UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Marcelo Eugenio Manfrin Mafalda

DESENVOLVIMENTO DE SISTEMA DE NAVEGAÇÃO AUTÔNOMO COM EXPLORAÇÃO DE FRONTEIRA

Florianópolis
2021

Marcelo Eugenio Manfrin Mafalda

DESENVOLVIMENTO DE SISTEMA DE NAVEGAÇÃO AUTÔNOMO COM EXPLORAÇÃO DE FRONTEIRA

Dissertação submetida ao Programa de Pós-graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia de Automação e Sistemas
Orientador: Prof. Dr. Eng. Marcelo Ricardo Stemmer

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Manfrin Mafalda, Marcelo Eugenio
DESENVOLVIMENTO DE SISTEMA DE NAVEGAÇÃO AUTÔNOMO COM EXPLORAÇÃO DE FRONTEIRA / Marcelo Eugenio Manfrin Mafalda ; orientador, Marcelo Ricardo Stemmer, 2021.
110 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Robótica Móvel. 3. Visão Computacional. 4. Planejamento de Trajetória. 5. Exploração de Fronteira. I. Stemmer, Marcelo Ricardo. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Engenharia de Automação e Sistemas. III. Título.

Marcelo Eugenio Manfrin Mafalda

DESENVOLVIMENTO DE SISTEMA DE NAVEGAÇÃO AUTÔNOMA COM EXPLORAÇÃO DE FRONTEIRA

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Marcelo Ricardo Stemmer, Dr.
Instituição Universidade Federal de Santa Catarina

Prof. Tiago Loureiro Figaro Da Costa Pinto, Dr.
Instituição Universidade Federal de Santa Catarina

Prof. Ubirajara Franco Moreno, Dr.
Instituição Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Automação e Sistemas.

ICP-Edu
Documento assinado digitalmente
Werner Kraus Junior
Data: 19/07/2021 17:50:59-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de Pós-Graduação

ICP-Edu
Documento assinado digitalmente
Marcelo Ricardo Stemmer
Data: 19/07/2021 14:25:24-0300
CPF: 376.093.279-72
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Eng. Marcelo Ricardo Stemmer
Orientador

Florianópolis, 2021.

Este trabalho é dedicado aos meus queridos pais.

AGRADECIMENTOS

À Universidade Federal de Santa Catarina (UFSC) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelos recursos fornecidos. A todos meus familiares e amigos, principalmente aos meus pais por todo o apoio e incentivo. Tambem a todos meus professores e em especial ao meu orientador Prof. Dr. Eng. Marcelo Ricardo Stemmer.

RESUMO

O campo da robótica vem obtendo significativos avanços nas últimas décadas. Uma de suas principais áreas, a da robótica móvel tenta solucionar uma gama de problemas incluindo neles o problema da localização, do SLAM (Localização e Mapeamento Simultâneos), do planejamento de trajetória e da exploração. O problema da navegação autonôma vem sendo estudado a tempos pela comunidade ciêntifica, já que possui uma grande importancia, dado o fato que para se locomover entre dois pontos o robô ou veiculo autônomo precisa de um algoritmo de planejamento de trajetória, porém com o advento do ROS (Sistema operacional de robôs) a situação ficou mais fácil, já que é possível utilizar algoritmos e programas em alto nível em diversos hardwares diferentes de baixo nível. As técnicas tradicionais tendem a considerar um modelo global do ambiente; no entanto, os problemas reais de planejamento de trajetórias usualmente estão no âmbito de ambientes desconhecidos ou parcialmente desconhecidos. Neste contexto entra a situação de exploração, em que o robô não possui um mapa e deseja navegar pelo ambiente, para isso é necessária uma estratégia de exploração. Já que caminhos ideais podem ser entendidos como trajetórias que melhor atingem um objetivo, minimizando a distância percorrida ou o tempo gasto, por exemplo, a estratégia de exploração utilizada aqui é a de exploração de fronteiras, a qual leva um robô sempre a navegar até a fronteira de um mapa conhecido com a finalidade de ampliá-lo. Tambem, a utilização de visão computacional em robôs móveis significa um grande aumento em suas habilidades sensoriais, o que significa uma maior versatilidade e segurança nas aplicações do robô. Neste trabalho é desenvolvido um sistema, utilizando ROS, que possa realizar a exploração de fronteira de modo a construir um mapa do ambiente de maneira autônoma. São abordados conceitos chave para o entendimento do sistema, como localização, SLAM, planejamento de trajetória, exploração, descritores de características visuais e odometria visual, bem como uma abrangente revisão de literatura sobre os últimos avanços na área. Ao final é demonstrado por meio de dois experimentos em simulação, utilizando o simulador gazebo, os resultados obtidos, ressaltando em conclusão as diferenças entre a odometria obtida por laser e a odometria visual.

Palavras-chave: Robótica Móvel. Visão Computacional. Exploração de Fronteira.

ABSTRACT

The field of robotics has been making significant advances in recent decades. One of its main areas, that of mobile robotics, tries to solve a range of problems, including the problem of location, SLAM (Simultaneous Location and Mapping), trajectory planning and exploration. The problem of autonomous navigation has been studied for a long time by the scientific community, since it has great importance, given the fact that to move between two points the robot or autonomous vehicle needs a trajectory planning algorithm, however with the advent of ROS (Robots operating system) the situation has become easier, since it is possible to use high-level algorithms and programs on several different low-level hardware. Traditional techniques tend to consider a global model of the environment; however, the real problems of trajectory planning are usually within the scope of unknown or partially unknown environments. In this context, the exploration situation enters, in which the robot does not have a map and wants to navigate the environment, for this an exploration strategy is necessary. Since ideal paths can be understood as trajectories that best reach an objective, minimizing the distance traveled or the time spent, for example, the exploration strategy used here is the frontier exploration, which leads a robot to always navigate to the border of a known map for the purpose of enlarging it. Also, the use of computer vision in mobile robots means a great increase in their sensory abilities, which means greater versatility and safety in the robot's applications. In this work, a system is developed, using ROS, that can carry out frontier exploration in order to build an autonomous map of the environment. Key concepts for understanding the system are addressed, such as location, SLAM, trajectory planning, exploration, descriptors of visual characteristics and visual odometry, as well as a comprehensive review of the literature on the latest advances in the area. At the end it is demonstrated by means of two experiments in simulation, using the gazebo simulator, the results obtained, emphasizing in conclusion the differences between the odometry obtained by laser and the visual odometry.

Keywords: Mobile Robotics. Computer Vision. Frontier Exploration.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente, estamos à beira de uma revolução na robótica. Uma variedade de sistemas robóticos foram desenvolvidos e demonstraram sua eficácia na realização de diferentes tipos de tarefas, incluindo ambientes domésticos inteligentes (AL-KHAWALDEH et al. 2016), aeroportos (ZHOU et al. 2016), shoppings (KANDA et al. 2009) e laboratórios de manufatura (CHEN et al. 2014). Uma inteligência deve ser incorporada ao robô para garantir a execução (quase) ótima da tarefa em consideração e com eficiência cumprir a missão. No entanto, a incorporação de inteligência no sistema robótico impõe a resolução de um grande número de problemas de pesquisa, como a navegação, que é um dos problemas fundamentais dos sistemas de robótica móvel.

Robôs são capazes de realizar tarefas executando movimentos em um espaço físico com objetos. Desta forma, eles são como dispositivos mecânicos versáteis equipados com sensores e atuadores sob o controle de um sistema computacional. (SACCHETIN, 2006). De acordo com Gaspar (1994), um robô móvel consiste em uma plataforma móvel sobre a qual é integrada percepção e ação no ambiente que o rodeia. A robótica agora está ganhando muito espaço em nossa vida cotidiana, em várias áreas na automação industrial moderna e em aplicações ciberfísicas. Isto exige incorporar inteligência nesses robôs para garantir soluções quase ótimas para a execução de tarefas. Assim, surgiram muitos problemas de pesquisa relacionados a aplicações robóticas, como o planejamento (caminho, movimento e missão) problemas de alocação de tarefas e rastreamento de navegação. Robôs móveis desempenham um papel significativo em uma variedade de ambientes e tarefas, e a robótica móvel cresceu rapidamente na última década, produzindo ferramentas para substituir o trabalho humano. (X. WANG et al. 2020). Em geral, os robôs móveis operam com informações incompletas sobre o ambiente em que atuam. Pesquisas consideráveis foram feitas sobre a aprendizagem do ambiente pelo robô ao se mover por meio da utilização de vários tipos de sensores. Uma variedade de sensores têm sido utilizados para realizar a tarefa de localização (BICCHI et al., 2004), como o *laser range finder* (LRF) (ZHANG; GHOSH, 2000) (SURMANN; NÜCHTER; HERTZBERG, 2003), sensores ultrassônicos (MORENO et al., 2002) sistema de posicionamento global (GPS) (AGRAWAL; KONOLIGE, 2006) (REINA et al., 2007), e unidade de medida inercial (IMU) (YI et al., 2007). No entanto, são geralmente caros e podem ter restrições, como o grau de detalhe da informação ou velocidade limitada. A visão computacional aborda o comportamento autônomo de um sistema suportado por informação sensorial visual. Para fornecer alguns

métodos de análise de sequências de imagens digitais para aplicações robóticas móveis, como tarefas de localização e navegação, a compreensão da informação visual, requer uma combinação de criação de conceitos de alto nível, bem como processamento e interpretação de *features* visuais inerentes. A visão evoluiu das aplicações das técnicas clássicas de reconhecimento de padrões e processamento de imagem para aplicações avançadas de entendimento da imagem, visão baseada em modelo, visão baseada em conhecimento e sistemas que apresentam capacidade de aprendizado (X. WANG et al. 2020). Como mencionado por Schneider (2018), podemos perceber que a introdução de visão em um robô móvel implica em um significativo aumento das suas capacidades sensoriais e, portanto, em um correspondente aumento na versatilidade e segurança na operação do robô. A navegação robótica baseada em visão tem sido um tema bem explorado por pesquisadores tanto na área de robótica como na área de visão computacional (NIN; OSORIO, 2011).

Para realização de tarefas de forma autônoma, ou seja, sem a intervenção humana, os robôs precisam ser capazes de realizar navegação. Conceitualmente o termo navegar na robótica, consiste em guiar um robô por um caminho de forma a levá-lo de uma posição e orientação iniciais até uma posição e orientação finais (BARRERA 2010). Conforme o robô se move precisamos acompanhar sua localização e orientação em relação a outros objetos no ambiente. O conhecimento preciso da localização do robô é crucial em qualquer forma de tarefas de navegação de alto nível, como seguir trajetórias planejadas com prevenção de obstáculos. Citado por Schneider (2018) o problema de localização robótica pode ser classificado em três tipos de problemas: Rastreamento de posição (ou localização local), localização global e o problema do robô sequestrado. A localização local do robô é atualizada com base no conhecimento de sua posição anterior (rastreamento). Isso implica que a localização inicial do robô deve ser conhecida. A localização global, em contrapartida, pressupõe que a localização inicial do robô é desconhecida. Isso significa que o robô pode ser colocado em qualquer lugar do ambiente, sem conhecimento disto, e é capaz de localizar-se globalmente dentro dele. O problema do robô sequestrado, no qual ele é movido involuntariamente, se assemelha ao problema de localização global somente se o robô perceber que foi sequestrado. A dificuldade surge quando o robô não sabe que ele foi movido para outro local e acredita que sabe onde está, mas na verdade não sabe (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Outro problema de grande importância no desenvolvimento de robôs móveis verdadeiramente autônomos é o mapeamento, que tem aplicações em navegação, telepresença, manipulação e mapeamento semântico (HENRY et al., 2012). O problema da

exploração autônoma em ambientes desconhecidos requer um sistema que possa estimar o estado do robô e mapear simultaneamente. Tal sistema é referido como SLAM (*Simultaneous Localization and Mapping*). SLAM, como o nome sugere, permite que o robô construa um mapa do ambiente e se localize no mapa simultaneamente. No algoritmo SLAM, o robô com a ajuda dos sensores localiza os pontos de referência quando eles são visitados pela primeira vez e suas posições são armazenadas em um mapa. As posições dos pontos de referência são calculadas e quando o robô visita esses pontos de referência novamente, esses pontos de referência são usados para melhorar a estimativa de posicionamento do robô. O mapeamento não é uma tarefa trivial, pois o robô precisa saber sua localização para poder desenvolver um mapa, e o local só pode ser determinado usando os recursos de um mapa.

A cognição geralmente representa a tomada de decisão proposital e execução que um sistema utiliza para atingir suas metas de ordem mais alta. No caso de um robô móvel, o aspecto específico da cognição está diretamente vinculado a mobilidade e é competência de navegação. Dado conhecimento parcial sobre seu ambiente e uma posição objetivo ou série de posições, a navegação abrange a capacidade do robô de agir com base em seus conhecimentos e valores de sensores, de modo a alcançar suas posições de objetivo de forma mais confiável possível. Dentro da comunidade de pesquisa em robótica móvel, muitas abordagens foram propostas para resolver o problema de navegação. Como provado a partir desta base de pesquisa ficará claro que, de fato, existem fortes semelhanças entre todas essas abordagens mesmo que pareçam, na superfície, bastante díspares. A principal diferença entre várias arquiteturas de navegação é a maneira pela qual decompõem o problema em subunidades menores (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004). Dado um mapa e uma localização objetivo, o planejamento do caminho envolve a identificação de uma trajetória que fará com que o robô alcance a localização objetivo quando executado. O planejamento de caminho é uma competência estratégica para a solução de problemas, pois o robô deve decidir o que fazer a longo prazo para atingir seus objetivos. A segunda competência é igualmente importante, mas ocupa o extremo oposto, tático. Dadas as leituras dos sensores em tempo real, evitar obstáculos significa modular a trajetória do robô para evitar colisões. Uma grande variedade de abordagens demonstrou conseguir evitar obstáculos competentemente, neste trabalho são também pesquisadas várias dessas abordagens. O problema de planejar trajetórias de veículos com base em informações locais, ou seja, obstáculos, posição da estrada ou pontos de referência fornecidos pelo usuário, é discutido. Para missões não tripuladas reais, é necessário um planejamento de longo alcance. Em particular, o sistema precisa planejar automaticamente um

caminho da posição atual do veículo para sua posição de objetivo. A situação é complicada pelo fato de que, usando apenas informações parciais sobre o meio ambiente, o caminho deve contornar obstáculos (STENTZ, 1993)

Além, definimos Planejamento, Localização e Mapeamento Simultâneos (SPLAM) como a tarefa de navegar por um ambiente desconhecido enquanto construímos um mapa e, ao mesmo tempo, garantir estimativas robustas de pose e mapa de robô sem nenhuma intervenção humana. O principal objetivo do SPLAM é planejar caminhos para o SLAM, de modo que a incerteza da posição do robô e do mapa na execução do caminho permaneçam mínimos e tratáveis. O planejamento é intercalado com SLAM e, portanto, a terminologia SPLAM. Enquanto rotinas típicas de SPLAM encontram caminhos enquanto o robô atravessa regiões conhecidas do mapa construído, aqui é tentada a formulação para uma situação de exploração. Exploração é a tarefa de fornecer ao robô seu próximo local de destino, para que ele possa mapear as áreas desconhecidas de um ambiente. Normalmente equipado com sensores (laser, sonar, etc), o robô só consegue ver até uma distância finita. Por isso, precisa ir além de sua posição atual para capturar e mapear outras partes do ambiente. Essa tarefa de identificar e fornecer ao robô um local de destino, de modo que, ao atingir o local de destino o robô possa explorar o ambiente desconhecido é feito por meio de uma estratégia de exploração. Inicialmente, você só consegue perceber até um determinado alcance de sua localização atual. Seu objetivo é explorar e construir um mapa que represente o ambiente completo e você quer que isso seja feito em um tempo mínimo. Para obter o máximo de novas informações sobre o ambiente, o melhor método é avançar para o limite entre o território conhecido e desconhecido. Este limite é denominado como fronteira. Quando um robô se move para uma fronteira, ele pode ver o espaço inexplorado e adicionar as novas informações ao mapa. Como resultado o território mapeado expande, empurrando de volta a fronteira entre o conhecido e o desconhecido. Desta forma, o robô se mantém estendendo o mapa conhecido até que seja completamente explorado. Esta estratégia é conhecida como exploração de fronteira (YAMAUCHI, 1997).

1.1 OBJETIVOS

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos desta Dissertação.

1.1.1 Objetivo Geral

Este trabalho tem como objetivo o estudo do problema de navegação, assim como a desenvolvimento de um sistema de navegação e simulação que possa ser utilizado em robôs móveis de driver diferencial, utilizando técnicas de visão computacional para possuir capacidades robustas de estimação de localização, mapeamento simultâneos e planejamento de trajetória dada uma certa posição final e ainda realizar exploração de ambientes desconhecidos. O sistema tem a proposta de aproveitar o *Robot Operating System* para facilitar sua implementação. O sistema utilizará uma câmera RGBD (Microsoft Kinect) para realizar a estimação de pose em uma simulação e um laser 2D em outra.

1.1.2 Objetivos Específicos

- Estudo de artigos relevantes para o trabalho na literatura.
- Desenvolver sistema com SLAM utilizando técnicas de odometria visual.
- Desenvolver parte de planejamento de trajetória em conjunto com os mapas gerados pelo algoritmo de SLAM.
- Aplicar o sistema em simulações de um robô de driver diferencial

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

A dissertação está organizada da seguinte forma: No capitulo dois é apresentada a fundamentação teórica que servira de base para o entendimento do trabalho. O capítulo três apresenta uma revisão da literatura para se comparar os estudos recentes na área. No capítulo quatro é exibida a simulação do sistema bem como experimentos e resultados. Finalmente no capitulo cinco é feita a conclusão do trabalho com uma discussão sobre trabalhos futuros.

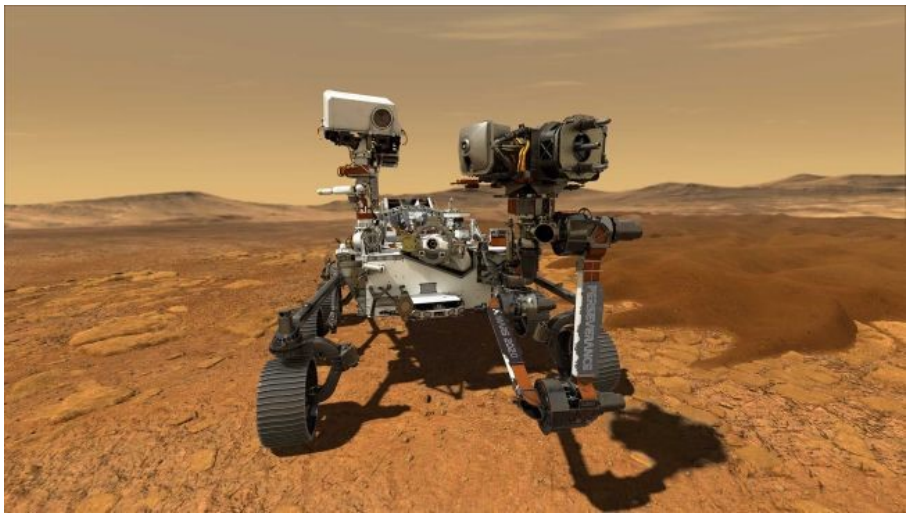
2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo serão apresentados os conceitos e definições que irão formar a base de conhecimento necessária para o entendimento do trabalho desenvolvido. Na área de robótica móvel serão abordados os conceitos de posição e orientação, navegação, localização, SLAM (localização e mapeamento simultâneo), planejamento de trajetória e o problema de exploração. Enquanto na parte de visão computacional serão discutidos descritores de características e a técnica de obtenção de odometria visualmente. Tambem será discutido brevemente o funcionamento do ROS *(Robot Operating System)*.

2.1 CONCEITOS RELACIONADOS A ROBÓTICA MÓVEL

Os robôs desempenham um papel crescente em nossas vidas e locais de trabalho em áreas tão diversas quanto manufatura, agricultura, aeroespacial, mineração e medicina alcançando assim grande sucesso no mundo da produção industrial. Porém, apesar deste sucesso os robôs comerciais sofrem de uma grande desvantagem: a falta de mobilidade. Um manipulador fixo possui uma grande limitação de movimentos que depende de onde ele é colocado. Em contraste, robôs móveis podem ser capazes de se deslocar entre diferentes ambientes utilizando de forma flexível as suas habilidades onde quer que sejam mais eficazes. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004). Além, eles podem percorrer áreas de risco, como os robôs Pioneer na usina nuclear de Chernobyl após um acidente nuclear e o Mars Rover (Figura 1) no planeta Marte, em ambientes que a presença humana se torna praticamente impossível.

Figura 1 – Mars Rover



Fonte: mars.nasa.gov

A robótica móvel é um ramo orientado para soluções que está se desenvolvendo rapidamente, mesclando ciências da engenharia e tecnologia da informação com disciplinas como ciências cognitivas, inteligência artificial e muitas outras. É essa interação interdisciplinar que tornou possível dominar a complexidade inerente dos robôs móveis. Robôs móveis são capazes de movimento independente e executar determinadas ações. Essencialmente, além de sua mobilidade, eles são capazes de funcionar autonomamente, sem a necessidade de intervenção humana. Um robô móvel necessita de mecanismos de locomoção que o permitam movimentar-se ao longo do seu ambiente. Siegwart, Nourbakhsh; Scaramuzza, (2004) afirmam que a navegação é uma das competências mais difíceis exigidas de um robô móvel. Para de fato obter esta habilidade há quatro estágios de navegação que precisam ser alcançados: a percepção, o robô deve entender, ou seja, saber interpretar os dados de seus sensores; localização, o robô deve conseguir saber sua localização no ambiente; cognição, o robô possuir poder de decisão para realizar escolhas sejam previsíveis ou não; e controle de movimento, o robô deve controlar os giros dos motores para atingir a trajetória desejada. Há diferentes mecanismos que podem habilitar um robô a se locomover, geralmente os mais utilizados são rodas, pernas articuladas ou hélices. Embora o campo da robótica móvel esteja atualmente em um estágio relativamente de desenvolvimento, protótipos e até alguns produtos em série estão começando a ser encontrados em uma ampla gama de setores. A robótica móvel também chegou às aplicações domésticas diárias com a invenção dos aspiradores de pó móveis, capazes de funcionar com

autonomia quase completa, ou até o desenvolvimento de ‘mordomos’ como demonstrado por Srinivasa et al (2010) na Figura 2.

Figura 2 – Robô HERB

Fonte: Srinivasa et al (2010)

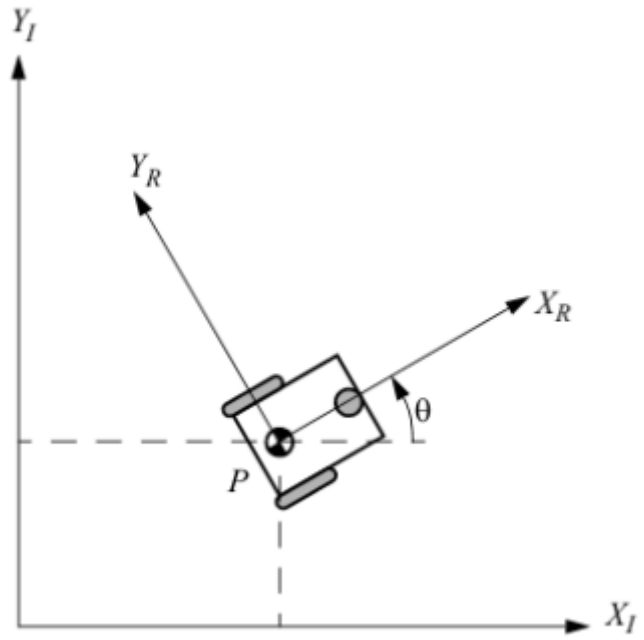
Apesar de exibirem semelhanças fundamentais, a robótica móvel, difere de muitas maneiras da robótica clássica. Suas diferenças centrais podem ser esclarecidas comparando-as com robôs industriais. Como a maioria dos robôs clássicos operam quase exclusivamente com base em um padrão predeterminado, eles não exibem independência operacional, não necessitando de sensores. Por outro lado, os robôs móveis dependem essencialmente de uma ampla gama de sensores, como câmeras, scanners a laser, ultrassom e outras tecnologias. É incorporando esses componentes que os sistemas de robôs móveis são capazes de responder a eventos que ocorrem espontaneamente. Outra distinção elementar é encontrada na maneira como os robôs são programados. A robótica móvel é baseada em funções e pode ser considerada orientada a tarefas, enquanto os robôs industriais são executados de acordo com uma tarefa explícita que compreende uma sequência estrita de comandos. A capacidade de um dispositivo móvel de planejar e executar seus movimentos de forma independente deve-se a uma

combinação de um sistema sensorial complexo e programação relativamente livre. A sequência de movimentos de um robô móvel é consequentemente guiada por mudanças no ambiente imediato, enquanto os robôs tradicionais, por via de regra, repetem constantemente o mesmo processo de trabalho. O ponto principal dos robôs móveis, no entanto, é que eles podem variar sua operação de acordo com o ambiente; como dispositivos autônomos, eles são capazes de explorar independentemente novos ambientes com base em suas próprias habilidades de percepção e aprendizado. Novas informações são sujeitas a processamento adicional e são levadas em consideração em futuras sequências de movimento no respectivo ambiente.

2.1.1 POSIÇÃO E ORIENTAÇÃO

A robótica móvel tem um lugar de destaque, pois os robôs não ficam restritos a um único local de ação e, por isso, têm uma maior aplicabilidade e versatilidade que os robôs manipuladores ou fixos. A posição e orientação de um corpo rígido no espaço são designados como a pose. A cinemática do robô descreve a aceleração, velocidade e todos os derivados de ordem superior da pose dos sistemas que compõem um mecanismo (SICILIANO; KHATIB, 2008). O entendimento do modelo cinemático do robô é imprescindível tanto para o projeto de robôs móveis apropriados para tarefas orientadas como para criar um programa de controle de hardware do robô (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004). A cinemática espacial e de corpo rígido pode ser vista como um estudo comparativo de diferentes maneiras de representar a pose de um corpo, translações e rotações, referidas em combinação como deslocamentos de corpo rígido, que também são expressas com essas representações (SICILIANO; KHATIB, 2008). O robô é modelado como um corpo rígido sobre rodas em um plano horizontal. A dimensionalidade total do robô no plano seria de 3 dimensões, duas para posição no plano e um para orientação ao longo do eixo vertical, que seria ortogonal ao plano, vemos um exemplo disto na Figura 3. O espaço de trajetórias de um robô móvel é importante porque define a gama de poses possíveis que ele pode alcançar em seu ambiente (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2004).

Figura 3 - O quadro de referência do robô em 2D.



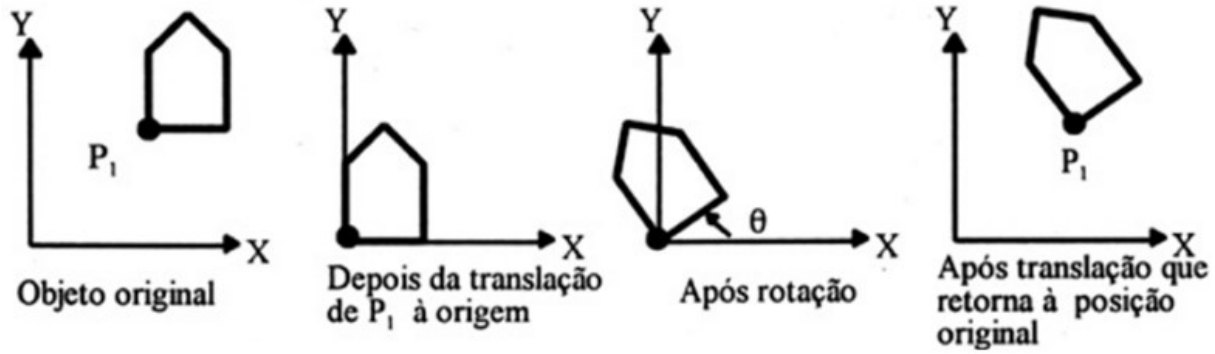
Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

Um sistema de coordenadas i consiste de uma origem, denotada O_i, e uma tripla de vetores ortogonais entre si, denotados X_i, Y_i e Z_i que são todos fixos em um corpo. Para que a pose possa ser denominada como a pose de uma coordenada relativa a outra, a pose de um corpo estará sempre denominada em relação a alguma outra pose. Desta forma, deslocamentos de corpos rígidos podem ser expressos como deslocamentos entre dois sistemas de coordenadas, um dos quais pode ser referido como fixo, enquanto o outro pode ser referido como em movimento. Isto demonstra que o observador está localizado em uma posição fixa dentro do sistema de referência fixo (SICILIANO; KHATIB, 2008). Para especificar a posição do robô no plano, é necessário estabelecer uma relação entre os sistemas de referência local e global do robô. A posição da origem do sistema de coordenadas i relativa ao sistema de coordenadas j pode ser indicada pelo seguinte vetor 3 × 1:

$$ {}^{j}P_i = \begin{pmatrix} {}^{j}P_i^x \\ {}^{j}P_i^y \\ {}^{j}P_i^z \end{pmatrix} \tag{1} $$

Os elementos deste vetor são as coordenadas cartesianas de O_i no sistema de coordenadas j, que são as projeções do vetor ${}^{j}P_i$ sobre os eixos apropriados. Os elementos do vetor também poderiam ser expressos como coordenadas esféricas ou cilíndricas de O_i em j. Tais aspectos apresentam vantagens para a análise de mecanismos robóticos incluindo juntas esféricas e cilíndricas (SICILIANO; KHATIB, 2008). Uma translação é um movimento no qual nenhum ponto no corpo rígido permanece na sua localização inicial e todas as linhas retas no corpo rígido permanecem paralelas à sua orientação inicial. A translação de um corpo no espaço pode ser representada pela combinação de suas localizações antes e depois da translação. Assim, qualquer exibição de localização pode ser usada para criar uma representação de deslocamento, e vice-versa (SICILIANO; KHATIB, 2008). Uma rotação é um movimento no qual pelo menos um ponto do corpo rígido se mantem na sua posição inicial e nem todas as linhas no corpo se mantem paralelas as suas orientações iniciais. Por exemplo, um corpo em uma orbita circular gira em torno de um eixo através do centro de sua trajetória circular, e cada ponto no eixo de rotação é um ponto no corpo que se mantem em sua localização inicial. Como no caso de posição e translação, qualquer exibição de orientação pode ser usada para criar uma exibição de rotação e vice-versa (SICILIANO; KHATIB, 2008). Segue na figura 4 um exemplo de translação e rotação de um robô.

