UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO HENRIQUE PAIOLA

SISTEMA DE NAVEGAÇÃO PARA ROBÔS MÓVEIS BASEADO EM SLAM USANDO SENSORES DE BAIXO CUSTO

BAURU
2019

PEDRO HENRIQUE PAIOLA

SISTEMA DE NAVEGAÇÃO PARA ROBÔS MÓVEIS BASEADO EM SLAM USANDO SENSORES DE BAIXO CUSTO

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Humberto Ferasoli Filho

BAURU
2019

Pedro Henrique Paiola SISTEMA DE NAVEGAÇÃO PARA ROBÔS MÓVEIS BASEADO EM SLAM USANDO SENSORES DE BAIXO CUSTO/ Pedro Henrique Paiola. – Bauru, 2019- 78 p. : il. (algumas color.) ; 30 cm.
Orientador: Prof. Dr. Humberto Ferasoli Filho
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Ciência da Computação, 2019.
1. Robótica Móvel 2. Navegação 3. Mapeamento 4. Localização 5. SLAM

Pedro Henrique Paiola

SISTEMA DE NAVEGAÇÃO PARA ROBÔS MÓVEIS BASEADO EM SLAM USANDO SENSORES DE BAIXO CUSTO

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Humberto Ferasoli Filho
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profª. Drª. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Renê Pegoraro
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, _____ de _____ de _____.

Dedico este trabalho à Deus, que me deu forças para concluir este projeto, e a minha família, que sempre me acompanhou durante toda minha caminhada.

Agradecimentos

Agradeço primeiramente a minha família, especialmente aos meus pais, que foram fundamentais na minha formação e na minha dedicação aos estudos; e ao meu padrinho, que partilha da minha paixão pelo conhecimento e que acompanhou o desenvolvimento deste projeto.

Agradeço a todos os professores que me instruíram ao longo da minha vida, pois eles me forneceram a bagagem que me permitiu chegar aqui.

Agradeço, em especial, ao meu orientador, sem o qual este projeto não teria sido possível, sempre me mostrando quais caminhos eu poderia trilhar.

Também agradeço aos meus amigos e colegas, que me acompanharam durante esta trajetória me dando forças, compartilhando conhecimento e bons momentos.

Resumo

Um dos principais problemas da robótica móvel é o problema da navegação, que consiste em permitir que um robô móvel se locomova, navegue, de um ponto a outro, dentro do ambiente em que ele está inserido. Uma das formas de resolver este problema é a partir de uma abordagem deliberativa, traçando uma rota até o ponto objetivo utilizando um mapa do ambiente. Mas, se o robô não possui um mapa prévio do ambiente, para conseguir adotar tal abordagem, ele deve ser capaz de construí-lo. Para isso, o robô deve ser capaz de mapear e se localizar no ambiente simultaneamente, o que consiste em um problema clássico da robótica conhecido como SLAM (*Simultaneous Localization and Mapping*). Dessa forma, o objetivo deste trabalho é apresentar um sistema de mapeamento e navegação utilizando um robô com sensores de baixo custo. Os resultados obtidos pelos experimentos refletem problemas gerados pela limitação e imprecisão dos sensores, porém são satisfatórios o suficientes para perceber que robôs com baixo custo podem ser utilizados na execução de tarefas relativamente complexas.
Palavras-chave: Robótica Móvel. Navegação. Mapeamento. Localização. SLAM.

Abstract

One of the main problems of mobile robotics is the navigation problem, which allows a mobile robot to move around, navigate from one point to another within the environment in which it is inserted. One way to solve this problem is from a deliberative approach, tracing a route to the objective point using a map of the environment. But if the robot does not have a prior map of the environment, to be able to adopt this approach, it must be able to build it. To do this, the robot must be able to map and locate in the environment simultaneously, which is a classic robotics problem known as SLAM (Simultaneous Localization and Mapping). Thus, the objective of this project is to present a mapping and navigation system using a robot with low cost sensors. The results obtained by the experiments reflect problems generated by the limitation and inaccuracy of the sensors, but are satisfactory enough to realize that low cost robots can be used to perform relatively complex tasks.
Keywords: Mobile Robotics. Navigation. Mapping. Localization. SLAM.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

EKF	*Extended Kalman Filter*
GPS	*Global Positioning System*
MCL	*Monte Carlo Localization*
PSD	*Position Sensing Device*
SLAM	*Simultaneous Localization and Mapping*

Sumário

1 Introdução

Quando falamos de robótica móvel, estamos nos referindo a robôs capazes de se mover no ambiente, seja esse ambiente terrestre, aquático ou aéreo. Inerente a isso, surge então o problema da navegação de robôs móveis, que trata de como podemos fazer um robô se locomover, navegar, pelo ambiente livre de colisão.

Em primeiro lugar, é importante considerar as características do ambiente em que o robô deve navegar, pois essas características geram desafios com que o robô deve lidar, como desviar, evitar ou "driblar" os obstáculos. Dentre essas características é importante diferenciar ambientes estáticos de ambientes dinâmicos. Em ambientes estáticos os elementos são fixos, o único a se mover é o robô, enquanto nos dinâmicos temos a presença de objetos móveis ou seres se locomovendo pelo ambiente.

Se o ambiente é estático, ele pode ser previamente conhecido, ou seja, o robô possui um mapa desse ambiente. Dessa forma, considerando esse mapa, o problema da navegação se resume ao planejamento de uma trajetória do ponto inicial ao ponto objetivo. Se o mapa é desconhecido, pode-se considerar a possibilidade do próprio robô mapear esse ambiente, o que também é um desafio, e alvo desse trabalho. Por outro lado, se o ambiente é dinâmico, possuir um mapa, por si só, não é suficiente, o robô deve ser capaz de agir reativamente ao que é detectado pelos seus sensores.

Nesse trabalho é apresentado um sistema de navegação que considera um ambiente terrestre, plano e estático. Esse sistema permite o robô móvel se locomover dentro desse ambiente, ao mesmo tempo que faz o seu mapeamento, e depois permitindo-o planejar trajetórias e, por consequência, atingir um ponto objetivo mais rapidamente.

1.1 Justificativa

Os robôs móveis são usados nas mais diversas aplicações, como transporte de materiais em hospitais, em depósitos e no chão de fábrica, veículos agrícolas autônomos, veículos urbanos autônomos, guia em museus, cadeira de rodas autônomas (BECKER, 2015; BIGHETI, 2011) e na exploração de ambientes inóspitos e de difícil acesso ao ser humano, seja pelas dimensões (tubulações), riscos (materiais tóxicos ou explosivos) ou distância (exploração de planetas).

Em todas essas aplicações, temos que lidar com o problema da navegação de robôs, sendo necessário que o robô tenha capacidade de se localizar com uma precisão aceitável e se locomover no ambiente.

O uso de abordagens híbridas para resolver o problema da navegação é interessante por serem, normalmente, mais eficientes, não necessitando de um mapa prévio do ambiente

e mantendo a possibilidade de lidar com situações inesperadas e com o seu dinamismo. A abordagem implementada neste trabalho é baseada em um problema clássico da robótica, o SLAM, que consiste em mapear o ambiente ao mesmo tempo que realiza a estimativa da localização do robô.

Além disso, nesse trabalho foi colocado em prática conteúdos aprendidos em diversas disciplinas do curso de Bacharelado em Ciência da Computação, entre elas podemos citar Algoritmos, Estruturas de Dados, Técnicas de Programação, Circuitos Digitais, Microcontroladores e Inteligência Artificial.

1.2 Objetivos

1.2.1 Objetivo Geral

Implementar um sistema de navegação de robôs, segundo uma abordagem híbrida e usando técnicas de SLAM.

1.2.2 Objetivos Específicos

- Programar as funções básicas do robô, como comandos de locomoção, leitura dos sensores e comunicação Bluetooth.
- Implementar a comunicação do robô com o computador, permitindo o envio e recebimento de dados.
- Implementar o tratamento das informações do sensor ultrassônico e do sensor infravermelho, considerando seus erros e imprecisões.
- Implementar a solução de mapeamento do ambiente, considerando as técnicas de SLAM, assim como a representação gráfica desse mapeamento.
- Implementar algoritmos para a navegação do robô, considerando o mapa gerado.
- Realizar testes e avaliar os resultados da implementação.

1.3 Metodologia

Para a realização desse trabalho, será usado o robô desenvolvido para este trabalho, o qual foi batizado de Frank, constituído por uma placa Arduino e munido de um sensor ultrassônico e um sensor infravermelho acoplado à um servo motor, rodas para a locomoção do

robô, e um módulo Bluetooth para realizar a comunicação com o computador. Os componentes do Frank serão melhor detalhados no capítulo sobre o Hardware. Também será usado o computador do autor, munido de um processador i7-7500U 2.7GHz e 16 GB de memória RAM.

Esse trabalho foi dividido em basicamente quatro etapas.

A primeira etapa foi o estudo e aprofundamento sobre o problema da navegação de robôs e das soluções apresentadas na literatura, especialmente as de abordagens híbridas e que usam técnicas de SLAM. Nessa etapa também também foi realizado o estudo de diversos conceitos relacionados a navegação de robôs móveis e ao SLAM, que foram necessários para a resolução do nosso problema.

A segunda etapa foi a definição da solução a ser implementada, considerando o levantamento bibliográfico realizado, assim como a escolha das ferramentas utilizadas para realizar a implementação, levando em consideração as funcionalidades necessárias e o domínio do autor sobre as mesmas.

Na terceira etapa foi realizada a implementação da solução, assim como a realização de testes iniciais. Nesta fase foram desenvolvidos o sistema embarcado no robô e o sistema em computador, envolvendo o mapeamento do ambiente, a navegação do robô e a interface com o usuário. Essa etapa foi realizada de forma concomitante as duas primeiras, permitindo a verificação das soluções que melhor se adequavam ao robô utilizado.

Por último, na quarta etapa foram feitos os testes finais e a avaliação de desempenho da implementação realizada, verificando se ela atingiu seu objetivo.

1.4 Organização do Trabalho

Este trabalho, além desta introdução, está divido em mais 8 capítulos:

No Capítulo 2 o problema da navegação de robôs móveis é tratado mais profundamente, assim como as abordagens utilizadas para solucioná-lo.

O Capítulo 3 trata sobre o problema da localização e as classes das soluções empregadas, dando ênfase para a odometria e para a localização baseada em mapas.

O Capítulo 4 aborda o problema do mapeamento, descrevendo os tipos de mapas existentes, com foco nas grades de ocupação, utilizadas na solução de mapeamento empregada neste trabalho. Neste capítulo também é apresentado um modelo probabilístico o tratamento de dados dos sensores utilizados para o mapeamento.

O Capítulo 5 descreve o problema da localização e do mapeamento simultâneos, conhecido como SLAM. Nele é descrito os principais problemas e desafios encontrados neste processo, assim como são apresentadas algumas soluções clássicas.

O Capítulo 6 apresenta o robô utilizado neste trabalho, descrevendo cada um dos seus componentes principais.

O Capítulo 7 descreve o sistema implementado, descrevendo sua arquitetura geral e detalhando seus principais módulos.

O Capítulo 8 traz os resultados obtidos pelo sistema implementado, descrevendo o ambiente de testes, os experimentos realizados e os problemas e deficiências encontrados.

Por fim, o Capítulo 9 traz as conclusões deste trabalho, assim como sugestões de melhorias que podem ser exploradas em trabalhos futuros.

2 Navegação de Robôs Móveis

O tema central deste trabalho é a navegação de robôs móveis, dessa forma, é necessário uma definição clara do que é um robô móvel. Matarić (2007) bem define um robô, dizendo: "um robô é um sistema autônomo que existe em um mundo físico, pode perceber o ambiente a sua volta e pode atuar nele para alcançar certos objetivos". A seguir os conceitos envolvidos nesta definição são melhor detalhados:

- É um sistema autônomo: um robô deve ser capaz de tomar suas próprias decisões, não sendo controlado por um humano. Ele pode até receber instruções de seres humanos, mas não pode ser completamente controlado por eles. Sendo assim, máquinas teleoperadas não são consideradas robôs.

- Existe no mundo físico: um robô deve existir no mundo físico, no nosso mundo, e dessa forma estará sujeito as leis da física e aos desafios decorrentes disso. Um robô que existe apenas em uma simulação computacional nunca terá que lidar com as verdadeiras propriedades do mundo físico, porque por mais complexas que sejam as simulações, elas não conseguem ser totalmente fiéis ao mundo real.

- Pode perceber o ambiente a sua volta: um robô é munido de sensores, pelos quais ele obtém informações do mundo. Ressaltando o fato de que um robô deve existir no mundo físico, as informações sobre o ambiente só podem ser obtidas através de sensores, assim como feito por seres humanos e animais.

- Pode atuar no seu ambiente: um robô deve ser capaz de agir, respondendo aos estímulos sensoriais. Uma máquina que não age não é um robô. As ações que um robô pode tomar são diversas, o que torna a robótica uma área tão ampla.

- Atua para atingir objetivos: uma máquina que existe no mundo físico e obtém informações sobre ele, mas age de forma aleatória ou inútil, não é um robô, porque não usa as

> informações obtidas e sua capacidade de agir para fazer algo útil para si mesmo ou para os outros. Um robô deve ter um ou mais objetivos e atuar de forma a tentar alcança-los. Assim como há diversas formas de atuar no ambiente, são diversos os objetivos possíveis, dos mais simples aos mais complexos.

Pensando nessa definição, quando falamos do problema da navegação de robôs móveis estamos falando em robôs capazes de se locomover pelo ambiente, cujo o objetivo é obter sucesso no seu deslocamento, de um ponto inicial a um ponto objetivo, livre de colisões com obstáculos, fixos ou móveis. Mas várias dificuldades são encontradas nesse processo, especialmente devido ao fato do robô estar inserido no mundo real, sujeito as suas propriedades, e dispondo apenas de seus sensores para obter informações desse mundo, sendo que esses sensores apresentam erros e imprecisões. A complexidade envolvida no problema da navegação encontra-se na multiplicidade de situações com as quais o robô pode se deparar (MARCHI, 2001).

Em seu livro, Matarić (2007) discute os principais problemas que podem estar envolvidos no processo de navegação, dependendo do cenário em que o robô está inserido, sendo eles:

- Localização: determinar a localização do robô.
- Busca: procurar a localização do objetivo.
- Planejamento de trajetória: planejar uma rota da posição do robô até a posição do objetivo.
- Cobertura: ser capaz de cobrir toda a área de um ambiente.
- SLAM: realizar a localização e o mapeamento simultaneamente.

Estes problemas serão melhor explorados ao longo deste trabalho.

As soluções para o problema da navegação podem ser classificadas em três abordagens: deliberativa, reativa ou híbrida. A divisão entre essas abordagens leva em consideração as estratégias de controle empregadas, e que serão melhor detalhadas a seguir.

2.1 Abordagem deliberativa

As estratégias de controle deliberativo se baseiam em uma estratégia puramente simbólica, onde a semelhança entre o ambiente e o seu modelo no robô (como um mapa, por exemplo) deve ser precisa (SECCHI, 2008). Dessa forma, para a aplicação de uma abordagem planejada ou deliberativa, deve-se assumir que o ambiente é estático e conhecido (MARCHI,

2001). A partir disso, tem que se considerar os diferentes cenários em que o robô pode estar inserido e nos problemas que ele pode enfrentar.

O SLAM não é um item a ser considerado nesta abordagem, pois o mapa do ambiente já é conhecido e não deve sofrer alterações. Já problema da busca e da cobertura devem ser considerados, mas apenas se a localização do objetivo não for conhecida. O problema da localização do robô, por sua vez, deve ser tratado de qualquer forma, do contrário de nada adiantará possuir um mapa do ambiente.

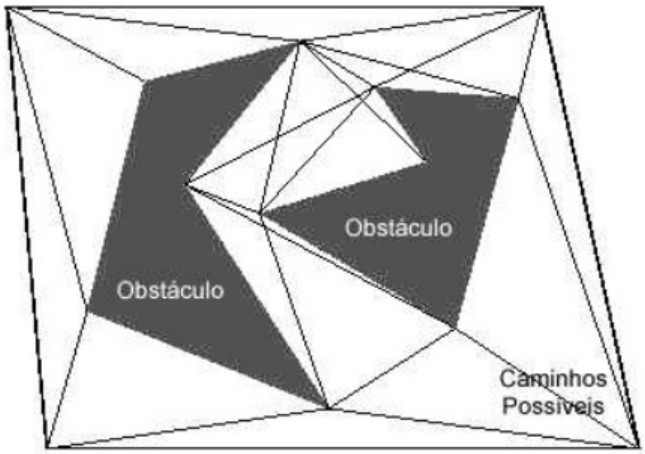
Por fim, se o robô é capaz de se localizar e conhece a posição do seu objetivo, resta apenas tratar o problema do planejamento de trajetória. E para planejar a trajetória, são utilizadas as informações armazenadas no mapa.

Existem diversos métodos para planejar uma trajetória entre dois pontos, e estes se dividem em três grandes classes de algoritmos: Métodos de Mapeamento de Caminhos (*Roadmap*), Métodos de Decomposição em Células e Método dos Campos Potenciais.

2.1.1 Métodos de Mapeamento de Caminhos

Os métodos desta classe buscam encontrar a conectividade do espaço livre no qual o robô está inserido, formando uma rede de curvas unidimensional. Tendo esta rede construída, o robô passa a possuir um conjunto de caminhos. Sendo assim, para planejar uma rota entre os pontos inicial e final basta busca nesta rede um subconjunto de caminhos que os conectem. Alguns métodos conhecidos desta classe são: grafos de visibilidade (exemplificado pela Figura 1), diagrama de Voronoi e método do caminho livre (MARCHI, 2001).

Figura 1 – Mapeamento de caminhos por grafo de visibilidade.



Fonte: Heinen (2002, p. 36)

2.1.2 Métodos de Decomposição em Células

Métodos de decomposição em células consistem em:

- Discretizar o ambiente em regiões, denominadas células;
- Construir um grafo não-dirigido, chamado de grafo de conectividade, onde os nós são as células e as arestas a relação de adjacência entre as células que os nós representam;
- Determinar as células inicial e final e buscar um caminho entre elas, utilizando algum método de busca no grafo de conectividade.
- Com a sequência de células encontradas, calcular o caminho dentro de cada célula que o robô deve efetivamente percorrer.

2.1.3 Método dos Campos Potenciais

Este método é baseado nos campos potenciais estudados pela Física. O robô passa a ser representado como uma partícula sob a influência de duas forças: uma força repulsiva, que o afasta dos obstáculos, e uma força atrativa, que o atrai para o objetivo.

A soma de todas as forças determina a direção e a velocidade do movimento da partícula, e a partir disso determina-se a rota do robô. Um dos motivos para a popularidade deste método é a sua simplicidade e elegância. Um método de campos potenciais simples pode ser implementado rapidamente e produz resultados aceitáveis (HEINEN, 2002).

Esse método, concebido para ser aplicado em abordagens reativas, possui uma deficiência, que é a possibilidade de convergir para mínimos locais. Algumas das soluções para este problema envolvem estratégias de busca, que faz o método perder sua característica reativa, pois o robô deve possuir um modelo do mundo, seja criado *a priori* (abordagem planejada) ou durante a execução (abordagem híbrida) (HEINEN, 2002; MARCHI, 2001).

