UNIVERSIDADE ESTADUAL DO MARANHÃO – UEMA
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO E SISTEMAS – PECS

MESTRADO PROFISSIONAL EM ENGENHARIA DE COMPUTAÇÃO E SISTEMAS

FRANCISCO DOS SANTOS VIANA

REDES NEURAIS APLICADAS NA ESTRATÉGIA DE PONDERAÇÃO DE PARTÍCULAS EM SLAM

São Luís
2019

UNIVERSIDADE ESTADUAL DO MARANHÃO – UEMA
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO E SISTEMAS – PECS

MESTRADO PROFISSIONAL EM ENGENHARIA DE COMPUTAÇÃO E SISTEMAS

FRANCISCO DOS SANTOS VIANA

REDES NEURAIS APLICADAS NA ESTRATÉGIA DE PONDERAÇÃO DE PARTÍCULAS EM SLAM

Dissertação apresentada ao programa do Mestrado Profissional em Engenharia de Computação e Sistemas da Universidade Estadual do Maranhão como parte dos requisitos para obtenção do título de Mestre em Engenharia da Computação.

Orientador: Prof. Dr. Areolino de Almeida Neto.

São Luís
2019

Viana, Francisco dos Santos.
Redes neurais aplicadas na estratégia de ponderação de partículas em SLAM / Francisco dos Santos Viana. – São Luís, 2019.

75 f

Dissertação (Mestrado) – Curso de Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2019.

Orientador: Prof. Dr. Areolino de Almeida Neto.

1.Robótica móvel. 2.SLAM. 3.Filtro de partículas. 4.Inteligência artificial. I.Título

CDU: 004.8

FRANCISCO DOS SANTOS VIANA

REDES NEURAIS APLICADAS NA ESTRATÉGIA DE PONDERAÇÃO DE PARTÍCULAS EM SLAM

Dissertação apresentada ao programa do Mestrado Profissional em Engenharia de Computação e Sistemas da Universidade Estadual do Maranhão como parte dos requisitos para obtenção do título de Mestre em Engenharia da Computação, sob a orientação do Prof. Dr. Areolino de Almeida Neto.

Aprovado em: 31 / 01 /2019

Prof. Dr. Areolino de Almeida Neto (Orientador)
Universidade Federal do Maranhão

Profª. Drª. Patrícia Helena Moraes Rêgo
Universidade Estadual do Maranhão

Prof. MSc. José Pinheiro de Moura
Universidade Estadual do Maranhão

Prof. Dr. Luciano Buonocore
Universidade Federal do Maranhão

A minha mãe por tudo que me ensinou e
por todo o incentivo e suporte durante
toda a minha vida acadêmica.

Agradecimentos

Em primeiro lugar, agradeço a Deus.

Agradeço a minha mãe Elma, por tudo que faz por mim.

A minha amiga Jéssica, pelo apoio e sugestões.

Aos meus colegas de mestrado pelos momentos de anseios e estudos juntos.

Agradeço a todos os professores que tive durante o mestrado pelo conhecimento e experiência transmitidos.

Ao meu orientador, professor Areolino, pela orientação e compreensão nos momentos de dificuldades.

Ao professor Luciano Buonocore por todo o suporte durante o desenvolvimento deste trabalho.

"Algumas pessoas veem a distância, outras veem a ponte”
(Autor desconhecido)

Resumo

Este trabalho apresenta uma nova metodologia para a ponderação da importância das partículas em um filtro de partículas usado para localização e mapeamento de um ambiente por um robô móvel, conhecido na literatura clássica como *Simultaneous Localization and Mapping* (Localização e Mapeamento Simultâneos). Neste trabalho, uma integração de inteligência artificial e métodos probabilísticos foi usada no processo de estimação da pose do robô. Para tanto, este foi dotado de um sensor de baixo custo e que apresenta ruído. O uso deste tipo de sensor torna-se prático porque possui custo menor e não necessita de muito processamento computacional, quando comparado, por exemplo, aos sensores que necessitam de tratamento de imagem. Para realizar sua operação, o robô deve ser capaz de manter uma estimativa da sua localização com base nos dados fornecidos por meio de fontes sensoriais, ou seja, adquirir e utilizar conhecimento sobre o mundo ao seu redor e assim possuir a habilidade de reconhecer obstáculos. No processo de aquisição de dados provenientes de fontes sensoriais com ruído, faz-se necessário o uso de técnicas de tratamentos destas informações. Nesse contexto, métodos probabilísticos tornam-se úteis nas aplicações de filtragem de dados. Neste trabalho, foi usada uma rede neural artificial integrada ao filtro de partículas para minimizar o erro na estimação da pose do robô. A metodologia é baseada em um procedimento simples de centros, raios e processo de correspondência durante a exploração do ambiente pelo robô. Nesta abordagem, uma rede neural artificial do tipo perceptron de multicamadas determina o raio de um círculo, cujo centro é a medida bruta do sensor, para o filtro de partículas ponderar as partículas e, assim, estimar a pose em cada estado do robô no ambiente. Finalmente, são apresentadas discussões sobre tal abordagem, bem como, a complexidade computacional, associação dos dados e os resultados gerados.

Palavras-chaves: Robótica móvel, SLAM, Filtro de Partículas, Inteligência Artificial.

Abstract

This work presents a new methodology for weighing the importance of particles in a particle filter used for locating and mapping an environment by a mobile robot, known in the classic literature as Simultaneous Localization and Mapping. In this work, an integration of artificial intelligence and probabilistic methods was used in the robot pose estimation process. For this, it was endowed with a sensor of low cost and that presents noise. The use of this type of sensor becomes practical because it has a lower cost and does not require a lot of computational processing when compared, for example, to sensors that need image processing. In order to perform its operation, the robot must be able to maintain an estimate of its location based on the data provided through sensory sources, that is, acquire and use knowledge about the world around it and thus have the ability to recognize obstacles. In the process of acquiring data from noise sensing sources, it is necessary to use techniques for processing this information. In this context, probabilistic methods become useful in data filtering applications. In this work, an artificial neural network integrated to the particle filter was used to minimize the error in the robot pose estimation. The methodology is based on a simple procedure of centers, rays and matching process during the exploration of the environment by the robot. In this approach, an artificial neural network of the multilayer perceptron type determines the radius of a circle, whose center is the raw sensor measurement, for the particulate filter to weigh the particles and thus estimate the pose in each state of the robot in the environment. Finally, discussions about such an approach are presented, as well as the computational complexity, association of the data and the results generated.

Key-words: Mobile Robotics, SLAM, Particle Filter, Artificial Intelligence.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

BP	Backpropagation
EKF	Extended Kalman Filter
EQM	Erro Quadrático Médio
MLP	Multi-Layer Perceptron
PF	Particles Filter
ROS	Robot Operating System
RNA	Rede Neural Artificial
RBPF	Filtro de Partículas Rao-Blackwellised
RBF	Radial Basis Function
SLAM	Simultaneous Localization and Mapping
SOM	Self Organizing Maps
VREP	Virtual Robot Experimentation Platform

Lista de Símbolos

$p(x^t)$	Probabilidade do conjunto x no instante t
$p^-(x^t)$	Probabilidade predita de x no instante t
η	Coeficiente de normatização
$p(x^t\|y)$	Função densidade da predição
X_k	Variável de estado no instante de tempo *t*
Y_k	Observações no instante de tempo *t*
f	Função não linear do estado no instante *t*-1
g	Função não linear da observação no instante *t*
V_k	Ruído do processo
U_k	Ruído da medição do sensor
Q_k	Covariância do ruído do processo
R_k	Covariância do ruído da medida do sensor
W	Peso das partículas
N_{eff}	Tamanho efetivo da amostra
v_k	Saída da combinação linear dos sinais de entrada
y_k	Sinal de saída de um neurônio *k*
φ	Função de ativação de um neurônio artificial
$f(n)$	Função matemática de ativação de uma RNA
$p_i(s)$	Conjunto de partículas no estado *s*
ϑ_j	Vetor de correspondência
η_i	Importância da partícula *i*
η	Importância das partículas
w_i	Peso normalizado da partícula *i*
r	Raio de distribuição das partículas

Sumário

1 Introdução

A área da robótica sempre chamou atenção da sociedade, em especial por sua comodidade e conforto gerados para todos. Tendo assumido um papel de fundamental importância em aplicações industriais, no setor aeroespacial, em domótica, no transporte de materiais em hospitais, em veículos agrícolas autônomos, em veículos urbanos autônomos, em guia em museus e outras aplicações (BURGARD, 1998; NEWMAN, 1999 e BAILEY, 2002).

Nas últimas décadas ocorreram avanços significativos nas tecnologias, em especial nos recursos de *hardware* e *software*. Em *hardware*, têm-se computadores e sistemas embarcados miniaturizados de custo reduzido, de maior robustez, de menor consumo de energia e com grande capacidade de processamento. Aliado a isso, têm-se os atuadores seguindo a mesma tendência, possibilitando a criação de robôs com maior grau de precisão, eficiência e elevado número de aplicações. Com tudo isso, os *softwares* ganharam mais espaço, podendo ser desenvolvidos novos algoritmos mais robustos nas áreas de controle, tomada de decisão, processamento de imagem, reconhecimento de voz, entre outros (ROMERO, 2014).

Diante de tais avanços, o robô deixou de ser um sistema limitado e passou a ser protagonista nos diversos setores da atividade humana. Com a inserção de técnicas mais sofisticadas de controle e inteligência artificial, as aplicações passaram a ter maior complexidade e interatividade com o homem. Atualmente, existem robôs de alta precisão usados em ambiente cirúrgico, limpeza de ambiente, entretenimento, setor aeroespacial e outros. Em seu livro, Robótica Móvel, Romero (2014) afirma que:

> Esse aumento de aplicações de robôs tem levado vários países a realizarem investimentos elevados em pesquisa e desenvolvimento. Por exemplo, a Coreia do Sul planejou investir aproximadamente 1 bilhão de dólares em 10 anos para o desenvolvimento de tecnologias em robótica para área de educação, na formação de professores robôs de crianças na pré-escola e jardim de infância. A União Europeia planejou investir 536 milhões de euros entre 2007-2012 para pesquisa relacionada às áreas de robótica e cognição.

Rodney Brooks, pesquisador do MIT e fundador da iRobot, afirmou em 2006:

> Estou convencido de que os robôs estão hoje onde os computadores estavam na década de 1980. Foi nessa época que eles começaram a aparecer em nosso meio, do mesmo modo que os robôs estão hoje. É claro, foram

> necessários ainda outros 15 anos antes de os computadores difundirem-se em nossa vida. Eu penso que em 15 anos, os robôs estarão em todos os lugares, assim como estão atualmente a internet e o e-mail.

Sebastian Thrun, um dos grandes especialistas em robótica, prevê que até 2030 será comum carros sendo conduzidos em vias públicas por robôs. Bill Gates (1999) em um artigo fez a seguinte afirmação:

> Os computadores *desktops* PCs irão deixar o seu lugar em cima das mesas para passarem a ver, ouvir, tocar e manipular objetos em lugares em que nós estaremos fisicamente presentes.

Os avanços descritos acima permitiram que pesquisas de novas técnicas na área de inteligência artificial e controle pudessem ser desenvolvidas. Novos desafios surgiram à medida que as aplicações foram avançando. Quando se trata da robótica móvel, foco deste trabalho, um dos grandes desafios é a localização e mapeamento de robô, como tarefas concorrentes, em determinado ambiente onde este pretende atuar.

Trabalhos mais recentes em robótica móvel baseiam-se em técnicas probabilísticas para estimar a pose do robô e a aquisição do mapa do ambiente por onde navega. Essas técnicas são necessárias porque os dados oriundos dos sensores e do ambiente em que ele opera possuem incertezas. Tais incertezas, se não tratadas, afetam o processo de reconhecimento do ambiente e, consequentemente, a atuação na exploração do robô.

O processo simultâneo de localização e mapeamento do ambiente pelo robô é chamado de SLAM (*Simultaneous Localization and Mapping*). Este processo é uma tarefa consideravelmente mais complexa do que as técnicas existentes apenas para mapeamento ou localização. No SLAM, o módulo de localização fornece uma estimativa da pose do robô referenciada a um sistema global ou mesmo local de coordenadas do ambiente. Havendo necessidade, o mapa pode ser atualizado com informações de localização obtidas pelo robô. Assim, pode-se atualizar o mapa com as informações descritas no sistema de coordenadas, em seguida, o mapa é usado pelo sistema de localização, levando em conta as informações mais atuais do ambiente. Neste processo, existe uma forte correspondência entre localização e mapeamento, onde erro em uma das etapas reflete na outra (THRUN, 2004 e BIGHETI, 2011).

Aplicações de localização e mapeamento simultâneos tem crescido nos últimos anos, além da complexidade dos sistemas envolvidos. As técnicas probabilísticas podem ser aperfeiçoadas a fim de que resultados com erros menores possam ser obtidos. Esforços por parte

da sociedade científica para melhorar e desenvolver técnicas aplicadas na inteligência artificial vem sendo realizados nos principais centros de pesquisas do mundo. Há um consenso entre os pesquisadores que, embora já existam importantes trabalhos em robótica, há muito a avançar, principalmente nas técnicas aplicadas no SLAM. Dessa forma, faz-se necessário que os sistemas tenham elevado grau de precisão na obtenção dos dados do ambiente para fins de localização e geração do mapa. Assim, o robô pode navegar pelo ambiente com segurança e realizar corretamente as atividades que lhe for dada.

Uma das técnicas usadas em SLAM é o filtro de partículas. É uma técnica probabilística, baseada no método de Monte Carlos, que visa a estimar a pose do robô. Quando o mapa e alguns parâmetros do filtro aumentam (número de partículas), a complexidade computacional também aumenta, gerando lentidão na geração do mapa (THRUN, 2002). Este trabalho propõe o uso de redes neurais artificiais para auxiliar o filtro de partículas na estimação da pose do robô, tendo como principal contribuição, fornecer uma metodologia que melhore o processo de ponderação das partículas baseada em sensores de baixo custo.

1.1 Motivação e justificativa

A localização e o mapeamento simultâneos do ambiente são um problema que consiste na capacidade do robô de construir um mapa a partir de um ambiente desconhecido e, simultaneamente, localizar-se no interior desse espaço, podendo operar de forma autônoma ou não. Ao realizar a tarefa do SLAM, o robô observa no ambiente que navega alguns pontos de interesse para serem utilizados como *marcos* durante o processo. A estimação da pose consiste na correlação (diferença) entre os novos *marcos*, observados no movimento atual do robô, com aqueles encontrados anteriormente.

Diversos sensores são usados no sistema de percepção do robô, sendo os mais comuns o *laser range finder*, câmera de vídeo, sonar e encoders. Os sensores *laser* e de câmera de vídeos baseiam-se, respectivamente, na leitura da distância de obstáculos e processamento de imagem. Os sensores sonar e *encoders* fazem, respectivamente, a leitura da distância do obstáculo à frente do robô e medem o deslocamento do robô baseado no movimento das rodas (processo de odometria). Os dois primeiros tipos de sensores demandam grande poder de processamento computacional no processo de extração das informações. Já os dois últimos, são baratos e fazem uso de pouco processamento (LEONARD *et al.*, 1992).

Sensores de baixo custo são também de baixa confiabilidade quando comparados a métodos mais otimizados (alto poder de processamento), em especial os sensores visuais.

Quando são usados, necessitam de correções nos valores medidos pelos mesmos, sendo usados filtros para esta tarefa.

A solução de problemas que envolvem a localização e o mapeamento simultâneos são importantes, porque um robô só será de fato inteligente se for capaz de localizar-se e orientar-se de modo autônomo. Este processo é de fundamental importância para atividades em que o acesso pelo homem é extremamente difícil, como fundo do mar, exploração de superfícies de outros planetas. Nesses locais de acesso dificultado, os sistemas de posicionamento, como o sistema GPS, são ineficazes e inaplicáveis.

No entanto, existem fortes incentivos ao uso de SLAM em novas aplicações da robótica, seja de cunho científico ou comercial: cadeira de rodas inteligente para deficientes, robô de baixo custo fazendo limpeza, guia turístico em aeroportos, etc. O processo de localização e mapeamento tem sido durante décadas um dos grandes desafios para a comunidade de pesquisadores na área de robótica, tendo despertado interesses em várias áreas como sistemas com visão computacional e inteligência artificial para desenvolvimento de aplicações reais (ROMERO, 2014).

Diante do exposto até então, a motivação deste trabalho é fornecer uma solução em SLAM, baseada na integração de filtro probabilístico com inteligência artificial, com o objetivo de melhorar o processo de obtenção das informações do ambiente explorado pelo robô, dotado de um sensor de baixo custo e que apresenta ruído.

1.2 Objetivos

Esta seção apresenta os objetivos gerais e específicos deste trabalho.

1.2.1 Objetivo geral

Apresentar uma nova metodologia, baseada na integração de inteligência artificial e métodos probabilísticos, para estimação da pose do robô em aplicações que envolvem localização e mapeamento simultâneos.

1.2.2 Objetivos específicos

- Aplicar rede neural artificial para obtenção do estado de um robô, baseando-se em um modelo de filtro probabilístico para estimação da pose;
- Desenvolver uma nova metodologia de ponderação da importância das partículas em um filtro de partículas;

- Aperfeiçoar aplicações em SLAM para robôs que usam sensores de baixo custo.

1.3 Organização do trabalho

Este trabalho é dividido nos seguintes capítulos:

- No Capítulo 2, é apresentada uma explicação geral sobre as tarefas de localização, mapeamento e SLAM. Além disso, serão abordados os conceitos do filtro de partículas e das redes neurais artificiais;

- No Capítulo 3, são apresentados os fundamentos do método proposto, onde serão explicados os conceitos matemáticos pertinentes à estratégia do cálculo de centros e raios na metodologia de ponderação das partículas;

- No Capítulo 4, são mostrados e discutidos os resultados obtidos nos experimentos com a metodologia proposta;

- O Capítulo 5 apresenta a conclusão deste trabalho e trabalhos futuros.

2 Referencial teórico

Estimação de estado é um problema que envolve diversas áreas: controle automático, predição de séries temporais, rastreamento, navegação de robôs, etc. Uma abordagem bastante comum de ser usada na modelagem desses problemas é baseada em espaço de estado. Quando o processo é linear e gaussiano, há uma solução ótima na forma fechada como o filtro de Kalman e o filtro de Kalman estendido, que é uma solução monomodal para sistemas que apresentam não-linearidades que veio em acréscimo ao filtro de Kalman. No entanto, nos problemas do mundo real, que possuem comportamento não linear e não-gaussianos, a informação disponível chega de modo sequencial e com ruído. Para tal tipo de processo, não há soluções fechadas, mas os estados ocultos do modelo podem ser estimados usando filtro de partículas, também fundamentado na teoria da estimação Bayesiana (BIGHETI, 2011).

