seguintes condições: o primeiro teste mostra o caminho que o robô realizou com o mapa da residência como entrada e sua localização foi utilizando o pacote *AMCL* (o número de partículas utilizado e outras configurações foram as padrões do pacote); o segundo teste propôs o robô realizar o percurso até F sem conhecer o ambiente e utilizando o SLAM através do pacote *gmapping* para localização; e por último, o teste realizado foi feito com a porta de entrada aberta e sem conhecimento prévio do ambiente. O resultado obtido está apresentado na figura 39.

Observa-se que na situação com a porta fechada, os caminhos seguidos nas figuras 39.a) e 39.b) mostram uma boa semelhança, pois o SLAM vai montando o mapa conforme há a operação e, neste caso, como não há grande complexidade do ambiente, não há grandes diferenças. No entanto, operar em uma residência aplicando o SLAM o tempo todo, implica um custo computacional relativamente maior, além de haver a limitação caso o robô enfrente o famoso problema de "sequestro do robô" durante seu serviço. Se isso ocorresse na situação da figura 39.a), o robô conseguiria se reencontrar no ambiente e seguir até o destino final, enquanto que no caso da figura 39.b), o robô iria criar paredes e distorcer totalmente o mapa, podendo oca-

sionar no não cumprimento do objetivo. A imagem da figura 39.c) mostra que o robô acabou percorrendo um percurso muito maior, pois a porta da casa estava aberta. Este comportamento não seria aceitável, pois há um grande risco de haver acidentes, além de ter uma eficiência reduzida e um custo maior para o mesmo objetivo. A situação também poderia ocorrer em locais com maiores dimensões e mais cômodos, gerando problema de tempo e percurso necessários muito maiores para o mesmo objetivo.

6 CONCLUSÃO E TRABALHOS FUTUROS

O projeto propõe o estudo da exploração autônoma em ambientes internos para conseguir criar um mapa com precisão o suficiente para que posteriormente sejam realizados serviços específicos. Essa área é complexa, pois envolve a implementação de métodos de fusão de sensores, de localização e mapeamento simultâneos e de navegação. O estudo foi capaz de trazer as vantagens e desvantagens das configurações de sensores e das diferentes técnicas de exploração estudadas.

A combinação de sensores foram peças-chave para garantir a exploração completa do ambiente mais complexo (arena 2), bem como para entender os efeitos de suas variações no desempenho dos pacotes de exploração aplicados. Os sensores para aplicação do SLAM com o *gmapping* que obtiveram os melhores desempenhos foram os lasers Lidar. O sonar não pôde completar o SLAM com o pacote *gmapping* e o Kinect ou Xtion possuem o ângulo de alcance muito mais baixos que os lasers, o que aumentou consideravelmente a quantidade de distorções nas representações do ambiente, inviabilizando a construção de mapas reutilizáveis com o *gmapping*. Apesar das limitações no SLAM, as montagens dos sensores possibilitaram uma navegação segura pelo ambiente residencial desconhecido.

O pacote *cam-exploration* apesar de ter os melhores resultados para ambientes com poucos obstáculos, não conseguiu completar a tarefa no ambiente simulando uma residência. A montagem de sensores 4, que possui o Xtion com o sensor a laser traseiro executando o SLAM e a navegação com o próprio Xtion e os sonares ao redor, obteve os piores resultados na arena mais simples e não conseguiu completar a arena mais complexa. Estes motivos justificam a contra-indicação dessas estratégias para a aplicação proposta.

Pelo que foi observado nas tabelas 12, 13 e 14 e pelo que foi discutido na subseção 5.2.3, quando se tem disponível apenas a montagem de sensores 1 (com o laser frontal para o SLAM e uma câmera de profundidade e sonares para navegação), indica-se a utilização do pacote de exploração *explore-lite*. Quando forem possíveis as utilizações das montagens de sensores 2 (com os lasers posicionados em direções opostas para o SLAM) ou 3 (com uma câmera de profundidade na traseira do robô em conjunto com o laser frontal para o SLAM), o pacote mais indicado é o RRT-exploration.

Durante os testes ficou evidente a necessidade de haver um pacote de navegação confiável para permitir a conclusão da exploração, e o conjunto de pacotes *navigation* mostraram bom desempenho para a proposta de aplicação. Todos os pacotes de exploração usam a metodologia de navegação em algum ponto de sua construção, principalmente no momento de dar pesos ou ganhos para as fronteiras candidatas a serem exploradas. Sua implementação não otimizada, pode incorrer na exploração incompleta do ambiente, como ocorreu na tentativa de aplicar o *cam-exploration* na arena 2.