2.2 Abordagem reativa

Visando trabalhar em ambientes reais, não conhecidos previamente pelo robô e dinâmicos, temos a abordagem reativa, na qual o robô toma decisões baseando-se nas leituras de seus sensores, buscando chegar ao ponto objetivo. Estratégias de controle reativo se baseiam em ações por reflexo, gerando ações de controle a partir de estímulos recebidos pelo robô (SECCHI, 2008).

Essa abordagem é oposta a abordagem planejada, uma vez que não há um plano prévio das ações, dispensando assim qualquer modelo do ambiente.

Essa capacidade reativa permite ao robô navegar em ambientes complexos, desconhecidos e dinâmicos, além de permitir a reação rápida a estímulos. Porém, como a capacidade cognitiva é ignorada e o robô não incorpora nenhuma habilidade de conhecimento do ambiente, sua aplicação em tarefas mais elaboradas se torna difícil. Além disso, não há nenhuma garantia que o caminho encontrado pelo robô será o melhor possível, e nem que ele de fato atingirá o objetivo.

2.3 Abordagem híbrida

Ambas as abordagens anteriores apresentam vantagens e desvantagens. A abordagem deliberativa apresenta formas para a integração do conhecimento do ambiente e a utilização deste ambiente para obter um plano de ação, porém para isso necessita do conhecimento prévio do ambiente, e falha quando temos um ambiente dinâmico. A abordagem reativa, por sua vez, responde rapidamente a estímulos sensoriais, sendo capaz de lidar com ambientes dinâmicos, mas não tendo um conhecimento global do ambiente, pode fazer com que o robô fique navegando aleatoriamente pelo ambiente, eventualmente atingindo o ponto objetivo. A abordagem híbrida visa unir as melhores características dessas abordagens e minimizar os problemas de ambas. Podemos, por exemplo, fazer com que o robô armazene as informações lidas pelos sensores, criando assim um mapa do ambiente, inicialmente desconhecido, que é consultado para o planejamento da trajetória (MARCHI, 2001).

Um exemplo de abordagem híbrida são os métodos que empregam estratégias de controle baseadas em comportamento. Nestes métodos são utilizados múltiplos módulos operando em paralelo, junto com técnicas de aprendizado para melhorar o desempenho do robô. Esses módulos são chamados por Brooks (1991) de comportamentos, que trabalham de forma paralela e assíncrona, permitindo a multiplicidade de objetivos, assim como a existência de diversos comportamentos que visam atender a um mesmo objetivo. Tais objetivos são atingidos de forma paralela, tendendo a um cenário final em que todos são atendidos simultaneamente. A tendência desta estratégia é conseguir algoritmos de controle confiáveis (característica da abordagem deliberativa) que tenham respostas rápidas aos estímulos (característica da abordagem reativa). Esta é uma estratégia robusta, onde a falha de um módulo não afeta os módulos independentes dele, mantendo o funcionamento do robô, ainda que sub-ótimo. Da mesma forma, a inserção de novos módulos independentes não muda a capacidade dos já implementados, garantindo a capacidade de adição de novas funcionalidades sem a exigência do retrabalho em todo o sistema (SECCHI, 2008).

Um exemplo de abordagem híbrida para o problema da navegação é a realização do mapeamento do ambiente antes da navegação do robô, permitindo que o planejamento prévio das ações a partir do mapa gerado, tal como na abordagem deliberativa, e a implementação de comportamentos reativos, que permitam o robô desviar de obstáculos dinâmicos ou que

não estavam presentes no mapa, conforme estes forem captados por seus sensores. Ou ainda, o mapa pode ser construído em tempo de execução, durante a navegação, com estratégia semelhante.

O objetivo deste trabalho foi implementar um sistema que seja capaz de mapear o ambiente para então utilizá-lo para navegação. Nos capítulos a seguir são tratados dois conceitos fundamentais para esta abordagem: Localização e Mapeamento. E por fim, como estes dois conceitos se relacionam.

3 Localização

O problema da localização consiste em determinar a posição do robô no ambiente em que ele está inserido. Ele pode ser visto como a busca da resposta a questão "Onde estou?", do ponto de vista do robô. De forma geral, o robô deve ser capaz de determinar suas coordenadas no plano ou no espaço, assim como sua orientação, a partir de um referencial. Diversos pesquisadores ressaltam a importância desse problema, Thrun et al. (2001), por exemplo, colocam o problema da localização como um problema chave para a Robótica Móvel. Em particular, considerando o escopo deste trabalho, ser capaz de determinar a localização do robô é fundamental tanto para o mapeamento do ambiente, quando para a navegação. Podemos classificar os métodos de localização em três categorias: Localização Relativa, Localização Absoluta e Fusão de Multi-sensores (SOUZA, 2008).

3.1 Localização Relativa

Nos métodos de localização relativa, também conhecida como *dead reckoning*, a estimativa da localização do robô é obtida a partir das estimativas de localizações anteriores. A desvantagem desses métodos é que o uso destas estimativas anteriores favorece a propagação e acúmulo de erros. Dentre os métodos de localização relativa, podemos citar a navegação inercial e a odometria.

3.1.1 Navegação Inercial

Esse método de localização utilizada as informações de giroscópios e acelerômetros para medir a orientação e aceleração. Os giroscópios são capazes de detectar pequenas acelerações na orientação do robô, enquanto os acelerômetros são capazes de detectar pequenas acelerações nos eixos cartesianos. A navegação inercial estima a localização do robô a partir da integração das informações dos sensores, ou seja, utilizando as estimativas anteriores junto com as informações sensoriais atuais para estimar a localização atual. Desta forma, este método tende

a ter um grande acúmulo de erros.

3.1.2 Odometria

A odometria é um dos métodos mais utilizados para determinar a localização de robô, sendo um método barato, com boa precisão em trechos pequenos e com uma alta taxa de amostragem (BORENSTEIN et al., 1996). Esse método determina a localização do robô pela integração incremental do movimento das suas rodas. Normalmente isso é feito utilizando-se *encoders* (sensores de rotação), que contam a quantidade de giros em cada roda, permitindo calcular a distância linear percorrida, assim como sua orientação.

Porém, sendo um método de localização relativa, em que a posição do robô é estimada a partir da integração das informações sobre o movimento das roda, é inevitável o acúmulo de erros (BORENSTEIN et al., 1996).

Em nosso robô, usaremos a odometria para estimar a sua localização, que posteriormente deve ser corrigida, como veremos mais adiante. Portanto, é importante detalhar melhor o funcionamento deste método, como a localização do robô é calculada e a que erros estamos sujeitos.

A seguir, apresentamos as fórmulas utilizadas para estimar a posição de um robô com duas rodas independentes de tração diferencial, baseado em Silva et al. (2018).

Em primeiro lugar, é necessário calcular a distância percorrida pela roda esquerda e a distância percorrida pela roda direita, o que pode ser feito utilizando a seguinte fórmula:

$$d = 2.\pi.R.\frac{\Delta tick}{N} \tag{3.1}$$

em que
$N =$ número de pulsos por volta dos *encoders*;
$\Delta tick =$ diferença da quantidade de pulsos atuais e a quantidade de pulsos da última medição;
$R =$ raio da roda.

A partir disso, a distância percorrida pelo ponto central do robô consiste na média das distâncias percorridas pelas duas rodas:

$$d_{centro} = \frac{d_{esquerda} + d_{direita}}{2} \tag{3.2}$$

Também podemos calcular a variação $\Delta\theta$ do ângulo de orientação do robô, com base nas distâncias percorridas por cada roda:

$$\Delta\theta = \frac{d_{direita} - d_{esquerda}}{b} \tag{3.3}$$

em que b é a distância entre as rodas (comprimento do eixo).

Sendo assim, a orientação atual θ' do robô, a partir da orientação anterior θ:

$$\theta' = \theta + \Delta\theta \tag{3.4}$$

E, por fim, a posição atual do robô pode ser obtida pelas fórmulas:

$$x' = x + d_{centro} cos(\theta') \tag{3.5}$$

$$y' = y + d_{centro} sen(\theta') \tag{3.6}$$

Como foi dito anteriormente, a odometria está sujeita a erros, e esses erros podem ser classificados como sistemáticos ou não-sistemáticos. Os erros sistemáticos são gerados devido às incertezas nos parâmetros do modelo cinemático do robô, como a diferença entre o diâmetro das rodas, comprimento do eixo diferente do nominal e taxa de amostragem finita dos *encoders*. Os erros não sistemáticos, por sua vez, são devidos a situações inesperadas, como irregularidades da superfície, escorregamento das rodas, falta de contato das rodas com a superfície e colisão (BORENSTEIN et al., 1996; SOUZA, 2008).

Na literatura são apresentados diversos métodos que visam modelar e corrigir os erros de odometria. Chenavier e Crowley (1992 apud SOUZA, 2008), por exemplo, apresentaram uma forma simplificada para modelagem dos erros de odometria baseada em experimentos, onde a partir de observações do comportamento do robô ao se movimentar foram deduzidos coeficientes que, multiplicados pelos deslocamentos realizados, fornecem uma estimativa aos erros de odometria. Já em Borenstein et al. (1996) encontramos um procedimento bastante conhecido e adotado para a medição e correção de erros sistemáticos, chamado UMBmark (*University of Michigan Benchmark*). Neste procedimento, o robô realiza uma trajetória quadrangular, aferindo os valores fornecidos pela odometria e os valores reais, medidos por um método de localização absoluta. A partir dos dados obtidos, o procedimento segue uma formulação matemática que fornece a estimativa dos erros de odometria e os parâmetros para sua calibração.

Alguns autores também partem de uma abordagem estatística na modelagem dos erros de odometria. Martinelli e Siegwart (2003 apud SOUZA, 2008), por exemplo, utilizaram um Filtro de Kalman Aumentado para estimar um vetor de estados contendo a configuração do robô e parâmetros que caracterizam os erros sistemáticos de odometria. As entradas do filtro são as leituras dos *encoders* e de sensores externoceptivos. Para estimar os erros não-sistemáticos utiliza-se outro Filtro de Kalman que tem como entrada os valores obtidos pelo Filtro de Kalman Aumentado.

3.2 Localização Absoluta

Nos métodos de localização absoluta, a estimativa da posição atual do robô é realizada de forma independente das estimativas anteriores. Sendo assim, como a localização atual não é derivada de integrações de diversas medidas realizadas ao longo do tempo, mas diretamente das medições atuais, não temos o acúmulo de erros como nos métodos de localização relativa. Podemos citar três modos principais de estimar a posição do robô por localização absoluta: utilizando *beacons*, *landmarks* ou baseado em mapas.

3.2.1 *Beacons*

Beacons são marcas que ativamente enviam informações sobre a localização do robô, como sinais de satélite e ondas de rádios. Sinais luminosos também podem ser utilizados, sendo assim, até mesmos as estrelas podem ser consideradas como *beacons*. O uso deste método tem várias aplicações que não são da área da robótica, como a navegação marinha e a navegação aérea (RODRIGUES, 2013; SOUZA, 2008).

De forma geral, a posição do robô pode ser estimada por *beacons* a partir do cálculo da distância percorrida pelo sinal emitido e por trilateração. O GPS (*Global Positioning System*) é um dos exemplos mais populares, o aparelho GPS recebe sinais de satélites e, por meio de trilateração, suas coordenadas são estabelecidas. Uma desvantagem de aparelhos como o GPS é sua pouca precisão em alguns casos, não sendo muito útil para robôs móveis que irão agir em ambientes pequenos.

3.2.2 landkmarks

Landmarks são elementos presentes no ambiente que podem ser facilmente perceptíveis pelo robô, através de seus sensores. Esses elementos podem ser nativos ao ambiente, chamados de *landmarks* naturais, ou inseridos propositalmente, chamados de *landmarks* artificiais.

Sistemas capazes de identificar *landmarks* naturais são ideais, sendo capazes de lidar de forma mais eficiente com o mundo real. Porém, essas *landmarks* normalmente são mais difíceis de serem identificadas do que as artificiais. *landmarks* naturais podem basear-se na estrutura do ambiente, como bordas e cantos de portas e janelas, quinas de paredes, ou até na variação de luminosidade de certa região de uma imagem, detectando fontes de luz, por exemplo (BIGHETI, 2011; SOUZA, 2008).

As *landmarks* artificiais, por outro lado, embora tenham a desvantagem de precisar adequar o ambiente, inserindo-as nele, possui como vantagem o fato de serem mais facilmente identificáveis do que as *landmarks* naturais, pois já são criadas com esse intuito. Além disso,

landmarks artificiais podem ser usadas como fonte de informação utilizadas para diferentes propósitos, não apenas para localização (BIGHETI, 2011). Exemplos de *landmarks* artificiais são código de barras (em especial, códigos QR) e figuras geométricas coloridas de fácil distinção. (SOUZA, 2008).

3.2.3 Mapas

Quando há um mapa do ambiente disponível em que o robô está inserido, a localização do robô pode ser estimada a partir comparação (*matching*) das medições sensoriais adquiridas pelo robô com as informações do mapa. Normalmente as medições adquiridas pelo robô são utilizadas para a criação de um mapa local, e então é verificado se esse mapa local corresponde a alguma região do mapa global (BORENSTEIN et al., 1996; SOUZA, 2008).

Métodos baseados em mapas tem algumas desvantagens, como exemplo, a necessidade do uso de sensores relativamente precisos, de uma grande quantidade de informações sensoriais para realizar a comparação e, normalmente, demandam um alto custo computacional. Além disso, o ambiente em que o robô está inserido deve ter características que permitam distinguir bem uma posição de outra. Se o robô estiver navegando por um longo corredor, por exemplo, em diversos pontos nesse corredor ele pode fazer leituras muito semelhantes, dificultando sua localização (BORENSTEIN et al., 1996; RODRIGUES, 2013; SOUZA, 2008).

Já em relação a vantagens, esses métodos utilizam a estrutura natural do ambiente, sem precisar de modificações, como a inserção de marcas artificiais. Métodos baseados em mapas também são essenciais para o SLAM, como veremos adiante, sendo assim, são importantes na geração e atualização do mapa do ambiente em si (BORENSTEIN et al., 1996).

A seguir, apresentaremos brevemente três métodos de localização baseados em mapas, com foco no Filtro de Partículas, que é utilizado no sistema apresentado neste trabalho.

3.2.3.1 Método de Markov

No método de Markov, a crença do robô sobre seu estado geralmente é representada por uma função de probabilidade multimodal, para representar cada posição possível dele dentro do mapa. Esse método é razoavelmente robusto, podendo averiguar múltiplas posições possíveis ao mesmo tempo, podendo estimar a postura do robô globalmente e com a capacidade de se recuperar de falhas de localização.

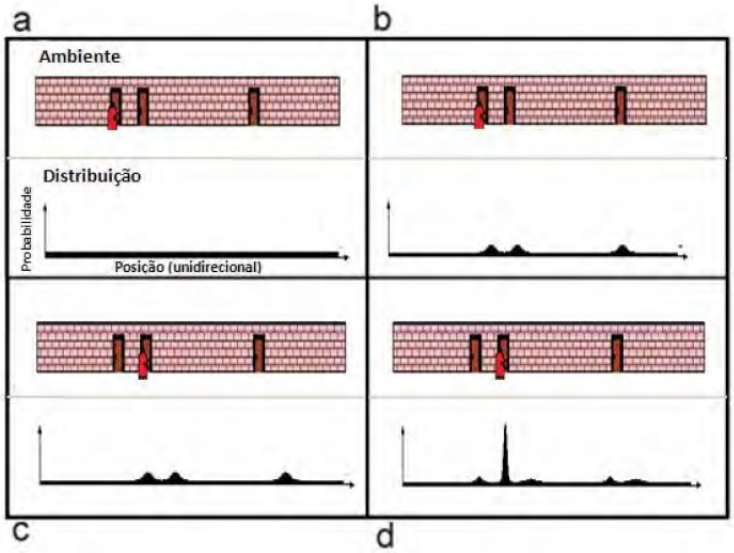
Esse método se adapta muito bem para mapas representado por grades de ocupação. Porém, para isso ele exige que o ambiente seja repartido em um número finito e discreto de possíveis posições para o robô, incluindo a sua orientação. Considerando que o robô se movimenta em um plano, por exemplo, precisamos de uma matriz tridimensional para representar todas as possíveis posições do robô. Dessa forma, esse é um processo que pode

exigir muita memória, para manter o mapa e a grade com todas as possíveis posições, e poder computacional, para atualizar a probabilidade de cada célula desta grade (RODRIGUES, 2013).

Para atualizar a probabilidade de cada possível posição, este método utiliza a fórmula de Bayes, permitindo integrar as novas informações obtidas pelos sensores do robô com a antiga crença do estado, resultante dos passos anteriores.

A Figura 2 ilustra o funcionamento do processo de localização pelo método de Markov, assumindo que o robô se move em apenas uma dimensão e que sua posição inicial não é informada. Quando não há nenhuma informação sobre a posição inicial do robô, inicialmente temos uma distribuição uniforme de probabilidade sobre todas as possíveis posições, como pode ser visto na Figura 2 a. A seguir, após a obtenção de dados sensoriais, o robô "descobre"que está do lado de uma porta, então temos um aumento na probabilidade de estar em locais próximos a portas, e a diminuição para todos os outros, como ilustrado na Figura 2 b. Em seguida, o robô se desloca, e a informação desse deslocamento é incorporado a "crença"do posicionamento do robô, deslocando a distribuição de probabilidade proporcionalmente (Figura 2 c). Agora, quando o robô verifica seus sensores, ele percebe que está novamente ao lado de uma porta, esta observação é multiplicada pela distribuição de probabilidade anterior, gerando a crença final do posicionamento, apresentada na Figura 2 d (BIGHETI, 2011). Neste ponto, nota-se que a posição com maior probabilidade é justamente a posição em que o robô se encontra.

Figura 2 – Exemplo de funcionamento do algoritmo de localização de Markov.



Fonte: Bigheti (2011, p. 18)

3.2.3.2 Filtro de Kalman

O filtro de Kalman é uma variação do filtro de Bayes, e que pode ser utilizado em diferentes contextos, em geral, em sistemas dinâmicos normalmente perturbados por ruídos, onde ele estima as variáveis do sistema utilizando uma distribuição gaussiana (BIGHETI, 2011). A maior vantagem de utilizar uma distribuição gaussiana para modelar o erro de uma certa grandeza é a facilidade de manipulação matemática, em especial pelo fato de uma função gaussiana poder ser gerada a partir de apenas dois parâmetros, a média μ e o desvio padrão σ.

Este filtro produz uma estimação ótima baseada no seu conhecimento do sistema e das informações provenientes dos seus sensores (RODRIGUES, 2013). Podemos separar as etapas para a produção desta estimativa em três estágios: predição, observação e atualização (BIGHETI, 2011).

Dentro da teoria do filtro de Kalman, assume-se que o sistema é linear, com ruído gaussiano unimodal e com média nula, porém pode-se encontrar na literatura diversas modificações que buscam adicionar ao filtro outras características, como por exemplo o filtro de Kalman estendido, que possibilita tratar de modelos não-lineares (RODRIGUES, 2013).

A localização por filtro de Kalman é capaz de rastrear a posição do robô a partir de uma posição inicial conhecida, sendo um método considerado bastante preciso e eficiente. É importante considerar que, o filtro de Kalman, em sua forma original, implementa a crença para estados contínuos, não sendo aplicável a espaço de estados discretos ou híbridos. A seguir, apresentamos os passos que normalmente são seguidos por um filtro de Kalman para localização, baseado em (RODRIGUES, 2013):

- Passo 1: Estimação Inicial da Postura. Nessa fase é feita uma estimação inicial da posição do robô, baseado na estimativa anterior e no deslocamento realizado pelo robô. No caso em que utiliza-se a odometria para detectar o deslocamento, aplica-se um modelo gaussiano de erro de movimentação proporcional ao deslocamento estimado pela odometria.

