UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA MECATRÔNICA

Willian Marinho Pinheiro

Algoritmo para localização e Mapeamento Simultâneo com Múltiplos Robôs em Ambientes Internos Utilizando Filtro de Partículas

Manaus
2015

Willian Marinho Pinheiro

Algoritmo para localização e Mapeamento Simultâneo com Múltiplos Robôs em Ambientes Internos Utilizando Filtro de Partículas

Trabalho de Conclusão de Curso desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II, apresentado na Universidade do Estado do Amazonas – UEA sob orientação do Me. Charles Luiz da Silva Melo, como pré-requisito para obtenção do título de Engenheiro Mecatrônico.

Orientador Me. Charles Luiz da Silva Melo
Coorientador: André Dias de Lima Machado

Manaus
2015

Willian Marinho Pinheiro

Algoritmo para localização e Mapeamento Simultâneo com Múltiplos Robôs em Ambientes Internos Utilizando Filtro de Partículas

Trabalho de Conclusão de Curso desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II, apresentado na Universidade do Estado do Amazonas – UEA sob orientação do Me. Charles Luiz da Silva Melo, como pré-requisito para obtenção do título de Engenheiro Mecatrônico.

Aprovado em de de 2015.

BANCA EXAMINADORA

Me. Charles Luiz da Silva Melo
Orientador

Dr. Walter Andrés Vermehren Valenzuela
Professor

Dr. Israel Mazaira Morales
Professor

Me. Marcelo Albuquerque
Professor

Manaus
2015

AGRADECIMENTOS

Agradeço aos meus colegas, meus professores e minha família por terem ajudado na construção desse trabalho.

Agradeço ao Prof. Orientador Me. Charles Luiz da Silva Melo e ao Prof. Co-orientador André Dias de Lima Machado pela paciente e dedicada orientação, competência e amizade.

Aos professores componentes da banca examinadora Prof. Me. Marcelo Albuquerque e Prof. Dr. Israel Mazaira, pelas importantes observações apresentadas.

Agradecimentos especiais são direcionados aos voluntários do GRAEST que contribuíram e que ainda contribuirão para a evolução dos meus estudos e pesquisa.

RESUMO

São denominados multirobôs o conjunto de dois ou mais agentes autônomos que cooperam entre si. Nos Sistemas Multi-Robô (MRS), os robôs buscam alcançar um objetivo específico. A essência do MRS reside na cooperação do time de robôs. O uso de MRS pode ser encontrados em várias áreas como, regaste em ambientes nocivos, distribuição de carga, transporte paralelos e simultâneo.

Os estudos sobre Localização e Mapeamento Simultâneo (SLAM) com robô individual e o seu sucesso aumentaram o interesse no estudo da Localização e Mapeamento Simultâneos com Múltiplos Robôs (MRSLAM). Isto porquê um time de robôs pode concluir uma tarefa, como explorar um ambiente, de forma mais eficiente e prática. Entretanto, MRSLAM possui muitos desafios, incluindos os de SLAM, tais como, união de mapas, escalabilidade, entre outros. No estudos sobre SLAM a utilização do Filtro de Partículas Rao-BlackWellized (RBPF) tem se mostrado bastante eficaz. Esses estudos, no entanto, geralmente possuem limitações como, a largura de banda na comunicação e o pré-conhecimento da posicão relativa entre os robôs.

A proposta deste trabalho é mostrar um algoritmo para MRSLAM com Sistema Operacional Robôtico (ROS). A base será o filtro RBPF para SLAM individual descrito por Hähnel, com três generalizações para multi-robôs. Na primeira, o filtro é utilizado em MRSLAM no qual a posição inicial dos robôs é conhecida. Depois, o filtro é abordado para o problema em que a posição inicial dos robôs é desconhecida. Nessa abordagem, supõe-se que aconteça um eventual encontro entre os robôs. Por último, o método anterior será extendido para combinar as informações obtidas antes do primeiro encontro dos robôs, adicionando o conceito de robôs virtuais que se movem retrocedendo no tempo. Este conceito permitirá a união da informação de todos os robôs em um mapa universal bidimensional.

Palavras-chave: algoritmo. SLAM. multi-robôs. filtro de partículas. mapa. ROS.

ABSTRACT

Multi robots are termed the set of two or more autonomous cooperating agents. In Multi-Robot Systems (MRS), the robots seek to achieve a specific goal. The essence of MRS lies in the cooperation of robots team. The use of MRS can be found in various areas, rescue in adverse environments, load distribution, parallel and simultaneous transport. Studies on Simultaneous Localization and Mapping (SLAM) single robot and its success increased interest in the study of Simultaneous Localization and Mapping with Multiple Robots (MRSLAM). A team of robot can complete a task, as explore an environment, more efficient and practical way. However, MRSLAM has many challenges, including SLAM problems, as union maps, scalability, among others. In studies of SLAM using the Rao-BlackWellized Particulate Filter (RBPF) has been effective. These studies, however, generally have limitations, as communication bandwidth and unknow robots relative pose. The purpose of this paper is demonstrate a algorithm to MRSLAM with Robot Operational System (ROS). The basis will be the filter RBPF for singlerobots described by Hähnel, with three generalization, First, the filter is used to MRS which the initial pose of robots is known. Next, the filter is discussed for the problem in which the initial pose of robots is unknown. This approach, is supposed that occur a eventual encounter between the robots. Finally, The previous method will be extended to combine informations obtained after and before the first encounter between robots, adding concept of virtual robots moving backwards in time.This idea allows combine the informations of all robots into a bidimensional commun map.

Key words: algorithm. SLAM. multi-robot. particule filter. common map. ROS.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Localização e mapeamento simultâneo (SLAM) é um problema de dificuldade elevada com grande número de soluções satisfatórias (LAKEMEYER; NEBEL, 2003). A maior parte dessas soluções tem como navegador um robô individual em um ambiente estático, utilizando meios de observação como a dispersão de pontos característicos 2D/3D ou os dados de sensores como, laser telêmetro de densidade 2D. Abordaremos a problemática da SLAM para múltiplos robôs (MRSLAM), levando em consideração que múltiplos robôs podem completar o mapeamento e a exploração de forma mais rápida que um único robô individual.

Existem inúmeras formulações para MRSLAM,(FENWICK; NEWMAN; LEONARD, 2002) utiliza uma abordagem "clássica", generalizando a formulação do filtro de Kalman extendido (EKF) no SLAM individual descrito por (SMITH; SELF; CHEESEMAN, 1990) e (DISSANAYAKE et al., 2001) para múltiplos robôs. (THRUN et al., 2003) utilizaram o filtro de informação com dispersão extendida para SLAM individual e generalizaram para múltiplos robôs. A vantagem do filtro de informação em relação ao EKF, incluem fácil implementação. (THRUN et al., 2001) descreve um algoritmo que mistura probabilidade de mapeamento com a localização Monte-Carlo (MCL), também conhecido como localização por filtro de partículas.

No trabalho de (THRUN et al., 2003), o principal desafio no mapeamento com múltiplos robôs está na especificação da posição inicial dos robôs, já que a maioria dos MRSLAM leva em consideração, para evitar esse tipo problema, que as referências (*landmarks*) são identificáveis, ou que os robôs iniciam de localizações próximas uma das outras. Thrun utiliza um método de combinação de referências anônimas tripla, para determinar as posições referenciais dos robôs.

Na proposta de (KO et al., 2003), cada robô cria seu próprio mapa universal, enquanto cálcula continuamente a posição do outro robô no mapa, então quando a posição estiver com uma alta probabilidade de determinação, os robôs executam *rendezvous* (ROY; DUDEK, 2001) para confirmar suas localizações. Alguns dos problemas sutís, como no caso em que os mapas universais estão incompletos, e não há garantia que um robô esteja sobre o mapa parcial de outro robô, há a necessidade de construir sensores para parte "desconhecida", assumindo pré-conceitos sobre o ambiente ou ainda por modelos de aprendizagem na coleta de experiência anteriores (STEWART et al., 2002).

(BIRK; CARPIN, 2006) propõem que cada robô construa um mapa de grades de ocupação (OGM) e depois mesclem os mapas utilizando procedimentos de pesquisa estocásticas. O algoritmo tem a vantagem de ser simples, e se baseia nas habilidades do algoritmo de SLAM individual para construir mapas com o mínimo de desvios ou distorções, ao mesmo tempo que cria hipóteses sobre o grau de auto-similaridade do ambiente. Neste

trabalho será desenvolvido um algoritmo para MRSLAM com base no filtro de partículas Rao-Blackwellized com três generalizações, afim de criar uma abordagem mais ampla sobre os problemas de SLAM, como a fusão dos mapas e posições relativas.

1.1 Formulação do Problema

Para um robô navegar dentro de um ambiente, um tarefa crucial é a construção do um mapa desse ambiente. Ao mesmo tempo, construir o mapa do ambiente e utilizá-lo para obter uma estimação da localização do robô. Este problema é normalmente chamado de Localização e Mapeamento Simultâneo (SLAM) e combina a estimação do posição do robô em relação ao mapa, com a construção do mapa utilizando os dados dos sensores e a posição relativa do robô. A maturidade das técnicas de SLAM individual foi reconhecida em meados de 2005. No entanto, a extensão dessas técnicas para múltiplos robôs, com objetivo de realizar tarefas cooperativas com SLAM em ambientes desconhecidos e/ou dinâmicos é um grande desafio.

1.2 Motivação

O fato de o MRSLAM ser uma área com evolução recente e muito ampla, podendo ser empregadas em situações de mapeamente de ambientes desconhecidos, procura e resgaste de sobreviventes em ambientes perigosos, sendo até utilizados como extensão da presença humana.

1.3 Justificativa

O problema normalmente indicado como Localização e Mapeamento Simultâneo para Múltiplos Robôs (MRSLAM). Em MRS a precisão com que cada robô pode modelar o ambiente tem grande impacto tanto na performance do time como individualmente. De um time de robôs é esperado uma representação consistente do ambiente de uma forma mais rápida que um robô individual. Embora MRS tenha a habilidade de melhorar a eficiência, precisão e robustez nas tarefas, ainda há problemas no MRSLAM, que necessitam de esforços adicionais, como estimar a posição de diferentes robôs, a fusão parcial de mapas para cada robô, que são etapas que não são necessárias no caso de SLAM individual. No MRSLAM, a dinâmica e a imprevisibilidade do ambiente bem como os ruídos das leituras dos sensores devem ser endereçadas. Em adição, ao contrário do SLAM individual, novos desafios e dificuldades são envolvidas, como coordenação dos robôs, integração das informações adquiridas de diferente robôs. um mapa coerente e a comunicação limitada.

Existe também, o fato de haver um aumento na necessidade de resposta efetivas para incidentes catastróficos e inesperados, tais como desatres naturais, acidentes industriais e

atos de terrorismo. Para estes fins, cooperação de robôs pode ser muito útil na assistência humana em muitas atividades, especialmente em cenários perigosos, extensão da percepção humana.

1.4 Objetivos

Neste subtópico serão apresentados o objetivo geral e os objetivos específicos do trabalho, focando-se nas tarefas principais do mesmo.

1.4.1 Objetivo Geral

Desenvolver um algoritmo SLAM para Múltiplos Robôs, utilizando o filtro de partículas RBPF juntamento com outras técnicas, auxiliados pelo Sistema Operacional Robôtico (ROS).

1.4.2 Objetivo Específico

- Apresentar SLAM, MRSLAM e também suas aplicações;
- Mostrar um algoritmo para MRSLAM;
- Aplicar o uso de técnicas do SLAM com auxílio do ROS;
- Simular um teste baseado nas técnicas que serão abordadas neste trabalho;

1.5 Organização do Trabalho

Este trabalho está organizado com os seguintes capítulos:

Capítulo 1 contém a introdução, que por sua vez apresenta os temas relacionados ao trabalho, objetivo, motivação e a metodologia do presente trabalho.

Capítulo 2 contém a fundamentação teórica do trabalho, que por sua vez apresenta uma abordagem do SLAM e as técnicas utilizadas para solucionar esse problema.

Capítulo 3 contém materiais utilizados para o desenvolvimento deste trabalho, descreve como foi feita a generalização do algoritmo de SLAM para MRSLAM e mostra os pacotes utilizadas do ROS para auxiliar na construção de um método para resolver MRSLAM.

