Dissertação apresentada à Pró-Reitoria de Pós-Graduação do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Programa de Pós-Graduação em Engenharia Eletrônica e Computação, Área de Sistemas e Controle.

Wellington Vieira Martins de Castro

SLAM DISTRIBUÍDO ENVOLVENDO NAVEGAÇÃO, GUIAMENTO E FUSÃO SENSORIAL PARA RECONSTRUÇÃO 2D

Dissertação aprovada em sua versão final pelos abaixo assinados:

Prof. Dr. Jacques Waldmann

Orientador

Prof. Dra. Emília Villani

Pró-Reitora de Pós-Graduação

Campo Montenegro
São José dos Campos, SP - Brasil
2022

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

de Castro, Wellington Vieira Martins
SLAM distribuído envolvendo navegação, guiamento e fusão sensorial para reconstrução 2D / Wellington Vieira Martins de Castro.
São José dos Campos, 2022.
115f.

Dissertação de Mestrado – Curso de Engenharia Eletrônica e Computação. Área de Sistemas e Controle – Instituto Tecnológico de Aeronáutica, 2022. Orientador: Prof. Dr. Jacques Waldmann.

1. Robôs. 2. Dinâmica de robôs. 3. Localização e mapeamento simultâneo. 4. Sensores. 5. Robótica. 6. Computação. 7. Controle. I. Instituto Tecnológico de Aeronáutica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

DE CASTRO, Wellington Vieira Martins. **SLAM distribuído envolvendo navegação, guiamento e fusão sensorial para reconstrução 2D**. 2022. 115f. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Wellington Vieira Martins de Castro
TITULO DO TRABALHO: SLAM distribuído envolvendo navegação, guiamento e fusão sensorial para reconstrução 2D.
TIPO DO TRABALHO/ANO: Dissertação / 2022

Wellington Vieira Martins de Castro
Av. Tívoli, 550. Apartamento 65
12.245-481 – São José dos Campos–SP

SLAM DISTRIBUÍDO ENVOLVENDO NAVEGAÇÃO, GUIAMENTO E FUSÃO SENSORIAL PARA RECONSTRUÇÃO 2D

Wellington Vieira Martins de Castro

Composição da Banca Examinadora:

Prof. Dr.	Cairo Lúcio Nascimento Júnior	Presidente	-	ITA
Prof. Dr.	Jacques Waldmann	Orientador	-	ITA
Prof. Dr.	Marcelo da Silva Pinho		-	ITA
Prof. Dr.	Reinaldo Augusto da Costa Bianchi	Membro Externo	-	FEI

ITA

Aos meus pais e amigos.

Agradecimentos

Primeiro gostaria de agradecer meus pais, Adriana e Emerson, por terem prezado minha educação desde os primeiros anos, pois seu apreço e esforços somados à minha vontade me fizeram chegar num lugar incomum para as pessoas de nossa realidade socioeconômica. Além disso, eles sempre são os primeiros a acreditar em mim em meus momentos de dúvida. Devo tudo a eles.

Agradeço ao ITA, pois o instituto figura desde cedo em meu imaginário e estudar aqui sempre foi um sonho. Quero agradecer também ao meu orientador, professor Jacques, por propor um tema de pesquisa tão rico e apaixonante. Também agradeço à CAPES pelo fundamental fomento na forma de bolsa de estudos.

Por fim, quero agradecer aos meus amigos tanto de graduação do LIA-FACOM da UFMS, quanto da pós-graduação do ITA. A amizade e incentivo de vocês tornam a vida mais leve e gostosa de ser vivida, principalmente nos dias difíceis de um mestrando. Também quero agradecer a minha namorada, Camila, por todo amor e carinho durante a última parte dessa jornada, e também por toda compreensão nos momentos que fui ausente.

Além disso, agradeço especificamente à Camila e ao Ivan por terem lido diversas versões desse texto sugerindo alterações pertinentes para facilitar seu entendimento e prazer de leitura.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

"O conhecido é finito, o desconhecido, infinito; intelectualmente estamos numa ilhota no meio de um oceano ilimitado de inexplicabilidade. Nossa função em cada geração é reinvindicar um pouco mais de terra firme."

— T. H. Huxley

Resumo

O problema de Localização e Mapeamento Simultâneos, conhecido pela sigla SLAM, pergunta se é possível para um robô ser colocado em um ambiente desconhecido a priori, e incrementalmente construir um mapa deste ambiente enquanto simultaneamente se localiza neste mapa sem a necessidade de infraestrutura de localização como GPS. A solução do problema SLAM é fundamental para a robótica móvel autônoma. Entretanto, apesar de já solucionado, não é uma tarefa trivial tanto do ponto de vista teórico como do ponto de vista da implementação. Dependendo da dinâmica do robô, sensores utilizados, recurso computacional disponível, necessidade de navegação e guiamento, a solução pode se tornar mais ou menos complexa. Este trabalho investiga uma solução multiagente distribuída e descentralizada em ambiente simulado para o problema SLAM 2D. Para isso emprega o uso do Filtro de Informação Estendido Esparso, juntamente com outros algoritmos de navegação, associação de dados e de representação de mapas. As vantagens da solução distribuída do problema SLAM, em relação ao problema original, são a divisão da carga de trabalho entre os agentes e a redundância de informação. O objetivo deste trabalho é simular um grupo de robôs que mapeiem o ambiente onde estão inseridos, de maneira ativa e descentralizada, considerando restrições de memória e processamento. Portanto, além de produzir algoritmos que capacitem os robôs a resolverem o problema SLAM Ativo Descentralizado e Distribuído, é preciso criar uma infraestrutura de software na qual o ambiente e os agentes serão simulados. Para isso utilizou-se o Sistema Operacional de Robô, ROS do inglês *Robot Operating System*, que provê um conjunto de bibliotecas e ferramentas pertinentes ao desenvolvimento em robótica, além de uma camada de comunicação utilizada pelos diferentes módulos do sistema (mapeamento, navegação, visão) para trocarem informações. Para a simulação do ambiente, sensores e agentes, utilizou-se o simulador Gazebo. A solução desenvolvida apresenta uso de memória significativamente reduzido, mais de 50%, em relação à abordagem clássica EKF-SLAM. Além disso, apesar de não possuir coordenação entre os agentes, observaram-se ganhos no tempo de exploração no cenário com dois agentes em comparação com um único agente.

Abstract

The Simultaneous Localization and Mapping problem, known by the acronym SLAM, asks whether it is possible for a robot to be placed in an environment unknown a priori, and incrementally build a map of this environment while simultaneously locating itself on this map without the need for a location infrastructure such as GPS. Solving the SLAM problem is critical for mobile robotics autonomy. However, although already solved, it is not a trivial task both from theoretical and implementation points of view. Depending on robot dynamics, sensors used, computing power available or need for navigation and guidance, the solution becomes more or less complex. We examine a distributed and decentralized multi-agent solution in a simulated environment for the 2D SLAM problem. For such purpose, the solution uses the Sparse Extended Information Filter, along with other algorithms for navigation, data association and map representation. The advantages of the distributed solution of the SLAM problem, in comparison with the original problem, are workload sharing among the agents and information redundancy. Our goal is to simulate a group of mobile robots mapping the environment where they were placed in an active and decentralized fashion while taking into account processing and memory constraints. Beyond implementing algorithms that enable robots to solve the Distributed and Decentralized Active SLAM problem, a software framework is needed to simulate the environment and the agents. We used the Robot Operating System (ROS) to provide a set of software libraries and tools relevant to robotics development, and also a communication layer used by the many system modules (mapping, navigation, vision) in information exchanging. The simulation of the environment, sensors and agents, the Gazebo simulator was used. The developed solution has significantly less memory usage, presenting a reduction of more than 50% when compared to the classic EKF-SLAM approach. In addition, despite the lack of coordination between the agents, gains in exploration time were observed in the scenario with two agents with respect to a single agent.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CML	*Concurrent Mapping and Localization* (Mapeamento e Localização Concorrentes)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
SLAM	*Simultaneous Localization and Mapping* (Localização e Mapeamento Simultâneos)
ISSR	*International Symposium on Robotics Research* (Simpósio Internacional de Pesquisa em Robótica)
ROS	*Robot Operating System* (Sistema Operacional de Robô)
KF	*Kalman Filter* (Filtro de Kalman)
EKF	*Extended Kalman Filter* (Filtro de Kalman Estendido)
EIF	*Extended Information Filter* (Filtro de Informação Estendido)
SEIF	*Sparse Extended Information Filter* (Filtro de Informação Estendido Esparso)
IMU	*Inertial Measurement Unit* (Unidade de Medição Inercial)
LiDAR	*Light Detection and Ranging*
RANSAC	*Random Sample Consensus*
IAE	*Integral Absolute Error* (Erro Absoluto Integral)

Lista de Símbolos

$\mathbf{x}_t^{\mathrm{r}}$	Pose do robô no instante t
$\mathbf{x}_{0:t}^{\mathrm{r}}$	Conjunto das poses do robô do instante 0 até t
$\mathbf{u}_t$	Entrada de controle no instante t
$\mathbf{u}_{0:t}$	Conjunto das entradas de controle to instante 0 até t
$\mathbf{z}_t$	Medida do sensor extrínseco no instante t
$\mathbf{z}_{0:t}$	Conjunto das medidas do sensor extrínseco do instante 0 até t
$\mathbf{m}$	Mapa do ambiente, constituído das coordenadas das *landmarks*
$p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0^{\mathrm{r}}\right)$	Função densidade de probabilidade (pdf) do problema online SLAM
$p\left(\mathbf{x}_{1:t}^{\mathrm{r}}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0^{\mathrm{r}}\right)$	Função densidade de probabilidade (pdf) do problema full SLAM
$\boldsymbol{\mu}_t$	Vetor de média do estado no instante t
$\overline{\boldsymbol{\mu}}_t$	Predição do vetor de média do estado no instante t
$\mathbf{P}_t$	Matriz de covariância no tempo t
$\overline{\mathbf{P}}_t$	Predição da matriz de covariância no tempo t
$\mathbf{K}_t$	Ganho de Kalman no tempo t
$\boldsymbol{g}_{\mathrm{r}}(\cdot, \cdot)$	Modelo de movimento do robô
$\boldsymbol{h}(\cdot)$	Modelo de medida do sensor laser
$\boldsymbol{f}(\cdot, \cdot)$	Modelo de medida inverso do sensor laser
$\mathbb{E}\left[\cdot\right]$	Operador esperança
$\mathbf{0}_{m \times n}$	Matriz nula de dimensão $m \times n$
$\mathbf{I}_n$	Matriz identidade de dimensão n
$Cov\left(\cdot\right)$	Covariância
$\boldsymbol{\Omega}_t$	Matriz de informação no tempo t
$\overline{\boldsymbol{\Omega}}_t$	Predição da matriz de informação no tempo t
$\boldsymbol{\xi}_t$	Vetor de informação no tempo t
$\overline{\boldsymbol{\xi}}_t$	Predição do vetor de informação no tempo t
$\mathbf{M}$	Matriz de projeção
$\mathbf{R}_t^{\mathrm{r}}$	Ruído do modelo cinemático do robô
$\mathcal{T}$	Transformação afim
$\mathcal{T}_r^{j \to k}$	Transformação de corpo rígido do sistema j para o sistema k

Sumário

1 Introdução

A robótica móvel está se desenvolvendo rapidamente, no entanto, aplicações em ambientes sem uma infraestrutura de mapeamento global exigem técnicas de mapeamento e percepção robustas para capacitar agente móveis a navegar de maneira autônoma em ambientes complexos (SAEEDI *et al.*, 2016). Aplicações dessa natureza não surgem apenas da exploração de outros planetas onde não há um Sistema de Posicionamento Global (GPS), por exemplo. Pelo contrário, há diversos cenários na Terra onde o GPS não funciona adequadamente, como debaixo d'água, em minas subterrâneas e dentro de construções. Este problema é denominado SLAM e resolvê-lo implica em empoderar soluções de robótica móvel, que atuem nesses cenários, com aplicações reais e comerciais.

O problema de Mapeamento e Localização Simultâneos, conhecido pela sigla SLAM por conta do termo em inglês *Simultaneous Localization and Mapping*, pergunta se é possível para um robô móvel ser colocado em um ambiente desconhecido a priori e incrementalmente construir um mapa deste ambiente enquanto simultaneamente se localiza neste mapa. Ou seja, tanto a trajetória da plataforma móvel quanto a localização das características do mapa (também conhecidas por *landmarks* ou, ainda, pontos de referência) são estimadas em tempo real sem a necessidade de nenhum conhecimento a priori de suas localizações (DURRANT-WHYTE; BAILEY, 2006), ou infra estrutura de localização prévia, como GPS.

SLAM também já foi conhecido como Mapeamento e Localização Concorrentes (CML, do inglês *Concurrent Mapping and Localization*), porém este termo caiu em desuso a partir de 1995 quando o termo SLAM foi cunhando por Durrant-Whyte *et al.* (1996) no Simpósio Internacional de Pesquisa em Robótica, ISSR, onde originalmente era chamado *Simultaneous Localization and Map Building*. A solução do problema SLAM é fundamental para atingir a robótica móvel autônoma e independente de operadores (DURRANT-WHYTE; BAILEY, 2006). Entretanto resolver o problema de localização e mapeamento simultâneos, apesar de solucionado, não é uma tarefa trivial tanto do ponto de vista teórico como do ponto de vista da implementação (DURRANT-WHYTE *et al.*, 1996).