O desenvolvimento deste capítulo apresenta as técnicas que serão usadas, bem como sua fundamentação matemática para aplicações em SLAM. Neste capítulo, serão abordados os fundamentos de localização e mapeamento, SLAM, modelos de filtros e redes neurais artificiais.

2.1 Localização

A pose em que o robô se encontra em um ambiente 2D do qual navega é a sua localização. Esta possui, como base para a sua estimação, as informações obtidas do mapa, distância para outros obstáculos, informações da sua pose anterior, informações passadas pelo vetor de controle (movimento a ser realizado) e a leitura obtida dos sensores. Em outras palavras, localizar significa estimar a pose (coordenada no eixo x, coordenada no eixo y e seu ângulo de orientação), a cada instante, no ambiente que o robô navega (SANTANA, 2011**)**.

Visando a localizar o robô de forma mais confiável, usam-se as informações de sonares, sensores infravermelhos e odômetros. Esse conjunto de dados sensoriais é importante, principalmente para lidar com as mudanças existentes no ambiente e com os ruídos dos sensores (THRUN, 1998).

Os métodos de localização são classificados em duas categorias: relativa e absoluta. A localização relativa fornece a pose do robô em relação a uma pose inicial, enquanto a absoluta fornece a pose global do robô e não faz uso das posições anteriores (BORENSTEIN e FENG, 1996).

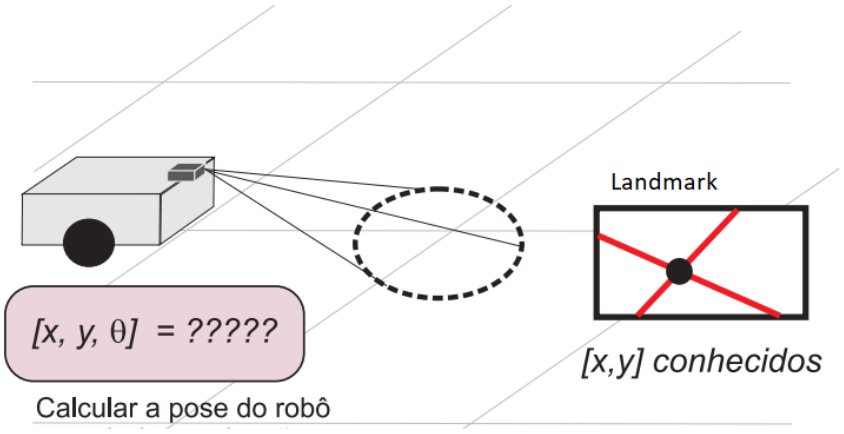
Para estimar a localização do robô, são normalmente usadas as informações de odometria por meio de sensores acoplados às suas rodas de tração. Esse método possui diversos erros, alguns de natureza mecânica das rodas e engrenagens, outros oriundos de irregularidades, inclinações e viscosidades existentes no terreno a ser explorado, além de desequilíbrio na distribuição de massa no robô.

Para resolver o problema da odometria, uma solução muito utilizada é a fusão sensorial. Este método consiste na combinação (fusão) dos dados vindos de outros sensores ao odômetro para aumentar a precisão da localização (SASIADEK e HARTANA, 2000). O uso de sensores em conjunto facilita o emprego de técnicas que associam as leituras atuais com as anteriores, para estimar a pose atual. Como as leituras provenientes dos sensores são armazenadas nos mapas, é de extrema importância que tais medidas não contenham erros para evitar falhas no processo de localização (SANTANA, 2011 e THRUN, 1998).

Na localização absoluta, o robô é capaz de reconhecer, através dos seus sensores, elementos distintos, como paredes, elementos distintos nos objetos ou ainda marcos (landmarks), estes últimos colocados intencionalmente no ambiente. No geral, os marcos estão em posições fixas e com as informações adquiridas pelos sensores, é possível identificá-los no ambiente. Sabendo-se a localização dos marcos é possível realizar a localização global do robô (BORENSTEIN *et al.*, 1996).

Na Figura 2.1, tem-se um exemplo de realização do processo de localização. O marco encontrado, a partir de uma câmera no robô, é a interseção entres as restas do piso. Conhecendo-se a localização desse ponto nas coordenadas globais, o sistema calcula a pose do robô no ambiente.

Figura 2.1: Processo de localização baseado em *landmarks*

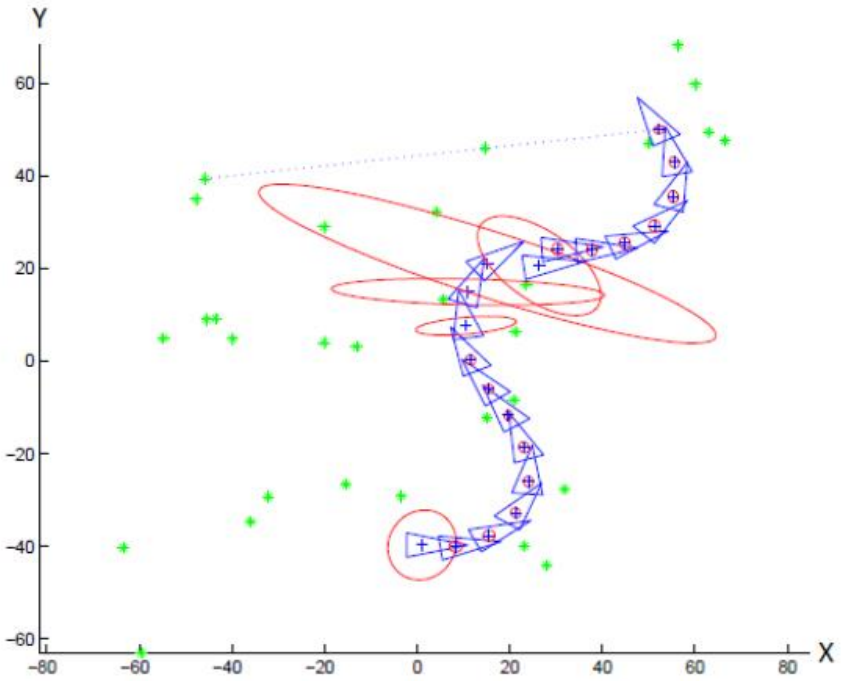


Fonte: Santana (2011)

O problema da localização é caracterizado por Newman (2006) da seguinte forma: "é dado um mapa **M** contendo um conjunto de obstáculos e um fluxo de observações de medições entre o veículo e os obstáculos. Parte-se do princípio que as associações entre as medições e os obstáculos observados são verdadeiras. Assume-se que o veículo de uso na navegação está equipado com um sensor de distância que retorna a distância para os obstáculos". O robô, quando em qualquer ponto do ambiente, deve navegar para conseguir identificar os obstáculos através dos sensores e, simultaneamente, calcular a distância que o separa de tal maneira que sempre deverá ter a sua pose atual no mapa.

Na Figura 2.2, tem-se a resolução de um típico problema da localização. Os asteriscos verdes representam os obstáculos. O robô, representado pelo triângulo azul, é inserido no ambiente e realiza sua primeira observação obtendo a sua localização e a incerteza dessa medição (círculo vermelho). À medida que ele realiza mais observações, essa incerteza diminui cada vez mais, porém ao parar de realizar observações (esse momento é representado na Figura 2.2, conforme a elipse vermelha cresce), a sua incerteza volta a aumentar. Ao voltar a realizar novas observações dos obstáculos, a incerteza volta a diminuir.

Figura 2.2: Problema da Localização: Robô estimando sua localização



Fonte: Newman (2006)

Os sensores sofrem influência, na sua medição, de várias fontes de ruído. Um ponto importante é saber como lidar com as incertezas e imprecisões. As possíveis origens desses erros são provenientes do movimento do robô dentro do ambiente, dos parâmetros que fazem parte da modelagem cinemática de controle do robô (por exemplo, roda de tamanho diferentes, medição errada de eixo, etc.) e por fatores de natureza não sistemática, como as incertezas provenientes de situações inesperadas (por exemplo, eventuais colisões com objetos inesperados, escorregamento das rodas etc.). O maior desafio da técnica de localização consiste em corrigir as informações oriunda dos sensores (BORENSTEIN *et al.*, 1996).

Segundo Thrun (2005) e Borenstein *et al.* (1996), o problema de localização pode ser classificado em três tipos:

- O problema de Rastreamento da Posição (*Position Tracking*): A pose inicial do robô em relação ao sistema de coordenada é conhecida. Contudo, com a movimentação do robô, os erros dos sensores prejudicam a localização. O problema de rastreamento da pose visa a estimar esses erros, de modo a fornecer com menor erro a pose local do robô. Em geral, erros são aproximados por meio de uma distribuição unimodal (por exemplo, distribuição normal);

- O problema de Localização Global (*Global Localization*): O robô não conhece a sua pose inicial no mapa. Assim, são buscadas estratégias para se estimar essa pose, considerando inclusive, o problema de rastreamento da pose;

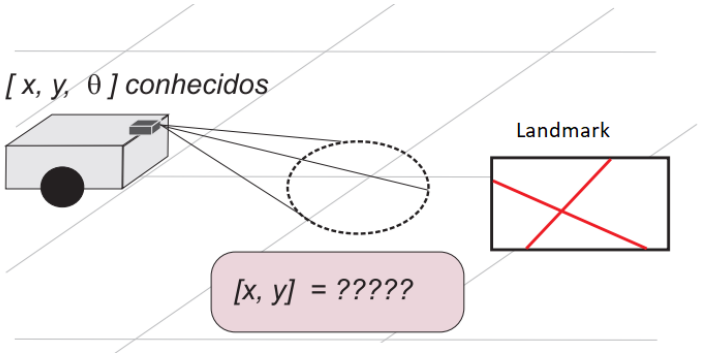
- O problema do Robô Raptado (*Kidnapped Robot Problem*): No problema do robô raptado, assume-se que a qualquer momento, o robô poderá ser transferido para outra pose sem ser informado. Nessa nova situação, o robô deverá detectar a alteração em sua pose e, logo em seguida, calcular a sua nova localização. Esse problema é interessante para verificar a capacidade de recuperação do robô em falhas repentinas, visando a manter o sistema de localização convergindo.

Existem diversas técnicas para estimar a localização do robô, a maioria com foco nos problemas de rastreamento da pose. Já o problema de localização global, métodos estatísticos têm sido usados com grande sucesso, conforme apontado por Thrun (2002).

2.2 Mapeamento

Técnicas de mapeamento de ambientes são usadas em várias áreas como geografia, geologia, construção civil, entre outras. O mapeamento é usado quando é necessário representar uma área previamente conhecida e geralmente é feito em uma escala diferente do ambiente representado, sendo usado dois tipos de mapeamento que são os topológicos e os métricos. Pode-se exemplificar o problema do mapeamento em robôs conforme a Figura 2.3 da seguinte forma: imagine um robô, que não possui o mapa local, sendo equipado com um sensor de visão. A partir de sua pose atual ([x, y, θ] conhecidos) ele realiza leituras com seus sensores e passará a identificar os obstáculos, onde descobre a distância que se encontra do objeto. Com essas informações ele dará início a construção do mapa e o processo se repete até ele ter mapeado todo o ambiente (SANTANA, 2011).

Figura 2.3: Processo de Mapeamento baseado em *landmarks*



Fonte: Santana (2011)

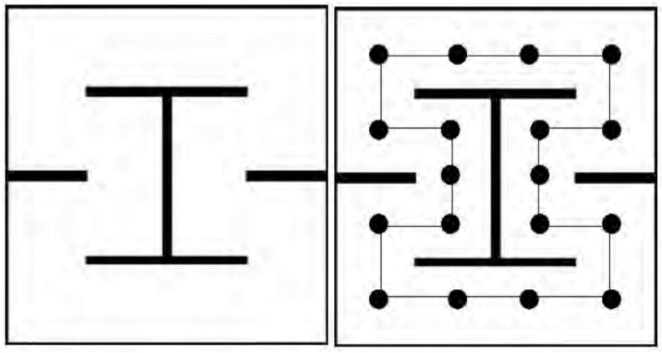
Pesquisas em mapeamento na robótica móvel iniciaram-se na década de 80, onde o mapeamento era amplamente dividido em dois tipos: topológico e métrico. Segundo Thrun *et al.* (2006), mapas topológicos são organizados na forma de grafo e utilizados para indicar possibilidades de navegação do robô por áreas contíguas do ambiente. Mapas métricos são usados em subáreas do ambiente e armazena informações mais precisas de seu contorno.

2.2.1 Mapas topológicos

Mapas topológicos, conhecido como mapas relacionais, fornecem as relações entre os vários locais ou objetos presentes no ambiente (marcos), geralmente sob a forma de grafos (MATARIC, 1990).

Na Figura 2.4, tem-se em (a) um ambiente ainda não mapeado. A Figura 2.4 mostra em (b) um mapa topológico representando o ambiente do robô através de grafos, onde os nós correspondem as situações, locais ou marcos distintos. Existindo um caminho direto entre dois nós, este serão interligados, indicando possibilidade de navegação do robô no ambiente.

Figura 2.4: Mapa topológico. (a) ambiente não mapeado, (b) ambiente mapeado.



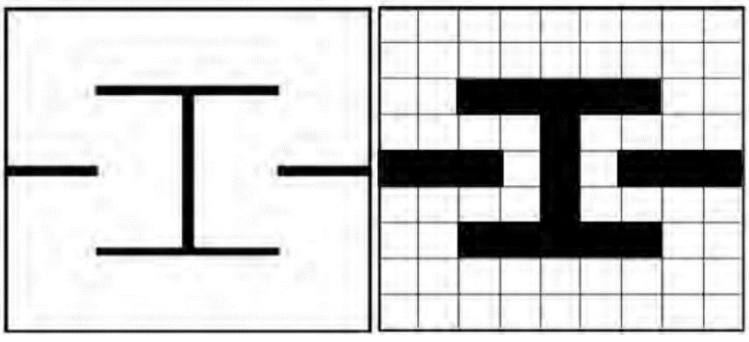
(a) (b)

Fonte: Oliveira (2010)

2.2.2 Mapas métricos

Os mapas métricos possuem várias formas de representação, sendo uma delas a divisão do ambiente em células de tamanhos iguais, também chamadas de grade (*grid*). Cada célula representa uma determinada parte fixa do espaço e armazena um valor que representa o estado desta célula como, por exemplo, a presença ou a ausência de um obstáculo (THRUN *et al.*, 1998). Essa forma é a mais conhecida e é chamada de grade de ocupação (*occupancy-grid)*. Nesta representação, tem-se uma matriz de ocupação multidimensional (normalmente 2D ou 3D) que mapeia o espaço em células, onde cada célula armazena uma estimativa probabilística de seu estado (ocupado ou livre) (MORAVEC e ELFES, 1985).

Figura 2.5: Mapa métrico 2D: (a) ambiente não mapeado, (b) ambiente em mapa métrico



(a) (b)

Fonte: Oliveira (2010)

Dependendo do tipo de representação do mapa, soluções que empregam mapas topológicos (CHOSET e NAGATANI, 2001; KUIPERS e BYAN, 1991), grade de ocupação (ELFES, 1989; THRUN *et al.*, 1998) e mapas geométricos (LEONARD e FEDER, 1991) prevaleceram. Foram propostas outras variantes como representação nebulosa do espaço (ORIOLO *et al.*, 1997), mapas auto-organizados (JANÉT, 1997) e componentes principais (KRÖSE, 2001; VLASSIS e KRÖSE, 1999). Porém, esses últimos modelos não tiveram receptividade por parte da comunidade cientifica.

2.2.3 Tipos de ambientes para criação dos mapas

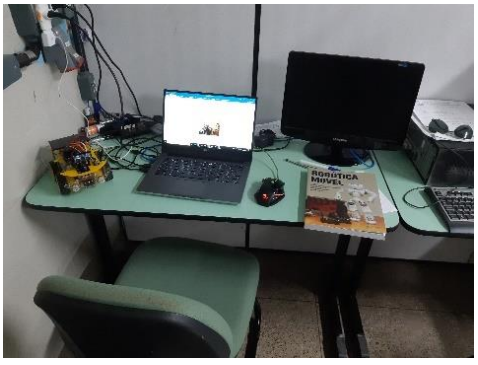
Os ambientes usados para criação dos mapas são classificados em: estático ou dinâmico, estruturado ou desestruturado e interno (*indoor*) ou externo (*outdoor*). Um ambiente é considerado dinâmico quando os objetos presentes na cena são móveis e estático quando todos os objetos são fixos. Na Figura 2.6, pode-se observar um local fabril onde existem diversos equipamentos que representam um ambiente interno estruturado e estático.

Figura 2.6: Ambiente interno estruturado e estático.

Fonte: Bigheti (2011)

A Figura 2.7 exibe um escritório onde as mesas, cadeiras, computadores e outros objetos colocados de modo desorganizados representam um ambiente interno desestruturado e estático.

Figura 2.7: Ambiente interno desestruturado e estático.



Fonte: Autor

A Figura 2.8 mostra uma imagem de um ambiente externo estruturado, com luminosidade não controlada e estático, com exceção de um veículo, das águas de um canal e das nuvens.

Figura 2.8: Ambiente externo estruturado e estático.

Fonte: Bigheti (2011)

É representado na Figura 2.9, um ambiente externo desestruturado, dinâmico com luminosidade não controlada e solo irregular.

Figura 2.9: Ambiente externo desestruturado e dinâmico.

Fonte: Bigheti (2011)

Nas aplicações reais, dificilmente são encontrados ambientes totalmente estáticos. Na prática, as implementações dos algoritmos consideram os objetos móveis como sendo ruídos do sistema. Assim, quando diante de um ambiente altamente dinâmico as técnicas tradicionais

não conseguem lidar de maneira eficiente com esse problema. Em seu trabalho, Bailey (2002) descreve sobre SLAM e a diferenciação entre objetos móveis e estáticos. Já a classificação entre ambiente estruturado e desestruturado depende das características geométricas dos objetos presentes na cena, sendo a reta a informação geométrica frequentemente extraída. Então, quando diante da dificuldade na identificação de formas geométricas conhecidas na cena, o ambiente é classificado como desestruturado. Em ambos os ambientes existem informações geométricas conhecidas. Entretanto, no ambiente interno, a geometria mais bem definida em relação ao ambiente externo. Por exemplo, janelas, portas e corredores são facilmente representados por retas. Assim, ter conhecimento do ambiente é importante na implementação dos algoritmos de mapeamento, uma vez que este processo influência diretamente nas características do tipo de informação que pode ser extraída dos sensores e no nível de ruído do sistema.

2.3 *Landmarks*

Landmarks são informações que estão presentes no ambiente de navegação do robô, servindo de referência na construção do mapa e são usados como dados no sistema de controle do robô, permitindo com isso a sua localização a partir dessas referências (BORENSTEIN *et al.*, 1996).

As informações extraídas do ambiente que podem ser caracterizadas como *landmarks* dependem diretamente do tipo de sensor usado para adquirir os dados e do algoritmo de processamento adotado para tratá-los. Independente de tais configurações, as informações adquiridas pelo robô devem fornecer confiabilidade. Os *landmarks* classificam-se em:

- Naturais;
- Artificiais.