Capítulo 4 contém a simulação feita para validar as técnicas empregadas neste trabalho e o resultado da mesma.

Capítulo 5 contém a conclusão e possíveis trabalhos futuros que podem ser abordados para melhorar esse trabalho.

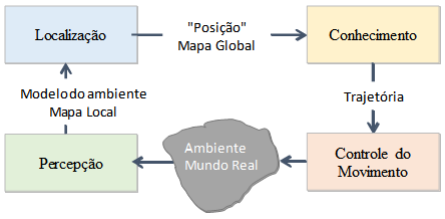
2 MRSLAM BREVE RESUMO

Neste capítulo, é apresentado um estudo de importantes problemas relacionados com MRSLAM. Serão mostrados alguns conceitos preliminares para ajudar nos problemas que serão discutidos em etapas posteriores. A definição e trabalhos relacionados ao SLAM e MRSLAM serão mostrados neste capítulo. No (THRUN; BURGARD; FOX, 2005) é apresentado um estado da arte mais extenso sobre SLAM.

2.1 Aproximações Clássicas do SLAM

Navegação é uma das habilidades com maior importância para a robôtica móvel. O sucesso da navegação depende dos resultados de quatro etapas: percepção, localização, conhecimento e controle do movimento como ilustrado na figura 1. O robô, primeiramente através dos sensores, faz o reconhecimento do ambiente "mundo real", e a partir desses dados cria um modelo desse ambiente, com essa mesma informação faz sua localização no ambiente, a posição no mapa modelo, com esse conhecimento ele descreve uma trajetória com os controles de movimento no ambiente real, fechando o loop, até a concretização da leitura total do mapa. Aliás, as quatro etapas consistem no processamento de informações eficientes dos sensores, determinando sua posição no ambiente, como interagir para atingir os resultados e comandando os atuadores externos para poder alcançar a trajetória planejada.

Figura 1 – representação do processo de navegação do robô móvel.



Fonte: Autor

No deslocamento, o robô estima sua posição baseado nas medições dos sensores, que em ambiente real possuem ruídos, e nos tipos de movimentos que podem ser envolvidos, como o deslizamento das rodas. A habilidade e a forma de interação com o ambiente depende da percepção. Os sensores permitem ao robô a habilidade de sentir o mundo, por isso sua forma de uso é muito importante quando desenvolvido no sistema autônomo. Sensores utilizados na Robôtica Móvel podem ser divididos em dois tipos os exteroceptivos e os proprioceptivos. Cada tipo de sensor será descrito em detalhes a seguir.

Sensores Exteroceptivos são aqueles que adquirem informação através da energia do ambiente. Dois tipos distintos de energia são utilizados: eletromagnética e acústica. Os sensores exteroceptivos são agrupados como ativos ou passivos, dependendo do fato da emissão ou não da energia no ambiente quando o sensoriamento é aplicado. Os sensores ativos emitem energia no ambiente obtendo medições da distância e da velocidade relativa pela tempo de reação da energia. As técnicas mais avançadas para medição de distância são baseadas na tempo de vôo, diferença de fase, modulação de frequência ou triangulação. Na medição da velocidade relativa é utilizado o Efeito Doppler. Os principais sensores de energia ativa utilizados na robôtica móvel são os LIDARs (*Light Detection and Rangings*), sonar, radar e os Sistemas de Navegação por Satélite (GNSS).

Sensores Proprioceptivos são aqueles que medem os parâmetros internos do sistema. Este tipo de sistema tem como vantagem o fato de não depender do ambiente para fazer as medições, sendo mais robustos. Sensores deste tipo geralmente utilizados na robôtica móvel são os de odometria e de inércia.

SLAM é definido como o problema da construção de um modelo de um mapa do ambiente, ou repetitivamente melhorando um mapa de um ambiente já existente, ao mesmo tempo que tenta localizar o robô neste mapa. Existem várias técnicas para compensar os erros, como o reconhecimento de características, a associação de dados, detecção de loops fechados. Utilizar filtros é o método mais utilizado na engenharia e em sistemas embarcados para compensar os erros. Um bom algoritmo de filtro pode reduzir os ruídos dos sinais enquanto aproveita as infomações úteis. Algumas das técnicas utilizadas no SLAM incluem como compensador de erros o filtro de Kalman, o filtro de partículas e os escaners de varredurra de faixa de dados (BONACCORSO; MUSCATO; BAGLIO, 2012).

O filtro de Kalman (KF) é um estimador linear recursivo que computa a mínima variância estimada para a condição do avanço no tempo, baseado nas observações lineares relacionados à condição. É possível obter resultados satisfatórios assumindo certas suposições lineares em relação ao ruídos do processo e das observações. O KF tem muitas aplicações nos problemas de navegação aeroespacial, de controle e também na robôtica em geral. É conhecido como um dos melhores estimadores para sistemas lineares com ruído do tipo Gaussiano. Para sistemas não-lineares o KF não é estritamente aplicado, a linearidade é importante para configurar o KF de forma eficiente. O Filtro de Kalman Extendido (EKF) é melhor atribuído para superar esta dificuldade, ele utiliza a linearização na estimação do estado. O EKF é usado para estimar a posição do robô utilizando dados da odometria e observações das referências espacias (*landmarks*). Esse filtro é geralmente descrito baseado na determinação da posição do robô por ele mesmo, assumindo a existênica de um mapa à priori. Quando inicialmente não há mapa, como ocorre no SLAM, as matrizes EKF são substituídas e uma nova solução é utilizada, denominada EKF-SLAM (WHYTE; BALIEY, 2006).

A solução do EKF-SLAM mostra diversas benefícios para o problema de localização e navegação, mas também possui limitações, tais como complexidade computacional, associação de dados e não-linearidade. Sobre as condições ideáis no EKF-SLAM, a covariância estimada da localização do robô e as posições individuais das referências (*landmarks*) podem ficar próximas de zero. No entanto, a complexidade computacional do estágio de correção cresce de forma quadrática com o aumento do número de *landmarks*, que geralmente é o problema em aplicações práticas do EKF-SLAM. Em adição, EKF-SLAM é extremamente sensível a referências incorrretas e a associação das observações. Além do mais, o problema da associação de observações se torna pior quando as referências são observadas novamente de muitos pontos de vista. E finalmente, a linearização dos modelos de movimento não-linear e dos modelos de observação podem levar a resultados com respostas inconsistentes. No (SMITH; SELF; CHEESEMAN, 1990) o EKF é apresentado como uma solução para situações de SLAM, o que o tornou uma abordagem clássica para resolver essa problemática. Muitos projetos são desenvolvidos basedos neste método, como (GUIVANT; NEBOT, 2001) e (DISSANAYAKE et al., 2001).

Devido as limitações do EKF nas complexidades quadráticas e a sensibilidade à falhas na associação de dados, surgiu como alternativa o algoritmo FastSLAM. Este algoritmo foi apresentado em (MONTEMERLO; THRUN, 2003) e (ROLLER et al., 2003) sendo o primeiro a considerar modelos não-lineares e distribuição multi-modal para resolver SLAM, com a vantagem de ser mais robusto na associação de dados.

O algoritmo de FastSLAM é baseado em uma característica muito importante na problemática do SLAM, que é a dependência condicional que existe entre duas *landmarks* diferentes colocadas no mapa, para fornecer a posição do robô. Em outras palavras, se a trajetória real do robô é conhecida, a estimativa de todas posições das *landmarks* no mapa pode ser feita de forma independente entre cada uma. Este fato permite a implementação de um filtro de partículas no SLAM, o Filtro de Partículas Rao-Blackwellized (RBPF). RBPFs foram introduzidos como meios efetivos para solucionar SLAM em (LEÓN et al., 2008),(CARLONE et al., 2010),(FENWICK; NEWMAN; LEONARD, 2002),(ZHANG; MENG; CHEN, 2009), onde cada partícula carrega um mapa individual do ambiente. O principal problema deste tipo método é sua complexidade computacional, medida com o número de partículas necessários para criar um mapa mais preciso. Na Tabela 1 são descritas as vantagens e desvantagens dos filtros apresentados nesta etapa.

Tabela 1 – Vantagens e Desvantagens entre EKF-SLAM e RBPF-SLAM.

Filtro	**Vantagens**	**Desvantagens**
EKF-SLAM	- Atualização linear das observações utilizando atualizações fracionadas; - Com o submapeamente é possível obter soluções constantemente;	- Assume um erro Gaussiano que nem sempre pode existir; - A linearização dos modelos não-lienares pode causar divergências;
RBPF-SLAM	- Simultâneamente mantém várias hipóteses; - Com um grande número de amostras é possível representar modelos não-lineares e não-Gaussianos com eficiência; - Permite associação de dados em paralelo; - Estima de forma online toda a trajetória do robô;	- O número de partículas aumenta com o tamanho da amostragem;

Fonte: Autor

2.2 Estendendo SLAM para Múltiplos Robôs

Ambos EKF e o RBPF são descritos para situações com MRSLAM. No entanto, outros algoritmos também são utilizados na literatura. O número de projetos que exploram a coodernação de uma equipe de robôs para completar tarefas vem crescendo em uma taxa muito rápida, o que é explicada pelas seguintes razões (ROCHA, 2006):

- O custo total de uma equipe de robôs simples pode ser menor que a de um único robô complexo;
- A habilidade de compartilhar informações entre os robôs permite uma redução das incertezas durante o processo de estimativa;
- Distribuição temporal - vários robôs podem fazer tarefas, sejam elas diferentes ou não, ao mesmo tempo;
- Distribuição espacial - vários robôs podem estar em diferentes lugares ao mesmo tempo;
- Decomposição do problema - Certos problemas são facilmente resolvidos, se divididos em pequenos problemas e direcionados a vários robôs;
- Confiança e robustez em relação ao erros;

Consequentemente, no intuito de generalizar SLAM para cooperação de multiplos robôs, é necessário responder as seguintes questões:

- Quantos mapas serão usados no processo?
- Qual é o algoritmo SLAM utilizado, como base, para a generalização de múltiplos robôs?
- Os robôs assumem o conhecimento prévio sobre suas posições iniciais?
- Qual é a metodologia utilizada para o alinhamento dos mapas?

Distribuição espacial e temporal contribuem para diminuir o tempo para completar subtarefas e missões, caso sejam utilizados vários robôs. Estas distribuições são relacionadas as operações que ocorrem simultaneamente no espaço e no tempo respectivamente. Estas características podem ser utilizadas em tarefas como exploração de grandes áreas ou detecção de alvos móveis, melhorando a perfomance nas tarefas como limpeza de determinadas áreas, ou missões de larga escala no mínimo tempo possível. Embora algumas tarefas não necessitem de soluções com múltiplos robôs, utilizar um robô simples simultaneamente com um complexo e robusto robô não é aconselhável devido a relação entre performance e confiança.

Soluções com múltiplos robôs oferecem maior flexibilidade durante gerenciamento na distribuição de riscos e complexidade. Se existe uma coincidência nas capacidades individuais de cada robô, o sistema será mais robusto no caso de haver alguma falha, não significando uma falha do sistema como um todo (ROCHA, 2006). MRS são, por estas razões, adequados para aplicações de procura e resgate e mapeamento do ambiente.

A principal dificuldade neste desafio consiste em achar uma estratégia eficiente com intuito de combinar informações adquiridas pelos sensores de todos os robôs. Para isso, o MRSLAM pode ser implementado de duas formas diferentes: considerando um único mapa global centralizado atualizado por cada robô, ou cada robô pode construir um mapa parcial do ambiente, que será combinado com todos os mapas parciais da equipe de robôs (ANDERSSON; NYGARDS, 2009) de forma distribuída.

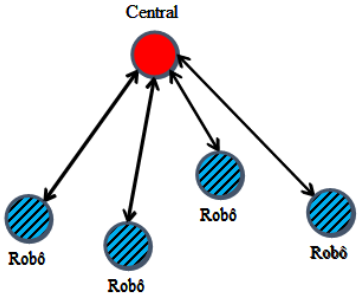
Relacionado ao processo de combinação na segunda opção, ela pode ser dividida em duas fases:

- Na primeira etapa, definida como alinhamento, a transformação das coordenadas é determinada com intuito de combinar a posição de um robô e suas respectivas *landmarks* em coordenadas referentes à outro robô.
- Na segunda etapa, as referências comuns precisam ser mescladas com intuito de gerar um mapa global. Esta etapa é chamada de Mesclagem dos Mapas.