Caracterização do problema

Imagine um robô se deslocando por um ambiente e dotado de um sensor extrínseco, capaz de capturar medidas relacionadas ao ambiente e, um sensor intrínseco capaz de medir os comandos de controle executados. Até o instante t as seguintes quantidades são observadas:

- $\mathbf{x}_t^{\mathrm{r}}$: o vetor de estados descrevendo a pose do robô no instante t.
- $\mathbf{x}_{1:t}^{\mathrm{r}} = \{\mathbf{x}_1^{\mathrm{r}}, \mathbf{x}_2^{\mathrm{r}}, \ldots, \mathbf{x}_{t-1}^{\mathrm{r}}, \mathbf{x}_t^{\mathrm{r}}\}$: histórico de poses do robô até o instante t.
- $\mathbf{u}_t$: o vetor de controle executado pelo robô no instante t.
- $\mathbf{u}_{1:t} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{t-1}, \mathbf{u}_t\}$: histórico de controles executados pelo robô até o instante t.
- $\mathbf{y}_t$: o vetor de medidas no instante t
- $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}, \mathbf{y}_t\}$: o conjunto de todas as medidas realizadas até o instante t.
- $\mathbf{m}$: o vetor de mapa, constituído pelas posições das características do ambiente consideradas pelo robô.

De maneira bastante sucinta, os problemas de SLAM consistem em estimar uma das seguintes distribuições de probabilidade:

$$p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0^{\mathrm{r}}\right) \tag{1.1}$$

$$p\left(\mathbf{x}_{1:t}^{\mathrm{r}}, \mathbf{m} \,|\, \mathbf{y}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0^{\mathrm{r}}\right) \tag{1.2}$$

Além da solução da primeira distribuição se preocupar apenas em estimar o estado atual, enquanto na segunda toda a trajetória, o histórico de poses até o instante t, é estimada. A diferença fundamental entre as duas soluções é que, para calcular a primeira distribuição, não são utilizadas entradas de controle e medidas posteriores a um instante t para estimar a pose $\mathbf{x}_t^{\mathrm{r}}$, enquanto na solução da segunda, medidas e controles posteriores podem ser utilizados para calcular poses anteriores a eles. Habitualmente os problemas de estimação dessas duas distribuições são conhecidos como *online* SLAM e *full* SLAM, respectivamente.

Embora as definições do problema em 1.1 e 1.2 sejam simples, resolvê-lo está longe de ser. A depender das características do sistema como a dinâmica do robô, sensores utilizados, recursos computacionais disponíveis, restrição de tempo real e necessidade de navegação e guiamento autônomos, sua solução pode se tornar mais ou menos complexa.

Difícil também é aprender a terminologia utilizada pelos pesquisadores para cada uma dessas características. Parte da terminologia do problema SLAM, pertinente a este trabalho, é abordada a seguir.

Taxonomia do problema SLAM

Como é de se esperar, há termos específicos para tratar cada aspecto de um sistema SLAM. Esta Seção visa apresentar os termos pertinentes a este trabalho, a fim de estabelecer um vocabulário comum que será utilizado em todas as seções e capítulos subsequentes a este.

Como mencionado anteriormente, há duas classes de problemas de SLAM: *online* SLAM e *full* SLAM, e, portanto, duas classes de algoritmos para resolvê-los. Os algoritmos *full* SLAM também são chamados de *offline SLAM* por serem normalmente utilizados em etapas de pós-processamento, como refinamento de mapas. Esses algoritmos exigem mais recursos, tanto de processamento quanto de memória para serem processados. Portanto, há grande dificuldade para utiliza-los embarcados nos agentes durante a etapa de exploração do ambiente.

Em contrapartida, os algoritmos que resolvem o problema de *online* SLAM são comumente utilizados de maneira embarcada, pois tendem a consumir menos recursos computacionais. A pose e o mapa estimado por eles podem ser utilizados no processo de tomada de decisão do agente durante a execução da tarefa de mapeamento.

Contudo, para que um robô consiga estimar precisamente sua pose $\mathbf{x}$, é necessário alimentar os algoritmos com as medidas $\mathbf{u}$, provenientes dos sensores intrínsecos (*encoders*, giroscópios e acelerômetros), e, também, com as medidas $\mathbf{y}$ do ambiente obtidas por sensores extrínsecos. O erro entre a posição esperada pelo robô de um objeto, dada a estimativa que o robô tem da sua pose, e a posição desse objeto lida pelo sensor extrínseco pode ser utilizado para atualizar a confiança que o robô tem sobre a sua pose, por exemplo. Aqui, objeto significa qualquer aspecto do ambiente com características suficientes que permita-o ser identificado, podendo ser desde objetos propriamente ditos como móveis e árvores, a pontos e quinas.

Essas medidas relacionadas ao ambiente, lidas pelos sensores extrínsecos (sonares, câmeras RGB, scanner laser, entre outros), possuem, em geral, duas componentes (no caso 2D) comumente denominados *Range* e *Bearing* ou, Distância e Azimute, respectivamente. A componente de *Range* é a distância do sensor até o objeto medido, enquanto *Bearing* é a direção angular do objeto em relação ao sensor. Porém, nem todo sensor é capaz de fornecer essas duas medidas; câmeras RGB, por exemplo, conseguem informar apenas a direção angular.

Então, de acordo com a presença/ausência dessas componentes os termos: *Range Only*, *Bearing Only* e *Range-Bearing* (Distância-Azimute) SLAM são utilizados para identificar qual classe de medidas do ambiente está sendo utilizada na solução do problema. *Range Only* significa que a medida possui apenas a componente de distância. Em medidas *Bearing Only*, apenas a direção angular é lida. Em Distância-Azimute ambas as quantidades são lidas; em Distância-Azimute SLAM são comumente utilizados sensores do tipo *LIDAR* (*Light Detection and Ranging*), que retornam uma nuvem de pontos na qual cada ponto é descrito pela distância e direção angular em relação ao sensor.

Com um algoritmo capaz de estimar 1.1 ou 1.2 e sensores apropriados para alimentá-lo, um robô é capaz de desempenhar SLAM como foi apresentado. Porém, ao mapear o ambiente, o robô pode explorá-lo de maneira autônoma ou, quando o cenário permite, ser controlado remotamente por um operador. Em cenários como a exploração de Marte tal controle é inviável, por exemplo. Quando a solução para o problema SLAM também incorpora a geração de trajetórias para exploração autônoma (ativa), é denominada SLAM Ativo.

Até o momento, o problema SLAM foi tratado aqui como se a tarefa fosse resolvida por um único agente/robô. Porém, é possível integrar mais robôs para executarem a tarefa de maneira conjunta, surgindo assim uma série de benefícios. O primeiro, e mais óbvio, benefício é que a tarefa pode ser executada mais rápido já que a carga de trabalho é dividida entre os agentes. Outro ponto é que, mesmo que um agente venha a sofrer um dano, a tarefa ainda pode ser concluída, pois o sistema pode reagir e redistribuir a tarefa entre os robôs restantes. Porém, esses benefícios vêm com o preço de um sistema complexo que lida com a coordenação e cooperação dos robôs (SAEEDI *et al.*, 2016). Essa abordagem com múltiplos robôs é chamada de SLAM Distribuído.

Além disso, dependendo da arquitetura do fluxo de informação entre os agentes, a abordagem SLAM Distribuído é subdividida em Centralizada e Descentralizada (CADENA *et al.*, 2016, p. 1316). Na arquitetura centralizada, há um nó central responsável por processar e distribuir o mapa global composto pelo mapa local de cada agente do sistema. Há, portanto um ponto de falha catastrófica: o nó central. Nessa arquitetura, é geralmente mais simples manter consistência e consenso entre agentes sobre o mapa global.

Em contrapartida, a arquitetura descentralizada não possui figura central; a comunicação e troca de mapas é realizada par a par entre os agentes. Neste arranjo, todo o processamento é feito na ponta; consenso e convergência se tornam mais complicados. Porém, o sistema se torna mais robusto com redundância de informação e ausência de falha catastrófica de um nó central.

1.1 Objetivo

O objetivo deste trabalho é simular um grupo de robôs que mapeiem o ambiente onde estão inseridos, sem nenhuma infraestrutura de localização como GPS, de maneira ativa e descentralizada, considerando restrições de memória e processamento.

Portanto, além de produzir algoritmos que capacitem os robôs a resolverem o problema SLAM Ativo Descentralizado e Distribuído, é preciso criar uma infraestrutura de software onde o ambiente e os agentes serão simulados. Para isso utilizou-se o Sistema Operacional de Robô, ROS do inglês *Robot Operating System*, que é um *framework* de código aberto e linguagem neutra (QUIGLEY *et al.*, 2009), amplamente utilizado pela indústria e pela academia. Ele provê um conjunto de bibliotecas e ferramentas pertinentes ao desenvolvimento em robótica, além de uma camada de comunicação utilizada pelos diferentes módulos do sistema (mapeamento, navegação, visão) para trocarem informações.

Dessa forma, o ROS torna este trabalho facilmente reutilizável em outras pesquisas, permitindo que cada um de seus módulos (simulação, visualização, SLAM e navegação) possa ser explorado e até modificado de forma individual. Além disso, permite que mais módulos sejam adicionados, estendendo as capacidades do sistema aqui desenvolvido.

Para a simulação do ambiente, sensores e agentes, utilizou-se o também amplamente difundido simulador Gazebo (KOENIG; HOWARD, 2004). Sua escolha se deu por conta de suas simulações fidedignas de sensores, massa, fricção, inúmeras outras variáveis físicas e também por sua natural integração com o ROS.

1.2 Estrutura de um sistema SLAM

Um sistema SLAM multiagente distribuído e descentralizado, como o desenvolvido neste trabalho, é resultado da sinergia entre diferentes módulos/algoritmos com funções e responsabilidades delimitadas. Cada um resolvendo um dos aspectos do problema, a listagem a seguir enumera, de maneira sucinta, os módulos que compõem a solução desenvolvida nesse trabalho:

- **Transformação de leituras em dados**: utilizar as leituras brutas do sensor LiDAR e extrair medidas para alimentar o modelo de medida.
- **Representação do ambiente**: há diferentes formas de representar o ambiente como mapas de *landmarks* ou mapas métricos em grade.
- **Exploração e Navegação**: para que os agentes mapeiem o ambiente é necessário que identifiquem regiões inexploradas e possuam capacidade de navegar até essas regiões evitando obstáculos ou até mesmo outros agentes.

- **Comunicação entre agentes**: durante a aproximação entre agentes, ocorre a troca dos mapas, tanto o mapa métrico quanto o de *landmarks*.

- **Estimação de posição relativa**: após uma comunicação é preciso estimar a posição relativa entre os agentes para que os mapas sejam combinados corretamente.

- **Associação de dados**: o robô utiliza os erros entre as medidas adquiridas e sua estimativa das mesmas para corrigir sua pose e a posição das *landmarks* em seu mapa. Para isso, é necessário estabelecer quais leituras são reobservações e, portanto, serão utilizadas para correção e quais são novas observações de *landmarks* que serão incorporadas ao mapa.

- **Filtro**: estima a pose do robô e a posição das *landmarks* de fato, utilizando como entrada as leituras dos sensores intrínsecos, as medidas processadas e as relações estabelecidas pela associação de dados.

Dessa forma, o desenvolvimento de um sistema SLAM também pode ser visto como um problema de Engenharia de *Software*, cujo objetivo é desenvolver módulos isolados que podem ser individualmente substituídos e/ou reutilizados desde que suas interfaces sejam mantidas.

Pensar o problema dessa forma torna o trabalho mais flexível, de fácil alteração e manutenção. Por exemplo: o algoritmo filtragem, utilizado para estimar a pose do robô e as posições das *landmarks*, pode ser substituído independentemente do *pipeline* de processamento de dados dos sensores, desde que sejam mantidas as estruturas de troca de informação entre eles. Outro aspecto importante é que a arquitetura de separação de responsabilidades força que as alterações no sistema sejam pontuais, tornando mais fácil o isolamento e identificação de erros decorrentes dessas alterações.

É comum agrupar esses módulos em dois macrogrupos denominados *back-end* e *front-end*. O *front-end* abstrai dados de sensores em modelos úteis para a estimação e também é onde geralmente são colocados os módulos de comportamento de alto nível como a exploração, enquanto o *back-end* utiliza o dado abstrato produzido pelo *front-end* para estimar o estado do sistema (CADENA *et al.*, 2016).

Neste trabalho, além do processamento dos dados do sensor extrínseco, o *front-end* também acumula outras responsabilidades como a representação métrica do ambiente, na forma de mapas de grade, a comunicação com outros agentes, a estimação da posição relativa entre agentes e, também, a navegação autônoma. O *back-end* é composto pelo algoritmo de estimação (filtro), responsável por estimar o estado do sistema dadas as leituras extrínsecas e intrínsecas, e pela associação de dados. Esse último normalmente é associado ao *front-end* (CADENA *et al.*, 2016), mas aqui foi colocado no *back-end* por conta da técnica utilizada estar incorporada ao filtro.

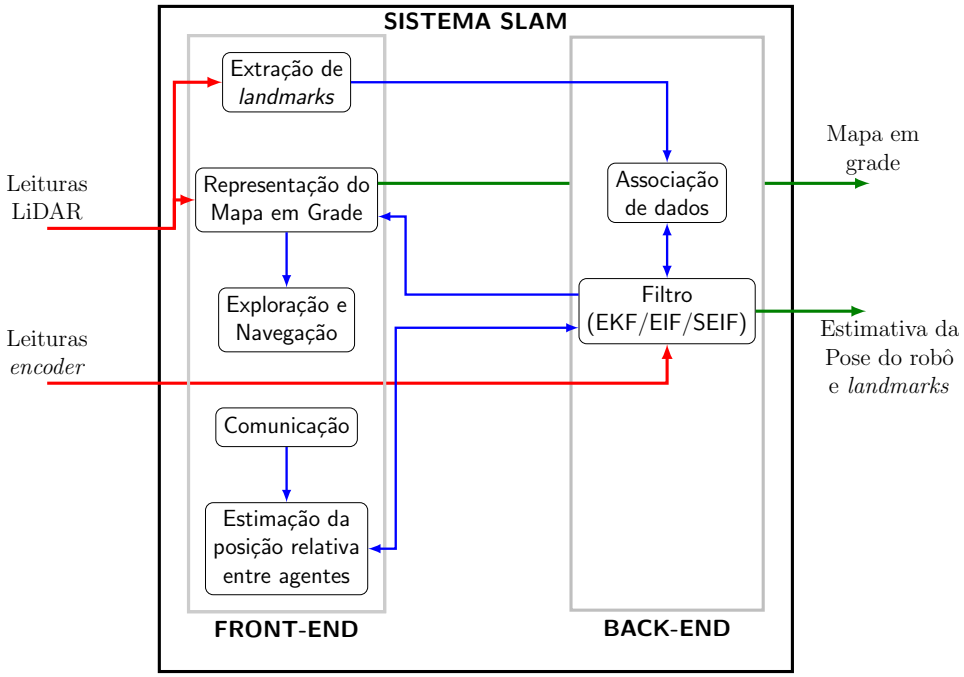


FIGURA 1.1 – Arquitetura simplificada do sistema SLAM desenvolvido neste trabalho. As entradas do sistema estão indicadas em vermelho, enquanto as saídas estão em verde. Em azul é possível observar algumas inter relações entre módulos.