O trabalho mostrou as possibilidades e as vantagens de se explorar um ambiente de serviço antes de realizar atividades corriqueiras ou quaisquer tarefas que envolvem o desloca-

mento. A principal conclusão foi em relação à segurança de operação e delimitação de área de serviço, rincipalmente pelo fato de não haver necessidade de ter barreiras físicas para impedir que um robô acesse locais indesejados durante os eventuais serviços que serão indicados.

O trabalho permitiu a criação do pacote *hera-exploration* para ser implementado no robô HERA da RoboFei, no entanto, é necessária a validação destes resultados em aplicações fora do ambiente simulado para a real validação da indicação feita neste trabalho.

6.1 TRABALHOS FUTUROS

O primeiro ponto a ser considerado para os trabalhos futuros é aplicação fora do ambiente simulado, pois diversos problemas que comumente são enfrentados em situações reais não foram consideradas, principalmente relacionados ao SLAM, por conta das imprecisões dos sensores e escorregamentos durante o deslocamento.

O aspecto principal a ser definido pelo robô durante uma exploração é o que deve ser considerado um obstáculo fixo no ambiente e o que deve ser tratado como algo móvel ou transitório. Essa definição certamente dependerá do tipo de aplicação que o mapa terá após a exploração. Se a ideia da exploração é reconhecer os limites do mapa para atuação corriqueira posterior, como no caso de um robô de serviços, só é necessário mapear paredes e móveis grandes ou que raramente são movidos, como sofás, camas, mesas, etc. Neste contexto, o principal ponto a ser estudado é o aprimoramento da implementação do SLAM, buscando tornar possível a aplicação em ambientes dinâmicos que permitam ignorar a presença de humanos ou objetos comumente transitórios na representação do mapa. O estudo do *Deep Learning* para reconhecimento de pessoas e de objetos já possui grandes avanços na literatura e pode ser utilizado em conjunto com estratégias de SLAM para esse objetivo.

Outra linha de pesquisa para dar continuidade ao trabalho é implementar características semânticas nos mapas durante a tarefa de exploração para conseguir diferenciar cômodos ou tipos de móveis para ser possível mudar estratégias de ação durante a atuação posterior do robô de serviços. Esse tipo de reconhecimento também pode ser obtido pela aplicação do *Deep Learning* para diferenciação de características chave de cozinhas, quartos, salas, móveis, etc.

Para aplicações reais o robô fatalmente encontrará situações adversas que impedirão seu progresso durante a exploração, por isso, será de grande valia a capacidade de reconhecer objetos obstruindo a passagem e avaliação da capacidade de removê-los do caminho para continuar a exploração sem necessidade da intervenção humana. Sendo assim, verifica-se que apesar de já terem sido bastante estudadas, as estratégias de exploração ainda possuem muitas possibilidades de otimização para garantir que robôs consigam explorar o ambiente autonomamente podendo lidar com situações cada vez mais complexas e desafiadoras.

REFERÊNCIAS

AGUNBIADE, O. Y.; ZUVA, T. Simultaneous Localization and Mapping in Application to Autonomous Robot. **2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)**, p. 1–5, 2018.

ANDERSEN, M. R. et al. Kinect Depth Sensor Evaluation for Computer Vision Applications. **Aarhus University**, p. 1–37, 2012.

AQUINO JUNIOR, P. et al. HERA: Home Environment Robot Assistant. In: II BRAHUR and III Brazilian Workshop on Service Robotics. [S.l.: s.n.], 2019.

BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (SLAM): Part II. **IEEE robotics & automation magazine**, IEEE, v. 13, n. 3, p. 108–117, 2006.

BERGEON, Y. T. et al. Low cost 3D mapping for indoor navigation. **International Conference on Military Technologies (ICMT) 2015**, p. 1–5, 2015.

BLOESCH, M. et al. CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, p. 2560–2568, 2018.

BOHEZ, S. et al. Sensor fusion for robot control through deep reinforcement learning. **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, p. 2365–2370, 2017.