Figura 4 – Exemplo de Translação e Rotação.



Fonte: inf.unioeste.br

Ainda assim existem muitas mais maneiras de representações de orientação e de posição. Alguns exemplos são ângulos de Euler, ângulo/eixo, quatérnions e matrizes de rotação. Um quatérnion pode ser demonstrado como um vetor de 4 componentes, sendo um vetor escalar

e um ordinário, ou como um número complexo com três partes imaginárias diferentes (HORN, 1987). Segundo Horn (1987), um quatérnion q é determinado com a forma:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{2}$$

onde os componentes *q0, q1, q2, q3* são escalares, com *q0* exibindo a parte real e os outros termos as três partes imaginárias, definidos como parâmetros de Euler, e *i*, *j* e *k* são operadores. Os operadores são compostos de modo que se obedeçam as seguintes regras:

$$\begin{gathered} ii = kk = jj = -1, \\ ij = k, jk = i, ki = j, \\ ji = -k, kj = -i, ik = -j \end{gathered} \tag{3}$$

Muitas vezes, *q0* é referido como a parte escalar do quatérnion, e $(q1\ q2\ q3)^T$ referido como a parte vetorial. Um quatérnion cuja norma é igual a 1 é chamado quatérnion unitário. Quatérnions unitários são usados para descrever a orientação, sendo extremamente úteis para questões em robótica que se encontram em singularidades representacionais em vetores/matrizes. Um vetor é definido na notação quatérnion como um quatérnion com $q_0 = 0$ Assim, um vetor $p = (P_x P_y\ P_z)^T$ pode ser definido como um quatérnion $p = p_x i + p_y j + p_z k$. Rotações tridimensionais podem ser representadas por quatérnions unitários (KELLY, 2013):

$$\tilde{q} = \cos(\frac{\theta}{2}) + \hat{w}\sin(\frac{\theta}{2}) \tag{4}$$

Onde a equação 4 representa o operador que gira pelo ângulo θ ao redor do eixo cujo vetor unitário é $\hat{w}$.

2.1.2 NAVEGAÇÃO

Através da localização o robô é capaz de calcular sua posição no ambiente, enquanto os algoritmos de mapeamento são responsáveis por criar um mapa desse ambiente, o que possibilita criar rotas para locais que devam ser alcançados e estabelecer outras tarefas a serem realizadas no ambiente (SACCHETIN 2005). Bellotto et al. (2008) afirmam que a navegação

pode ser descrita resumidamente como o processo de determinar uma trajetória adequada e segura entre um ponto inicial e um ponto de objetivo para um robô que navega entre eles. Em particular, nas últimas décadas, a navegação visual para robôs móveis tornou-se uma fonte de inúmeras contribuições de pesquisa, a partir dos domínios de visão e controle, uma vez que as estratégias de navegação baseadas em visão podem aumentar o escopo de aplicação de veículos móveis autônomos, se tornando cada vez mais comum em aplicações como localização, construção automática de mapas, navegação autônoma, seguimento de caminhos, inspeção, monitoramento ou detecção de situações de risco (BARBOSA 2017). Na maioria dos casos o desempenho de um algoritmo de navegação está fortemente associado a uma localização precisa do robô no ambiente. Assim, em robótica móvel, a localização exerce um papel fundamental na tarefa de navegação, dado que é necessária para praticamente todo o tipo de planejamento de trajetória mesmo com os mapas locais atualizados incorporando novas características ocasionadas por mudanças no ambiente a cada execução do algoritmo que não se encerra até que o robô consiga chegar ao seu destino. Para atingir um objetivo, um robô móvel autônomo deve ser capaz de localizar-se dentro do ambiente onde está navegando e relativamente ao destino objetivo. (BELLOTTO et al., 2008). Thrun et al. (2002) definem o problema da localização robótica como sendo a determinação da pose do robô em relação ao mapa do ambiente, sendo este mapa definido em um sistema global de coordenadas. Com isso a localização seria a transformação de coordenadas entre o sistema de coordenadas local do robô e o sistema de coordenadas global externo aonde o mapa está inserido. Com relação ao ambiente em que o robô será inserido, o problema da localização pode estar relacionado à ambientes estáticos, ou seja, onde é assumido que o único agente em movimento no ambiente é o próprio robô, permanecendo os demais objetos sem movimento. Em ambientes dinâmicos outros objetos sofrem alterações ou mudam de posição com o tempo, como por exemplo, a movimentação de pessoas, mobília que é reposicionada na cena, ou portas que são abertas ou fechadas, ou mesmo outros robôs móveis em uma aplicação cooperativa, como uma frota de robôs. Apesar dos ambientes dinâmicos representarem melhor os ambientes reais, apresentam também maiores dificuldades e complexidades computacionais, uma vez que eventos dinâmicos persistentes precisam ser modelados no sistema de localização (como por exemplo, a inclusão de alterações dinâmicas no vetor de estado ou a filtragem de dados provenientes de sensores).(SANCHES 2009). Thrun (1998) faz a distinção entre dois tipos de solução para o problema da localização. Na localização baseada em marcos de localização, objetos do ambiente (marcos) são utilizados como referencial para prover localização. Na localização por

casamento de modelos (“model matching”), características geométricas, como segmentos de retas, são extraídas de imagens do ambiente e comparadas com um modelo do ambiente, como por exemplo, um mapa geométrico, possibilitando assim a verificação dos erros da odometria. Na odometria a estimativa da posição é feita a partir da contagem de pulsos de “encoders” instalados nas rodas do robô, integrando-se seu movimento de maneira consecutiva (BEKEY, 2005), (RIBEIRO, 1999). O modelo cinemático do movimento do robô é utilizado, a partir desta contagem de pulsos, para a estimativa da pose do robô a cada instante. A trajetória do robô pode desta maneira ser estimada. O termo *dead reckoning* está relacionado com a determinação da localização de um agente móvel baseado em cálculos provenientes da odometria. Existem diferentes métodos que visam encontrar a localização do robô. Entre eles, os mais comuns são o *dead reckoning* e a localização de mapa a priori. A localização *dead reckoning* é um cálculo da posição do robô com base nas leituras da odometria das rodas. O método *dead reckoning* seria bastante eficaz na ausência de erro de movimento, mas para todas as aplicações práticas, cada mudança de posição do robô inclui um componente de erro. Então a cada movimento subseqüente do robô esses erros aumentam continuamente. Portanto, não importa quão preciso seja o modelo de movimento, mesmo que ele carregue uma pequena quantidade de erro a incerteza na posição do robô continuará aumentando e eventualmente a estimativa de posicionamento será tão imprecisa que não poderá mais ser usada. Por essa razão, o algoritmo de *dead reckoning* por si só, não é um mecanismo adequado para realizar a localização a longo prazo. No entanto, a localização de mapa a priori tem a capacidade de corrigir erros de movimento. Mas, no centro da localização de mapa a priori, está a suposição de que um mapa do ambiente contendo definições exatas de marcos distintos está disponível com antecedência. Assim, um robô equipado com qualquer tipo de sensor pode detectar a localidade próxima para obter os pontos de referência e mapear esses pontos de referência com o mapa a priori do ambiente para obter uma estimativa precisa da posição do robô. A principal desvantagem da localização baseada em mapas a priori é que ela exige que o mapa seja construído a partir do ambiente explorado antes da navegação autônoma pelo ambiente. Além disso, o mapa resultante é estático e não pode se adaptar a mudanças no ambiente ou crescer com a exploração dos limites do mapa original. Por isso é necessário um sistema capaz de corrigir erros de movimento, enquanto possa construir simultaneamente um mapa robusto do ambiente, evitando assim a necessidade de ter um mapa a priori construído antes da exploração pelo robô. Os principais problemas dos métodos baseados em odometria consistem no fato de que, para se obter a precisão necessária para aplicações em robótica, são necessários encoders de alta

resolução, os quais podem aumentar significativamente os custos de implementação e que, em odometria, o erro é acumulativo. Uma alternativa ao uso dos encoders ópticos é a utilização de métodos baseados em visão computacional, que podem fornecer uma boa precisão mesmo com câmeras de baixa resolução (CARVALHO FILHO 2009). Estes métodos frequentemente utilizam técnicas de fluxo óptico, localização de marcos naturais ou artificiais, visão global e correspondências de imagens para localizar o robô no ambiente (HEINEN, 2002). Mesmo assim, podendo estabelecer uma boa precisão, estes métodos possuem suas limitações, tais como o a necessidade de marcos naturais ou artificiais e o tamanho do ambiente, que podem restringir sua utilização a somente certos ambientes ou tarefas.

2.1.3 LOCALIZAÇÃO

A localização é o processo de inferir a pose do robô dentro de um mapa (FILLIAT; MEYER, 2003). As abordagens locais para o problema de localização exigem que se tenha a posição inicial exata do robô a priori, pois elas somente são capazes de compensar os erros odométricos para manter coerente a informação da localização do robô. Ou seja, elas não conseguem recuperar a posição do robô se ela for perdida devido a falhas. As abordagens globais são mais robustas. Elas podem estimar a posição de um robô mesmo sem nenhuma informação a priori sobre sua localização. Outra vantagem das abordagens globais é a sua capacidade de se recuperar de falhas. Algumas podem até mesmo tratar um tipo especial de falha conhecido como o "problema do robô sequestrado", no qual o robô é retirado de seu local e posicionado em um outro sem que ele seja informado sobre esta manobra. Este problema é mais complexo do que a localização global sob total incerteza, pois neste caso, ao invés de não ter informações sobre sua posição, ele possui uma informação errônea. (MEZENCIO 2002). As abordagens que existem podem ser agrupadas em três paradigmas distintos: métrico, topológico e híbrido. As abordagens no paradigma topológico demonstram a conectividade de diferentes locais, em geral exibindo ambientes como uma lista de lugares significativos conectados via arcos. Por outro lado, abordagens no paradigma métrico obtêm as propriedades geométricas do ambiente do robô. O modelo híbrido permite o uso das outras abordagens de modo complementar (THRUN et al., 1998) (TOMATIS, 2001). Siegwart, Nourbakhsh e Scaramuzza (2004) afirmam que se alguém pudesse colocar um sensor de GPS (sistema de posicionamento global) preciso a um robô móvel, grande parte do problema de localização seria dispensado. O GPS iria informar o robô de sua posição exata, dentro de casa e ao ar livre, de modo que a

resposta à pergunta, "Onde estou?", estaria sempre prontamente disponível. Infelizmente, tal sensor não é atualmente prático. A rede GPS existente fornece precisão dentro de alguns metros, o que é inaceitável para a localização de robôs móveis em escala humana, bem como robôs móveis em miniatura. Além disso, as tecnologias GPS podem não funcionar em ambientes fechados ou em áreas obstruídas e, portanto, são limitadas em seu espaço de trabalho. Também não funcionariam em outros planetas devido a falta dos satélites GPS, como é o caso dos robôs enviados a Marte. (SCHNEIDER 2018). Uma solução para este problema seria usar um IPS (*Indoor Positioning System*), um sistema de posicionamento *indoor*, o qual gera uma boa estimativa de localização em ambientes fechados. Os sensores e atuadores do robô desempenham um papel primordial em todas as maneiras de se obter a localização. E por consequência da imprecisão e lacunosidade desses sensores e atuadores que a localização apresenta problemas tão difíceis. Os sensores são a entrada elementar do robô para o processo de percepção e, portanto, o grau em que os sensores podem discernir o estado do mundo é crítico. Algo que reduz o conteúdo de informação útil das leituras do sensor, o ruído do sensor, induz uma limitação na consistência dos dados do sensor. Uma solução é levar em consideração múltiplas capturas de dados, empregando fusão temporal ou fusão multissensorial para aumentar o conteúdo de informação geral dos sensores do robô (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.1.3.1 Localização por técnicas probabilísticas

Como já foi visto, em situações reais os sensores e atuadores de robôs móveis estão sujeitos a defeitos que resultam em imprecisões nas suas operações. O ruído dos sensores induz uma limitação na coerência das leituras dos sensores no mesmo estado e, como efeito, no número de bits úteis disponíveis a partir de cada captura de dados do sensor. Outra deficiência dos sensores robóticos móveis, a não exclusividade das leituras dos sensores, faz com que eles produzam pouco conteúdo de informação e pode, por exemplo, fazer com que o robô se confunda entre lugares distintos porém semelhantes, aumentando ainda mais o problema da percepção e, portanto, da localização (SCHNEIDER 2018). Ainda, os atuadores também sofrem ruídos, uma única ação realizada por um robô móvel pode levar a várias possibilidades distintas, embora, do ponto de vista do robô, o estado inicial antes da ação ser tomada seja bem conhecido. Portanto, os atuadores de robôs móveis introduzem incerteza sobre o estado futuro, o simples ato de mover tende a aumentar a incerteza de um robô móvel (SIEGWART; NOURBAKHSH;

SCARAMUZZA, 2011). Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), na odometria (apenas sensores das rodas) e no *dead reckoning* a atualização da posição é baseada em sensores proprioceptivos. O movimento do robô, detectado com encoders nas rodas ou sensores de rumo, é integrado para calcular a posição. Como os erros de medição do sensor são integrados, o erro de posição se soma ao longo do tempo. Assim, a posição tem que ser atualizada de tempos em tempos por outros métodos de localização, caso contrário, o robô não consegue garantir uma estimatição de posição precisa ao longo prazo. A razão das abordagens probabilísticas para localização de robôs móveis terem sido desenvolvidas é que os dados provenientes dos sensores do robô são afetados por erros de medição e, portanto, só podemos calcular a probabilidade de o robô estar em uma determinada posição. Esta nova área de pesquisa é chamada de robótica probabilística. A ideia chave em robótica probabilística é representar a incerteza usando teoria de probabilidade (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.1.3.1.1 Localização por Monte Carlo

Existem diversas formas de se implementar a técnica de Localização de Monte Carlo. Elas diferem principalmente em três pontos:

- A representação da crença do robô;
- Nos procedimentos de atualização da crença;
- E nos modelos probabilísticos de ação e percepção.

O algoritmo de Monte Carlo, representa a crença *Bel(s)* por um conjunto de amostras s associadas a um fator numérico de importância *p*. Esse fator indica a probabilidade de a amostra ser relevante para a determinação da posição do robô. A crença inicial é obtida gerando-se aleatoriamente N amostras da distribuição prévia $P_{(s0)}$, e atribuindo-se o fator de importância uniforme N^{-1} para cada amostra. Abaixo segue um exemplo da execução do algoritmo:

S’ = 0
Psum = 0
Enquanto Psum < Pmax faça:
    Crie uma amostra aleatória < s, p > de S de acordo com p1,..., pN
    Gere um S’ aleatório de acordo com P(s’|a, s, m)

P’ = P(o|s’,m)

Adicione < s’, p’ > a S’

Psum = Psum + p’

Fim-enquanto

Normalize os fatores de importância p em S’

Retorne S’

Leituras de sensores *o* e ações *a* são processadas em pares. Monte Carlo constrói uma nova crença repetindo a seguinte sequência de ”suposições”: primeiro, um estado aleatório *s* é gerado a partir da crença atual, sob consideração dos fatores de importância. Para aquele estado *s*, Monte Carlo supõe um novo estado s de acordo com o modelo de ação *P(s‘|a, s, m)* no qual *m* contém os dados do mapa. O fator de importância para esse novo estado *s* recebe um valor proporcional à consistência perceptual desse estado, como medido por *P(o|s, m)*. A nova amostra, juntamente com seu fator de importância, é memorizada e o laço básico é repetido. A geração de amostras é finalizada quando a soma total dos fatores de importância exceder um limite P_{max}, ou quando o próximo par < *o, a* > chega. Finalmente, os fatores de importância são normalizados e o método de Monte Carlo retorna o novo conjunto de amostras definido como a crença corrente. Uma desvantagem desse método é a de que ele possui uma tendência a descartar hipóteses corretas e às vezes não se recupera de falhas de localização global (não resolvendo o problema do robô sequestrado). Felizmente, essa deficiência pode ser facilmente resolvida adicionando amostras aleatórias, matematicamente justificadas, assumindo que o robô pode ser sequestrado com uma pequena probabilidade. Em contrapartida, ele possui uma série de vantagens sobre os outros métodos, dentre elas como citados por SCATENA (2002):

• Extremamente eficiente, pois a computação tem seu foco em regiões de alta probabilidade.

• Mais fácil de implementar do que as abordagens de Localização de Markov alternativas.

• Robusto a ruídos e a mudanças que afetem a disponibilidade de recursos computacionais.

• Capacidade de se adaptar aos recursos. Adaptando o número de amostras, a qualidade da solução depende dos recursos computacionais disponíveis. Ao trocar o processador do robô

por um mais rápido, Monte Carlo pode utilizar estes recursos adicionais sem alterações no código.

2.1.4 LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS

É uma tarefa que consiste em estimar a posição de um robô em seu ambiente por meio de informações de seus sensores. O mapeamento robótico aborda o problema de aquisição de modelos espaciais de ambientes físicos através de robôs móveis e é comumente tratado como um dos problemas mais importantes na busca da construção de robôs móveis verdadeiramente autônomos (THRUN et al., 2002). O processo de mapeamento de ambientes pode ser dividido em duas tarefas: síntese do mapa e exploração. A síntese do mapa é uma tarefa passiva do ponto de vista do controle. Ela consiste em dadas as observações do ambiente, construir uma representação do mesmo. A exploração é um processo ativo que visa controlar o robô de forma que ele visualize com seus sensores todas as partes do ambiente (MEZENCIO 2002). O problema de representar o ambiente em que o robô se move é dual ao problema de representar a posição ou posições possíveis, do robô no ambiente. As decisões tomadas em relação à representação ambiental podem ter impacto nas opções disponíveis para a representação da posição do robô. Muitas vezes a fidelidade da representação de posição é limitada pela fidelidade do mapa (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Como já mencionado antes, para que um robô possa se locomover de forma autônoma em ambientes não estruturados é necessário que ele saiba sua localização a cada instante. Isso pode ser alcançado utilizando os métodos descritos na seção anterior. Porém, esses métodos requerem a existência de um mapa do ambiente, que pode ser difícil de se construir manualmente. Nessa seção serão expostos métodos de mapeamento que, a partir dos dados providos pelos sensores do robô consigam gerar um mapa. Conforme Thrun (1998) com relação a mapas para uso em ambientes internos existem dois grandes tipos: mapas baseados em grades de ocupação (“grid-based”) e mapas topológicos. Mapas baseados em grades são formas de mapas métricos, quantitativos, onde medidas precisas do ambiente são representadas. Nos mapas baseados em grades onde: $x(e)$ e $y(e)$ é o sistema de coordenadas global; $x(r)$ e $y(r)$ é o sistema de coordenadas local do robô. Thrun (1998) menciona que apesar de serem fáceis de construir e manter, estes mapas representam dificuldades para a tarefa de planejamento da trajetória em ambientes maiores, conforme indicado à frente. Mapas topológicos são tipicamente baseados em marcos de localização, ao invés de estarem relacionados a medidas precisas (Bekey, 2005). O ambiente é

segmentado em regiões distintas e de interesse, denominadas nós topológicos, sendo que nós adjacentes são interconectados por arcos, formando grafos. Conforme Fox et al. (1999), em um mapa topológico os nós são “lugares significativos”. Os mapas topológicos são mais compactos que os mapas métricos baseados em grade (Bekey, 2005). Thrun (1998) faz menção da vantagem deste tipo de mapa não precisar depender de informações métricas precisas e de permitir que o planejamento de rotas seja feito com mais eficiência. Entretanto, o autor menciona o problema da ambiguidade que ocorre no uso de mapas topológicos, sobretudo em ambientes maiores. Em determinadas situações pode ficar difícil a determinação da localização do robô, a partir dos dados de percepção do robô, para dois locais que sejam parecidos. Existem também modelos híbridos que combinam os dois tipos de mapas. (Thrun,1998).

O problema mais fundamental em robótica, a localização e mapeamento simultâneo (THRUN 2005) é tratado nesta seção. Comumente abreviado por SLAM - Simultaneous Localization and Mapping, O problema SLAM surge quando o robô não tem acesso ao mapa do ambiente, ou não tem acesso a sua posição. Ao invés disso, tudo que é fornecido a ele são medidas $z_{1:t}$ e controles $u_{1:t}$. O termo localização e mapeamento simultâneo descreve o problema resultante: No SLAM, o robô adquire o mapa do ambiente enquanto simultaneamente se localiza em relação a esse mapa. O objetivo do SLAM é recuperar o trajeto do robô e o mapa do ambiente usando apenas os dados coletados por seus sensores proprioceptivos e exteroceptivos. Esses dados são tipicamente o deslocamento do robô estimado a partir da odometria e características (por exemplo, cantos, linhas, planos) extraídas de um laser, sensor ultrassónico ou de imagens de câmeras (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). A dificuldade do SLAM se dá pelo fato de ambas, trajetória estimada e características extraídas serem corrompidas por ruído. Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), o problema geral de construção de mapas é análogo ao problema do ovo e da galinha. Para localização, o robô precisa saber onde as características estão, enquanto para mapeamento, o robô precisa saber onde ele está no mapa. Nesta seção, é apresentada uma definição matemática do problema de SLAM, e é descrito o método desenvolvido utilizado neste trabalho para solucionar o problema de SLAM, por mais que haja outros como: EKF-SLAM e SLAM com filtro de partículas.

2.1.4.1 Definição matemática de SLAM

As terminologias e a definição matemática de SLAM são apresentadas por Siegwart, Nourbakhsh e Scaramuzza (2011), a pose de um robô em dado instante *t* é dada por x_t. Assim, a trajetória do robô é definida por:

$$X_T = \{ x_0, x_1, x_2, \dots, x_T \} \tag{5}$$

Onde T pode ser infinito. Em SLAM, assume-se que a localização inicial do robô x_0 é distinta, enquanto as outras posições são desconhecidas. O movimento do robô é calculado durante os tempos $t-1$ *e* t. Tais dados podem ser obtidos de sensores proprioceptivos (por exemplo, dos encoders das rodas do robô) ou das entradas de controle enviadas aos motores. A sequência de movimentos relativos do robô pode então ser escrita como:

$$U_T = \{ u_0, u_1, u_2, \dots, u_T \} \tag{6}$$

M representa o mapa verdadeiro do ambiente:

$$M = \{m_0, m_1, m_2, \dots, m_T\} \tag{7}$$

Sendo m_i, $i = 0 \cdots n-1$, vetores exibindo a posição dos pontos de referência no mapa, que podem ser pontos, linhas, planos ou qualquer tipo de propriedade de alto nível (por exemplo, portas). Para efeito de simplificar, o mapa é considerado estático. Enfim, considerando que o robô faz uma medida a cada tempo, podemos supor a sequência de observações de pontos de referência no quadro de referência do sensor conectado ao robô:

$$Z_T = \{ z_0, z_1, z_2, \dots, z_T \} \tag{8}$$

Por exemplo, se o robô é equipado com uma câmera, a observação z_i pode ser um vetor exibindo as coordenadas de um canto ou as de uma linha na imagem. Se, em vez disso, o robô é equipado com um sensor de laser 2D, esse vetor pode representar a posição de um canto ou uma linha no quadro do sensor de laser. De acordo com terminologia definida, podemos

agora demonstrar o SLAM como o problema de recuperar um modelo do mapa *M* e a trajetória do robô X_T a partir da odometria U_T e as observações Z_T (SCHNEIDER 2018). Na literatura, diferimos entre o problema completo do SLAM e o problema do SLAM online. O problema de SLAM completo consiste em estimar a próxima probabilidade através de X_T e *M* dos dados, isto é:

$$p(X_T, M \mid Z_T, U_T) \tag{9}$$

O problema de SLAM online, por sua vez, consiste em estimar a próxima probabilidade da articulação através de x_T e *M* dos dados, logo:

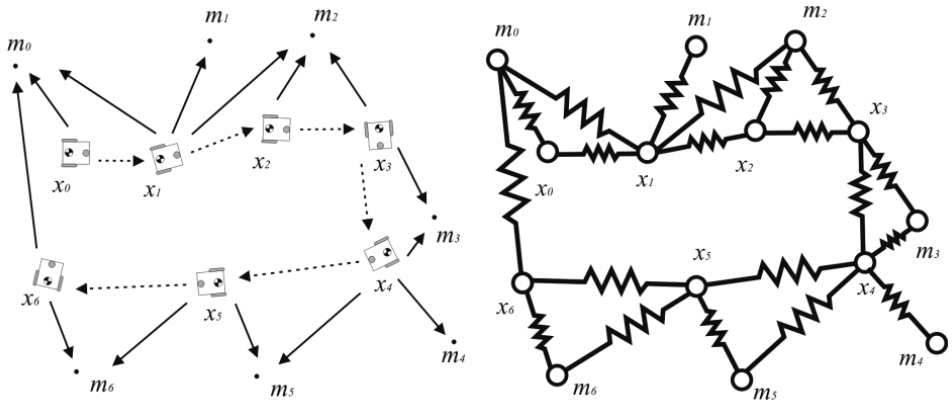
$$p(x_T, M \mid Z_T, U_T) \tag{10}$$

Portanto, o problema de SLAM completo tenta obter toda a trajetória do robô X_T, enquanto o problema de SLAM online tenta estimar apenas a pose atual do robô x_T.

2.1.4.2 SLAM baseado em grafos

Introduzido por Lu e Milios (1997), o SLAM baseado em grafos vem da idéia de que o problema do SLAM pode ser interpretado como um grafo esparso de nós e restrições entre nós. Os nós do grafo são as poses do robô $(x_0, x_1, \dots, x_T)$ e as *n* características no mapa $(m_0, m_1, \dots, m_{n-1})$. As restrições são a posição relativa entre poses consecutivas do robô e a posição relativa entre as poses do robô e as características observadas a partir desses locais (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). A Figura 4 ilustra um exemplo de construção de grafo em um cenário simplificado com um robô diferencial. De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), a principal propriedade do SLAM baseado em grafos é que as restrições não devem ser consideradas como restrições rígidas, mas como restrições suaves, pois é relaxando essas que podemos calcular a solução para o problema completo do SLAM, ou seja, a melhor estimativa do trajeto do robô e do mapa do ambiente. Em outras palavras, o SLAM baseado em grafos representa as localizações do robô e características como os nós de uma rede com molas. A solução SLAM pode ser encontrada computando o estado de energia mínima desta rede (SCHNEIDER 2018).