1.3 Organização do trabalho

O próximo capítulo apresenta os modelos de cinemática e medida utilizados para o robô e o sensor, respectivamente. A organização do restante do trabalho é pautada pela Figura 1.1, o Capítulo 3 trata da maioria dos aspectos do *back-end*; começa discorrendo acerca de uma técnica clássica para resolver o problema SLAM e termina discutindo a técnica utilizada nesse trabalho; a porção do filtro responsável pela troca de mapas fica de fora desse capítulo para ser tratada em conjunto com a estimação de pose relativa posteriormente.

O Capítulo 4 é sobre os diversos componentes do *front-end*, exceto a estimação de posição relativa entre os agentes. Tanto esse cálculo quanto a troca de mapas do *back-end* são concentrados no Capítulo 5 por comporem a característica fundamental de sistemas multiagentes. No Capítulo 6 são apresentados o ambiente e o robô utilizados nas investigações e também é disponibilizado um apontamento para a implementação.

Por fim, o Capítulo 7 avalia e discute o sistema desenvolvido através de experimentos no ambiente simulado. O Capítulo 8, a Conclusão, sumariza os resultados, elenca as contribuições deste trabalho e sugere melhorias para trabalhos futuros.

2 Visão Geral do Sistema

Nesse capítulo são descritos o modelo cinemático do robô e os modelos de medida inverso e direto, utilizados pelos algoritmos de estimação para predizer poses e medidas.

2.1 Modelo do robô

Na Figura 2.1 é representado o esquemático de um robô diferencial; as características mais importantes nesse tipo de construção são o raio da roda, r, e a distância entre os eixos das rodas, $2d$. A pose do robô, no instante t, é definida como:

$$\mathbf{x}_t = \begin{bmatrix} \phi_t & x_t & y_t \end{bmatrix}^T \tag{2.1}$$

em que ϕ é o ângulo do eixo $\hat{x}_r$ do sistema de coordenadas móvel do robô com o eixo $\hat{x}$ do sistema de coordenadas estático $\{s\}$ e (x, y) é a origem do sistema de coordenadas do robô representada no sistema estático.

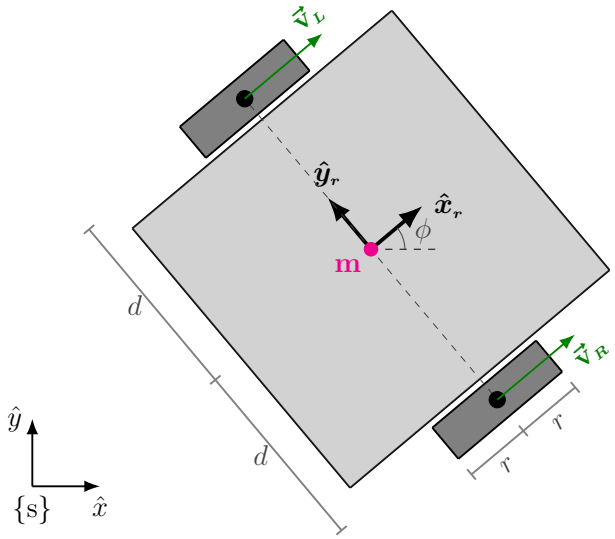


FIGURA 2.1 – Visão superior do esquemático de um robô diferencial. O eixo z está apontando para fora da folha. O sistema de coordenadas móvel do robô está posicionado no ponto médio do eixo das rodas (cinza escuro). Na imagem, $\{s\}$ é um sistema de coordenadas estático.

O modelo de movimento do robô, na Equação 2.2 usa apenas a informação dos *encoders*. Na expressão abaixo, a entrada $\mathbf{u}_t$ é composta pelos deslocamentos angulares, u_t^L e u_t^R, das rodas esquerda e direita durante o intervalo $]t-1, t]$.

$$\begin{aligned}\mathbf{x}_t &= \boldsymbol{g}_{\mathrm{r}}(\mathbf{x}_{t-1}, \mathbf{u}_t) \\ &= \begin{bmatrix}\phi_{t-1}\\ x_{t-1}\\ y_{t-1}\end{bmatrix} + \begin{cases} \begin{bmatrix} \alpha_t \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L}\left(\sin(\phi_{t-1}+\alpha_t) - \sin(\phi_{t-1})\right) \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L}\left(-\cos(\phi_{t-1}+\alpha_t) + \cos(\phi_{t-1})\right) \end{bmatrix} & \text{se } u_t^L \neq u_t^R \\ \begin{bmatrix} 0 \\ r \cdot u_t^R \cos(\phi_{t-1}) \\ r \cdot u_t^R \sin(\phi_{t-1}) \end{bmatrix} & \text{caso contrário} \end{cases}\end{aligned} \tag{2.2}$$

em que:

$$\alpha_t = \frac{r}{2d}\left(u_t^L - u_t^R\right) \tag{2.3}$$

Assim como os modelos de medida na próxima Seção, o modelo de movimento do robô de acionamento diferencial é não linear. Portanto, é necessário calcular sua matriz jacobiana em torno de um ponto $\mathbf{x}_{t-1}$ para linearização do modelo em técnicas como o Filtro de Kalman Estendido (EKF) e o Filtro de Informação Estendido Esparso (SEIF), apresentadas mais adiante. O jacobiano do modelo $\boldsymbol{g}_{\mathrm{r}}(\cdot,\cdot)$ está na Equação 2.4.

$$\mathbf{G}_R = \mathbf{I}_3 + \begin{cases} \begin{bmatrix} 0 & 0 & 0 \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L}\left(\cos(\phi_{t-1}+\alpha_{t-1}) - \cos(\phi_{t-1})\right) & 0 & 0 \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L}\left(\sin(\phi_{t-1}+\alpha_{t-1}) - \sin(\phi_{t-1})\right) & 0 & 0 \end{bmatrix} & \text{se } u_t^L \neq u_t^R \\ \begin{bmatrix} 0 & 0 & 0 \\ -r \cdot u_t^R \sin(\phi_{t-1}) & 0 & 0 \\ r \cdot u_t^R \cos(\phi_{t-1}) & 0 & 0 \end{bmatrix} & \text{caso contrário} \end{cases} \tag{2.4}$$

2.2 Medidas e o modelo de medida Distância-Azimute

A medida gerada pelo sensor LiDAR, utilizado nessa investigação, consiste em uma nuvem de pontos planar. Essa nuvem de pontos é processada e dela são extraídas es-

timativas dos centros dos cilindros (utilizados como *landmarks*) presentes no ambiente representado na Figura 6.1, no Capítulo 6. Esses centros são as medidas utilizadas pelo algoritmo de SLAM, eles são descritos em termos de coordenadas polares (r, θ) no sistema de coordenadas do sensor.

2.2.1 Modelo de medida Distância-Azimute

O modelo de medida calcula a medida que espera-se ser lida pelo sensor, quando o sistema está no estado $\mathbf{x}_t$. Na Figura 2.2, é representado um sistema composto por um robô e três *landmarks* i, j e k. Logo o vetor de estados, $\mathbf{x}$, é formado pela pose do robô, e pelas posições das landmarks no mapa:

$$\mathbf{x} = \begin{bmatrix} \phi & x & y & m_x^i & m_y^i & m_x^j & m_y^j & m_x^k & m_y^k \end{bmatrix}^T \tag{2.5}$$

O modelo de medida para a j-ésima *landmark* é dado por

$$\boldsymbol{h}^j(\mathbf{x}) = \begin{bmatrix} r^j \\ \theta^j \end{bmatrix} = \begin{bmatrix} \sqrt{(m_x^j - x_l)^2 + (m_y^j - y_l)^2} \\ \arctan\left(\dfrac{m_y^j - y_l}{m_x^j - x_l}\right) - \phi \end{bmatrix} \tag{2.6}$$

sendo

$$\begin{cases} x_l = x + d\cos\phi \\ y_l = y + d\sin\phi \end{cases} \tag{2.7}$$

Como o modelo de medida em 2.6 é não linear, é necessário linearizá-lo para soluções como EKF-SLAM e seus derivados, como SEIF-SLAM. Sua matriz jacobiana, $\mathbf{H}$ para a j-ésima *landmark*, descrita na Equação 2.10, ela é composta pelos jacobianos $\mathbf{H}_\mathrm{r}$, calculado com relação à pose do robô, e pelo jacobiano $\mathbf{H}_M$, calculado com relação à posição da *landmark* no vetor de estado.

$$\mathbf{H}_\mathrm{r}^j = \begin{bmatrix} \dfrac{d}{r^j}\left(\delta_x \sin\phi - \delta_y \cos\phi\right) & \dfrac{-\delta_x}{r^j} & \dfrac{-\delta_y}{r^j} \\ -\left(\dfrac{d}{[r^j]^2}\left(\delta_y \sin\phi + \delta_x \cos\phi\right) + 1\right) & \dfrac{\delta_y}{[r^j]^2} & \dfrac{-\delta_x}{[r^j]^2} \end{bmatrix} \tag{2.8}$$

$$\mathbf{H}_\mathrm{m}^j = \begin{bmatrix} \dfrac{\delta_x}{r^j} & \dfrac{\delta_y}{r^j} \\ \dfrac{-\delta_y}{[r^j]^2} & \dfrac{\delta_x}{[r^j]^2} \end{bmatrix} \tag{2.9}$$

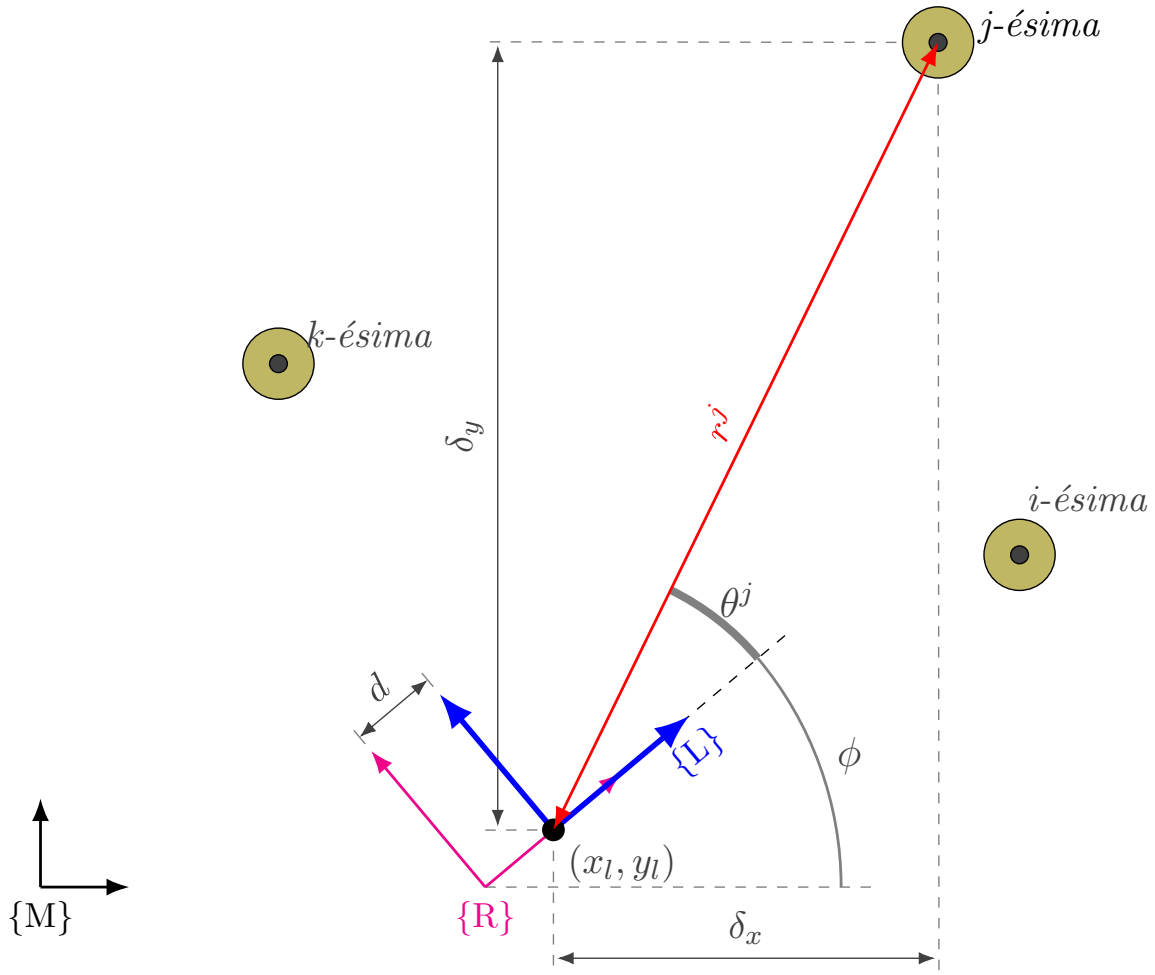


FIGURA 2.2 – Esquemático do modelo de medida Distância-Azimute. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. A medida (r^j, θ^j) se refere à i-ésima *landmark* no mapa.

$$\mathbf{H}^j(\mathbf{x}) = \begin{bmatrix} \mathbf{H}^j_{\mathrm{r}} & \mathbf{0}_{2\times 2} & \cdots & \mathbf{H}^j_{\mathrm{m}} & \cdots & \mathbf{0}_{2\times 2} \end{bmatrix} \tag{2.10}$$

2.2.2 Modelo de medida inverso Distância-Azimute

O modelo de medida descrito na Seção anterior é também conhecido como modelo de medida direto, pois descreve a medida que espera-se ler quando o sistema está em um dado estado. Mas também há o modelo de medida inverso, que informa sobre o estado a partir de uma medida. Esse modelo é útil durante o descobrimento de novas *landmarks*, pois ele dá meios para que suas posições sejam incorporadas ao vetor de estados. Na Figura 2.3 está representado o momento no qual o robô descobre a p-ésima *landmark* do ambiente.