A resposta para cada uma dessas questões e declarações permite classificar o MRSLAM em dois métodos diferentes. Em centralizada, os robôs enviam seus mapas parciais para um servidor central, que é responsável pelo alinhamento e mesclagem dos mapas; e então a central retorna o resultado para cada robô. Este tipo de sistema necessita de constante

monitoramento sobre as posições dos robôs e uma comunicação pesada de/para o servidor central. Por outro lado, os robôs podem trocar seus mapas locais quando se encontrarem no ambiente (*rendezvous*) e alinha-los internamente, está abordagem é conhecida como distribuída ou descentralizada. A representação do mapa global construído por cada robô pode variar, dependendo dos robôs que cada um encontrou e compartilhou mapa. Neste abpordagem descentralizadas, métodos de detecção mútuas são obrigatórios para relatar as posições dos robôs e auxiliar no alinhamento e mesclagem dos mapas locais. Propriedades de ambos os casos em MRSLAM serão abordadas a seguir.

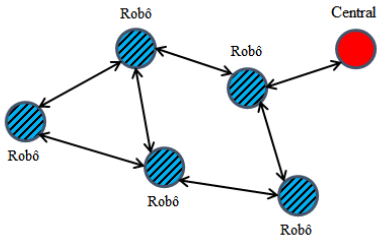
Figura 2 – Ditribuição centralizada.



Fonte: Autor

Arquiteturas centralizadas são caracterizadas por ter um único controlador que é individualmente responsável por tomar decisões. Assume-se que o processo principal tem um modelo global do mundo que permite teoricamente produzir ótimas soluções para problemas com múltiplos agentes. Duas classes diferentes de sistemas centralizados podem ser consideradas: Totalmente centralizada ou Parcialmente centralizada (ROCHA, 2006). Um sistema centralizado é intrissicamente coordenado e pode levar a soluções satisfatórias, coerentes e compreensivas, mas possui muitas limitações. Dependendo do tamanho da equipe, é muito difícil ou até mesmo impraticável manter um modelo global do ambiente em um único agente, baseado em observações locais e potencialmente inconsistentes entre os agentes locais. Esse tipo de sistema geralmente é mais custoso com base nos recursos e tempo, que fazem uso de grandes carregamentos e comunicação constante entre local e central, o que pode resultar em muitos congestionamentos na comunicação. Ela também tem uma arquitetura que pode ser insegura, pois toda a informação está localizada em um agente central, e que não é permitido haver falhas. Na figura 2 é ilustrado um esquema de sistema centralizado

Figura 3 – Distribuição Descentralizada



Fonte: Autor

Arquiteturas descentralizadas são feitas da rede de agentes independentimente tanto de forma lógica quanto física. Cada agente é capaz de “pensar" sobre os planos, escolhendo suas próprias ações e compreender a dinâmica do sistema a partir da interação com outros agentes. Esta abordagem pode ser classificada pelo nível de autonomia de cada agente, sendo Hierárquica ou Distribuída (ROCHA, 2006). As arquiteturas descentralizadas possuem muitas vantagens quando comparadas com as centralizadas, tais como tolerância a falhas, garantia de segurança, robustez, paralelismo, flexibilidade e escalabilidade. Sistemas baseados em controle distribuído e dados distribuídos tem performance superior em termos de recursos, comunicação e robustez quando comparados aos sistemas centralizados uma vez que não possuem controlador centralizado. Pode ser muito flexível já que a função de cada agente pode mudar baseado nas condições, como ilsutrado na figura 3. Aqui pode haver a comunicação entres os agentes sem que haja necessidade de passar pela central, teoricamente os agentes são independentes, mas necessitam da contribuição dos outros agentes para concluir a tarefa designada para o grupo.

Na tabela 2 podemos ver as vantagens e desvantagens de ambas as arquiteturas.

Tabela 2 – Vantagens e Desvantagens entre sistemas centralizados e descentralizados.

Sistema	**Vantagens**	**Desvantagens**
Centralizado	- Intrinsecamente coordenada; - Permite soluções satisfatórias, coerentes e abragentes - Informações globais podem conduzir a ótimos resultados;	- Soluções custosas em relação a tempo de recursos devido ao alto fluxo de comunicação entre os agentes locais e o central; - Tarefas de alta complexidade podem se tornar impraticáveis devido ao tamanho da área de procura; - Para um grande número de agentes pode tornar-se impraticável armazenar todas as informações do ambiente em um único servidor; - Requer informação sobre o posicionamento inicial dos robôs; - Não Confiável;
Descentralizado	- Agentes independentes dividem a tarefa de construir um mapa global; - Alta tolerância a erros e falhas; - Robustez, flexibilidade e escalabilidade; - Paralelismo nas tarefas; - Maiores garantias de sucesso;	- Alto grau de incerteza, que complica obter um comportamento global coerente do sistema; - Coordernação ineficiente do sistema resulta em dinâmicas complexas criando variações não-lineares e situações de caos; - Problema de detecção mútua;

Fonte: Autor

2.3 Trabalhos relacionados ao MRSLAM

No (THRUN et al., 2003), MRSLAM é resolvido utilizando Filtro de Informação de Espaço Extendido (SEIF) no qual o mapa e as posições dos robôs são representadas utilizando Campos Aleatórios de Markov. Este método descentralizado foca na atualização dos subgrupos de todas as *landmarks* no intuito de ganhar eficiência computacional. SEIF cria sub-mapas de forma dinâmica, por esta razão, os resultados são bastante precisos. O algoritmo evita problemas frequentes nas bordas dos sub-mapas onde a estimativa pode ficar instável. O algoritmo foi testado com sucesso utilizando oito robôs com dados coletados na Parque Victoria, Sydney.

No trabalho de (BURGARD et al., 2005), pontos alvos são escolhidos para cada robô com o objetivo de explorar diferentes áreas do mapa ao mesmo tempo. Neste método, que é centralizado, o custo do alcance e a utilidade do ponto alvo são calculadas. A

comunicação limitada dos robôs é considerada, e uma técnica de coordenação eficiente é descrita. É mostrada uma redução significante no tempo de execução das tarefas, quando comparado à outros métodos de exploração que não deixam explícito a coordenação dos robôs. Testes experimentais mostram que a performance do algoritmo é escalável com base na comunicação limitada.

O projeto de (CARLONE et al., 2010) produz um novo método descentralizado, que é baseado no RBPF (FastSLAM). Este método tem como base a comunicação limitada, levando em consideração a distância entre os robôs, e que a posição inicial dos robôs é desconhecida. O método utiliza câmeras para detecção mútua entre os robôs e só permite o alinhamneto parcial dos mapas quando acontece o *rendezvous* (encontro entre os robôs). Este método mostrou eficiência e robustez, construindo um mapa de um ambiente real com sucesso.

De acordo com (ANDERSSON; NYGARDS, 2009) o problema de alinhamento e a combinação de mapas criados por vários robôs utilizando observações feitas por eles são arquivados. A solução para este problema utiliza o Mapeamento e a Suavização Colaborativa (C-SAM) para combinar mapas criados por diferentes robôs em uma situação distribuída. A principal contribuição deste trabalho é o algoritmo utilizado para resolver o problema da associação de dados e eliminar observações falsas quando há o alinhamento do mapa no *rendezvous*. Este trabalho possui similaridades com (ZHOU; ROUMELIOTIS, 2006), só que este último utiliza EKF para resolver o problema da localização.

O trabalho de (FOX et al., 2006) apresenta um MRS distribuído para missões de exploração e mapeamento. O sistema permite a uma equipe de robôs explorar com eficiência um ambiente inciando de posições diferentes e desconhecidas. Com objetivo de assegurar a consistência do processo de mesclagem de dados, e para ocorrer o compartilhamento dos mapas os robôs ficam verificando ativamente suas posições relativas. Utilizando mapas compartilhados, os robôs coordenam suas estratégias de exploração com intuito de maximizar a eficiência da tarefa. Este sistema foi testado no simulador Saphira[1], sendo bastante eficiente e robusto.

No trabalho de (CHANG et al., 2007) é apresentado um algoritmo de MRSLAM descentralizado que utiliza mapas topológicos. Os vértices contém informação métricas locais e os cantos descrevem a posição relativa das mapas locais que estão próximos. Neste trabalho, os mapas são mesclados naturalmente entre os robôs através da inclusão de uma borda que estabelece a conexão entre os mapas topológicos, e a estimativa das posições relativas do robôs é feita pela otimização da borda. Os testes experimentais foram realizados com os robôs Pioneer 3-DX, que validaram a performance do algoritmo. A principal limitação deste método é a necessidade de otimizar a informação dos mapas topológicos offline, sendo um ponto crítico para ambientes extensos.

[1] Saphira Technical Manual. Available at: <http://www.cs.jhu.edu/~hager/Public/ICRAtutorial/Konolige-Salphira/saphira.pdf>

Tabela 3 – Características de cada algoritmo apresentado.

Algoritmo	**Características**
Bayesiana (THRUN; LIU, 2005)	- Utiliza filtro SEIF, onde a posição e mapas do agentes são representados por Campos aleatórios de Markov; - Permitir aumento na eficiência computacional; - A criação de mapas dinâmicos contribui para resultados mais precisos; - Previne problemas de estimação nas bordas do sub-mapa; - Testado com sucesso em ambiente real utilizando oito robôs;
Coordenação com comunicação restrita (BURGARD et al., 2005)	- Utiliza pontos alvos para explorar diferentes áreas no intuito de maximizar recursos e tempo de execução das tarefas; - Considera a associação de custo da faixa para cada ponto alvo, bem como sua importância para conclusão da tarefa; - Comunicação limitada entre agentes; - Técnica de coordenação mostra uma diminuição significativa no tempo de execução das tarefas; - Escalável, se levar em consideração a comunicação limitada;
RBPF (CARLONE et al., 2010)	- Utiliza filtro de partículas generalizado para o MRSLAM; - Solução para situações em que a posição inicial é desconhecida ou conhecida; - Mescla as observações dos robôs em um único mapa, quando *rendezvous* acontece; - Comunicação via rede Wireless; - Habilidade de detecção mútua dos robôs; - Representação do ambiente utilizando grades de ocupação; - Testado com sucesso em ambiente real utilizando quatro robôs homogêneos; - Latência durante troca de informações entre agentes, porém o resultado final apresenta eficiência e robustez;
C-SAM (ANDERSSON; NYGARDS, 2009)	- Algoritmo C-SAM; - Elimina observações falsas e resolve o problema dae associação de dados durante alinhamento do mapa; - Solução para situações em que a posição inicial é desconhecida ou conhecida; - Método mais suave para fusão de mapas criados a partir de diferentes robôs;
Distribuição mapas compartilhados (FOX et al., 2006)	- Método Distribuído; - Solução para situações em que a posição inicial é desconhecida ou conhecida; - Representação do ambiente utilizando mapas métricos compartilhados; - Utiliza uma estratégia para maximizar a eficiência da exploração; - Sistema eficiente e robusto; - Algoritmo testado na simulação no ambiente Saphira utilizando três robôs;
Método mapas Topológicos (CHANG et al., 2007)	- Representação do ambiente utilizando mapas topológicos; - Testado tanto em ambiente simulado com real utilizando três robôs 3-DX Pioneer; - Principal limitação é a necessidade de otimizar o mapa topológico offline;

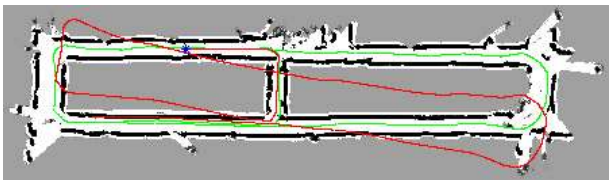
Fonte: Autor

2.4 Representação dos Mapas

A Construção dos mapas de ambientes desconhecidos cooperativamente é um dos campos de aplicação do MRS. Da mesma forma, que é importante para um robô autônomo móvel aprender e continuar aprendendo sobre modelos do ambiente. O principal problema do modelo de representação do mapa é lidar com a alta dimensão dos ambientes a serem mapeados (THRUN et al., 2002). Enquanto que, um mapa geométrico 2D muito detalhado pode necessitar de uma quantidade imensa de memória, uma descrição baseada em ambientes topológicos, tais como corredores, intersecção, salas e portas, pode não necessitar de muita memória para modelar o mesmo ambiente, mas também mostrará um mapa menos detalhado. No trabalho de (THRUN; BÜCKEN, 1996), são utilizados dois métodos para mapeamento de ambientes internos, são os mapas de grades de ocupação e os mapas topológicos.