Figura 5 - Exemplo de construção de grafo. Restrições entre nós no grafo são representadas como molas



Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Existe uma grande vantagem das técnicas de SLAM baseadas em grafos sobre o EKF-SLAM. No EKF SLAM, o custo computacional e o de memória para atualizar e armazenar a matriz de covariância cresce de maneira quadrática com a quantidade de características obtidas. Por outro lado, no SLAM baseado em grafos, o tempo de atualização do grafo é constante e a memória necessária é linear com o número de características. No entanto, a otimização final do gráfico pode se tornar computacionalmente pesada se a trajetória do robô for longa. Existem algoritmos SLAM baseados em grafos que mostraram resultados impressionantes e muito bem-sucedidos com até cem milhões de recursos. No entanto, esses algoritmos tentam otimizar todo o caminho do robô e, portanto, foram implementados para trabalhar offline. Algumas das implementações online usaram abordagens de subdivisão do mapa (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.1.5 PLANEJAMENTO DE TRAJETÓRIA

Mudar de um lugar para outro é uma tarefa trivial para os humanos. Decidir como mover-se leva uma fração de segundo. Para um robô, uma tarefa tão elementar e básica é um dos principais desafios. Na robótica autônoma, o planejamento de caminhos é um dos problemas centrais (KOUBAA et al. 2018). O problema típico é encontrar um caminho para um robô, seja ele um braço robótico ou um robô móvel, de uma posição inicial para uma posição objetivo com segurança. O problema consiste em encontrar um caminho de uma posição inicial para uma posição alvo. Esse problema é tratado de várias maneiras na literatura, dependendo

do modelo de ambiente, o tipo de robô, a natureza da aplicação, etc. Uma navegação eficiente por robôs móveis precisa de um algoritmo de planejamento de caminho eficiente, pois a qualidade do caminho gerado afeta a aplicação. Normalmente, a minimização da distância percorrida é o principal objetivo do processo de navegação, pois influencia as outras métricas, como o tempo de processamento e o consumo de energia (KOUBAA et al. 2018). Mesmo antes do advento de robôs móveis acessíveis, o campo de planejamento de caminhos era fortemente estudado por causa de suas aplicações na área de robótica de manipuladores industriais. Curiosamente, o problema de planejamento de caminho para um manipulador com, por exemplo, seis graus de liberdade, é muito mais complexo do que o de um robô de driver diferencial operando em uma superfície plana. Portanto, embora possamos nos inspirar nas técnicas inventadas para manipulação, os algoritmos de planejamento de caminho usados por robôs móveis tendem a ser mais simples aproximações devido aos graus de liberdade muito reduzidos. Além disso, robôs industriais muitas vezes operam na velocidade mais rápida possível devido ao impacto econômico do rendimento em uma linha de fábrica, então, a dinâmica e não apenas a cinemática de seus movimentos são significativos, complicando ainda mais o planejamento e a execução do caminho. Em contraste, uma série de robôs móveis operam em velocidades tão baixas que a dinâmica raramente é considerada durante o planejamento de caminho, simplificando ainda mais a instanciação do problema do robô móvel. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

A representação do ambiente do robô pode variar de uma descrição geométrica contínua a um mapa geométrico baseado em decomposição ou até mesmo um mapa topológico. A primeira etapa de qualquer sistema de planejamento de caminho é transformar este modelo ambiental possivelmente contínuo em um mapa discreto adequado para o algoritmo de planejamento de caminho escolhido. Os planejadores de caminho diferem em como eles efetuam esta decomposição discreta. Podemos identificar três estratégias gerais de decomposição:

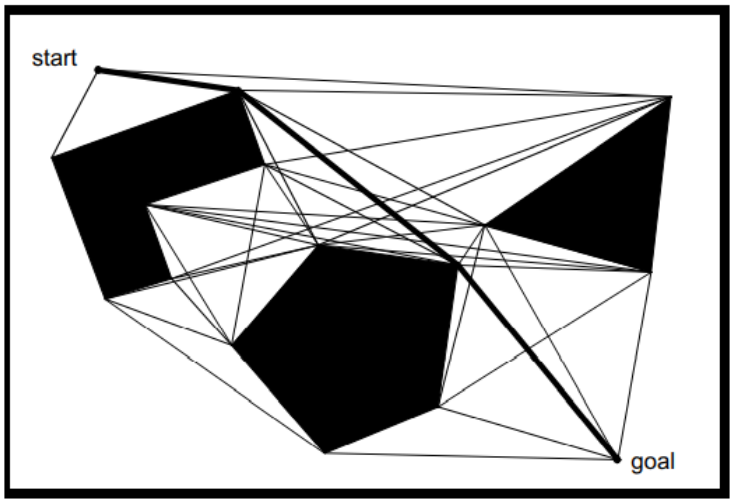
- Road map: Identificação de um conjunto de rotas dentro do espaço livre.
- Decomposição de células: Discriminação entre células livres e ocupadas.
- Campos Potenciais: Utilização de uma função matemática sobre o espaço.

2.1.5.1 Planejamento por Road Map

As abordagens por *road map* capturam a conectividade do espaço livre do robô em uma rede de 1 dimensão de curvas ou linhas, chamadas mapas de estradas. Uma vez que um roteiro (*road map*) é construído, ele é usado como uma rede de segmentos de estrada (caminho) para o planejamento do movimento do robô. O planejamento do caminho é, portanto, reduzido a conectar as posições iniciais e de objetivo do robô à rede de estradas, em seguida, procurando por uma série de estradas da posição inicial do robô até sua posição de objetivo. O roteiro é uma decomposição do espaço de configuração do robô com base especificamente na geometria do obstáculo. O desafio é construir um conjunto de estradas que, juntas, possibilitem o robô para ir a qualquer lugar em seu espaço livre, enquanto minimiza o número total de estradas. São descritas duas abordagens abaixo que alcançam este resultado com tipos dramaticamente diferentes de estradas. No caso de grafos de visibilidade os caminhos chegam o mais próximo possível dos obstáculos e resultam em caminhos de solução com mínimo comprimento. No caso dos diagramas de Voronoi os caminhos ficam o mais longe possível dos obstáculos. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Grafos de visibilidade: O grafo de visibilidade para um espaço de configuração poligonal consiste em arestas unindo todos os pares de vértices que podem se ver (incluindo as posições iniciais e objetivas como vértices também). As linhas retas desobstruídas (estradas) que unem esses vértices são obviamente as distâncias mais curtas entre eles. A tarefa do planejador de caminho é, portanto, encontrar o caminho mais curto da posição inicial até a posição objetivo ao longo das estradas definidas pelo grafo de visibilidade, como demonstrado na figura 6.

Figura 6 – Grafo de visibilidade

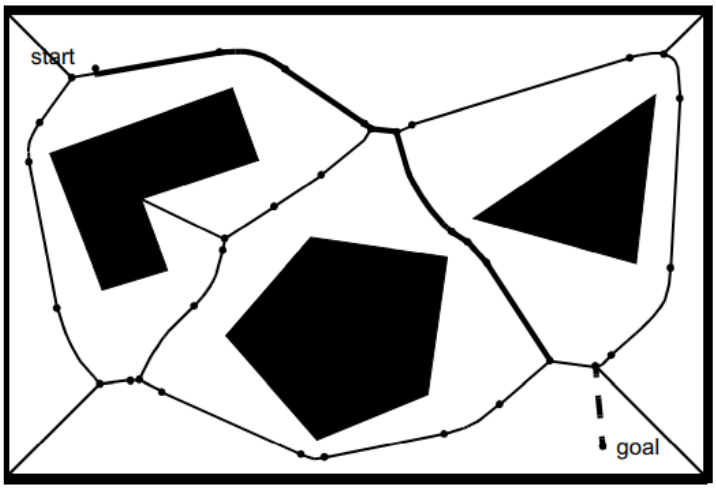


Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

O planejamento do caminho por grafo de visibilidade é moderadamente popular na robótica móvel, em parte porque a implementação é bastante simples. Particularmente quando a representação ambiental descreve objetos no ambiente como polígonos em um espaço contínuo ou discreto, a pesquisa no grafo de visibilidade pode empregar as descrições dos polígonos de obstáculo prontamente. Porém o tamanho da representação e o número de arestas e nós aumentam com o número de polígonos de obstáculo. Portanto, o método é extremamente rápido e eficiente em ambientes esparsos, mas podem ser lentos e ineficientes em comparação com outras técnicas quando usadas em ambientes densamente povoados. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Diagrama de Voronoi: Em contraste com a abordagem do grafo de visibilidade, um diagrama de Voronoi é um método completo do roteiro (*road map*) que tende a maximizar a distância entre o robô e obstáculos no mapa, como podemos ver pela figura 7. Para cada ponto no espaço livre, é calculada sua distância para o obstáculo mais próximo.

Figura 7 – Diagrama de voronoi



Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

Quando os obstáculos do espaço de configuração são polígonos, o diagrama de Voronoi consiste em segmentos retos e parabólicos. Algoritmos que encontram caminhos no roteiro de Voronoi são completos, assim como métodos de gráfico de visibilidade, porque a existência de um caminho no espaço livre implica a existência de um no diagrama de Voronoi também (ou seja, ambos os métodos são completos). No entanto, o caminho no diagrama de Voronoi geralmente está longe de ser ideal no sentido de comprimento total do caminho.

O diagrama de Voronoi tem uma fraqueza importante no caso de sensores de localização de alcance limitado. Uma vez que este algoritmo de planejamento de caminho maximiza a distância entre o robô e objetos no ambiente, qualquer sensor de curto alcance no robô estará em perigo de deixar de detectar seus arredores. Se esses sensores de curto alcance forem usados para localização, então o caminho escolhido será muito ruim do ponto de vista da localização. Por outro lado, o método do grafo de visibilidade pode ser projetado para manter o robô tão próximo quanto desejado dos objetos no mapa. Há, no entanto, uma vantagem sutil importante que o método do diagrama de Voronoi tem sobre a maioria das outras técnicas de prevenção de obstáculos: executabilidade. Dado um determinado caminho via planejamento do diagrama de Voronoi, um robô com sensores de alcance, como um laser ou ultrassônicos, podem seguir uma vantagem de Voronoi no mundo físico usando regras de controle simples que correspondem aos usadas para criar o diagrama de Voronoi: o robô maximiza as leituras de mínimos locais em seus valores de sensor. Este sistema de controle irá naturalmente manter o robô conectado as bordas de Voronoi, para que o movimento baseado em Voronoi possa mitigar a imprecisão do *encoder*. Esta propriedade física interessante do diagrama de Voronoi foi usada

para conduzir o mapeamento automático de um ambiente, encontrando e movendo-se nas bordas de Voronoi desconhecidas, então construindo um mapa Voronoi consistente do ambiente (Chong et al. 1997).

2.1.5.2 Planejamento por decomposição de células

A ideia por trás da decomposição celular é discriminar entre áreas geométricas, ou células, que são livres e áreas ocupadas por objetos. O algoritmo básico de planejamento do caminho de decomposição da célula pode ser resumido da seguinte forma:

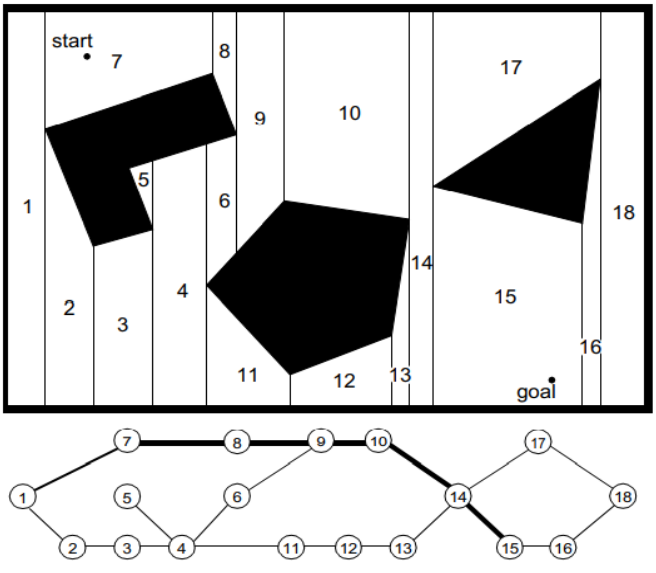
Divida o mapa em regiões simples e conectadas chamadas de "células".

Determine quais células abertas são adjacentes e construa um "grafo de conectividade".

Encontre as células nas quais as configurações iniciais e de objetivo se encontram e procure um caminho no grafo de conectividade para unir a célula inicial e objetivo.

A partir da sequência de células encontradas com um algoritmo de busca apropriado, calcule um caminho dentro de cada célula, por exemplo, passando pelos pontos médios dos limites da célula ou por uma sequência de movimentos de acompanhamento de parede e movimentos ao longo de linhas retas, tal como na figura 8.

Figura 8 – Decomposição de células



Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

Um aspecto importante dos métodos de decomposição de células é a colocação dos limites entre as células. Se os limites são colocados em função da estrutura do ambiente, de modo que a decomposição seja sem perdas, o método é denominado decomposição celular exata. Se a decomposição resultar em uma aproximação do mapa real, o sistema é denominado decomposição celular aproximada.

Decomposição celular exata: Pela figura 8 vemos que representa a decomposição celular exata, em que o limite das células é baseado na criticidade geométrica. As células resultantes são completamente livres ou ocupadas e, portanto, o planejamento do caminho na rede está completo, como os métodos baseados em roteiro (*road map*) acima. A abstração básica por trás de tal decomposição é que a posição particular do robô dentro de cada célula do espaço livre não importa; o que importa é a capacidade do robô de atravessar cada célula livre para células livres adjacentes. A principal desvantagem da decomposição celular exata é que o número de células e, portanto, a eficiência computacional do planejamento geral do caminho depende da densidade e da complexidade dos objetos no ambiente, assim como acontece com os sistemas baseados em mapas de estradas. A vantagem chave é o resultado dessa mesma correlação. Em ambientes extremamente esparsos, o número de células será pequeno, mesmo que o tamanho geométrico do ambiente seja muito grande. Assim, a representação será eficiente no caso de ambientes grandes e esparsos. Na prática, devido às complexidades na implementação, a decomposição exata da célula é uma técnica usada raramente em aplicações de robôs móveis, embora permaneça uma sólida escolha quando uma representação sem perdas é altamente desejável, por exemplo, para preservar a integridade totalmente. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Decomposição celular aproximada: Em contraste, a decomposição celular aproximada é uma das técnicas mais populares para o planejamento do caminho do robô móvel. Isso se deve em parte à popularidade das representações ambientais baseadas em grades. Essas representações baseadas em grade são próprias decomposições de tamanho de grade fixas e, portanto, são idênticas a uma célula aproximada de decomposição do meio ambiente. A forma mais popular disso é a decomposição de célula de tamanho fixo. O tamanho da célula não depende dos objetos particulares em um ambiente em absoluto, e assim passagens estreitas podem ser perdidas devido à natureza inexata da tesselação. Na prática, isso raramente é um problema devido ao tamanho de célula muito pequeno usado (por exemplo, 5 cm de cada lado). O grande benefício da decomposição de células de tamanho fixo é a baixa complexidade computacional do planejamento de caminhos. Por exemplo, NF1, muitas vezes chamado de

grassfire, é uma técnica eficiente e simples de implementar para encontrar rotas em tais matrizes de células de tamanho fixo (LATOMBE et al. 1991). O algoritmo simplesmente emprega expansão *wavefront* da posição do objetivo para fora, marcando para cada célula sua distância para a célula objetivo (JACOBS et al. 1989). Este processo continua até que a célula correspondente a posição inicial do robô é alcançada. Neste ponto, o planejador de caminho pode estimar a distância do robô até a posição do objetivo, bem como recuperar uma trajetória de solução específica simplesmente vinculando células adjacentes e sempre mais próximas do objetivo. Dado que todo o *array* pode estar na memória, cada célula é visitada apenas uma vez ao olhar para o caminho discreto mais curto da posição inicial até a posição de objetivo. Então, a busca é linear apenas no número de células. Assim, a complexidade não depende da dispersão e densidade do ambiente, nem sobre a complexidade das formas dos objetos no ambiente. Formalmente, essa transformação de *grassfire* é simplesmente uma pesquisa *breadth-first search* implementada no espaço restrito de uma matriz de adjacência. O custo fundamental da abordagem de decomposição fixa é a memória. Para um ambiente grande, mesmo quando esparso, esta grade deve ser representada em sua totalidade. Praticamente, devido à queda do custo da memória computacional, esta desvantagem foi mitigada nos últimos anos. Em contraste com o método exato de decomposição celular, a abordagem aproximada pode sacrificar a completude, mas é matematicamente menos envolvente e, portanto, mais fácil de implementar. Em contraste com a decomposição de células de tamanho fixo, a decomposição de células de tamanho variável se adaptará à complexidade do ambiente e, portanto, ambientes esparsos conterão adequadamente menos células, consumindo dramaticamente menos memória. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

2.1.5.3 Planejamento por campos potenciais

O planejamento do caminho do campo potencial cria um campo, ou gradiente, através do mapa do robô que direciona o robô para a posição de objetivo de várias posições anteriores como descrito por Latombe (1991). Esta abordagem foi originalmente inventada para o planejamento da trajetória de um robô manipulador e é usado com frequência e em muitas variantes na comunidade de robótica móvel. O método de campo potencial trata o robô como um ponto sob a influência de um campo potencial artificial *U(q)*. O robô se move seguindo o campo, assim como uma bola rolaria colina abaixo. O objetivo (um mínimo neste espaço) atua como uma força atrativa sobre o robô e os obstáculos atuam como picos, ou forças repulsivas.

Tal campo potencial artificial suavemente orienta o robô em direção ao objetivo, evitando simultaneamente obstáculos conhecidos. Se novos obstáculos aparecerem durante o movimento do robô, pode-se atualizar o campo potencial para integrar esta nova informação.

2.1.5.4 Algoritmos de planejamento de trajetória

Existem vários algoritmos de planejamento de trajetória como djikstra, A*, entre outros, porém aqui só será apresentado o que é utilizado no sistema desenvolvido. No ROS estão disponíveis tanto o algoritmo *Trajectory Rollout* quanto o *Dynamic Window Approach* sendo este segundo o escolhido por questões de eficiencia.

2.1.5.4.1 DWA (*Dynamic Window Approach*)

A abordagem de janela dinâmica (*Dynamic Window Approach*), é especialmente projetada para lidar com as restrições impostas por velocidades e acelerações limitadas, pois é derivada diretamente da dinâmica de movimento de robôs móveis. Em suma, a abordagem considera periodicamente apenas um curto intervalo de tempo ao calcular o próximo comando de direção para evitar a enorme complexidade do problema geral de planejamento de movimento. A aproximação de trajetórias durante tal intervalo de tempo por curvaturas circulares resulta em um espaço de busca bidimensional de velocidades de translação e rotação. Este espaço de busca é reduzido às velocidades admissíveis, permitindo que o robô pare com segurança. Devido às acelerações limitadas dos motores, uma outra restrição é imposta às velocidades: o robô considera apenas as velocidades que podem ser alcançadas no próximo intervalo de tempo. Essas velocidades formam a janela dinâmica que está centrada em torno das velocidades atuais do robô no espaço de velocidade. Entre as velocidades admissíveis dentro da janela dinâmica, a combinação de velocidade de translação e rotação é escolhida pela maximização de uma função objetivo. A função objetivo inclui uma medida de progresso em direção ao local do objetivo, a velocidade de avanço do robô e a distância até o próximo obstáculo na trajetória. A combinação de todos os objetivos leva a uma estratégia de prevenção de colisões muito robusta e elegante. A abordagem difere das abordagens anteriores porque: é derivada diretamente da dinâmica de movimento de um robô móvel, portanto, leva a inércia do robô em consideração, o que é particularmente importante se um robô com limites de torque viaja em alta velocidade. Esta abordagem é particularmente útil para robôs que viajam em

velocidades ainda mais altas e para robôs de baixo custo com torques de motor limitados, para os quais as restrições impostas pela dinâmica de movimento são ainda mais imperativas.

As abordagens de prevenção de colisão para robôs móveis podem ser divididas em duas categorias: global e local. As técnicas globais, como roteiro, decomposição de células e métodos de campo potencial geralmente assumem que um modelo completo do ambiente do robô está disponível. A vantagem das abordagens globais reside no fato de que uma trajetória completa do ponto de partida ao ponto de destino pode ser calculada off-line. No entanto, as abordagens globais não são adequadas para evitar obstáculos rapidamente. Sua força é o planejamento de caminhos globais. Além disso, esses métodos têm se mostrado problemáticos quando o modelo global é impreciso ou simplesmente não está disponível, como é normalmente o caso na maioria dos ambientes internos. Uma segunda desvantagem dos métodos globais é sua lentidão devido à complexidade inerente do planejamento do movimento do robô (SCHWARTZ et al. 1987). Isso é particularmente problemático se o modelo mundial subjacente muda instantaneamente, devido à necessidade de ajustes repetidos do plano global. Nesses casos, o planejamento em um modelo global geralmente é muito caro para ser feito repetidamente. As abordagens locais ou reativas, por outro lado, usam apenas uma pequena fração do modelo global para gerar o controle do robô. Isso traz a desvantagem óbvia de que eles não podem produzir soluções ótimas. As abordagens locais são facilmente presas em mínimos locais (como configurações de obstáculos em forma de U). No entanto, a principal vantagem das técnicas locais sobre as globais reside em sua baixa complexidade computacional, o que é particularmente importante quando o modelo global é atualizado com frequência com base em informações do sensor. Na abordagem de janela dinâmica a busca por comandos que controlam o robô é realizada diretamente no espaço de velocidades. A dinâmica do robô é incorporada ao método reduzindo o espaço de busca àquelas velocidades que são alcançáveis sob as restrições dinâmicas. Além desta restrição, apenas velocidades que são consideradas seguras em relação aos obstáculos são executadas. Essa poda do espaço de busca é feita na primeira etapa do algoritmo. Na segunda etapa, a velocidade que maximiza a função objetivo é escolhida entre as velocidades restantes. O algoritmo é executado da seguinte maneira:

1. Espaço de busca: O espaço de busca das velocidades possíveis é reduzido em três etapas:

(a) Trajetórias circulares: a abordagem de janela dinâmica considera apenas trajetórias circulares (curvaturas) determinadas exclusivamente por pares (v,ω) de velocidades de translação e rotação. Isso resulta em um espaço de busca de velocidade bidimensional.

(b) Velocidades admissíveis: A restrição às velocidades admissíveis garante que apenas trajetórias seguras sejam consideradas. Um par (v,ω) é considerado admissível, se o robô for capaz de parar antes de atingir o obstáculo mais próximo na curvatura correspondente.

(c) Janela dinâmica: A janela dinâmica restringe as velocidades admissíveis àquelas que podem ser alcançadas em um curto intervalo de tempo devido às acelerações limitadas do robô.

2. Otimização: A função objetivo:

$$G(v,\omega) = \sigma(\alpha.heading(v,\omega) + \beta.dist(v,\omega) + \gamma.vel(v,\omega)) \quad (11)$$

é maximizada com relação à posição atual e orientação do robô, esta função utiliza os seguintes aspectos:

(a) Título alvo: *heading* é uma medida de progresso em direção ao local da meta. É máximo se o robô se mover diretamente em direção ao alvo.

(b) Liberação: *dist* é a distância até o obstáculo mais próximo na trajetória. Quanto menor for a distância até um obstáculo, maior será o desejo do robô de se mover em torno dele.

(c) Velocidade: *vel* é a velocidade de avanço do robô e suporta movimentos rápidos. A função σ suaviza a soma ponderada dos três componentes e resulta em mais afastamento lateral dos obstáculos.

Em princípio, a abordagem proposta assume apenas informações geométricas sobre a localização relativa dos obstáculos. Portanto, é adequado para sensores de proximidade como transdutores ultrassônicos ou como *laser scanners* e até câmeras. Conhecendo a geometria do robô e o ângulo de sua câmera, as informações dos pixels são convertidas em informações de proximidade. No entanto, a estimativa de proximidade resultante só é precisa se um obstáculo tocar o chão. Obstáculos em diferentes alturas levam a uma superestimativa da distância, o que pode fazer com que o robô colida.

2.1.6 EXPLORAÇÃO

Exploração é o processo de seleção de pontos-alvo que produzem a maior contribuição para uma função de ganho de informação específica em um ambiente inicialmente desconhecido (TOPIWALA et al. 2018). A exploração de um ambiente desconhecido é um problema fundamental no campo da robótica móvel autônoma que trata da exploração de áreas desconhecidas enquanto cria um mapa do ambiente. Normalmente o operador mapeia o ambiente e esse mapa é usado pelo robô para subsequente navegação, evitando obstáculos. A exploração tem o potencial para remover o ser humano do processo para gerar um mapa de um ambiente desconhecido. Uma abordagem concisa e consistente foi proposta por Yamauchi et al (1997) com uma questão central na exploração: Dado a conhecimento atual sobre o espaço, para onde devemos mandar o robô para obter o máximo de informações? Isso pode ser respondido pelo conceito de fronteiras.

A exploração baseada na fronteira é a técnica mais comum relacionada à exploração, em que fronteiras são regiões entre espaço conhecido e espaço inexplorado. Ao mudar para uma nova fronteira, podemos continuar construindo o mapa do ambiente, até que não haja novas fronteiras a serem detectadas. (USLU et al. 2015). Existem outras técnicas baseadas em fronteiras como *Wavefront Frontier Detector* (WFD)(VERBIEST et al. 2015) e detecção rápida de fronteiras (KEIDAR et al. 2012).

2.2 CONCEITOS RELACIONADOS A VISÃO COMPUTACIONAL

2.2.1 Descritores e detectores de características

Uma característica (*feature*) é uma informação visual relevante para resolver o problema computacional relacionada a uma determinada tarefa. Podem ser estruturas específicas na imagem, como pontos, arestas ou objetos. Os detectores de características são algoritmos que obtém uma imagem e geram descritores de características ou vetores de características. Os descritores de características codificam informações em uma série de números e agem como uma espécie de impressão digital numérica que pode ser usada para diferenciar um recurso de outro. Os detectores de características devem fornecer regiões que são usadas para o cálculo dos descritores, extraindo as características da imagem. Uma vez que foram obtidas, estas características podem ser extraídas. O extrator calcula um descritor dos pixels em torno de cada ponto de interesse (TUYTELAARS; MIKOLAJCZYK, 2007).

Segundo Mikolajczyk e Schmid (2005), diversas técnicas para descrever regiões de interesse em imagens tem sido desenvolvidas. O descritor mais simples é um vetor de pixels da imagem. Há inúmeros tipos e formatos de descritores e detectores de características, mas por motivos de concisão serão apresentados os seguintes pois residem como opções para a detecção de características do algoritmo de SLAM utilizado, os quais são amplamente utilizados devido à sua fácil disponibilidade, bom desempenho de detecção e correspondência e um custo computacional relativamente pequeno:

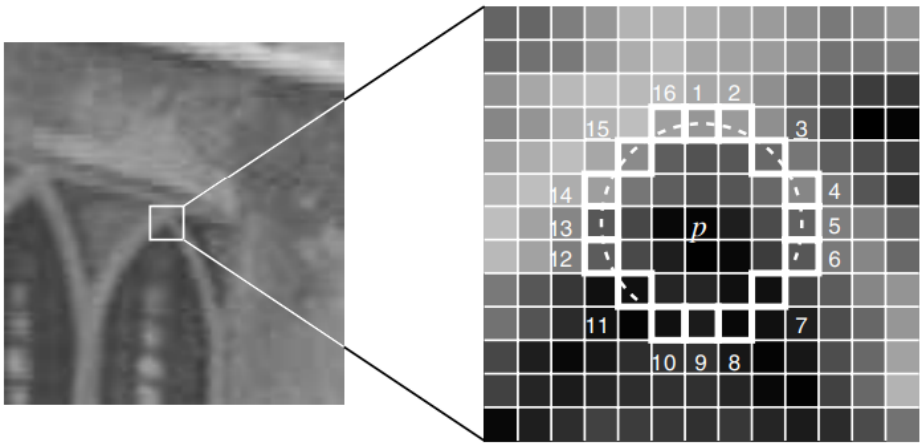
2.2.1.1 FAST

Claramente, um detector de alta velocidade é de uso limitado se os recursos produzidos forem inadequados para o processamento posterior. Em particular, a mesma cena vista de duas posições diferentes deve produzir recursos que correspondem aos mesmos locais 3D do mundo real (Schmid et al. 2000). Os FAST *features* mostram que, apesar de ser construído principalmente para velocidade, o detector supera significativamente os detectores de recursos existentes. A detecção de cantos é usada como a primeira etapa de muitas tarefas de visão, como rastreamento, SLAM (localização e mapeamento simultâneos), localização, correspondência de imagens e reconhecimento. Uma borda (geralmente uma mudança gradual na intensidade) em uma imagem corresponde ao limite entre duas regiões. Nos cantos das regiões, esse limite muda de direção rapidamente. Várias técnicas foram desenvolvidas que envolvem a detecção e o encadeamento de arestas com o objetivo de encontrar cantos na aresta encadeada, encontrando máximos de curvatura (Langridge et al. 1987) mudança de direção (Haralick et al. 1993) ou mudança na aparência (Cooper et al. 1991). Outra classe de detectores de canto funciona examinando um pequeno fragmento de uma imagem para ver se “parece” com um canto. Como as segundas derivadas não são calculadas, uma etapa de redução de ruído (como suavização gaussiana) não é necessária. Consequentemente, esses detectores de canto são computacionalmente eficientes, uma vez que apenas um pequeno número de pixels é examinado para cada canto detectado. Um corolário disso é que eles tendem a ter um desempenho insatisfatório em imagens apenas com recursos de grande escala, como imagens desfocadas. O detector de cantos FAST pertence a esta categoria.