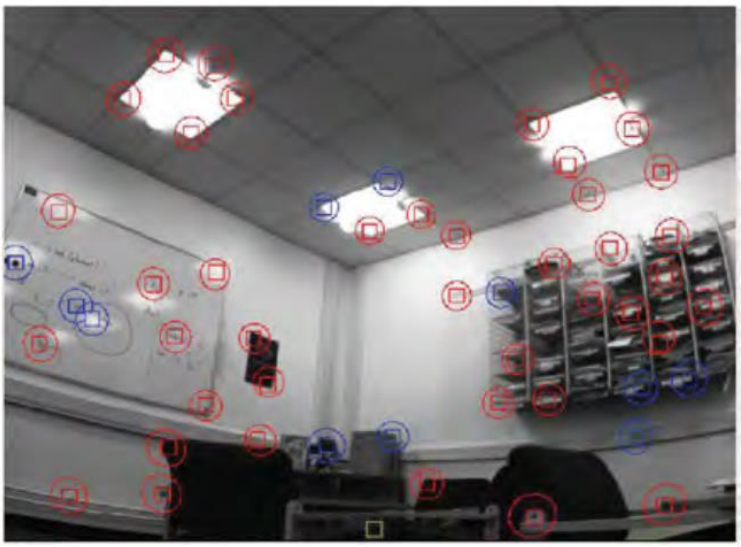
O *landmark* artificial consiste em inserir, intencionalmente, pontos de interesses no ambiente de exploração do robô. Esse método é detectado facilmente pelos sensores do robô, pois suas características foram artificialmente criadas já com esse intuito (LAJES, 1998). O *landmark* artificial serve tanto de referência para a localização e navegação do robô, bem como fonte de informação a ser usada para outro fim. A Figura 2.10 mostra um exemplo de *landmarks* artificiais fixados no piso do ambiente de trabalho do robô para detecção pelo sensor.

Figura 2.10: Exemplo de *landmark* artificial

Fonte: Autor

O *landmark* é considerado natural quando as informações já fazem parte do ambiente. A Figura 2.11 demonstra um exemplo de *landmarks* naturais que foram detectados através de sensor de imagem. Os *landmarks* naturais são provenientes da própria estrutura do ambiente como as bordas e cantos das portas e janelas, assim como podem ser oriundos da variação da intensidade de luminosidade de certa região de uma imagem.

Figura 2.11: *Landmarks* naturais detectados com sensor de imagem



Fonte: Bigheti (2011)

No processo de detecção de *landmark* no ambiente de exploração, são usadas basicamente, as seguintes fases:

1. Obter os dados do ambiente através do sensor;
2. Analisar os dados visando a identificar as informações que caracterizam um *landmark*;
3. Armazenar as características referentes aos *landmarks* na memória do sistema de controle do robô para gerar o mapa do ambiente. Esses *landmarks* também são usados na estimação da localização do robô.

A escolha do tipo de sensor usado na obtenção dos dados é baseada na análise do tipo de ambiente e o poder de processamento do sistema embarcado do robô. A análise do ambiente é importante porque os sensores possuem limitações e essas devem ser respeitadas na implementação de um sistema robusto.

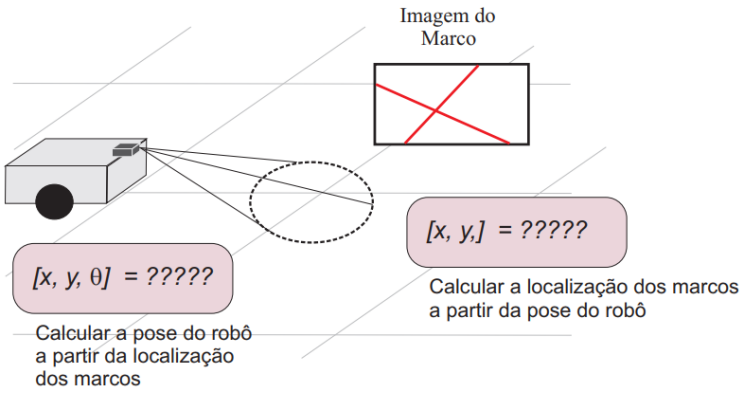
2.4 SLAM

Ao se imaginar um ser humano em um ambiente estranho, sem conhecer nada sobre o meio em que se encontra, ele tem a capacidade de integrar-se nesse ambiente apenas pela observação e exploração. Então, o que antes era considerado desconhecido, torna-se familiar. Nesse processo, o uso da memória é fundamental para que consiga distinguir entre locais já explorados de locais a serem explorados, permitindo que se navegue sem se perder. Outra situação bem interessante é o de um ser humano, abandonado numa floresta sem nenhuma informação. Com o passar do tempo, após ter explorado todos os locais a sua volta, ele passa a conhecer o ambiente onde se encontra, criando em sua mente pontos de referências que lhe permitem localizar-se em cada instante. Este processo de localização e mapeamento, facilmente realizado por um ser humano, vem sendo estudado ao longo dos anos em aplicações na robótica móvel, sendo conhecida com a sigla de SLAM **(**THRUN, 2004; BIGHETI, 2011).

A localização e o mapeamento simultâneos buscam solucionar o problema de um robô móvel que seja colocado em um local desconhecido pertencente a um ambiente desconhecido e ele consiga construir um consistente mapa, além de conseguir ao mesmo tempo determinar sua pose neste mapa. Esse processo ocorre de forma paralela, ou seja, à medida que o robô obtém informações para construir um mapa do ambiente, ele usa essas informações para saber em que ponto ele está inserido nesse ambiente (MONTEMERLO, 2003). Assim, o veículo deve

possuir atuadores e sensores para conseguir encontrar e identificar obstáculos, locomover-se e coletar as informações de odometria geradas e processar todos esses dados **(**BAILEY, 2002; DURRANT-WHYTE e BAILEY, 2006). A Figura 2.12 ilustra de forma simplificada o problema do SLAM.

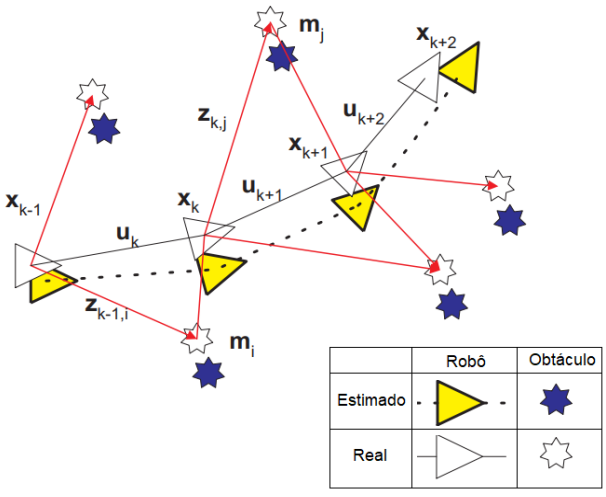
Figura 2.12: Robô realizando SLAM



Fonte: SANTANA (2011)

A Figura 2.13 mostra um robô realizando a tarefa de SLAM, onde *k* é o tempo discreto da estimação do estado do filtro, que considera as variáveis relacionadas a pose e ao mapa, $\boldsymbol{x_k}$ é a pose atual do robô, $\boldsymbol{u_k}$ é o vetor de controle aplicado em k, $\boldsymbol{m_j}$ é a posição atual do i-ésimo *landmark* e $\boldsymbol{z_k}$ é a distância entre o robô e o i-ésimo *landmark*. Quando o robô detecta os *landmarks* $\boldsymbol{m_{j-1}}$, sua localização $\boldsymbol{x_{k-1}}$ torna-se conhecida através do cálculo da distância $\boldsymbol{z_{k-1}}$ e essa informação é utilizada na formação do mapa até aquele ponto explorado. Ao receber um sinal $\boldsymbol{u_k}$ o robô executa o movimento e em seguida realiza a nova observação, obtendo o estado $\boldsymbol{x_k}$.

Figura 2.13: Processo de exploração de Ambiente



Fonte: Adaptado de Durrant-Whyte e Bailey (2006)

Após um longo período de navegação, as incertezas dos sensores produzem acúmulo de erros na estimação dos estados, tendo como consequência incoerência na localização e mapeamento. Diante disso, métodos de correção dos erros nas medições dos sensores são necessários para uma boa estimação dos estados do robô. Seria inviável a navegação em ambiente que não se tenha boa estimação da localização dos obstáculos neste meio. Um exemplo prático dessa problemática seria um robô de uso doméstico que acaba colidindo com os objetos do ambiente por não possuir medidas corretas.

2.5 Sensores em SLAM

Na robótica, os sensores desempenham papel fundamental no processo de percepção, interpretação do cenário e construção de mapa. É importante destacar que as informações fornecidas pelas fontes sensoriais possuem imprecisões. As primeiras tentativas de lidar com esse problema buscaram produzir sensores mais precisos. Entretanto, o aumento na precisão elevou o preço dos sensores (BOLTON, 2010).

Devido ao elevado preço dos sensores de grande precisão, atualmente, usam-se sensores de baixo custo, que, embora tenham precisão inferior, a baixa precisão é contornada com emprego de novas técnicas computacionais. Os sensores usados em SLAM são vistos na Figura 2.14. Um dos mais utilizados são os *scanners laser*, pois são precisos, eficientes e fornecem uma alta quantidade de dados por segundo para serem processados (alto custo computacional). As maiores problemáticas destes sensores são porque não possuem grande desempenho em superfícies formadas por vidro e são muito caros (WELLE *et al.*, 2010; BURGARD *et al.*, 2009; BLANCO *et al.*, 2008; ANOUSAKI e KOPOULOS, 2007).

Figura 2.14: Sensores usados em SLAM: a) *laser*, b) sonar, c) câmera

(a) (b) (c)

Fonte: Souza (2011)

Outro sensor muito utilizado é o sonar, devido ao seu baixo custo. Entretanto, este tipo de sensor não possui boa precisão na leitura e enfrenta grandes problemas de reflexão das ondas

sonoras (HUANG *et al.*, 2010; YAP e SHELTON, 2009; FENG *et al.*, 2008).

Existe também a opção de uso das câmeras como sensores. A maior característica deste sensor reside na riqueza de informações que ele é capaz de fornecer, qualitativa e quantitativa (WILLIAMS e REID, 2010; KOOTSTRA e SCHOMAKER, 2009; MILFORD e WYETH, 2008). O uso destes tipos de sensores não apresenta muita preocupação do ponto de vista da memória e do poder de processamento. Isto se deve pelo rápido aumento da capacidade de processamento dos computadores e disponibilidade de memória, permitindo que maiores quantidades de informações sejam tratadas.

Combinações ou fusão de sensores é muito comum na literatura. DIOSI *et al*. (2005) trabalharam com *laser* e sonar, AHN e LEE (2007) usaram um sonar combinado com uma câmera e FU *et al.* (2007b) usaram um *laser* com uma câmera na solução de problemas de SLAM.

Independente do sensor usado, as características obtidas do ambiente possuem ruídos. Devido a isso, a relação espacial entre as características detectadas e o robô são incertas porque os dados oriundos dos sensores são imprecisos. Por isso, soluções matemáticas devem ser capazes de representar, manipular e propagar a incerteza espacial (SKRZYPCZYNSKI, 2009).

Das várias abordagens possíveis, a abordagem probabilística é a mais adequada, porque sua fundamentação teórica fornece uma maneira de provar a convergência do filtro SLAM e de otimizar as soluções propostas. Isso ocorre porque os erros não podem ser previstos individualmente, mas, quando analisados em conjunto, podem ser tratados estatisticamente, apresentando padrões que podem ser modelados (THRUN *et al.*, 2005). Trata-se de um problema de minimização, com intuito de eliminar os erros aleatórios gerados pelos ruídos provenientes das fontes sensoriais. (ELFIES, 1987).

2.6 Filtros usados em SLAM

Os algoritmos utilizados na solução dos problemas em robótica móvel, incluindo o SLAM, de acordo com Thrun (2002), estão ligados ao filtro de Bayes (regra de Bayes). É um método probabilístico que usa evidências amostrais para calcular a probabilidade de eventos. Os algoritmos são:

- Filtro de Kalman-KF;
- Filtro de Kalman Estendido- EKF;
- EIF SLAM;
- SEIF;

- *FastSLAM*;

2.6.1 Filtro de Bayes

O filtro de Bayes integra dados temporais com intuito de calcular a probabilidade recursivamente. Assim, para o cálculo da predição, considera-se o resultado obtido no passo anterior, conforme mostram as Equações 2.1 e 2.2. A Equação 2.1 integra a probabilidade do passo anterior (fase de predição do filtro) e seu resultado é utilizado na Equação 2.2 para calcular a probabilidade predita (fase de atualização). Nessas equações, *t* é a variável discreta de controle temporal, onde somente o estado *t*-1 é considerado para a integração. Essa definição torna o algoritmo computável, assumindo que o sistema dinâmico é do tipo modelo Markov. O modelo Markov de primeira ordem assume que a informação amostral *y* no instante *t* depende somente da probabilidade de *x* nesse mesmo instante; tendo a probabilidade de *x* no instante *t* dependendo somente do instante *t*-1 (BIGHETI, 2011).

$$p^{-}(x^{t}) = \int p^{-}x^{t}|x^{t-1})p^{-}(x^{t-1})dx \tag{2.1}$$

$$p(x^{t}|y) = \eta p^{-}(y^{t}|(x^{t})\ p^{-}(x^{t}) \tag{2.2}$$

Em termos práticos, na fase de predição, a informação *y* representa a leitura do sensor e *x* representa um vetor contendo a pose do robô e os *landmarks* no ambiente, sendo os elementos desse vetor referenciados através de *x*. Nas equações do filtro de *Bayes*, é necessário especificar o modelo de percepção $p(y^{t}|x^{t})$, o controle aplicado ao movimento para a transição do estado t-1 para t do movimento, $p(x^{t}|x^{t-1})$ e a representação da probabilidade predita $p(x^{t}|y)$. A especificação pode ser feita de várias formas, dependendo do tipo de sensor, da mecânica do robô e do tipo de ambiente. Devido as diferentes formas de tratar os dados, diversos algoritmos são baseados no filtro de Bayes (BIGHETI, 2011). O filtro segue por três estágios:

- Predição;
- Observação;
- Atualização.

2.6.2 Filtro de Kalman

O filtro de Kalman (KF) é uma das variações mais usadas do filtro de *Bayes*, sendo utilizado em sistemas dinâmicos, normalmente perturbados por ruídos, para estimar variáveis do sistema utilizando a distribuição gaussiana. Seu uso para mapeamento na robótica foi introduzido por Smith (1990), para a construção de mapa contendo a estimativa de relacionamento espacial de objetos e suas incertezas. O filtro de Kalman é um método recursivo ótimo para solução de problemas lineares relacionados à filtragem de dados, onde o erro quadrático é minimizado (KALMAN, 1960). Através da variável de observação, outra variável, não observável, denominada "variável de estado ou processo" pode ser estimada, procedimento conhecido como inferência estatística. A partir desses cálculos recursivos estima-se o estado $x_v(k)$, que evolui de acordo com os modelos do processo e de observação.

A complexidade computacional para a execução do KF é $O(n^2)$, onde *n* representa o número de *landmarks*. A complexidade computacional quadrática permite que somente alguns *landmarks* possam ser utilizadas no mapeamento, enquanto que normalmente os mapas de ambientes naturais contêm milhões de *landmarks* (MONTERMELO *et al.*, 2002).

Os modelos de medição (sensores) e de movimento do robô utilizados no KF são funções lineares. No entanto, os dados gerados pelos sensores e o resultado dos comandos de movimento executados pelo robô nem sempre se comportam linearmente, podendo ocasionar erros no mapeamento, portanto o KF é limitado a sistemas lineares, sendo, ineficiente para sistemas não-lineares.

2.6.3 Filtro de Kalman estendido

Devido a limitação do filtro de Kalman a sistemas dinâmicos lineares, foi introduzida uma extensão neste, conhecida como EKF (*Extended Kalman Filter*), onde as funções lineares do KF passaram a se comportar de acordo com a dinâmica do robô e dos sensores. Com essas mudanças, os dados dos sensores e do sistema de controle são interpretados como lineares e um ruído gaussiano é inserido nesses dados. A ideia central do EKF é linearizar as funções via série de Taylor de primeira ordem no ponto de interesse (NEBOT, 2002).

2.6.4 FastSLAM

O *FastSLAM* é um algoritmo de mapeamento proposto por Montemerlo (2002), que tem como princípio básico o filtro de Kalman para estimar a localização dos *landmarks*, em um ambiente com distribuição gaussiana e o filtro de partículas para projetar a pose do robô no ambiente. Assim, o mapa gerado é composto por um conjunto de *landmarks*, cujas poses foram estimadas numa distribuição gaussiana. Uma condição importante deste algoritmo é o conhecimento a *priori* do caminho a ser explorado pelo robô, sendo tais informações fornecidas pelo vetor de controle e a correlação com o mapa feita pelo EKF.

Essa técnica apresenta avanço em relação a implementações feitas com o uso do KF, que possuem complexidade computacional quadrática em relação ao número de *landmarks*. Com isso, cada KF é utilizado somente para estimar a localização de cada *landmark* e a pose do robô, estimada pelo filtro de partículas. Nessa nova divisão, a complexidade computacional se reduz para uma proporção logarítmica definida na ordem $0(\mathrm{Mlog\ N})$, onde *M* representa a quantidade de partículas e *N* de *landmarks*. Os experimentos realizados pelos autores dessa técnica demonstraram que no mínimo 100 partículas são necessárias para obter bons resultados, tornando a complexidade computacional desta técnica inferior às implementações que utilizam o filtro de Kalman nas duas operações: estimação da pose do robô e dos *landmarks* (MONTEMERLO, 2003).

2.6.4.1 Filtro de partículas

O filtro de partículas (PF) é usado em localização nos algoritmos de SLAM, sendo baseado em amostras para redes Bayesianas dinâmicas, conhecidas em diversas áreas com o nome de Monte-Carlo (BAILEY, 2006). Esse filtro possibilita tratar qualquer tipo de distribuição de probabilidade, diferentemente do KF, em que a distribuição é gaussiana, sendo esse um dos principais motivos de seu uso em técnicas de mapeamento na área da robótica (GORDON, SALMOND e SMITH, 1993; DOUCET, FREITAS e GORDON, 2001; ARULAMPALAM *et al.*, 2002). Apesar dessa vantagem, o PF não é eficiente para tratar com a dimensionalidade dos mapas, já que o número de partículas necessárias para representar os estados no mapa cresce numa proporção que torna inviável o seu uso na maioria das situações reais de mapeamento. Com intuito de tornar o PF mais eficiente, foi proposto o filtro de partículas Rao-Blackwellised (RBPF), que é uma extensão do PF tradicional (conhecido como

versão 1.0) com capacidade de exploração de forma mais eficiente (BAILEY, 2006). É o mesmo FastSLAM, com a diferença importante na forma de amostragem das partículas (fase de predição), sendo chamado de FastSLAM versão 2.0.