- Passo 2: Observação. O robô coleta as informações dos sensores exteroceptivos e extrai desses dados as características desejadas. Como exemplo temos a extração de retas a partir das distâncias medidas, buscando estimar a geometria do ambiente.

- Passo 3: Estimação das medições. Baseado na estimação da postura realizada no passo 1, o robô estima quais leituras ele obteria através dos sensores exteroceptivos.

- Passo 4: Casamento de Características(*Matching*). Neste passo o robô identifica a melhor combinação entre as características extraídas durante a etapa de observação (passo 2) e as características estimadas no passo 3. Como resultado temos o erro entre o que o robô observou e o que estaria observando caso estivesse na posição estimada no passo 1.

- Passo 5: Estimação Final da Postura. Por fim, o filtro funde as informações provenientes da odometria e da extração de características do ambiente, considerando o erro estimado no passo 4, para atualizar o estado da crença da posição do robô.

3.2.3.3 Filtro de Partículas

O filtro de partículas é um filtro não-paramétrico Bayesiano que pode ser utilizado para estimar posições posteriores do robô e representar as crenças, ou seja, representar a probabilidade do robô estar em uma determinada posição. Este filtro pode atuar tanto para a localização global quanto local, e é conhecido também como *Monte Carlo Localization* (MCL). A ideia principal do filtro de partículas é representar a crença por um conjunto de amostras chamadas de partículas (THRUN et al., 2001), em que cada partícula possui um peso associado. No nosso caso, podemos definir cada partícula como uma possível posição do robô, através da tupla (x, y, θ), e o peso de cada partícula será relativo a probabilidade do robô se encontrar naquela posição.

Sendo um filtro não-paramétrico, onde a representação da crença posterior $bel(x_t)$ é aproximada através das partículas e seus pesos, o filtro de partículas pode representar um espaço muito mais amplo de distribuição que, por exemplo, funções gaussianas. Além disso, esse filtro é mais favorável pra representar densidades de probabilidade multi-modais (RODRIGUES, 2013).

Esse método pode processar medidas brutas dos sensores, nem a necessidade de extração de características. Ele também pode ser aplicado em localização global e, em alguns casos, resolver problemas de sequestro do robô, em que o robô é "sequestrado", sendo levado para uma posição desconhecida, enquanto acredita estar na posição anterior. Além disso, este método é relativamente fácil de implementar (THRUN et al., 2001).

A eficiência da aproximação do filtro de partículas está diretamente relacionada com o tamanho do conjunto de partículas. Aumentando-se o número total de partículas, aumenta-se a eficiência da aproximação, porém aumenta-se também a demanda de poder computacional. Desta forma, a implementação pode se adaptar aos recursos computacionais disponíveis, mas no geral, esse método requer uma maior complexidade computacional se comparada com outros, como o filtro de Kalman (RODRIGUES, 2013).

No sistema apresentado por este trabalho é utilizado um filtro de partículas para estimar a localização do robô a partir de uma estimativa inicial obtida pela odometria e da comparação das informações sensoriais com o mapa parcial do ambiente. Sua implementação segue os passos descritos a seguir, que por sua vez são baseados no algoritmo descrito em (RODRIGUES, 2013):

- Passo 1: Reamostragem. As partículas de maior peso obtidas na aplicação anterior do

filtro, no momento t_{k-1}, são usadas como partículas de referência e servem de ponto de partida para a reamostragem das demais partículas, através da aplicação de ruídos gaussianos para cada parâmetro da partículas.

- Passo 2: Estimação do novo estado de cada partícula. A partir do estado anterior das partículas e do deslocamento do robô estimado através da odometria entre os instantes t_{k-1} e t_k, uma nova estimação de estado é feita.

- Passo 3: Estimação de observações. Com base nas novas posições de cada partícula e das leituras dos sensores exteroceptivos, é estimado o mapa local que seria obtido por cada uma dessas partículas.

- Passo 4: Cálculo dos pesos. Nesse passo realiza-se um casamento de características entre o mapa global com o mapa local de cada partícula, buscando o mapa local que melhor se encaixa no mapa global. A partir dessa operação de *matching* atribui-se pesos as partículas.

3.3 Fusão Multi-sensores

Os métodos de localização que utilizam fusão de multi-sensores buscam realizar uma melhor estimativa da localização do robô a partir de várias informações sensoriais lidas de múltiplos sensores.

Em particular, quando isso é feito utilizando sensores heterogêneos, pode-se obter melhores resultados, visto que algo que não é percebido por um determinado tipo de sensor, talvez possa ser facilmente detectado por outro. Além disso, a fusão de múltiplos sensores pode reduzir os efeitos dos erros na medição.

Nesse método é comum o uso de técnicas probabilísticas, onde as incertezas e a confiabilidade das informações são tratadas de maneira mais adequada. Algumas das técnicas mais comuns são o Filtro de Kalman Extendido e o Filtro de Partículas, apresentados anteriormente sendo aplicados como métodos de localização. Essa fusão também pode ser feito através do uso de Redes Neurais Artificias, tendo como vantagem a flexibilidade e a facilidade de utilização, não requerendo a elaboração de modelos matemáticos complexos (MARCHI, 2001).

4 Mapeamento

Podemos definir um mapa como uma representação de um ambiente (MARCHI, 2001). Dessa forma, o processo de mapeamento é o nome dado à tarefa de construir representações de um ambiente, a partir dos dados obtidos dele (BIGHETI, 2011).

O mapeamento é uma tarefa comum em várias atividades humanas, como a arquitetura, cartografia, aviação, atividades militares e outras. E quando tratamos de mapeamento por robôs, a utilização de mapas é necessária para que eles consigam construir um modelo do mundo ao seu redor e utilizarem esse conhecimento como auxílio para cumprirem os seus objetivos. Para poder realizar o mapeamento, o robô deve ser capaz de perceber o ambiente ao seu redor, e isso é feito através de sensores que interagem com o ambiente de alguma forma, como, por exemplo, explorando propriedades da luz (câmeras, laser, infravermelho) ou de ondas mecânicas (sonares). (BIGHETI, 2011).

Porém, o mapeamento robótico não é uma tarefa tão simples, estando sujeita a vários problemas. Em primeiro lugar, temos que os sensores estão sujeitos a imprecisões e ruídos em suas leituras, de forma que suas leituras nem sempre correspondem a realidade. Além disso, conhecer a localização do robô é importante para o mapeamento, e esse processo também está sujeito a erros e imprecisões, como foi tratado no capítulo anterior. Sendo assim, o primeiro desafio no processo de mapeamento é modelar tais erros e buscar minimiza-los (SOUZA, 2008).

A dimensão do ambiente também pode ser um problema, quanto maior o ambiente mais custoso e menos preciso será o mapa. Também surge o desafio conhecido como o problema da correspondência de dados, ou associação de dados. Esse problema consiste em determinar se medições tomadas em diferentes instantes de tempo correspondem ao mesmo objeto ou não. Outro desafio está no mapeamento de ambientes dinâmicos, onde temos o trafego constante de objetos ou pessoas, e este talvez seja o menos trivial dos desafios citados, considerando que a maioria dos algoritmos de mapeamento consideram o processo sendo feito em ambientes estáticos (SOUZA, 2008).

Existem duas abordagens principais para a representação de mapas: a topológica e a métrica. As próximas seções detalham melhor ambas as abordagens, com foco maior em uma abordagem métrica, que são as grades de ocupação.

4.1 Mapeamento Topológico

Os mapas topológicos, ou mapa relacionais, fornecem as relações entre os vários locais ou objetos presentes no ambiente, geralmente sob a forma de um grafo, onde os nós correspondem a lugares significativos e as arestas contém informações de navegação entre os nós.

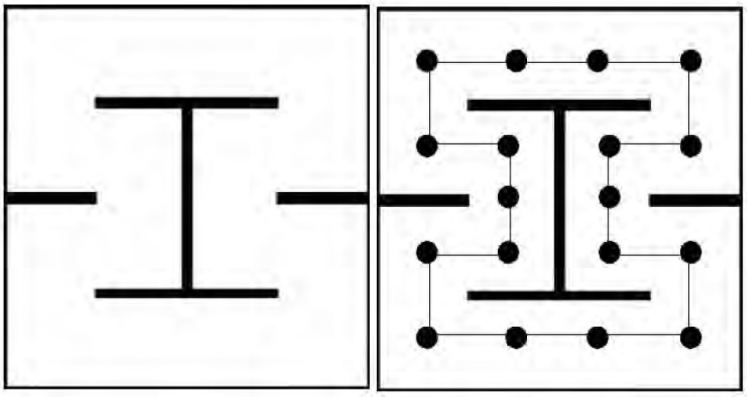
Uma característica importante dos mapas topológicos é a simplicidade da modelagem hierárquica do ambiente, o que facilita a manipulação do modelo e permite trabalhar com diferentes resoluções. Com isso os mapas topológicos permitem um planejamento eficiente, são compactos para armazenamento e não é necessário conhecer a localização exata do robô, mas apenas conhecer o nó em que ele se encontra.

Por outro lado, este tipo de representação apresenta grandes dificuldades quanto ao

problema da correspondência, nem sempre sendo capazes de reconhecer um local já visitado. E isso acarreta na dificuldade de construir e manter mapas para ambientes grandes, além de permitir que sejam produzidos caminhos sub-ótimos para a navegação do robô entre dois pontos (SOUZA, 2008).

Na Figura 3 pode-se observar um exemplo de mapa topológico, onde os nós presentes na Figura 3 (b) representam locais ou marcos distintos. E, caso haja um caminho direto que ligue dois pontos distintos desse ambiente, os nós correspondentes são conectados através de um segmento de reta, representando esse caminho como uma aresta.

Figura 3 – (a) Ambiente não mapeado. (b) Representação de um mapa topológico.



Fonte: Oliveira (2010, p. 43)

Como exemplo de método prático para a obtenção de um mapa topológico de um ambiente, pode-se citar o trabalho de (FABRIZI; SAFFIOTTI, 2000 apud SOUZA, 2008). Eles implementaram um sistema de mapeamento topológico a partir de informações sobre espaços livres. Para isso, o ambiente foi primeiramente discretizado em uma grade de ocupação, que constitui um mapa métrico, como apresentado na próxima seção. Com uma grade de ocupação para representar o ambiente, sua topologia foi extraída combinando o método de topologia digital, que é definido como a extração da topologia de um espaço discreto, com uma ferramenta de processamento de imagens baseada em morfologia matemática.

4.2 Mapeamento Métrico

O mapeamento métrico produz uma definição geométrica do ambiente em que o robô está inserido. Dessa forma, temos uma representação mais detalhada do ambiente, incluindo os objetos que o compõe como paredes, obstáculos e passagens, mantendo uma boa relação com o mundo real.

Mapas métricos normalmente são mais fáceis de se construir, representar e manter. Eles também são mais adequadas para tratar o problema da correspondência, reconhecendo

lugares com mais facilidade, já que é baseado na geometria do ambiente.

Por outro lado, a construção de um mapa métrico requer a determinação precisa da localização do robô, e o planejamento de rotas a partir de mapas métricos possui um alto custo computacional (SOUZA, 2008).

Os mapas métricos podem ser representados por mapas de características, que armazenam informações geométricas de formas encontradas nos ambientes; ou por grades de ocupação, que consiste em discretizar os espaços contínuos do ambiente em uma matriz que armazena a probabilidade de ocupação de cada célula.

No sistema desenvolvido neste trabalho se utiliza uma grade de ocupação para representar o mapa do ambiente. Por esse motivo, a seguir é apresentado o método das grades de ocupação com maior detalhamento.

4.2.1 Grades de Ocupação

As grades de ocupação, ou *occupancy grids*, é um método proposto por (ELFES, 1987), e, posteriormente, formalizado em sua tese (ELFES, 1989a). A seguir, o método é descrito tendo base em (ELFES, 1989b), assim como em trabalhos posteriores que se basearam nessa solução, como (SOUZA, 2008) e (MILSTEIN, 2008).

A ideia das grades de ocupação é representar o ambiente em uma grade, ou matriz multi-dimensional de 2 ou 3 dimensões, onde as células dessa grade armazenam a probabilidade de ocupação do espaço representado por aquela células. Isto porque um robô só é capaz de obter informações sobre seu ambiente indiretamente, através de seus sensores, estando sujeito a erros e imprecisões, de forma que a melhor forma de construir um modelo espacial do ambiente é utilizando métodos probabilísticos.

Sendo assim, os estados ou valores das células são estimados pela interpretação dos dados obtidos dos sensores de distância modelados por uma função de densidade de probabilidade. E para a atualização do estado das células, com o decorrer de novas leituras durante o processo de mapeamento, utiliza-se regras probabilísticas Bayesianas. Como resultado final, a grade de ocupação constitui um modelo espacial probabilístico que pode ser utilizado como um mapa do ambiente do robô, auxiliando em tarefas de navegação.

Esse método considera que as células do mapa são independentes entre si, o que permite que o método seja eficiente. Além disso, temos que o mapa depende apenas do histórico de localizações do robô, $x^t = \{x_0, ..., x_t\}$ e das leituras dos sensores em cada localização, $z^t = \{z_0, ..., z_t\}$.

Como dito anteriormente, para atualizar o estado das células com a composição incremental de dados sensoriais, utiliza-se o teorema de Bayes, de forma que a probabilidade de uma célula $m_{x,y}$ estar ocupada em um instante t pode ser estimada recursivamente pela

aplicação da Equação 4.1.

$$p(m_{x,y}|x^t, z^t) = \frac{p(z_t|x^t, z^{t-1}, m_{x,y})p(m_{x,y}|x^{t-1}, z^{t-1})}{p(z_t|x^t, z^{t-1})} \tag{4.1}$$

Nesta equação, $p(z_t|x^t, z^{t-1}, m_{x,y})$ representa o modelo probabilístico do sensor de alcance, $p(m_{x,y}|x^{t-1}, z^{t-1})$ o estado da célula $m_{x,y}$ no instante $t-1$ e $p(z_t|x^t, z^{t-1})$ é o valor real medido pelo sensor.

Supondo que o mapeamento é realizado em um ambiente estático, temos que a medição dos sensores em um instante t independe das medições passadas. Podemos considerar também que as medições realizadas pelo robô em um instante t dependem da localização x_t, mas são independentes das localizações anteriores x^{t-1}. Isso implica nas equações 4.2 e 4.3.

$$p(z_t|x^t, z^{t-1}, m_{x,y}) = p(z_t|x_t, m_{x,y}) \tag{4.2}$$

$$p(z_t|x^t, z^{t-1}) = p(z_t|x_t) \tag{4.3}$$

Dessa forma, podemos simplificar a Equação 4.1, resultando na Equação 4.4.

$$p(m_{x,y}|x^t, z^t) = \frac{p(z_t|x_t, m_{x,y})p(m_{x,y}|x^{t-1}, z^{t-1})}{p(z_t|x_t)} \tag{4.4}$$

Com isso, podemos aplicar a regra da Probabilidade Total à Equação 4.4, obtendo a Equação 4.5, que estima a probabilidade da célula $m_{x,y}$ estar ocupada, tendo como base o modelo probabilístico do sensor, o valor de ocupação da célula no instante anterior, a posição do robô e a medição atual do sensor.

$$p(m_{x,y}|x^t, z^t) = \frac{p(z_t|x_t, m_{x,y})p(m_{x,y}|x^{t-1}, z^{t-1})}{\sum_{x,y} p(z_t|x_t, m_{x,y})p(m_{x,y}|x^{t-1}, z^{t-1})} \tag{4.5}$$

Devido a instabilidade numérica com probabilidades próximas de 0 ou 1, é comum calcular o *log-odds* (logarítmo da probabilidade) de $p(m_{x,y}|x^t, z^t)$ ao invés de $p(m_{x,y}|x^t, z^t)$. O *log-odds* de $p(m_{x,y}|x^t, z^t)$ é definido por:

$$l^t_{x,y} = log\frac{p(m_{x,y}|x^t, z^t)}{1 - p(m_{x,y}|x^t, z^t)} \tag{4.6}$$

E o valor da probabilidade pode ser recuperado pela seguinte equação:

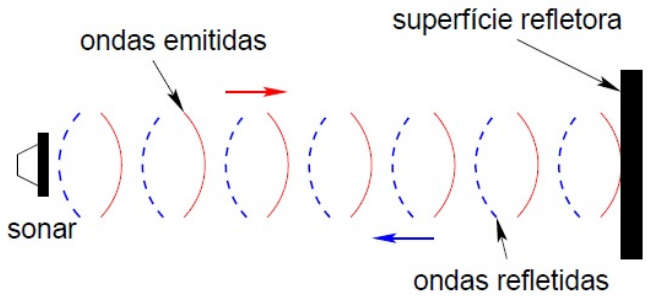
$$p(m_{x,y}|x^t, z^t) = 1 - \frac{1}{e^{l^t_{x,y}}} \tag{4.7}$$

Neste trabalho é utilizado duas classes de sensores de distância, utilizados para obter informações do ambiente em que o robô se situa, sendo eles o sonar e o sensor de distância infravermelho. Por este motivo, a seguir será descrito o funcionamento destes sensores, assim como será apresentado um modelo probabilístico que será usado na interpretação dos dados sensoriais durante o processo de mapeamento.

4.2.1.1 Sensor de Ultrassom e Infravermelho

O sensor de ultrassom, ou sonar, é capaz de medir distâncias utilizando o tempo de propagação de ondas sonoras. O seu funcionamento se baseia na propagação de ondas ultrassônicas, geradas por um emissor, refletidas por algum objeto e capturadas por um receptor, como ilustrado pela Figura 4. As primeiras aplicações de sonares foram na área marítima, sendo utilizados como um instrumento de navegação auxiliar, e mesmo hoje ainda são muito usados no estudo e pesquisa dos oceanos, por exemplo, na determinação da profundidade do fundo do mar (BIGHETI, 2011; SOUZA, 2008).

Figura 4 – Funcionamento do sonar.



Fonte: Souza (2008, p. 32)

A variável de medida de um sonar é o tempo de trajeto do ultrassom entre a face do emissor, a superfície do objeto e a face do receptor. Conhecendo o tempo do trajeto do ultrassom, considerando que a velocidade de propagação do ultrassom no ar é praticamente constante, é possível inferir a distância entre o sensor e o objeto detectado.

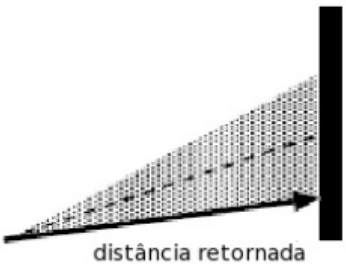
Este sensor é bastante utilizado na área da robótica, na detecção de obstáculos e no processo de mapeamento, isso devido ao seu baixo custo e a simplicidade da resposta fornecida.

Porém, como qualquer sensor, o sonar possui algumas desvantagens que comprometem seu desempenho, sofrendo imprecisões nas leituras de distância. Algumas dos fatores que podem ocasionar em problemas são:

- Imprecisões no circuito temporizador: imprecisões relacionadas ao circuito temporizador do sonar limitam a precisão das medições. Normalmente os fabricantes disponibilizam *datasheets* indicando a precisão dos sensores.