Método de grades produzem mapas métricos mais precisos, são baseados em representações detalhadas, como uma planta baixa do ambiente. Este tipo de mapa exige uma localização precisa para o robô que devido a falta de eficiência dos sistemas odométricos tornam esta tarefa complexa. São fáceis de construir, mas podem ser difíceis de manter, dependendo da resolução e dimensões do ambiente. Em adição, o tamanho do ambiente, a configuração das células e a capacidade da memória tornam o planejamento da trajetória uma tarefa complicada. O exemplo mais comum de mapa métrico é a de grades de ocupação, uma representação do mapa onde cada célula da grade contém um valor de probabilidade, que indica se a localização referênciada é um espaço livre ou parte de um obstáculo (ELFES, 1990). É relativamente simples de implementar e possue uma natureza interativa, a figura 4 ilustra um exemplo de mapa de grades de ocupação.

Figura 4 – Exemplo de mapa métrico, mapa de grades de ocupação

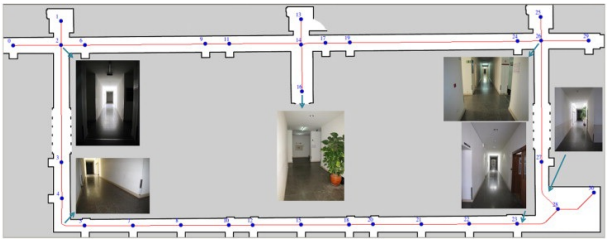


Fonte: (PORTUGAL; ROCHA, 2012)

Mapas topológicos, por outro lado, produzem mapas gráficos que podem ser utilizados de maneira mais eficiente. São mais simples, permitindo um planejamento eficiente e não exigem uma determinação precisa da posição do robô. Neste tipo de mapa, vértices correspondem a lugares importantes ou *landmarks*, que são conectadas por cantos que representam os caminhos entre eles (PORTUGAL; ROCHA, 2012). No entanto, a falta de precisão deste método torna difícil o reconhecimento e manutenção constante em ambientes de larga escala, particulamente se a informação dos sensores é ambígua, como exemplo é a dificuldade de reconhecer ambientes que são parecidos (ROCHA, 2006). Planejamento da

trajetória é simples desde que seja possível adaptar o algoritmo de procura por gráficos. A figura 5 ilustra um exemplo de mapa topológico e um resumo das caracteríscas de ambos os tipos de mapas são descritos na tabela 4

Figura 5 – Exemplo de mapa topológico



Fonte: (PORTUGAL; ROCHA, 2012)

Além disso, mesclagem de mapas consiste na construção do modelo de um ambiente com os dados de diferentes robôs. Com a posição inicial conhecida, bem como a posição relativa, mesclagem de mapas torna-se somente uma extensão do mapeamento com um único robô. Não conhecer as posições relativas dos robôs, torna o trabalho muito mais difícil, já que não fica claro como e onde os mapas parciais dos robôss podem ser alinhados.

Tabela 4 – Características das representação de mapa baseado em grades e topológicas

Tipo	**Características**
Grades de Ocupação	- Mapas métricos; - Fáceis de construir; - Fáceis para representar e manter; - Alto custo computacional; - Problemas de memória, com o aumento no tamanho dos mapas; - Reconhecimento do ambiente não é ambíguo;
Topológicos	- Mapas gráficos; - Eficiente para uso; - Mais simples que os mapas baseados em grades; - Não necessita da posição exata dos robôs; - Vértices correspondem às *landmarks*; - Vértices conectados por cantos representam o caminho entre eles; - Difícil de manter constantemente em ambientes de larga escala; - Reconhecimento do ambiente geralmente são ambíguos;

Fonte: Autor

2.5 Comunicação entre robôs

Outro assunto importante é a comunicação entre os robôs no MRS, por exemplo eles podem ser simples e ter comunicação limitada. A comunicação é crucial para a arquitetura do grupo, pois influenciam na interação entre eles. Comunicação na literatura é dividido em dois campos: implícita e explícita.

Comunicação implícita também conhecida como estigmergia é um método de comunicação através do ambiente, adequado em situações na qual a resposta em tempo real não é exigida, e é comum em socidades de insetos (exemplo é o feromônio nas trilhas das formigas). Por outro lado, ela é uma técnica de mensagem direta que pode ser aplicada quando o número de agentes móveis é pequeno e quando é necessário uma reação rápida (RYBSKI et al., 2007).

A comunicação entre robôs podem aumentar o raciocínio e percepção aumentando a eficiência das missões. Enquanto a maioria das pesquisas na comunicação para MRS tem sido devotados para desenvolver eixos que na maioria das vezes são relacionados com a estrutura da comunicação, também existem outras questões importantes sobre comunicação que devem ser respondidas:

- O que deve ser comunicado?
- Quando comunicar?

A idéia por trás dessas questões é evitar a comunicação de informações redundantes e então utilizar os recursos de comunicação de forma eficiente. Portanto, o MRSLAM proposto deve suportar tanto controle como dados distribuídos; permiter compartilhar dados dos sensores de forma eficiente entre um time robôs móveis cooperativos, tirando vantagem da cooperação entre robôs homogêneos para construir mapas em tempos menores do que os de um único robô ou até mesmo de uma equipe com um número menor de robôs, especialmente depois de melhorar a arquitetura com o mecanismo de coordenação.

Como a comunicação é sempre limitada, tanto nos recursos aplicados para perceber o mundo, quanto na largura de banda do canal de comunicação, utilizar de forma eficiente esses recursos é crucial para ampliar a arquitetura cooperativa para uma equipe com muitos robôs, sem limitá-los a simples sistemas cooperativos reativos e livres, limitados ou até sem consciência. Essas questões evitam comunicação de informação redundante e utilizam de formar eficiente os recursos da comunicação.

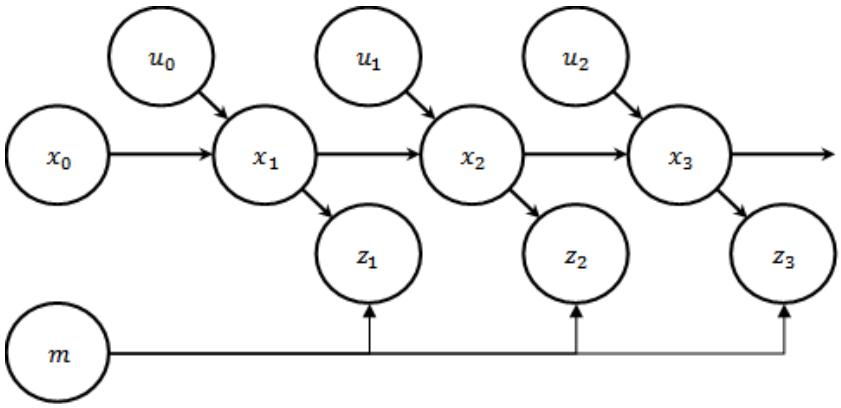
Agora que os problemas de localização, mapeamento, representação e comunicação limitada foram introduzidas, uma estratégia MRSALM que incorpora implicitamente esses assuntos foi implementada e é descrita com mais detalhes no próximo capítulo. Como representação, a abordagem métrica utilizada é mapas de grades de ocupação, a arquitetura de coordenação é distribuída, os robôs se comunicam quando estão próximos, mais detalhes serão discutidos no próximo capítulo.

3 IMPLEMENTAÇÃO DE SISTEMA SLAM PARA MULTI-ROBÔS

Primeiramente descreveremos um algoritmo para MRSLAM baseado no filtro RBPF e generalizado de três formas a partir do SLAM individual. São elas o MRSLAM com posição inicial conhecida, MRSLAM com posição inicial desconhecida e MRSLAM com posição inicial desconhecida com uso de robôs virtuais. Após isso, será abordado algumas características do ROS, que serão utilizadas como base para a implementação do algoritmo de forma simulada.

3.1 SLAM Robô Individual

Figura 6 – Rede Bayesiana para SLAM robô individual.



Fonte: Autor

A equação para SLAM robô individual é descrita como seguintes. $x_{1:t}$ corresponde a sequência de posições do robô nos tempos $1, 2, ..., t$, $z_{1:t}$ corresponde a sequência de observações dos sensores nos tempos $1, 2, ..., t$, $u_{0:t-1}$ corresponde a sequência de ações executadas pelo robô. O objetivo é calcular a probabilidade posterior indicada pela equação $p(x_{1:t}, m|z_{1:t}, u_{0:t-1}, x_0)$ através da trajetória do robô $x_{1:t}$ e o mapeamento m, informada a posição inicial x_0. Essa equação é descrito como o produto de dois termos:

$$\begin{aligned} p(x_{1:t}, m|z_{1:t}, u_{0:t-1}, x_0) &= p(m|x_{1:t}, z_{1:t}, u_{0:t-1}, x_0) \\ & \quad p(x_{1:t}|z_{1:t}, u_{0:t-1}, x_0) \end{aligned} \tag{3.1}$$

O primeiro termo é a distribuição sobre os possíveis mapas e o segundo é a distribuição sobre as possíveis trajetórias. Uma vez conhecido a trajetória do robô $x_{1:t}$ é possível calcular o mapeamento, que corresponde ao primeiro termo. Há a possibilidade de aproximar os

próximos caminhos e mapa, com o filtro de partículas no qual cada amostra representa um caminho completa do robô, e um mapa é condicionado para cada amostra. Utilizaremos o filtro de partículas Rao-Blackwellized (DOUCET et al., 2000), tomando como base as condições de dependência do SLAM individual (Figura 6). Para cada partícula (i) temos uma enupla correspondente $x_t^{(i)}, m_t^{(i)}, w_t^{(i)}$ na qual $x_t^{(i)},, m_t^{(i)}$ são respectivamente a posição e o mapa gerado no instante t, e $w_t^{(i)}$ é o peso da partícula. Dado as informações (as ações e observações), a etapa de atualização do filtro será descrita por:

$$\begin{aligned} x_t^{(i)} &= U(u_{t-1}, x_{t-1}^{(i)}) \\ m_t^{(i)} &= M(z_t, x_t^{(i)}) + m_{t-1}^{(i)} \\ w_t^{(i)} &= W(z_t, x_t^{(i)}, m_{t-1}^{(i)}) w_{t-1}^{(i)} \end{aligned} \tag{3.2}$$

Onde, U é um modelo de ação que retorna uma posição obtida da distribuição $p(x_t^{(i)}|x_{t-1}^{(i)}, u_{t-1})$; W é o modelo de sensor $p(z_t^{(i)}|x_{t-1}^{(i)}, m_{t-1}^{(i)})$; e M é um gerador de mapa incremental que retorna uma grade de ocupação parcial. Este algoritmo é muito simples para implementar uma vez que U e W são idênticos aos utilizados no algoritmo de localização Monte-Carlo (THRUN et al., 2001) e M é um algoritmo básico de traçador de raios. Uma grande limitação desse algoritmo, é quando o ambiente é extremamente extenso, e o número de partículas é inevitavelmente pequeno, fazendo com que nenhuma partícula gere um mapeamento auto-consistente. Para reduzir esse erro, será adotado uma combinação de odometria com laser para "estabilizar"posições informadas pela odometria.