Considerando o modelo de espaço de estado não linear descrito por:

$$X_k = f(X_{k-1}, V_{k-1}) \tag{2.3}$$

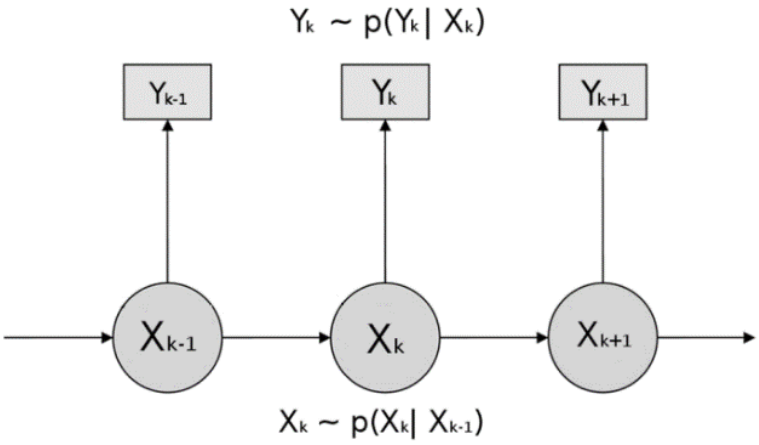
$$Y_k = g(X_k, U_k) \tag{2.4}$$

onde:

- X_k é a variável de estado no instante de tempo k;
- Y_k são as observações no instante de tempo k;
- As funções f e g são não lineares e geram X_k e Y_k;
- V_k é o ruído de movimento;
- U_k é o ruído da medição, que podem ser assumidos como variáveis aleatórias com distribuição normal e covariância Q_k e R_k.

Na Equação 2.3, a função densidade de probabilidade $p(X_k|X_{k-1})$ refere-se à transição de estado. Já a Equação 2.4 implica que a função densidade de probabilidade $p(Y_k| X_k)$ representa a observação, onde o ambiente é medido pelos sensores do robô. O problema consiste na estimação da variável de estado X_k, quando somente dados das observações estão disponíveis, por exemplo, $Y_{1:k} = \{Y_1, Y_2, ..., Y_k\}$. Assumindo que X_k segue um processo de Markov, a distribuição da predição pode ser definida como $p(X_k|Y_{1:k})$. A Figura 2.15 mostra graficamente a dependência entre os estados e as observações.

Figura 2.15: Dependência Markoviana entre os estados e as observações



Fonte: Lima (2011)

Assim, o problema de estimação pode ser formulado como a minimização do erro quadrático médio (EQM) estimada, definida por:

$$\hat{X}_k = E[X_k|Y_{1:k}] = \int X_k p(X_k|Y_{1:k}) dX_k \tag{2.5}$$

onde *E* representa o valor esperado.

O método de Monte Carlo utiliza, para representar a distribuição da predição, um conjunto de pontos aleatórios (amostras ou partículas). A partir da função densidade do estado inicial X_0 do sistema, juntamente com uma etapa de transição e uma de observação, obtém-se a função densidade do instante seguinte. Com isso, a função densidade predita em qualquer tempo discreto k, $p(X_k|Y_{1:k})$, pode ser avaliada de modo recursivo. No processo de transição, deseja-se propagar a próxima etapa de tempo *k*, por meio da densidade de transição dada pela Equação 2.6.

$$p(X_k|Y_{1:k-1}) = \int p(X_k|X_{k-1})\, p(X_{k-1}|Y_{1:k-1}) dX_{k-1} \tag{2.6}$$

Na etapa de atualização, que envolve a aplicação do teorema de Bayes (BAYES e PRICE, 1763), à medida que novos dados chegam, é calculada a função densidade de probabilidade, de acordo com a Equação 2.7.

$$p(X_k|Y_{1:k}) = \frac{p(Y_k|X_k)\, p(X_k|Y_{1:k-1})}{\int p(Y_k|X_k)\, p(X_k|Y_{1:k-1}) dX_k} \tag{2.7}$$

As duas etapas fornecem a solução ótima para o problema de estimação, tendo o inconveniente da dificuldade de resolver analiticamente a integração multidimensional. Um método alternativo é obtido através do método Sequencial de Monte Carlo (GORDON *et al.*, 1993; DOUCET *et al.*, 2001; ARULAMPALAM *et al.*, 2002; CAPPE *et al.*, 2007). Esse método afirma que: "A distribuição de probabilidade *p*, da Equação 2.7, pode ser extremamente complexa. Entretanto, pode-se amostrar a partir de uma outra distribuição *q* (distribuição de importância), onde sorteia-se *N* amostras aleatórias de X_k, $k = 1 \ldots N$ a partir de *q* ao invés de *p*. Entretanto, para garantir que o estimador da esperança da Equação 2.5 não seja viciado é necessário fazer uma correção. Este ajuste consiste em definir um peso positivo para cada um dos pontos aleatórios. Tem-se que o valor requerido para o peso é proporcional a relação $r = p/q$

avaliada no ponto aleatório, onde r é chamado de *função importância"*. Logo, a esperança da Equação 2.5 pode ser estimada como a média ponderada.

$$\hat{X}_k = E[X_k|Y_{1:k}] = \int X_k \frac{q(X_k|Y_{1:k})p(X_k|Y_{1:k})}{q(X_k|Y_{1:k})} dX_k$$

$$= \int X_k r(X_k|Y_{1:k}) q(X_k|Y_{1:k}) dX_k \approx \sum_{k=1}^{N} \frac{W_k}{\sum_{j=1}^{N} W_j} X_k \quad (2.8)$$

Onde W_k é peso da partícula k e W_j é o peso total das N partículas. Desse modo, o resultado da integral através de um conjunto de amostras pode ser calculado como:

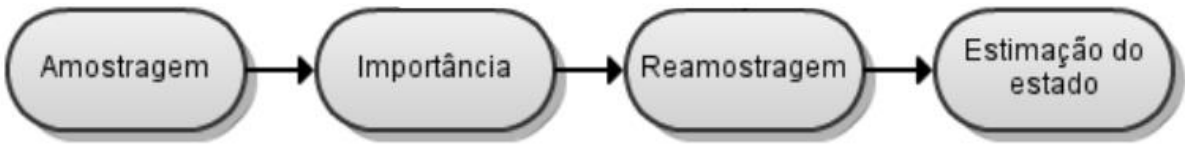
$$\hat{X}_k = \sum_{i=1}^{N} X_k^{(i)} \widehat{W}_k^{(i)} \quad (2.9)$$

onde $\hat{W}_k^{(i)}$ é o peso normalizado (variando entre 0 e 1) da amostra i no instante k. O peso de cada amostra representa a probabilidade do estado real estar no estado representado por essa amostra.

Visando a aproximação da função de densidade predita, usa-se o algoritmo Amostrando e Reamostrando pela Importância (SIR). Nesse algoritmo, as partículas que contribuíram pouco para estimar a pose do robô são reamostradas para que estas obtenham uma nova importância.

O algoritmo SIR consiste em três etapas, como mostrado na Figura 2.16. Na etapa de amostragem, as partículas são distribuídas através de uma função densidade de probabilidade. Em seguida, são obtidas as partículas com maior importância no processo e as partículas de menor importância são reamostradas, para que, finalmente, seja estimado o estado do robô.

Figura 2.16: Estrutura do funcionamento do filtro de partículas

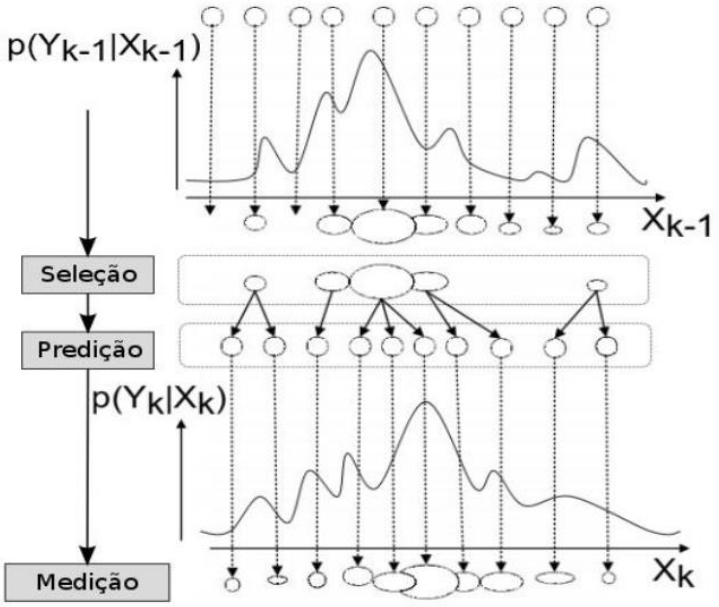


Fonte: Adaptado de Lima (2011)

A Figura 2.17 mostra as partículas representadas por círculos e seus pesos são os tamanhos dos círculos. Os círculos de tamanho maior representam os pesos de maior importância, de forma análoga, os círculos menores representam pesos de menor importância. A importância de cada partícula é usada para incorporar as medições do atual estado do filtro nas partículas, calculando a importância de cada uma dentro do conjunto (THRUN *et al*., 2005).

Na etapa de seleção, as partículas mais promissoras são selecionadas e suas importâncias são atribuídas para as partículas menos promissoras na etapa de predição (processo de reamostragem). Na etapa de medição, é feita a estimação tomando como base as partículas e sua ponderação, ou seja, cada medida bruta armazenada pela partícula é ponderada.

Figura 2.17: Ilustração do algoritmo FP



Fonte: Adaptado de Lima (2011)

2.6.4.2 O problema da degeneração

Para diminuir a degeneração de um conjunto de partículas, deve-se minimizar a variância dos pesos. Assim, para minimizar essa variância é preciso (HUANG *et al.*, 2010):

- Para partículas com peso relativamente grande, diminuir seus pesos.
- Para partículas com peso relativamente pequeno, aumentar seus pesos.
- Manter a ordem dos pesos das partículas, ou seja, os pesos maiores continuam maiores e os pesos menores continuam menores.

Um modo de minimizar a variância e a degeneração é tentar manter sempre um conjunto de partículas com a maior quantidade de informações possível, ou seja, maximizar o peso absoluto das partículas. Assim, todas as partículas tendem a um valor limite superior e a variância tende a zero ao normalizar esses pesos. No entanto, deve-se tomar o cuidado de não impedir a exploração adequada do espaço de busca, para evitar cair em um máximo local. Assim, a reamostragem busca evitar a degeneração substituindo as partículas com peso pequeno por novas partículas com pesos relativamente maiores. Apesar desse método reduzir o problema, a degeneração continua existindo, uma vez que a nova amostra pode continuar tendo peso relativamente pequeno.

Um modo de medir a degeneração é através do tamanho efetivo da amostra (N_{eff})(KONG; LIU; WONG, 1994; AIUBE, 2005), que é calculado pela Equação 2.10.

$$N_{eff} = \frac{N}{1 + Var^{\pi(*|Y_{0:k})}(W * (x_{0:k})} \tag{2.10}$$

onde π é a função densidade de probabilidade, usada na tentativa de aproximar a função densidade real do problema.

Por ser difícil calcular a variância da Equação 2.8, uma aproximação de N_{eff} torna-se mais interessante para essa medição. Nessa aproximação, é possível avaliar o grau de degeneração do algoritmo através da Equação 2.11 abaixo.

$$\widehat{N}_{eff} = \frac{1}{\sum_i^N (W^i)^2} \tag{2.11}$$

onde $\widehat{N}_{eff}$ rpresenta o tamanho amostral efetivo e e W representa o peso das partículas.

O tamanho efetivo da amostra aumenta quando a variância do valor dos pesos é reduzida, ou seja, quando a degeneração apresentada pelas partículas é reduzida. Em outras palavras, o tamanho efetivo da amostra diminui quando a degeneração aumenta.

O processo de reamostragem pode ser realizada quando $\widehat{N}_{eff}$ for menor que uma constante N_{eff}, onde essa constante indicaria a ocorrência de degeneração. Em resumo, a ideia da reamostragem consiste em eliminar as amostras que possuem um peso muito pequeno, mantendo apenas as amostras que tem um maior peso (ARULAMPALAM, 2002).

2.6.5 Filtro "Extended information form SLAM"

Extended Information Form SLAM Algorithm (EIF SLAM) é um algoritmo de mapeamento baseado no filtro de Informações Estendidas. O EIF SLAM é considerado um método *offline* porque constrói o mapa do ambiente somente após a sua exploração. Durante a etapa de exploração, as informações provenientes do movimento planejado e dos sensores são armazenadas na memória do sistema controlador do robô. É indicado para situações em que o conjunto de dados possui tamanho fixo e a memória do sistema é capaz de suportar todos esses dados no instante em que o mapa está sendo construído. Os dados no mapa são representados pelos dois primeiros momentos da gaussiana, semelhante ao EKF, chamado nessa abordagem por vetor e matriz de informação. Entretanto, o método de atualização das informações é diferente entre os dois algoritmos. A grande vantagem desse método é a sua capacidade de gerar mapas precisos e de grandes dimensões. Na construção do mapa, os dados são formados por um conjunto de medidas $z_{1:t}$ relacionadas à variáveis de correspondências que indicam a localização dos *landmarks* $c_{1:t}$ e por um conjunto de controle $u_{1:t}$. Com base nesses dados, o algoritmo inicializa um vetor que contém todas as posições que o robô possuiu durante a coleta dos dados. Assim, passos conhecidos como construção, redução e resolução são repetidos enquanto um limite desejado de precisão não for alcançado (THRUN e BURGARD, 2005).

2.6.6 Filtro "Sparse extended information form SLAM"

Uma versão recente do EIF SLAM é o *Sparse Extended Information Form Algorithm* (SEIF). Nessa metodologia, o algoritmo realiza o mapeamento *online*, ou seja, durante o processo de exploração do ambiente. O SEIF difere do EIF SLAM porque mantem uma matriz esparsa de informação, cujo objetivo possui caráter computacional. Assim, as equações de atualização do SEIF são executadas com tempo constante (THRUN *et al.*, 2004).

2.7 Redes neurais artificiais

Há duas abordagens na solução de problemas que envolvem aplicações computacionais, algorítmica e a conexionista. Na primeira, é executado uma sequência finita de passos que apresentam resultados coerentes a partir de um conjunto de regras pré-definidas. Mas, tal abordagem apresenta dificuldades de fornecer características do aprendizado quando comparado a técnica conexionista, entre elas as Redes Neurais Artificiais.

As redes neurais artificiais (RNAs) são sistemas de computação baseados nas características do sistema nervoso e nos neurônios do ser humano. De acordo (HAYKIN,1999), uma RNA assemelha-se ao cérebro humano em dois aspectos:

- Conhecimento é adquirido pela rede através de um processo de aprendizagem;
- Forças de interconexão entre os neurônios, conhecidas como pesos sinápticos ou pesos, são utilizadas para armazenar o conhecimento.

As RNAs são formadas por um conjunto de neurônios que interagem entre si baseados nas características de processamento de informação e nas características de suas interconexões semelhantes aos neurônios reais. São sistemas paralelos de computação com possibilidade de implementação em *hardware* ou em *software*.

As redes neurais possuem capacidade de coletar, utilizar e armazenar informações baseadas em experimentos (aprendizagem). O processo de aprendizagem é feito a partir de algoritmos de aprendizagem, onde os pesos sinápticos da rede são alterados de forma ordenada visando a alcançar o resultado desejado.

A partir da informação vinda de um banco de dados (pares de entradas e saídas), as redes neurais devem "aprender" e generalizar o conhecimento contido nas amostras, de forma a conseguir responder a qualquer outra entrada não pertencente ao conjunto de treinamento (HAYKIN, 1999). Algumas características importantes de uma rede neural artificial são:

- Robustez e tolerância a falhas: a retirada de alguns neurônios não afeta substancialmente o seu desempenho global;

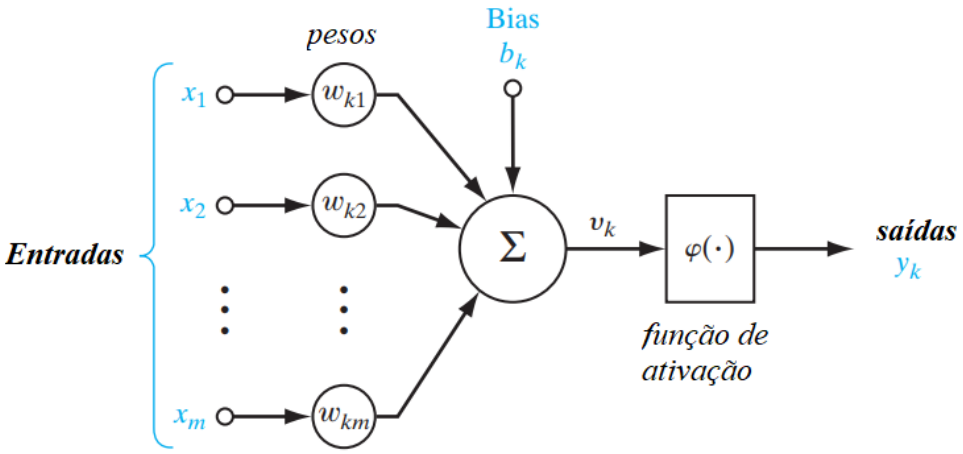
- Flexibilidade: através do processo de aprendizagem pode ser ajustado novos pesos, conseguindo aprender novas ações com base na informação contida nos dados de treinamento;

- Processamento de informação incerta: mesmo que a informação fornecida esteja incompleta, afetada por ruído, ainda consegue-se obter um raciocínio correto;

- Paralelismo: uma grande quantidade de neurônios está ativa ao mesmo tempo. Não existe a restrição de um processador que obrigatoriamente trabalhe uma instrução após outra.

O modelo artificial de neurônio é mostrado na Figura 2.18, sendo uma generalização do modelo de Mcculloch e Pitts (1943).

Figura 2.18: Modelo de um Neurônio artificial



Fonte: Autor

Internamente, existe um módulo que combina as entradas, visando a coletar e combinar as informações vindas de outros neurônios. Em geral, essa combinação é um somatório entre as entradas ponderadas do neurônio, cujo resultado é conhecido como entrada líquida do neurônio. Em seguida, uma função matemática é aplicada, chamada função de ativação. O módulo de combinação das entradas e a função de ativação formam o corpo do neurônio responsável por apresentar a saída do neurônio a partir dos valores dos vetores de pesos e de entrada. Esse modelo possui um sinal adicional ***bias (b)*** que favorece ou limita a possibilidade de ativação do neurônio.