- Distância reduzida: O ultrassom emitido se constitui em um feixe, com uma região de maior sensibilidade chamada de feixe principal, considerada como a região de operação do sensor. Supondo que a superfície do objeto refletor não esteja perpendicular ao sonar, um lado do feixe alcançara primeiro a superfície do objeto retornando a distância mais próxima, como exemplificado pela Figura 5. Se o sistema que utiliza os dados sensoriais considera que as leituras estão ao longo do eixo principal do sonar, então o dado retornado estará errado.

Figura 5 – Exemplo de distância reduzida.



Fonte: Souza (2008, p. 34)

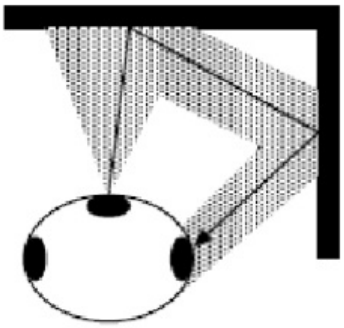
- Reflexões especulares: os feixes sonoros podem sofrer reflexões especulares (ou múltiplas reflexões). Isso acontece quando o ângulo entre o feixe incidente e a superfície do objeto é pequena, de forma que o feixe é totalmente refletido para fora do alcance do sensor, produzindo leituras falsas. Na Figura 6 pode-se ver um exemplo de reflexão especular de uma onda sonora, devido ao ângulo de incidência.

Figura 6 – Exemplo de reflexão especular.

Fonte: Souza (2008, p. 34)

- Leituras cruzadas: As leituras cruzadas são uma possível consequência das reflexões especulares. Uma leitura cruzada pode acontecer quando se utiliza mais de um sonar no mesmo robô, de forma que um feixe disparado por um sonar pode acabar sendo detectado por um outro sonar, gerando medidas erradas. Na Figura 7 encontra-se um exemplo de leitura cruzada em um robô munido de três sonares. Tal problema pode ser corrigido pelo sincronismo dos disparos dos feixes.

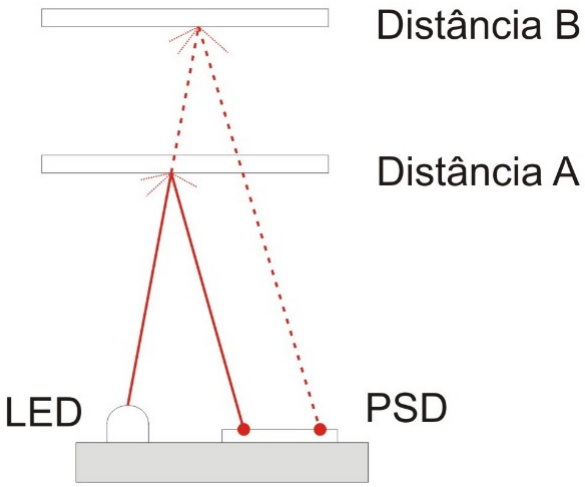
Figura 7 – Exemplo de leitura cruzada.



Fonte: Souza (2008, p. 34)

Os sensores de alcance baseados em infravermelho, por sua vez, são capazes de medir a distância do sensor até um objeto pela emissão de feixe luminoso infravermelho por um dispositivo emissor, que refletirá na superfície do objeto e será captado por outro dispositivo receptor.

Figura 8 – Funcionamento de um sensor de distância infravermelho.



Fonte: Patsko (2006)

Na maioria dos dispositivos desta classe de sensores, o emissor consiste em um diodo laser ou led infravermelho, o qual emite um raio de um determinado espectro de frequências. Como dispositivo receptor é utilizado um PSD (*Position Sensing Device* - Dispositivo de Monitoramento de Posição), que através de fotodiodos é capaz de detectar o raio luminoso emitido. De acordo com a distância do objeto em que a luz refletiu, o raio incide em um ponto diferente no dispositivo, e a posição em que este raio incide é identificada pelo PSD. Dessa forma, através de triangulação, o sensor é capaz de estimar a distância do objeto detectado. (PATSKO, 2006; SOUZA; BORGES, 2009). O funcionamento de um sensor de distância infravermelho é exemplificado na Figura 8.

Sensores infravermelhos apresentam baixo custo, tamanho reduzido, robustez diante de ruídos eletromagnéticos e a capacidade de detectar quase todo tipo de material, desde que não sejam transparentes ou translúcidos. Porém, devido ao seu funcionamento semelhante ao sonar (cálculo da distância a partir da emissão, reflexão e recepção de um sinal), ele também está sujeito a erros semelhantes ao sensor anterior. Além disso, o sensor pode sofrer ruídos devido a iluminação do ambiente.

Devido aos problemas que podem ocorrer com as leituras desses sensores, levando a imprecisão dos dados, construir um modelo probabilístico que compreenda todas essas incertezas é uma tarefa árdua. Dessa forma, os modelos utilizados costumam ser um pouco mais simplistas.

Um modelo bastante simplista é o modelo do sensor ideal, o qual se caracteriza por não ter incertezas em suas medições, sendo caracterizado pela função dada pela Equação 4.8. Nesta função, z representa a medição retornada pelo sensor e $d_{x,y}$ a distância euclidiana entre o sensor e a célula $m_{x,y}$, que está sendo mapeada (SOUZA, 2008).

$$p(z|d_{x,y}) = \begin{cases} 1, & \text{se } z = d_{x,y} \\ 0, & \text{caso contrário} \end{cases} \tag{4.8}$$

Baseado nesse modelo, a função que estima a ocupação de uma célula, calculada pela Equação 4.5, pode ser estimada pela Figura 9. Pode-se perceber que, já que no modelo ideal não são consideradas incertezas, todas as células com a distância menor do que a lida pelo sensor são consideradas livres, com probabilidade de ocupação 0, enquanto a célula com a exata distância lida pelo sensor é considera ocupada com 100% de certeza. Por fim, as células com distância maior que a lida têm probabilidade $0,5$, isso porque elas estão, supostamente, atrás do nosso objeto, e nada pode-se inferir sobre elas apenas com essa leitura.

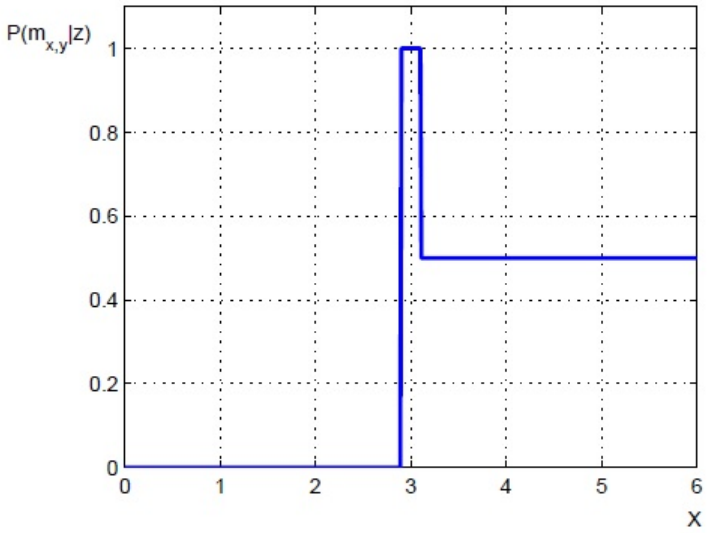
Agora, considerando uma incerteza em relação a distância medida, através da aplicação de ruído gaussiano de média 0 e variância σ_z^2, podemos obter o modelo dado pela Equação 4.9. E esse modelo gera a função que estima a ocupação de uma célula ilustrada pela Figura 10.

$$p(z|d_{x,y}) = \frac{1}{\sqrt{2\pi\sigma_z}} e^{-\frac{(z-d_{x,y})^2}{2\sigma_z^2}} \tag{4.9}$$

Por fim, o modelo proposto por Elfes (1989a) trata também as incertezas relacionadas ao ângulo do sensor ultrassônico, através de uma distribuição Gaussiana bidimensional, representada pela Equação 4.10. O gráfico presente na Figura 11 representa este modelo.

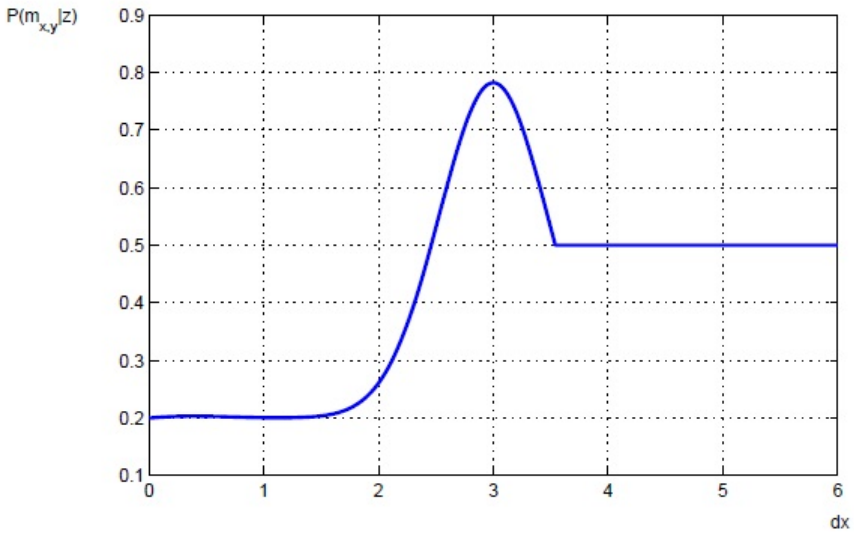
$$p(z,\theta|d_{x,y},\theta_{x,y}) = \frac{1}{2\pi\sigma_z\sigma_\theta} e^{-\frac{1}{2}\left(\frac{(z-d_{x,y})^2}{\sigma_z^2}+\frac{(\theta-\theta_{x,y})^2}{\sigma_\theta^2}\right)} \tag{4.10}$$

Figura 9 – Função de ocupação para um sensor ideal com leitura $z = 3$.



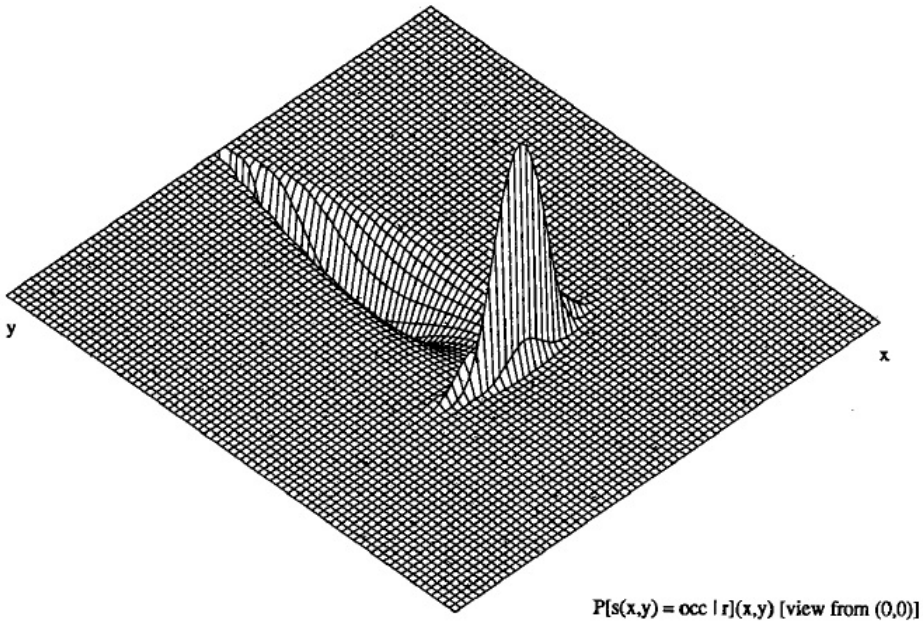
Fonte: (SOUZA, 2008, p. 36)

Figura 10 – Função de ocupação para um sensor modelado por uma distribuição Gaussiana com leitura $z = 3$.



Fonte: (SOUZA, 2008, p. 37)

Figura 11 – Função de ocupação para um sensor modelado por uma distribuição Gaussiana bidimensional.



Fonte: (ELFES, 1989b, p. 49)

O modelo proposto por Elfes é adotado neste trabalho, tanto para o sonar como para o sensor infravermelho. Sendo assim, para cada leitura realizada, a grade de ocupação é atualizada conforme a distribuição gerada pela equação que representa este modelo, utilizando como parâmetros os dados sensoriais obtidos.

5 SLAM

Se um ser humano é colocado em um ambiente totalmente desconhecido, ele tem a capacidade de se familiarizar com o lugar a partir da exploração e observação. A partir dessa exploração e observação o ser humano passa a conhecer o ambiente, como se o estivesse mapeando mentalmente, e ele ainda é capaz de se localizar nesse "mapa" a partir de pontos de referência que foram estabelecidos durante sua observação (BIGHETI, 2011). Muitos estudos da área da Robótica Móvel tratam de tentar fornecer aos robôs essa capacidade, inerente ao ser humano, de mapear e se localizar simultaneamente dentro de um ambiente desconhecido, o que deu origem ao problema conhecido como *Simultaneous Localization and Mapping* (SLAM).

Suponha que um robô móvel esteja posicionado em um ambiente desconhecido e que sua localização também seja desconhecida, e esse robô deve ser capaz de mapear e navegar por esse ambiente. Para realizar o mapeamento, é importante conhecer a localização do robô, sem a qual se impossibilita a construção do mapa em si, não sendo possível determinar a posição dos obstáculos detectados. Uma forma de se localizar no ambiente é ter um mapa consistente dele, o qual podemos consultar e comparar com os dados locais obtidos dos sensores, como visto no Capítulo 3. Mas, para obter um mapa consistente, precisamos de uma estimativa apurada da localização do robô, logo um paradoxo se apresenta. Uma solução é realizar a localização e o mapeamento simultaneamente, o que constitui o problema SLAM (BECKER, 2015).

O SLAM é considerado um problema difícil, pois ele envolve a execução de dois processos paralelos e relacionados. O que também causa uma certa dificuldade é a existência de locais com características semelhantes, gerando ambiguidades que dificultam o processo de localização do robô (MATARIĆ, 2007). Por tal motivo, soluções para o SLAM normalmente envolvem modelos e métodos probabilísticos.

Os estudos sobre esse problema têm origem em 1986, na Conferência de Robótica e Automação do IEEE em São Franscisco, com a discussão sobre métodos de localização e mapeamento pelos pesquisadores Hugh Durrant-Whyte, Peter Cheeseman, Jim Crowley, Raja Chatila, Oliver Faugeras e Randal Smith. Desde então, diversos métodos foram propostos para diferentes tipos de ambientes, sendo que os melhores resultados se deram para ambientes estáticos, estruturados e de menor escala. Já para ambientes dinâmicos, desestruturados e de larga escala ainda existem muitos problemas a serem solucionados (BIGHETI, 2011).

5.1 Problemas e desafios

Existem diversos desafios e problemas que precisam ser enfrentados para resolver o problema do SLAM, decorrente de diferentes fatores. Em sua dissertação, Bigheti (2011)

sintetiza alguns dos principais problemas apontados na literatura ao longo das décadas, sendo eles:

- Problema da mudança do ambiente: esse problema é encontrado em ambientes dinâmicos, onde pode haver objetos em movimento e alterações no ambiente. O dinamismo do ambiente é difícil de ser tratado, e um dos motivos é a dificuldade em definir o que é realmente dinâmico e o que é ruído gerado pelos sensores.

- Problema da associação de dados: o problema de relacionar as observações realizadas pelo robô em diferentes instantes, ou com as informações armazenadas em um mapa. Esse problema se torna particularmente difícil quando não se conhece a localização precisa do robô e quando o ambiente possui muitos lugares parecidos, considerando a forma como o robô o percebe.

- Problema da estratégia de exploração: a exploração de ambientes previamente conhecidos e modelados é relativamente fácil, mas no caso do SLAM o ambiente é, a principio, desconhecido, sendo necessário criar estratégias de exploração. Essas estratégias podem ser criadas considerando diferentes intuitos, como visitar o maior número possível de regiões desconhecidas, gastar a menor quantidade de tempo e energia possíveis ou evitar que o robô se perca.

- Problema da dimensionalidade: a dimensão dos objetos e do ambiente está diretamente relacionada com o nível de detalhamento do mapa do ambiente e com o formato das informações armazenadas. Se tratando do ambiente interno de uma residência, por exemplo, para a construção de um mapa topológico algumas dezenas de vértices podem ser suficientes para criar o modelo do ambiente, porém para a construção de um mapa métrico, utilizando grades de ocupação, será necessário uma matriz de tamanho proporcional a área da casa.

- Erros gerados pelos sensores: tratar os ruídos gerados pelos sensores é um dos problemas mais complexos do mapeamento. O acúmulo desses erros pode afetar a interpretação futura dos dados e gerar um modelo do ambiente que não corresponda com fidelidade o ambiente real.

5.2 Soluções

Nesta seção são apresentadas algumas das soluções para o problema do SLAM.

5.2.1 EKF-SLAM

O Filtro de Kalman foi citado anteriormente neste trabalho, sendo utilizado como um método de localização baseado em mapas. O Filtro de Kalman também pode ser aplicado no SLAM, tanto no processo de localização quanto de mapeamento, normalmente quando esse mapeamento é baseado em *landmarks*. Em especial, o EKF-SLAM, que utiliza como base o Filtro de Kalman Estendido, ou, em inglês, *Extended Kalman Filter* (EKF), de onde vem o nome do método. Está é uma das soluções mais conhecidas para o SLAM.

O EKF-SLAM é um processo recursivo, onde o estado do robô no tempo t é calculado através da crença do estado no tempo $t-1$, do comando de deslocamento u entre os tempos $t-1$ e t e a medida z feita pelos sensores exteroceptivos no tempo t (BONTEMPO, 2012). Entende-se por sensores exteroceptivos os sensores que adquirem informações externas ao robô, informações referentes ao ambiente, como a distância de objetos. E sensores proprioceptivos aqueles que medem valores internos ao robô, como contagem de giros do motor e carga da bateria.

A crença do estado do robô no tempo $t-1$ é dado pelas variáveis aleatórias de distribuição normal $\mu_{x_{t-1}}$, $\mu_{y_{t-1}}$ e $\mu_{\theta_{t-1}}$, representando as coordenadas x, y e a orientação θ estimadas do robô no instante $t-1$, com covariância igual a Σ_{t-1}.

Inicialmente, o Filtro de Kalman Estendido estima o estado do robô, na etapa de predição (BIGHETI, 2011), utilizando apenas as informações dos sensores proprioceptivos, como os *enconders* para a odometria. Essas informações fornecem o parâmetro u_t. Essa etapa pode ser representadas pelas Equações 5.1 e 5.2, onde g é a função de transição de estado do robô (no caso deste trabalho, é utilizado a odometria), G_t o jacobiano da função de transferência de estado, V_t a covariância da função de transição de estado e R_t é a covariância da transferência de estado.

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \tag{5.1}$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t R_t V_t^T \tag{5.2}$$

Por fim, a partir das informações adquiridas na etapa de observação pelos sensores exteroceptivos, é realizada a etapa de atualização, onde o estado estimado e a sua covariância são alterados considerando as novas informações. As Equações 5.3, 5.4 e 5.5 são utilizadas nessa etapa, onde H_t é o jacobiano da função de observação, Q_t a covariância da função de observação, z_t as coordenadas da *landmark* observada, $\hat{z}_t$ as coordenadas estimadas da *landmark* e K o ganho de Kalman (BIGHETI, 2011).