BROSSARD, M.; BONNABEL, S.; BARRAU, A. Invariant Kalman Filtering for Visual Inertial SLAM. **2018 21st International Conference on Information Fusion (FUSION)**, p. 2021–2028, 2018.

CHEN, Y. et al. LiDAR-Video Driving Dataset: Learning Driving Policies Effectively. **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**, p. 5870–5878, 2018.

CHENG, H.; CHEN, H.; LIU, Y. Topological Indoor Localization and Navigation for Autonomous Mobile Robot. **IEEE Transactions on Automation Science and Engineering**, v. 12, p. 729–738, 2015.

CHOSET, H.; BURDICK, J. W. Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph. **I. J. Robotics Res.**, v. 19, p. 96–125, 2000.

CHRISTIANSEN, P. TractorEYE: Vision-based Real-time Detection for Autonomous Vehicles in Agriculture. **AU Library Scholarly Publishing Services**, 2018.

CONNER, D. C.; WILLIS, J. Flexible Navigation: Finite state machine-based integrated navigation and control for ROS enabled robots. In: SOUTHEASTCON 2017. [S.l.: s.n.], 2017. p. 1–8.

DRROBOT. **URM06-PULSE Ultrasonic SKU SEN0151**. 2019. Disponível em: <https://wiki.dfrobot.com/URM06-PULSE_Ultrasonic_SKU_SEN0151>. Acessado em: 22 de jul. 2019.

DUBÉ, R. et al. Non-uniform sampling strategies for continuous correction based trajectory estimation. **2016 IEEE International Conference on Robotics and Automation (ICRA)**, p. 4792–4798, 2016.

DURRANT-WHYTE, H. F.; BAILEY, T. Simultaneous Localisation and Mapping (SLAM) : Part I The Essential Algorithms. Citeseer, 2006.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. **Computer**, IEEE, v. 22, p. 46–57, 1989.

EVERETT, H. R. **Sensors for Mobile Robots: Theory and Application**. Natick, MA, USA: A. K. Peters, Ltd., 1995.

FOX, D.; BURGARD, W.; THRUN, S. The dynamic window approach to collision avoidance. **IEEE Robotics Automation Magazine**, v. 4, n. 1, p. 23–33, 1997.

GAO, W. et al. An improved Frontier-Based Approach for Autonomous Exploration. **2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)**, p. 292–297, 2018.

GERKEY B. **gmapping**. 2018. Disponível em: <http://wiki.ros.org/gmapping>. Acessado em: 01 de dez. 2019.

GONZÁLEZ-BAÑOS, H.; LATOMBE, J. C. Navigation Strategies for Exploring Indoor Environments. **I. J. Robotics Res.**, v. 21, p. 829–848, 2002.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. **IEEE Transactions on Robotics**, v. 23, n. 1, p. 34–46, 2007.

GURUJI, A. K.; AGARWAL, H.; PARSEDIYA, D. K. Time-efficient A* Algorithm for Robot Path Planning. **Procedia Technology**, v. 23, p. 144–149, 2016. 3rd International Conference on Innovations in Automation and Mechatronics Engineering 2016, ICIAME 2016 05-06 February, 2016.

HASAUINO. **rrt-exploration**. 2014. Disponível em: <https://github.com/hasauino/rrt_exploration>. Acessado em: 01 de abr. 2020.

HOKUYO. **Scanning Laser Range Finder UTM-30LX/LN Specification**. 2008. Disponível em: <https://projects.asl.ethz.ch/datasets/lib/exe/fetch.php?media=hardware:tiltinglaser:utm-30lx-ln-specification.pdf>. Acessado em: 22 de jul. 2019.

HÖRNER, J. **Map-merging for multi-robot system**. 2016. Bachelor's thesis – Charles University in Prague, Faculty of Mathematics e Physics, Praga. Disponível em: <https://is.cuni.cz/webapps/zzp/detail/174125/>. Acessado em: 01 de mar. 2020.

IRALABDISCO. **ira-laser-tools**. 2014. Disponível em: <https://github.com/iralabdisco/ira_laser_tools>. Acessado em: 01 de mar. 2020.

JIANHAO, D.; MEIQIN, L.; WEIHUA, S. Efficient exploration for real-time robot indoor 3D mapping. **2015 34th Chinese Control Conference (CCC)**, p. 6078–6083, 2015.

KAM, M.; XIAOXUN ZHU; KALATA, P. Sensor fusion for mobile robot navigation. **Proceedings of the IEEE**, v. 85, n. 1, p. 108–119, 1997.