O processo sináptico é representado pelos pesos (*w*) que amplificam cada um dos sinais recebidos, fornecendo a rede a capacidade de acumular conhecimento à medida que são ajustados. A função de ativação (*f*) defini o modo como o neurônio responde ao nível de excitação, limitando e definindo a saída da rede neural. A função de ativação pode ter diferentes formulações, sendo as mais usadas: linear, sigmoide logarítma, tangente

hiperbólica e gaussiana.

Em termos matemáticos, pode-se definir o neurônio da Figura 2.18 pelo par de equações:

$$v_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.12}$$

e

$$y_k = \varphi(v_k + b_k) \tag{2.13}$$

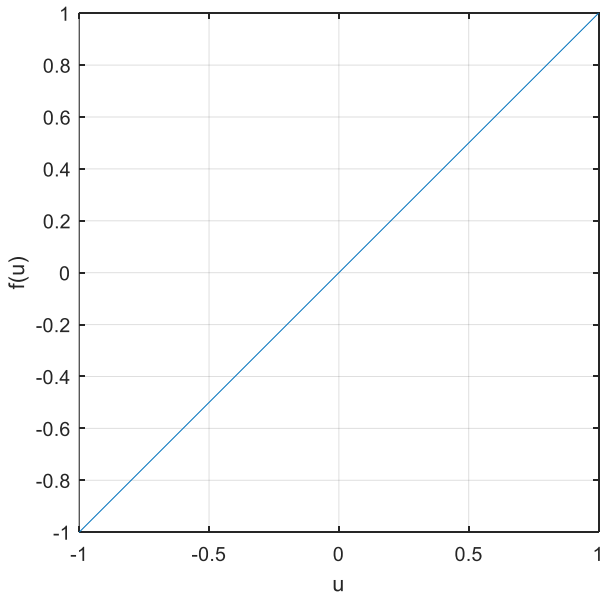
onde $x_1, x_2, \ldots, x_m$ são os sinais de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ são, respectivamente, os pesos das entradas do neurônio; u_k é a saída da combinação linear dos sinais de entrada; b_k é a bias; $\varphi(.)$ é a função de ativação; e y_k é o sinal de saída do neurônio.

As funções de ativação mais comuns possuem as seguintes formulações matemática e o seu respectivos gráficos:

- Função Linear

$$f(u) = u \tag{2.14}$$

Figura: 2.19: Gráfico da função linear

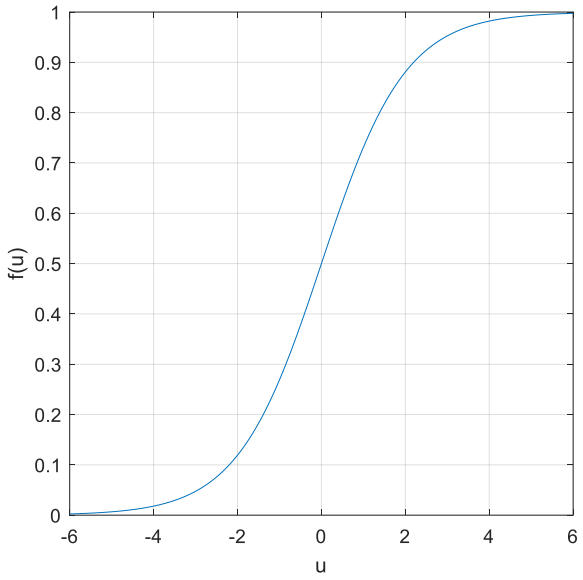


Fonte: Autor

- Função sigmoide logarítmica

$$f(u) = \frac{1}{1 + e^{-\beta u}} \tag{2.15}$$

Figura: 2.20: Gráfico da função sigmoide logarítmica

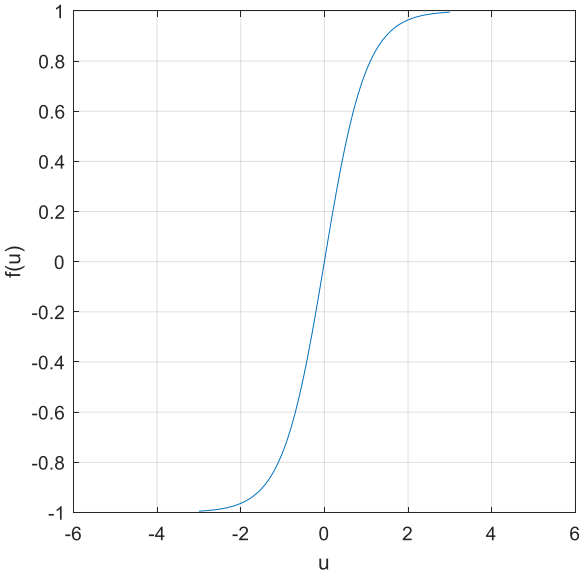


Fonte: Autor

- Função tangente hiperbólica

$$f(u) = \frac{1 - e^{-\beta u}}{1 + e^{-\beta u}} \tag{2.16}$$

Figura: 2.21: Gráfico da função tangente hiperbólica

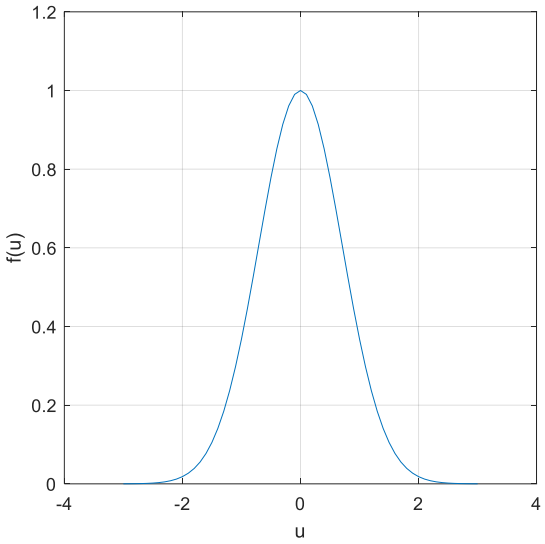


Fonte: Autor

- Função gaussiana

$$f(u) = e^{-\frac{(u-c)^2}{2\sigma}} \tag{2.17}$$

Figura: 2.22: Gráfico da função gaussiana



Fonte: Autor

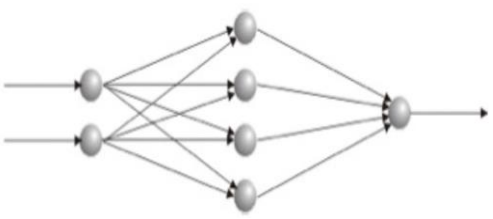
2.7.1 Estrutura das redes neurais artificiais

As disposições (arquitetura) de uma rede neural são as diversas formas de organização dos componentes da RNA, dependendo da necessidade da aplicação, estando diretamente ligadas com o algoritmo de aprendizagem usado para treinar a rede. Basicamente, os itens que compõem a estrutura de uma rede neural são:

- Unidades processadoras ou neurônios artificiais;
- Conexões entre os neurônios artificiais.

Em uma rede neural, os neurônios estão organizados na forma de camadas (conforme Figura 2.23).

Figura 2.23: RNA organizadas em camadas



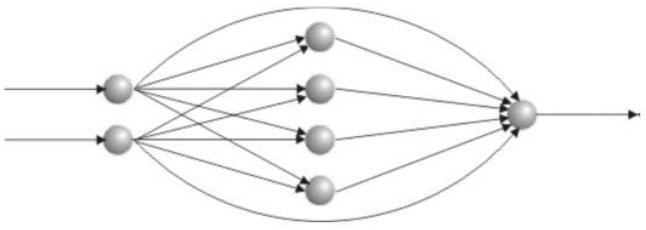
Fonte: de Almeida Neto (2003)

A primeira coluna representa os neurônios da camada de entrada, cuja função é receber os dados para serem processados pela RNA nas camadas seguintes. O elemento central é conhecido como camada escondida, oculta ou intermediária, nomenclatura que varia entre alguns autores. É a camada responsável por realizar o primeiro processamento. A última coluna representa a camada de saída, cuja função é realizar o processamento final e apresentar o resultado ao meio externo. Assim como na camada de entrada, é obrigatório que exista um neurônio para cada saída da rede.

A quantidade de camadas e neurônios está relacionado ao problema em que a rede será aplicada. Assim, a definição correta dessas informações é fundamental para a modelagem da arquitetura da RNA. Segundo Carvalho e Ludermir (2007), neurônios individuais são sistemas com capacidade computacional limitada. Entretanto, um conjunto de neurônios artificiais conectados na forma de uma rede é capaz de solucionar problemas complexos. O modo como as conexões podem estar arranjadas dá-se o nome de topologia e as RNAs podem ter dois tipos: *feedforward* e *feedback*.

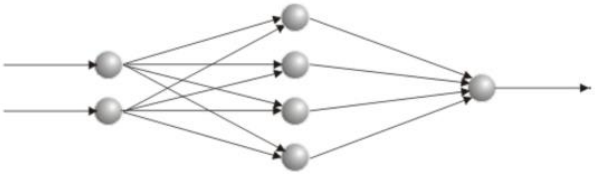
Na topologia *feedforward*, o fluxo de dados ocorre em apenas um sentido, partindo de um neurônio e podendo ir para qualquer neurônio de qualquer camada a frente. Uma variação da topologia *feedforward* é a estritamente *feedforward*, que também apresenta fluxo de dados em apenas um sentido, porém o neurônio de uma camada só pode estar conectado a qualquer neurônio da camada seguinte. Na topologia *Feedback* o fluxo de dados pode ocorrer em ambos os sentidos. As Figuras 2.24 a 2.26 mostram as topologias citadas:

Figura 2.24: Topologia *feedforward*



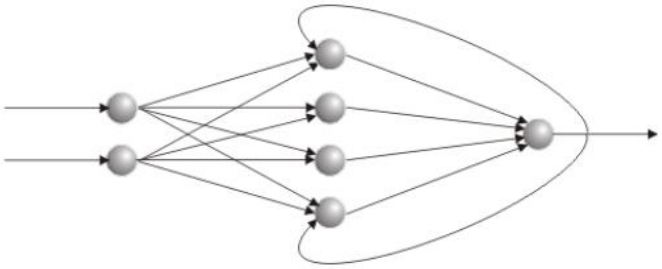
Fonte: de Almeida Neto (2003)

Figuras 2.25: Topologia estritamente *feedforward*



Fonte: de Almeida Neto (2003)

Figuras 2.26: Topologia *feedback*



Fonte: de Almeida Neto (2003)

2.7.2 Tipos de RNA

Existem diversos tipos de RNAs, cada um com suas características e aplicações. A redes neurais MLP são um tipo de rede bastante usada para o desenvolvimento de atividades científicas, sendo esta rede a utilizada neste trabalho. É derivada da rede Perceptron de única camada e sua estrutura é composta de no mínimo três camadas, uma de entrada, uma intermediária e uma de saída.

As quantidades de camadas escondidas e de neurônios na rede MLP não são calculadas e sim estimadas e ajustadas, baseado na experiência do projetista da rede. Entretanto, deve-se tomar cuidado, pois um número excessivo de camadas intermediárias pode comprometer a convergência global, haja vista que os algoritmos de treinamento utilizados se baseiam no cálculo do erro da rede para corrigir os pesos.

A rede *Radial Basis Function* (RBF) tem uma estrutura semelhante à MLP, porém possui apenas uma camada intermediária. Ela recebe este nome por utilizar função de ativação de base radial na camada intermediária, um exemplo comum é a função gaussiana. Esse tipo de rede é utilizado como aproximador de funções e em problemas de classificação.

A rede de Hopfield é um tipo de rede recorrente que possui organização auto associativa, onde todos os neurônios estão no mesmo nível e não existe divisão por camadas. Sua topologia possui características das topologias *feedforward* e *feedback*. Assim, é um tipo de rede muito utilizado em problemas que envolvem otimização.

A rede de Kohonen é uma RNA de aprendizado não supervisionado. Ela faz parte de um grupo de algoritmos chamado Self Organizing Maps (SOM) que tem como função principal categorizar os padrões de entrada inseridos na rede. Por isso, esse tipo de rede é indicado para problemas de reconhecimento de padrão e classificação.

2.7.3 Aprendizado

O aprendizado consiste em treinar a rede com um conjunto de amostras de pares de entrada e saída, fazendo com que ela seja capaz de reagir corretamente na sua saída quando um outro valor que não faz parte do treinamento for colocado na sua entrada. Dentro dessa abordagem, o aprendizado pode ser na forma supervisionada, não supervisionada e semi-supervisionada (HAYKIN, 1999).

No aprendizado supervisionado, um elemento externo verifica o quanto a rede está próxima de uma solução aceitável (erro), realizando alterações no treinamento dos pesos entre os neurônios, tentando alcançar a menor diferença entre as saídas desejadas.

No aprendizado não supervisionado, inicialmente, as saídas da rede não são conhecidas, funcionando de modo a distinguir classes de padrões diferentes dos dados apresentados à rede, através de algoritmos de aprendizado baseados geralmente em conceitos de vizinhança e agrupamento. O ajuste da rede é feito de acordo com regularidades estatísticas dos dados de entrada, de tal forma que ela cria categorias, otimizando em relação aos parâmetros livres da rede uma medida da quantidade que é independente da tarefa a ser executada.

O aprendizado semi-supervisionado teve seu desenvolvimento recente, por isso é menos comum. Seu principal uso é para o treinamento de classificadores quando uma grande quantidade de exemplos não rotulados está disponível junto a um pequeno conjunto de exemplos rotulados. Em outras palavras, o principal benefício deste tipo de aprendizado é o melhor aproveitamento da base de dados.

Na implementação desses tipos de aprendizado, escolhe-se um algoritmo de treinamento, o qual dispõe das regras de ajuste dos pesos. Existem vários algoritmos disponíveis para cada tipo de RNA, porém novos modelos ou variações vêm sendo apresentados constantemente pela comunidade científica. Neste trabalho, será utilizado o algoritmo Backpropagation para o treinamento de redes MLP.

O algoritmo Backpropagation ficou bastante conhecido a partir de 1986 e foi uma solução para o problema de treinamento de redes neurais de várias camadas. Pois até então, não existia nenhum algoritmo capaz de atualizar todos os pesos da rede baseado no erro de saída. Este algoritmo implementa o aprendizado do tipo supervisionado e seu principal objetivo é minimizar o erro quadrático médio de saída da RNA. Para tanto, ele utiliza o método do gradiente descendente.

O funcionamento do BP ocorre em duas fases: *forward* e *backward*. Na primeira fase não há modificação dos valores dos pesos, os sinais de entrada propagam-se até a camada de

saída com o objetivo de calcular o erro da rede. Esta fase inicia com a alimentação dos neurônios da camada de entrada, que simplesmente recebe os padrões de entrada para processamento pela rede. O primeiro processamento de fato ocorre na primeira camada escondida. Por esse motivo, alguns autores não consideram a camada de entrada como uma camada de neurônios

3 Metodologia proposta

Neste capítulo, aborda-se uma nova estratégia, baseada em técnicas de inteligência artificial, para estimação da pose de robôs. A problemática será discutida, com abordagem matemática, bem como a solução proposta e implementada em simulação. A metodologia adotada visa a melhorar as aplicações de uso de sensores de baixo custo quando empregados em robôs que realizam SLAM.

3.1 Estratégia de ponderação das partículas

Os grandes problemas do filtro de partículas são: o processo de distribuição das partículas, quantidade de partículas e, em especial, o processo de ponderação (peso) das partículas.

A ponderação é um fator que deve ser considerado no filtro porque ela informa o nível de importância que determinada partícula possui dentro do processo de estimação da pose do robô. Buonocore *et al* (2017) propôs que a ponderação das partículas seja baseada em uma matemática simples, tendo como dados a distribuição das partículas e a medição do sensor. Nesse processo, os dados do sensor são as medidas brutas e, assim, foram usados como métodos de correspondência com os dados brutos adquiridos pelo sensor e mantidos pelas partículas. Inicialmente, o conjunto de partículas é gerada (distribuição das partículas) usando uma função gaussiana. No processo de correspondência, é verificado se as medidas brutas adquiridas no estado atual, obtidas pelo sensor, já estão armazenadas em cada uma das partículas geradas. Se tal condição for verdadeira, o peso das partículas aumenta. Caso contrário, o peso não é alterado e as medidas (que não foram verificadas) serão incluídas nas medidas brutas da avaliação da partícula.

Considere, por exemplo, que cada uma das partículas $i = \{1, ..., N\}$ geradas em um estado específico s pode ser definida como:

$$p_i(s) = \{ \alpha_s, \omega_s, \gamma_s \} \; \forall s = \{1, \dots, N\} \tag{3.1}$$

onde α corresponde ao vetor de pose definido por $[x_{1:s} \;\; y_{1:s} \;\; \theta_{1:s}]^T$, ω é o vetor usado para armazenar as medidas brutas obtidas do ambiente e γ refere-se à grade de ocupação local (mapa métrico). O subíndice (1: s) do vetor de pose corresponde ao conjunto de medidas de todos os estados observados.

A função do processo de correspondência é verificar se existe uma medida bruta armazenada na partícula que possa ser considerada como válida em relação a medida sendo adquirida pelo sensor. A verificação dessas medidas (para cada partícula *i*) é feita baseada em uma processo simples com formulação descrita na Equação 3.2. Esse método emprega uma geometria de centros e círculos. O centros são definidos pelas medidas brutas e os círculos são definidos por um raio.

$$\vartheta_j = \begin{cases} 1 \; se \; m(.) \in f(c,r) \\ 0 \; se \; m(.) \notin f(c,r) \end{cases} \tag{3.2}$$

Na Equação 3.2, tem-se que j = 0,1,2, ..., N_e, sendo Ne o número total de elementos do vetor ω (Equação 3.1), *m* refere-se a função que verifica se a medida adquirida pelo sensor do robô encontra-se já no conjunto de dados brutos (centro c) da partícula em uma distância não maior que o raio (r), caso em que retorna o valor 1. Caso contrário, retorna o valor 0. O valor do raio é definido usando uma boa precisão das medidas do *laser scanner* (sensor do robô) e os dados do filtro. O valor de *r* pode ser escolhido para obter a melhor relação entre o tempo de processamento e a precisão dos contornos do mapa.