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \tag{5.3}$$

$$\mu_t = \bar{\mu}_t + K(z_t - \hat{z}_t) \tag{5.4}$$

$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \tag{5.5}$$

Neste trabalho não utiliza-se o EKF-SLAM. O primeiro motivo para isso é a utilização das grades de ocupação. O EKF-SLAM está preparado para trabalhar com *landmarks*, e não com mapas representados em grades de ocupação, ainda que isso seja possível, mas seria necessário extrair as *landmarks* da grade de ocupação.

Outro motivo para não utilizar esse método foi a escolha dos sensores. O sonar e o sensor infravermelho, devido aos seus funcionamentos, não são capazes de extrair das *landmarks* o ângulo do robô e, sendo assim, não são obtidos bons resultados. Isso é afirmado por Bontempo (2012), que utiliza uma bússola para contornar tal problema.

5.2.2 FastSLAM

O FastSLAM é um método de SLAM proposto por Montemerlo et al. (2002) que utiliza tanto o filtro de Kalman quanto o filtro de partículas. De forma geral, o filtro de Kalman é utilizado para estimar a posição das *landmarks* e o filtro de partículas, baseado no método MCL, para estimar a localização do robô (BIGHETI, 2011).

Uma das principais vantagens do FastSLAM sobre os métodos baseados puramente no filtro de Kalman, como o EKF-Slam, está na sua eficiência. O FastSLAM possui uma complexidade menor do que o EKF-Slam.

O EKF-Slam tem complexidade computacional de $O(K^2)$ em que K representa o número de *landmarks*, o que é um fator bastante limitante, visto que um ambiente natural pode conter milhões de *landmarks* (MONTEMERLO et al., 2002).

O FastSLAM, por sua vez, possui complexidade $O(MlogK)$, em que M é o número de partículas utilizadas e K o número de *landmarks*. Sendo que, utilizando um filtro de partículas Rao-Blackwellised (DOUCET et al., 2000), que é uma extensão mais eficiente do filtro de partículas tradicional, Montemerlo et al. (2002) concluem que 100 partículas é suficiente para obter bons resultados na maioria dos casos.

5.2.3 SLAM com Grades de Ocupação

Uma característica interessante do FastSLAM é que, diferente da maioria das outras soluções, é possível adaptá-lo quase que diretamente para ser utilizado com grades de ocupação, sem a necessidade de extrair informações de *landmarks*. Sendo assim, a grade de ocupação

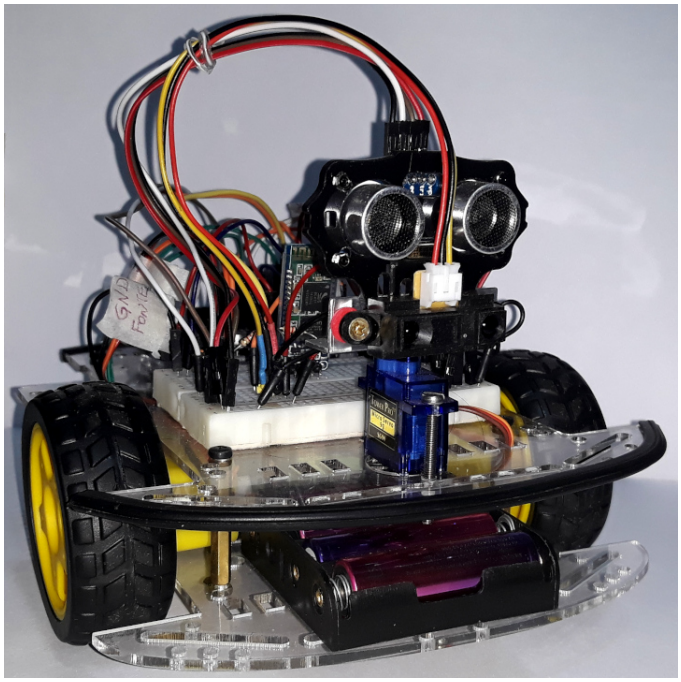
utilizada para representar o ambiente pode ser utilizada e atualizada conforme as técnicas tradicionais, como as apresentadas anteriormente neste trabalho (MILSTEIN, 2008).

Alguns dos métodos que podem ser citados que utilizam grades de ocupação são o GridSLAM (HäHNEL et al., 2003 apud MAFFEI et al., 2013) e o DP-SLAM (ELIAZAR; PARR, 2003 apud MAFFEI et al., 2013). Ambos os métodos são baseados no FastSLAM, em especial no uso de filtro de partículas para a correção da localização do robô.

Neste trabalho, a implementação realizada para resolver o problema do SLAM é baseada nos trabalhos que envolvem os métodos que utilizam grades de ocupação. O método implementado é mais simples do que algumas das soluções encontradas e estudadas, devido ao tempo disponível para a realização deste projeto e aos recursos disponíveis.

6 Descrição do Hardware

Figura 12 – Robô Frank.

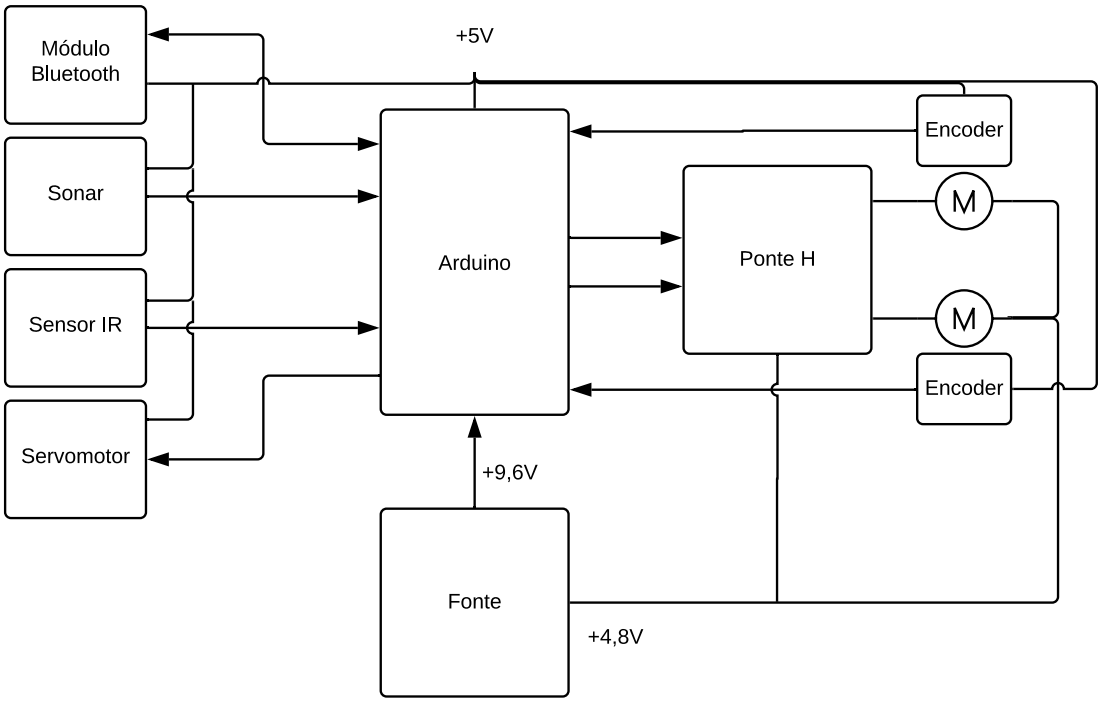


Fonte: Autor (2019)

Neste capítulo é apresentado o robô desenvolvido para este trabalho, assim como os seus componentes. Em suma, o robô Frank, como o denominamos, é constituído por uma placa Arduino e munido de um sensor ultrassônico e um sensor infravermelho acoplado à um servo motor, o que permite o robô coletar dados sobre o ambiente em que ele está inserido; rodas para a locomoção do robô, e um módulo Bluetooth para realizar a comunicação com o computador.

Na Figura 12 é apresentada uma foto do Frank, enquanto a Figura 13 consiste em um diagrama em blocos que detalha a conexão entre os componentes do robô.

Figura 13 – Diagrama de blocos do circuito do Frank.



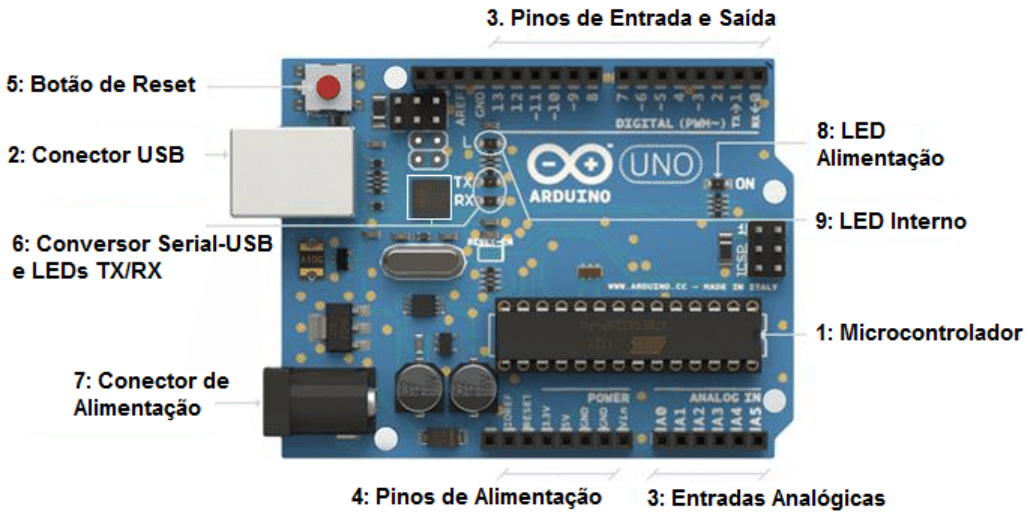
Fonte: Autor (2019)

6.1 Arduino

O Arduino é uma plataforma *open-source* de prototipagem eletrônica com hardware e software flexíveis, criado para ser de fácil uso para diferentes tipos de desenvolvedores (MOTA, 2017a). O Arduino foi criado em 2005 por cinco pesquisadores: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. Sua primeira versão era uma placa composta por um microcontrolador Atmel, circuitos de entrada/saída e que podia ser facilmente conectada à um computador via USB e programada utilizando uma linguagem baseada em C++ (THOMSEN, 2014).

Atualmente existem diversos modelos de placas Arduino, sendo a mais popular a placa Arduino Uno (MOTA, 2017a), que foi adotada neste trabalho. Um exemplo de Arduino Uno pode ser visto na Figura 14. A seguir, no Quadro 1, encontram-se as especificações desta placa.

Figura 14 – Placa Arduino Uno.



Fonte: Mota (2017a)

Quadro 1 – Especificações do Arduino Uno.

Microcontrolador	ATmega328P
Tensão de operação	5V
Tensão de alimentação (recomendada)	7-12V
Tensão de alimentação (limite)	6-20V
Entradas e saídas digitais	14 (das quais 6 podem ser PWM)
Entradas analógicas	6
Corrente contínua por pino de E/S	20mA
Corrente contínua para o pino 3.3V	50mA
Memória Flash	32 KB (ATmega328P) dos quais 0.5 KB são usadaos pelo *bootloader*
Memória SRAM	2 KB (Atmega328P)
EEPROM	1 KB (ATmega328P)
Velocidade do Clock	16MHz
Dimensões	68,58mm x 53,34mm
Peso	25g

Fonte: Arduino (2019)

6.2 Sonar HC-SR04

O sonar usado é o modelo HC-SR04. Ele possui quatro pinos: VCC, *Trigger*, *Echo* e GND. O pino *Trigger* é responsável pela emissão de um pulso ultrassônico, e o pino *Echo*

indica o recebimento do pulso, de forma que é possível calcular distância do sensor ao alvo considerando o tempo percorrido da emissão ao recebimento do pulso.

No capítulo sobre mapeamento o funcionamento de um sonar foi devidamente detalhado, assim como os problemas e erros aos quais ele está sujeito.

Na Figura 15 encontra-se uma foto de um sonar HC-SR04.

Figura 15 – Sonar HC-SR04.

Fonte: FilipeFlop (2019)

A seguir, no Quadro 2, encontramos as especificações desse sensor. É importante considerar que a precisão descrita para esse sensor é para situações ideais, não considerando, por exemplo, a ocorrência de reflexões especulares ou leituras cruzadas.

Quadro 2 – Especificações do Sonar HC-SR04.

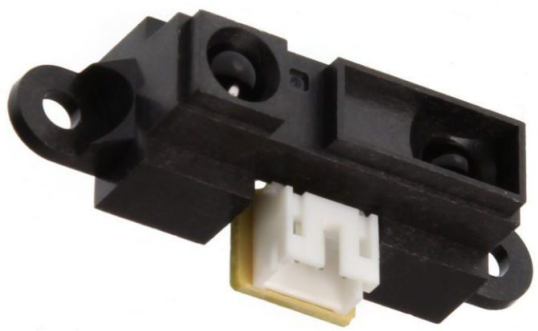
Tensão de alimentação	5V DC
Corrente consumida	15mA
Frequência de operação	40kHz
Distância máxima	4m
Distância mínima	2cm
Precisão	$>$ 3mm
Ângulo de medição	15°
Sinal de entrada (Trigger)	Pulso TTL de 10μs
Sinal de saída (Echo)	Pulso TTL proporcional à distância detectada
Dimensões	45mm x 20mm x 15mm

Fonte: ElecFreaks (2011, p. 1)

6.3 Sensor infravermelho Sharp GP2Y0A21YK0F

O sensor infravermelho Sharp GP2Y0A21YK0F é utilizado, juntamente com o sonar, para medir as distâncias do robô aos objetos inseridos no ambiente. Uma imagem deste sensor pode ser visto na Figura 16.

Figura 16 – Sharp GP2Y0A21YK0F.



Fonte: Vida de Silício (2018)

Este sensor possui apenas 3 pinos: VCC, GND e o pino A0 para saída analógica (SHARP, 2006). As especificações deste sensor se encontram no Quadro 3.

Quadro 3 – Especificações do Sharp GP2Y0A21YK0F.

Alimentação	4,5 - 5,5V
Corrente consumida	30mA
Distância máxima	80cm
Distância mínima	10cm

Fonte: Adaptado de Sharp (2006)

6.4 Módulo *Bluetooth* HC-06

Neste trabalho é utilizado um módulo *Bluetooth* HC-06 ZS-040 para realizar a comunicação entre e robô e o computador. Este módulo utiliza a versão 2 do padrão *Bluetooth*, permitindo abstrair toda a parte de comunicação sem fio e focar apenas no envio e recebimento de dados via comunicação serial (GUANGZHOU HC INFORMATION TECHNOLOGY, 2006). A Figura 17 traz uma imagem deste módulo.

O módulo *Bluetooth* HC-06 funciona apenas no modo *slave* (GUANGZHOU HC INFORMATION TECHNOLOGY, 2006), ou seja, ele apenas pode aceitar conexões de outros dispositivos, mas não pode parear outros dispositivos *Bluetooth*. Para este trabalho, o modo *slave* atende bem aos objetivos.

O HC-06 possui 4 pinos, sendo eles: VCC, GND, TX (para transmissão de dados) e RX (para recebimento de dados). A seguir, no Quadro 4, encontra-se especificações mais detalhadas deste módulo.

Figura 17 – Módulo *Bluetooth* HC-06.

Fonte: Vida de Silício (2019)

Quadro 4 – Especificações do Módulo *Bluetooth* HC-06.

Pinos	VCC, GND, TX e RX
Tensão de Alimentação	3,6 - 6V
Cobertura de sinal	até 10m
Especificação *Bluetooth*	v2.0 + EDR
Frequência	banda de 2.4GHz ISM
Taxa de dados (Assíncrono)	2.1Mbps (max) / 160 Kbps
Taxa de dados (Síncrono)	1Mbps / 1Mbps
Dimensões	43mm x 16mm x 7mm

Fonte: Adaptado de Guangzhou HC Information Technology (2006)

6.5 Servomotor SG90

O servomotor, também chamado simplesmente de servo, pode ser definido, de forma simplificada, como um motor que tem sua posição angular controlada através de um sinal PWM. Sendo assim, servomotores costumam ser utilizados para posicionar e manter um objeto em uma determinada posição. Neste trabalho, será utilizado um servomotor SG90 para controlar a posição dos sensores de alcance. Um servomotor deste modelo pode ser visto na Figura 18.

Servomotores geralmente possuem 3 terminais: VCC (vermelho), GND (preto ou marrom) e o terminal que recebe o sinal PWM (amarelo, laranja ou branco). As especificações do servomotor SG90 se encontram no Quadro 5.

Figura 18 – Servomotor SG90.

Fonte: Mota (2017b)

Quadro 5 – Especificações do Servomotor SG90.

Voltagem de operação	4.8 V
Ângulo de rotação	180°
Velocidade	0,1s/60° (4,8V)
Torque	1,8kg/cm (4,8V)
Dimensões	34.5mm x 32.5mm x 12.6mm
Peso	19g

Fonte: Adaptado de TowerPro (2019)

6.6 Motores DC e Ponte H

Para a locomoção do robô, serão usados dois motores DC de 80 RPM, com rodas de 68mm de diâmetro acopladas. As especificações dos motores se encontram no Quadro 6.

Quadro 6 – Especificações do Motor DC utilizado.

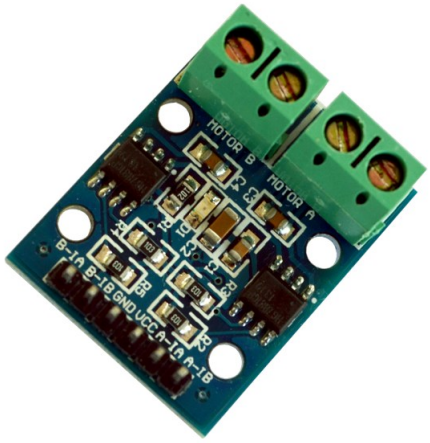
Tensão de operação:	3 - 6V
Diâmetro do eixo	5,35mm
Relação da caixa de redução	120:1
Rotação	80RPM (6V)
Dimensões	70mm x 37mm x 22,5mm (ignorando o eixo)
Peso	28g

Fonte: UsinaInfo (2019)

Para controlar ambos os motores é utilizado um módulo com o circuito integrado Ponte H HG7881 (L9110S) que permite controlar a velocidade e sentido da rotação dos motores. Esse modelo de Ponte H possui uma tensão de operação entre 2,5 e 12V. Um exemplo deste módulo se encontra na Figura 19.

Como indicado na Figura 13, o Frank alimenta o módulo de Ponte H com 4,8V. E, nos experimentos realizados, o robô manteve uma velocidade média de 14cm/s, alcançando uma velocidade máxima de 18cm/s.

Figura 19 – Ponte H HG7881.

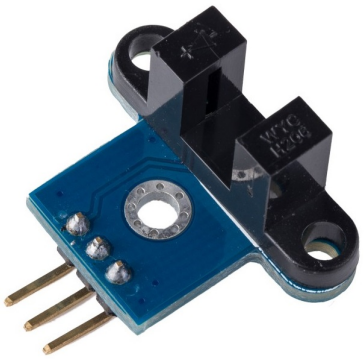


Fonte: Banana Robotics (2019)

6.7 *Encoders*

Como dito na seção sobre odometria, *encoders* são utilizados para contar a quantidade de giros das rodas do do robô. Neste trabalho será utilizado dois encoders, modelo H206, ilustrado pela Figura 20.