KLEEMAN, L. Optimal estimation of position and heading for mobile robots using ultrasonic beacons and dead-reckoning. **Proceedings 1992 IEEE International Conference on Robotics and Automation**, 2582–2587 vol.3, 1992.

KONOLIGE, K.; MARDER-EPPSTEIN, E. **navfn**. 2014. Disponível em: <http://wiki.ros.org/navfn>. Acessado em: 01 de ago. 2019.

KWON, T.; SONG, J. Thinning-based topological exploration using position probability of topological nodes. **Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.**, p. 797–802, 2006.

LABBE M. **rrt-exploration**. 2014. Disponível em: <https://github.com/hasauino/rrt_exploration>. Acessado em: 01 de abr. 2020.

LAVALLE, S. M. **Rapidly-Exploring Random Trees: A New Tool for Path Planning, Report No. TR 98-11**. 1998. – Computer Science Department, Iowa State University. Disponível em: <http://janowiec.cs.iastate.edu/papers/rrt.ps>. Acessado em: 01 de out. 2019.

LEE, S. et al. 2D SLAM Solution for Low-Cost Mobile Robot based on Embedded Single Board Computer. **Advances in Nano, Bio, Robotics and Energy (ANBRE17)**, Seul, 2017.

LIM, E. **sensor-pointcloud**. 2017. Disponível em: <https://github.com/eliotlim/sensor_pointcloud>. Acessado em: 01 de mar. 2020.

LOSCH, R. et al. Design of an Autonomous Robot for Mapping, Navigation, and Manipulation in Underground Mines. **2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, p. 1407–1412, 2018.

LU, D. V.; HERSHBERGER, D.; SMART, W. D. Layered costmaps for context-sensitive navigation. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. [S.l.: s.n.], 2014. p. 709–715.

MAITY, S.; SAHA, A.; BHOWMICK, B. Edge SLAM: Edge Points Based Monocular Visual SLAM. **2017 IEEE International Conference on Computer Vision Workshops (ICCVW)**, p. 2408–2417, 2017.

MARDER-EPPSTEIN E.; PERKO E. **base_local_planner**. 2019. Disponível em: <http://wiki.ros.org/base_local_planner>. Acessado em: 01 de out. 2019.

MARDER-EPPSTEIN, E. **Costmap_2D**. 2018. Disponível em: <http://wiki.ros.org/costmap_2d>. Acessado em: 01 de out. 2019.

_____. **dwa_local_planner**. 2014. Disponível em: <http://wiki.ros.org/dwa_local_planner>. Acessado em: 01 de nov. 2019.

MARDER-EPPSTEIN, E. et al. The Office Marathon: Robust Navigation in an Indoor Office Environment. **International Conference on Robotics and Automation (ICRA)**, 2010.

MATAMOROS, M. et al. **RoboCup@Home 2019: Rules and Regulations (draft)**. [S.l.: s.n.], 2019. http://www.robocupathome.org/rules/2019_rulebook.pdf.

MOORE, T.; STOUCH, D. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In: PROCEEDINGS of the 13th International Conference on Intelligent Autonomous Systems (IAS-13). [S.l.]: Springer, jul. 2014.

MORENZA-CINOS, M. et al. Development of an RFID inventory robot (AdvanRobot). **Studies in Computational Intelligence**, p. 387–417, maio 2017.

MUR-ARTAL, R.; TARDÓS, J. D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. **IEEE Transactions on Robotics**, v. 33, p. 1255–1262, 2017.

NOREEN, I.; KHAN, A.; HABIB, Z. Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions. **International Journal of Advanced Computer Science and Applications**, v. 7, nov. 2016.

PARK, Chanoh et al. Elastic LiDAR Fusion: Dense Map-Centric Continuous-Time SLAM. **2018 IEEE International Conference on Robotics and Automation (ICRA)**, p. 1206–1213, 2018.

PARK, J.; SONG, J. Sensor fusion-based exploration in home environments using information, driving and localization gains. **Appl. Soft Comput.**, v. 36, p. 70–86, 2015.

PIMENTEL F. **hera**. 2020. Disponível em: <https://gitlab.com/fpimentel/hera>. Acessado em: 01 de abr. 2020.