O processo opera em duas etapas. Para construir um detector de canto para um determinado n, primeiro, os cantos são detectados a partir de um conjunto de imagens (de preferência do domínio de aplicação alvo) usando o critério de teste de segmento para n e um

limite conveniente. Ele usa um algoritmo lento que para cada pixel simplesmente testa todos os 16 locais no círculo ao redor dele, como podemos ver na figura 9.

Figura 9 - Detecção de canto de teste de segmento de ponto em imagem.



Fonte: (Rosten et al. 2006)

Para cada localização no círculo x ∈ {1..16}, o pixel relativo a posição p (denotado por p → x) pode estar em um desses três estados:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \ \ (darker) \\ s, & I_p - t < I_{p\to x} < I_p + t \ \ (similar) \\ b, & I_p + t \le I_{p\to x} \ \ (brighter) \end{cases} \tag{12}$$

Escolher um *x* e computar $S_{p\to x}$ para todas $p \in P$ (o *set* de todos os pixels nas imagens de treinamento) partições *P* em três *subsets*, *Pd, Ps, Pb,* aonde cada *p* é designado para $P_{S_{p\to x}}$.

Seja K_p. uma variável booleana verdadeira se p for um canto e falso caso contrário. O estágio 2 emprega o algoritmo usado em ID3 (Quinlan 1986) e começa selecionando o *x* que produz a maior parte das informações sobre se o pixel candidato é um canto, medido pela entropia de K_p. A entropia de *K* para o conjunto *P* é:

$$H(P)=(c+\bar{c})\ log_2(c+\bar{c}) - c\ log_2\ c - \bar{c}\ log_2\ c \tag{13}$$

aonde $c = \{p|K_p$ é verdadeiro$\}$(número de cantos)

e $\bar{c} = \{p|K_p$ é falso$\}$(número de não cantos)

A escolha de *x* então produz o ganho de informação:

$$H(P) - H(P_d) - H(P_s) - H(P_b) \tag{14}$$

Tendo selecionado o *x* que leva ao maior ganho de informação, o processo é então aplicado recursivamente em todos os três *subsets* i.e. x_b é selecionado para particionar P_b em $P_{b,d}$, $P_{b,s}$, $P_{b,b}$, x_s é selecionado para particionar P_s in to $P_{s,d}$, $P_{s,s}$, $P_{s,b}$ e assim em diante, onde cada *x* é escolhido para produzir o máximo de informações sobre o conjunto ao qual é aplicado. O processo termina quando a entropia de um subconjunto é zero. Isso significa que todos os p neste subconjunto têm o mesmo valor de K_p, ou seja, eles são todos cantos ou não cantos. Isso cria uma árvore de decisão que pode classificar corretamente todos os cantos vistos no conjunto de treinamento e, portanto, (em uma aproximação) incorpora corretamente as regras do detector de canto FAST escolhido. Essa árvore de decisão é então convertida em código C, criando uma longa sequência de instruções if-then-else aninhadas que são compiladas e usadas como um detector de cantos. O FAST é criado usando aprendizado de máquina para derivar um detector de canto muito rápido e de alta qualidade. Tem as seguintes vantagens:

- É muitas vezes mais rápido do que outros detectores de canto existentes.
- Altos níveis de repetibilidade sob grandes mudanças de aspecto e para diferentes tipos de recursos.

No entanto, também sofre de uma série de desvantagens:

- Não é robusto para níveis elevados de ruído.
- Pode responder a linhas de 1 pixel de largura em certos ângulos, quando a quantização do círculo perde a linha.
- Depende de um *threshold*.

2.2.1.2 BRIEF

Binary Robust Independent Elementary Features (BRIEF), apresentado por Calonder et al. (2010), é um descritor binário de uso geral robusto às classes típicas de transformações de imagem fotométricas e geométricas que pode ser concordado com detectores arbitrários. O algoritmo SIFT usa um vetor de dimensão 128 para descritores e uma vez que utiliza números

de ponto flutuante, requer aproximadamente 512 bytes. Do mesmo modo, SURF também necessita um mínimo de 256 bytes (CALONDER et al., 2010). O desvantajoso é que quanto mais memória, maior é o tempo necessário para o processo de correspondência. Os descritores podem ser comprimidos usando várias técnicas como PCA (Análise de Componentes Principais) (JOLLIFFE, 2014) e LDA (LI; YUAN, 2005). LSH (*Local Sensitive Hashing*) (KULIS; GRAUMAN, 2009) também é usado para marcar esses descritores SIFT de números de ponto flutuante para cadeias binárias. Essas sequências demonstram descritores binários que são usados para unir características usando a distância de Hamming (XOR seguido por uma contagem de bits) e podem permutar a habitual distância euclidiana (ALAHI; ORTIZ; VANDERGHEYNST, 2012). Primeiro é requerido que seja calculado o descritor completo antes do processamento adicional para a redução de dimensionalidade. BRIEF não precisa deste processo já que computa as cadeias binárias diretamente nas regiões da imagem. Os bits individuais são obtidos através da comparação das intensidades de pares sem precisar de uma fase de treinamento (CALONDER et al., 2010). Calonder et al. (2010) criam um vetor com as respostas dos testes, que são calculados após a suavização da região da imagem. Mais especificamente, define-se um t teste na região p de tamanho SxS como:

$$t(p; x, y) = \begin{cases} 1, & se\ P(\mathrm{x}) < P\ (y) \\ 0, & Caso\ Contrario \end{cases} \tag{15}$$

Onde p(x) é a intensidade do pixel na após suavização. A escolha exclusiva de um conjunto de pares de localização n_d (x, y) define um conjunto de testes binários (CALONDER et al., 2010).

$$f_{nd}(p) = \sum_{1 \le i \le nd} 2^{i-1} t(p; x_i, y_i) \tag{16}$$

Selecionando um conjunto de pares *(x,y)* BRIEF utiliza uma região de imagem suavizada. Em seguida, as comparações de intensidade de pixel são feitas nesses locais, resultando em 1 ou 0. Segundo Calonder et al. (2010), experimentos mostraram que apenas 256 bits, ou até mesmo 128 bits, muitas vezes são suficientes para obter resultados muito bons de correspondência. A escolha de um conjunto de pares *(x, y)* particulares define um conjunto de testes binários.

2.2.1.3 ORB

O método proposto baseia-se no conhecido detector de *keypoints* FAST (ROSTEN et al. 2006) e no descritor BRIEF (CALONDER, et al., 2010) por esse motivo, chamamos de ORB (*Oriented FAST and Rotated BRIEF*). Ambas as técnicas são atraentes devido ao seu bom desempenho e baixo custo. O ORB modifica o detector FAST para detectar *keypoints*, junto com um componente de orientação rápido e preciso para calcular a orientação, pois somente o FAST não calcula. Para resolver este problema, Rublee et al. (2011) utilizam uma medida de orientação de canto, a intensidade do centroide, definida por Rosin (1999) que assume que a intensidade de um canto está deslocada do seu centro, e este vetor pode ser utilizado para atribuir uma orientação. Rosin (1999) define o momento da região como:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{17}$$

Com esses momentos encontra-se o centroide:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{18}$$

A orientação da região é dada pela equação:

$$\theta = a\tan 2(m_{01} m_{10}) \tag{19}$$

Onde *atan2* retorna ângulo cuja tangente é o quociente de dois números especificados respeitando o quadrante. Para melhorar a invariância a rotação desta medida, os momentos são calculados com *x* e *y* permanecendo dentro de uma região circular de raio r (RUBLEE et al., 2011). Em relação aos descritores, ORB utiliza BRIEF. ORB utiliza a versão orientada de BRIEF (oBRIEF), para resolver o mau desempenho de BRIEF com rotação, rotacionando a direção do descritor de acordo com a orientação do keypoint (RUBLEE et al., 2011). Rublee et al. (2011) afirmam que qualquer conjunto de características de *n* testes binários no local (x_i, y_i), define a matriz 2 x *n*:

$$S = \begin{pmatrix} x_1, & \dots & , x_n \\ y_1, & \dots & , y_n \end{pmatrix} \tag{20}$$

Usando a orientação θ da região e a matriz de rotação correspondente R_θ, Rublee et al. (2011) constroem uma versão rotacionada S_θ de S:

$$S_\theta = R_\theta S \tag{21}$$

Então, a versão rotacionada de BRIEF é dada por:

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S\theta \tag{22}$$

Desta forma, Rublee et al. (2011) discretizam o ângulo a incrementos de 12 graus, e constrói uma tabela de pesquisa de padrões BRIEF pré-computados. O conjunto correto de pontos S_θ será usado para calcular o seu descritor, enquanto a orientação θ do ponto de interesse é consistente em todos os pontos de vista. Assim, ao contrário BRIEF, ORB é comparativamente invariante a escala e rotação enquanto ainda emprega a eficiente distância de Hamming no processo de combinação (RUBLEE et al., 2011).

2.2.2 Odometria visual

A odometria visual é um método que permite a um agente estimar sua localização espacial (pose) utilizando somente informações visuais obtidas por câmeras de forma incremental, através da análise das mudanças do movimento induzidas nas imagens das câmeras a bordo (IKEDO et al. 2017). O uso de odometria visual em aplicações com robôs oferece uma opção atraente em relação a outros métodos de odometria por razões como o preço, acessibilidade, precisão, e por não possuir desvantagens dos demais métodos, como o drift de encoders em rodas (IKEDO et al. 2017).

Existem diferentes métodos de odometria visual dependendo de como é feita a captura dos dados: Monocular, Stereo e Omnidirecional. Como principal diferença temos que a monocular é mais simples porem não possui a informação de profundidade. Assim como também diferentes classificações como: Método Indireto e Método Direto, são classificações ligadas a odometria visual e referentes a abordagem de processamento da informação. Para se obter uma estimativa de pose e movimento no espaço tridimensional por meio de imagens, procura-se computar um estimador X por meio de um modelo probabilístico que recebe as

medidas Y, extraídas das imagens de entrada (ENGEL et al. 2018). O método indireto primeiro pré-processa a imagem recebida pelo sensor da câmera gerando medidas intermediarias, que são então usadas pelo modelo probabilístico como entrada para fazer a estimativa. Como as medidas intermediarias feitas são medidas geométricas, como pontos e vetores, diz-se que o método indireto trabalha com a otimização do erro geométrico; já o método direto utiliza diretamente as informações vindas da imagem da câmera, o qual realiza uma otimização fotométrica. Também temos a classificação entre métodos densos e esparsos no que diz respeito a quantidade de pontos que foram amostrados. Métodos densos tentam em geral capturar o máximo de pontos possíveis enquanto métodos esparsos obtém uma quantidade limitada de pontos sem ter uma relação pré-estabelecida, ou seja, pontos independentes entre si.

Num sistema utilizando odometria visual a estimação da pose é realizada com base em um algoritmo. Para que o algoritmo produza os melhores resultados possíveis, as seguintes condições têm que ser verificadas: suficiente iluminação no ambiente, e a cena tem de permanecer estática e conter textura suficiente para permitir a extração do movimento aparente. Além disso, deve ser garantido que os frames capturados de modo contínuo contenham suficiente sobreposição da cena (DIAS 2013). Uma vez que a odometria visual calcula a trajetória da câmera de forma incremental, pose após pose, os erros introduzidos por cada novo movimento, frame após frame, vão-se acumulando ao longo do tempo. Isto gera um desvio da trajetória estimada em relação à trajetória real. Para algumas aplicações é de extrema importância manter o desvio o mais pequeno possível, o que pode ser feito através da otimização local das últimas *m* poses da câmera. Esta ação denomina-se *windowed bundle adjustment* como proposto por FRAUNDORFER et al (2010).

Outros problemas em odometria visual são os *outliers* e correspondências falsas que podem levar a poses erradas, e por conseguinte as estimativas seguintes irão conter este erro. Assim, a trajetória global estimada também irá conter este erro. Para que o movimento da câmara seja estimado com precisão é importante que os *outliers* sejam removidos de uma forma robusta, para esse efeito pode-se usar o algoritmo RANSAC (do inglês *Random Sample Consensus*)(FISHLER 1981).

2.3 ROBOT OPERATING SYSTEM

O Robot Operating System, mais conhecido como ROS, é um framework para se trabalhar com robôs que facilita a implementação de técnicas de alto nível em hardware de

baixo nível. Muito útil pelo caso de reuso de informações e desenvolvimento em grupo. Sua composição varia de drivers a algoritmos de última geração e tudo em código aberto [ROS.org] Ele fornece serviços padrões como abstração de hardware, controle de dispositivos de baixo nível e passagem de mensagens entre processos. ROS possui uma arquitetura simples, sempre haverá uma instancia em uma máquina conhecida como mestre a qual irá regular toda a comunicação entre os diferentes 'nós' (programas) do ROS. Começou a ser desenvolvido em Stanford em 2000 e após foi aperfeiçoado em 2007 pela Willow Garage.

O ROS não é um sistema operacional no sentido tradicional de gerenciamento e agendamento de processos; em vez disso, fornece uma camada de comunicação estruturada acima dos sistemas operacionais como um host de um cluster de computação heterogêneo (Quigley et al. 2009) Os objetivos do ROS podem ser resumidos como:

• *Peer-to-peer*

Um sistema criado usando o ROS consiste em vários processos, potencialmente em vários hosts diferentes, conectados em uma topologia ponto a ponto. Embora estruturas baseadas em um servidor central (CARMEN et al. 2003) também possam obter os benefícios do design de vários processos e vários hosts, um servidor de dados central é problemático se os computadores estiverem conectados em uma rede heterogênea.

• *Tools-based*

Em um esforço para gerenciar a complexidade do ROS, optou-se por um design de microkernel, onde um grande número de pequenas ferramentas são usadas para criar e executar os vários componentes do ROS, em vez de criar um ambiente monolítico de desenvolvimento e tempo de execução.

• *Multi-lingual*

Ao escrever código, muitas pessoas têm preferências por algumas linguagens de programação acima de outras. Essas preferências são o resultado de trocas pessoais entre tempo de programação, facilidade de depuração, sintaxe, eficiência de tempo de execução e uma série de outras razões, tanto técnicas quanto culturais. Por esses motivos, ROS é o projetado para ser

neutro em termos de idioma. Atualmente, o ROS suporta quatro idiomas muito diferentes: C ++, Python, Octave e LISP, com outras portas de idioma em vários estados de conclusão como demonstra Quigley et al (2009). A especificação do ROS está na camada de mensagens. A negociação e a configuração de conexão ponto a ponto ocorrem no XML-RPC, para o qual existem implementações razoáveis na maioria dos idiomas principais.

• *Thin*

Todo o desenvolvimento de drivers e algoritmos são incentivados a ocorrer em bibliotecas independentes que não dependem do ROS. O sistema de compilação do ROS realiza compilações modulares dentro da árvore do código-fonte, e seu uso do CMake facilita comparativamente a adoção dessa ideologia "*thin*". A colocação de praticamente toda a complexidade nas bibliotecas e a criação de pequenos executáveis que expõem a funcionalidade da biblioteca ao ROS permitem uma extração e reutilização mais fácil do código além da intenção original.

• *Free and Open-Source*

O código fonte completo do ROS está disponível ao público. Isso é crítico para facilitar a depuração em todos os níveis da pilha de software. O ROS é distribuído sob os termos da licença BSD, que permite o desenvolvimento de projetos não comerciais e comerciais.

Os conceitos fundamentais da implementação do ROS são nós, mensagens, tópicos e serviços. Nós são processos que executam a computação. O ROS foi projetado para ser modular em uma escala refinada: um sistema geralmente é composto por muitos nós. quando muitos nós estão em execução, é conveniente renderizar as comunicações ponto a ponto como um grafo, com processos como nós do grafo e os links ponto a ponto como arcos. Os nós se comunicam através de mensagens. (Quigley et al. 2009).

Uma mensagem é uma estrutura de dados. Os tipos primitivos padrão (inteiro, ponto flutuante, booleano etc.) são suportados, assim como matrizes de tipos e constantes primitivas. Um nó envia uma mensagem publicando-a em um determinado tópico, que é simplesmente uma string como "odometria" ou " mapa." Um nó interessado em um determinado tipo de dados se

inscreverá no tópico apropriado. Pode haver vários editores e assinantes simultâneos para um único tópico, e um único nó pode publicar e/ou assinar vários tópicos. (Quigley et al. 2009).

Embora o modelo de publicação/assinatura com base em tópicos seja um paradigma de comunicação flexível, seu esquema de roteamento de "transmissão" não é apropriado para transações síncronas, o que pode simplificar o design de alguns nós. No ROS, isso é chamado de serviço, definido por um nome de sequência e um par de mensagens: uma para a solicitação e outra para a resposta. Isso é análogo aos serviços da web, que são definidos por URIs e possuem documentos de solicitação e resposta de tipos bem definidos. (Quigley et al. 2009).

2.4 CONCLUSÃO

Esta seção apresentou uma série de conceitos fundamentais para o entendimento deste trabalho. Foram expostos conceitos que elucidam o problema de navegação robótica, especificamente nas tarefas de mapeamento, localização, planejamento de trajetória e exploração, e como esses problemas podem ser resolvidos com o auxílio de técnicas de visão computacional e robótica probabilística. Além disso, foram apresentados descritores de características locais, úteis no processo de encontrar marcações naturais que podem ser utilizadas nas fases de mapeamento e localização de robôs móveis, tambem foi apresentada uma noção do conceito de odometria visual bem como uma breve descrição do sistema operacional de robôs (ROS).

3 TRABALHOS RELACIONADOS

Com o constante desenvolvimento de técnicas de visão computacional e robótica móvel, atingimos um paradigma em que novas soluções se constituem de aprimoramentos de técnicas já desenvolvidas e bem embasadas em teoremas e experimentos como o SLAM visual. Neste capítulo serão apresentados trabalhos relacionados ao tema desta dissertação de acordo com as respectivas categorias, mostrando o desenvolvimento e apontando tendências e caminhos para evolução da área de descritores de características, odometria visual, localização, SLAM, planejamento de trajetórias, exploração e ROS. A principal vantagem de sistemas baseados em visão computacional está na grande quantidade de informação que estes disponibilizam. Porém o tempo de processamento computacional necessário para a compreensão e conversão desta grande porção de informação em dados úteis é, normalmente, bastante elevado. De forma geral, os sistemas baseados em visão utilizam algum tipo de representação simplificada, a qual foca em um tipo específico de informação.

3.1 AVALIAÇÃO DE DESCRITORES DE CARACTERÍSTICAS

Mikolajczyk e Schmid (2005) compara o desempenho de descritores calculados para regiões de interesse. O número de correspondências corretas sobre número total de correspondências, que é o *recall* é utilizado na avaliação como critério de comparação a precisão, e é realizado para diferentes transformações de imagem, comparando filtros direcionáveis (*Steerable Filters*) (FREEMAN; ADELSON et al., 1991), PCA-SIFT, invariantes diferenciais (*differential invariants*) (KE; SUKTHANKAR, 2004) (KOENDERINK; DOORN, 1987), *shape context* (BELONGIE; MALIK; PUZICHA, 2002), *spin images* (LAZEBNIK; SCHMID; PONCE, 2003), SIFT invariantes de momento (GOOL; MOONS; UNGUREANU, 1996), filtros complexos (SCHAFFALITZKY; ZISSERMAN, 2002) e correlação cruzada para tipos de regiões de interesse desiguais. Eles concluem que o desempenho dos descritores é em grande parte independente do detector e que os descritores baseados em SIFT possuem o melhor desempenho.

Um estudo comparativo entre os descritores apresentados na revisão teórica foi realizado por Dwarakanath, Eichhorn, Halvorsen e Griwodz1 (2012) o qual faz uma a análise comparativa de diferentes métodos do estado da arte para detecção e descrição de características locais em imagens. Os resultados experimentais demonstram que detectores e descritores

binários de características (ORB) e de ponto flutuante (SIFT e SURF) mostram significante diferenças de desempenho entre o os extratores de recursos em termos de precisão, tempo de execução e robustez para desfocar, distorção da lente e ruído térmico de vários níveis. Haja vista a quantidade de trabalhos nesta área, dentre esses, muitos trabalhos surgiram com objetivo de avaliar algoritmos de correspondência de pontos em imagens, por meio de diferentes estratégias e métricas avaliativas também temos o trabalho de Bekele, Teutsch e Schuchert (2013) o qual faz uma avaliação dos *keypoints* de última geração de descritores. Também temos o trabalho de Lowe (2004) apresenta um método para extrair características invariantes distintas de imagens que podem ser usadas para realizar uma correspondência confiável entre diferentes visualizações de um objeto ou cena. Similarmente temos o trabalho de van de Sande, Gevers e Snoek (2008) que estuda as propriedades de invariância dos descritores de cores em maneira estruturada. As propriedades de invariância dos descritores de cores são mostradas analiticamente usando uma invariância de propriedades em relação às transformações fotométricas. Um uso eficaz de descritores também é feito em Brito et. Al. (2016) o qual apresenta uma análise comparativa de diferentes métodos do estado da arte para detecção e descrição de características locais em imagens, com o objetivo de solucionar de forma robusta e eficiente o problema de autocalibração de câmeras.

Barbosa (2017) apresentou uma comparação entre os descritores BRIEF, BRISK, FREAK, ORB, SIFT e SURF para verificar sua aplicabilidade no problema de navegação robótica através de quatro testes: Teste local, com a maioria dos descritores obtendo o comportamento desejado; Teste global, em que todos os descritores menos o FREAK obtiveram o comportamento desejado; Teste de velocidade de correspondência, o qual obteve a correspondência mais rápida no BRIEF; Teste de dissimilaridade, em que o SIFT obteve o melhor desempenho. Em seguida, Barbosa (2017) avalia os algoritmos de localização e mapeamento propostos com os descritores BRIEF, BRISK, ORB, SIFT e SURF, através dos seguintes critérios de avaliação: Número de keyframes; Relative Pose Error (RPE); Absolute Trajectory Error (ATE). Verificou-se que todos os descritores analisados, exceto o FREAK, atendem às características necessárias para o sistema de mapeamento e localização na forma que foi proposta. Os critérios de RPE e ATE indicam que os descritores BRISK, SURF e ORB obtém erros relativamente menores do que os outros descritores.

3.2 ODOMETRIA VISUAL

Como dissertado por Dias (2013) é apresentado um algoritmo de Odometria Visual e efetuada uma avaliação do seu desempenho, que em conclusão, apresenta bons resultados e ainda pode ser melhorado com pequenos ajustes, como por exemplo o uso do algoritmo *bundle adjustment*. Na generalidade dos percursos lineares mostrou ser capaz de estimar a pose, apresentando erros menores. Também temos o trabalho de Fontes e Maia (2016) que desenvolve uma abordagem rápida e precisa de odometria visual para uma câmera RGB-D navegando em ambiente estático. O algoritmo utiliza SURF (Speeded Up Robust Features), RANSAC (Random Sample Consensus) e Mínimo Quadrado Médio para estimar a transformação rígida de seis parâmetros entre quadros sucessivos do vídeo.

Bem com o trabalho de Ikedo e Colombini (2017) que investiga o método *Direct Sparse Odometry* (DSO) de odometria visual aplicado ao contexto de robôs móveis. Propondo uma variação do método DSO temos o trabalho de Engel, Koltun e Cremers (2018) em que é proposta uma nova formulação de odometria visual esparsa direta, a qual combina um modelo probabilístico totalmente direto (minimizando um erro fotométrico) com otimização conjunta consistente de todos os parâmetros do modelo, incluindo geometria, representado como profundidade inversa em um quadro de referência e movimento da câmera. Outro trabalho que segue a mesma linha de proposição de diferentes técnicas de odometria visual como Aladem e Rawashdeh (2018) apresentando um sistema de estimativa de movimento em tempo real de baixa sobrecarga, a precisão do algoritmo supera as abordagens quadro a quadro típicas por manter um mapa local limitado, exigindo significativamente menos memória e poder computacional.

Ainda, o trabalho de Zhan et al. (2020) nos mostra um algoritmo monocular de odometria visual que aproveita a aprendizagem profunda. A maioria dos sistemas VO (*Visual Odometry*)/SLAM existentes com desempenho superior são baseados na geometria e têm que ser cuidadosamente projetado para diferentes cenários de aplicações. Além disso, a maioria dos sistemas monoculares sofre de problemas de deriva de escala. Neste trabalho, é revisado os conceitos básicos de odometria visual e explorado a maneira certa de integrar aprendizado profundo com geometria epipolar e o método *Perspective-n-Point* (PnP). Especificamente, são treinadas duas redes neurais convolucionais (CNNs) para estimar a profundidades de visão

única e fluxos ópticos com duas visões como saídas intermediárias. Com as previsões profundas, é projetado um algoritmo de odometria visual quadro a quadro simples, mas robusto que supera métodos puramente baseados em aprendizagem profunda e baseados em geometria. Mais importante ainda, o sistema não sofre do problema de desvio de escala. Experimentos extensivos no conjunto de dados KITTI mostra a robustez do sistema.

Também temos o trabalho de Xie et al. (2021) em que é proposto um algoritmo de odometria visual monocular baseado em mapa virtual-real híbrido. A ideia central é que é reprocessados os recursos do segmento de linha para gerar os pontos de correspondência de interseção virtuais, que podem ser usados para construir o mapa virtual. A introdução do mapa virtual pode melhorar a estabilidade do algoritmo de odometria visual em um ambiente de baixa textura. Especificamente, primeiro são combinados segmentos de linha para gerar pontos de correspondência de interseção virtual e, em seguida, com base nos pontos de correspondência de interseção virtual, é triangulado para obter um mapa virtual, combinado com o mapa real construído sobre os recursos de pontos comuns para formar um mapa virtual-real 3D híbrido. Finalmente, usando o mapa híbrido, a estimativa de pose de câmera contínua pode ser encontrada. Extensos resultados experimentais demonstraram a robustez e eficácia do método proposto em várias cenas de baixa textura.

3.3 LOCALIZAÇÃO

DeSouza e Kak (2002) demonstram que uma vez que em localização absoluta, a pose inicial do robô é desconhecida, o sistema de navegação deve conseguir construir uma correspondência entre as observações e as possibilidades derivadas de toda a base de dados. Devido às incertezas associadas às observações, é possível que o mesmo conjunto de observações corresponda a múltiplas possibilidades. As indeterminações resultantes na localização podem ser resolvidas por métodos como: localização de Markov (THRUN, 2000), processos de Markov parcialmente observáveis (SIMMONS; KOENIG, 1995), usando intervalos para representar incertezas (KROTKOV, 1989) (ATIYA; HAGER, 1993), filtros de Kalman com múltiplas hipóteses baseados em uma mistura de Gaussianas (COX; LEONARD, 1994), localização de Monte Carlo (BLAKE; ISARD, 1997) (DELLAERT et al., 1999) e por triangulação determinística (SUGIHARA, 1988). Nestes casos, o algoritmo de localização deve somente acompanhar as incertezas na posição do robô quando executa comandos de movimento e, quando ultrapassarem um limiar, usar os sensores para uma correção em sua posição. Em

geral, técnicas de localização probabilísticas se transformaram na abordagem preferida para representação e atualização das incertezas de posições enquanto o robô se move (DESOUZA; KAK, 2002).

Também temos o trabalho de Atiya e Hager (1993) o qual descreve um algoritmo para determinar localização do robô a partir de pontos de referência visuais. As principais vantagens desse algoritmo são o uso de uma única tolerância geométrica para descrever o erro de observação, sua capacidade de reconhecer conjuntos ambíguos de correspondências, sua capacidade de calcular limites sobre o erro na localização e execução rápida. No trabalho de Carvalho Filho et. Al. (2009) o artigo é focado no desenvolvimento de um sistema de estimação de posição e orientação baseado em técnicas de visão computacional usando correspondência de imagens. A comparação das imagens é realizada usando uma técnica de casamento de pontos baseada nas distâncias entre os pontos de interesse. Neste sentido, as distâncias entre os pontos de interesse são agrupadas em conjuntos de distâncias e as distribuições de distâncias destes conjuntos são usadas para realizar a correspondência das imagens.

Em Saurer, Fraundorfer e Pollefeys (2010) é descrita uma abordagem de localização visual para robôs móveis na qual a localização do robô é realizada como reconhecimento de local. A abordagem usa recursos visuais globais (por exemplo, descritor GIST [OLIVA; TORRALBA, 2001]) para semelhança e uma etapa de verificação geométrica usando pontos de fuga. O reconhecimento do local ocorre comparando a imagem recém capturada com as do banco de dados previamente armazenadas.

3.4 LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS (SLAM)

Aulinas et al. (2008) pesquisa técnicas no campo da Localização e mapeamento simultâneos (SLAM). Em particular, está focado nas técnicas existentes disponíveis para acelerar o processo, com o objetivo de lidar com cenários de grande escala. O principal campo de pesquisa que é investigado é algoritmos de filtragem como forma de reduzir a quantidade de dados. Segundo os autores quase todas as abordagens atuais não podem executar mapas consistentes para grandes áreas, principalmente devido ao aumento do custo computacional e devido às incertezas que se tornam grandes quando o cenário se torna maior.