Figura 20 – Encoder H206.



Fonte: eLab Peers (2019)

Esse *encoder* emite um feixe de luz infravermelha que, quando rompido pelas perfurações do disco *encoder* emite um sinal, permitindo a contagem de interrupções do feixe e, por consequência, do quanto a roda se movimentou. Os discos *encoders* utilizados neste trabalho possuem 20 perfurações, como o da Figura 21.

Figura 21 – Disco *encoder*.

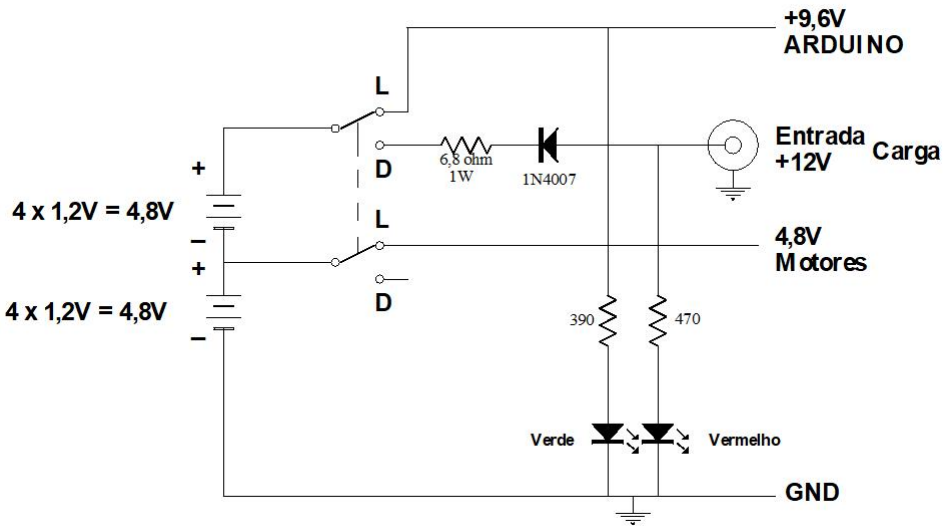
Fonte: Eletrogate (2019)

6.8 Fonte

Para a alimentação do robô, foi desenvolvida uma fonte utilizando 8 pilhas recarregáveis de 1,2V, com capacidade de 2500 mAh. Essa fonte fornece 9,6V para a alimentação do Arduino e 4,8V para a alimentação dos motores DC.

As pilhas podem ser recarregadas conectando uma fonte externa de 12V. A corrente de carga adotada foi de 10% do valor da corrente das baterias. Na Figura 22 encontra-se o desenho do circuito da fonte.

Figura 22 – Fonte de alimentação.



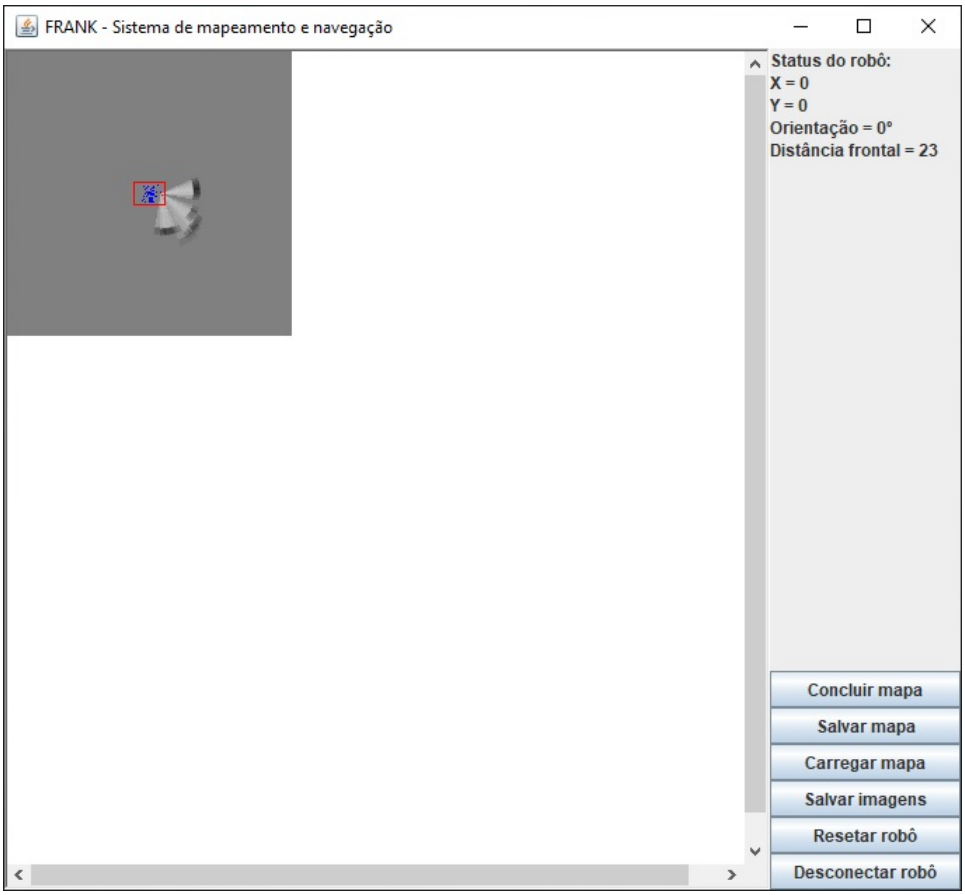
Fonte: Autor (2019)

7 Descrição do software

Neste capítulo é descrito o funcionamento do software desenvolvido. Para a implementação do sistema embarcado no robô foi utilizada a linguagem C++, modificada para Arduino, e para o sistema *Desktop* foi utilizada a linguagem Java. Na Figura 23 é apresentada a tela principal do sistema *Desktop*.

A linguagem Java foi escolhida devido a familiaridade do autor e por ela fornecer recursos e desempenho suficientes para a solução implementada.

Figura 23 – Tela principal do sistema Desktop.



Fonte: Autor (2019)

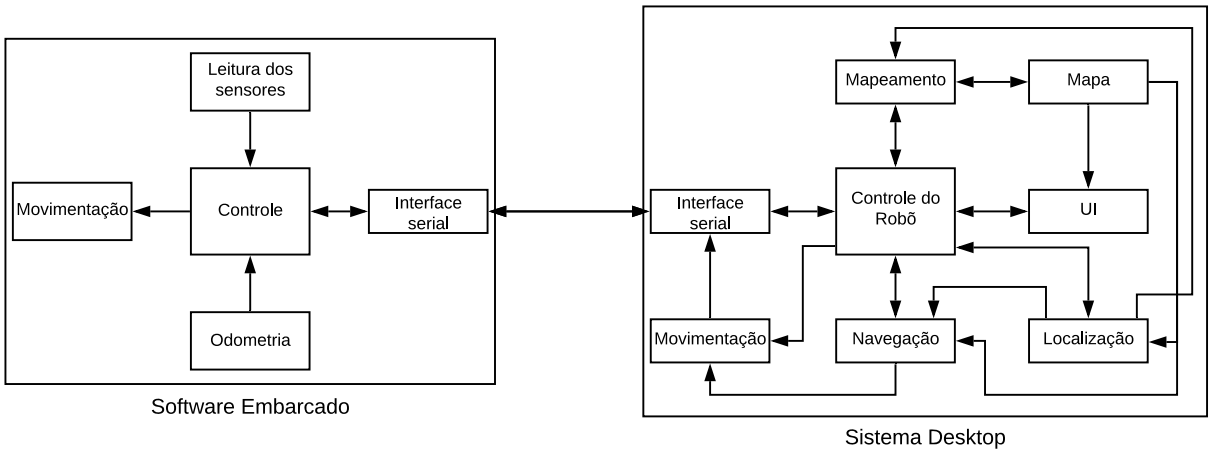
Através deste sistema é possível mapear o ambiente em que o robô está inserido, assim como navegar entre dois pontos criando uma rota entre a localização do robô e um ponto do mapa. Além disso, o sistema ainda permite controlar o robô manualmente e salvar o resultado de um mapeamento, que pode ser carregado posteriormente pela aplicação.

A seguir será apresentada a arquitetura geral deste sistema, detalhando os pontos mais importantes.

7.1 Arquitetura geral

Na Figura 24 temos um diagrama representando a arquitetura do sistema embarcado e do sistema *Desktop* e como eles se comunicam.

Figura 24 – Diagrama da arquitetura geral do Software.



Fonte: Autor (2019)

O software embarcado é responsável pelo controle de mais baixo nível do robô, lendo os dados dos sensores de distância, contando os pulsos dos *encoders* e acionando os atuadores para movimentar o robô e rotacionar os sensores de distância (utilizando o servomotor). Além disso, a estimação da localização do robô por odometria também é realizada no software embarcado.

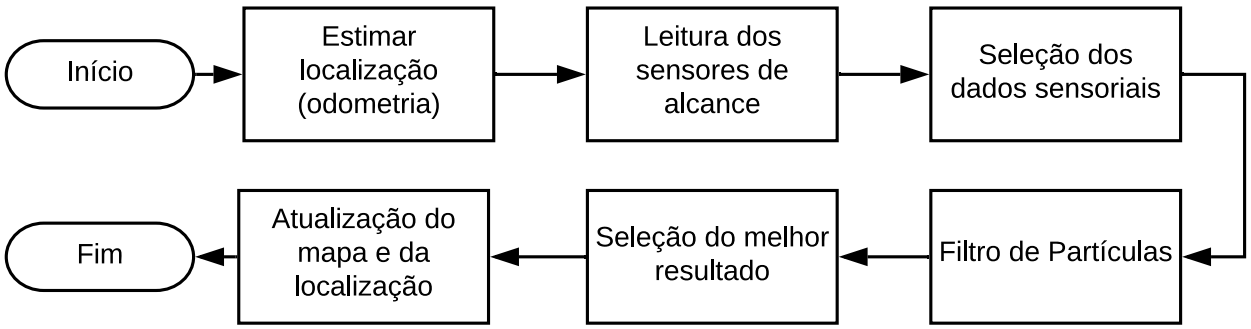
Todo o restante é realizado pelo sistema *Desktop*, que consiste basicamente em utilizar os dados sensoriais do robô para mapear o sistema e corrigir a estimativa da localização do robô, assim como criar rotas de navegação dentro do mapa e enviar as instruções de movimentação para o robô.

Como o foco deste trabalho é o estudo e implementação de um sistema de mapeamento e localização baseado em SLAM, nas seções a seguir serão detalhadas os módulos relativos a este problema.

7.2 Mapeamento e Localização

O processo de mapeamento pode ser descrito pelo fluxograma da Figura 25. Quando dado o comando para o robô realizar o mapeamento, ele inicialmente considera a localização do robô estimada pela odometria e realiza a leitura dos sensores de distância.

Figura 25 – Fluxograma do processo de mapeamento.



Fonte: Autor (2019)

Com os dados "brutos" obtidos dos sensores, é realizada a seleção dos dados sensoriais. Isso é feito comparando os dados do sonar com os dados do sensor infravermelho, selecionando a média da distâncias, desde que a diferença das leituras esteja abaixo de um certo limiar. Caso a quantidade de dados selecionados desta forma seja muito pequena, o sistema tenta selecionar mais alguns dados. Para isso, é importante considerar os erros que as leituras podem apresentar.

Através dos experimentos realizados, pôde-se perceber que o sonar apresenta diversos problemas com reflexões especulares, e quando isso acontece a leitura costuma ser muito maior que a distância real. Por este motivo, leituras do sonar que ultrapassem 2 metros são descartadas, pois possuem alta probabilidade de serem leituras incorretas. A dificuldade maior é quando as leituras são relativas aos cantos das paredes, neste caso ocorre reflexão especular e a distância lida é maior que a real, porém a diferença não é grande a ponto disso poder ser facilmente percebido e os dados descartados.

O sensor infravermelho, por sua vez, não apresenta muitos problemas com reflexões especulares nos cantos do ambiente, de forma que as distâncias lidas correspondem bem com os dados reais. Por outro lado, este sensor ainda apresenta imprecisões em suas leituras, porém com o sonar havia a vantagem de que descartando leituras de grandes distâncias, a maioria das leituras incorretas eram eliminadas. O mesmo não acontece com o sensor infravermelho, na realidade, a maioria das leituras incorretas geradas por ele apresentavam distâncias menores que as reais, conforme mostrado no próximo capítulo. Esta característica do sensor dificulta a remoção de leituras incorretas.

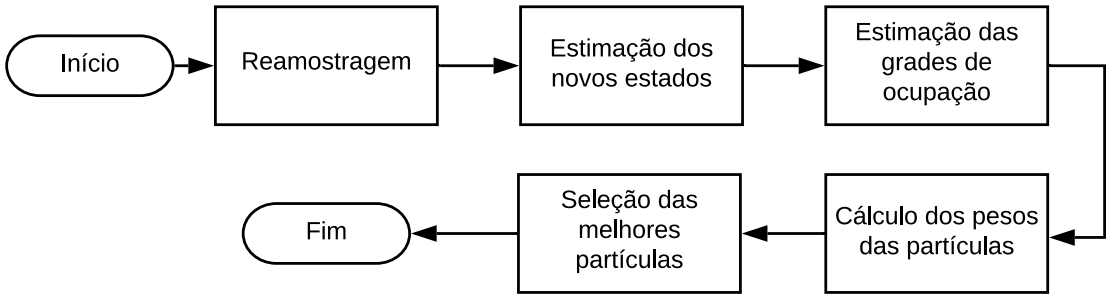
Por isso, devido aos erros e imprecisões de ambos os sensores, é importante o processo de seleção de dados, visando reduzir o efeito destes problemas, melhorando a precisão dos dados.

Após isso, um filtro de partículas é utilizado, recebendo como parâmetros a estimativa

inicial da localização do robô e os dados sensoriais selecionados. Este filtro de partículas irá gerar diversas partículas, que armazenam uma estimativa da localização do robô junto com a estimativa do mapa local. Então a melhor partícula é selecionada, e a localização do robô e o mapa são atualizados definitivamente.

A exploração do ambiente durante o processo de mapeamento é realizada pelo usuário, enviando comandos de locomoção para o robô. Além disso, a seleção da melhor partícula pode ser feita de forma manual, ainda que o sistema calcule o peso de cada partícula e, na maioria dos casos, como pôde ser avaliado durante os experimentos, a partícula de maior peso corresponda a melhor estimativa de localização e do mapa. Essas simplificações, ainda que gerem um certo impacto na autonomia do robô, foram realizadas devido a complexidade da implementação de um sistema que resolva o problema do SLAM com desempenho razoável e com sensores de baixo custo.

Figura 26 – Fluxograma do filtro de partículas implementado.



Fonte: Autor (2019)

O filtro de partículas implementado para a correção da localização é representado pelo fluxograma da Figura 26, consistindo de uma adaptação do algoritmo apresentado anteriormente para um filtro de partículas, no Capítulo 3. Nos experimentos realizados neste trabalho foram utilizadas 70 partículas. Dentro do filtro de partículas, é importante destacar a estimação das grades de ocupação e o cálculo dos pesos das partículas.

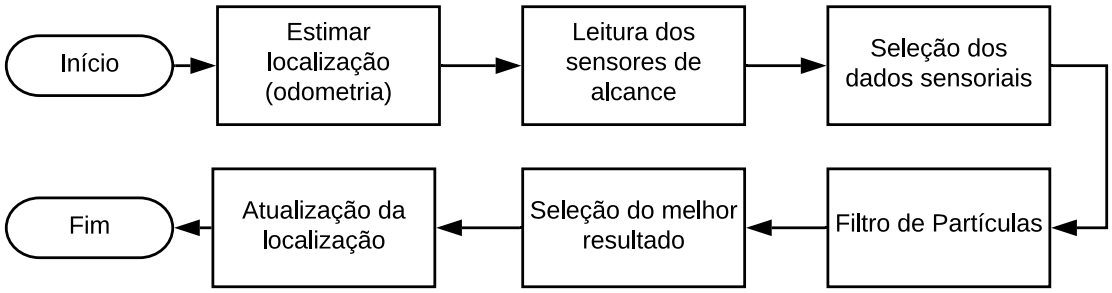
A estimação das grades de ocupação é baseada na teoria apresentada anteriormente, e é realizada considerando a localização de cada partícula e os dados sensoriais lido. Para diminuir o tempo computacional necessário, é estimada apenas uma grade de ocupação para cada conjunto de dados sensoriais, considerando que o robô está localizado na origem do sistema de coordenadas, e esta grade é transladada e rotacionada conforme necessário para se adequar a cada partícula.

O cálculo do peso das partículas, por sua vez, é realizado considerando o casamento do mapa anterior com o mapa local de cada partícula. Mais especificamente, esse casamento considera a sobreposição de células consideradas ocupadas, quanto mais células ocupadas sobrepostas, maior o peso da partícula. Além disso, a sobreposição de células mais próximas ao

robô são consideradas mais relevantes do que sobreposições mais distantes, pois posições mais distantes tendem a apresentar erros maiores.

O problema deste casamento de características é que nem sempre a melhor sobreposição corresponde ao mapa que melhor representa o mundo real. Por exemplo, caso o robô esteja navegando por um corredor, as leituras realizadas em dois pontos distintos, porém próximos, deste corredor tendem a ser muito semelhantes. Desta forma, provavelmente o "melhor" resultado estimado será a sobreposição dos dois pontos, de forma que as grades de ocupação também se sobreponham. Porém, isto não representa a realidade, uma vez que o robô teve um deslocamento e a sobreposição das grades de ocupação não deveria ser total.

Figura 27 – Fluxograma do processo de correção da localização.



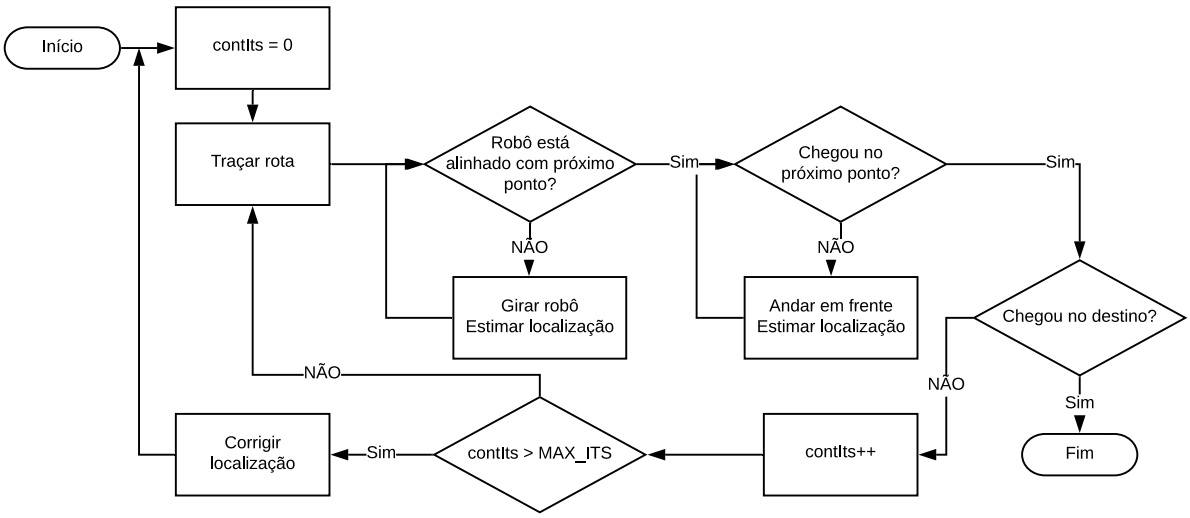
Fonte: Autor (2019)

Por fim, o fluxograma apresentado na Figura 27 descreve o processo de correção da localização utilizado durante a navegação. Esse processo é similar ao processo de mapeamento, com a diferença de que apenas a localização é corrigida, não havendo alterações no mapa. Na prática, o número de dados sensoriais utilizados é reduzido, tornando o processo mais rápido que o mapeamento.