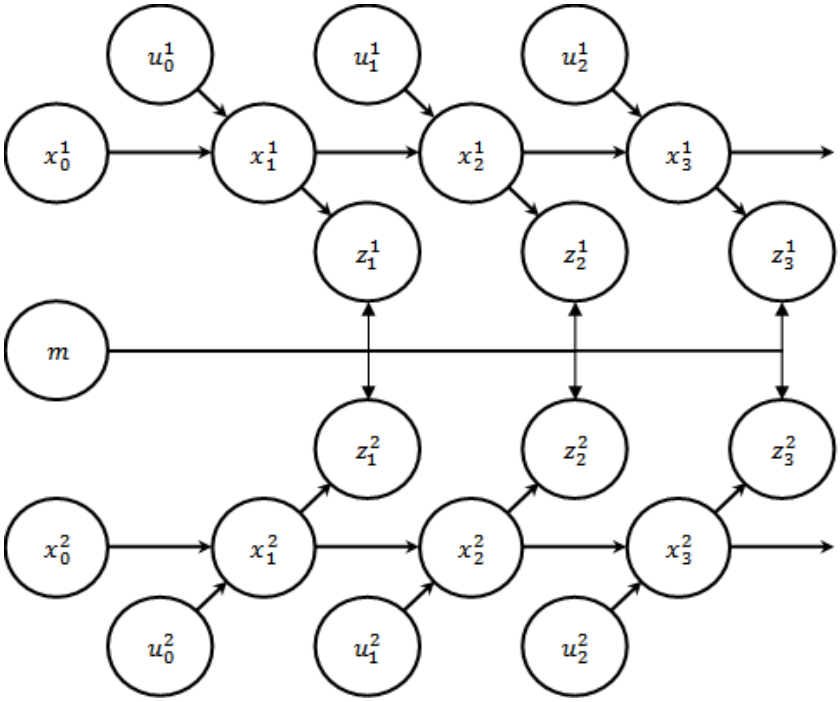
3.2 MRSLAM com Posicão Inicial Fornecida

Foi generalizado, utilizando lógica direta, do modelo SLAM individual para utilização com múltiplos robôs, dado que a posicão inicial dos robôs são conhecidas. Considerando dois robôs com observações no tempo $1, 2, ..., t$; $x_{1:t}^1$ corresponde ao caminho do robô 1, e $x_{1:t}^2$ corresponde ao caminho do robô 2. O objetivo dessa equação é estimar a distribuição de probabilidade das próximas trajetórias dos robôs e o mapa simultâneamente:

$$\begin{aligned} p(x_{1:t}^1, x_{1:t}^2, m|z_{1:t}^1, u_{0:t-1}^1, x_0^1, z_{1:t}^2, u_{0:t-1}^2, x_0^2) &= p(m|x_{1:t}^1, z_{1:t}^1, x_{1:t}^1, z_{1:t}^2) \\ & \quad p(x_{1:t}^1|z_{1:t}^1, u_{0:t-1}^1, x_0^1) \\ & \quad p(x_{1:t}^2|z_{1:t}^2, u_{0:t-1}^2, x_0^2) \end{aligned} \tag{3.3}$$

O primeiro termo mostra a distribuição dos mapas, e os restantes a distribuição das possíveis trajetórias dos robôs. Essa equação assume que as trajetórias dos robôs são independentes, e as informações gravadas por um não dependem da posição do outro (Figura 7). Para construção do filtro de partículas, cada partícula possue as propriedades $(x_{1:t}^{1(i)}, x_{1:t}^{2(i)}, m_t^{(i)}, w_t^{(i)})$ onde $x_{1:t}^{1(i)}$ e $x_{1:t}^{2(i)}$ são as posições instantâneas dos robôs, $m_t^{(i)}$ é o mapa universal e $w_t^{(i)}$ é o peso da partícula. Dada as informações $(z_{1:t}^1, u_{0:t-1}^1, z_{1:t}^2, u_{0:t-1}^2)$,

Figura 7 – Rede Bayesiana para MRSLAM com posições iniciais fornecidas.



Fonte: Autor

o passo de atualização para o filtro será descrito por:

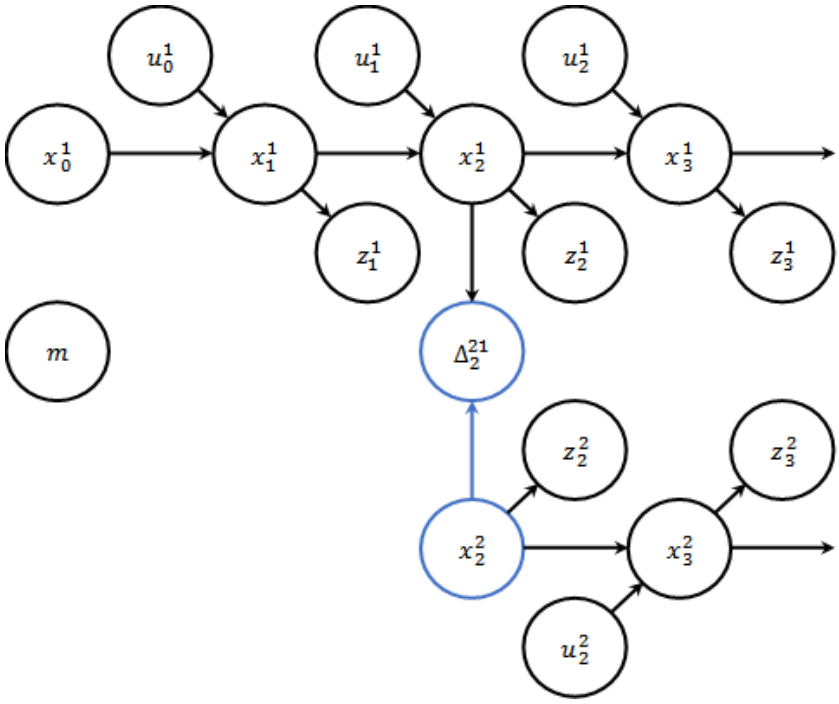
$$\begin{aligned} x_t^{1(i)} &= U(u_{t-1}^1, x_{t-1}^{1(i)}) \\ x_t^{2(i)} &= U(u_{t-1}^2, x_{t-1}^{2(i)}) \\ m_t^{(i)} &= M(z_t^1, x_t^{1(i)}) + M(z_t^2, x_t^{2(i)}) + m_{t-1}^{(i)} \\ w_t^{(i)} &= W(z_t^1, x_t^{1(i)}, m_{t-1}^{(i)}) W(z_t^2, x_t^{2(i)}, m_{t-1}^{(i)}) w_{t-1}^{(i)} \end{aligned} \tag{3.4}$$

onde U, M e W são os mesmos do SLAM individual. Esse equação pode ser usado para qualquer número de robôs. Aqui existem duas limitações principais. A primeira é o tamanho do ambiente que precisa ser maior do que o visto no SLAM individual. O outro é o processo de reamostragem do filtro no caso de um robô parar e o outro continuar se movendo, podendo levar a uma divergência no filtro.

3.3 MRSLAM com Posicão Inicial Desconhecida

Para robôs muito separados, determinar suas posicões iniciais pode ser impraticável. Para esses casos, será adotado o conceito de encontro para determinar as posições relativas dos robôs. O mapeamento inicial será feito por um robô (que iniciará de uma posição arbitrária) e aguardamos o primeiro encontro com cada robô adicional, somente será considerado o primeiro encontro. Δ_s^{21} corresponde à posição referencial do robô 2 indicada

Figura 8 – Rede Bayesiana para MRSLAM com posição inicial desconhecida, interações entre o mapa m e as observações $z_1^1, z_2^2, ...$ Foram omitidas para melhor clareza.



Fonte: Autor

pelo robô 1 no tempo s. A fatoração será dada por:

$$\begin{aligned} p(x_{1:t}^1, x_{s:t}^2, m | z_{1:t}^1, u_{0:t-1}^1, x_0^1, z_{s:t}^2, u_{s:t-1}^2, \Delta_s^{21}) &= p(m | x_{1:t}^1, z_{1:t}^1 x_{s:t}^2, z_{s:t}^2) \\ & \quad p(x_{1:t}^1 | z_{1:t}^1, u_{0:t-1}^1, x_0^1) \\ & \quad p(x_{s:t}^2 | z_{s:t}^2, u_{s:t-1}^2, x_s^1, \Delta_s^{21}) \end{aligned} \tag{3.5}$$

Serão feitas duas aproximações, iremos ignorar a dependência entre as trajetórias dos robôs, e trataremos como variáveis independentes, e que a incerteza na posição referencial Δ_s^{21} é pequena. Com essas aproximações o filtro é idêntico ao descrito na seção anterior, mas com o fato de inicializarmos o robô 2 na posição $x_s^{2(i)}$ no tempo s usando a posição Δ_s^{21}:

$$x_s^{2(i)} = \Delta_s^{21} \oplus x_s^{1(s)} \tag{3.6}$$

onde $\oplus$, indica uma transformada de coordenada 2D. O filtro irá unir todos os dados do robô 1, e todos os dados do robô 2 no intervalo em que $t \geq s$.

3.4 Algoritmo MRSLAM

Figura 9 ilustra o algoritmo para SLAM multi-robôs. Como entendimento, considere um grupo com três robôs postos em localizações distantes e desconhecidas no ambiente.

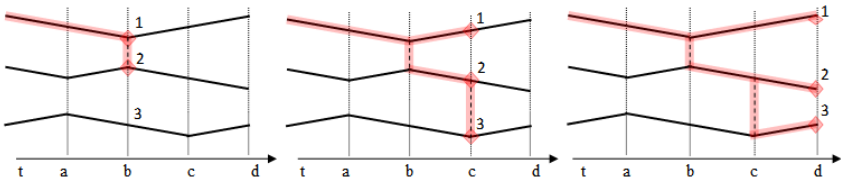
Figura 9 – Algoritmo SLAM para multi-robôs (portugol).

```
        # processamento da informação do robô individual
        # r: identificação do robô gerador de dados
        # (u, z): mudança da posição (odometria) e do sensor (laser)
        # r': identificação do robô observado
        # Δ: posição relativa do robô observado ( se houver)
0       definir atualizador (r, u, z, r', Δ)

        # Se o robô foi adicionado ao mapa...
1       se r está combinado
          # Faz atualização do sensor
2           atualizar_filtro(r, u, z)
          # Verifica encontros
3           se r e r' não está combinado
                # Inicia o filtro para o novo robô
4                   iniciar_filtro(r, r', Δ)
                # Adiciona robô para a lista combinado
5                   anexar(combinado,r')
6       retorna
```

Fonte: Autor

Assume-se que a trasmissão das informações entre os robôs é uma conexão wireless que possua mínima ou nenhuma interferência, e eles consigam fazer reconhecimento mútuo e determinar posição relativa (para robôs nas proximidades e dentro do raio de 'visão'). Os robôs exploram individualmente o ambiente até um eventual encontro, figura 10. Considere

Figura 10 – Diagrama de eventos para um experimento com três robôs imaginários.



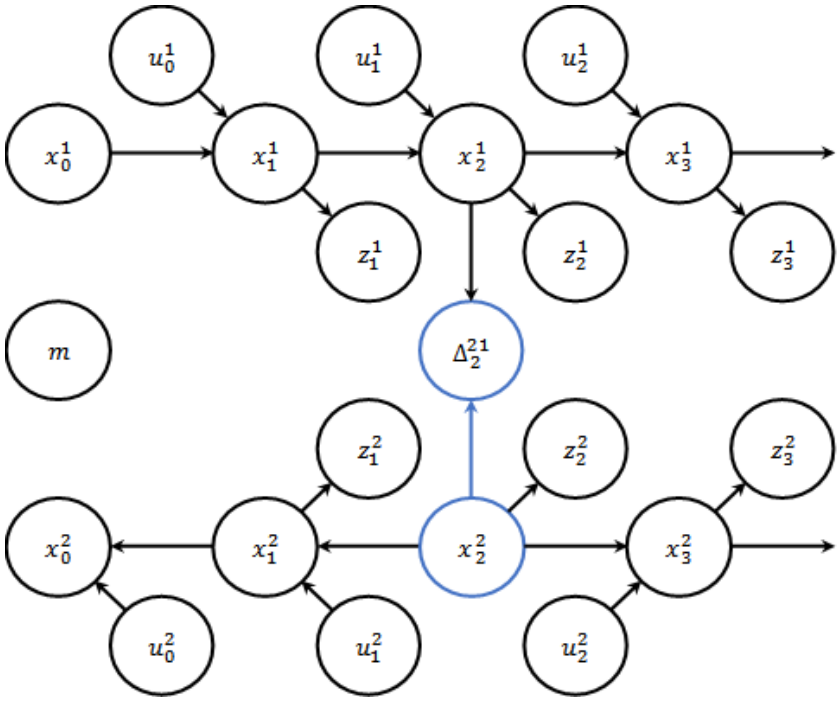
Fonte: Autor

que o robô 1, desconhecendo a posição inicial dos outros robôs, efetue o SLAM individual. Seguindo as regras do algoritmo, o robô se adiciona ao conjunto combinado e atualiza o filtro com suas próprias informações (linha 2). No instante em que o robô 1 encontro o robô 2 (indicado por $t = b$) e definem suas posições referênciais, é incrementado ao filtro de partícula uma nova variável x^2 que representa a posição do robô 2, está variável é inicializada no filtro de acordo com a Equação 3.8 (linha 4). Essa etapa ignora qualquer incerteza (supondo ser insignificante) relacionada ao cálculo da posição referencial. As medições seguintes de ambos os robôs serão adicionadas ao mapa. No instante em que o robô 2 encontro o robô 3 (indicado por $t = c$) é incrementado ao filtro de partícula uma nova variável de estado x^3 que representa a posição do robô 3, está variável é inicializada

com valores, conforme descrito anteriormente. As medições seguintes dos robôs serão adicionadas ao mapa.