O modelo de medida inverso $\boldsymbol{f}(\cdot,\cdot)$ é descrito na Equação 2.11. Como pode ser observado, assim como o modelo de medida "direto", o modelo de medida inverso é não linear

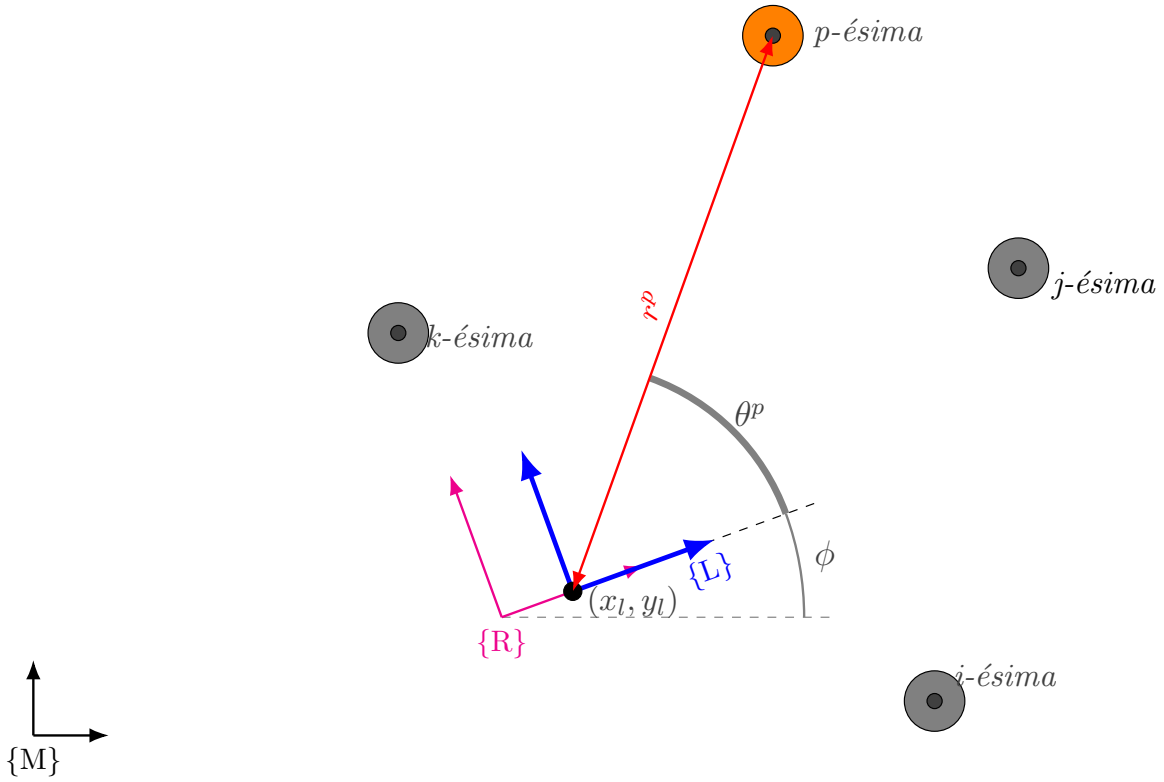


FIGURA 2.3 – Esquemático do modelo de medida Distância-Azimute. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. Os círculos representam *landmarks*, as conhecidas pelo robô (portanto presentes no vetor de estados), em cinza, e recém descobertas, em laranja. A p-ésima *landmark* acaba de ser encontrada pelo robô.

e requer linearização para ser utilizado com o clássico Filtro de Kalman Estendido (EKF) SLAM e seus algoritmos derivados, apresentados no Capítulo 3. Sua matriz jacobiana, $\mathbf{F}$, na Equação 2.14 é composta pelos jacobianos parciais em relação ao vetor de estados, $\mathbf{F}_X$, e à medida da nova landmark encontrada, $\mathbf{F}_Y$, mostrados nas Equações 2.12 e 2.13, respectivamente.

$$\boldsymbol{f}(\mathbf{x}, \mathbf{y}^p) = \begin{bmatrix} m_x^p \\ m_y^p \end{bmatrix} = \begin{bmatrix} x_l + r^p \cos(\phi + \theta^p) \\ y_l + r^p \sin(\phi + \theta^p) \end{bmatrix} \tag{2.11}$$

$$\mathbf{F}_X = \begin{bmatrix} \begin{bmatrix} -d\sin\phi - r^p \sin(\phi + \theta^p) & 1 & 0 \\ d\cos\phi + r^p \cos(\phi + \theta^p) & 0 & 1 \end{bmatrix} & \mathbf{0}_{2\times n-3} \end{bmatrix} \tag{2.12}$$

$$\mathbf{F}_Y = \begin{bmatrix} \cos(\phi + \theta^p) & -r^p \sin(\phi + \theta^p) \\ \sin(\phi + \theta^p) & r^p \cos(\phi + \theta^p) \end{bmatrix} \tag{2.13}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_X & \mathbf{F}_Y \end{bmatrix} \tag{2.14}$$

3 Aplicação de estimação ao problema SLAM (*Back-end*)

O Filtro de Kalman (KF) é uma técnica de estimação ótima para sistemas lineares com ruídos gaussianos. Nele, a função densidade de probabilidade do estado estimado é representada por uma gaussiana, Equação 3.1, parametrizada pelos momentos centrais média e covariância. O KF foi desenvolvido simultaneamente em 1958 por Peter Swerling e em 1960 por Rudolf Kalman (THRUN *et al.*, 2005, p. 40). Apesar de sua otimalidade ser garantida apenas para sistemas lineares, ele é aplicado em sistemas não lineares também. Para isso, é feita uma aproximação linear em torno da estimativa do estado atual do sistema, utilizando-se expansão por série de Taylor, e a premissa de que os termos de ordem maior ou igual a dois são desprezíveis.

$$p(\mathbf{x}) = \eta \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{3.1}$$

Essa técnica derivada do KF para sistemas não lineares é conhecida como Filtro de Kalman Estendido (EKF). O EKF é muito utilizado em aplicações reais, pois a grande maioria dos sistemas reais são não lineares, como o movimento de um robô diferencial, por exemplo. Além disso, o modelo de medida do sensor é, muitas vezes, uma função não linear do estado do sistema.

Neste capítulo, serão descritas as alterações necessárias no EKF clássico para que ele possa ser aplicado na resolução do problema SLAM, estimando a pose do robô e a posição das *landmarks*, o que é conhecido como EKF-SLAM e proposto por Smith e Cheeseman (1986). Além disso, também serão abordadas técnicas decorrentes do EKF-SLAM, como EIF-SLAM (Filtro de Informação Estendido) e SEIF-SLAM (Filtro de Informação Estendido Esparso) criado por Thrun *et al.* (2004b), sendo esta última a técnica de estimação utilizada neste trabalho.

3.1 Filtro de Kalman Estendido

Para que seja possível estimar o estado de um sistema utilizando-se KF, é necessário conhecer duas equações: a primeira, denominada modelo do sistema, modela a transição de estado do sistema a partir do estado anterior e da entrada aplicada, Equação 3.2; a segunda, chamada de modelo de medida, relaciona o estado do sistema com a medida, gerada pelo sensor, Equação 3.3.

$$\mathbf{x}_t = \boldsymbol{g}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \tag{3.2}$$

$$\mathbf{y}_t = \boldsymbol{h}(\mathbf{x}_t) + \boldsymbol{\delta}_t \tag{3.3}$$

Como o modelo do sistema $\boldsymbol{g}(\cdot,\cdot)$, e o modelo de medida $\boldsymbol{h}(\cdot)$ não são exatos, suas incertezas e erros de modelagem são aproximados por ruídos gaussianos $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ e $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$. Quando $\boldsymbol{g}(\cdot,\cdot)$ e/ou $\boldsymbol{h}(\cdot)$ não são lineares, o EKF pode ser utilizado para estimar o estado do sistema.

As Equações de 3.4 até 3.10 definem o EKF [1] para o sistema não linear acima. Sendo $\mathbf{G}_t$ o jacobiano do modelo do sistema no estado estimado atualizado prévio $\boldsymbol{\mu}_{t-1}$ e $\mathbf{H}_t$ é o jacobiano do modelo de medida no estado estimado predito $\overline{\boldsymbol{\mu}}_t$.

$$\overline{\boldsymbol{\mu}}_t = \boldsymbol{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) \tag{3.4}$$

$$\overline{\mathbf{P}}_t = \mathbf{G}_t \mathbf{P}_{t-1} \mathbf{G}_t^T + \mathbf{R}_t \tag{3.5}$$

$$\mathbf{z}_t = \mathbf{y}_t - \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) \tag{3.6}$$

$$\mathbf{Z}_t = \mathbf{H}_t \overline{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{Q}_t \tag{3.7}$$

$$\mathbf{K}_t = \overline{\mathbf{P}}_t \mathbf{H}_t^T \mathbf{Z}_t^{-1} \tag{3.8}$$

$$\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + \mathbf{K}_t \mathbf{z}_t \tag{3.9}$$

$$\mathbf{P}_t = \overline{\mathbf{P}}_t - \mathbf{K}_t \mathbf{Z}_t \mathbf{K}_t^T \tag{3.10}$$

em que:

- $\overline{\mathbf{P}}_t$ é a matriz de covariância do estado predito
- $\mathbf{z}_t$ é o vetor de inovação
- $\mathbf{Z}_t$ é a matriz de covariância da inovação, também conhecido por matriz de inovação
- $\mathbf{K}_t$ é o Ganho de Kalman
- $\mathbf{P}_t$ é a matriz de covariância do estado atualizado

[1] O leitor pode estranhar a Equação 3.10. Normalmente ela é escrita na forma $\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\overline{\mathbf{P}}_t$, porém, de acordo com Lewis *et al.* (2017, p. 73), a forma em 3.10 é uma alternativa melhor na presença de erros de arredondamento e é frequentemente utilizada em implementações de software.

Porém, a resolução do problema SLAM utilizando o EKF, não é uma aplicação direta das Equações acima. São necessárias algumas alterações, pois em SLAM o vetor de medidas tem tamanho variável. Esses detalhes e outras particularidades da aplicação do EKF em SLAM serão tratados na próxima Seção.

3.2 EKF-SLAM

Para aplicar o EKF na solução de SLAM, é necessário entender como o vetor de estados $\mathbf{x}$ é composto (aqui o subíndice t é omitido, pois não é importante para esta discussão). Como tanto a pose do robô quanto o mapa são estimados, o vetor de estados é composto por ambos como apresentado na Equação 3.11.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_\mathrm{r} \\ \mathbf{x}_\mathrm{m} \end{bmatrix} \tag{3.11}$$

O vetor $\mathbf{x}_\mathrm{m}$, que representa o mapa, é composto pela posição (x, y) das *landmarks* identificadas. Seu tamanho é variável e cresce à medida que o robô navega pelo ambiente e mede novas *landmarks*.

$$\mathbf{x}_\mathrm{m} = \begin{bmatrix} m_x^1 \\ m_y^1 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix} \tag{3.12}$$

Utilizando a definição do vetor de estados do EKF-SLAM acima (Equação 3.11), as próximas três Seções (3.2.1, 3.2.2 e 3.2.3) descrevem as alterações necessários e/ou desejáveis no EKF para sua aplicação em SLAM. O algoritmo completo pode ser encontrado no Apêndice A.1.

3.2.1 EKF-SLAM: Predição (Movimento do robô)

Em SLAM apenas uma parte do vetor de estados é variante no tempo, a pose do robô. Isso significa que apenas a porção $\mathbf{x}_\mathrm{r}$ é alterada pela entrada $\mathbf{u}$, logo o modelo do sistema consiste apenas no modelo de movimento do robô $\boldsymbol{g}_\mathrm{r}$ concatenado com as posições das *landmarks*:

$$\mathbf{x}_t = \begin{bmatrix} \boldsymbol{g}_\mathrm{r}(\mathbf{x}_{\mathrm{r},\,t-1}, \mathbf{u}_t) \\ \mathbf{x}_\mathrm{m} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_{\mathrm{r},\,t} \\ \mathbf{0} \end{bmatrix} \tag{3.13}$$

Portanto, o passo de predição do vetor de média do estado do EKF-SLAM torna-se:

$$\overline{\boldsymbol{\mu}}_t = \begin{bmatrix} \boldsymbol{g}_{\mathrm{r}}(\boldsymbol{\mu}_{\mathrm{r},\,t-1}, \mathbf{u}_t) \\ \boldsymbol{\mu}_{\mathrm{m}} \end{bmatrix} \tag{3.14}$$

Em termos de implementação, isso significa que apenas as componentes da pose e seus respectivos endereços na memória são modificados no vetor de estados. Dessa forma, o passo de predição do vetor de estados do EKF-SLAM 2D tem complexidade $\mathcal{O}(3)$ (constante), enquanto no EKF essa complexidade é $\mathcal{O}(n)$, onde n é o tamanho do vetor de estados.