7.3 Arquitetura da Navegação

Já em relação a navegação, uma vez que foi gerado um mapa do ambiente em que o robô está inserido, pode-se adotar uma abordagem deliberativa. Desta forma, o algoritmo de navegação implementado consiste basicamente em traçar uma rota até o ponto de destino e enviar as instruções de movimentação para que o robô siga essa rota. Tal processo é representado na Figura 28.

Figura 28 – Fluxograma do processo de navegação.



Fonte: Autor (2019)

Para traçar a rota, primeiramente a grade de ocupação que representa o ambiente é pré-processada, classificando cada célula em: ocupada, livre e segura para navegação, livre e não segura para navegação. As células classificadas como livres mas não seguras são aquelas mais próximas as células ocupadas, pelas quais se o robô navegar corre o risco de colidir com algum obstáculo, devido as imprecisões na localização.

Com o mapa pré-processado, a rota entre a localização do robô e o ponto de destino é traçada utilizando o algoritmo A*, buscando encontrar o menor caminho entre os dois pontos, mas priorizando sempre as células livres e seguras para navegação e nunca passando por células ocupadas.

Com a rota traçada, basta que o robô a siga. Porém, devido as imprecisões na obtenção da sua localização, é possível que ele se desvie da rota. Por isso, após alguns movimentos sua localização é corrigida, utilizando os dados dos sensores exteroceptivos e o filtro de partículas, e uma nova rota é gerada a partir da nova localização estimada do robô.

8 Resultados

Ao longo do desenvolvimento deste trabalho diversos experimentos foram realizados, com o objetivo de testar e avaliar os sensores e algoritmos utilizados. Neste capítulo é apresentado

e analisado o resultado de alguns dos experimentos finais.

8.1 Erros de odometria

A precisão da odometria impacta diretamente no problema da localização do robô e, por consequência, nos processos de mapeamento e localização. Sendo assim, foram realizados alguns experimentos para avaliar o crescimento dos erros de odometria.

Para a realização dos experimentos em relação ao deslocamento linear, o robô se moveu a partir de um referencial fixo até um determinado ponto, baseando-se na distância calculada pela odometria. A partir desta movimentação, utilizando uma trena e um transferidor, foram medidos os valores reais de deslocamento e orientação.

Os dados experimentais estão dispostos na Tabela 1, assim como o cálculo da média e do desvio padrão dos erros.

Tabela 1 – Dados resultantes dos experimentos de deslocamento linear do robô.

	100cm		200cm	
Teste	ϵ_{lin}(cm)	ϵ_{ang}(graus)	ϵ_{lin}(cm)	ϵ_{ang}(graus)
1	0	7	-1	1
2	-1	-5	-1	-3
3	-2	1	-3	-4
4	-2	0	-2	0
5	-1	1	1	-1
6	0	2	-1	-5
7	1	-4	-2	-4
8	-1	-3	-3	3
Média	-0.75	-0.13	-1.5	-1.63
Desvio	1.04	3.62	1.31	2.64

Fonte: Autor (2019)

Com estes dados é possível aproximar o crescimento dos erros lineares e angulares a partir de um deslocamento linear pelas funções descritas na Equação 8.1 e na Equação 8.2, respectivamente.

$$E_{ll}(\Delta l) = 0,0075\Delta l \tag{8.1}$$

$$E_{l\theta}(\Delta l) = 0,01\Delta l \tag{8.2}$$

Também foram realizados experimentos considerando movimentos de rotação do robô. Nestes experimentos não foram detectados erros lineares consideráveis, por isto na Tabela 2 consta apenas os erros angulares.

Tabela 2 – Dados resultantes dos experimentos de deslocamento angular do robô.

	90°	180°	360°
Teste	ϵ_{ang}(graus)	ϵ_{ang}(graus)	ϵ_{ang}(graus)
1	9	19	27
2	6	9	22
3	7	10	28
4	4	10	26
5	9	19	23
6	12	7	29
Média	7.83	12.33	25.83
Desvio	2.79	5.28	2.79

Fonte: Autor (2019)

Dessa forma é possível aproximar o crescimento dos erros angulares a partir de um deslocamento angular pela função descrita na Equação 8.3.

$$E_{\theta\theta}(\Delta\theta) = 0,07\Delta\theta + 1,08 \tag{8.3}$$

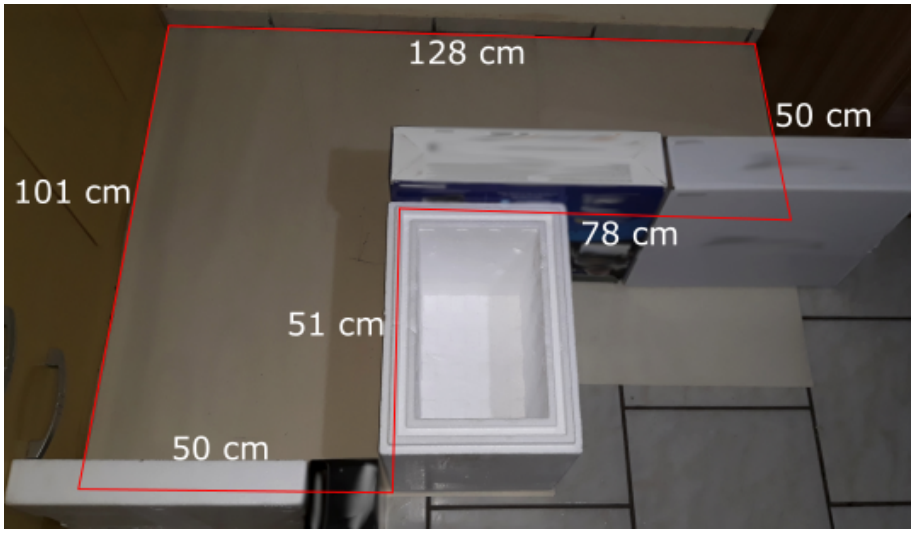
Desta forma conseguimos uma aproximação do crescimento dos erros gerados pela odometria, em que destaca-se os erros angulares. Estes erros impactam diretamente no processo de mapeamento e navegação do robô. Na próxima seção, inclusive, é apresentado o resultado de um mapeamento utilizando apenas a estimativa da localização obtida pela odometria.

8.2 Resultados do mapeamento

O foco deste trabalho foi implementar uma solução de mapeamento do ambiente, para qual foi necessário o estudo e aplicação de técnicas de SLAM.

Para testar o processo de mapeamento, o robô Frank foi colocado no ambiente apresentado na Figura 29. Este é um ambiente de dimensões reduzidas, uma vez que o Frank dificilmente obteria bons resultados em um ambiente maior, devido a limitação e imprecisão dos seus sensores.

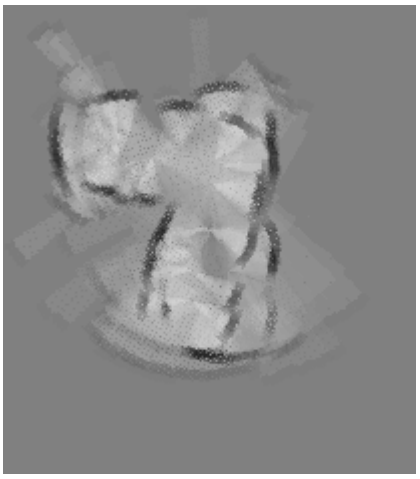
Figura 29 – Ambiente utilizado nos experimentos.



Fonte: Autor (2019)

Para ilustrar a necessidade de uma estimativa mais precisa da localização do robô durante o processo de mapeamento, na Figura 30 é apresentado o resultado de um mapeamento utilizando apenas a odometria para obter a localização do robô. Nas Figuras 31 e 32 tenta-se sobrepor a planta do ambiente no mapa gerado. Neste experimento foram utilizados tanto dados do sonar quanto do sensor infravermelho.

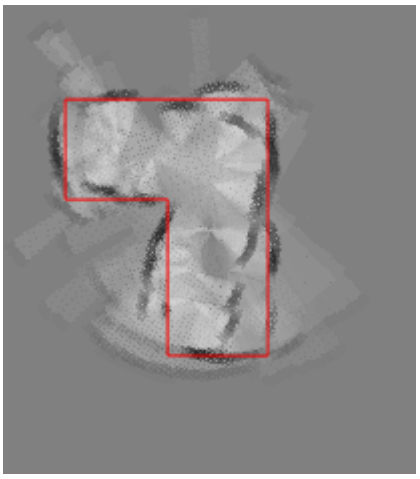
Figura 30 – Mapeamento sem correção da localização.



Fonte: Autor (2019)

Neste resultado pode-ser observar o quanto o erro de odometria influencia na qualidade do mapa. Como apresentado na seção anterior, o maior erro apresentado nas estimativas de localização obtidas pela odometria é em relação a orientação do robô, e isto também pode ser percebido no mapa gerado, onde o ambiente parece se "rotacionar" ao longo do processo de mapeamento.

Figura 31 – Primeira comparação do mapa sem correção da localização com a planta do ambiente.



Fonte: Autor (2019)

Figura 32 – Segunda comparação do mapa sem correção da localização com a planta do ambiente.

Fonte: Autor (2019)

Os próximos resultados apresentados já utilizam um filtro de partículas para estimar a localização do robô. Nas Figuras 33 e 34 temos o resultado do mapeamento utilizando apenas os dados do sonar.

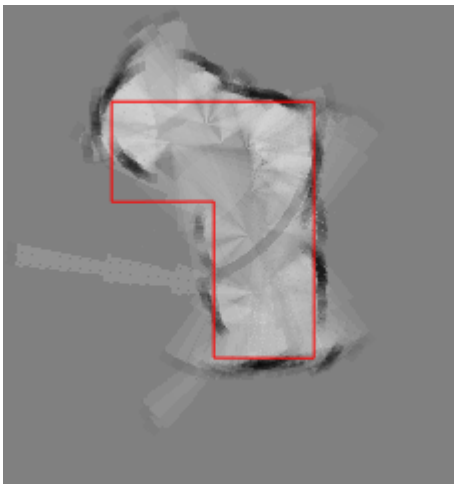
Neste resultado pode-se perceber que as células considerados ocupadas estão melhor definidas do que no resultado anterior (as células ocupadas apresentam uma coloração mais próxima do preto do que no mapa anterior, indicando que a probabilidade desta célula estar ocupada é maior), isto ocorre devido ao uso do filtro de partículas, que busca realizar o casamento de características, considerando a sobreposição destas células.

Por outro lado, este exemplo ainda apresenta uma certa distorção do ambiente, em que a parte superior parece ter sido rotacionada e não corresponde com a planta. Podemos

Figura 33 – Mapeamento utilizando apenas dados do sonar.

Fonte: Autor (2019)

Figura 34 – Comparação do mapa obtido com sonar com a planta do ambiente.



Fonte: Autor (2019)

considerar que este mapa está divido em dois mapas locais, que correspondem relativamente bem a geometria do ambiente, mas que não estão conectados corretamente. Isto é um problema, porém já é menos grave do que acontecia no experimento anterior. Isso porque agora é utilizado um filtro de partículas para corrigir a localização do robô, e este filtro pode conseguir uma boa estimativa da localização do robô dentro de cada mapa local.

Vale ressaltar que neste mapa pode-se perceber alguns dos erros que ocorrem com as leituras do sonar, especialmente as reflexões especulares. Boa parte delas pôde ser eliminada ao remover as leituras que ultrapassam um certo limiar, mas algumas ainda persistiram, o que pode ser percebido pelo fato do mapa apresentar células livres e ocupadas além dos limites do ambiente, em especial nos cantos do ambiente.

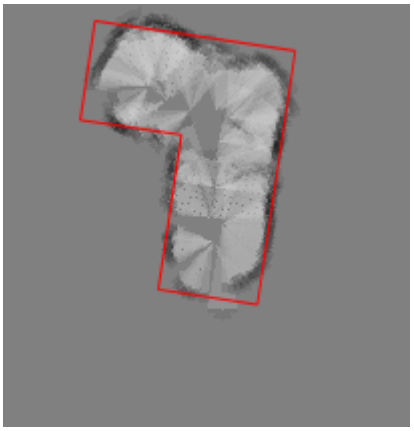
Nas Figuras 35 e 36 são apresentados os resultados de um mapeamento do ambiente

utilizando apenas os dados do sensor infravermelho. Em comparação com os experimentos anteriores, este já obteve um resultado mais satisfatório.

Figura 35 – Mapeamento utilizando apenas dados do sensor de distância infravermelho.

Fonte: Autor (2019)

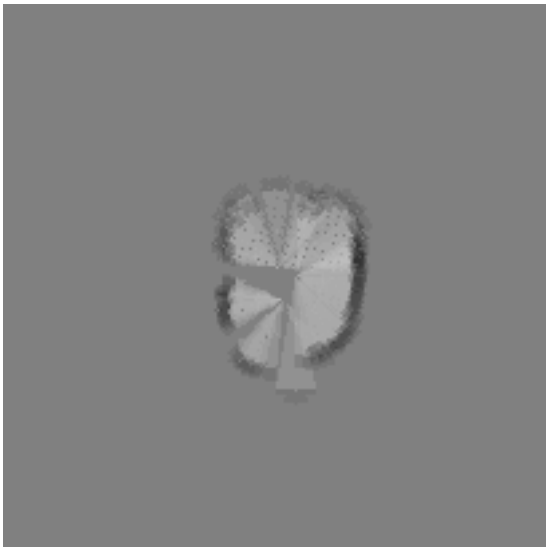
Figura 36 – Comparação do mapa obtido com sensor de distância infravermelho com a planta do ambiente.



Fonte: Autor (2019)

O sensor infravermelho, porém, apresenta uma certa característica que pode gerar problemas durante o processo de mapeamento. No sonar as reflexões especulares ou leituras de pontos que estão além do seu alcance máximo geram leituras com distâncias muito grandes (considerando o seu alcance), de forma que o simples ato de eliminar as leituras que estejam acima de um certo limiar garante que boa parte das leituras "erradas", que destoam bastante das distâncias reais, seja eliminada. Já no sensor infravermelho isso não acontece, as leituras "erradas" não se comportam muito diferente das leituras "corretas". Em primeiro lugar, o alcance deste sensor é muito menor que o sonar, o que dificultaria a determinação de um limiar de corte. Mas isto também não resolveria, porque muitas das leituras erradas na realidade são menores do que as distâncias reais, tornando muito mais difícil determinar se uma leitura é "correta" ou não.

Figura 37 – Mapeamento parcial do ambiente utilizando o sensor infravermelho.



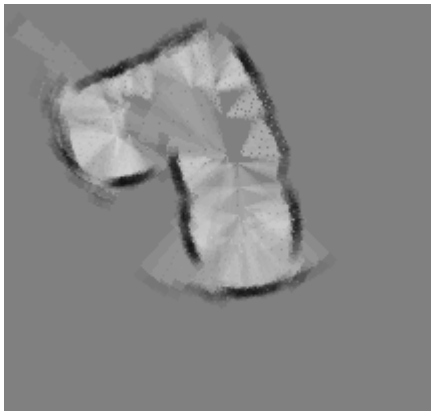
Fonte: Autor (2019)

A Figura 37, por exemplo, mostra o resultado de uma etapa do experimento de mapeamento que gerou o mapa da Figura 35. Pode-se perceber, neste mapa, que existem células consideradas como ocupadas onde deveria ser espaço livre, decorrente de leituras de distâncias muito menores que as reais.

O mapa final acabou não sendo muito prejudicado por isso, neste caso, o que se deve ao funcionamento do algoritmo de mapeamento baseado nas grades de ocupação. Com as leituras seguintes, a probabilidade de ocupação das células foram atualizadas e o que prevaleceu foi as leituras que as indicavam como células livres.

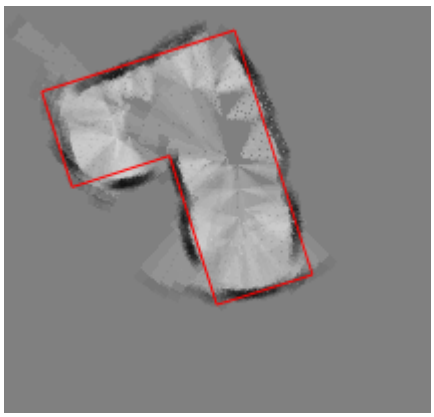
Por fim, visando reduzir os problemas de ambos os sensores, são utilizado as leituras de ambos, o que permite uma melhor seleção dos dados, conforme descrito no Capítulo 7. Nas Figuras 38, 39, 40 e 41 são apresentados os resultados de dois experimentos realizados com essa abordagem.

Figura 38 – Primeiro mapeamento utilizando ambos os sensores.



Fonte: Autor (2019)

Figura 39 – Comparação do primeiro mapa utilizando ambos os sensores com a planta do ambiente.

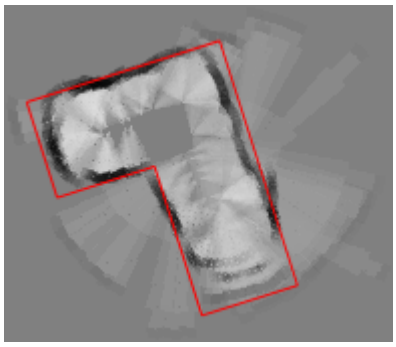


Fonte: Autor (2019)

Figura 40 – Segundo mapeamento utilizando ambos os sensores.

Fonte: Autor (2019)

Figura 41 – Comparação do segundo mapa utilizando ambos os sensores com a planta do ambiente.



Fonte: Autor (2019)

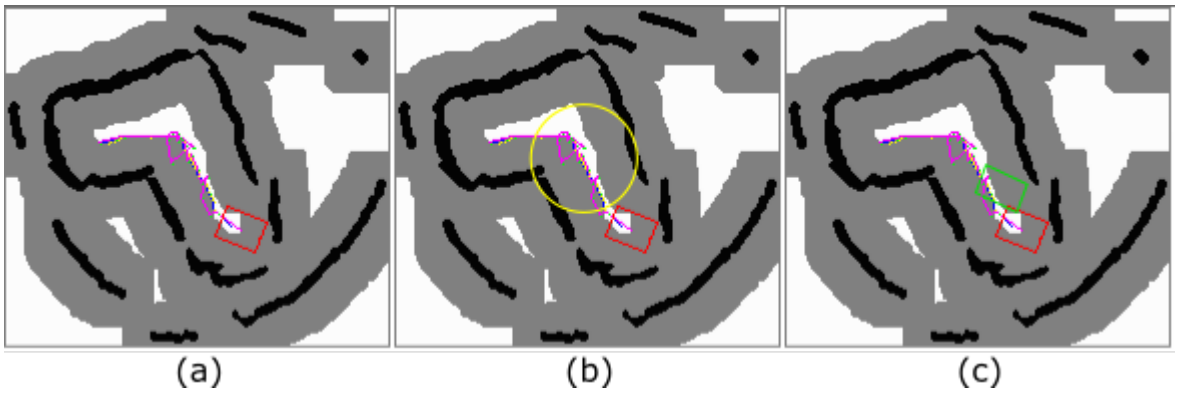
Os resultados são relativamente melhores que os dos experimentos anteriores. Além disso, é importante considerar que são utilizados apenas sensores de baixo custo, e dessa forma, com uma taxa de erro alta, especialmente se comparado com sensores mais robustos. Sendo assim, pode-se considerar que os resultados obtidos são satisfatórios para fins de estudo do problema.