Então, se ϑ = 1, a medida adquirida pelo sensor está dentro do círculo e a correspondência ocorre (*match*). Por outro lado, se ϑ = 0, a medida está fora do círculo e a correspondência não ocorre. Assim, as medidas que recebem ϑ = 0 são inseridas no vetor (ω) das medidas brutas da partícula avaliada. Consequentemente, o vetor (ω) cresce com o tempo de exploração e com o uso de pequenas valores de correspondência de raio. A quantidade de correspondência que ocorreu em uma partícula específica *i* pode ser calculada da seguinte maneira:

$$\eta_i = \sum_{j=1}^{N_e} \vartheta_j \tag{3.3}$$

onde η_i representa a quantidade de correspondências que ocorreram na partícula *i*. Assim, o peso de cada partícula pode ser calculado através da relação entre as quantidades de correspondências ocorridas na partícula *i* e as quantidades totais de correspondências de todas as partículas do filtro, sendo esse processo definido pela seguinte equação:

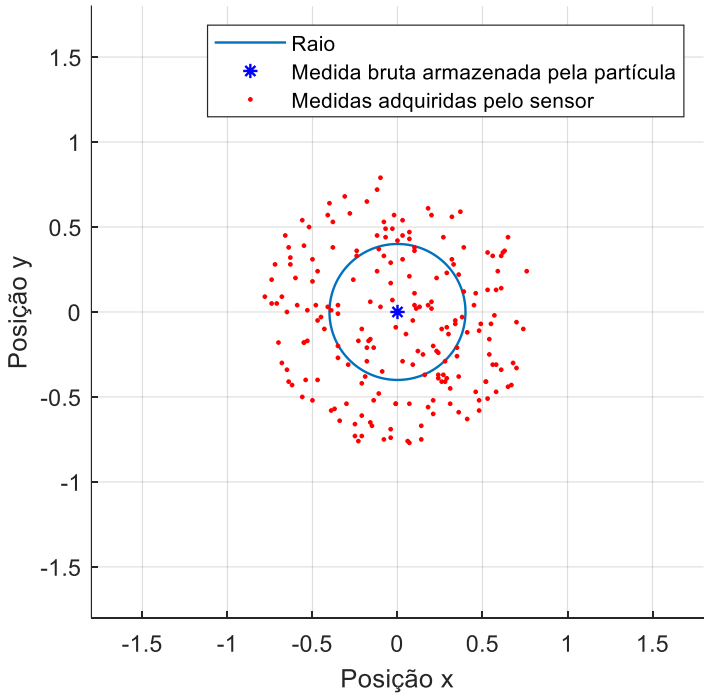
$$w_i = \frac{\eta_i}{\eta} \tag{3.4}$$

Nesta equação, w_i é o peso da partícula i, η_i é o número total de correspondências obtidas pelas partículas do filtro e η é o número total de correspondência considerando-se todas as partículas. Após este processo de ponderação, o mapa de ocupação local das partículas é atualizado (SLAM) e, em seguida, as hipóteses promissoras (procedimento de reescalonamento) são usadas por um novo conjunto de partículas (que são geradas nos próximos estados). O método descrito acima é mais prático e computacionalmente mais simples quando comparado as equações do filtro de partícula explicadas no capítulo anterior.

3.2 Problemáticas existentes na estratégia de ponderação das partículas

A estratégia apresentada necessita que o raio do círculo seja adequado à distribuição das partículas, tendo os dados do sensor como sendo o centro do círculo, conforme explicado anteriormente. A escolha do raio fica condicionada a uma análise empírica, onde, após análise na simulação, adota-se um único valor de raio. Este método pode ser demonstrado na Figura 3.1.

Figura 3.1: Visualização do método de centros e raios

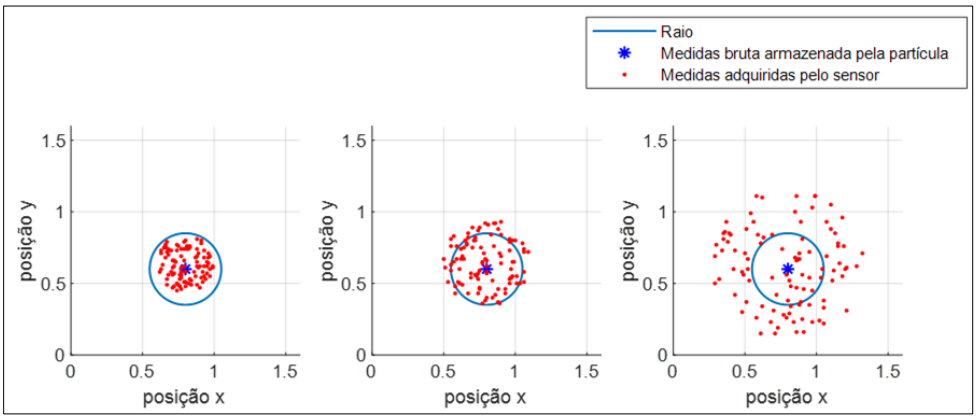


Fonte: Autor

O raio de valor fixo consiste em usar um círculo único para todo processo de movimento do robô no ambiente, enquanto que um raio variável gera círculos diferenciados em função desses valores em cada processo. Considerando-se o uso de um raio fixo e se este for pequeno, irá produzir erros pequenos, já um raio grande irá aumentar estes erros. Em resumo, em cada processo, ocorreram medidas espalhadas, podendo ser muitas delas dentro ou fora do círculo, de acordo com o raio escolhido.

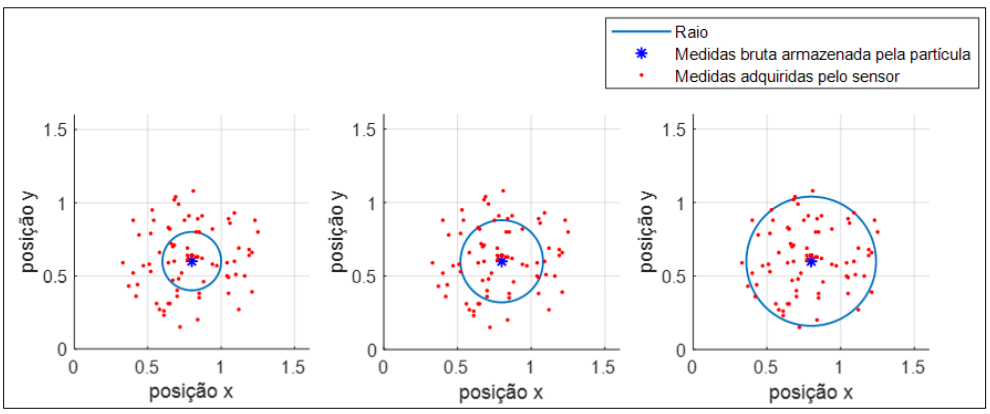
As Figuras 3.2 e 3.3 demonstram as situações acima no processo do filtro para três momentos distintos na ação do robô no ambiente. Na primeira figura, tem-se a situação de um raio fixo com as medidas sendo distribuídas em três possibilidades, onde pode-se observar o problema da ocorrência de poucas medidas fazendo *match*. Na segunda figura, é demonstrado o raio sendo alterado a cada pose do robô, com intuito de obter o maior numero de *match* possível.

Figura 3.2: Distribuição das medidas no círculo de raio fixo:
(a): todas pertencentes; (b): a maioria pertencente; (c): maior dispersão



Fonte: Autor

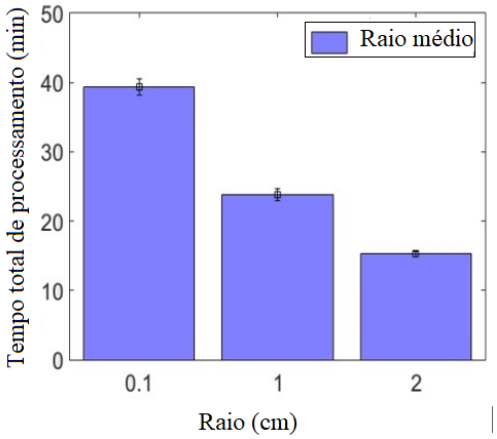
Figura 3.3: Distribuição das medidas com raio variável
(a): raio pequeno; (b): raio médio; (c): raio grande



Fonte: Autor

O uso de um único raio de valor pequeno eleva o tempo que o robô gasta para realizar o SLAM do ambiente. Essa lentidão ocorre porque as medidas que não pertencem ao círculo (não fizeram *match*) serão acrescentadas ao vetor ω (Equações 3.1 e 3.2), consequentemente cada partícula tem esse vetor de tamanho aumentado. Os tamanhos dos vetores, aliados à quantidade de partículas, tendem a gerar mais tempo de processamento computacional, tornando impraticável o uso de tal método em sistemas embarcados, onde capacidade de memória e processamento são baixos quando comparados aos computadores mais robustos. A Figura 3.4 mostra a relação de tempo médio de processamento de acordo com o raio escolhido em um dado ambiente no qual foi realizado SLAM.

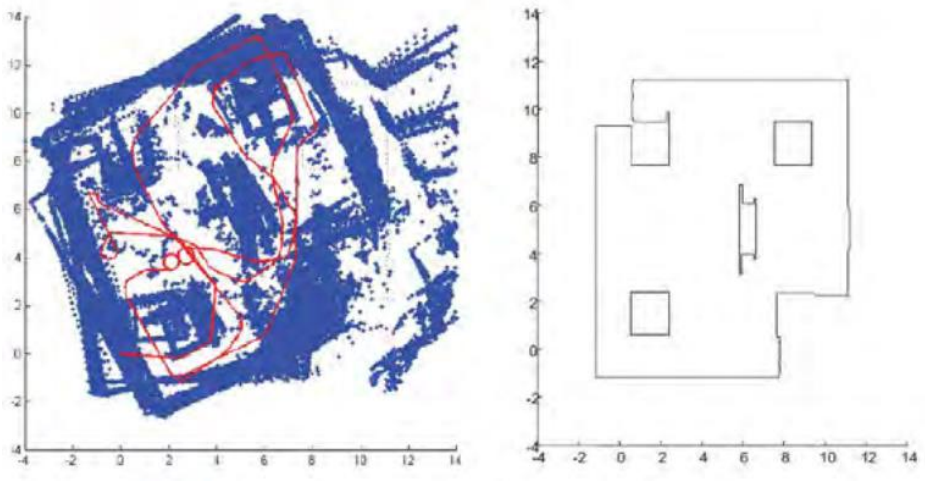
Figura 3.4: Tempo de exploração de um ambiente de acordo com o valor de raio adotado



Fonte: Adaptado de Buonocore *et al* (2017)

Conclui-se que um raio variável seria capaz de otimizar o tempo de processamento. Com este problema solucionado, a questão gira em torno de verificar se a variação dos raios não afetaria a estimação da pose, uma vez que cada partícula teria o vetor ω de tamanho elevado e diferente. Inclusive, este é o maior problema que deve ser solucionado, pois de nada adiantaria melhorar o tempo de processamento que o robô gera o mapa com informações erradas, pois apresentaria distorções no mapa, devido às associações de dados das medidas. Na Figura 3.5, tem-se um mapa completo de um ambiente que foi explorado por um robô, onde se percebe a problemática da associação de dados.

Figura 3.5: Exemplo de erro na associação de dados na geração de mapa de um ambiente

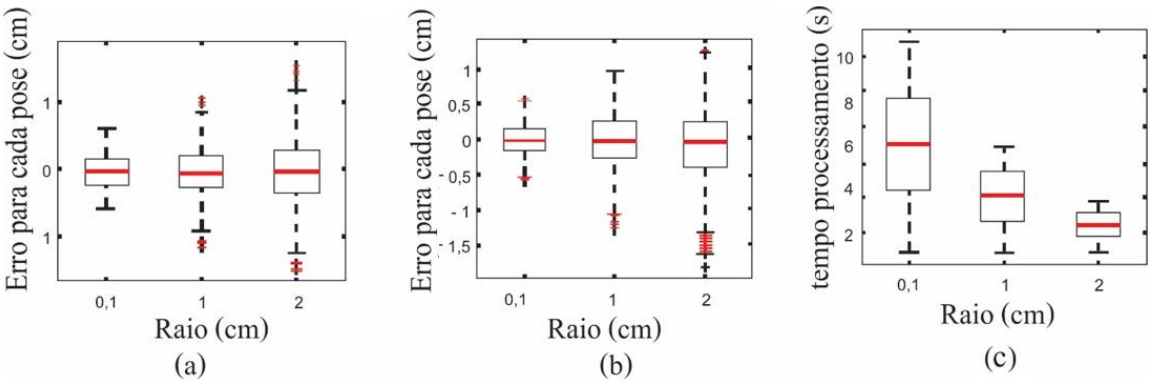


Fonte: Bigheti (2011)

Buonocore *et al.* (2017) mostraram em seu trabalho que buscaram encontrar um raio capaz de fornecer um erro pequeno na estimação da pose, conforme pode ser observado na Figura 3.6.

Figura 3.6: Erros na pose para um dado raio escolhido

(a) Erro na coordenada x; (b) Erro na coordenada y; (c) tempo de processamento de acordo com o raio



Fonte: Adaptado de Buonocore *et al.* (2017)

Com as informações das Figuras 3.4 e 3.6, pode-se concluir que: um raio grande melhora o tempo de processamento, mas aumenta o erro das poses; um raio pequeno diminui o erro na estimação, mas aumenta o tempo de execução. A escolha de um raio capaz de fornecer a pose do robô com o mínimo de erro (problema de minimização), ou seja, máximo de *matches*, em um tempo adequado, é um grande desafio.

3.3 Inteligência artificial na estratégia de ponderação das partículas

Nota-se que diante dos problemas explicados nos parágrafos anteriores, torna-se impraticável o uso de método empírico, pois para encontrar o raio (método empírico) com menor erro, seria necessário um gasto de tempo na simulação por parte do projetista. Assim, o uso de técnica de inteligência artificial na escolha do raio mais adequado torna-se uma ótima estratégia. Neste trabalho, usou-se uma rede neural artificial do tipo MLP, com algoritmo de aprendizagem *backpropagation*, aprendizagem supervisionada, com 202 neurônios na camada de entrada, 20 neurônios na camada intermediária e um neurônio na camada de saída. A estratégia de aprendizado da rede neural artificial consiste em uma base de dados que contem a distribuição das partículas, medidas brutas do sensor e o raio ótimo, obtidos em cada estado do robô.

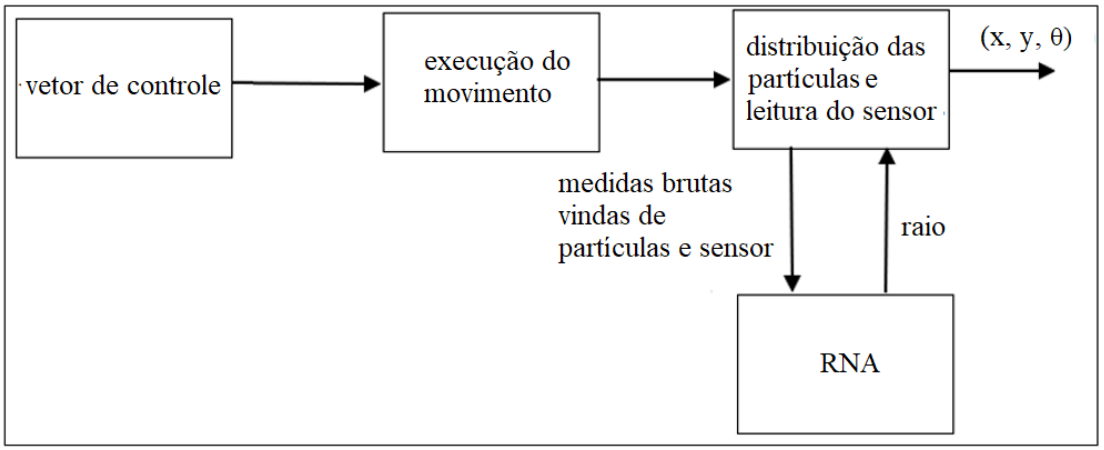
Para resolver o problema descrito neste trabalho, a RNA deve ser capaz de fornecer o raio que minimize o erro na estimação da pose, feita pelo filtro de partículas, dado que ela possui como informações somente os dados brutos do sensor e a disposição das partículas. O erro mínimo procurado é aquele que nas três informações, em conjunto, da pose seja o menor possível. Não significa que este erro seja o menor para cada informação, pois as três informações, quando analisadas separadamente, apresentam erros distintos de acordo com o raio aplicado em cada estado de exploração pelo robô no ambiente. Este problema pode ser definido através de uma função dada por:

$$r_s = f(\alpha_i, \omega_i) \;\; \forall(\alpha, \omega) \in s \quad \text{e } i = \{1, ..., N\} \tag{3.5}$$

onde r representa o raio das medidas brutas mantidas nas partículas i com pose α e dados do sensor ω, ambos pertencentes ao espaço s (ambiente de exploração).

Na Figura 3.7, tem-se o esquema de obtenção do raio. O vetor de controle é responsável por fornecer o tipo de movimento a ser executado (angular ou linear) e o local de parada do robô. O robô executa o movimento ordenado, parando no ponto determinado. O filtro de partícula então é acionado, distribuindo as partículas por uma dada região em torno do robô. Em seguida, a RNA analisa os dados provenientes do filtro e do sensor e fornece o raio que minimize o erro na estimação da pose. Finalmente, com o raio fornecido, o filtro estima a localização do robô no ambiente.

Figura 3.7: Estratégia de obtenção do raio



Fonte: Autor

3.4 Materiais utilizados e métodos

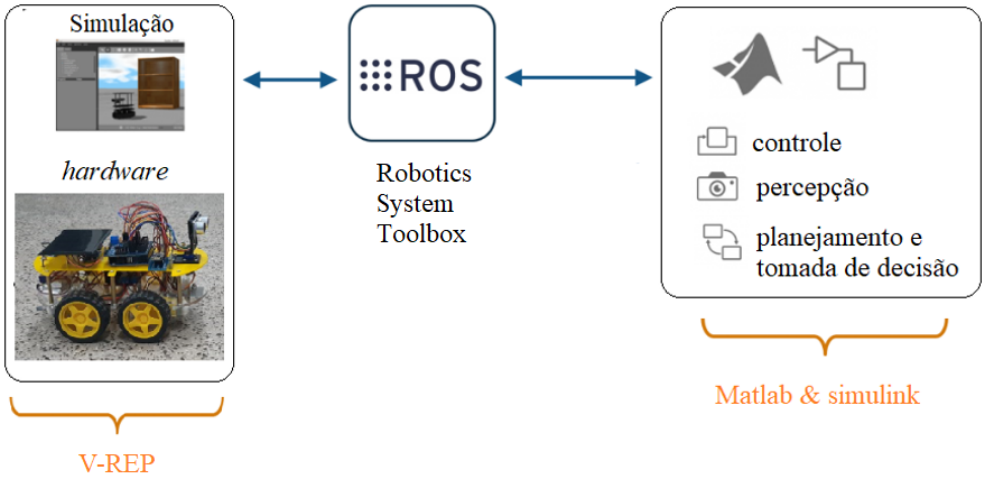
Para realizar a implementação da metodologia proposta neste trabalho, usaram-se ferramentas computacionais, pois uma implementação real demandaria muitos gastos na montagem e aquisição de componentes de *hardware* e dependeria de fontes financeiras. Os algoritmos do filtro de partículas, mapa de movimento do robô, ação de controle e geração de mapa foram feitos no Matlab. Trata-se de uma ferramenta computacional de grande interatividade e alta performance voltado para o cálculo numérico, permitindo realizar tarefas de análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos, reunidos em um ambiente de fácil utilização.