Zhang e Ghosh (2000) propõe um novo esquema para construção de mapas e descrever técnicas de localização para um robô móvel equipado com um laser 2D. É proposto usar segmentos de linha como elemento básico para fins de localização e construir o mapa. Os

segmentos de linha fornecem informações geométricas consideráveis sobre a cena que também pode ser usada para obter informações de localização precisas e rápidas. É introduzido um novo conceito de segmento de linha fechada ou *Closed Line Segment* (CLS), que consiste apenas em segmentos de linha e define uma região fechada e conectada. Segmentos de linhas virtuais são desenhados, para os pontos que não adequadamente descrevem um segmento de linha nos dados do intervalo.

No trabalho de Kummerle, Grisetti e Burgard (2011) é desenvolvida uma abordagem para estimar simultaneamente um mapa de ambiente, a posição dos sensores do robô e seus parâmetros cinemáticos. O método não requer conhecimento prévio sobre o ambiente e depende apenas de palpite inicial aproximado dos parâmetros da plataforma. A abordagem proposta realiza estimativas on-line dos parâmetros e é capaz de se adaptar a alterações não estacionárias da configuração do ambiente.

Também temos o trabalho de Strom, Nenci e Stachniss (2015) o qual é considerado o problema de como selecionar pontos de vista que suportam o processo de mapeamento subjacente. É proposta uma nova abordagem que faz previsões sobre a estrutura dos ambientes em áreas inexploradas, contando com mapas adquiridos anteriormente. Esta abordagem busca encontrar semelhanças entre os atuais ambientes do robô e mapas adquiridos anteriormente armazenados em um banco de dados para prever como o ambiente pode expandir nas áreas desconhecidas. Isso permite prever possíveis futuros fechamentos do laço mais cedo. Esse conhecimento é usado na seleção de pontos para fechar loops ativamente e, assim, reduzir a incerteza do robô.

Uma avaliação das técnicas de SLAM é feita por Santos et al. (2013), a qual compara 5 algoritmos diferentes disponíveis no ROS. Entre eles o HectorSLAM, Gmapping, KartoSLAM, CoreSLAM e LagoSLAM. Com o HectorSLAM: A estimativa da pose 2D é baseada no alinhamento dos pontos finais do feixe do laser com o mapa obtido até o momento. Os pontos finais são projetados no mapa e as probabilidades de ocupação são estimadas. A informação odométrica não é usada. O GMapping também funciona com dados de um laserscan para gerar um grid map 2D, porem utilizando um filtro de partículas. KartoSLAM: É um algoritmo SLAM baseado em gráfo. Nesse caso, cada nó representa uma pose do robô ao longo de sua trajetória e um conjunto de medições de sensor. CoreSLAM: Mais um SLAM utilizando dados de laserscan. Dividido em duas etapas, a primeira gera os dados com as distancias. A segunda escolhe as melhores distancias através de um filtro de partículas simples. LagoSLAM:

Outro algoritmo de SLAM baseado em gráfo, o qual optimaliza o resultado através de um filtro linear.

Valencia et al. (2013) propõe um método que usa diretamente a Pose do grafo de restrições do SLAM para determinar o caminho entre duas configurações com menor acumulação de incerteza, ou seja, o caminho mais confiável para o objetivo. O método mostra melhores resultados de navegação quando comparado às estratégias de planejamento de caminho padrão, tanto em conjuntos de dados e experimentos do mundo real.

3.4.1 SLAM visual

Navegação e posicionamento de robôs móveis com sistemas de visão são geralmente realizados com o auxílio de referências naturais ou artificiais (ALEKSANDROVICH et al., 2013). Em Se, Lowe e Little (2001) é examinada a localização global, sem nenhuma estimativa de localização prévia. Isso é conseguido combinando marcos visuais distintos do *frame* atual com um mapa em um banco de dados. Uma abordagem de transformada Hough e uma abordagem RANSAC são comparadas, mostrando que o RANSAC é muito mais eficiente para combinar *features* específicos, mas pior para combinar *features* não específicos. Além disso, uma localização global robusta pode ser alcançada combinando um pequeno mapa da região local.

No trabalho de DeSouza e Kak (2002) é realizada uma survey sobre o desenvolvimento dos últimos 20 anos na área de visão da navegação por robôs móveis. Dois grandes componentes do trabalho tratam da navegação interna e externa. Para cada componente, é subdividido ainda mais o tratamento do assunto com base em ambientes estruturados e não estruturados. Como resultado, reconhece a importância do conhecimento prévio e as várias formas da qual pode ser modelado. Os modelos podem ser geométricos, topológica, baseada em aparência etc. Representações diferentes de esquemas variam em relação ao grau de conhecimento métrico contido, exibindo diferentes graus de robustez no que diz respeito às variações de iluminação, etc.

Sturm et al. (2012) apresentam um novo *benchmark* para a avaliação de sistemas RGB-D SLAM. Foi gravado um grande conjunto de sequências de imagens de um Microsoft Kinect com alta precisão e *ground truth* de poses da câmera sincronizados temporalmente com um sistema de captura de movimento. A trajetória *ground truth* foi obtido a partir de um sistema de captura de movimento com oito câmeras de rastreamento (100 Hz). O conjunto de dados

consiste em 39 sequências que foram gravados em um ambiente de escritório e um industrial. O conjunto de dados cobre uma grande variedade de cenas e movimentos de câmera. Para estimular a comparação de diferentes abordagens, são fornecidas ferramentas de avaliação automática, tanto para a avaliação do *drift* de sistemas de odometria visual e o erro de pose global do SLAM. O site de referência [https://vision.in.tum.de/data/datasets/rgbd-dataset] contém todos os dados, detalhados descrições das cenas, especificações dos formatos de dados, código de exemplo e ferramentas de avaliação.

Em Henry et al. (2012) é apresentado uma investigação em como câmeras RGB-D podem ser utilizadas para construir mapas 3D densos de ambientes indoor. É desenvolvido um sistema de mapeamento que utiliza um algoritmo novo de otimização de juntas combinando *features* visuais e alinhamento baseado em forma. As informações visuais e de profundidade também são combinadas para o fechamento de loop com base na detecção de visualização, seguida de otimização de pose para obter mapas globalmente consistentes. Também temos o trabalho de Lin et al. (2012) em que apresenta um novo algoritmo para localização e mapeamento de robôs móveis com base em visão estéreo. Primeiro, um novo método é proposto para extrair *features* de imagem invariáveis distintos, que são cunhados como PLOT (*Polynomial Local Orientation Tensor*). A estabilidade destas *features* para translação, redimensionamento, rotação e iluminação de imagens os torna adequados para robôs móveis, localização e construção de mapas utilizando marcos. Os marcos visuais relativos ao robô podem ser obtidos coincidindo as PLOT *features*. A localização do robô móvel é obtida combinando-se estes pontos de referência distintos do quadro atual com o banco de dados do mapa. O algoritmo RANSAC é empregado para estimativa de pose de robô móvel devido a sua eficiência. Enquanto isso, os marcos visuais no banco de dados do mapa são atualizados correspondentemente. Resultados experimentais mostram que o método proposto com base nos PLOT *features* alcança localização e mapeamento para robô móvel com grande precisão.

Em Oliver et al. (2012) temos um estudo da viabilidade de um sistema de navegação utilizando técnicas de SLAM em ROS junto com um sensor Microsoft Kinect. Também utilizando técnicas de SLAM com ROS temos o trabalho de Klaser, Osorio e Wolf (2014) em que é desenvolvido um sistema de navegação autônoma baseado em visão com câmera estéreo e SLAM em que é apresentado um método para lidar com a nuvem de pontos 3D ruidosa produzida por câmera estéreo para criar um mapa de navegação e marcar obstáculos com uma abordagem probabilística do mapa de ocupação. E então temos o trabalho de Bonin-Font, Ortiz e Oliver (2014) o qual apresentam uma pesquisa com trabalhos, dos anos 90 até 2014, que

constituem um amplo progresso nas técnicas de navegação visual para veículos subaquáticos terrestres, aéreos e autônomos. O artigo trata de duas das principais abordagens: navegação baseada em mapas e navegação sem mapas.

Endres et al. (2014) desenvolveram um moderno sistema de SLAM 3D para Sensores RGB-D, como o Microsoft Kinect. Sua abordagem extrai características (*keypoints*) visuais das imagens coloridas e usa as imagens de profundidade para localizá-las em 3D. RANSAC é utilizado para estimar as transformações entre *keypoints* associados e otimizar o gráfico de poses usando otimização não-linear. Finalmente, o mapa 3D volumétrico do ambiente é registrado em forma de nuvem de pontos ou *Octomap*, que podem ser usados para localização de robôs, navegação e planejamento de caminhos.

Labbé e Michaud (2014) produziram o RTAB-Map (*Real-Time Appearance-Based Mapping*) que é um tipo de RGB-D SLAM baseado em grafos com um detector que amplia o fechamento de ciclo baseado em aparências. O detector de fechamento de ciclo (*loop closure detector*) usa uma abordagem *bag-of-words* para determinar a probabilidade de uma imagem nova provir de um local anterior ou de um novo local. Quando uma hipótese de fechamento de ciclo é encontrada, uma nova restrição é colocada no gráfo do mapa e, em seguida, um otimizador de grafos minimiza os erros do mapa. Um tipo de gerenciamento de memória é usado para limitar o número de locais usados na detecção de fechamento de ciclo e otimização do grafo (LABBE; MICHAUD, 2013), para que as restrições em tempo real em ambientes de grande escala sejam sempre respeitadas. O RTAB-Map pode ser utilizado sozinho com um Kinect ou câmera estéreo para mapear com RGB-D em 6 DoF, ou em um robô equipado com um sensor de laser para mapear em 3 DoF.

Uma abordagem similar baseada em aparências é descrita por Fidalgo e Ortiz (2013) na qual uma nova medida de similaridade de imagem entre imagens é introduzida baseada no número de correspondências e as suas distancias associadas. Também para otimizar tempo de processamento correspondências entre a imagem atual e localizações previamente visitadas são determinadas utilizando um índice baseado em um *set* de *KD-trees* aleatório. Além disso, um filtro discreto de Bayes é usado para prever candidatos no loop, levando em consideração os relacionamentos anteriores entre correspondências.

No trabalho de Naseer et al. (2015) é apresentada uma nova abordagem para localização visual de robôs móveis em ambientes externos, que é capaz de lidar com mudanças sazonais substanciais. É formulada a correspondência de imagem como um problema de fluxo de custo mínimo em um gráfo de associação de dados para explorar efetivamente a sequência

de informação. Isso permite lidar com não correspondência em sequência de imagens que resultam de oclusão temporal ou de visitar lugares novos. Também temos o trabalho de Schneider (2018) o qual apresenta a implementação, através do ROS, de um Filtro de Kalman Estendido para a fusão de três fontes de sensoriamento diferentes e estima a posição e orientação do robô de forma mais robusta, precisa e, consequentemente, realiza um mapeamento também mais fiel do ambiente em 3D, com registro em uma forma de nuvem de pontos, obtida através das imagens de cor e profundidade de uma câmera Kinect acoplada ao robô.

3.5 PLANEJAMENTO DE TRAJETÓRIAS

Kambhampati e Davis (1986) apresentam métodos de planejamento de trajetória de curta distância para robôs móveis utilizando estruturas de dados *quadtrees* hierárquicas. É observado que a heurística mais importante para um planejamento de trajetória de espaço é evitar o excesso de detalhes em partes do espaço que não afetam a operação de planejamento. A representação *quadtree* naturalmente provém tal descrição. É proposto que planejamento de trajetórias curtas sejam baseadas na decomposição do espaço livre em unidades maiores que pixels para o planejamento se tornar global. Decomposições hierárquicas como *quadtrees* são boas para atingir tais metas pois o custo de representação é pequeno. Além o robô móvel precisa continuamente atualizar a informação do caminho sendo planejado enquanto o navega, ao obter novas informações. Para fazer isso eficientemente são utilizadas funções de adicionar e remover representações para manter o custo da imagem do mapa relativamente baixo. Simmons e Koenig (1995) apresentam uma pesquisa que utiliza modelos de Markov parcialmente observáveis para rastrear com robustez a localização de um robô em ambientes de escritório e direcionar suas ações orientadas a *goals.* A abordagem mantém explicitamente uma distribuição de probabilidade ao longo das possíveis localizações do robô, levando em conta várias fontes de incerteza, incluindo conhecimento aproximado do ambiente e incerteza do sensor e do atuador. O planejamento de trajetória é realizado utilizando um algoritmo de busca A* no mapa topológico para encontrar um caminho ao destino. É utilizado este planejamento para buscas nas arestas e nodos no mapa, baseado na expectativa de tempo total de distância ao destino e estimativas de quanto tempo se leva para curvas. Um planejador de caminho simples e heurísticas de seleção de ação são usadas para direcionar o objetivo da posição do robô. As vantagens dessa abordagem incluem a capacidade de responder pela incerteza na posição inicial do robô, incerteza do atuador, ruído do sensor e incerteza na interpretação dos dados do sensor.

Em Kavraki et al. (1996) um novo método de planejamento de movimento para robôs em espaços de trabalho estáticos é apresentado. Este método prossegue em duas fases: uma fase de aprendizagem e uma fase de consulta. Na fase de aprendizagem um *roadmap* probabilístico é construído e armazenado como um grafo cujos nós correspondem a configurações livres de colisão e cujas arrestas correspondem a caminhos possíveis entre estas configurações. Esses caminhos são computados utilizando um simples e rápido planejador de caminhos. Na fase de consulta dadas configurações de início e destino o robô conecta dois nodos do *roadmap*. O qual é buscado um caminho juntando estes dois nodos. O método pode ser aplicado virtualmente a qualquer robô holonômico, requerindo somente a seleção de alguns parâmetros os quais dependem da cena em que o robô se encontra e seu espaço de navegação. Uma discussão sobre qual planejador local é feita sobre os *tradeoffs* entre a velocidade de obtenção do caminho e sua precisão, no final optando por um algoritmo A* em uma representação C-space.

Meikle e Yates (2002) descrevem um novo algoritmo para robôs móveis autônomos na construção de mapas e planejamento de trajetórias em um ambiente desconhecido. O robô aprende sobre o ambiente usando uma interpretação probabilística baseada em visualização de cantos visíveis e *features* de cantos. Nenhuma tentativa é feita para localizar *features* no mundo físico real utilizando o tradicional sistema de coordenadas euclidianas, na verdade o algoritmo produz um mapa somente através das medidas de similaridades visuais. É um algoritmo de construção de mapas desenvolvido, cuja robustez é independente do tamanho do ambiente aprendido. Isso permite que grandes ambientes serem aprendidos de forma confiável. O robô foi capaz de planejar um caminho entre quaisquer locais conhecidos e navegar ao longo desse caminho com sucesso utilizando um algoritmo CLAM (*Clustering Large Applications Using Metaheuristics*) para navegação local e navegação global utilizando uma medida de similaridade entre vizinhos sem qualquer medida euclidiana. Bortoff (2000) apresenta um algoritmo de planejamento de caminho em duas etapas para UAVs. O algoritmo gera um caminho furtivo através de um conjunto de sites de radar inimigo de localização conhecida e fornece uma maneira intuitiva de compensar furtividade versus comprimento do caminho. Na primeira etapa, um subótimo caminho *rough-cut* é gerado através dos sites de radar, construindo e pesquisando em um grafo baseado em polígonos de Voronoi. Na segunda etapa, um conjunto de equações diferenciais ordinárias não lineares é simulado, usando a solução do grafo como condição inicial. As ODEs descrevem a dinâmica de um conjunto de massas virtuais localizadas em um campo de força virtual. As forças virtuais empurram as massas longe dos radares e em direção um ao outro. As ODEs são simulados para encontrar uma localização exponencialmente

como solução de equilíbrio estável, interpretada como caminho ideal. Também temos o trabalho de Srinivasa et al. (2009) que descreve a arquitetura, algoritmos e experimentos com HERB (*Home Exploring Robotic Butler*) um manipulador móvel autônomo que executa tarefas úteis de manipulação em casa. A localização é realizada utilizando um algoritmo monte carlo adaptativo. O planejamento de trajetória é realizado utilizando o algoritmo A* junto com destinos gerados pelas tarefas com uma grade de ocupação gerada por um laser.

Em Siagian et al. (2014) é descrita uma nova abordagem que utiliza vários módulos de percepção visual como reconhecimento de lugares, de *landmarks* e detecção de linhas de estrada, complementados por sinais de um laser para desvio de obstáculos. Utilizando mapas de ocupação 3D gerados pelo laser é utilizado uma combinação dos algoritmos RTT (*rapidly exploring random tree*) e busca heurística incremental para realizar a navegação no mapa usando o modelo cinemático do robô. No trabalho de Mac et al. (2016) são propostas técnicas de planejamento de trajetória que são divididas em duas categorias: métodos clássicos e métodos heurísticos. Os métodos clássicos consistem em decomposição de célula, método de campo potencial, redes de *subgoals* e *road maps*. Os algoritmos baseados em heurística no planejamento do caminho do robô são redes neurais, lógica fuzzy, algoritmos inspirados na natureza e algoritmos híbridos, com o método de campo potencial obtendo bons resultados. Em Beno et al. (2016) é apresentada uma implementação eficiente de mapeamento e navegação utilizando somente câmeras RGBD. O mapa criado pela técnica de SLAM visual é convertido em uma estrutura *Octree* e navegado utilizando um algoritmo A*. O sistema de mapeamento criado é capaz de operar em tempo real, mesmo em um pequeno robô móvel e pode ser facilmente estendido com novas técnicas de estado da arte. A proposta de codificação do mapa *Octree* criado habilita uma distribuição eficiente do mapa entre múltiplos robôs na rede. O algoritmo de planejamento de trajetória apresentado com as devidas otimizações é capaz de rodar em tempo real mesmo em mapas de alta resolução. Em Song et al. (2019) é apresentado o desenvolvimento de um algoritmo A* aprimorado proposto e aplicado ao Springer USV. Um novo processo de suavização de caminho com três suavizadores de caminho foi desenvolvido para melhorar o desempenho da rota reduzindo "entalhes" desnecessários, sem *waypoints* redundantes e oferecendo uma rota mais contínua. Ambos resultados de simulação e experimentais mostram que o algoritmo A * suavizado supera o algoritmo convencional em ambientes esparsos e desordenados que foram rasterizados uniformemente.

Asadi et al. (2019) apresenta uma arquitetura de rede neural profunda, leve e eficiente para executar em uma plataforma embarcada em tempo real. O modelo proposto segmenta o

espaço navegável em uma sequência de imagens (ou seja, um fluxo de vídeo), essencial para um veículo autônomo baseado na visão computacional. Este artigo apresenta um novo modelo de segmentação semântica eficiente com contribuições como um novo dataset baseado em pixels para ambientes de construção civil. Em Popovic et al. (2020) o trabalho introduz uma estrutura geral de planejamento de caminhos para monitorar cenários usando um robô aéreo, enfocando problemas nos quais o valor das informações do sensor é distribuído de maneira desigual e a área de destino é desconhecida a priori. A abordagem é capaz de aprender e focar em regiões de interesse via adaptação para mapeamento discreto ou contínuo de variáveis no terreno usando dados de resolução variável recebidos de sensores probabilísticos. No trabalho de Panda et al. (2020) temos uma revisão com o intuito de resumir várias estratégias de planejamento de trajetória para AUVs (*Autonomous Underwater Vehicles*) nas bases de caracterização de ambientes subaquáticos como previsíveis e imprevisíveis. Os algoritmos utilizados no planejamento de trajetória de um único AUV e múltiplos AUVs são revisados nas bases de diferentes ambientes, tipo de trajetória gerada, custo de caminho e recurso para desvio de obstáculos. Méritos e deméritos de todos os métodos foram discutidos brevemente. Tipos de caminhos gerados pelos métodos são classificados como o tempo ideal (solução tempo de tempo mínimo), energia ideal (solução energia mínima), subótimo (solução quase ideal) e ótimo (melhor solução possível). Os custos do caminho são comparados como baixo, moderado e alto. Colisão e prevenção de obstáculos são discutidos como alcançados, limitados e pobres com base em se o algoritmo se concentrou nessas questões ou não.

3.6 ROBOT OPERATING SYSTEM (ROS)

Zaman et al. (2011) demonstram em seu trabalho como o Robot Operating System (ROS) provem serviços como os de sistemas operacionais para operar robôs. Utilizando o problema de localização, mapeamento e navegação autônoma como exemplo de problemas populares no campo da robótica. A navegação autônoma em ambientes dinâmicos não é somente desafiadora, mas também cobre muitos fatores situacionais mesmo em ambientes indoor, o qual afetam o processo de navegação e mapeamento. O trabalho presente descreve como um sistema de controle baseado em ROS é utilizado para localização, mapeamento e navegação em ambientes indoor. O mapeamento de diferentes ambientes é apresentado, além, alguns fatores associados com ambientes indoor que podem afetar a navegação autônoma são apresentados. Moore et al. (2016) apresentam um pacote de software de ROS,

robot_localization. O pacote atualmente contem uma implementação de um filtro de kalman extendido (EKF). Ele suporta um número ilimitado de entradas de vários tipos de sensores e permite usuários para personalizar quais dados do sensor são fundidos com a estimativa do estado atual. Takaya et al. (2016) demonstram a importância do uso de simulações no desenvolvimento de um robô móvel, para o teste de componentes de software, comportamento do robô e teste de algoritmos de controle em diferentes ambientes. Neste artigo, é apresentado um ambiente de simulação para robôs móveis baseados em ROS e Gazebo. Neste trabalho, tarefas de navegação autônoma e simulação de mapeamento 3D utilizando programas de controle sob ROS são apresentados. Os resultados da simulação experimental são satisfatórios e mostram a usabilidade do ambiente desenvolvido.

Em Da Silva et al. (2017) percebemos a recente adoção do ROS como framework padrão em robótica e sua contribuição com soluções para vários problemas na área. Um desses problemas sendo o de localização e mapeamento simultâneos (SLAM) ao qual há vários tipos de pacotes e aplicações diferentes disponíveis para serem utilizadas no ROS. Considerando que sempre precisa se avaliar e escolher uma solução apropriada para algum problema em questão este trabalho apresenta uma avaliação experimental de algoritmos de SLAM diferentes utilizando sensores RGBD e disponíveis em ROS, sendo eles o Gmapping, Hector SLAM, ORB SLAM, ORB SLAM 2 e RTAB-Map. Os algoritmos são testados em diferentes cenários e operações realísticas de acordo com os critérios estabelecidos para enfatizar aspectos práticos dos sistemas selecionados. Os experimentos conduzidos neste trabalho devem prover *insights* para roboticistas procurando soluções de SLAM para aplicações indoor. No trabalho de Marin-Plaza et al. (2018) vemos a análise da performance de um método de planejamento de trajetória baseado em *Time Elastic Bands* (TEB) em uma plataforma real baseada no modelo ackermann. O estudo é conduzido analisando a trajetória gerada dos planejadores locais e globais. O software utilizado é o framework *Robot Operating System* (ROS) da *Open Source Robotics Foundation.* A plataforma utilizada é o iCab da universidade Carlos III. Este trabalho foi validado com um teste em que dentro do campus o iCab realizou navegação autônoma entre um ponto inicial e um ponto final sem colisões. Em Quigley et al. (2009) é apresentado uma visão geral do ROS, um sistema de código aberto para robôs. ROS não é um sistema operacional no sentido tradicional de gerenciamento de processos e escalonamento, mas provem uma estrutura de comunicações sobre um sistema operacional *host* de computação heterogênea. Neste trabalho é discutido como o ROS se relaciona com frameworks de software para robôs

existentes e brevemente discute alguns dos pacotes de software e aplicações disponíveis que utilizam ROS.

3.7 EXPLORAÇÃO

Yamauchi (1997) introduz uma nova abordagem para exploração baseada no conceito de fronteiras, regiões nas bordas entre o espaço aberto e espaço inexplorado. Ao se mover a uma nova fronteira um robô móvel pode estender seu mapa em território novo até que todo o ambiente já tenha sido explorado. É descrito um método para detectar fronteiras em grades de evidencia e navegação através destas fronteiras. Também é introduzida uma técnica para minimizar reflexões especulares em grades de evidencia utilizando laser. A abordagem também foi testada com um robô móvel explorando um ambiente real de escritório com uma variedade de obstáculos. Uma vantagem da abordagem descrita é a habilidade de explorar espaços largos e grandes e pequenos e curtos, com paredes e obstáculos em posições arbitrárias. Estratégias de exploração são usadas para guiar robôs móveis para a construção de mapas. Usualmente estratégias de exploração funcionam de maneira gananciosa ao avaliar um número de observações candidatas na base de uma função de utilidade e selecionando a melhor delas. Em Dayanand et al. (2013) é realizada uma pesquisa das abordagens existentes na área de exploração de fronteiras e de vários tipos de SLAM os quais podem ser uteis no processo de exploração de área discutido.

Uslu et al. (2015) realiza uma implementação de uma estratégia autônoma de exploração baseada em fronteiras. Fronteiras são definidas como bordas que são calculadas através de mapeamento e navegação entre áreas conhecidas e desconhecidas. A implementação da exploração baseada em fronteiras é compatível com ROS. Além é utilizada uma plataforma real no estudo para testes e para avaliar os efeitos de alvos em diferentes fronteiras. Os dados são comparados e analisados em termos de comprimento total de trajetória e tempo total de exploração. A exploração autônoma de ambientes desconhecidos por um robô móvel pode ser benéfica dado que robôs podem navegar em ambientes desconhecido sem mapas serem utilizados. Também provem a possibilidade de se construir mapas sem a interação humana. Vemos isso no trabalho de Verbiest et al. (2017) o qual utiliza um método de exploração de fronteiras para exploração. Resultados de simulação e mundo real aplicados com um robô móvel são apresentados e discutidos, ambos utilizando ROS. Sendo exploração o processo de selecionar pontos de destino aos quais trazem a maior contribuição para uma função de ganho

em um ambiente inicial desconhecido, a exploração baseada em fronteiras é a abordagem mais comum para exploração, aonde fronteiras são bordas de regiões exploradas e inexploradas do espaço. Em Topiwala et al. (2018) uma estratégia nova de exploração baseada em fronteiras é desenvolvida, chamada de *Wavefront Frontier Detector* (WFD) e implementada em gazebo. A vantagem deste algoritmo é que o robô pode explorar largos espaços abertos e também espaços pequenos. Além o mapa gerado com esta técnica é comparado e validado com mapa gerado utilizando pacotes de ROS.

3.8 CONCLUSÃO

De modo geral, o uso de descritores de características locais para comparar pontos de referência naturais (características locais) tem se mostrado como uma técnica satisfatória e de ampla utilização na área de visão computacional, encontrando resultados desejáveis e satisfatórios para as tarefas de localização e mapeamento. As técnicas de Odometria visual também tem obtidos resultados satisfatórios, principalmente com um bom ajuste de parâmetros, sendo este um dos problemas na área, pois é realizada a combinação de muitos algoritmos como RANSAC e muitos descritores como SURF os quais dependem de uma boa parametrização. As técnicas de localização já possuem extensa literatura com filtros de markov, monte carlo e de kalman, os quais são utilizados tanto para obtenção da pose quanto para fusão de sensores. No problema de SLAM vimos várias soluções probabilísticas quanto com dados de laser utilizando ROS. Na área de SLAM visual vemos o extensivo uso de câmeras RGBD como o Kinect da Microsoft, combinados com algoritmos como RANSAC e dados como nuvem de pontos os quais possuem bons resultados de mapeamento possibilitando uma navegação segura. No Planejamento de trajetória apesar de novas técnicas serem desenvolvidas ao longo dos anos podemos notar o continuo uso de algoritmos clássicos como A* pela sua facilidade de implementação e bons resultados independentemente da situação, como é utilizado em vários trabalhos diferentes. O *Robot Operating System* (ROS) já se estabeleceu como *framework* padrão na área da robótica como vimos pelos trabalhos apresentados. Vemos pela seção de exploração que a exploração baseada em fronteiras a qual procura aumentar o tamanho de um mapa conhecido pelas bordas desconhecidas já é bem consolidada como a alternativa mais viável na exploração.

4 IMPLEMENTAÇÃO E COMPARAÇÃO DE DOIS SISTEMAS DE NAVEGAÇÃO

Neste capítulo será demonstrado os dois experimentos realizados em simulação no ambiente gazebo através do ROS, um com slam via laser, outro com slam visual utilizando uma câmera RGBD, bem como conceitos para o entendimento do processo de desenvolvimento dos sistemas.