3.5 MRSLAM com Robôs Virtuais

Figura 11 – Rede Bayesiana para MRSLAM posição inicial desconhecida e atualizações em tempo-reverso. interações entre o mapa m e as observações $z_1^1, z_1^2, ...$ Foram omitidas para melhor clareza.



Fonte: Autor

Utilizaremos como base a equação da seção 3.3, e passaremos a etapa na qual as informações do robô 2 são adquiridas, isso antes do primeiro encontro com o robô 1. Para isso dividiremos o caminho do robô em duas partes $x_{1:s-t}^2$ e $x_{s:t}^2$, que indicam respectivamente as trajetórias antes e depois do encontro no tempo s. A nova equação da probabilidade será dada por:

$$\begin{aligned} p(x_{1:t}^1, x_{1:t}^2, m | z_{1:t}^1, u_{0:t-1}^1, x_0^1, z_{1:t}^2, u_{s:t-1}^2, \Delta_s^{21}) \quad = \quad & p(m | x_{1:t}^1, z_{1:t}^1, x_{1:s-1}^2, z_{1:s-1}^2, x_{s:t}^2, z_{s:t}^2) \\ & p(x_{1:t}^1 | z_{1:t}^1, u_{0:t-1}^1, x_0^1) \\ & p(x_{1:s-1}^2 | z_{1:s-1}^2, u_{0:s-1}^2, x_s^1, \Delta_s^{21}) \\ & p(x_{s:t}^2 | z_{s:t}^2, u_{s:t-1}^2, x_s^1, \Delta_s^{21}) \end{aligned} \tag{3.7}$$

Nesta equação do filtro, o robô 2 possui dois casos, o eventual que corresponde ao avanço do robô, e o não eventual correspondente ao movimento retrocedente no tempo do robô. Cada partícula é indicada por $x_t^{1(i)}$,$x_t^{2(i)}$,$\bar{x}_t^{2(i)}$,$m_t^{(i)}$,$w_t^{(i)}$, onde $x_t^{2(i)}$ é a posição do robô 2 no

instante t, e $\bar{x}_t^{2(i)}$ é posição do robô 2 (virtual) no instante $2s-t$. As posições são inciadas no primeiro encontro pela equação:

$$x_s^{2(i)} = \Delta_s^{21} \oplus x_s^{1(s)} \quad e \quad \bar{x}_s^{2(i)} = \Delta_s^{21} \oplus x_s^{1(s)} \tag{3.8}$$

É acrescentado a informação correspondente ao movimento de retrocesso no tempo ao filtro, indicada por $(\bar{z}_t^2, \bar{u}_t^2) = (z_{2s-t}^2, u_{2s-t}^2)$, então:

$$\begin{aligned}
x_t^{1(i)} &= U(u_{t-1}^1, x_{t-1}^{1(i)}) \\
x_t^{2(i)} &= U(u_{t-1}^2, x_{t-1}^{2(i)}) \\
\bar{x}_t^{2(i)} &= \bar{U}(\bar{u}_{t-1}^2, \bar{x}_{t-1}^{2(i)}) \\
m_t^{(i)} &= M(z_t^1, x_t^{1(i)}) + M(z_t^2, x_t^{2(i)}) + M(\bar{z}_t^2, \bar{x}_t^{2(i)}) + m_{t-1}^{(i)} \\
w_t^{(i)} &= W(z_t^1, x_t^{1(i)}, m_{t-1}^{(i)}) W(z_t^2, x_t^{2(i)}, m_{t-1}^{(i)}) W(\bar{z}_t^2, \bar{x}_t^{2(i)}, m_{t-1}^{(i)}) w_{t-1}^{(i)}
\end{aligned} \tag{3.9}$$

aqui somente foi adicionado um novo modelo $\bar{U}$, designado para o termo não eventual, o filtro funciona de forma que os dois termos o eventual e não eventual sejam atualizados com mesmo ritmo.

3.6 Algoritmo MRSLAM com Robôs Virtuais

Figura 12, ilustra o algoritmo para múltiplos robôs com robôs virtuais. Como entendimento, considere o mesmo grupo de robôs da seção 3.3. Figura 13, ilustra o diagrama de encontros para os instantes $t = a$ e $t = b$. Faz-se as mesmas considerações da seção 3.3 para o robô 1, mas agora suas informações são adicionadas a uma *lista* (linha 1), e depois removidas, após a atualização do filtro (linhas 4-5). Informações dos outros robôs são adicionadas em *lista* individuais (linha 1), bem como qualquer encontro que tenha ocorrido entre os robô 2 e 3. No instante em o robô 1 encontro o robô 2 (indicado por $t = b$) e definem suas posições referênciais, são incrementadas ao filtro de partícula duas novas variáveis de estado $x^2, \bar{x}^2$ que representa as posições eventual e não eventual do robô 2, essas posições são iniciadas no filtro com valores de acordo com a Equação 3.8 (linhas 7-8). Essa etapa ignora qualquer incerteza (supondo ser insignificante) relacionada ao cálculo da posição referencial. As informações da lista para o robô 2 são divididos em duas partes (linhas 9-10); uma lista eventual com as informações obtidas depois do encontro no instante $t = b$ (lista vazia), e uma lista não eventual com todas as informações obtidas antes do instante $t = b$ (toda informação até esse instante). Para as chamadas seguintes de atualização da função, a lista eventual, é utilizada para atualizar as informações da instância x^2 (linhas 4-5), e a lista não eventual as informações da instância $\bar{x}^2$ (linhas 13-14). No instante $t = c$, há um encontro entre o robô 2 e 3, indicado pela lista não eventual do robô 2 (encontro que ocorreu no passado no instante $t = a$). São incrementados ao filtro de partícula duas novas variáveis de estado do robô 3 $x^3, \bar{x}^3$, e suas posições são

Figura 12 – Algoritmo MRSLAM com robôs virtuais (portugol).

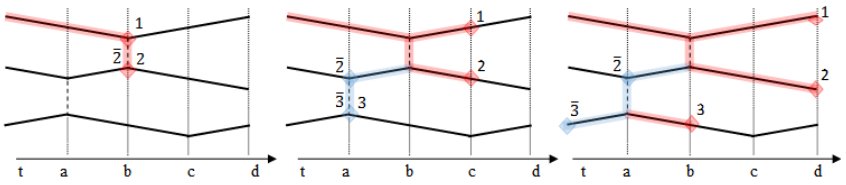
```
        # processamento da informação do robô individual
        # r: identificação do robô gerador de dados
        # (u, z): mudança da posição (odometria) e do sensor (laser)
        # r': identificação do robô observado
        # Δ: posição relativa do robô observado ( se houver)
0       definir atualizador (r, u, z, r', Δ):

        # Adiciona dados a lista
1       anexar(lista[r], (u, z, r', Δ))

        # Se o robô foi adicionado ao mapa...
2       se r está combinado
3           se tamanho(lista[r]) > 0;
          # Faz eventuais atualizações
4                   (t, u, z, r', Δ) = remover(lista[r])
5                   atualizar_filtro(r, u, z)
              # Verifica eventuais encontros
6                   se r e r' não está combinado
                    # Inicia o filtro para causas eventuais e não eventuais
7                           iniciar_filtro(r, r', Δ)
8                           iniciar_filtro(r, r̄', Δ)
                    # Divide a fila em componentes eventuais e não eventuais
9                           inversa[r'] = inverso(lista[r'][0: t − 1])
10                          lista[r'] = lista[r'][t + 1:]
                    # Adiciona robô para a lista combinado
11                          anexar(combinado,r')
12          se tamanho(inversa[r]) > 0;
              # Faz eventuais atualizações
13                  (t, u, z, r', Δ) = remover(inversa[r])
14                  atualizar_filtro(r̄, −u, z)
              # Verifica eventuais encontros
15                  se r e r' não está combinado
                    # Inicia o filtro para causas eventuais e não eventuais
16                          iniciar_filtro(r̄, r', Δ)
17                          iniciar_filtro(r̄, r̄', Δ)
                    # Divide a fila em componentes eventuais e não eventuais
18                          inversa[r'] = inverso(lista[r'][0: t − 1])
19                          lista[r'] = lista[r'][t + 1:]
                    # Adiciona robô para a lista combinado
20                          anexar(combinado,r')
21      retorna
```

Fonte: Autor

iniciadas no filtro utilizando a instância não eventual do robô 2 (linhas 16-17). A lista para robô 3 é dividida em dois componentes o eventual e o não eventual (linhas 18-19), e as atualizações posteriores dos filtros unem as informações de ambas as listas ao mapa.

..

Figura 13 – Diagrama de eventos para um experimento com três robôs imaginários com atualizações de tempo-reverso.



Fonte: Autor

3.7 MRS em ROS

Neste trabalho, *Robot Operation System* (ROS) é utilizado como desenvolvimento para o framework. ROS é um framework para desenvolvimento de softwares de robôs e é mantido por Willow Garage [1]. Esse sistema utiliza uma arquitetura de publicação/assinatura e possui uma instância central mestre conhecida como *roscore*. O *roscore* é uma coleção de *nodes* e programas que são pré-requisítos para o sistema ROS. Cada *node* reporta sua informação de registro para o *roscore*. Um *node* assina um *topic e* requisita a informação de conexão ao *roscore*, que conecta diretamente (via P2P) ao *node* publicador, que receberá as mensagens do mesmo. A interpretação e a execução de algoritmos são controlados por estes *nodes* no ROS, que podem ser programados em C++, Python, Java ou Lisp. Portanto, *nodes* customizados podem ser desenvolvidos intuitivamente e com alto nível de abstração. ROS contém uma variedade de ferramentas e algoritmos que podem ser utilizados nos robôs (QUIGLEY et al., 2009).

O uso de comunicação sem fio em ROS apresenta um desafio, e o foco da maioria das soluções é para dar suporte à múltiplos mestres. Para isto utilizaremos um pacote chamado *wifi_comm*[2] que é utilizado para publicação e assinatura de *topics* de/para mestres “externos". Isto é possível pela abertura de uma conexão entre as máquinas com o Protocolo da Internet (IP) que compartilha um *topic* comum e retransmite informações específicas de um *topic* local através de uma conexão disponível.

Outro desafio a se superar quando se tratando de multiplos robôs é que cada robô deve possuir um *namespace* diferente, para evitar interpretações erradas das informações transmitidas entre os robôs. Para lidar com isto, quando as mensagens são passadas entre os robôs com nome similar é necessário ser aplicado um remapeamento do *frame_id*, ambos para transformar os dados, quanto para utilizar uma convenção comum de nomeação. O uso do parâmetro *tf_prefix* é uma forma padronizada e recomendada no uso de múltiplos robôs, quando compartilham transformações de estruturas das referências. Adicionalmente, pode precisar remapear os *topics* de cada robô individualmente, por exemplo no caso onde diferentes robôs utilizam o mesmo *node* internamente: a saída do sensor *Laser Range*

[1] Site do Willow Garage:<http://willowgarage.com/>
[2] pacote *wifi_comm*:<http://www.ros.org/wiki/wifi_comm>

Finder (LFD) do robô 1 pode não ser publicada no mesmo *topic* que a saída do LRF de um robô 2. Cada robô deve publicar a mensagem dos sensores em diferentes e apropriados *topic*. Definindo um *prefix* permitirá que cada robô utilize um único *namespace* para todos os seus dados e transformadas, este *prefix* irá mudar coisas como referências de odometria do robô (geralmente nomeadas "/odom") em "r1/odom" e "r2/odom" automaticamente, dependendo do robô. ROS suporta replicação de funcionalidades pela permissão de *nodes* e arquivos de descrição de cluster inteiros de *roslaunch* para serem movidos em um *namespace* filho, que assegura que não haverá duplicação dos nomes.