Para a construção do ambiente, escolha do robô e do sensor, usou-se o V-REP (*Virtual Robot Experimentation Platform*). Trata-se de um *software* desenvolvido para a criação e simulação de robôs para diversas aplicações, tanto móveis quanto estáticos, de solo, aéreos ou aquáticos. Diversos modelos de robôs amplamente utilizados já estão disponíveis no V-REP, sendo também possível o desenvolvimento do seu próprio robô e os cenários para a simulação. O inconveniente é que esta ferramenta somente está disponível para sistemas operacionais Linux. Assim, computadores com sistemas operacionais Windows precisam ter instalados uma máquina virtual Linux.

Para realizar a comunicação entre o Matlab, instalado no sistema operacional Windows, com o V-REP, usou-se o ROS (*Robot Operating System*). Este programa é um *middleware* que se hospeda em um sistema operacional, fornecendo bibliotecas e ferramentas para auxiliar o desenvolvimento de software e criação de aplicativos robóticos. Ele fornece abstração de *hardware*, *drivers* de dispositivos, bibliotecas, visualizadores, passagem de mensagens,

gerenciamento de pacotes e muito mais. O ROS é completamente *open source* e livre para uso, customização e comercialização. A Figura 3.8 resume o processo de simulação.

Figura 3.8: Sistema de simulação com Matlab e V-REP



Fonte: Autor

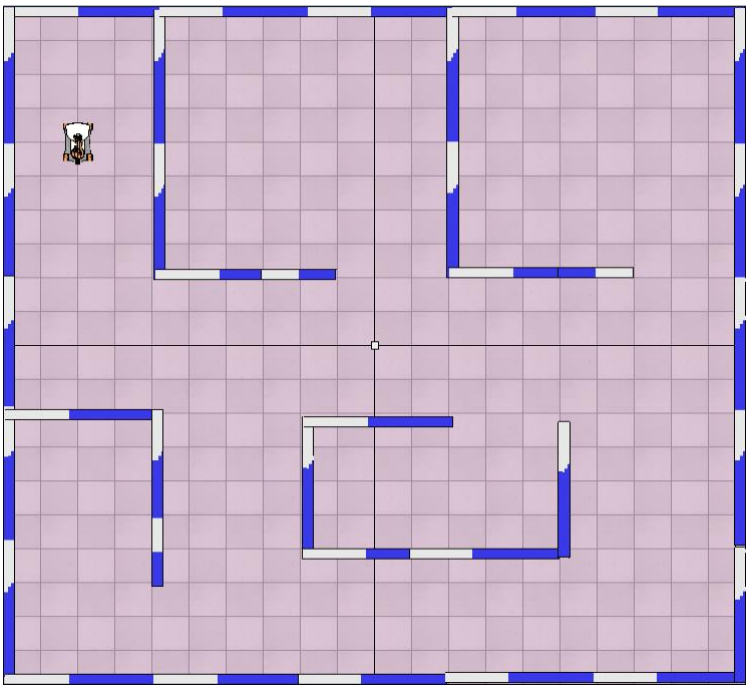
4 Experimentos e resultados

Com objetivo de testar a metodologia proposta, foram realizados vários experimentos em ambiente de simulação. Os experimentos iniciais foram realizados usando a plataforma de simulação V-REP, instalada em uma máquina virtual com sistema operacional Linux, para criação do ambiente de exploração e inserção do robô. O sistema de inteligência e controle do robô, processamento de dados e geração de mapa foram feitos no Matlab, instalado no ambiente Windows. O robô desloca-se de acordo com os dados ordenados pelo vetor de controle, onde sua velocidade de movimento linear e angular são também controladas. O robô é dotado de um sensor *laser scanner* usado no processo de SLAM.

No vetor de controle, é informado ao robô se ele deve mover-se linearmente, no eixo x ou no eixo y, ou realizar movimento angular. A cada movimento realizado, em pontos específicos, é feito a leitura do sensor.

Na Figura 4.1, tem-se a imagem do ambiente de exploração construindo no V-REP. O robô parte de um ponto especifico, ou seja, a pose é conhecida. Trata-se de uma aplicação com exploração não autônoma, pois cada movimento que o robô fará é determinado por uma matriz de movimento (vetor de controle). A grande questão deste processo consiste em verificar se a pose desejada de parada do robô coincide com a sua pose no mapa.

Figura 4.1: Ambiente a ser explorado pelo robô na simulação



Fonte: Nojosa (2016)

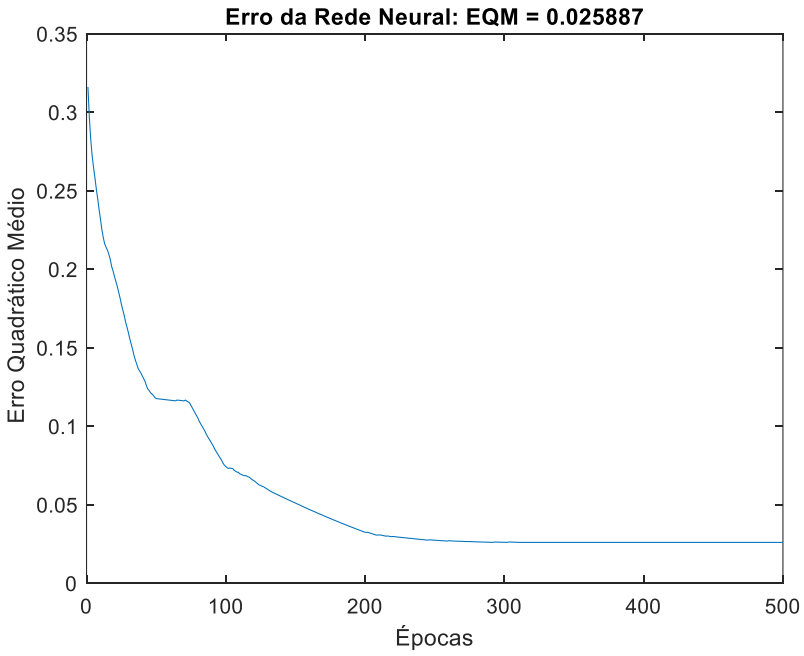
Estimar a pose do robô que realiza SLAM no ambiente é um processo complexo, haja vista a matemática envolvida, além do erro nas informações provenientes dos sensores. Por isso, neste trabalho, usaram-se técnicas de inteligência artificial auxiliando métodos probabilísticos na estimação da localização. Conforme explicado anteriormente, a RNA deve ser capaz de fornecer os raios para o filtro de partículas realizar a ponderação das N partículas.

Desta forma, a RNA deve passar por um processo de aprendizagem para saber definir o tamanho adequado para os raios do filtro de partículas. Neste ponto, foram realizados diversos treinamentos com diferentes configurações da RNA, por exemplo, dados de entrada, função de ativação, taxa de aprendizado, quantidade de neurônios na camada escondida e uso de bias foram os principais elementos modificados nos diferentes treinamentos.

Por exemplo, na Figura 4.2, tem-se o erro quadrático médio da rede neural com as seguintes configurações: 202 neurônios na camada de entrada, 20 neurônios na camada escondida, um neurônio na camada de saída, função de ativação tangente hiperbólica na cada escondida, função linear com k=1na camada de saída e sem uso de bias. O treinamento consistiu de 500 épocas, tendo uma taxa de aprendizado 0,01.

Esta configuração apresentou os melhores resultados na estimação dos raios do filtro de partículas. Outras configurações foram também adequadas, porém com resultados inferiores, enquanto que algumas não tiveram bons resultados. Na melhor configuração, o tempo de treinamento levou aproximadamente 12 minutos.

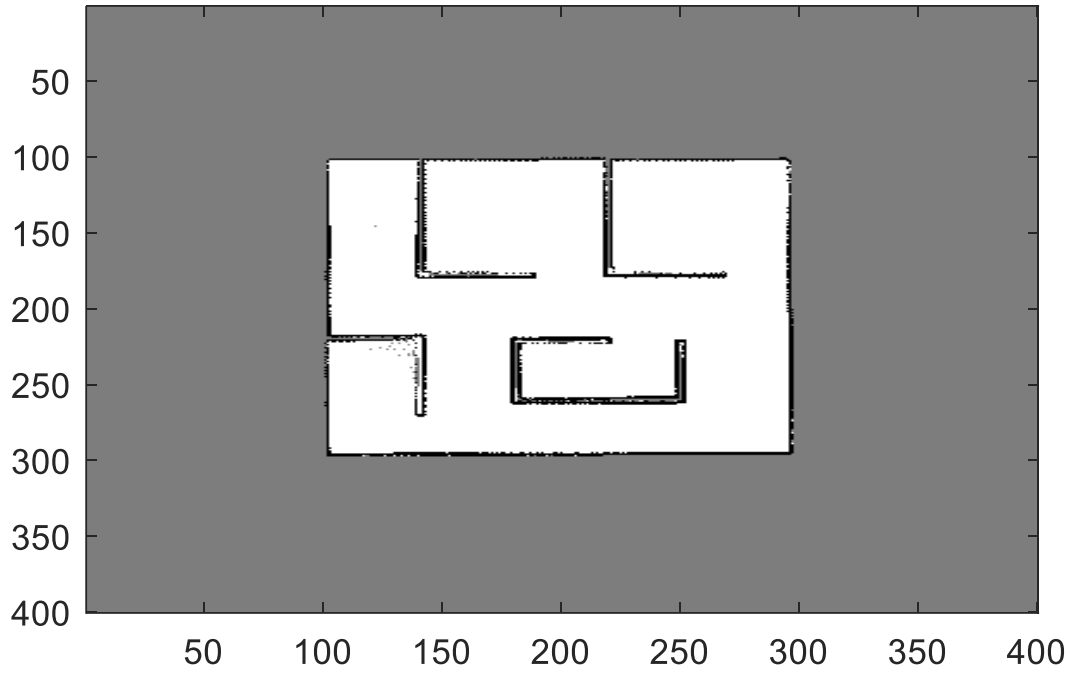
Figura 4.2: Erro quadrático médio da RNA



Fonte: Autor

Com o uso da configuração da rede acima, o resultado obtido na geração do mapa pode ser visto na Figura 4.3. Nesta figura, são fornecidas informações sobre o formato do ambiente e obstáculos existentes. Para o desenvolvimento do mapa métrico, foi utilizado grid de 200x200, células de 5x5 cm e área de 10 x 10 m^2. Essas configurações são importantes porque definem o formato e escala do mapa gerado.

Figura 4.3: Mapa gerado pelo algoritmo

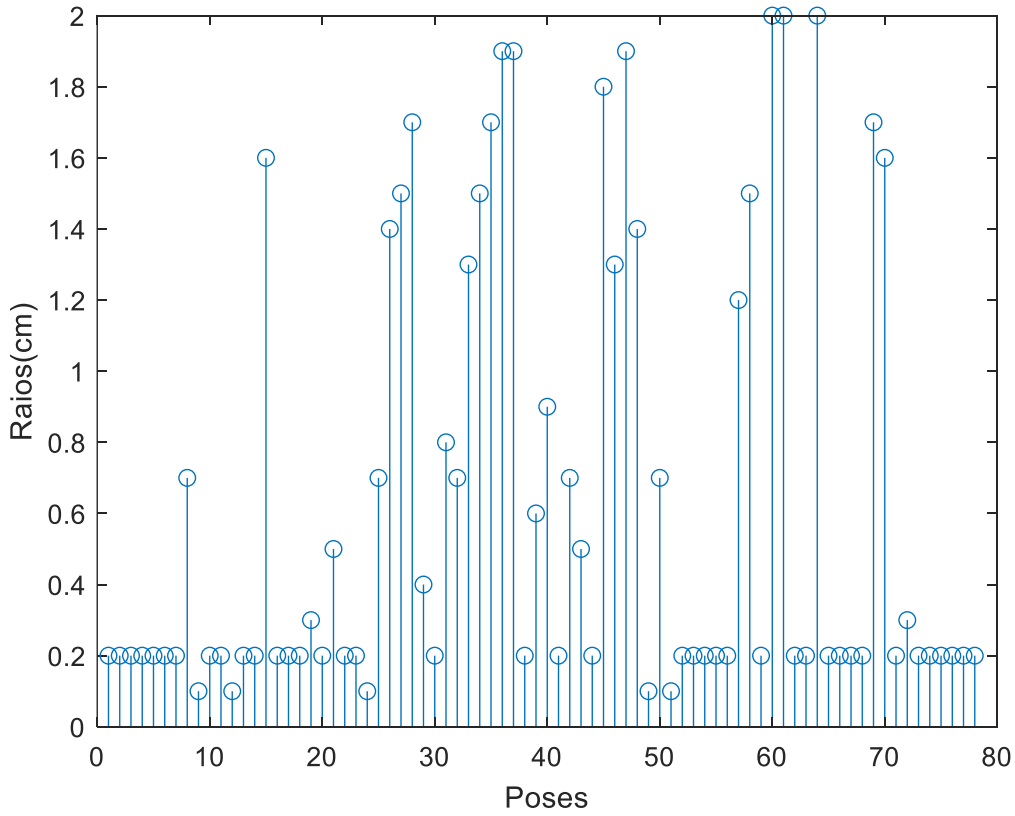


Fonte: Autor

Na Figura 4.4, têm-se os raios fornecidos pela rede para cada pose do robô. Um fato interessante ocorre quando o robô realiza movimento angular. Neste tipo de movimento, a rede fornece raios pequenos e constante. Isso ocorre devido as coordenadas x e y do robô não sofrerem grandes alterações em seus valores durante o movimento angular e, dessa forma, a maioria das medidas adquiridas na varredura do *laser* deverão estar contidas nas medidas brutas armazenadas pelas partículas. Assim, uma quantidade menor de medidas seria necessária para estimar a pose. Quando o robô realiza movimento linear, ocorre um aumento de medidas brutas no vetor de correspondência das partículas na medida em que ele explora lugares ainda desconhecidos.

A dinâmica no raio ocorre devido as medidas que não fizeram *match* serem inseridas no vetor de correspondência. Assim, na próxima pose do robô, a RNA fornece um raio que consiga diminuir o erro na estimação da pose e, consequentemente, consiga obter maior número de *match*, fazendo com que o vetor de correspondência não aumente.

Figura 4.4: *Matching* fornecidos pela rede para cada pose do robô



Fonte: autor

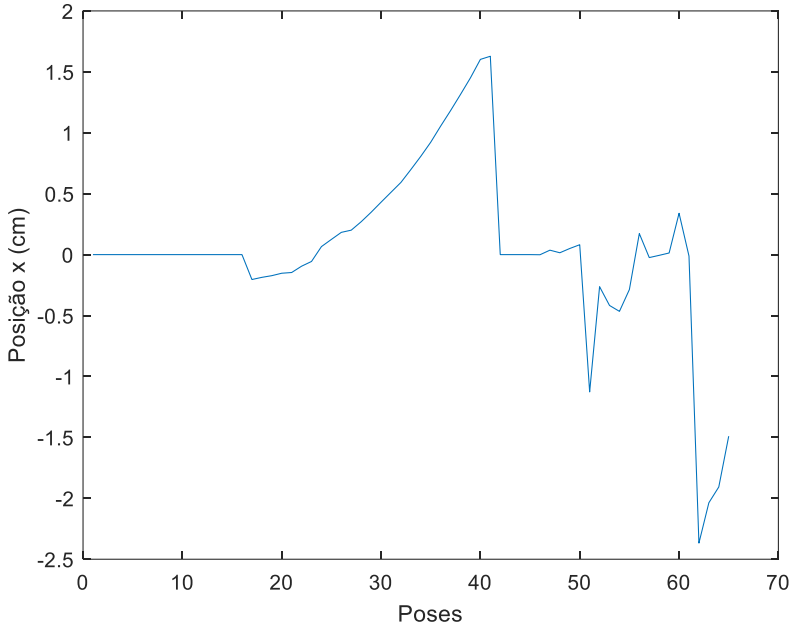
Nas Figuras 4.5, 4.6 e 4.7, têm-se os erros em (x, y, θ) para cada pose. As diferenças dos erros nas posições são maiores em movimento linear, devido à distribuição das partículas e ao raio adotado não ser capaz de fornecer um baixo erro no eixo x e no eixo y, simultaneamente.

O erro obtido na coordenada x não é o mesmo obtido na coordenada y, pois durante o desenvolvimento da metodologia proposta neste trabalho, verificou-se que a escolha de um raio que fornecesse um erro mínimo em umas das coordenadas não necessariamente ocorreria a mesma situação na outra coordenada. Diante disso, foi desenvolvida a estratégia de obtenção do raio que conseguisse fornecer um erro pequeno em ambos os eixos (raio ótimo).

Observa-se nas Figuras 4.5 e 4.6 que, no início do processo de exploração do robô no ambiente, não há erros nas coordenadas x e y, uma vez que o robô está realizando apenas

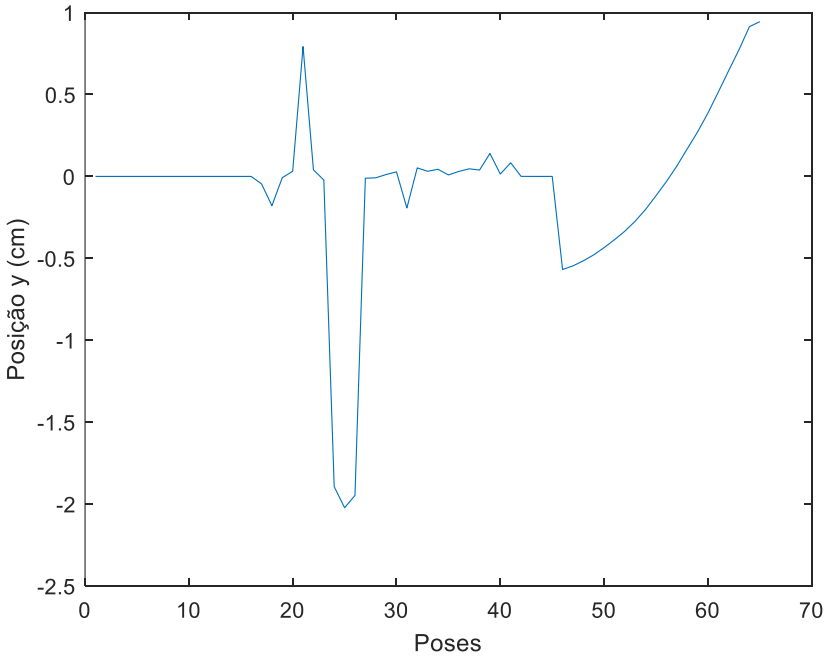
movimento angular. Após isso, o robô realiza movimento linear, consequentemente as médias brutas do movimento angular, que não fizeram *match*, irão influenciar na estimação da pose e no erro nas coordenadas x e y.