4.1 GAZEBO

A simulação de robôs é uma ferramenta essencial na caixa de ferramentas de todo roboticista. Um simulador bem projetado torna possível testar algoritmos rapidamente, projetar robôs, realizar testes de regressão e treinar o sistema de IA usando cenários realistas. Gazebo oferece a capacidade de simular com precisão e eficiência populações de robôs em ambientes internos e externos complexos. Ao seu alcance está um motor de física robusto, gráficos de alta qualidade e interfaces gráficas e programáticas convenientes. O Gazebo é gratuito e possui uma grande comunidade. Dentre alguns dos recursos oferecidos pelo Gazebo estão:

- Simulação Dinâmica: Acesso a vários motores de física de alto desempenho, incluindo ODE, Bullet, Simbody e DART.

- Gráficos 3D avançados: Utilizando OGRE, Gazebo fornece renderização realista de ambientes, incluindo iluminação de alta qualidade, sombras e texturas.

- Sensores e ruído: Gere dados de sensor, opcionalmente com ruído, de localizadores de distância a laser, câmeras 2D / 3D, sensores estilo Kinect, sensores de contato, força-torque e muito mais.

- Plugins: Podem ser desenvolvidos plugins personalizados para robôs, sensores e controle ambiental. Os plugins fornecem acesso direto à API do Gazebo.

- Modelos de Robôs: Muitos robôs são fornecidos, incluindo PR2, Pioneer2 DX, iRobot Create e TurtleBot. Ou crie o seu próprio usando SDF.

- Transporte TCP / IP: Execute a simulação em servidores remotos e faça interface com o Gazebo por meio de transmissão de mensagens baseada em soquetes usando o Google Protobufs.

4.2 URDF

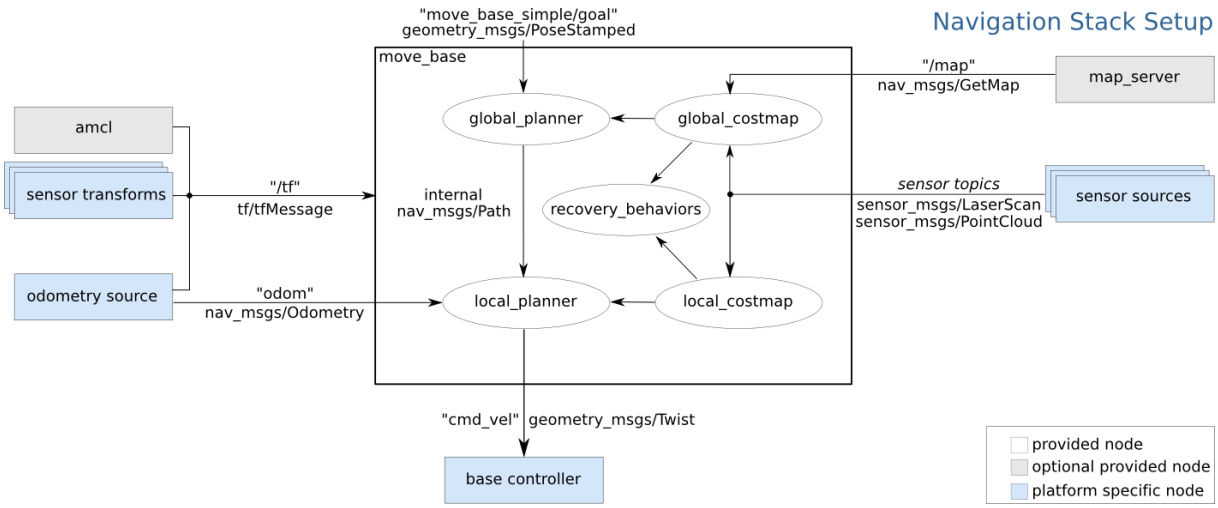
O *Unified Robotic Description Format* (URDF) é um formato de arquivo XML usado no ROS para descrever todos os elementos de um robô. Contém informações sobre os componentes físicos do modelo (<*links*>) e como esses componentes estão conectados (<*joints*>). Os links têm três conjuntos principais de propriedades: <*visual*>, <*collision*> e <*inertial*>. Propriedades visuais definem como o modelo aparecerá no Gazebo e RViz, as propriedades de colisão definem o aspecto físico no espaço que o link ocupa, e as propriedades inerciais definem a massa e a inércia do link. (<*joints*>) simplesmente definem o tipo de conexão entre um link pai e filho como fixo, rotativo ou contínuo. A propriedade <*origin*> especifica a localização da origem do link filho em relação à origem do link pai. Para juntas rotativas e contínuas. a propriedade do eixo deve ser especificada e definido o eixo sobre o qual o link filho pode girar. Além disso, as juntas de rotação também devem especificar os esforços limite, o amortecimento da junta e atrito. Apenas estruturas em árvore podem ser representadas. Para usar um arquivo URDF no Gazebo, algumas tags adicionais específicas da simulação devem ser adicionadas para funcionar corretamente com o Gazebo. Embora os URDFs sejam um formato útil e padronizado em ROS, faltam muitos recursos que não foram atualizados para lidar com as necessidades em evolução da robótica. URDF só pode especificar as propriedades cinemáticas e dinâmicas de um único robô isoladamente. O URDF não pode especificar a pose do próprio robô dentro de um mundo. Também não é um formato de descrição universal, uma vez que não pode especificar loops de juntas (ligações paralelas) e não descreve atrito e outras propriedades. Além disso, ele não pode especificar coisas que não sejam robôs, como luzes, mapas de altura, etc.

4.3 PACOTES ROS

4.3.1 Move_base

O pacote move_base fornece uma implementação de uma ação que, dado um objetivo no mundo, tentará alcançá-lo com uma base móvel. O nó move_base vincula um planejador global e local (dwa_planner) para realizar sua tarefa de navegação global. O nó move_base também mantém dois mapas de custos, um para o planejador global e outro para o planejador local (Costmap_2D). O nó move_base fornece uma interface ROS para configurar, executar e interagir com a pilha de navegação em um robô. Uma visão de alto nível do nó move_base e sua interação com outros componentes é mostrada na figura 10.

Figura 10 – Visão geral do nó move_base

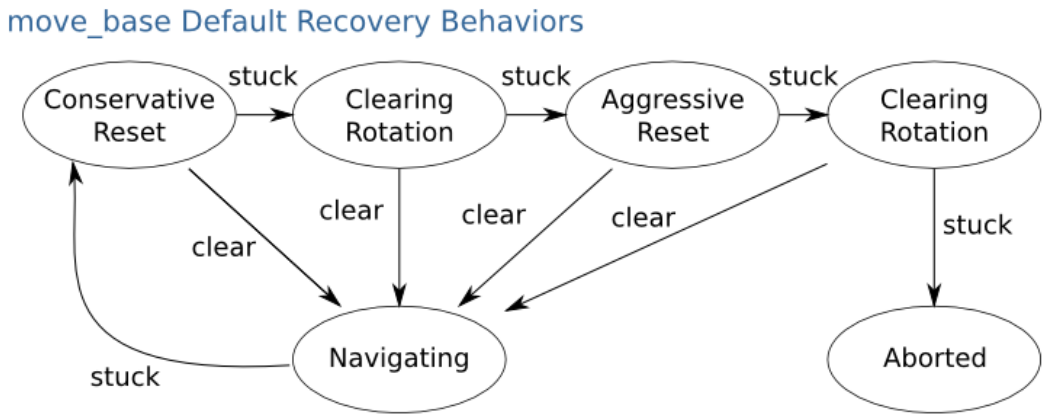


Fonte: http://wiki.ros.org/move_base

O azul varia de acordo com a plataforma do robô, o cinza é opcional, mas é fornecido para todos os sistemas e os nós brancos são obrigatórios, mas também são fornecidos para todos os sistemas. Executando o nó move_base em um robô que está configurado corretamente resulta em um robô que tentará atingir uma pose de objetivo com sua base dentro de uma tolerância especificada pelo usuário. Na ausência de obstáculos dinâmicos, o nó move_base acabará por

ficar dentro desta tolerância de seu objetivo ou envia uma falha de sinal para o usuário. O nó move_base pode opcionalmente executar comportamentos de recuperação quando o robô se percebe como travado. Por padrão, o nó move_base executará as seguintes ações para tentar limpar o espaço como visto na figura 11.

Figura 11 – Comportamento de recuperação



Fonte: http://wiki.ros.org/move_base

Primeiro, os obstáculos fora de uma região especificada pelo usuário serão removidos do mapa do robô. Em seguida, se possível, o robô executará uma rotação no local para liberar espaço. Se isso também falhar, o robô limpará seu mapa de forma mais agressiva, removendo todos os obstáculos fora da região retangular na qual ele pode girar no lugar. Isso será seguido por outra rotação no local. Se tudo isso falhar, o robô considerará seu objetivo inviável e notificará o usuário de que foi abortado. Esses comportamentos de recuperação podem ser configurados usando o parâmetro recovery_behaviors e desabilitados usando o parâmetro recovery_behavior_enabled.

4.3.2 Costmap_2D

Este pacote fornece uma implementação de um mapa de custo 2D que leva dados do sensor do ambiente, constrói uma grade de ocupação 2D ou 3D dos dados (dependendo se uma implementação baseada em voxel é usada) e aumenta os custos em um mapa de custo 2D com base na grade de ocupação e um raio de inflação especificado pelo usuário. Este pacote também fornece suporte para inicialização baseada em map_server de um mapa de custos, mapas de

custos baseados em janela contínua e assinatura baseada em parâmetros com configuração de tópicos de sensores. O pacote costmap_2d fornece uma estrutura configurável que mantém informações sobre onde o robô deve navegar na forma de uma grade de ocupação. O mapa de custos usa dados do sensor e informações do mapa estático para armazenar e atualizar informações sobre obstáculos no mundo através do objeto costmap_2d :: Costmap2DROS.

O mapa de custos subscreve automaticamente os tópicos de sensores no ROS e se atualiza de acordo. Cada sensor é usado para marcar (inserir informações de obstáculo no mapa de custo), limpar (remover informações de obstáculo do mapa de custo) ou ambos. Uma operação de marcação é apenas um índice em uma matriz para alterar o custo de uma célula. Uma operação de limpeza, no entanto, consiste em traçar o raio através de uma grade da origem do sensor para fora para cada observação relatada. Se uma estrutura tridimensional for usada para armazenar informações sobre obstáculos, as informações sobre obstáculos de cada coluna serão projetadas em duas dimensões quando colocadas no mapa de custos. Embora cada célula no mapa de custo possa ter um dos 255 valores de custo diferentes, a estrutura subjacente que ela usa é capaz de representar apenas três. Especificamente, cada célula nesta estrutura pode estar livre, ocupada ou desconhecida. Cada status tem um valor de custo especial atribuído a ele na projeção no mapa de custos. As colunas que têm um certo número de células ocupadas são atribuídas a um custo *costmap_2d :: LETHAL_OBSTACLE*, as colunas que têm um certo número de células desconhecidas são atribuídas a um custo *costmap_2d :: NO_INFORMATION* e outras colunas são atribuído a um custo *costmap_2d :: FREE_SPACE*. O mapa de custos executa os ciclos de atualização do mapa na taxa especificada pelo parâmetro *update_frequency*. A cada ciclo, os dados do sensor entram, as operações de marcação e compensação são realizadas na estrutura de ocupação subjacente do mapa de custo, e essa estrutura é projetada no mapa de custo onde os valores de custo apropriados são atribuídos conforme descrito acima. Depois disso, cada inflação de obstáculo é executada em cada célula com um custo *costmap_2d :: LETHAL_OBSTACLE*. Isso consiste em propagar os valores de custo de cada célula ocupada para um raio de inflação especificado pelo usuário.

4.3.3 GMapping

GMapping é um filtro de partículas *Rao-Blackwellized* altamente eficiente para aprender mapas de grade a partir de dados de laser. Os filtros de partículas *Rao-Blackwellized* foram introduzidos como meios eficazes para resolver o problema de localização e mapeamento

simultâneos (*SLAM*)(Grisetti 2005). Essa abordagem usa um filtro de partículas em que cada partícula carrega um mapa individual do ambiente. Consequentemente, uma questão chave é como reduzir o número de partículas. São apresentadas técnicas adaptativas para reduzir o número de partículas em um filtro de partículas *Rao-Blackwellized* para mapas de grade de aprendizagem. É proposta uma abordagem para calcular uma distribuição precisa da proposta levando em consideração não apenas o movimento do robô, mas também a observação mais recente. Isso diminui drasticamente a incerteza sobre a pose do robô na etapa de previsão do filtro. Além disso, é aplicada uma abordagem para realizar seletivamente operações de reamostragem que reduz seriamente o problema de esgotamento de partículas. A abordagem leva dados brutos de laser e odometria. Esta versão é otimizada para scanners a laser de longo alcance, como scanner SICK LMS ou PLS. Lasers de curto alcance, como o scanner Hokuyo, não funcionarão muito bem com as configurações de parâmetro padrão.

4.3.4 RTABMAP_ros

RTAB-Map (*Real-Time Appearance-Based Mapping*) é uma abordagem SLAM baseada em gráfo utilizando Lidar, Câmera estéreo e RGB-D, baseada em um detector de fechamento de loop baseado em aparência incremental. O detector de fechamento de loop usa uma abordagem de *bag-of-words* para determinar a probabilidade de uma nova imagem vir de um local anterior ou de um novo local. Quando uma hipótese de fechamento de loop é aceita, uma nova restrição é adicionada ao gráfo do mapa e, em seguida, um otimizador de gráfo minimiza os erros no mapa. Uma abordagem de gerenciamento de memória é usada para limitar o número de locais usados para detecção de fechamento de loop e otimização de gráfo, de modo que as restrições em tempo real em ambientes de grande escala sejam sempre respeitadas. O *RTAB-Map* pode ser usado sozinho com um Kinect portátil, uma câmera estéreo ou um lidar 3D para mapear 6DoF, ou em um robô equipado com um laser para mapear 3DoF. Este pacote é um *wrapper* ROS do RTAB-Map, que pode ser usado para gerar nuvens de pontos 3D do ambiente e / ou para criar um mapa de grade de ocupação 2D para navegação.

4.3.5 Octomap

A biblioteca OctoMap implementa uma abordagem de mapeamento de grade de ocupação 3D, fornecendo estruturas de dados e algoritmos de mapeamento em C++

particularmente adequados para robótica. A implementação do mapa é baseada em uma octree e foi projetada para atender aos seguintes requisitos:

- Modelo 3D completo: O mapa é capaz de modelar ambientes arbitrários sem suposições anteriores sobre isso. Os modelos de representação ocuparam tanto áreas como espaço livre. Áreas desconhecidas do ambiente são codificadas implicitamente no mapa. Embora a distinção entre espaço livre e ocupado seja essencial para a navegação segura do robô, informações sobre áreas desconhecidas são importantes, por exemplo, para a exploração autônoma de um ambiente.

- Atualizável: É possível adicionar novas informações ou leituras do sensor a qualquer momento. A modelagem e atualização são feitas de forma probabilística. Isso leva em conta o ruído do sensor ou medições que resultam de mudanças dinâmicas no ambiente, por exemplo, por causa de objetos dinâmicos. Além disso, vários robôs são capazes de contribuir para o mesmo mapa e um mapa previamente gravado é extensível quando novas áreas são exploradas.

- Flexível: A extensão do mapa não precisa ser conhecida com antecedência. Em vez disso, o mapa é expandido dinamicamente conforme necessário. O mapa é multi-resolução de forma que, por exemplo, um planejador de alto nível é capaz de usar um mapa grosso, enquanto um planejador local pode operar usando uma resolução fina. Isso também permite visualizações eficientes que variam de visões gerais grosseiras a visualizações detalhadas de perto.

- Compacto: O mapa é armazenado de forma eficiente, tanto na memória quanto no disco. É possível gerar arquivos compactados para uso posterior ou troca conveniente entre robôs, mesmo com restrições de largura de banda.

4.3.6 DWA_Local_Planner

Este pacote fornece uma implementação da Abordagem de Janela Dinâmica para a navegação local do robô em um plano. Dado um plano global a seguir e um mapa de custos, o planejador local produz comandos de velocidade para enviar a uma base móvel. Este pacote suporta qualquer robô cuja pegada pode ser representada como um polígono ou círculo convexo e expõe sua configuração como parâmetros ROS que podem ser definidos em um arquivo de inicialização. Os parâmetros para este planejador também são reconfiguráveis dinamicamente. O *wrapper* ROS deste pacote adere à interface *BaseLocalPlanner* especificada no pacote *nav_core*. O pacote *dwa_local_planner* fornece um controlador que aciona uma base móvel no plano. Este controlador serve para conectar o planejador de caminho ao robô. Usando um mapa, o planejador cria uma trajetória cinemática para o robô ir do ponto inicial até o local de destino. Ao longo do caminho, o planejador cria, pelo menos localmente em torno do robô, uma função de valor, representada como um mapa de grade. Esta função de valor codifica os custos de passagem pelas células da grade. O trabalho do controlador é usar esta função de valor para determinar as velocidades dx, dy, dtheta para enviar ao robô. A ideia básica do algoritmo Dynamic Window Approach (DWA) é a seguinte:

> Pegue uma amostra discreta no espaço de controle do robô (dx, dy, dtheta).
>
> Para cada velocidade amostrada, execute simulação direta a partir do estado atual do robô para prever o que aconteceria se a velocidade amostrada fosse aplicada por algum (curto) período de tempo.
>
> Avalie (pontue) cada trajetória resultante da simulação de avanço, usando uma métrica que incorpore características como: proximidade de obstáculos, proximidade do gol, proximidade do caminho global e velocidade. Descarte as trajetórias ilegais (aquelas que colidem com obstáculos).
>
> Escolha a trajetória de maior pontuação e envie a velocidade associada para a base móvel.
>
> Limpe e repita.

4.3.7 Frontier_exploration

Implementação de exploração de fronteira para ROS, estendendo-se na pilha de navegação existente *(costmap_2d, move_base*). Ele aceita objetivos de exploração via *actionlib* (*Rviz* UI fornecido), envia comandos de movimento para *move_base*. O pacote frontier_exploration fornece um plugin de camada *costmap_2d BoundedExploreLayer* e nós de cliente / servidor *actionlib explore_client e explore_server*. Os nós fornecidos podem ser usados para demonstrar a funcionalidade da camada de mapa de custos, executando uma tarefa de exploração de fronteira limitada por uma área de polígono definida pelo usuário. A exploração de fronteira usando este pacote requer uma configuração de robô real ou simulada que fornece a seguinte funcionalidade:

Scanner a laser ou sensor semelhante que pode liberar espaço e marcar obstáculos.

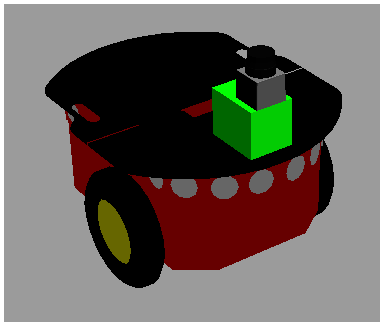
Pilha de navegação configurada corretamente que pode aceitar objetivos de ação para *move_base*.

(Opcional) *Global / map* fornecido *por map_server, gmapping* ou o mapa de custo global de *move_base*

4.4 SIMULAÇÃO UTILIZANDO SLAM COM LASER

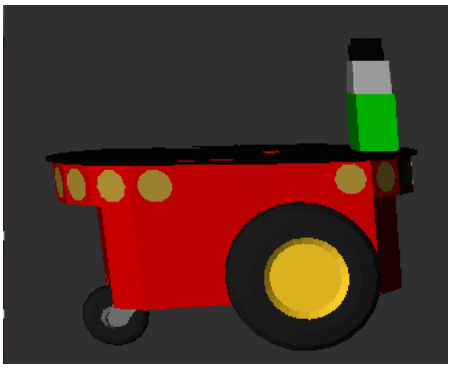
As simulações são realizadas em um ubuntu 16.04 (xenial) e usando ROS versão Kinetic juntamente com Gazebo versão 7.16. A primeira simulação é feita utilizando um modelo URDF de um Pioneer 3DX, um robô de driver diferencial e uma roda livre como podemos ver pelas figuras 12 e 13. Tambem foi adicionado um scaner a laser Hokuyo no arquivo URDF para ser utilizado no robô.

Figura 12 – Modelo URDF Pioneer 3DX.



Fonte: Elaborada pelo autor.

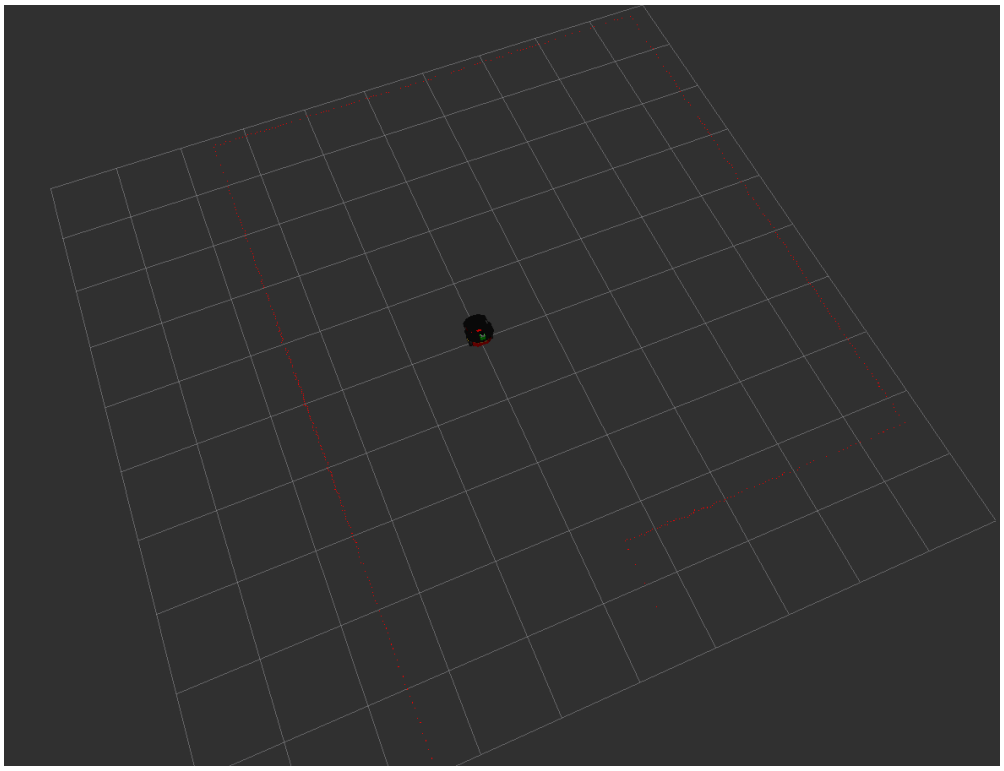
Figura 13 – Modelo URDF Pioneer 3DX visto de lado.



Fonte: Elaborada pelo autor.

Na figura 14 podemos ver um exemplo de leituras do sensor *laserscanner* nas linhas pontilhadas em vermelho no visualizador RVIZ.

Figura 14 – Linhas pontilhadas das leituras do *laserscanner*.



Fonte: Elaborada pelo autor.

O mapa escolhido foi um labirinto, o qual se beneficia das medições feitas com laser, e foram designados 8 salas objetivo do labirinto para o robô mapear, destacadas nos retângulos vermelhos, como podemos ver pela figura 15.

Figura 15 – Mapa de labirinto no Gazebo com salas objetivo destacadas.

Fonte: Elaborada pelo autor.

Primeiramente foi configurado própriamente o pacote move_base, configurando-se os parametros locais e globais do planejador de caminhos, que utiliza um algoritmo *DWA* (Janela de aproximação Dinâmica). A pose e twist do robô são retornados em tempo real do Gazebo através de um serviço do ROS chamado /gazebo/get model state. Um controle PID também é utilizado configurado por um arquivo pioneer controller params.yaml, o qual publica mensagens do tipo /cmd_vel para o pacote /move_base. Os parâmetros nos quatro arquivos de configuração utilizados são descritos a seguir. Mais informações sobre os parâmetros disponíveis e como ajustá-los podem ser encontradas em http://wiki.ros.org/navigation/Tutorials/RobotSetup

4.4.1 base local planner params.yaml

Este arquivo contém parâmetros relacionados à cinemática do robô. Comandos de mínimo e máximo das velocidades são definidas aqui, bem como os limites de aceleração do robô. Alguns outros parâmetros são:

Escape vel - a velocidade a ser aplicada quando o robô está preso (deve ser negativa para que o robô reverta sua direção).

Sim time - quantidade de tempo para simular trajetórias em segundos.

Meter scoring - booleano para denotar se as medidas de distância são em metros.

Heading lookahead - distância em metros ao longo das trajetórias de rotação no local para comparar rumos.

Pdist scale - peso descrevendo a importância do robô ficar perto do caminho.

Gdist scale - peso que descreve a importância do robô atingir a meta.

A escala pdist e a escala gdist parecem operar de maneira relativa, uma escala gdist muito mais elevada vai fazer com que o robô corte atalhos de forma mais agressiva, enquanto uma escala pdist mais alta irá encorajar o robô a acompanhar mais de perto o caminho planejado. Para mais informações sobre planejador local de base acesse: http://wiki.ros.org/base_local_planner . Abaixo segue os parâmetros como configurados:

TrajectoryPlannerROS:
max_vel_x: 1.0
min_vel_x: 0.0
max_vel_theta: 1.0
min_vel_theta: -1.0
min_in_place_vel_theta: 0.4
escape_vel: -0.2
acc_lim_theta: 3.2
acc_lim_x: 2.5
acc_lim_y: 2.5
xy_goal_tolerance: 0.25
yaw_goal_tolerance: 3.15
holonomic_robot: *false*

sim_time: 1.7
meter_scoring: *true*
heading_lookahead: 0.325
pdist_scale: 0.8
gdist_scale: 1.0
planner_frequency: 0.5

4.4.2 costmap common params.yaml

Os parâmetros definidos neste arquivo .yaml são aplicados ao planejador local e global. A pegada do robô e os parâmetros do sensor normalmente seriam incluídos neste arquivo. Também é aceitável não incluir um arquivo common_params.yaml se você escolher definir os parâmetros separadamente nos arquivos de parâmetro local e global. Abaixo seguem os parâmetros como configurados:

obstacle_range: 2.0
raytrace_range: 3.0
footprint: [[0.20, 0.20], [0.20, -0.20], [-0.30, -0.20], [-0.30, 0.20]]
transform_tolerance : 0.5

4.4.3 global costmap params.yaml

O mapa de custo global atualiza e publica frequências que podem ser menores do que as do planejador local, uma vez que não precisa levar em conta a detecção de colisões locais. O raio de inflação determina o tamanho do *buffer* de obstáculos (em células ou metros, dependendo do parâmetro de *Meter scoring*). A zona de *buffer* contém um campo de custo variável e existe para fora da região de colisão (região ocupada por obstáculo + pegada do robô). A forma do campo de custo, ou seja, a rapidez com que o custo cai para 0, é determinada pelo *cost scaling factor*. Quanto maior for esse valor, mais rapidamente o custo diminui. Definir o valor muito alto causa o planejador sugerir caminhos que atravessam muito perto de regiões de alto custo e isso pode resultar em espasmos de execução. Se sensores ruídosos ou um robô menos ágil for usado, as atualizações do mapa podem frequentemente encontrar o robô em

regiões inesperadas de alto custo, forçando manobras de correção agressivas. Abaixo seguem os parâmetros como configurados:

global_costmap:
global_frame: */map*
robot_base_frame: *base_link*
update_frequency: 5.0
publish_frequency: 1.0
static_map: *true*
inflation_radius: 1.5
cost_scaling_factor: 5.0

4.4.4 local costmap params.yaml

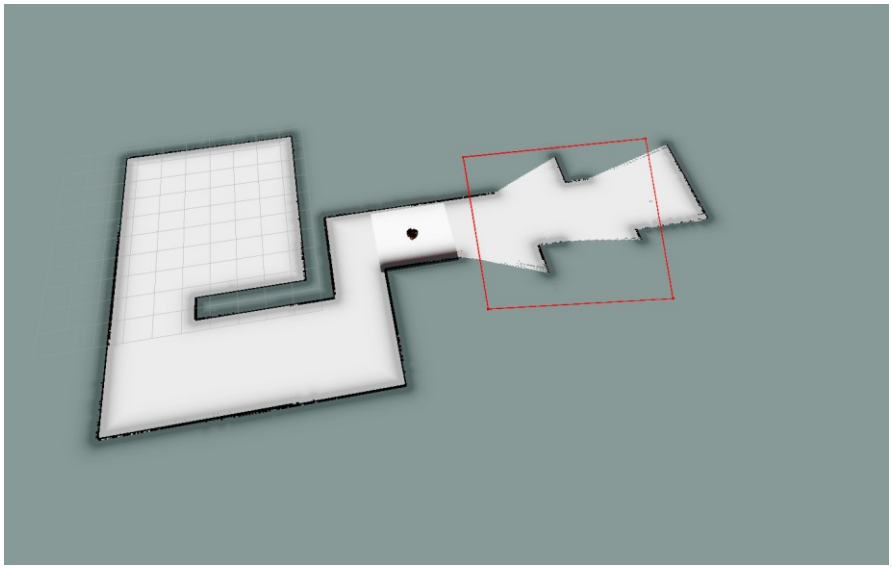
As frequências de atualização e publicação do mapa de custo local devem ser tão altas ou mais altas do que as do mapa de custos global para garantir a prevenção de colisões. O parâmetro do mapa estático deve ser definido como falso já que o mapa de custos local deve se mover com o robô conforme ele navega no espaço. Configurando o parâmetro de janela rolante para verdadeiro permitirá que o planejador descarte mapas antigos do mundo, que não são mais necessários para evitar colisões locais (as informações do mapa ainda serão armazenadas no mapa de custo global). A largura e a altura do mapa de custos local devem ser escolhidas criteriosamente, levando em consideração o robô, alcance do sensor e o ambiente em que estará operando. O planejador local atribui um caminho que tenta corresponder ao do caminho global final que é visível na janela do mapa local. Assim, se a janela local é definida muito grande, o planejador local tentará fazer curvas muito mais cedo e, dependendo do ambiente, estará mais sujeito a travar. Em termos do fator de escala de custo local, a experiência sugere que a definição do fator de escala de custo local menor do que o fator de escala de custo global resulta em trajetórias mais suaves. Uma escala de baixo custo local permite manobras de prevenção de colisão mais suaves quando novos obstáculos são observados, como quando virando cantos. Abaixo seguem os parâmetros como configurados:

local_costmap:
global_frame: *odom*

robot_base_frame: *base_link*
update_frequency: 5.0
publish_frequency: 5.0
static_map: *false*
rolling_window: *true*
width: 3.0
height: 3.0
resolution: 0.05
publish_cost_grid: *true*
inflation_radius: 2.0
cost_scaling_factor: 5.0

Após configurados os parâmetros globais e locais utilizados do pacote 2d_costmap pelo move_base e configurado o SLAM gmapping é hora de utilizar o pacote Frontier_exploration. Ele funciona da seguinte forma, como a visualização da simulação é feita através do RVIZ (*ROS Visualization Tool*), ele tambem é utilizado para definir o polígono de exploração do Frontier_exploration, como podemos ver pela figura 16.