8.3 Resultados da navegação

Uma vez que o ambiente tenha sido mapeado, o mapa gerado pode ser usado para auxiliar o robô na sua navegação. Para testar o método de navegação implementado e descrito no Capítulo 7 foi utilizado o mesmo ambiente dos experimentos de mapeamento, assim como um dos mapas gerados.

Primeiramente, o mapa escolhido foi processado, classificando as células da grade de ocupação como: células ocupadas (indicadas pela cor preta), células livres e seguras para navegação (indicadas pela cor branca) e células livres mas não seguras (cor cinza). Com o mapa pronto para a navegação, o sistema permite selecionar o ponto de destino do robô através do ponteiro do mouse.

Figura 42 – Primeiro experimento de navegação.



Fonte: Autor (2019)

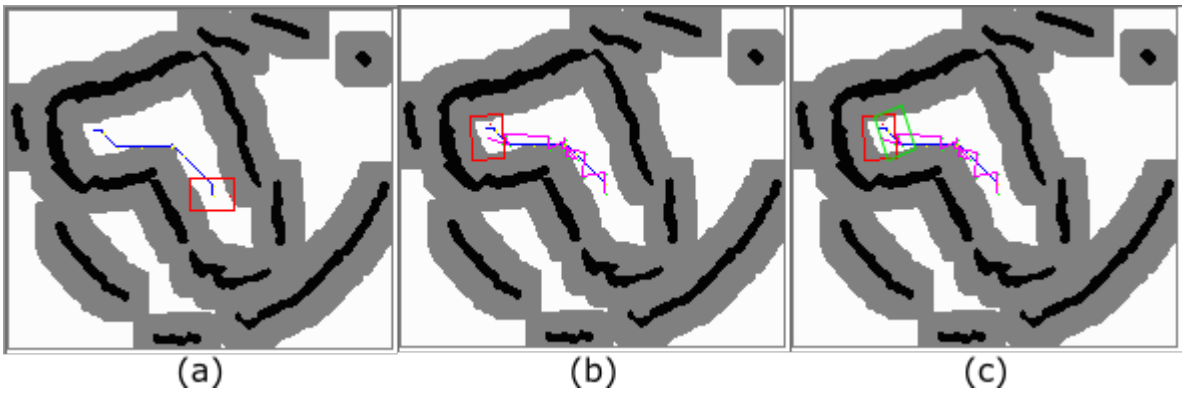
Na Figura 42 é apresentado o resultado de um dos experimentos de navegação. No primeiro quadro (Figura 42 a) a imagem foi gerada pelo software, onde os segmentos de reta azul (pouco visíveis nesta figura) representam a rota inicial gerada pelo algoritmo de navegação, e os segmentos de rota de cor magenta representam a rota por qual o robô realmente navegou. É importante considerar que existem erros na precisão no desenho da rota percorrida pelo robô, pois ela se baseia nas localizações estimadas do robô, portanto sujeitas a imprecisões.

Na Figura 42 b é destacada a região em que o robô teve mais problemas ao navegar. Esta foi a região em que ele enfrentou mais problemas para determinar sua localização, o que

o levou ao ficar girando e andando de um lado para o outro, até que foi capaz de corrigir sua posição de forma satisfatória e, assim, atingir seu objetivo.

Já na Figura 42 c comparamos a posição final estimada do robô com a posição real observada no mundo real (representada pelo retângulo verde). Neste experimento se observou um erro considerável na localização do robô. Porém, vale destacar que a orientação estimada do robô está muito próxima da orientação real, o que é um resultado interessante ao considerarmos que a maior imprecisão da odometria do Frank se dá justamente na estimativa da orientação, e que o robô realizou diversos giros durante sua tentativa de seguir a rota planejada.

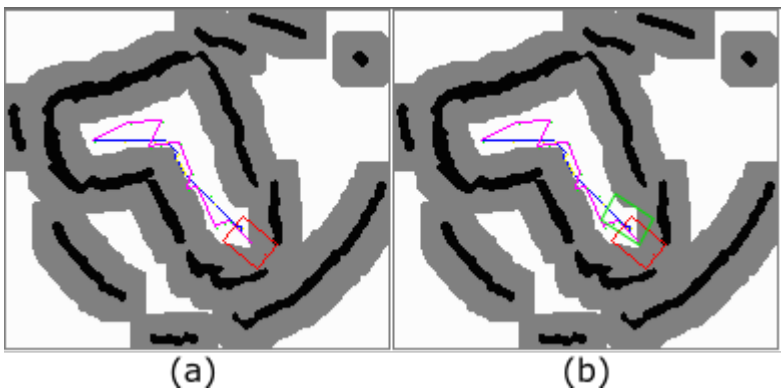
Figura 43 – Segundo experimento de navegação.



Fonte: Autor (2019)

Após este experimento foram realizadas algumas melhorias no algoritmo de navegação, permitindo a obtenção de resultados ligeiramente melhores. Na Figura 43, por exemplo, pode-se perceber que o robô "fugiu" menos da rota traçada inicialmente, ainda que tenha feito alguns giros e contornos desnecessários, e a posição final estimada e a real estão muito mais próximas que no experimento anterior.

Figura 44 – Terceiro experimento de navegação.



Fonte: Autor (2019)

O terceiro experimento foi realizado logo após o segundo, tendo como posição inicial a posição final do experimento anterior, e o resultado está representado na Figura 44. O resultado

final também é relativamente satisfatório, principalmente se comparado com o primeiro, tanto ao considerarmos a diferença da posição final estimada e da real, quanto ao considerarmos a trajetória realizada.

Vale destacar que a correção da localização, neste experimento, foi essencial para o sucesso da navegação, visto que a posição inicial estimada já não correspondia com muita precisão à posição real. Se apenas a odometria fosse utilizada, o acúmulo de erros dificilmente permitiria que o robô atingisse o seu objetivo com a mesma precisão.

Neste último experimento, porém, ao realizar a curva, próximo a um canto, o robô colidiu sua parte traseira com a parede. Nos primeiros experimentos realizados colisões eram mais comuns. Com as melhorias do algoritmo as colisões reduziram, porém não há garantia de que não ocorrerão, especialmente colisões com as partes laterais ou traseira do robô, que não são dotadas de sensores que permitam perceber a aproximação de um obstáculo de forma reativa.

Outra desvantagem é o tempo de navegação. Nos experimentos realizados, em um ambiente de pequenas dimensões, o robô levou de 3 a 5 minutos para concluir o seu percurso.

9 Conclusões

Neste trabalho foi apresentado um sistema de mapeamento e navegação para robôs móveis, tendo como referencial teórico diversos trabalhos desenvolvidos nessa área, em especial os que tratam sobre o SLAM. Para este sistema foi considerado também que seria utilizado um robô móvel com sensores de baixo custo, visto que a maioria das soluções encontradas utilizavam sensores de custo mais elevado, principalmente os chamados *laser rangefinders*.

Com isso em mente, conclui-se que os resultados foram satisfatórios, do ponto de vista de estudo do problema, em especial com a utilização de sensores de baixo custo. Embora tanto o processo de mapeamento quanto o processo de navegação ainda sofram com as limitações e imprecisões dos sensores e da odometria, o sistema foi capaz de criar um mapa para um ambiente interno de dimensões reduzidas sem grandes distorções e permitindo a correção da localização do robô com uma precisão muito melhor do que se teria apenas com os dados da odometria. E com o mapa gerado, o robô também foi capaz de navegar de sua localização inicial até um determinado ponto, alcançando o seu objetivo com uma precisão aceitável.

Esses resultados apontam que é possível a construção e utilização de robôs de custos não tão elevados para a realização de tarefas complexas, como é o caso do SLAM. Pensando nisso, a seguir são sugeridas algumas possibilidades de melhorias que podem ser exploradas em trabalhos futuros, tendo em vista as limitações e problemas apresentados.

Em primeiro lugar, pode-se pensar em melhorar a qualidade dos dados dos sensores

exteroceptivos, utilizados tanto para a estimação do mapa quanto para a estimação da localização. Para isso poderia-se utilizar sensores de melhor precisão e/ou maior alcance (mas buscando ainda utilizar sensores de baixo custo) e utilizar um número maior de sensores, posicionados nas laterais e traseira do robô. A utilização de mais sensores também ajudaria evitar colisões durante o processo de navegação, permitindo o robô agir reativamente ao detectar a aproximação de um objeto através de um dos seus sensores.

Com a utilização de diversos sensores, seria necessário um estudo mais aprofundado de técnicas de fusão sensorial para a integração dos dados gerados por esses sensores, permitindo a obtenção de dados mais precisos. Existem diversas técnicas de fusão sensorial, deste a utilização de métodos matemáticos como os filtros de Kalman até a aplicação de redes neurais artificiais.

Ainda pensando em adições ao hardware do robô, a adição de uma bússola pode melhorar consideravelmente a precisão da localização do robô, em especial a sua orientação. Isso também permitiria a utilização de outros métodos de SLAM, como os que utilizam filtro de Kalman para a correção da localização, os quais dificilmente obteriam bons resultados com a precisão atual da odometria do Frank. Além disso, pode-se pensar na melhoria da comunicação do robô com o computador, utilizando um módulo *Wireless* no lugar do módulo *Bluetooth*, por exemplo.

Já em relação ao software, os principais esforços em obter melhorias deveriam ser em relação a correção da localização. O método apresentado neste trabalho, que consiste em um filtro de partículas, nem sempre retorna a partícula que mais se aproxima da posição real do robô. Isso dificulta o processo de navegação, e para que isto não afetasse tanto a qualidade do mapa, foi permitido ao usuário escolher qual partícula produzia um mapa melhor (normalmente a melhor partícula se encontrava entre a primeira e quinta posição). Isso, porém, afetou a autonomia do robô, não permitindo que ele realizasse um mapeamento totalmente autônomo. Por tal motivo também não foi implementado um algoritmo de exploração para o processo de mapeamento, o que também deveria ser explorado em um trabalho futuro.

Outra possibilidade de melhoria, tanto para o processo de mapeamento quanto para de localização, é a detecção de *landmarks* nas grades de ocupação, em especial, de retas. Além disso permitir a utilização de outros métodos, baseados em *landmarks*, isso também poderia ser utilizado para selecionar melhor os dados sensoriais utilizados, ou até estimar a leitura dos sensores. Isso porque uma das dificuldades na correção da localização, em especial durante a navegação, reside nos problemas característicos de cada sensor, como as reflexões especulares. Porém, uma vez que o mapa do ambiente seja representado por retas, conhecendo a posição do sonar, por exemplo, pode-se estimar o ângulo de incidência do sinal sonoro que será emitido, e dessa forma avaliar se haverá reflexão especular ou não. Isso pode ser utilizado para eliminar informações possivelmente corrompidas, ou ainda para simular qual seria a leitura realizada considerando tais reflexões.

Além disso, para melhorar a eficiência do sistema pode-se pensar no uso de processa-

mento paralelo, em especial nas operações envolvendo as grades de ocupação, pois cada a probabilidade de ocupação de cada célula é independente das células vizinhas.

Referências

ARDUINO. *Arduino Uno Rev3*. 2019. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 21 set. 2019.

BANANA ROBOTICS. *HG7881 (L9110) Dual Channel Motor Driver Module*. 2019. Disponível em: <https://www.bananarobotics.com/shop/HG7881-(L9110)-Dual-Channel-Motor-Driver-Module>. Acesso em: 24 set. 2019.

BECKER, T. *Navegação de um robô móvel baseado em um modelo de consciência artificial*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2015. Disponível em: <http://repositorio.utfpr.edu.br/jspui/bitstream/1/1605/1/CT_PPGCA_M_Becker%2C%20Thiago_2016.pdf>. Acesso em: 06 mar. 2019.

BIGHETI, J. A. *Navegação de robôs em ambientes internos usando SLAM*. Dissertação (Mestrado) — Universidade Estadual Paulista, 2011. Disponível em: <http://hdl.handle.net/11449/87178>. Acesso em: 06 mar. 2019.

BONTEMPO, A. P. *Uma Abordagem Híbrida para Localização e Mapeamento Simultâneos para Robôs Móveis com Sonares Através de Filtro de Kalman Estendido*. Dissertação (Mestrado) — PUC-Rio, 2012.

BORENSTEIN, J.; EVERETT, H.; FENG, L. et al. Where am i? sensors and methods for mobile robot positioning. *University of Michigan*, Citeseer, v. 119, n. 120, p. 27, 1996.

BROOKS, R. A. New approaches to robotics. *Science*, American Association for the Advancement of Science, v. 253, n. 5025, p. 1227–1232, 1991.

CHENAVIER, F.; CROWLEY, J. L. Position estimation for a mobile robot using vision and odometry. In: IEEE. *Proceedings 1992 IEEE International Conference on Robotics and Automation*. Nice, 1992. p. 2588–2593.

DOUCET, A.; FREITAS, N. D.; MURPHY, K.; RUSSELL, S. Rao-blackwellised particle filtering for dynamic bayesian networks. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Springer, 2000. p. 176–183.

ELAB PEERS. *H206 Opto-Coupler Speed Sensor With Kit*. 2019. Disponível em: <https://www.elabpeers.com/h206-opto-coupler-speed-sensor.html>. Acesso em: 24 set. 2019.

ELECFREAKS. *Ultrasonic Ranging Module HC - SR04*. 2011. Disponível em: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>. Acesso em: 23 set. 2019.

ELETROGATE. *Disco Encoder para Sensor de Velocidade*. 2019. Disponível em: <https://www.eletrogate.com/disco-encoder-para-sensor-de-velocidade>. Acesso em: 31 out. 2019.

ELFES, A. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, IEEE, v. 3, n. 3, p. 249–265, 1987.

ELFES, A. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Tese (Doutorado), Pittsburgh, PA, USA, 1989. AAI9006205.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. *Computer*, IEEE, v. 22, n. 6, p. 46–57, 1989.

ELIAZAR, A.; PARR, R. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In: IJCAI. *Proceedings of the International Joint Conferences on Artificial Intelligence Organization*. Acapulco, 2003. v. 3, p. 1135–1142.

FABRIZI, E.; SAFFIOTTI, A. Extracting topology-based maps from gridmaps. In: IEEE. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. San Francisco, 2000. v. 3, p. 2972–2978.

FILIPEFLOP. *Sensor de Distância Ultrassônico HC-SR04*. 2019. Disponível em: <https://www.filipeflop.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/>. Acesso em: 23 set. 2019.

GUANGZHOU HC INFORMATION TECHNOLOGY. *Product Data Sheet: HC-06*. Cantão, 2006. Rev. 2. Disponível em: <https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf>. Acesso em: 31 out. 2019.

HEINEN, F. J. *Sistema de controle híbrido para robôs móveis autônomos*. Dissertação (Mestrado) — Universidade do Vale do Rio do Sinos, 2002.

HäHNEL, D.; BURGARD, W.; FOX, D.; THRUN, S. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: IEEE. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. Las Vegas, 2003. v. 1, p. 206–211.

MAFFEI, R.; JORGE, V.; KOLBERG, M.; PRESTES, E. Segmented dp-slam. In: IEEE. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tóquio, 2013. p. 31–36.

MARCHI, J. Navegação de robôs móveis autônomos: estudo implementação de abordagens. Florianópolis, SC, 2001. Disponível em: <https://repositorio.ufsc.br/xmlui/handle/123456789/81441>. Acesso em: 06 mar. 2019.

MARTINELLI, A.; SIEGWART, R. Estimating the odometry error of a mobile robot during navigation. In: EPFL. *Proceedings of European Conference on Mobile Robots*. Varsóvia, 2003.

MATARIĆ, M. J. *The robotics primer*. Cambridge: Mit Press, 2007.

MILSTEIN, A. Occupancy grid maps for localization and mapping. *Motion planning*, InTech, pp381-408, p. 381–408, 2008.

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, v. 593598, 2002.

MOTA, A. *O que é Arduino e como funciona?* 2017. Disponível em: <https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/>. Acesso em: 21 set. 2019.

MOTA, A. *O que é servomotor?* 2017. Disponível em: <https://portal.vidadesilicio.com.br/o-que-e-servomotor/>. Acesso em: 23 set. 2019.

OLIVEIRA, J. R. d. *Um sistema integrado para navegaçao autônoma de robôs móveis*. Dissertação (Mestrado) — Universidade de Sao Paulo, 2010.

PATSKO, L. F. *Aplicações, funcionamento e utilização de sensores*. 2006. Disponível em: <https://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_aplicacoes_e_funcionamento_de_sensores.pdf>. Acesso em: 19 out. 2019.

RODRIGUES, D. P. *Estudo comparativo de métodos de localização para robôs móveis baseados em mapa*. Dissertação (Mestrado) — Universidade Estadual de Campinas, 2013.

SECCHI, H. A. Uma introdução aos robôs móveis. *Universidade Nacional de San Juan*, San Juan, 2008.

SHARP. *GP2Y0A21YK0F*. [S.l.], 2006. Disponível em: <https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf>. Acesso em: 31 out. 2019.

SILVA, R. M. da; GARCIA, T. R.; CUADROS, M. A. d. S. L.; GUERRA, R. da S.; GAMARRA, D. F. T. Controle da trajetória de um robô móvel utilizando odometria e lógica fuzzy. 2018.

SOUZA, A. A. d. S. *Mapeamento com sonar usando grade de ocupação baseado em modelagem probabilística*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2008.

SOUZA, E. S. de; BORGES, J. C. N. *Projeto de readequação de um robô*. 2009. Disponível em: <http://www.dainf.ct.utfpr.edu.br/~fabro/IF66J/Relatorios_Finais/2009_2/MonografiaOI3_2009_2_ReadequacaoBellator.pdf>. Acesso em: 03 out. 2019.

THOMSEN, A. *O que é Arduino?* 2014. Disponível em: <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 21 set. 2019.

THRUN, S.; FOX, D.; BURGARD, W.; DELLAERT, F. Robust monte carlo localization for mobile robots. *Artificial intelligence*, Elsevier, v. 128, n. 1-2, p. 99–141, 2001.

TOWERPRO. *SG90 Digital*. 2019. Disponível em: <http://www.towerpro.com.tw/product/sg90-7/>. Acesso em: 31 out. 2019.

USINAINFO. *Motor DC 3-6V 80RPM com Caixa de Redução e Eixo Duplo 120:1*. 2019. Disponível em: <https://www.usinainfo.com.br/micromotores-e-motores/motor-dc-3-6v-80rpm-com-caixa-de-reducao-e-eixo-duplo-1201-3162.html>. Acesso em: 24 set. 2019.

VIDA DE SILÍCIO. *Sensor de Distância Infravermelho Sharp GP2Y0A21YK0F*. 2018. Disponível em: <https://portal.vidadesilicio.com.br/sensor-distancia-infravermelho-sharp-gp2y0a21yk0f/>. Acesso em: 02 out. 2019.

VIDA DE SILÍCIO. *HC-06 - Módulo Bluetooth*. 2019. Disponível em: <https://www.vidadesilicio.com.br/hc-06-modulo-bluetooth>. Acesso em: 24 set. 2019.