A matriz de covariância, $\mathbf{P}$, também é parcialmente atualizada, pois o jacobiano do sistema na Equação 3.13 possui forma esparsa:

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.15}$$

Então, a equação do erro de predição do EKF, em 3.5, torna-se:

$$\begin{aligned} \overline{\mathbf{P}}_t &= \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{P}_{t-1} \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{R}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{t-1}^{\mathrm{rr}} & \mathbf{P}_{t-1}^{\mathrm{rm}} \\ \mathbf{P}_{t-1}^{\mathrm{mr}} & \mathbf{P}_{\mathrm{mm}} \end{bmatrix} \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{R}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} \mathbf{P}_{t-1}^{\mathrm{rr}} \left[\mathbf{G}_t^{\mathrm{r}}\right]^T + \mathbf{R}_t^{\mathrm{r}} & \mathbf{G}_t^{\mathrm{r}} \mathbf{P}_{t-1}^{\mathrm{rm}} \\ \mathbf{P}_{t-1}^{\mathrm{mr}} \left[\mathbf{G}_t^{\mathrm{r}}\right]^T & \mathbf{P}_{\mathrm{mm}} \end{bmatrix} \end{aligned} \tag{3.16}$$

A complexidade dessa operação é da ordem de $\mathcal{O}(n)$ (vide nota[2]) por conta do termo $\mathbf{G}_t^{\mathrm{r}} \mathbf{P}_{t-1}^{\mathrm{rm}}$, enquanto no caso geral do EKF em que o jacobiano $\mathbf{G}_t$ é denso, essa complexidade é $\mathcal{O}(n^{2.376})$, que é a complexidade da multiplicação de matrizes $n \times n$ (COPPERSMITH; WINOGRAD, 1990). A Figura 3.1 ilustra as porções do vetor de estados e da matriz de covariância modificadas no passo de predição do EKF-SLAM.

[2]Como discutido no Apêndice A.2, apesar da multiplicação de matrizes possuir complexidade assintótica cúbica no pior caso, como a matriz $\mathbf{G}_t^{\mathrm{r}}$ possui dimensão 3×3 e a matriz $\mathbf{P}_{t-1}^{\mathrm{rm}}$ possui dimensão $3 \times (n-3)$ (onde n é o tamanho do vetor de estados) há apenas uma dimensão variável (a quantidade de colunas da matriz $\mathbf{P}_{t-1}^{\mathrm{rm}}$) e portanto a complexidade se torna $\mathcal{O}(n)$.

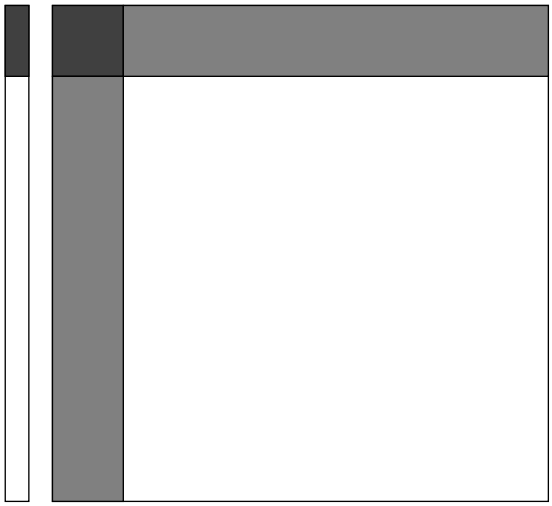


FIGURA 3.1 – Partes modificadas do vetor de média do estado e da matriz de covariância durante o movimento do robô. O vetor de média do estado é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As partes modificadas, em tons de cinza, correspondem ao estado do robô $\boldsymbol{\mu}_{\mathrm{r}}$ e sua autocovariância $\mathbf{P}_{\mathrm{rr}}$ (cinza escuro), e às covariâncias cruzadas, $\mathbf{P}_{\mathrm{rm}}$ e $\mathbf{P}_{\mathrm{mr}}$, entre o robô e o mapa (cinza claro). Note que as partes correspondentes ao mapa, $\boldsymbol{\mu}_{\mathrm{m}}$ e $\mathbf{P}_{\mathrm{mm}}$, permanecem inalteradas (branco). Adaptado de SOLÀ (2014, p. 10).

3.2.2 EKF-SLAM: Atualização

Assim como na predição, no passo de atualização há algumas particularidades que devem ser levadas em conta no EKF-SLAM. Ao contrário de um sistema convencional, em SLAM o vetor de medidas é variável, seu tamanho depende da quantidade de *landmarks* que vão sendo avistadas pelo robô enquanto ele navega pelo ambiente. Ou seja, no EKF-SLAM o vetor de medidas é sempre "incompleto". Normalmente, a inovação $\mathbf{z}_t$ é computada para cada medida de maneira individual e é denotada por $\mathbf{z}_t^j$.

$$\mathbf{z}_t^j = \mathbf{y}_t^j - \boldsymbol{h}^j(\overline{\boldsymbol{\mu}}_t) \tag{3.17}$$

Além disso, como o jacobiano do modelo de medida na Equação 2.10 é esparso, a covariância da inovação é computada por:

$$\mathbf{Z}_t^j = \begin{bmatrix} \mathbf{H}_{\mathrm{r}}^j & \mathbf{H}_{\mathrm{m}}^j \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_{\mathrm{rr}} & \overline{\mathbf{P}}_{\mathrm{rm^j}} \\ \overline{\mathbf{P}}_{\mathrm{rm^j}}^T & \overline{\mathbf{P}}_{\mathrm{m^j m^j}} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{\mathrm{r}}^j \\ \mathbf{H}_{\mathrm{m}}^j \end{bmatrix} + \mathbf{Q}_t \tag{3.18}$$

As dimensões do vetor inovação e das matrizes na equação acima são constantes e dependem apenas da dimensão da pose do robô e da dimensão da medida. Portanto, aqui as complexidades ao se computar a inovação $\mathbf{z}_t^j$ e de sua covariância $\mathbf{Z}_t^j$ são constantes, enquanto no EKF essas complexidades são $\mathcal{O}(m)$ e $\mathcal{O}(nm^2)$, respectivamente, onde m é o tamanho do vetor de medidas. Cabe enfatizar que, aqui, esse cálculo de complexidade constante deve ser repetido para cada observação presente no vetor de medidas, ou seja, no

EKF-SLAM as computações da inovação e sua covariância possuem complexidade linear na quantidade de medidas obtidas.

A computação do ganho do filtro Kalman $\mathbf{K}_t$ também é influenciado pelo tamanho constante da matriz de covariância da inovação (2×2, no caso deste trabalho), Equação 3.18, e pela esparsidade do jacobiano do modelo de medida, na Equação 2.10. Ademais, se todos os cálculos triviais de multiplicação por zero não forem feitos, a complexidade na computação de $\mathbf{K}_t^j$ se torna $\mathcal{O}(n)$ no EKF-SLAM.

Por fim, as complexidades da atualização e sua matriz de covariância, Equações 3.9 e 3.10, são $\mathcal{O}(n)$ e $\mathcal{O}(n^2)$, respectivamente. A Figura 3.2 mostra as porções do vetor de estados e da matriz de covariância do sistema SLAM utilizadas na computação da inovação e de sua matriz de covariância.

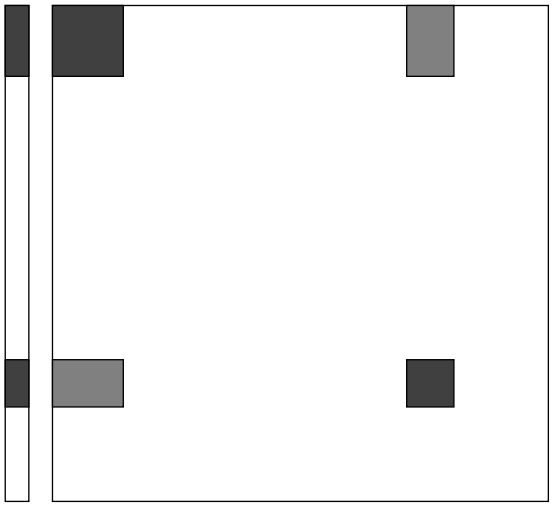


FIGURA 3.2 – Partes utilizadas do vetor de média do estado e da matriz de covariância durante a computação da inovação, quando uma *landmark* é observada. O vetor de média do estado é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As porções utilizadas, em tons de cinza, correspondem ao estado do robô $\boldsymbol{\mu}_{\mathrm{r}}$ e à posição da *landmark* $\mathbf{m}^j$, suas autocovariâncias $\mathbf{P}_{\mathrm{rr}}$ e $\mathbf{P}_{\mathrm{m^j m^j}}$ (cinza escuro), covariâncias cruzadas, $\mathbf{P}_{\mathrm{rm^j}}$ e $\mathbf{P}_{\mathrm{m^j r}}$, entre o robô e a j-ésima *landmark* (cinza claro). Adaptado de (SOLÀ, 2014, p. 8).

A Figura 3.3 deixa claro que todos os elementos do vetor de média do estado e da matriz de covariância são atualizados pelas Equações 3.9 e 3.10, mesmo o cálculo da inovação sendo esparso. Isso ocorre porque no EKF todas as *landmarks* são correlacionadas, mesmo que muitas dessas correlações sejam pouco significativas. Essa correlação "fraca" será explorada pelo Filtro de Informação Estendido Esparso, a fim de obter-se um algoritmo de estimação mais eficiente.

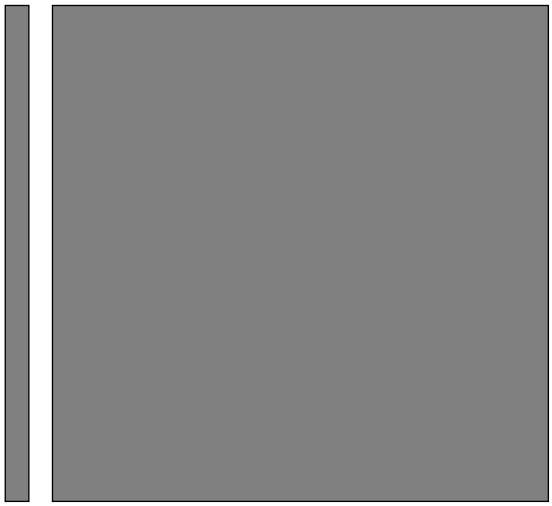


FIGURA 3.3 – O vetor de média do estado e a matriz de covariância são completamente atualizados durante a observação de uma *landmark*. Retirado de (SOLÀ, 2014, p. 8).

3.2.3 EKF-SLAM: Inserção de *landmark* (aumento do vetor de estados)

Nas Seções anteriores, 3.2.1 e 3.2.2, foram tratadas as diferenças do EKF-SLAM para o EKF, nas etapas/operações já conhecidas pelo usuário comum do EKF, etapas de predição e atualização. No entanto, em EKF-SLAM uma nova operação aparece: a etapa de inserção de *landmark*. Ela ocorre quando o robô observa uma *landmark* que ainda não está no mapa, $\mathbf{x}_{\mathrm{m}}$, e, portanto, não é possível calcular a inovação na Equação 3.17. Nesse caso, a nova *landmark* deve ser adicionada ao vetor de média do estado e à matriz de covariância, aumentando a dimensão do sistema.

Para adicionar uma nova *landmark* ao vetor de estado, será definida a função $\boldsymbol{\sigma}(\cdot,\cdot)$; ela gera um novo vetor de estados que é resultado da concatenação do vetor atual com a posição da nova *landmark* computada segundo o modelo de medida inverso descrito na Equação 2.11 da Seção 2.2.2, a partir da leitura $\mathbf{y}^j$.

$$\boldsymbol{\sigma}(\mathbf{x}_t, \mathbf{y}^j) = \begin{bmatrix} \mathbf{x}_t \\ \boldsymbol{f}(\mathbf{x}_t, \mathbf{y}^j) \end{bmatrix} \tag{3.19}$$

Porém, não basta apenas adicionar a nova *landmark* no vetor de estados, é necessário adicioná-la também na matriz de covariância. Quando o robô observa uma nova *landmark*, é esperado que o erro de estimação da posição dessa nova *landmark* seja influenciado pelo erro da pose do robô, no momento da leitura, e pelo erro de medição do sensor.