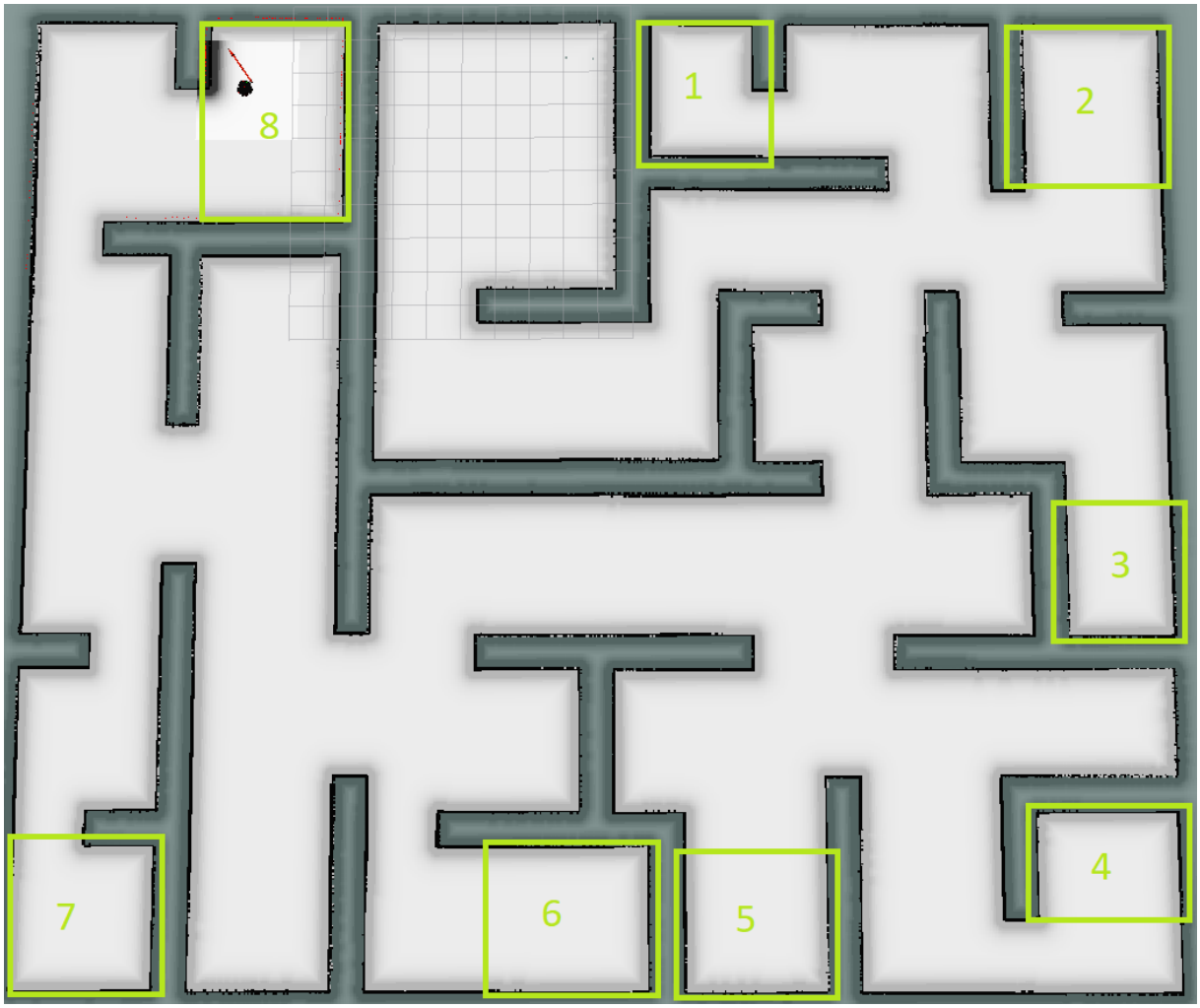
Figura 16 – Polígono do pacote exploração de fronteira.



Fonte: Elaborada pelo autor.

Então com isso já é possível navegar pelo labirinto até o fim visto pela figura 17. O comportamento de recuperação padrão do pacote move_base é ativado se o robô se percebe travado, o que possibilita uma exploração autonoma dentro dos limites do polígono definido. Como podemos ver pela figura 17 todas as salas objetivo foram mapeadas com sucesso, sendo destacadas em verde. Foram utilizados um total de 21 polígonos para mapear completamente o ambiente, durante um tempo de 1.105 segundos. Um grafo RQT com todos os nós do ROS ativos pode ser visto no anexo A. Segue link do video da simulação sendo realizada: https://youtu.be/bT1-qAWB-ng

Figura 17 – Mapa completamente explorado.

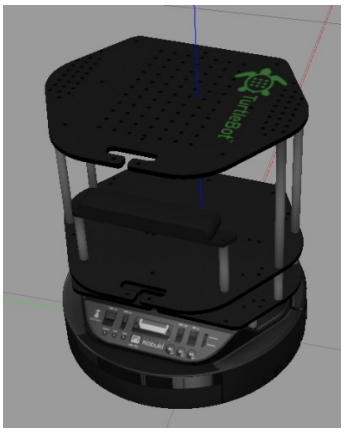


Fonte: Elaborada pelo autor.

4.5 SIMULAÇÃO COM SLAM UTILIZANDO CÂMERA RGBD

A simulação dois é realizada utilizando um modelo URDF de um turtlebot como podemos ver na figura 18.

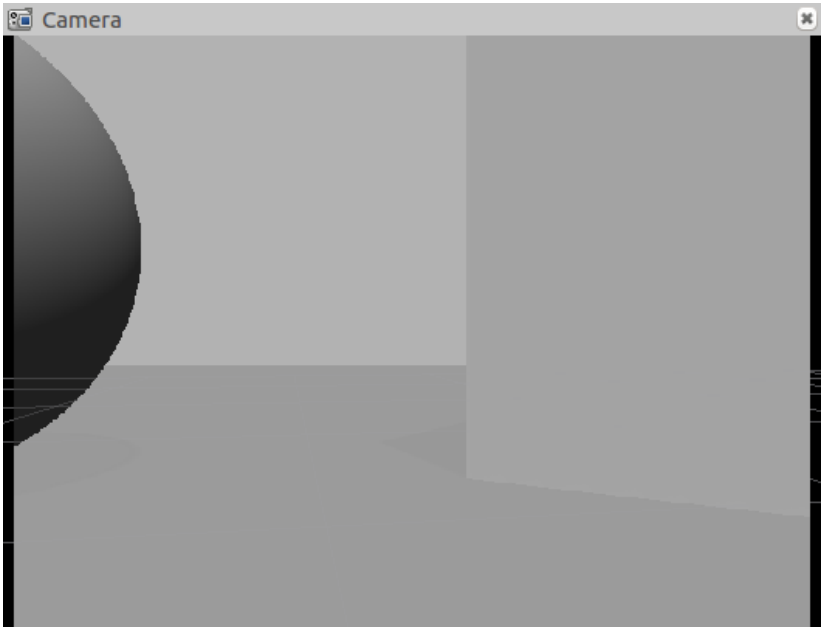
Figura 18 – Modelo URDF de um turtlebot



Fonte: Elaborada pelo autor.

Podemos ver através da visão da câmera do robô na figura 19.

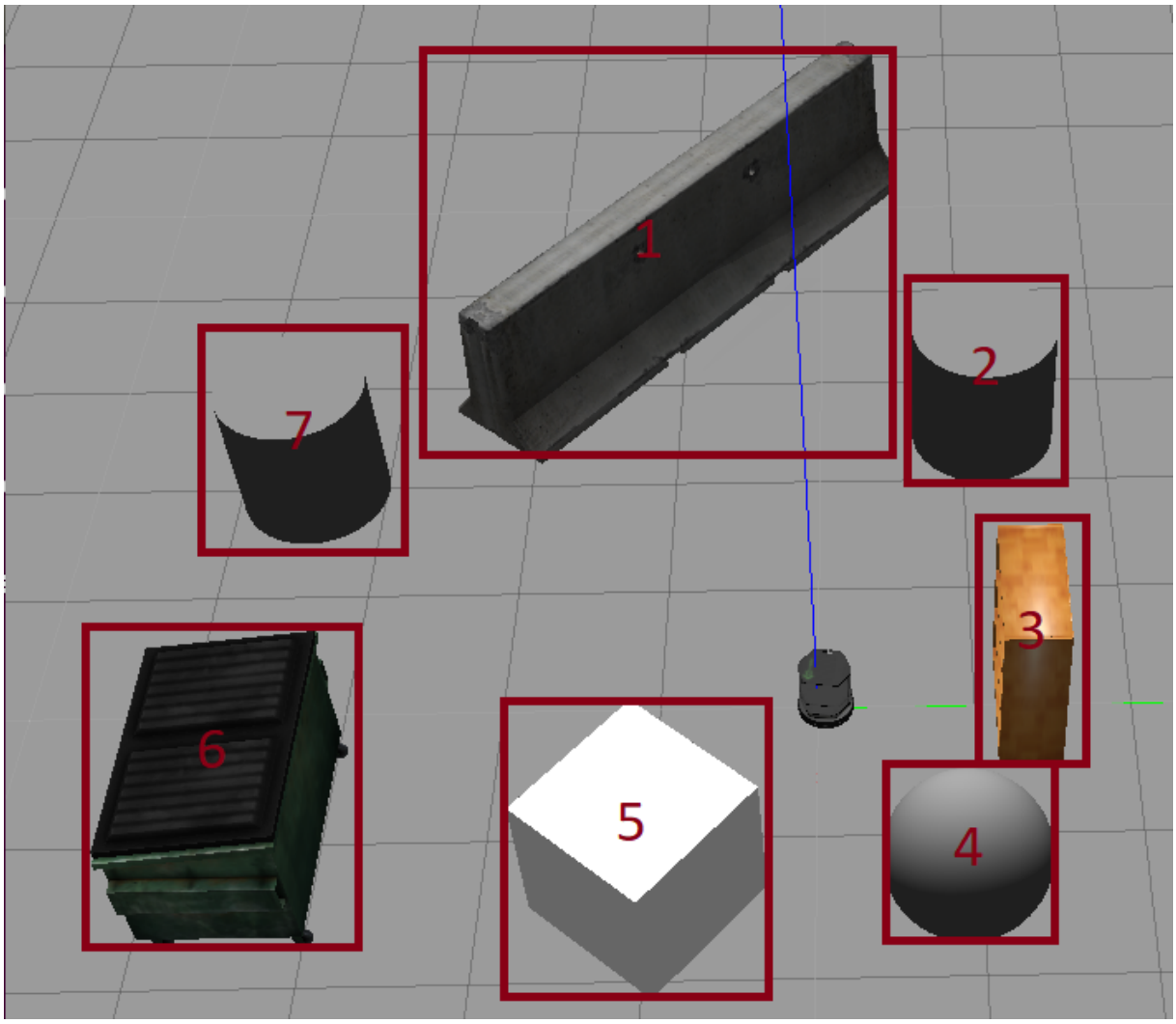
Figura 19 – Visão da câmera RGBD Kinect.



Fonte: Elaborada pelo autor.

Esta simulação tem o diferencial de usar um SLAM com odometria visual, o RTABMAP, para isso é adicionada uma câmera RGBD ao modelo URDF. Por isso o mapa utilizado foi escolhido por ter 7 diferentes marcos visuais, os quais facilitam a localização do robô, como vemos na figura 20, sendo os marcos destacados por retângulos vermelhos.

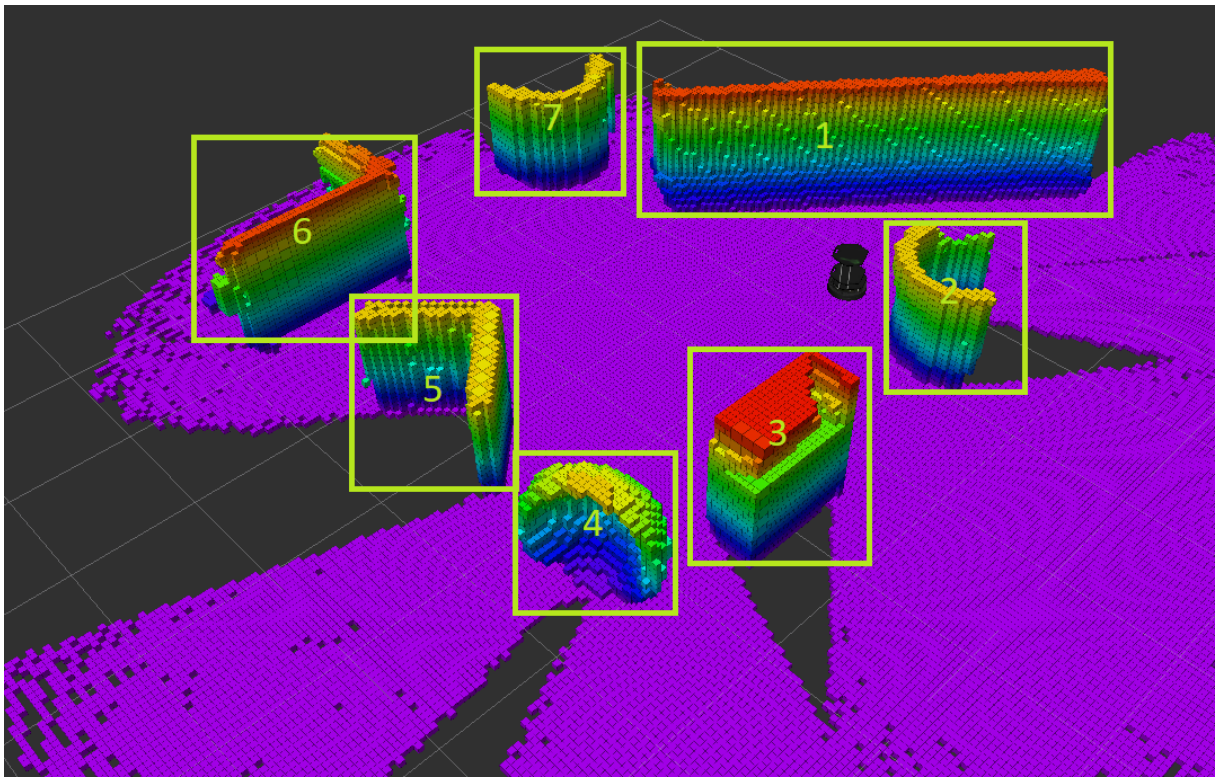
Figura 20 – Mapa do Gazebo para a Simulação 2.



Fonte: Elaborada pelo autor.

O sistema segue a mesma arquitetura da primeira simulação, com a configuração do pacote move_base, seus comandos de entrada sendo mensagens /cmd_vel e seu modo de recuperação se estiver trancado. Em seguida é configurado o RTABMAP, com o ORB como detector de *features* utilizado. Após é lançado o nó do Frontier_exploration para realizar a exploração dentro de uma área retangular demarcada através do RVIZ, como podemos ver pela figura 21 o resultado.

Figura 21 – Resultado SLAM visual com representação pelo Octomap.



Fonte: Elaborada pelo autor.

Neste caso o robô consegue navegar somente dentro da área com os marcos visuais, pois se olha entre obstaculos que possuem um espaço vazio muito grande ou uma parede lisa ele perde a odometria e não consegue se localizar novamente. Por isso que o mapa fica limitado ao número de marcos visuais visíveis pelo robô para se localizar. A representação 3D do mapa é feita no RVIZ utilizando o pacote Octomap, com diferentes cores representando diferentes alturas das medições. Foi utilizado somente um poligôno para mapear todos os 7 marcos visuais como destacado pelos retângulos verdes na figura 21. A simulação durou um total de 130 segundos. No Anexo B pode ser encontrado o grafo RQT contendo os nós ROS utilizados. Segue link com o video da simulação sendo realizada: https://youtu.be/Y93mpVEaXcs

5 CONCLUSÃO

5.1 CONCLUSÕES

Neste trabalho foi apresentado o desenvolvimento de um sistema capaz de realizar a exploração de maneira autônoma dentro de um limite estabelecido. A exploração de fronteira se mostra uma técnica eficaz para mapear ambientes previamente desconhecidos, tanto com SLAM tradicional quanto com SLAM visual. Foram realizados dois experimentos que trazem algumas peculiaridades. O primeiro foi realizado utilizando como sensor somente um *laser scanner* o qual se provou mais do que capaz de realizar o SLAM e a navegação em um ambiente como um labirinto, mas pode se provar ineficaz em ambientes que possuam obstáculos que não chegam a altura da posição do laser no robô, como por exemplo uma mesa de centro, o que torna perigosa a navegação do robô, pois o mesmo não consegue detectar o obstáculo eficazmente. Fora isso, o laser possui um bom alcance em duas dimensões e pode detectar rapidamente obstáculos de maneira dinâmica através de um mapa de custos local. O segundo experimento utilizou uma câmera RGBD (*Red Green Blue Depth*) como fonte de sensoriamento e com isso foi utilizado um SLAM visual para se localizar e mapear o ambiente. Como principal desafio se notou a impossibilidade de navegação em ambientes aos quais não possuem vários marcos visuais, tornando a localização uma tarefa impossivel para somente este sensor se deparado com uma parede lisa ou marcos muito distantes. Porém dentro do ambiente com marcos visuais o sistema foi capaz de mapear autônomamente a área demarcada pelo polígono através da interface do RVIZ. Tambem foi notado a facilidade de implementação de técnicas de alto nível com a utilização do ROS, mesmo que em simulação.

5.2 TRABALHOS FUTUROS

Como trabalhos futuros ficam a implementação do sistema em um hardware de um robô real, o que infelizmente não foi possível. Também fica a possibilidade de realizar uma fusão de sensores entre a câmera RGBD e o sensor a laser para corrigir a falha que a falta de marcos visuais causa no SLAM visual. Também fica a sugestão de criar um servidor de ação do ROS que automaticamente crie os polígonos necessários para o funcionamento do pacote frontier_exploration.

REFERÊNCIAS

AL-KHAWALDEH, Mustafa; AL-NAIMI, Ibrahim; CHEN, XI; MOORE, Philip. Ubiquitous robotics for knowledge-based auto-configuration system within smart home environment. **2016 7Th International Conference On Information And Communication Systems (Icics)**, [S.L.], p. 139-144, abr. 2016. IEEE. http://dx.doi.org/10.1109/iacs.2016.7476100.

ZHOU, Jie-Hua; ZHOU, Ji-Qiang; ZHENG, Yong-Sheng; KONG, Bin. Research on Path Planning Algorithm of Intelligent Mowing Robot Used in Large Airport Lawn. **2016 International Conference On Information System And Artificial Intelligence (Isai)**, [S.L.], p. 375-379, jun. 2016. IEEE. http://dx.doi.org/10.1109/isai.2016.0086

KANDA, Takayuki; SHIOMI, Masahiro; MIYASHITA, Zenta; ISHIGURO, Hiroshi; HAGITA, Norihiro. An affective guide robot in a shopping mall. **Proceedings Of The 4Th Acm/Ieee International Conference On Human Robot Interaction - Hri '09**, [S.L.], p. 173-180, 2009. ACM Press. http://dx.doi.org/10.1145/1514095.1514127.

CHEN, Chiu-Hung; LIU, Tung-Kuan; CHOU, Jyh-Horng. A Novel Crowding Genetic Algorithm and Its Applications to Manufacturing Robots. **Ieee Transactions On Industrial Informatics**, [S.L.], v. 10, n. 3, p. 1705-1716, ago. 2014. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tii.2014.2316638.

SACCHETIN, Marcelo Carvalho. **Análise e implementação de algoritmos para localização e mapeamento de robôs móveis baseada em computação reconfigurável.** 2005. 120 f. Dissertação (Mestrado) - Curso de Ciências de Computação e Matemática Computacional, Instituto de Ciências Matemáticas e de Computação - Icmc, Usp, São Carlos, 2005.

GASPAR, J. **Visão para robótica móvel: Detecção de obstáculos sobre pavimento plano.** Tese (Doutorado) — Master thesis in Engenharia Electrotécnica e de Computadores, IST, 1994.

WANG, Xiaochun; WANG, Xiali; WILKES, Don Mitchell. **Machine Learning-based Natural Scene Recognition for Mobile Robot Localization in An Unknown Environment**. Xi'an, Shaanxi, China: Springer, 2020. 340 p.

SCHNEIDER, Danilo Giacomin. **SISTEMA DE LOCALIZAÇÃO E MAPEAMENTO 3D SIMULTÂNEOS EM ROBÔS MÔVEIS**: fusao de sensores para localização com filtro de kalman estendido. 2018. 111 f. Dissertação (Mestrado) - Curso de Pós-Graduação em Engenharia de Automação e Sistemas, DAS, Universidade Federal de Santa Catarina, Florianópolis, 2018.

BICCHI, A. et al. On the problem of simultaneous localization, map building, and servoing of autonomous vehicles. In: **Advances in control of articulated and mobile robots**. [S.l.]: Springer, 2004. p. 223–242.

ZHANG, L.; GHOSH, B. K. Line segment based map building and localization using 2D laser rangefinder. **Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on,** IEEE, v. 3, p. 2538–2543 vol.3, 2000.

SURMANN, H.; NÜCHTER, A.; HERTZBERG, J. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. **Robotics and Autonomous Systems**, v. 45, n. 3, p. 181–198, 2003. ISSN 09218890.

MORENO, L. et al. A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors. **Journal of Intelligent and Robotic Systems**, Kluwer Academic Publishers, v. 34, n. 2, p. 135–154, 2002. ISSN 09210296.

AGRAWAL, M.; KONOLIGE, K. Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS. In: **18th International Conference on Pattern Recognition (ICPR'06)**. IEEE, 2006. v. 3, p. 1063–1068. ISBN 0-7695-2521-0.

REINA, G. et al. Adaptive Kalman Filtering for GPS-based Mobile Robot Localization. In: **2007 IEEE International Workshop on Safety, Security and Rescue Robotics**. IEEE, 2007. p. 1–6. ISBN 978-1-4244-1568-7.

YI, J. et al. IMU-based localization and slip estimation for skid-steered mobile robots. In: **2007 IEEE/RSJ International Conference on Intelligent Robots and Systems**. IEEE, 2007. p. 2845–2850. ISBN 978-1-4244-0911-2.

NIN, M. C.; OSÓRIO, F. Navegação de robôs móveis autônomos e detecção de humanos baseada em sensor laser e câmera térmica. **Mostra Nacional de Robótica**, p. 1–6, 2011.

BARRERA, Alejandra. **Mobile Robots Navigation**. [S;I]: Intechopen, 2010. 682 p.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots**. [S.l.]: MIT press, 2011.

HENRY, Peter; KRAININ, Michael; HERBST, Evan; REN, Xiaofeng; FOX, Dieter. RGB-D mapping: using kinect-style depth cameras for dense 3d modeling of indoor environments. **The International Journal Of Robotics Research**, [S.L.], v. 31, n. 5, p. 647-663, 10 fev. 2012. SAGE Publications. http://dx.doi.org/10.1177/0278364911434148.

STENTZ, A. Optimal and efficient path planning for unknown and dynamic environments. **Carnegie Mellon Robotics Institute Technical Report** CMURI-TR-93-20. 1993.

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'**, [S.L.], p. 1-6, ago. 1997. IEEE Comput. Soc. Press. http://dx.doi.org/10.1109/cira.1997.613851.

SRINIVASA, Siddhartha S.; FERGUSON, Dave; HELFRICH, Casey J.; BERENSON, Dmitry; COLLET, Alvaro; DIANKOV, Rosen; GALLAGHER, Garratt; HOLLINGER, Geoffrey; KUFFNER, James; WEGHE, Michael Vande. HERB: a home exploring robotic butler. **Autonomous Robots**, [S.L.], v. 28, n. 1, p. 5-20, 17 nov. 2009. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s10514-009-9160-9.

SICILIANO, B.; KHATIB, O. **Springer handbook of robotics**. [S.l.]: Springer Science & Business Media, 2008

HORN, B. K. Closed-form solution of absolute orientation using unit quaternions. **JOSA A**, Optical Society of America, v. 4, n. 4, p. 629–642, 1987.

KELLY, A. **Mobile Robotics: Mathematics, Models, and Methods**. [S.l.]: Cambridge University Press, 2013.

BELLOTTO, N. et al. Appearance-based localization for mobile robots using digital zoom and visual compass. **Robotics and Autonomous Systems**, Elsevier, v. 56, n. 2, p. 143–156, 2008.

BARBOSA, Flávio Gabriel Oliveira. **SISTEMA DE LOCALIZAÇÃO, MAPEAMENTO E REGISTRO 3D PARA ROBÓTICA MÓVEL BASEADO EM TÉCNICAS DE VISÃO COMPUTACIONAL**. 2017. 140 f. Dissertação (Mestrado) - Curso de Pós-Graduação em Engenharia de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2017.

THRUN, S. et al. Robotic mapping: A survey. **Exploring artificial intelligence in the new millennium**, v. 1, p. 1–35, 2002.

SANCHES, V. L. M. **Um sistema de localização robótica para ambientes internos baseado em redes neurais.** 107 p. Tese (Doutorado) — Universidade de São Paulo, 2009.

THRUN, S. et al. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In: **AAAI/IAAI**. [S.l.: s.n.], 1998. p. 989–995.

BEKEY, G.A. Autonomous Robots - **From Biological Inspiration to Implementation and Control, Chp**.14. MIT Press, 2005.

RIBEIRO, M.I. **Localização em Robótica Móvel – Odometria**. Lisboa: Instituto Superior Técnico, Out. 1999.

CARVALHO FILHO, José G. N. de; MOLINA, Lucas; BENSEBAA, Kamel; CARVALHO, Elyson A. N.; FREIRE, Eduardo O.. ESTIMAÇÃO DE POSIÇÃO E ORIENTAÇÃO PARA ROBÔS MÓVEIS. **[S.I]**, São Cristóvão-Se, p. 1-6, jan. 2009.

HEINEN, F.; OSÓRIO, F. "HyCAR - A Robust Hybrid Control Architecture for Autonomous Robots". In: **HIS 2002 – Proc. Of Hybrid Intelligent Systems**, Santiago, Chile. Soft-Computing Systems - Amsterdam: IOS Press. V. 87, p. 830-840. 2002

FILLIAT, D.; MEYER, J.-A. Map-based navigation in mobile robots:: I. a review of localization strategies. **Cognitive Systems Research,** Elsevier, v. 4, n. 4, p. 243–282, 2003.

MEZENCIO, Rovilson. **Implementação do Método de Campos Potenciais para Navegação de Robôs Móveis Baseada em Computação Reconfigurável.** 2002. 112 f. Dissertação (Mestrado) - Curso de Ciência de Computação e Matemática Computacional, Instituto de Ciências Matemáticas e de Computação Icmc-Usp, São Carlos - Sp, 2002.