Figura 4.5: Erros na coordenada x em cada pose do robô



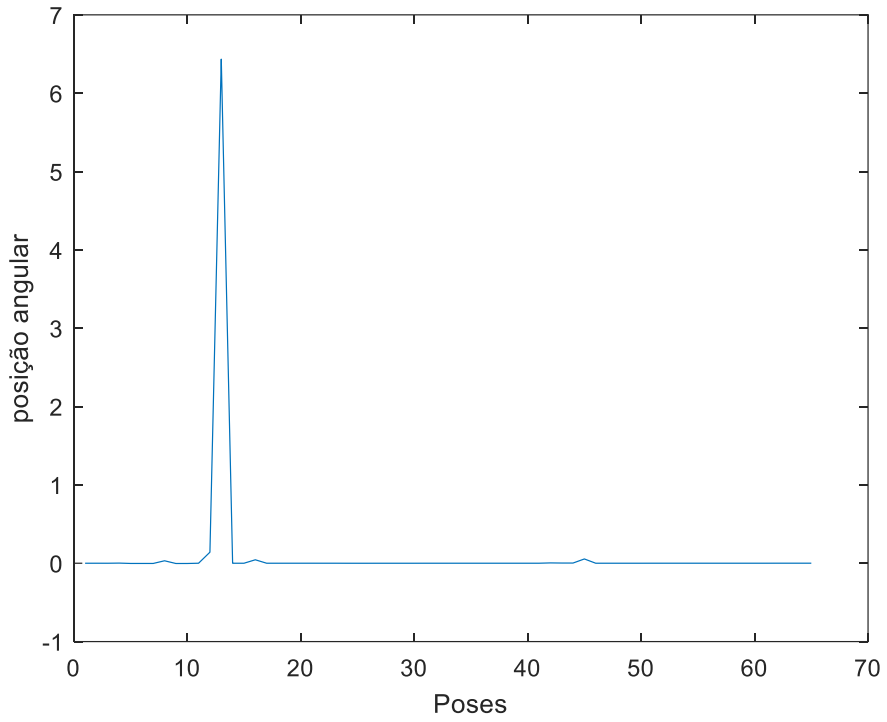
Fonte: autor

Figura 4.6: Erros na coordenada y em cada pose do robô



Fonte: autor

Figura 4.7: Erros na coordenada angular θ em cada pose do robô



Fonte: autor

A fim de verificar a eficiência do método proposto neste trabalho, foram aplicados raios fixos de 0.1, 0.5, 1, 1.5 e 2 cm no processo de exploração do ambiente. Em seguida, esses valores foram comparados ao raio ótimo fornecido pela RNA. A Tabela 1 apresenta o erro quadrático médio de acordo com o raio escolhido. Observou-se que o raio fornecido pela RNA apresentou os menores erros nas três grandezas da pose do robô, quando comparado aos outros valores de raios.

Tabela 1: Erro quadrático médio das poses de acordo com o raio

Medidas	**Erro quadrático médio**					
	raio (RNA)	**0.1 cm**	**0.5 cm**	**1 cm**	**1.5 cm**	**2 cm**
Coordenada x	0.4981	10.3626	3.4049	0.6530	0.6561	0.6170
Coordenada y	0.2641	7.9821	0.4478	0.3075	0.1543	0.2608
Ângulo teta	0.6373	0.6530	0.6411	0.6411	0.6411	0.6373

Fonte: Autor

Percebe-se pela tabela acima que, para a coordenada x, o erro quadrático médio apresentado pela solução via RNA diminuiu aproximadamente 19,27% do melhor caso com

raio fixo (2 cm). Na coordenada y, observa-se que houve um aumento de aproximadamente 71,16% no erro quadrático medido apresentado pela solução via RNA quando comparada ao melhor caso (raio fixo de 1,5 cm). Diante dos resultados obtidos, observa-se que a metodologia desenvolvida neste trabalho apresentou melhor desempenho em relação ao método de uso de raio fixo.

5 Conclusão

Uma melhoria no sistema para navegação de robôs móveis usando o algoritmo probabilístico para o mapeamento e localização simultâneos do robô foi proposto, simulado e avaliado neste trabalho. Para simulação de navegação, utilizou-se a integração do filtro de partículas com um sensor *laser scanner*. No experimento de coleta dos dados das medidas do ambiente, a principal dificuldade constatada foi lidar com alta taxa de erro em cada parada do robô, sendo realizados cansativos ajustes nas velocidades angulares e lineares. Este problema ocorre devido a limitação do processador e baixa taxa de fps (quadro por segundo, em inglês), na comunicação entre o Matlab e o V-REP. Uma solução possível seria fazer uso do método odométrico para realizar este ajuste. No teste de comparação entre os dados do vetor de controle fornecido ao robô e os dados no V-REP os resultados mostraram um desempenho satisfatório.

Há um crescente uso de robôs móveis em aplicações específicas de monitoramento, inspeção e para auxilio humano nas tarefas que exigem precisão ou que são difíceis de serem executadas. Diante disso, cada vez mais torna-se necessário o emprego de sensores de baixo custo em tais robôs. Então, este projeto vem ao encontro com uma necessidade emergente da sociedade, onde robôs móveis, autônomos e não-autônomos dotados de sensores (o tipo depende da aplicação do robô), estão sendo cada vez mais solicitados para diversas aplicações. Nesse contexto, a metodologia proposta busca atender esta demanda, incorporando inovações tecnológicas no desenvolvimento do *hardware* e do *software* de um robô que, simultaneamente, navega, mapeia e localiza-se em um ambiente. A metodologia proposta foi desenvolvida com o emprego de um computador do tipo *notebook* com processador Core i7 7ª geração de dois núcleos e memória de 16 GB. A configuração do computador é um item importante, pois os resultados da metodologia proposta neste trabalho são mais rapidamente obtidos considerando-se aspectos de desempenho, comunicação e memória do computador.

Neste trabalho, apresentou-se uma nova metodologia que tem como principal contribuição obter a pose de um robô que realiza medições do ambiente e navega usando um sensor de baixo custo e que apresenta ruído na sua medição.

Outra contribuição é a integração de técnicas de inteligência artificial e métodos probabilísticos para solucionar problemas de SLAM, criando novas possibilidades de que outras técnicas possam ser utilizadas neste processo, podendo ser aplicado a outros tipos de sensores.

A fim de analisar a metodologia desenvolvida neste trabalho, foram realizadas

comparações com diferentes valores de raios fixos. Nestas comparações, observou-se que o método proposto apresentou bons resultados de desempenho.

As sugestões de trabalhos futuros incluem:

- Inserção de duas fontes sensoriais no processo de obtenção dos dados brutos e assim, verificar possíveis melhorias na estimação da pose com o método proposto neste trabalho.

- Realizar o mapeamento com múltiplos robôs, a fim de verificar possíveis melhorias (tempo, qualidade, etc.) na criação do mapa do ambiente quando comparado ao processo feito por um único robô.

- Analisar outro modelo de rede neural na integração com o filtro probabilístico que consiga fornecer melhores resultados na estimação da pose do robô, levando em consideração o tempo de exploração e de processamento.

REFERÊNCIAS

AHN, S. and K. Lee. *SLAM with visual plane: Extracting vertical plane by fusing stereo vision and ultrasonic sensor for indoor environment*. IEEE international Conference on Robotics and Automation. p. 4787-4794, 2007.

ANOUSAKI, G. C. and KOPOULOS, K. J. *Simultaneous localization and map building of skid-steered robots.* IEEE Robotics and Automation Magazine, p. 79-89, 2007.

ARULAMPALAM, M. S. *et al*. *A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking*. IEEE Transactions on Signal Processing, v. 50, p. 174–188, 2002.

BAILEY, T. Mobile robot localization and mapping in extensive outdoor environments. PhD thesis. Australian Centre for Field Robotics, University of Sydney, August 2002.

BAILEY, T.; DURRANT-WHYTE, H. F. Simultaneous Localization and Mapping (SLAM): part I the essential algorithms. Robotics & Automation Magazine, IEEE v.13, ed. 2, p.99– 110, June 2006a

BAILEY, T.; DURRANT-WHYTE, H. F. Simultaneous Localization and Mapping (SLAM): part II state of the art. Robotics & Automation Magazine, IEEE v.13, ed. 3, p.108–117, September 2006b.

BAILEY T., NIETO J., GUIVANT J., STEVENS M., AND NEBOT E.; Consistency of the EKF-SLAM Algorithm. In IEEE/RSJ International Conference on Intelligent Robots and Systems. Beijing, P.R. China, 9 - 15 October 2006, 10.1109/IROS.2006.281644, 7 pages, 2006c.

BAYES, M.; PRICE, M. An essay towards solving a problem in the doctrine of chances. By the Late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions*, v. 53, p. 370–418, 1763.

BIGHETI, Jeferson André. NAVEGAÇÃO DE ROBÔS EM AMBIENTES INTERNOS USANDO SLAM. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, 2011.

BLANCO, J.; MADRIGAL, J. e GONZALEZ, J. *A new approach for large-scale localization and mapping: Hybrid metric-topological SLAM.* IEEE International Conference on Robotics and Automation, p. 2061-2067,2008.

BORENSTEIN, J.; EVERETT, H.R.; FENG, L. *Where am I? Sensors and methods for mobile robot positioning*, University of Michigan, USA, 1996.

BORENSTEIN, J. and FENG. L. , "*UMBmark: A Benchmark Test for Measuring DeadReckoning Errors in Mobile Robots*." *SPIE Conference on Mobile Robots, Philadelphia, 1995.*

BOLTON, W. *Mecatrônica: Uma abordagem multidisciplinar*. 4ª edição, Porto Alegre. Editora Bookman, 2010.

BRAGA, A. P.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B., Redes Neurais Artificiais: Teoria e Aplicações, LTC, 2007.

BUONOCORE, L., SANTOS, S., NETO, A., OLIVEIRA, A. and JR., C. *A New Particle Weighting Strategy for Robot Mapping FastSLAM.* 14th International Conference on Informatics in Control, Automation and Robotics, v.2, p. 322-328,2017.

BURGARD, W. *et al*. The interactive museum tour-guide robot. In: National Conference on Artificial Intelligence. Proceedings, 1998.

BURGARD, W.; STACHNISS, G.; GRISETTI, B.; STEDER, R.; KUMMERLE, C.; DORNHEGE, M.; RUHNKE, A; KLEINER e TARDOS, J. *A comparison of SLAM algorithms based on a graph of relations.* IEEE International Conference on Robotics and Automation, p. 2085-2089,2009.

CAPPE, O.; GODSILL, S. J.; MOULINES, E. *An overview of existing methods and recent advances in sequential monte carlo*. Proceedings of the IEEE, v. 95, n. 5, p. 899–924, Jul. 2007.

CHOSET, H.; NAGATANI, K. *topological simultaneous localization and mapping (SLAM)*: Toward exact localization without explicit localization. IEEE Transactions on Robotic and Automation, v.17, n 2, p 125-137, April 2001.

DE ALMEIDA NETO, A., Aplicações de Múltiplas Redes Neurais em Sistemas Mecatrônicos, ITA, 2003.

DIOSI, A., G. Taylor e L. Kleeman. *Interactive SLAM using laser and advanced sonar'*. IEEE International Conference on Robotics and Automation, pp. 1103-1108, 2005.

DOUCET, A.; FREITAS, N. de; GORDON, N. *Sequential Monte Carlo Methods in Practice*. Berlin: Springer-Verlag, 2001.

DURRANT-WHYTE, Hugh; BAILEY, Tim. *Simultaneous Localization and Mapping (SLAM): Part I The Essential Algorithms*. 9 f. IEEE, 2006.

ELFES, A. *Occupancy grids:* a probabilistic framework for robot perception and navigation. Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

FENG, H.; YONGCHUN, F.; YUTAO, W. e TAO, B. *Practical feature based simultaneous localization and mapping using sonar data.* Chinese control conference, p. 16-18, 2008.

FU, S.; LIU, L. Gao and Y. Gai. *SLAM for mobile robots using laser range finder and monocular vision*. IEEE International Conference on Robotics and Automation. Roma, Itália, pp. 91-96, 2007b.

GORDON, N. J.; SALMOND, D. J.; SMITH, A. F. M. Novel approach to nonlinear/nongaussian bayesian state estimation. In: *IEE Proceedings F: Radar and Signal Processing*. [S.l.: s.n.], 1993. v. 140, n. 2, p. 107–113.

HAYKIN, S., *Neural Networks: a comprehensive foundation*. New York: MacMillan College Publishing Co., 1999.

HAYKIN, S., Neural Networks: a comprehensive foundation, Prentice Hall, 1999.

HUANG, L.; HE, B.; ZHANG, T. *An autonomous navigation algorithm for underwater vehicles based on inertial measurement units and sonar.* International Asia Conference on Informatics in Control, automation and robotics, p. 311-314, 2010.

JANÉT, J. *et al*. Autonomous mobile robot global self-localization using Kohonen and region -feature neural networks. *Journal of Robotic Systems*, v 14, n. 4, p. 263-282, 1997.

KALMAN, R. E. A new approach to linear filtering and prediction problems. Journal of Basic Engineering, 1960.

KOOTSTRA, G. e SCHOMAKER, L. *Using symmetrical regions of interest to improve visual SLAM.* IEEE/RSJ International Conference on Robotics and Automation, p. 930-935,2009.

KUIPERS, B.; BYAN, Y.T. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representation. *Journal of Robotics and Autonomous Systems*, v.8, p.47-63,1991.

KRÖSE, B. J. A. *et al*. A probabilistic model for appearance-based robot localization. *Image and Vision Computing* v.19, p.381-391,2001.

LEONARD, J. J. *et al*. Direct sonar sensing for mobile robot navigation. Kluwer Academic Publishers, 1992.

LEONARD, J. J.; FEDER, H. J.S. *A computationally efficient method for large-scale concurrent mapping and localization*. In: Proceedings of the Ninth International Symposium on Robotics Research, p. 1442-1447, 1991.

LIMA, Leandro Muniz de. *Filtro de partículas hibridizado com métodos da computação natural para detecção e rastreamento*. 59f. Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, 2011.

MATARIC, M. J. A distributed model for mobile robot environment-learning and navigation. Cambridge, 1990.

MCCULLOCH, W. and PITTS, W. "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

MILFORD, M. J. e WYETH, F. G. *Single camera vision-only SLAM on a suburban road network.* IEEE International Conference on Robotics and Automation, 2008.

MONTERMELO, M. *et al*. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: National Conference on Artificial Intelligence. Proceedings. Edmonton (Canada), 2002.

MONTEMERLO, M. (2003a). *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Ph.D. dissertation, Dept. Elect. Eng., Robotics Institute, Carnegie Mellon University.

MONTEMERLO, M.; THRUN, S. (2003b). *Simultaneous localization and mapping with unknown data association using FastSLAM*. In *IEEE International Conference on Robotics and Automation, pp. 1985–1991*.

MONTEMERLO, M.; THRUN, S. K. D. W.-B. (2003c). FastSLAM 2.0: *An improved particle filtering algorithm for simultaneous localization and mapping that provably converges*. In *International Joint Conference on Artificial Intelligence, p. 1151–1156.*

MORAVEC, H. P.; ELFES, A. High resolution maps from wide angle sonar. In: International Conference on Robotics and Automation. Proceedings. IEEE, 1985.

NOJOSA, R. V. F. Implementação de uma Estrutura de Simulação Usando Matlab – Ros – V-REP para Validação de um Algoritmo de Exploração não Autônomo. Monografia do Curso de Graduação em Engenharia Elétrica, Universidade Federal do Maranhão, 2016.

NEWMAN, Paul Michael. *EKF Based Navigation and SLAM: Background Material, Notes and Example Cod*e. 94 f. SLAM Summer School 2006, Oxford.

NEWMAN, P. M.; On the Structure and Solution of the Simultaneous Localization and Map Building Problem. PhD thesis, ACFR, Univ. of Sydney, Austrália, March. 1999.

OLIVEIRA, J. R. *Um sistema integrado para navegação autônoma de robôs móveis.* 2010. 100 f. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2010.

ORIOLO, G.; ULIVI, G.; VENDITTELLI, M. Fuzzy maps: a new tool for mobile robot perception and planning. *Journal of Robotic Systems,* v. 14, n. 3, p. 179-197, 1997.

ROMERO, Roseli; OSÓRIO, Fernando; WOLF, Denis e PRESTES, Edson. *Robótica Móvel*. Editora LTC. 2014.

SANTANA, André Macedo. *Localização e Mapeamento de Ambientes Planos Usando Visão Monocular e Representação Híbrida do Ambiente.* 149 f. Tese (Doutorado)-Engenharia da Computação, Universidade Federal do Rio Grande do Norte, 2011.

SASIADEK J. Z., and HARTANA P.; Sensor Fusion for dead-reckoning mobile robot navigation. Proceeding of the 6th IFAC Symposium on Robot Control, SYROCO, 2000.

SASIADEK J. Z., and HARTANA P.; Odometery and sensor data fusion for mobile robot navigation. Proceeding of the 6th IFAC Symposium on Robot Control, SYROCO, 2000.

SMITH, R., SELF M., and CHEESMAN P.; Estimating uncertain spatial relationships in robotics. In Autonomous robot vehicles, I. J. Coxand, G. T. Wilforn, Ed., 1990.

SMITH, R., SELF M., AND CHEESMAN P.; Estimating uncertain spatial relationships in robotics. In Autonomous robot vehicles, I. J. Coxand, G. T. Wilforn, Ed., 1990.

SKRZYPCZYNSKI, P. *Simultaneos localizations and mapping: A feature based probabilistic approach*. Joirnal of Applied Mathematics and Computer Science, vol. 19, No. 4, p. 575-588, 2009.

THRUN, S.; BURGARD, W.; FOX, D. A probabilistic approach to concurrent mapping and localization for mobile robotics. *Machine Learning*, Hingham, 1998.

THRUN, Sebastian; *Robotic mapping: a survey. Exploring Artifical Intelligence in the New Millenium*.Morgan Kaufmann,2002.

THRUN, S. *et al*. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics and Automation Magazine*, New York, n. 4, p. 79-91, December, 2004a.

THRUN, S.; MONTEMERLO, M. K. D. W.-B. N. J. N. E. *An Efficient Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Journal of Machine Learning Research, 2004.

THRUN, S.; BURGARD, W. F. D. *Probabilistic Robotics*. The MIT Press, 2005.

WELLE, J.; SCHULZ, D.; BACHARAN, T. e CREMERS, A. *Optimization techiniques for laser-based 3D particle filter SLAM*. IEEE International Conference on Robotics and Automation, Alaska, p. 3535-3530, 2010.

WILLIAMS, B. e REID, I. *On combining visual SLAM and visual odometry*. IEEE International Conference on Robotics and Automation, Alaska, p. 3525-3530, 2010.

YAP, T. N. e SHELTON, C. R. SLAM in large indoor environments with low-cost, noisy, and sparse sonars. IEEE International Conference on Robotics and Automation, Alaska, p. 12-17, 2009.

VLASSIS, N. KRÖSE, B. J. A. *Robot environment modeling via principal component regression.* In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999.