Inicializar a covariância da nova *landmark* com ∞ (ou números muito grandes), como indicado em (THRUN *et al.*, 2005, p. 317), pode ser inadequado. Portanto, devemos calcular o erro, $\boldsymbol{\alpha}$, da nova estimativa do vetor aumentado, de maneira análoga à forma

como é feita nos passos de predição e atualização do EKF.

$$\begin{aligned}
\boldsymbol{\alpha} &= \mathbf{x}_t^* - \boldsymbol{\mu}_t^* \\
&= \begin{bmatrix} \mathbf{x}_t \\ \boldsymbol{f}(\mathbf{x}_t, \mathbf{y}) \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_t \\ \boldsymbol{f}(\boldsymbol{\mu}_t, \mathbf{y}^j) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_t - \boldsymbol{\mu}_t \\ \boldsymbol{f}(\mathbf{x}_t, \mathbf{y}) - \boldsymbol{f}(\boldsymbol{\mu}_t, \mathbf{y}^j) \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\eta}_t \\ \cancel{\boldsymbol{f}(\boldsymbol{\mu}_t, \mathbf{y}^j)} + \mathbf{F}_X \boldsymbol{\eta}_t + \mathbf{F}_Y \boldsymbol{\delta} - \cancel{\boldsymbol{f}(\boldsymbol{\mu}_t, \mathbf{y}^j)} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\eta}_t \\ \mathbf{F}_X \boldsymbol{\eta}_t + \mathbf{F}_Y \boldsymbol{\delta} \end{bmatrix} \text{ em que } \boldsymbol{\eta}_t = \mathbf{x}_t - \boldsymbol{\mu}_t \text{ e } \boldsymbol{\delta} = \mathbf{y} - \mathbf{y}^j
\end{aligned} \tag{3.20}$$

A matriz de covariância do sistema aumentado, $\mathbf{P}_t^*$, é obtida por:

$$\begin{aligned}
\mathbf{P}_t^* &= \mathbb{E}\left[\boldsymbol{\alpha}\,\boldsymbol{\alpha}^T\right] \\
&= \begin{bmatrix} \mathbf{P}_t & \mathbf{P}_t \mathbf{F}_X^T \\ \mathbf{F}_X \mathbf{P}_t & \mathbf{F}_X \mathbf{P}_t \mathbf{F}_X^T + \mathbf{F}_Y \mathbf{Q} \mathbf{F}_Y^T \end{bmatrix}
\end{aligned} \tag{3.21}$$

Portanto, a matriz de covariância do sistema aumentado é a matriz de covariância do sistema antes da inserção da nova *landmark*, concatenadas as covariâncias cruzadas $\mathbf{P}_t \mathbf{F}_X^T$ e $\mathbf{F}_X \mathbf{P}_t$, e a covariância $\mathbf{F}_X \mathbf{P}_t \mathbf{F}_X^T + \mathbf{F}_Y \mathbf{Q} \mathbf{F}_Y^T$ da nova *landmark* inserida no mapa.

Vale notar que o jacobiano $\mathbf{F}_X$ descrito na Equação 2.12 é esparso; logo, a complexidade de $\mathbf{P}_t \mathbf{F}_X^T$ pode ser reduzida de $\mathcal{O}(n^2)$ para $\mathcal{O}(n)$ se as computações envolvendo produtos com o elemento nulo forem ignoradas. Portanto, a operação de inserção de *landmark* tem custo $\mathcal{O}(n)$. A Figura 3.4 mostra o vetor de média do estado e a matriz de covariância com as novas inserções destacadas.

3.3 Filtro de Informação Estendido (EIF)

O Filtro de Kalman Estendido, apresentado na Seção 3.1 utiliza a parametrização de momentos (Equação 3.1) para representar a distribuição de probabilidade gaussiana. Já o Filtro de Informação utiliza a chamada representação canônica, composta pelo vetor de informação, $\boldsymbol{\xi}$, e pela matriz de informação, $\boldsymbol{\Omega}$. Definidos a seguir:

$$\boldsymbol{\xi} = \mathbf{P}^{-1} \boldsymbol{\mu} \tag{3.22}$$

$$\boldsymbol{\Omega} = \mathbf{P}^{-1} \tag{3.23}$$

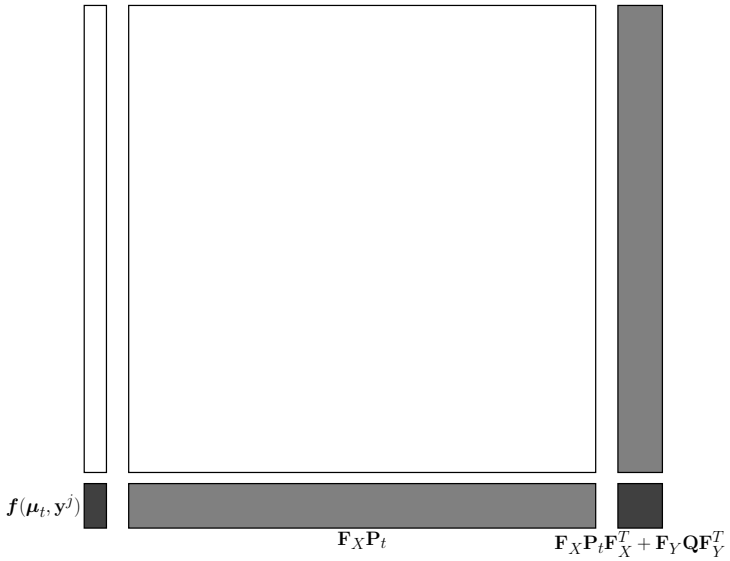


FIGURA 3.4 – Vetor de média do estado e matriz de covariância aumentados após inserção de nova *landmark*. As partes adicionadas, em cinza, correspondem às covariâncias cruzadas entre a nova landmark e o vetor de estados anterior (cinza claro), e à média da nova *landmark* e sua covariância (cinza escuro). Adaptado de (SOLÀ, 2014, p. 11).

Com as parametrizações acima, o EKF pode ser reescrito na forma do Filtro de Informação Estendido (THRUN *et al.*, 2005, p. 76):

$$\boldsymbol{\mu}_{t-1} = \boldsymbol{\Omega}_{t-1}^{-1}\boldsymbol{\xi}_{t-1} \tag{3.24}$$

$$\overline{\boldsymbol{\Omega}}_t = (\mathbf{G}_t\boldsymbol{\Omega}_{t-1}^{-1}\mathbf{G}_t^T + \mathbf{R}_t)^{-1} \tag{3.25}$$

$$\overline{\boldsymbol{\mu}}_t = \boldsymbol{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) \tag{3.26}$$

$$\overline{\boldsymbol{\xi}}_t = \overline{\boldsymbol{\Omega}}_t\,\overline{\boldsymbol{\mu}}_t \tag{3.27}$$

$$\boldsymbol{\Omega}_t = \overline{\boldsymbol{\Omega}}_t + \mathbf{H}_t^T\mathbf{Q}_t^{-1}\mathbf{H}_t \tag{3.28}$$

$$\boldsymbol{\xi}_t = \overline{\boldsymbol{\xi}}_t + \mathbf{H}_t^T\mathbf{Q}_t^{-1}\left[\mathbf{y}_t - \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + \mathbf{H}_t\overline{\boldsymbol{\mu}}_t\right] \tag{3.29}$$

em que,

- $\overline{\boldsymbol{\Omega}}_t$ é a matriz de informação predita
- $\overline{\boldsymbol{\xi}}_t$ é o vetor de informação predito

e a distribuição de probabilidade na Equação 3.1 pode ser reescrita da seguinte forma:

$$p\left(\mathbf{x}\right) = \eta \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}\mathbf{x} + \mathbf{x}^T\boldsymbol{\xi}\right) \tag{3.30}$$

como demonstrado em (THRUN *et al.*, 2005, p. 74 e 75).

Uma vantagem do Filtro de Informação é que ele tende a ser numericamente mais estável. Além disso, representar alto nível de incerteza é numericamente mais seguro

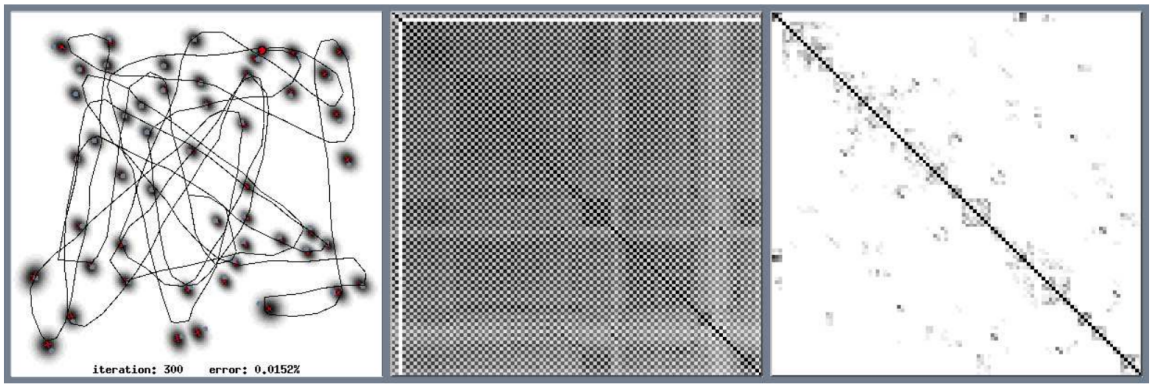


FIGURA 3.5 – Comparação entre a matriz de covariância e a matriz de informação de um sistema com 50 *landmarks*. À esquerda, mapa de *landmarks* do sistema; ao centro, a matriz de covariância do EKF-SLAM e, à direita, a matriz de informação. Retirado de Thrun *et al.* (2004a).

quando comparado com o Filtro de Kalman; aqui basta definir $\mathbf{\Omega} = \mathbf{0}$, enquanto no KF é necessário utilizar valores muito grandes na matriz de covariância. Outro aspecto interessante do IF é sua naturalidade para sistemas multi-robôs, nos quais a informação é coletada de maneira descentralizada (THRUN *et al.*, 2005, p. 78).

Porém, as principais desvantagens do EIF são a necessidade da recuperação da média em 3.24 e a predição da matriz de informação em 3.25, pois ambas operações envolvem a inversão da matriz de informação, cuja complexidade é $\mathcal{O}(n^{2.376})$. Embora, no EKF também seja necessário inverter a matriz de covariância da inovação, $\mathbf{Z}$, essa usualmente possui dimensão menor que a matriz de informação. Em geral, para sistemas de grande dimensão, acredita-se que o EIF é computacionalmente inferior, do ponto de vista de tempo de execução, em relação ao EKF. Por esse motivo, ele é menos utilizado que o EKF, na prática (THRUN *et al.*, 2005, p. 78).

No entanto, ao empregar o EIF no problema SLAM, nota-se que grande parte dos blocos fora da diagonal principal da matriz de informação são quase nulos, ou seja, agregam pouca informação ao sistema, como mostrado na imagem da direita na Figura 3.5. Isso se deve à estrutura do problema SLAM, pois grande parte das correlações entre *landmarks* (portanto fora da diagonal principal) são propagadas pela incerteza da pose do robô quando as *landmarks* são observadas. Apenas *landmarks* dentro de uma mesma vizinhança são observadas juntas, resultando em alta correlação.

Esse aspecto é explorado pelo Filtro de Informação Estendido Esparso (SEIF); por meio de aproximações o SEIF mantém a matriz de informação bloco-diagonalizada, aproximando os blocos fora da diagonal para zero. Isso leva o SEIF a otimizar operações e ter complexidade de tempo constante, enquanto mantém uso linear de memória. A próxima Seção descreve o SEIF e seus detalhes de implementação.

3.4 SEIF-SLAM

Essa Seção descreve o Filtro de Informação Estendido Esparso (SEIF) e como ele enfrenta as principais desvantagens do EIF clássico no contexto de SLAM. Será mostrado como ele mantém complexidade linear no uso de memória e complexidade de tempo constante nos passos de predição e atualização, independentemente do número de *landmarks* no mapa/ambiente.

Para tal, o SEIF mantém a matriz de informação com formato próximo ao de uma matriz bloco-diagonal por meio do uso de *landmarks* ativas e inativas, que serão descritas mais adiante. Além disso, a recuperação da média, na Equação 3.24, é modelada como um problema de otimização. As Seções a seguir são baseadas na discussão em (THRUN *et al.*, 2005, Capítulo 12.4).

3.4.1 SEIF-SLAM: Landmarks ativas e inativas

A diferença fundamental entre o SEIF-SLAM e o EIF-SLAM está na estrutura da matriz de informação, no SEIF ela é esparsa, ou melhor, *esparsificada*. Enquanto no EKF-SLAM/EIF-SLAM temos que $Cov\left(\mathbf{m}^j, \mathbf{m}^k\right) \neq \mathbf{0}, \forall \{j, k\}$, ou seja, que as posições de todas as *landmarks* são correlacionadas, o SEIF tenta eliminar a maioria dessas correlações a fim de obter uma matriz de informação esparsa.

Para isso, ele mantém dois conjuntos de landmarks $\mathbf{m}_t^+$ e $\mathbf{m}_t^-$, cuja inter-relação está descrita na Equação 3.31. O conjunto $\mathbf{m}_t^+$ é composto pelas *landmarks* ativas, que estão "ligadas" ao robô no tempo t, ou seja, $Cov\left(\mathbf{x}_{R,t}, \mathbf{m}_t^+\right) \neq 0$. Já o conjunto $\mathbf{m}_t^-$ é formado pelas *landmarks* inativas, que não estão correlacionadas com a pose atual do robô, ou seja, $Cov\left(\mathbf{x}_{R,t}, \mathbf{m}_t^-\right) = 0$.

$$\begin{cases} \mathbf{m}_t^+ \cup \mathbf{m}_t^- & = \mathbf{x}_{\mathrm{m}} \\ \mathbf{m}_t^+ \cap \mathbf{m}_t^- & = \emptyset \end{cases} \tag{3.31}$$

Uma das consequências desse esquema é que as *landmarks* não são globalmente correlacionadas entre si, como ocorre no EKF-SLAM e EIF-SLAM. Na verdade, aqui, elas são localmente correlacionadas com sua vizinhança, em que vizinhança é definido como o conjunto de *landmarks* presentes em $\mathbf{m}_t^+$ concomitantemente. Portanto, a inovação de uma *landmark* observada afeta apenas a pose do robô e de sua vizinhança, ao contrário do que acontece no EKF/EIF em que a inovação de uma *landmark* afeta todo o sistema.

Embora o conceito dos conjuntos de *landmarks* ativas e passivas seja introduzido por Thrun *et al.* (2004b), não é dada uma descrição detalhada de política para construir e manter esses conjuntos ao longo do tempo. O único trabalho encontrado que descreve uma política para esses conjuntos consiste em uma proposta do SEIF denotada Filtro de

TABELA 3.1 – Exemplo da política do conjunto de *landmarks* ativas implementada, com cardinalidade $k = 3$.