3.8 SLAM em ROS

Iremos dar o primeiro passo para desenvolver uma solução para MRSLAM. Para isto, utilizaremos um algortimo de teste em cada robô. De todos os existentes no ROS *Gmapping*[3] é o pacote mais estável e geralmente utilizado com resultados já reconhecidos, tendo suporte e atualizações periódicas pela comunidade ROS. Este pacote provê um SLAM baseado em laser com um *node* chamado *slam_gmapping*, com esse *node* podemos criar uma mapa 2D por grades de ocupação a partir dos dados coletados pelo LRF. O *slam_gmapping* transforma cada entrada do escaner em um quadro odometrico do robô. O mapa pode ser visto no visulizador ROS: *rviz*[4].

3.9 Descoberta e o distanciamento dos companheiros de equipe

Primeiro assume-se que cada robô possue um ID e endereço de IP próprio. De início, cada robô carrega uma tabela com informações pré-definidadas de todos os robôs integrantes da equipe, nomeando seu IP e ID. Utilizando as características do protocolo OLSR adhoc, uma lista de integrantes da equipe ativos é mantida por toda a missão, utilizando o OLSR deamon (OLSRD)[5]. O OLSRD é utilizado neste trabalho para prover endereços IPs que podem ser alcançados para uma dada máquina, bem como para medir a qualidade da conexão entre elas. A qualidade da conexão é uma métrica que pode ser computada, dependendo da força do sinal e da taxa de pacotes perdidos. Esta informação é diponibilizada no ROS, utilizando *wifi_comm*.

3.10 Topics Dinâmicos, Rendezvous e Troca de Mapas

Como esta é uma abordagem de MRS distribuído e genérico, onde o número de robôs é variável, não é possível definir todos os canais de comunicação no início da missão.

[3] Pacote ROS Gmapping. Disponível no: <www.ros.org/wiki/gmapping>
[4] Visualizador ROS, Disponível em: <www.ros.org'wiki/rviz>
[5] Página do OLSRD:<http://www.olsr.org/>

Portanto o sistema deverá ter suporte a *topics* dinâmicos que serão criados no decorrer da missão. Na maioria dos casos, o usuário define *topics* estáticos para cada *node* sempre que o assinante ou o publicador são iniciados.

Topics dinâmicos são criados de forma similar aos *topics* estáticos. Entretanto, eles somente são definidos quando o robô gera uma tabela de descoberta. Sua nomeação através de manipulação de cadeia é bastante simples, uma vez que os IDs dos robôs são conhecidos. Um *rendezvous* é um encontro entre dois ou mais agentes em um lugar e tempo designado. O problema do *rendezvous* é achado em todo lugar na natureza. MRS também têm uma necessidade inerente para a capacidade de *rendezvous* inter-agente. A habilidade para encontrar localizações facilitadas, permite a exploração colaborativa do mapa e uma comunicação confiável, já que a maioria dos hardwares dos agentes somente são capazes de se comunicar à curta distância. Geometria Ambiental, tecnológia de transmissão via wireless, considerações energéticas e atmosféricas, tudo contribui para limitar a comunicação em curta distância. Uma restrição comum que raramente satisfaz no mundo real para uma comunicação ser bem-sucedida é manter a linha de visão entre os agentes. No entanto, MRS para a maioria das aplicações reais são compensativos, somente se o nível de comunicação for alta, quando comparado com sistemas de agentes individuais ou MRS que não se comunicam. Em um ambiente desconhecido, no entanto, a hipótese de um ponto absoluto para *rendezvous* não pode ser adotada. Uma estratégia comum em MRSLAM é considerar que um agente deve esperar para ser encontrado por outros agentes.

3.11 Alinhamento de mapa

O problema da fusão de dois mapas representada como grades de ocupação é semelhante ao problema de registo da imagem estudado em visão computacional (BROWN, 1992). Os mapas podem ser, de fato, visto como imagens, e a fusão dos mapas pode ser vista como um caso particular de registro de imagem. A fusão de mapas é abordado em duas etapas: alinhamento e fusão. Na fase de alinhamento, o objetivo é a transformação dos diferentes sistemas de referências dos mapas locais. Nesta etapa, a maioria das soluções envolve tentar encontrar a posição relativa dos robôs. Neste sentido, a posição relativa dos robôs já é supostamente conhecida, já que os robôs estabeleceram um encontro, com intuito de medir suas posições relativas, utilizando como meio de identificação, por exemplo, câmeras e/ou códigos de cores.

O conjunto de correspondências e estas estimativas são usados como entrada de uma minimização do quadrados que eliminam as linhas externas e obtém um solução final que é expresso no mesmo sistema de referência global. Felizmente, ROS oferece um pacote para o alinhamento que faz exatamente isso, chamado *mapstitch*[6].

[6] Pacote ROS mapstich. Disponível no:<http://ros.org/wiki/mapstich>

Como existem alguns problemas de alinhamento que não são totalmente resolvidos por esse pacote, como caso da multiplicação de matrizes, o que causaria com que um dos mapas translada-se infinitamente. Foi utilizado um detector, *Speeded-Up Robust Features* (SURF), disponível no OpenCV[7] um framework de visão computacional que está incluso no ROS.

3.12 Detecção Mútua

Para se beneficiar da abordagem de mapeamento com múltiplos robôs, eles devem ser capazes de se localizarem um em relação ao outro de maneira rápida, precísa e confiável. Isto significa que os robôs que trabalham com um objetivo em comum, devem possuir a capacidade de perceber um ao outro. Para atingir essa capacidade, assumiremos que temos uma equipe de robôs homogêneos e que as dimensões físicas dos robôs são conhecidos. A fim de evitar o mapeamento incorreto do meio ambiente, devido às obstruções causadas por outros robôs da mesma equipe, é adicionado um processo que filtra a presença do robô. Este processo, definido por *mutual detection*, benefícia-se do fato de receber a transformação do resultado do alinhamento dos mapas entre dois robôs e a posição relativa desses robôs no mapa, bem como sua posições relativas por meio de um referencial comum. Isso permite o uso de uma área delimitada cujo centro é localizado no ponto médio entre os robôs detectados. O formato dos robôs é aproximado de maneira a ficar parecido com um círculo, que possui raio de acordo com o modelo do robô. A definição desta técnica é fundamental para o processo de filtragem, uma vez que pode-se verificar se as varreduras a laser dos robôs se intersectam na região definida. Nestas situações, um robô estará detectando um segundo robô e o mapa gerado não irá corresponder ao ambiente real. Então, será feita uma nova varredura com o sensor, na qual todas as verificações que se cruzam na região entre os robôs são descartados. O resultado do processo de filtragem é um mapa do ambiente, sem as presenças dos robôs.

Um novo *node* chamado "*filter*" foi criada para reenviar as mensagens de escanamento do sensor e as mensagens filtradas pelo módulo *mutual detection* filtrado em uma taxa 10 Hz. Este *node* funciona de uma forma bastante simples. Quando o escaner laser de um robô está detectando outro robô da equipe, um sinalizador é ativado e o módulo *mutual detection* irá iniciar a filtragem no escaneamento. Esse *node*, que inicialmente só publicou mensagens originais do laser do robô para *Gmapping* agora irá publicar mensagens filtrados do laser pelo módulo *mutual detection* proposto. O *node* irá filtrar as mensagens enquanto o robô está detectando um companheiro de equipe. Para voltar para o mensagens de laser normais, no final do módulo *mutual detection*, um sinalizador é ativado mais um vez e novamente o sistema informa ao *node* para voltar ao estado normal.

[7] Página do OpenCV: <http://opencv.org/>

3.13 Mesclagem de Mapa

Mesclagem de mapas é uma tarefa desafiadora. Combinar os mapas parciais de todos os robôs em um mapa global permite ao time de robôs evitar uma exploração repetida e indesejada das mesmas regiões por diferentes robôs. Neste trabalho, é proposto um metodologia para fusão de mapas que pode ser utilizado em qualquer ambiente genérico. O algoritmo apresentado é capaz de combinar com sucesso um mapa parcial de grades de ocupação de robôs com um grau limitado de regiões sobrepostos entre seus mapas. Nesta subseção, apresentamos a abordagem para manipular grades de ocupação e fundi-los em mapa único e global. A fim de alcançar este objetivo, utilizaremos um pacote estável disponível em ROS, chamado *occupancy_grid_utils*[8]. Este pacote contém algumas utilidades para lidar com grades de ocupação, que incluem coordenar as conversões, caminhos mais curtos, traçador de raios, e união de múltiplas grades desalinhadas, que são fundamentais para nossa meta: um mapa global construído a partir de inúmeros mapas incompletos.

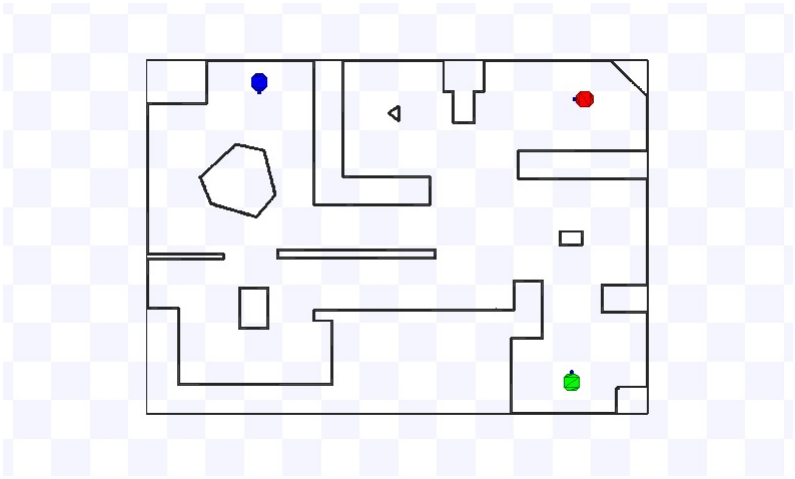
[8] Pacote ROS *occupancy_grid_utils*. Disponível no:<http://www.ros.org/wiki/occupancy_grid_utils>

4 RESULTADOS

Este capítulo descreve e discute o resultado experimental da simulação considerada neste trabalho. O teste utiliza inteiramente a abordagem distribuida, onde cada robô é responsável pelo alinhamento e a mesclagem de mapas com o objetivo de obter um mapa global do ambiente. Para representar o ambiente será considerada mapas de grades de ocupação.

4.1 Resultado da Simulação

Figura 14 – Teste, arena #1 utilizado nas simulações no Stage com três robôs em suas respectivas posições iniciais.



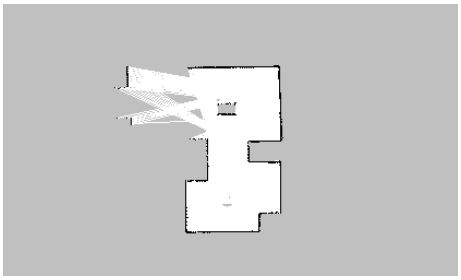
Fonte: Autor

Os resultados que foram obtidos neste projeto consideram a implementação dos módulos descritos no capítulo 3. A figura 14 ilustra a janela do Stage[1]. Esta arena é a simulação de um ambiente fechado de dimensões (13x9 metros) com vários obstáculos.

Para distinguir os robôs , diferentes cores foram atribuídas para cada robô: rob_0 é vermelho, rob_1 é azul e rob_2 é o verde. O objetivo da simulação é contruir toda a arena de teste utilizando três mapas parciais dos robôs. O mapas parciais obtidos possuem resolução de 0,04m/pixel e o tamanho de 544 células. Cada robô envia seu mapa parcial para os outros robôs com um limite de 30 segundos (nas simulações não está disponível a medição da qualidade da conexão), mesclando os mapas em um mapa global e publicando em um adequado *topic* interno.

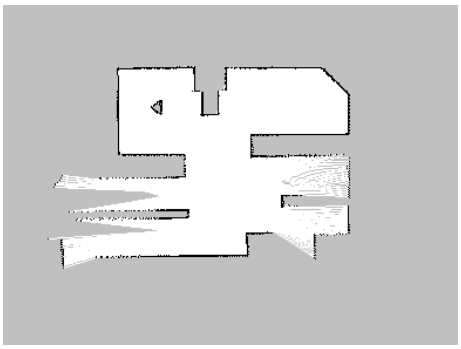
[1] Pacote ROS Stage. disponível no: <http://www.ros.org/wiki/stage>

Figura 15 – Mapa parcial obtido pelo rob_0.