TOMATIS, N. Hybrid, metric-topological, mobile robot navigation. EPFL, 2001.

SCATENA, Jean Miler. **Implementação de mapas topológicos para navegação de robôs móveis baseadas em computação reconfigurável**. 2002. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2002. doi:10.11606/D.55.2002.tde-07022003-150324.

FOX, D.; BURGARD, W.; THRUN, S. Markov Localization for Mobile Robots in Dynamic Environments. **Journal of Artificial Intelligence Research**, v. 11, p. 391–427, 1999. ISSN 1076 - 9757.

THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. **Probabilistic Robotics**. Cambridge, Massachusetts: MIT Press, 2005. 647 p.

LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. **Autonomous robots**, Springer, v. 4, n. 4, p. 333–349, 1997.

KOUBAA, Anis; BENNACEUR, Hachemi; CHAARI, Imen; TRIGUI, Sahar; AMMAR, Adel; SRITI, Mohamed-Foued; ALAJLAN, Maram; CHEIKHROUHOU, Omar; JAVED, Yasir. **Robot Path Planning and Cooperation**: foundations, algorithms and experimentations. Saudi Arabia: Springer, 2018. 201 p.

CHONG, Kok Seng; KLEEMAN, L.. Accurate odometry and error modelling for a mobile robot. **Proceedings Of International Conference On Robotics And Automation**, Albuquerque, Usa, p. 2783-2788, 1997. IEEE. http://dx.doi.org/10.1109/robot.1997.606708.

LATOMBE, J-C.; BARRAQUAND, J. "Robot Motion Planning: A Distributed Presentation Approach." **International Journal of Robotics Research**, 10: 628–649, 1991.

JACOBS, R.; CANNY, J. "Planning Smooth Paths for Mobile Robots," in **Proceeding. of the IEEE Conference on Robotics and Automation**, IEEE Press,1989, pp. 2–7.

SCHWARTZ, J.T.; SCHARIR M.; HOPEROFT J., **Planning Geometry and Complexity of Robot Motion**, NJ, Norwood: Ablex Publishing Corporation, 1987.

TOPIWALA, Anirudh; INANI, Pranav; KATHPAL, Abhishek. Frontier Based Exploration for Autonomous Robot. **[S.L.]**, Maryland, Usa, p. 1-7, jan. 2018.

USLU, Erkan; CAKMAK, Furkan; BALCILAR, Muhammet; AKINCI, Attila; AMASYALI, M. Fatih; YAVUZ, Sirma. Implementation of frontier-based exploration algorithm for an autonomous robot. **2015 International Symposium On Innovations In Intelligent Systems And Applications (Inista)**, [S.L.], p. 1-7, set. 2015. IEEE. http://dx.doi.org/10.1109/inista.2015.7276723.

VERBIEST K., BERRABAH S.A., COLON E. (2015) Autonomous Frontier Based Exploration for Mobile Robots. In: Liu H., Kubota N., Zhu X., Dillmann R., Zhou D. (eds)

Intelligent Robotics and Applications. ICIRA 2015. Lecture Notes in Computer Science, vol 9246. Springer.

KEIDAR, Matan; KAMINKA, Gal A.. Robot Exploration with Fast Frontier Detection: theory and experiments. **Proceedings Of The 11Th International Conference On Autonomous Agents And Multiagent Systems (Aamas 2012)**, Valencia, Spain, p. 1-8, jun. 2012.

TUYTELAARS, T.; MIKOLAJCZYK, K. Local Invariant Feature Detectors: A Survey. **Foundations and Trends® in Computer Graphics and Vision**, Now Publishers Inc., v. 3, n. 3, p. 177–280, 2007. ISSN 1572-2740. Disponível em: <http://www.nowpublishers.com/article/Details/CGV-017>.

MIKOLAJCZYK, K.; SCHMID, C. A performance evaluation of local descriptors. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 27, n. 10, p. 1615–1630, 2005.

SCHMID, Cordelia; MOHR, Roger; BAUCKHAGE, Christian. Evaluation of Interest Point Detectors. **International Journal Of Computer Vision**, [S.L.], v. 37, n. 2, p. 151-172, 2000. Springer Science and Business Media LLC. http://dx.doi.org/10.1023/a:1008199403446.

LANGRIDGE, D.J. Curve encoding and the detection of discontinuities. **Computer Graphics And Image Processing**, [S.L.], v. 20, n. 1, p. 58-71, set. 1982. Elsevier BV. http://dx.doi.org/10.1016/0146-664x(82)90073-9.

HARALICK, Robert M.; SHAPIRO, Linda G.. **Computer and Robot Vision**: volume 2. [S;I]: Addison-Wesley Publishing Company, 1993. 630 p.

COOPER, J.; VENKATESH, S.; KITCHEN, L.. Early jump-out corner detectors. **Proceedings. 1991 Ieee Computer Society Conference On Computer Vision And Pattern Recognition**, [S.L.], p. 1-2, ago. 2002. IEEE Comput. Sco. Press. http://dx.doi.org/10.1109/cvpr.1991.139783.

ROSTEN, Edward; DRUMMOND, Tom. Machine Learning for High-Speed Corner Detection. **Computer Vision – Eccv 2006**, [S.L.], p. 430-443, 2006. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/11744023_34.

QUINLAN, J. R.. Induction of decision trees. **Machine Learning**, [S.L.], v. 1, n. 1, p. 81-106, mar. 1986. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/bf00116251.

CALONDER, M. et al. Brief: Binary robust independent elementary features. In: SPRINGER. **European conference on computer vision**. [S.l.], 2010. p. 778–792.

JOLLIFFE, I. Principal Component Analysis. In: **Wiley StatsRef: Statistics Reference Online**. Chichester, UK: John Wiley & Sons, Ltd, 2014. Disponível em: <http://doi.wiley.com/10.1002/9781118445112.stat06472>.

LI, M.; YUAN, B. 2d-lda: A statistical linear discriminant analysis for image matrix. **Pattern Recognition Letters**, Elsevier, v. 26, n. 5, p. 527–532, 2005.

KULIS, B.; GRAUMAN, K. Kernelized locality-sensitive hashing for scalable image search. In: **2009 IEEE 12th International Conference on Computer Vision**. IEEE, 2009. p. 2130–2137. ISBN 978-1-4244-4420-5. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5459466>.

RUBLEE, E. et al. Orb: An efficient alternative to sift or surf. In: IEEE. **Computer Vision (ICCV), 2011 IEEE international conference** on. [S.l.], 2011. p. 2564–2571.

ROSIN, P. L. Measuring corner properties. **Computer Vision and Image Understanding**, Elsevier, v. 73, n. 2, p. 291–307, 1999.

IKEDO, W. N.; COLOMBINI, E. L.. **Odometria visual para robôs**. Campinas: Universidade Estadual de Campinas, 2017. 12 p.

ENGEL, Jakob; KOLTUN, Vladlen; CREMERS, Daniel. Direct Sparse Odometry. **Ieee Transactions On Pattern Analysis And Machine Intelligence**, [S.L.], v. 40, n. 3, p. 611-625,

1 mar. 2018. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tpami.2017.2658577.

DIAS, Luís António Pires. **Odometria Visual usando imagens RGB-D**. 2013. 68 f. Dissertação (Mestrado) - Curso de Engenharia Electrotécnica e de Computadores, Departamento de Engenharia Electrotécnica e de Computadores, Universidade de Coimbra, Coimbra, 2013.

FRAUNDORFER, F.; SCARAMUZZA, D.; POLLEFEYS, M.. A constricted bundle adjustment parameterization for relative scale estimation in visual odometry. **In Robotics And Automation (Icra), 2010 Ieee International Conference**, [S.I], p. 1899-1904, 2010.

FISCHLER, Martin A.; BOLLES, Robert C.. Random sample consensus. **Communications Of The Acm**, [S.L.], v. 24, n. 6, p. 381-395, jun. 1981. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/358669.358692.

QUIGLEY, Morgan; GERKEY, Brian; CONLEY, Ken; FAUST, Josh; FOOTE, Tully; LEIBS, Jeremy; BERGER, Eric; WHEELER, Rob; NG, Andrew. ROS: an open-source Robot Operating System. **Conference: Icra Workshop On Open Source Software**, [S.I], p. 1-6, jan. 2009.

MONTEMERLO, M.; ROY, N.; THRUN, S.. Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) Toolkit. **Proc. Of The Ieee/Rsj Intl. Conf. On Intelligent Robots And Systems (Iros)**, Las Vegas, Nevada, p. 2436-2441, out. 2003.

MIKOLAJCZYK, K. et al. A performance evaluation of local descriptors. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 27, n. 10, p. 1615–1630, 2005. ISSN 0162-8828.Disponível em: <http://doi.ieeecomputersociety.org/10.110910.1109/TPAMI.2005.188>.

FREEMAN, W. T.; ADELSON, E. H. The design and use of steerable filters. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 13, n. 9, p. 891–906, 1991.

KE, Y.; SUKTHANKAR, R. Pca-sift: A more distinctive representation for local image descriptors. In: IEEE. **Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on**. [S.l.], 2004. v. 2, p. II–506.

KOENDERINK, J. J.; DOORN, A. J. van. Representation of local geometry in the visual system. **Biological cybernetics**, Springer, v. 55, n. 6, p. 367–375, 1987.

BELONGIE, S.; MALIK, J.; PUZICHA, J. Shape matching and object recognition using shape contexts. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 24, n. 4, p. 509–522, 2002.

LAZEBNIK, S.; SCHMID, C.; PONCE, J. Sparse texture representations using affine-invariant neighborhoods. In: CITESEER. **In Proc. IEEE Conf. Comp. Vision Patt. Recog**. [S.l.], 2003.

GOOL, L. V.; MOONS, T.; UNGUREANU, D. Affine/photometric invariants for planar intensity patterns. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 1996. p. 642–651.

SCHAFFALITZKY, F.; ZISSERMAN, A. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2002. p. 414–431.

DWARAKANATH, Deepak; EICHHORN, Alexander; HALVORSEN, Pål; GRIWODZ, Carsten. Evaluating performance of feature extraction methods for practical 3D imaging systems. **Proceedings Of The 27Th Conference On Image And Vision Computing New Zealand - Ivcnz '12**, [S.L.], p. 1-6, 2012. ACM Press. http://dx.doi.org/10.1145/2425836.2425887.

BEKELE, Dagmawi; TEUTSCH, Michael; SCHUCHERT, Tobias. Evaluation of binary keypoint descriptors. **2013 Ieee International Conference On Image Processing**, [S.L.], p. 1-5, set. 2013. IEEE. http://dx.doi.org/10.1109/icip.2013.6738753.

LOWE, David G.. Distinctive Image Features from Scale-Invariant Keypoints. **International Journal Of Computer Vision**, [S.L.], v. 60, n. 2, p. 91-110, nov. 2004. Springer Science and Business Media LLC. http://dx.doi.org/10.1023/b:visi.0000029664.99615.94.

SANDE, Koen e A van de; GEVERS, T; SNOEK, Cees G M. Evaluating Color Descriptors for Object and Scene Recognition. **Ieee Transactions On Pattern Analysis And Machine Intelligence**, [S.L.], v. 32, n. 9, p. 1582-1596, set. 2010. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tpami.2009.154.

BRITO, Darlan Nunes; PÁDUA, Flávio Luis Cardeal; LOPES, Aldo Peres Campos e; DALIP, Daniel Hasan. Análise comparativa de detectores e descritores de características locais em imagens no âmbito do problema de autocalibração de câmeras. **Revista Brasileira de Computação Aplicada**, [S.L.], v. 8, n. 3, p. 85, 8 nov. 2016. UPF Editora. http://dx.doi.org/10.5335/rbca.v8i3.6166.

FONTES, Afonso Henriques; MAIA, José Everardo Bessa. ODOMETRIA VISUAL PARA CÂMERAS RGB-D EM MOVIMENTO. **XXI Congresso Brasileiro de Automática - Cba2016**, Fortaleza/Ce, p. 1-6, 2016.

ALADEM, Mohamed; RAWASHDEH, Samir. Lightweight Visual Odometry for Autonomous Mobile Robots. **Sensors**, [S.L.], v. 18, n. 9, p. 2837, 28 ago. 2018. MDPI AG. http://dx.doi.org/10.3390/s18092837.

ZHAN, Huangying; WEERASEKERA, Chamara Saroj; BIAN, Jia-Wang; REID, Ian. Visual Odometry Revisited: what should be learnt?. **2020 Ieee International Conference On Robotics And Automation (Icra)**, [S.L.], p. 4203-4211, maio 2020. IEEE. http://dx.doi.org/10.1109/icra40945.2020.9197374.

XIE, Xiuchuan; YANG, Tao; NING, Yajia; ZHANG, Fangbing; ZHANG, Yanning. A Monocular Visual Odometry Method Based on Virtual-Real Hybrid Map in Low-Texture Outdoor Environment. **Sensors**, [S.L.], v. 21, n. 10, p. 3394, 13 maio 2021. MDPI AG. http://dx.doi.org/10.3390/s21103394.

DESOUZA, G. N.; KAK, A. C. Vision for mobile robot navigation: A survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 24, n. 2, p. 237–267, 2002.

THRUN, S. Probabilistic algorithms in robotics. **Ai Magazine**, v. 21, n. 4, p. 93, 2000.

SIMMONS, R.; KOENIG, S. Probabilistic robot navigation in partially observable environments. In: **IJCAI**. [S.l.: s.n.], 1995. v. 95, p. 1080–1087.

KROTKOV, E. Mobile robot localization using a single image. In: IEEE. **Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on**. [S.l.], 1989. p. 978–983.

ATIYA, S.; HAGER, G. D. Real-time vision-based robot localization. **IEEE Transactions on Robotics and Automation**, IEEE, v. 9, n. 6, p. 785–800, 1993.

COX, I. J.; LEONARD, J. J. Modeling a dynamic environment using a bayesian multiple hypothesis approach. **Artificial Intelligence**, Elsevier, v. 66, n. 2, p. 311–344, 1994.

BLAKE, A.; ISARD, M. The condensation algorithm-conditional density propagation and applications to visual tracking. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1997. p. 361–367.

DELLAERT, F. et al. Monte Carlo localization for mobile robots. In: **IEEE. Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on**. [S.l.], 1999. v. 2, p. 1322–1328.

SUGIHARA, K. Some location problems for robot navigation using a single camera. **Computer vision, graphics, and image processing**, Elsevier, v. 42, n. 1, p. 112–129, 1988.

SAURER, Olivier; FRAUNDORFER, Friedrich; POLLEFEYS, Marc. Visual localization using global visual features and vanishing points. **Computer Vision And Geometry Group**, Eth Zürich, Switzerland, p. 1-9, 2010.

OLIVA, Aude; TORRALBA, Antonio. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. **International Journal Of Computer Vision**, [S.L.], v. 42, n. 3, p. 145-175, 2001. Springer Science and Business Media LLC. http://dx.doi.org/10.1023/a:1011139631724.

AULINAS, Josep; PETILLOT, Yvan; SALVI, Joaquim; XAVIER, Lladó. The SLAM problem: a survey. **Frontiers In Artificial Intelligence And Applications**, [S.L.], v. 184, n. , p. 363-371, 2008. IOS Press. http://dx.doi.org/10.3233/978-1-58603-925-7-363.

KUMMERLE, Rainer; GRISETTI, Giorgio; BURGARD, Wolfram. Simultaneous calibration, localization, and mapping. **2011 Ieee/Rsj International Conference On Intelligent Robots And Systems**, [S.L.], p. 1-6, set. 2011. IEEE. http://dx.doi.org/10.1109/iros.2011.6094817.

STROM, Daniel Perea; NENCI, Fabrizio; STACHNISS, Cyrill. Predictive exploration considering previously mapped environments. **2015 Ieee International Conference On Robotics And Automation (Icra)**, [S.L.], p. 1-6, maio 2015. IEEE. http://dx.doi.org/10.1109/icra.2015.7139574.

SANTOS, João Machado; PORTUGAL, David; ROCHA, Rui P.. An evaluation of 2D SLAM techniques available in Robot Operating System. **2013 Ieee International Symposium On Safety, Security, And Rescue Robotics (Ssrr)**, Sweden, p. 1-7, 2013.

VALENCIA, Rafael; MORTA, Marti; ANDRADE-CETTO, Juan; PORTA, Josep M.. Planning Reliable Paths With Pose SLAM. **Ieee Transactions On Robotics**, [S.L.], v. 29, n. 4, p. 1050-1059, ago. 2013. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tro.2013.2257577.

ALEKSANDROVICH, Y. D. et al. Mobile robot navigation based on artificial landmarks with machine vision system. **World Applied Sciences Journal**, v. 24, n. 11, p. 1467–1472, 2013.

SE, S.; LOWE, D.G.; LITTLE, J.J.. Vision-based global localization and mapping for mobile robots. **Ieee Transactions On Robotics**, [S.L.], v. 21, n. 3, p. 364-375, jun. 2005. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tro.2004.839228.

STURM, Jrgen; ENGELHARD, Nikolas; ENDRES, Felix; BURGARD, Wolfram; CREMERS, Daniel. A benchmark for the evaluation of RGB-D SLAM systems. **2012 Ieee/Rsj International Conference On Intelligent Robots And Systems**, [S.L.], p. 1-8, out. 2012. IEEE. http://dx.doi.org/10.1109/iros.2012.6385773.

HENRY, Peter; HERBST, Evan; KRAININ, Michael; REN, Xiaofeng. RGB-D Mapping: Using Kinect-Style Depth Cameras for Dense 3D Modeling of Indoor Environments. **The International Journal Of Robotics Research**, Seattle, p. 647-663, abr. 2012.

LIN, Rui; WANG, Zhenhua; SUN, Rongchuan; SUN, Lining. Vision-based mobile robot localization and mapping using the PLOT features. **2012 Ieee International Conference On Mechatronics And Automation**, [S.L.], p. 1-7, ago. 2012. IEEE. http://dx.doi.org/10.1109/icma.2012.6285115.

OLIVER, Ayrton; KANG, Steven; WÜNSCHE, Burkhard C.; MACDONALD, Bruce. Using the Kinect as a navigation sensor for mobile robotics. **Proceedings Of The 27Th Conference On Image And Vision Computing New Zealand - Ivcnz '12**, [S.L.], p. 1-6, 2012. ACM Press. http://dx.doi.org/10.1145/2425836.2425932.

KLASER, Rafael Luiz; OSÓRIO, Fernando Santos; WOLF, Denis. Vision-Based Autonomous Navigation with a Probabilistic Occupancy Map on Unstructured Scenarios. **2014 Joint Conference On Robotics: Sbr-Lars Robotics Symposium And Robocontrol**, São Carlos, Brasil, p. 1-5, out. 2014.

BONIN-FONT, Francisco; ORTIZ, Alberto; OLIVER, Gabriel. Visual Navigation for Mobile Robots: a survey. **Journal Of Intelligent And Robotic Systems**, [S.L.], v. 53, n. 3, p. 263-296, 1 maio 2008. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s10846-008-9235-4.

ENDRES, Felix; HESS, Jurgen; STURM, Jurgen; CREMERS, Daniel; BURGARD, Wolfram. 3-D Mapping With an RGB-D Camera. **Ieee Transactions On Robotics**, [S.L.], v. 30, n. 1, p.

177-187, fev. 2014. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tro.2013.2279412.

LABBE, M.; MICHAUD, F. Online global loop closure detection for large-scale multi-session graph-based slam. In: IEEE. **Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on**. [S.l.], 2014. p. 2661–2666.

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

GARCIA-FIDALGO, Emilio; ORTIZ, Alberto. Probabilistic Appearance-Based Mapping and Localization Using Visual Features. **Pattern Recognition And Image Analysis**, [S.L.], p. 277-285, 2013. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-38628-2_33.

NASEER, T.; SPINELLO, L.; BURGARD, W.; STACHNISS, C. Robust Visual Robot Localization Across Seasons Using Network Flows. **Proceedings of the AAAI Conference on Artificial Intelligence**, *[S. l.]*, v. 28, n. 1, 2014.

KAMBHAMPATI, S.; DAVIS, L.. Multiresolution path planning for mobile robots. **Ieee Journal On Robotics And Automation**, [S.L.], v. 2, n. 3, p. 135-145, 1986. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/jra.1986.1087051.

SIMMONS, Reid; KOENIG, Sven. Probabilistic Robot Navigation in Partially Observable Environments. **Proceedings Of 14Th International Joint Conference On Artificial Intelligence (Ijcai '95)**, [S.I], p. 1080-1087, jul. 1995.

KAVRAKI, L.e.; SVESTKA, P.; LATOMBE, J.-C.; OVERMARS, M.H.. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. **Ieee Transactions On Robotics And Automation**, [S.L.], v. 12, n. 4, p. 566-580, 1996. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/70.508439.

MEIKLE, S.; YATES, R.. Computer vision algorithms for autonomous mobile robot map building and path planning. **Proceedings Of The Thirty-First Hawaii International**

Conference On System Sciences, [S.L.], p. 1-10, ago. 2002. IEEE Comput. Soc. http://dx.doi.org/10.1109/hicss.1998.656291.

BORTOFF, S.A.. Path planning for UAVs. **Proceedings Of The 2000 American Control Conference. Acc (Ieee Cat. No.00Ch36334)**, [S.L.], p. 1-5, 2000. IEEE. http://dx.doi.org/10.1109/acc.2000.878915.

SIAGIAN, Christian; CHANG, Chin Kai; ITTI, Laurent. Autonomous Mobile Robot Localization and Navigation Using a Hierarchical Map Representation Primarily Guided by Vision. **Journal Of Field Robotics**, [S.L.], v. 31, n. 3, p. 408-440, 7 abr. 2014. Wiley. http://dx.doi.org/10.1002/rob.21505.

MAC, Thi Thoa; COPOT, Cosmin; TRAN, Duc Trung; KEYSER, Robin de. Heuristic approaches in robot path planning: a survey. **Robotics And Autonomous Systems**, [S.L.], v. 86, p. 13-28, dez. 2016. Elsevier BV. http://dx.doi.org/10.1016/j.robot.2016.08.001.

BENO, Peter; PAVELKA, Vladimir; DUCHON, Frantiek; DEKAN, Martin. Using Octree Maps and RGBD Cameras to Perform Mapping and A* Navigation. **2016 International Conference On Intelligent Networking And Collaborative Systems (Incos)**, [S.L.], p. 1-8, set. 2016. IEEE. http://dx.doi.org/10.1109/incos.2016.107.

SONG, Rui; LIU, Yuanchang; BUCKNALL, Richard. Smoothed A* algorithm for practical unmanned surface vehicle path planning. **Applied Ocean Research**, [S.L.], v. 83, p. 9-20, fev. 2019. Elsevier BV. http://dx.doi.org/10.1016/j.apor.2018.12.001.

ASADI, Khashayar; CHEN, Pengyu; HAN, Kevin; WU, Tianfu; LOBATON, Edgar. Real-Time Scene Segmentation Using a Light Deep Neural Network Architecture for Autonomous Robot Navigation on Construction Sites. **Computing In Civil Engineering 2019**, [S.L.], p. 1-8, 13 jun. 2019. American Society of Civil Engineers. http://dx.doi.org/10.1061/9780784482438.041.

POPOVIć, Marija; VIDAL-CALLEJA, Teresa; HITZ, Gregory; CHUNG, Jen Jen; SA, Inkyu; SIEGWART, Roland; NIETO, Juan. An informative path planning framework for UAV-based

terrain monitoring. **Autonomous Robots**, [S.L.], v. 44, n. 6, p. 889-911, 4 fev. 2020. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s10514-020-09903-2.

PANDA, Madhusmita; DAS, Bikramaditya; SUBUDHI, Bidyadhar; PATI, Bibhuti Bhusan. A Comprehensive Review of Path Planning Algorithms for Autonomous Underwater Vehicles. **International Journal Of Automation And Computing**, [S.L.], v. 17, n. 3, p. 321-352, 4 jan. 2020. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s11633-019-1204-9.

ZAMAN, Safdar; SLANY, Wolfgang; STEINBAUER, Gerald. ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues. **2011 Saudi International Electronics, Communications And Photonics Conference (Siecpc)**, [S.L.], p. 1-5, abr. 2011. IEEE. http://dx.doi.org/10.1109/siecpc.2011.5876943.

MOORE, Thomas; STOUCH, Daniel. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. **Intelligent Autonomous Systems 13**, [S.L.], p. 335-348, 3 set. 2015. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-08338-4_25.

TAKAYA, Kenta; ASAI, Toshinori; KROUMOV, Valeri; SMARANDACHE, Florentin. Simulation environment for mobile robots testing using ROS and Gazebo. **2016 20Th International Conference On System Theory, Control And Computing (Icstcc)**, [S.L.], p. 1-6, out. 2016. IEEE. http://dx.doi.org/10.1109/icstcc.2016.7790647.

SILVA, Bruno M. F. da; XAVIER, Rodrigo S.; NASCIMENTO, Tiago P. do; GONCALVES, Luiz M.G.. Experimental evaluation of ROS compatible SLAM algorithms for RGB-D sensors. **2017 Latin American Robotics Symposium (Lars) And 2017 Brazilian Symposium On Robotics (Sbr)**, [S.L.], p. 1-6, nov. 2017. IEEE. http://dx.doi.org/10.1109/sbr-lars-r.2017.8215331.

MARIN-PLAZA, Pablo; HUSSEIN, Ahmed; MARTIN, David; LAESCALERA, Arturo de. Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. **Journal Of Advanced Transportation**, [S.L.], v. 2018, p. 1-10, 2018. Hindawi Limited. http://dx.doi.org/10.1155/2018/6392697

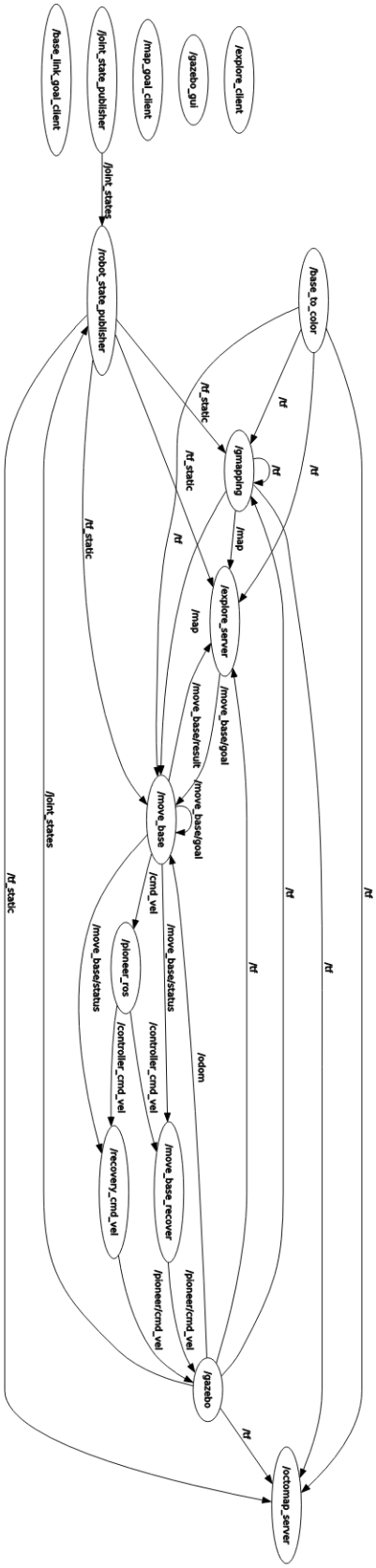
DAYANAND, V.; RAHUL, K. Sharma; GIREESH, T. Kumar. Comparative Study of Algorithms for Frontier based Area Exploration and Slam for Mobile Robots. **International Journal Of Computer Applications**, [S.L.], v. 77, n. 8, p. 37-42, 18 set. 2013. Foundation of Computer Science. http://dx.doi.org/10.5120/13417-1086.

USLU, Erkan; CAKMAK, Furkan; BALCILAR, Muhammet; AKINCI, Attila; AMASYALI, M. Fatih; YAVUZ, Sirma. Implementation of frontier-based exploration algorithm for an autonomous robot. **2015 International Symposium On Innovations In Intelligent Systems And Applications (Inista)**, [S.L.], p. 1-7, set. 2015. IEEE. http://dx.doi.org/10.1109/inista.2015.7276723.

VERBIEST, K.; BERRABAH, S. A.; COLON, E.. Autonomous Frontier Based Exploration for Mobile Robots. **Intelligent Robotics And Applications**, [S.L.], p. 3-13, 2015. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-22873-0_1.

GRISETTIY, G.; STACHNISS, C.; BURGARD, W.. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. **Proceedings Of The 2005 Ieee International Conference On Robotics And Automation**, Barcelona, Spain, p. 2432-2437, 2005. IEEE. http://dx.doi.org/10.1109/robot.2005.1570477.

ANEXO A – Grafo Simulação 1



ANEXO B – Grafo Simulação 2