Tempo	Landmarks observadas	$\mathbf{m}^+$	$\mathbf{m}^-$
t_0	$\{\}$	$\{\}$	$\{\}$
t_1	$\{1, 2, 3\}$	$\{1, 2, 3\}$	$\{\}$
t_2	$\{2, 3, 4\}$	$\{2, 3, 4\}$	$\{1\}$
t_3	$\{3, 4\}$	$\{2, 3, 4\}$	$\{1\}$
t_4	$\{1, 4, 5\}$	$\{4, 1, 5\}$	$\{2, 3\}$
t_5	$\{1, 6, 7\}$	$\{1, 6, 7\}$	$\{2, 3, 4, 5\}$

Informação Estendido Exato (ESEIF) (WALTER *et al.*, 2007).

No entanto, neste trabalho é implementada uma política diferente de Walter *et al.* (2007); aqui o conjunto $\mathbf{m}_t^+$ contém as k últimas *landmarks* observadas até o instante t, em que k é a cardinalidade do conjunto. As *landmarks* vão entrando e saindo desse conjunto conforme o robô navega no ambiente e novas *landmarks* vão sendo observadas enquanto outras deixam de sê-lo, conforme exemplo na Tabela 3.1.

Note que o conjunto de *landmarks* ativas $\mathbf{m}^+$ em um dado instante t não é necessariamente composto pelas *landmarks* observadas neste instante, como evidenciado na linha t_3 da Tabela 3.1. Quando um subconjunto de *landmarks* presentes em $\mathbf{m}^+$ foram inseridas dentro de um mesmo passo de execução e precisa-se escolher alguma para ser substituída, escolhe-se a de menor índice como mostrado na linha t_2 da Tabela 3.1.

Nas próximas Seções, ficará claro que a cardinalidade k definida para o conjunto de landmarks ativas limitará a quantidade de elementos longe da diagonal principal da matriz de informação, tornando-a esparsa. É essa característica que confere ao SEIF-SLAM a complexidade linear em memória e tempo constante de atualização e predição.

3.4.2 SEIF-SLAM: Passo de predição

As Seções que se seguem derivam as equações do passo de predição do SEIF-SLAM a partir das equações 3.25 a 3.27 de predição do EIF. A esparsidade da matriz de informação é usada como premissa para garantir o tempo de execução constante. A esparsificação em si será tratada mais adiante; por hora, vamos assumir que a matriz é esparsa.

Antes é importante relembrar que o jacobiano do sistema SLAM tem a seguinte forma:

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \qquad \text{(3.15 repetida)}$$

Além disso, vamos definir o ruído do modelo do sistema, $\mathbf{R}_t$, como:

$$\begin{aligned}\mathbf{R}_t &= \begin{bmatrix}\mathbf{R}_t^{\mathrm{r}} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & 0 & 0\cdots 0\\ 0 & 1 & 0 & 0\cdots 0\\ 0 & 0 & 1 & 0\cdots 0\end{bmatrix}^T \mathbf{R}_t^{\mathrm{r}} \begin{bmatrix}1 & 0 & 0 & 0\cdots 0\\ 0 & 1 & 0 & 0\cdots 0\\ 0 & 0 & 1 & 0\cdots 0\end{bmatrix}\\ &= \mathbf{M}_{x_{\mathrm{r}}}^T \mathbf{R}_t^{\mathrm{r}} \mathbf{M}_{x_{\mathrm{r}}}\end{aligned} \tag{3.32}$$

em que $\mathbf{M}$ é uma a matriz de projeção de um subconjunto dos elementos do vetor de estado estimado.

3.4.2.1 Predição da Matriz de Informação

Primeiro, vamos reescrever a Equação 3.25 em termos de $\mathbf{\Phi}_t$ e, $\mathbf{R}_t^r$:

$$\overline{\mathbf{\Omega}}_t = (\mathbf{\Phi}_t^{-1} + \mathbf{M}_{x_{\mathrm{r}}}^T \mathbf{R}_t^{\mathrm{r}} \mathbf{M}_{x_{\mathrm{r}}})^{-1} \tag{3.33}$$

Onde:

$$\begin{aligned}\mathbf{\Phi}_t &= (\mathbf{G}_t \mathbf{\Omega}_{t-1}^{-1} \mathbf{G}_t^T)^{-1}\\ &= \left[\mathbf{G}_t^T\right]^{-1} \mathbf{\Omega}_{t-1} \left[\mathbf{G}_t\right]^{-1}\end{aligned} \tag{3.34}$$

Aplicando o lema da inversão (Apêndice A.1) em 3.33, temos:

$$\begin{aligned}\overline{\mathbf{\Omega}}_t &= \mathbf{\Phi}_t - \mathbf{\Phi}_t \mathbf{M}_{x_{\mathrm{r}}}^T \left([\mathbf{R}_t^{\mathrm{r}}]^{-1} + \mathbf{M}_{x_{\mathrm{r}}} \mathbf{\Phi}_t \mathbf{M}_{x_{\mathrm{r}}}^T\right)^{-1} \mathbf{M}_{x_{\mathrm{r}}} \mathbf{\Phi}_t\\ &= \mathbf{\Phi}_t - \boldsymbol{\kappa}_t\end{aligned} \tag{3.35}$$

Para calcularmos

$$\boldsymbol{\kappa}_t = \mathbf{\Phi}_t \mathbf{M}_{x_{\mathrm{r}}}^T \left([\mathbf{R}_t^{\mathrm{r}}]^{-1} + \mathbf{M}_{x_{\mathrm{r}}} \mathbf{\Phi}_t \mathbf{M}_{x_{\mathrm{r}}}^T\right)^{-1} \mathbf{M}_{x_{\mathrm{r}}} \mathbf{\Phi}_t \tag{3.36}$$

em tempo constante, temos que calcular $\mathbf{\Phi}_t$ em tempo constante a partir de $\mathbf{\Omega}_{t-1}$. Para isso, vamos representar $\mathbf{G}_t$ como na Equação 3.15, e calcular sua inversa:

$$\begin{aligned}
\mathbf{G}_t^{-1} &= \begin{bmatrix} \mathbf{G}_t^{\mathrm{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^{-1} \\
&= \begin{bmatrix} [\mathbf{G}_t^{\mathrm{r}}]^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} && \text{(inversão de matriz bloco diagonal)} \\
&= \mathbf{I}_n + \begin{bmatrix} [\mathbf{G}_t^{\mathrm{r}}]^{-1} - \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&= \mathbf{I}_n + \overbrace{\mathbf{M}_{x_{\mathrm{r}}}^T \left([\mathbf{G}_t^{\mathrm{r}}]^{-1} - \mathbf{I}_3 \right) \mathbf{M}_{x_{\mathrm{r}}}}^{\mathbf{\Psi}_t} \\
&= \mathbf{I} + \mathbf{\Psi}_t
\end{aligned} \tag{3.37}$$

Note que $\mathbf{\Psi}_t$ é uma matriz quadrada de dimensão n, onde apenas os elementos do bloco superior esquerdo 3×3 , são diferentes de zero. Usando a Equação 3.37 na Equação 3.34 temos:

$$\begin{aligned}
\mathbf{\Phi}_t &= \left[\mathbf{G}_t^T\right]^{-1} \mathbf{\Omega}_{t-1} \left[\mathbf{G}_t\right]^{-1} \\
&= \left(\mathbf{I} + \mathbf{\Psi}_t^T\right) \mathbf{\Omega}_{t-1} \left(\mathbf{I} + \mathbf{\Psi}_t\right) \\
&= \mathbf{\Omega}_{t-1} + \underbrace{\mathbf{\Psi}_t^T \mathbf{\Omega}_{t-1} + \mathbf{\Psi}_t^T \mathbf{\Omega}_{t-1} \mathbf{\Psi}_t + \mathbf{\Omega}_{t-1} \mathbf{\Psi}_t}_{\boldsymbol{\lambda}_t} \\
&= \mathbf{\Omega}_{t-1} + \boldsymbol{\lambda}_t
\end{aligned} \tag{3.38}$$

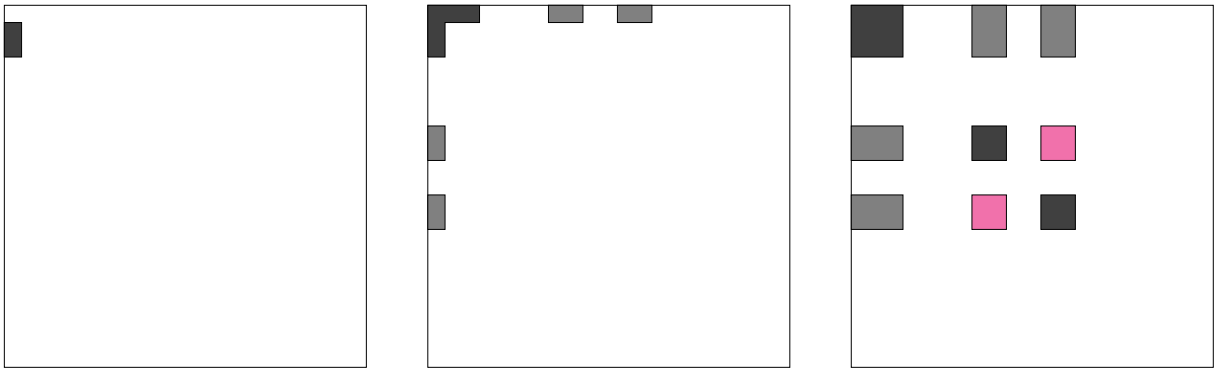
Como $\mathbf{\Psi}_t$ é esparsa e com quantidade invariante no tempo de elementos não nulos, $\boldsymbol{\lambda}_t$ também será, pois ambas dependem apenas do modelo de movimento do robô e das covariâncias cruzadas da posição do robô com as posições das *landmarks* ativas, que são quantidades constantes.

Portanto, computar $\mathbf{\Phi}_t$ a partir de $\mathbf{\Omega}_{t-1}$ tem carga computacional constante, pois resulta da subtração dos elementos não nulos de $\boldsymbol{\lambda}_t$ de $\mathbf{\Omega}_{t-1}$. Na Figura 3.6 são representadas as matrizes $\mathbf{\Psi}_t, \boldsymbol{\lambda}_t, \boldsymbol{\kappa}_t$ obtidas durante a execução do SEIF-SLAM com cardinalidade do conjunto de *landmarks* ativas igual a dois.

A princípio, pode parecer que a quantidade de elementos não nulos é significativa em relação ao tamanho das matrizes. Porém, essa percepção se deve ao fato das matrizes representadas na Figura 3.6 serem pequenas, o tamanho delas foi escolhido de modo a facilitar a visualização.

3.4.2.2 Predição do vetor de informação

Abaixo é apresentada uma série de manipulações para que a predição do vetor de informação, na Equação 3.27, possa ser realizada em tempo constante no SEIF-SLAM.



(a) $\boldsymbol{\Psi}_t$ (b) $\boldsymbol{\lambda}_t$ (c) $\boldsymbol{\kappa}_t$

FIGURA 3.6 – Representação das matrizes $\boldsymbol{\Psi}_t, \boldsymbol{\lambda}_t, \boldsymbol{\kappa}_t$, necessárias para calcular a matriz de informação predita durante o movimento do robô. Os elementos nulos são representados em branco, e os não nulos em cinza/magenta. As matrizes acima pertencem a um sistema SEIF-SLAM de um robô diferencial e sensor laser do tipo LiDAR, com duas *landmarks* ativas, ou seja, $|\mathbf{m}^+| = 2$. A quantidade de elementos não nulos é uma constante dada em função do modelo de movimento do robô e do cardinalidade do conjunto $\mathbf{m}^+$, independentemente do tamanho do mapa. Neste momento a terceira e quinta *landmark* estavam ativas. Os blocos em magenta representa a informação cruzada, entre as *landmarks* em $\mathbf{m}^+$, gerada pelo movimento do robô.

Primeiro, vamos reescrever o modelo de movimento na Equação 2.2, como:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \boldsymbol{\delta}_t \tag{3.39}$$

Partindo da Equação 3.27 e utilizando 3.39 acima, temos:

$$\begin{aligned}
\overline{\boldsymbol{\xi}}_t &= \overline{\boldsymbol{\Omega}}_t \, \overline{\boldsymbol{\mu}}_t \\
&= \overline{\boldsymbol{\Omega}}_t \left(\boldsymbol{\mu}_{t-1} + \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \right) \\
&= \overline{\boldsymbol{\Omega}}_t \left(\boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \right) \\
&= \overline{\boldsymbol{\Omega}}_t \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \\
&= \left(\overline{\boldsymbol{\Omega}}_t + \underbrace{\boldsymbol{\Omega}_{t-1} - \boldsymbol{\Omega}_{t-1}}_{\mathbf{0}} + \underbrace{\boldsymbol{\Phi}_t - \boldsymbol{\Phi}_t}_{\mathbf{0}} \right) \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \\
&= \left(\underbrace{\overline{\boldsymbol{\Omega}}_t - \boldsymbol{\Phi}_t}_{-\boldsymbol{\kappa}_t} + \boldsymbol{\Omega}_{t-1} + \underbrace{\boldsymbol{\Phi}_t - \boldsymbol{\Omega}_{t-1}}_{\boldsymbol{\lambda}_t} \right) \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \\
&= \left(\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t + \boldsymbol{\Omega}_{t-1} \right) \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \\
&= \left(\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t \right) \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \boldsymbol{\Omega}_{t-1} \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t} \\
&= \left(\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t \right) \boldsymbol{\mu}_{t-1} + \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t}
\end{aligned} \tag{3.40}$$

Como $\boldsymbol{\lambda}_t$ e $\boldsymbol{\kappa}_t$ são ambas esparsas, o produto $\left(\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t \right) \boldsymbol{\mu}_{t-1}$ contém um número determinado de elementos não nulos e, portanto, é computado em tempo constante. O produto $\mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t}$ resulta em um vetor nulo exceto pelos primeiros três elementos, e ao multiplicá-lo pela matriz de informação predita, que também é esparsa, temos como resultado um

vetor esparso (THRUN *et al.*, 2005, p. 398). Portanto, é necessário um número constante de operações, que independe do tamanho do mapa, para calcular o vetor de informação predito [3]. O passo de predição do SEIF-SLAM é apresentado no Algoritmo 3.1.