PIMENTEL, J. M. **Exploração autônoma de ambientes baseada em ganho de Informação.** 2016. Dissertação (Mestrado) – Universidade Federal de Minas Gerais – Departamento de Ciência da Computação, Belo Horizonte.

PIRE, T. et al. Stereo parallel tracking and mapping for robot localization. **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, p. 1373–1378, 2015.

QUINLAN, S.; KHATIB, O. Elastic bands: connecting path planning and control. In: [1993] Proceedings IEEE International Conference on Robotics and Automation. [S.l.: s.n.], 1993. 802–807 vol.2.

RAVANKAR, A. A. et al. A hybrid topological mapping and navigation method for large area robot mapping. **2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)**, p. 1104–1107, 2017.

ROCKEY, C. **depthimage-to-laserscan**. 2018. Disponível em: <http://wiki.ros.org/depthimage_to_laserscan>. Acessado em: 01 de fev. 2020.

ROS.ORG. **Documentation**. 2018. Disponível em: <http://wiki.ros.org>. Acessado em: 01.08.2019.

_____. **Navigation**. 2019. Disponível em: <http://wiki.ros.org/navigation>. Acessado em: 20 de jun. 2019.

SCHULTZ, A. C. et al. Unifying Exploration, Localization, Navigation, and Planning Through a Common Representation. In: ICRA. [S.l.: s.n.], 1999.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots**. 2nd. [S.l.]: The MIT Press, 2011.

STACHNISS, C.; GRISETTI, G.; BURGARD, W. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In: ROBOTICS: Science and Systems. [S.l.: s.n.], 2005.

STACHNISS, C.; HAHNEL, D.; BURGARD, W. Exploration with active loop-closing for FastSLAM. **2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)**, v. 2, 1505–1510 vol.2, 2004.

SUN, X. et al. Robotic autonomous exploration SLAM using combination of Kinect and laser scanner. **2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)**, p. 632–637, 2017.

SUNDERHAUF, N. et al. Place categorization and semantic mapping on a mobile robot. **2016 IEEE International Conference on Robotics and Automation (ICRA)**, p. 5729–5736, 2016.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. [S.l.]: The MIT Press, 2005.

THRUN, S. et al. Robust Monte Carlo localization for mobile robots. **Artif. Intell.**, v. 128, p. 99–141, 2001.

TOROSLU, I.; DOĞAN, M. K. Effective sensor fusion of a mobile robot for SLAM implementation. **2018 4th International Conference on Control, Automation and Robotics (ICCAR)**, p. 76–81, 2018.

TSUN, M. T. K. et al. Proposing a Sensor Fusion Technique Utilizing Depth and Ranging Sensors for Combined Human Following and Indoor Robot Navigation. **Proceedings of the Fifth International Conference on Network, Communication and Computing (ICNCC '16)**, 2016.

UMARI, Hassan; MUKHOPADHYAY, Shayok. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In: IEEE. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S.l.: s.n.], 2017. p. 1396–1402.

UTEXAS-BWI. **eband_local_planner**. 2018. Disponível em: <https://github.com/utexas-bwi/eband_local_planner>. Acessado em: 01 de out. 2019.

WANG, H.; YU, Y.; YUAN, Q. Application of Dijkstra algorithm in robot path-planning. In: 2011 Second International Conference on Mechanic Automation and Control Engineering. [S.l.: s.n.], 2011. p. 1067–1069.

WANG, H. et al. Mobile robot SLAM methods improved for adapting to search and rescue environments. **2017 Chinese Automation Congress (CAC)**, p. 988–993, 2017.

WEITENG, Z. et al. Improved reversely a star path search algorithm based on the comparison in valuation of shared neighbor nodes. In: 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP). [S.l.: s.n.], 2013. p. 161–164.

WELCH, G.; BISHOP, G. **An Introduction to the Kalman Filter 2 1 The Discrete Kalman Filter In 1960**. 1994. – Department of Computer Science University of North Carolina.

WURM, K. M.; STACHNISS, C.; BURGARD, W. Coordinated multi-robot exploration using a segmentation of the environment. **2008 IEEE/RSJ International Conference on Intelligent Robots and Systems**, p. 1160–1165, 2008.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. **Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'**, p. 146–151, 1997.

ZHANG, Q. et al. Ear-based exploration on hybrid metric/topological maps. **2014 IEEE/RSJ International Conference on Intelligent Robots and Systems**, p. 3081–3088, set. 2014.