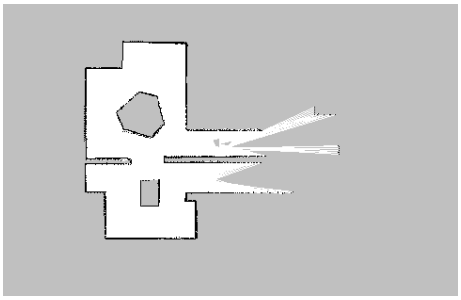
Fonte: Autor

Figura 16 – Mapa parcial obtido pelo rob_1.



Fonte: Autor

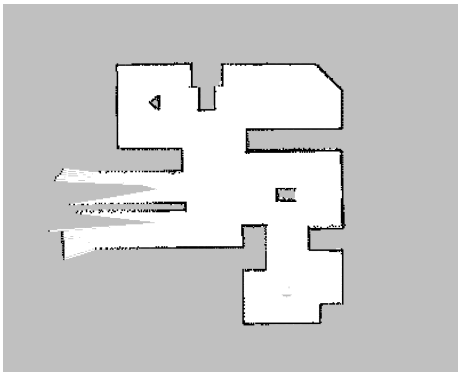
Figura 17 – Mapa parcial obtido pelo rob_2.



Fonte: Autor

Figuras 15, 16 e17 ilustram a representação dos mapas parciais de cada robô antes das mesclagem. Quando o rob_0 compartilha seu mapa com rob_1, os mapas parciais dos robôs são atualizados com base no resultado da mesclagem dos mapas, como ilustrado na figura 18. Então, quando o rob_1 compartilha seu mapa com rob_2, a mesclagem dos mapas parciais gera um mapa global com a contribuição de todos os robôs. Este mapa global final tem uma resolução de 0,01 m/pixel e tamanho de 3329x2177 células. A figura19 ilustra a mesclagem entre os mapas parciais das figuras 17 e 18.

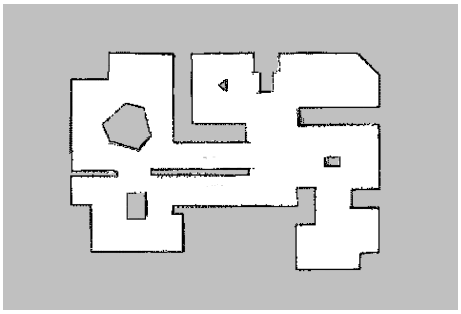
Figura 18 – Mapa mesclado do rob_0 e rob_1.



Fonte: Autor

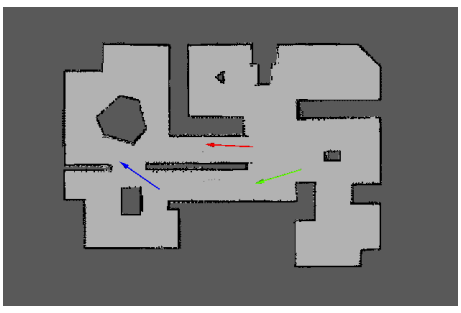
A Figura 19 ilustra a representação do mapa global final, após a navegação dos robôs na arena. Os resutados obtidos na figura 20 ilustram o alinhamento e a mesclagem correta dos mapas parciais de cada robô e com isto pode ser concluido que o teste foi satisfatório, visto que o mapa global obtido é parecido com o mapa no mundo virtual do *Stage*. Também é possível verificar, que para ambientes com maior extensão, o *Gmapping* pode ter a performance extendida. A principal razão para este teste ser satisfatório é graças ao fato de o ambiente simulado possuir várias *landmarks*

Figura 19 – Mapa global com as informações de todos os robôs.



Fonte: Autor.

Figura 20 – Mapa global com a posições do robôs indicadas, *rvez*.



Fonte: Autor.

5 CONCLUSÃO

Localização e Mapeamento Simultâneo para Múltiplos Robôs ainda possui um grande potencial para pesquisa e este trabalho é uma das formas de atingir o objetivo de obter um mapa global com a ajuda de múltiplos robôs.

Diferentes soluções para o problema do SLAM foram apresentadas. Como por exemplo, a utilização de um método distribuído, onde cada par de robôs compartilham os mapas de acordo com sua distância geométrica, qualidade e tempo de comunicação, com objetivo de mesclar suas contribuições de forma local. No entento, extensões para múltiplos robôs ainda são escassaz. MRSLAM são caracterizados por limitações na comunicação e também pelo fato de a posição inicial dos robôs ser geralmente conhecida. Neste trabalho, um método distribuido para MRSLAM foi mostrado, onde as posições iniciais são desconhecidas e um algoritmo RBPF-SLAM foi utilizado. Descrição da implementação do alinhamento de grades de ocupação, detecção mútua de robôs e mesclagem de mapas no sistema ROS também foram abordadas. Além da comunicação limitada entre os robôs, também foram verificados outros desafios, que incluem os recursos limitados disponíveis, como memória de armazenamento e o esforço computacional. Para identificar claramente cada robô e os dados observados por cada um em ROS, foi utilizado um separador de *namespace* conhecido como *prefix*. O alinhamento e mesclagem do mapa local depende do encontro entre robôs no ambiente. Durante o *rendezvous*, os robôs trocam seus mapas e filtram as observações de cada um no mapa mesclado. O processo de otimização do alinhamento/mesclagem é baseado na comunicação entre os robôs: a troca de informações ocorre quando os robôs estão próximos um do outro e possuem uma conexão de alta qualidade.

O resultado da simulação mostra que os mapas obtidos são capazes de fornecer características parecidas com a do ambiente simulado, indicando que o método adotado, com melhorias, pode trazer boas pespectivas no futuro. Os resultados mostram ainda a possibilidade de utilizar um time com múltiplos robôs para explorar e navegar em ambientes desconhecidos. No entanto, ainda há algumas limitações, como quando há *rendezvous* com mais de dois robôs. Pelo fato de o alinhamento ser feito via P2P, se torna difícil lidar com detecção mútua para robôs que não estão incluídos na lista de tarefas, o que pode resultar em uma possível detecção desse tipo de robô no mapa global, havendo necessidade de melhorar ou até mesmo trocar o modo de detecção mútua, afim de evitar ou minimizar esse tipo de problema.

5.1 Trabalhos Futuros

Alguns problemas ainda permanecem em aberto e correspondem a futuras orientações que podem ser tomadas para melhorar este trabalho. A técnica SLAM pode ser extendida para ambientes externos. Otimização da troca de mapas utilizando métodos de compressão podem ser usados para evitar sobrecarga na comunicação e no tempo necessário para troca de informações entre os robôs. Outra idéia interessante seria implementar um módulo para alinhamento dos mapas que não dependa do OpenCV, e que possa deixar este processo mais rápido, leve e compatível com vários robôs em vez de ser limitado a somente a um par de robôs.

REFERÊNCIAS

ANDERSSON, L. A.; NYGARDS, J. On multi-robot map fusion by inter-robot observations. In: IEEE. *Information Fusion, 2009. FUSION'09. 12th International Conference on*. [S.l.], 2009. p. 1712–1721. 17, 21, 22

BIRK, A.; CARPIN, S. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, IEEE, v. 94, n. 7, p. 1384–1397, 2006. 10

BONACCORSO, F.; MUSCATO, G.; BAGLIO, S. Laser range data scan-matching algorithm for mobile robot indoor self-localization. In: IEEE. *World Automation Congress (WAC), 2012*. [S.l.], 2012. p. 1–5. 14

BROWN, L. G. A survey of image registration techniques. *ACM computing surveys (CSUR)*, ACM, v. 24, n. 4, p. 325–376, 1992. 36

BURGARD, W. et al. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, IEEE, v. 21, n. 3, p. 376–386, 2005. 20, 22

CARLONE, L. et al. Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In: IEEE. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. [S.l.], 2010. p. 243–249. 15, 21, 22

CHANG, H. J. et al. Multi-robot slam with topological/metric maps. In: IEEE. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. [S.l.], 2007. p. 1467–1472. 21, 22

DISSANAYAKE, M. G. et al. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, IEEE, v. 17, n. 3, p. 229–241, 2001. 10, 15

DOUCET, A. et al. Rao-blackwellised particle filtering for dynamic bayesian networks. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. [S.l.], 2000. p. 176–183. 27

ELFES, A. Autonomous robot vehicles, chapter sonarbased real-world mapping and navigation. Springer, Berlin, 1990. 23

FENWICK, J. W.; NEWMAN, P. M.; LEONARD, J. J. Cooperative concurrent mapping and localization. In: IEEE. *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. [S.l.], 2002. v. 2, p. 1810–1817. 10, 15

FOX, D. et al. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, IEEE, v. 94, n. 7, p. 1325–1339, 2006. 21, 22

GUIVANT, J. E.; NEBOT, E. M. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *Robotics and Automation, IEEE Transactions on*, IEEE, v. 17, n. 3, p. 242–257, 2001. 15

KO, J. et al. A practical, decision-theoretic approach to multi-robot mapping and exploration. In: IEEE. *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. [S.l.], 2003. v. 4, p. 3232–3238. 10

LAKEMEYER, G.; NEBEL, B. *Exploring artificial intelligence in the new millennium*. [S.l.]: Morgan Kaufmann, 2003. 10

LEÓN, A. et al. Multi-robot slam and map merging. In: *IX Workshop of Physical Agents (WAF 08)*. [S.l.: s.n.], 2008. p. 171–176. 15

MONTEMERLO, M.; THRUN, S. Simultaneous localization and mapping with unknown data association using fastslam. In: IEEE. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. [S.l.], 2003. v. 2, p. 1985–1991. 15

PORTUGAL, D.; ROCHA, R. P. Extracting topological information from grid maps for robot navigation. In: *ICAART (1)*. [S.l.: s.n.], 2012. p. 137–143. 23, 24

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: *ICRA workshop on open source software*. [S.l.: s.n.], 2009. v. 3, n. 3.2, p. 5. 34

ROCHA, R. P. P. d. Building volumetric maps with cooperative mobile robots and useful information sharing: a distributed control approach based on enthropy. 2006. 16, 17, 18, 19, 23

ROLLER, D. et al. Fastslam 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: *Proceedings of the International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2003. 15

ROY, N.; DUDEK, G. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, Springer, v. 11, n. 2, p. 117–136, 2001. 10

RYBSKI, P. E. et al. Communication strategies in multi-robot search and retrieval: Experiences with mindart. In: *Distributed Autonomous Robotic Systems 6*. [S.l.]: Springer, 2007. p. 317–326. 25

SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating uncertain spatial relationships in robotics. In: *Autonomous robot vehicles*. [S.l.]: Springer, 1990. p. 167–193. 10, 15

STEWART, B. et al. The revisiting problem in mobile robot map building: A hierarchical bayesian approach. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. [S.l.], 2002. p. 551–558. 10

THRUN, S.; BÜCKEN, A. *Learning Maps for Indoor Mobile Robot Navigation.* [S.l.], 1996. 23

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press, 2005. 13

THRUN, S. et al. Robust monte carlo localization for mobile robots. *Artificial intelligence*, Elsevier, v. 128, n. 1, p. 99–141, 2001. 10, 27

THRUN, S.; LIU, Y. Multi-robot slam with sparse extended information filers. In: SPRINGER. *Robotics Research. The Eleventh International Symposium*. [S.l.], 2005. p. 254–266. 22

THRUN, S. et al. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 2003. 10, 20

THRUN, S. et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, Morgan Kaufmann San Francisco, CA, p. 1–35, 2002. 23

WHYTE, H.; BALIEY, T. Simultaneous localization and mapping (slam) part 1 the essential algorithms. *IEEE Robotics & Automation Magazine*, 2006. 14

ZHANG, L.; MENG, X.-j.; CHEN, Y.-w. A fastslam algorithm based on the auxiliary particle filter with stirling interpolation. In: IEEE. *Information and Automation, 2009. ICIA'09. International Conference on*. [S.l.], 2009. p. 167–172. 15

ZHOU, X. S.; ROUMELIOTIS, S. I. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In: IEEE. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. [S.l.], 2006. p. 1785–1792. 21