Algoritmo 3.1 SEIF-SLAM passo de predição

1: **function** SEIF-SLAM-PREDICTION($\boldsymbol{\xi}_{t-1}, \boldsymbol{\mu}_{t-1}, \boldsymbol{\Omega}_{t-1}, \mathbf{u}_t$)
2: $\quad \boldsymbol{\Psi}_t \leftarrow \mathbf{M}_{x_\mathrm{r}}^T \left([\mathbf{G}_t^\mathrm{r}]^{-1} - \mathbf{I}_3 \right) \mathbf{M}_{x_\mathrm{r}}$
3: $\quad \boldsymbol{\lambda}_t \leftarrow \boldsymbol{\Psi}_t^T \boldsymbol{\Omega}_{t-1} + \boldsymbol{\Psi}_t^T \boldsymbol{\Omega}_{t-1} \boldsymbol{\Psi}_t + \boldsymbol{\Omega}_{t-1} \boldsymbol{\Psi}_t$
4: $\quad \boldsymbol{\Phi}_t \leftarrow \boldsymbol{\Omega}_{t-1} + \boldsymbol{\lambda}_t$
5: $\quad \boldsymbol{\kappa}_t \leftarrow \boldsymbol{\Phi}_t \mathbf{M}_{x_\mathrm{r}}^T \left([\mathbf{R}_t^\mathrm{r}]^{-1} + \mathbf{M}_{x_\mathrm{r}} \boldsymbol{\Phi}_t \mathbf{M}_{x_\mathrm{r}}^T \right)^{-1} \mathbf{M}_{x_\mathrm{r}} \boldsymbol{\Phi}_t$
6: $\quad \overline{\boldsymbol{\Omega}}_t \leftarrow \boldsymbol{\Phi}_t - \boldsymbol{\kappa}_t$
7: $\quad \overline{\boldsymbol{\xi}}_t \leftarrow (\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t) \boldsymbol{\mu}_{t-1} + \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t}$
8: $\quad \overline{\boldsymbol{\mu}}_t \leftarrow \boldsymbol{\mu}_{t-1} + \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t}$
9: $\quad$ **return** $\overline{\boldsymbol{\xi}}_t, \overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\Omega}}_t$
10: **end function**

3.4.3 SEIF-SLAM: Recuperação da média

Outro desafio a ser resolvido no SEIF-SLAM para manter invariante o tempo requerido para computar a predição e/ou a atualização é computar o vetor de média do estado, fundamental para as linearizações dos modelos de movimento e medida, e para computar o vetor de informação na etapa de esparsificação. Na Equação 3.24 do Filtro de Informação Estendido, essa computação usa a inversão da matriz de informação e, mesmo numa matriz esparsa como a do SEIF, o tempo requerido para essa operação não é constante.

Para contornar essa inversão, no SEIF esse passo é executado de uma forma completamente diferente: ele é modelado como um problema de otimização. Em (THRUN *et al.*, 2005, Cap. 12.6), mostra-se que o vetor de média do estado é o ponto de máximo da quadrática na representação canônica da Gaussiana Equação 3.30. Então, o vetor de

[3] Comumente a orientação do robô é normalizada para o intervalo $[-\pi, \pi[$ no vetor de média do estado; portanto é necessário ajustar alguns elementos do vetor de informação para que ele continue obedecendo a identidade $\boldsymbol{\xi} = \boldsymbol{\Omega}\boldsymbol{\mu}$. Porém, esse ajuste não viola o caráter constante do passo de atualização, pois a quantidade de elementos é fixa de acordo com $|\mathbf{m}^+|$.

média do estado atualizado $\boldsymbol{\mu}_t$ pode ser calculado como:

$$\begin{aligned}\boldsymbol{\mu}_t &= \underset{\boldsymbol{\mu}}{\text{argmax}}\ \exp\left(-\frac{1}{2}\,\boldsymbol{\mu}^T\boldsymbol{\Omega}_t\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\xi}_t\right) \\ &= \underset{\boldsymbol{\mu}}{\text{argmin}}\ -\left(-\frac{1}{2}\,\boldsymbol{\mu}^T\boldsymbol{\Omega}_t\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\xi}_t\right) \\ &= \underset{\boldsymbol{\mu}}{\text{argmin}}\ \frac{1}{2}\,\boldsymbol{\mu}^T\boldsymbol{\Omega}_t\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{\xi}_t\end{aligned} \tag{3.41}$$

A minimização acima pode ser realizada por qualquer algoritmo de minimização como gradiente descendente, gradiente conjugado, entre outros. Nesse trabalho foi utilizado a descida de coordenadas (THRUN *et al.*, 2004b, Sec. 3.3), apresentada no Algoritmo 3.2, que, apesar de não ser tão sofisticada quanto os algoritmos citados, resolveu a tarefa.

Em problemas de otimização, um bom palpite inicial, próximo da vizinhança do ponto de interesse, é fundamental para a convergência do algoritmo. No caso de SLAM, esperamos que o estado atualizado sempre esteja próximo do estado predito; portanto, o vetor $\overline{\boldsymbol{\mu}}_t$ é utilizado como ponto de partida da otimização.

Algoritmo 3.2 Etapa de recuperação da média no SEIF-SLAM

1: **function** SEIF-SLAM-RECUPERAÇÃO-MÉDIA($\overline{\boldsymbol{\mu}}_t, \boldsymbol{\xi}_t, \boldsymbol{\Omega}_t, \mathrm{K}$)
2: $\quad\boldsymbol{\mu}_t \leftarrow \overline{\boldsymbol{\mu}}_t$
3: $\quad$**for** $j \in \mathbf{m}^+$ **do**
4: $\quad\quad k \leftarrow 0$
5: $\quad\quad$**while** $k < \mathrm{K}$ **do**
6: $\quad\quad\quad \boldsymbol{\mu}_t^{m^j} \leftarrow (\mathbf{M}_{m^j}\boldsymbol{\Omega}_t\mathbf{M}_{m^j}^T)^{-1}\,\mathbf{M}_{m^j}\left[\boldsymbol{\xi}_t - \boldsymbol{\Omega}_t\overline{\boldsymbol{\mu}}_t + \boldsymbol{\Omega}_t\mathbf{M}_{m^j}^T\mathbf{M}_{m^j}\overline{\boldsymbol{\mu}}_t\right]$
7: $\quad\quad\quad \boldsymbol{\mu}_t^{x_\mathrm{r}} \leftarrow (\mathbf{M}_{x_\mathrm{r}}\boldsymbol{\Omega}_t\mathbf{M}_{x_\mathrm{r}}^T)^{-1}\,\mathbf{M}_{x_\mathrm{r}}\left[\boldsymbol{\xi}_t - \boldsymbol{\Omega}_t\overline{\boldsymbol{\mu}}_t + \boldsymbol{\Omega}_t\mathbf{M}_{x_\mathrm{r}}^T\mathbf{M}_{x_\mathrm{r}}\overline{\boldsymbol{\mu}}_t\right]$
8: $\quad\quad\quad k \leftarrow k + 1$
9: $\quad\quad$**end while**
10: $\quad$**end for**
11: $\quad$**return** $\boldsymbol{\mu}_t$
12: **end function**

Como pode ser notado na linha 3 do Algoritmo 3.2, dentre todas as *landmarks*, apenas as ativas são atualizadas, além da pose do robô, é claro. Isso se deve novamente à esparsidade da matriz de informação, pois a atualização de uma *landmark* só contribui para a estimação da posição de suas vizinhas e da pose do robô. Portanto, a complexidade da recuperação da média também é constante, dado que K (o número de repetições de otimização de cada coordenada) e a cardinalidade do conjunto $\mathbf{m}^+$ são ambos constantes.

3.4.4 SEIF-SLAM: Passo de atualização

Por hora, assim como foi feito com o EKF-SLAM na Equação 3.17 vamos assumir que é possível calcular a inovação $\mathbf{z}_t^j$ a partir da medida $\mathbf{y}_t^j$. É claro que na prática, é preciso antes uma etapa que estabeleça correspondência entre as medidas lidas e as *landmarks* existentes no vetor de estados. Essa correspondência é fundamental para utilizar a *landmark* $\mathbf{m}^i$ correta no cálculo do modelo de medida $\boldsymbol{h}(\cdot)$, Equação 2.6.

O Algoritmo 3.3 pressupõe que tais correspondências são conhecidas. A Seção 3.5 abordará uma maneira para estabelecer essas correspondências. Ao contrário dos outros passos, a esparsidade da matriz de informação não influencia nem altera nada em relação ao Filtro de Informação Estendido no passo de atualização. As equações de atualização nas linhas 3 e 4 abaixo, são as mesmas equações 3.28 e 3.29 do EIF.

Algoritmo 3.3 Etapa de atualização do SEIF-SLAM com associação conhecida

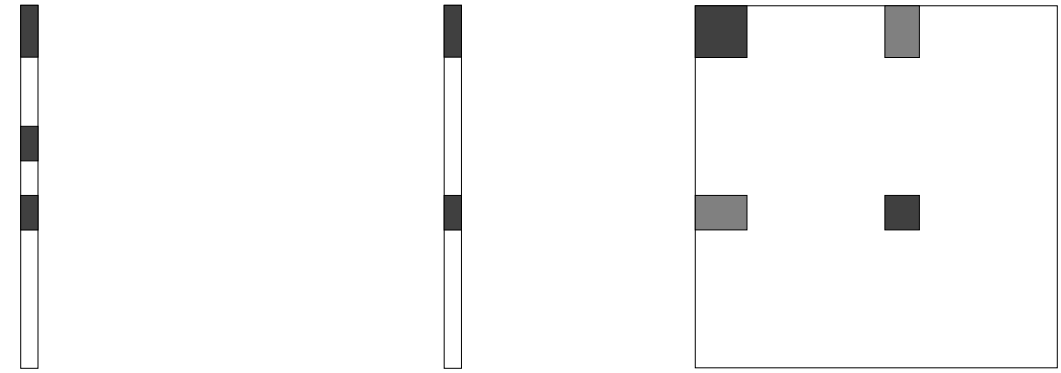
1: **function** SEIF-SLAM-ATUALIZAÇÃO($\overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\xi}}_t, \overline{\boldsymbol{\Omega}}_t, \mathbf{y}, j =$ índice da *landmark*)
2: $\quad \mathbf{z} \leftarrow \mathbf{y} - \boldsymbol{h}^j(\overline{\boldsymbol{\mu}}_t)$
3: $\quad \boldsymbol{\Omega}_t \leftarrow \overline{\boldsymbol{\Omega}}_t + \mathbf{H}_t^T \mathbf{Q}_t^{-1} \mathbf{H}_t$
4: $\quad \boldsymbol{\xi}_t \leftarrow \overline{\boldsymbol{\xi}}_t + \mathbf{H}_t^T \mathbf{Q}_t^{-1} \left[\mathbf{z} + \mathbf{H}_t \overline{\boldsymbol{\mu}}_t\right]$
5: $\quad \boldsymbol{\mu}_t \leftarrow$ SEIF-SLAM-RECUPERAÇÃO-MÉDIA($\overline{\boldsymbol{\mu}}_t, \boldsymbol{\xi}_t, \boldsymbol{\Omega}_t$, K)
6: $\quad$ **return** $\boldsymbol{\xi}_t, \boldsymbol{\mu}_t, \boldsymbol{\Omega}_t$
7: **end function**

A atualização acima altera apenas os elementos referentes ao robô e à j-ésima *landmark* tanto na matriz de informação quanto no vetor de informação, como é ilustrado na Figura 3.7. O SEIF herda o caráter "local" da atualização do EIF, e difere do EKF onde a atualização altera todos os elementos do vetor de média do estado e da matriz de covariâncias como foi mostrado na Figura 3.3.

3.4.5 SEIF-SLAM: Inserção de nova *landmark*

A etapa de atualização na Seção anterior ocorre quando uma medida é associada com uma *landmark* presente no mapa; porém, quando uma *landmark* é observada pela primeira vez, ela não é associada a nenhuma *landmark* prévia e, portanto, deve ser inserida nos vetores de estado e informação, e na matriz de informação.

Para derivar a inserção de *landmark* no EIF/SEIF, vamos partir dos resultados da inserção de *landmark* no EKF apresentados na Seção 3.2.3. Por comodidade, repetiremos



(a) $\boldsymbol{\mu}_t$ (b) $\boldsymbol{\xi}_t$ (c) $\boldsymbol{\Omega}_t$

FIGURA 3.7 – Representação dos vetores média e informação, $\boldsymbol{\mu}_t$ e $\boldsymbol{\xi}_t$ respectivamente, e da matriz de informação $\boldsymbol{\Omega}_t$ na etapa de atualização. O conjunto $\mathbf{m}^+$ possui tamanho 2, e a segunda *landmark* é observada. Note que no vetor de média do estado, a pose do robô e todas as *landmarks* ativas são atualizadas, enquanto que no vetor e matriz de informação, apenas a pose do robô e a posição da *landmark* observada são atualizados.

a Equação 3.21 com o resultado do aumento da matriz de covariância:

$$\mathbf{P}_t^* = \begin{bmatrix} \mathbf{P}_t & \mathbf{P}_t\mathbf{F}_X^T \\ \mathbf{F}_X\mathbf{P}_t & \mathbf{F}_X\mathbf{P}_t\mathbf{F}_X^T + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T \end{bmatrix} \tag{3.21 repetida}$$

e aplicando a equivalência $\boldsymbol{\Omega}^{-1} = \mathbf{P}$, pode-se reescrever a expressão acima como:

$$[\boldsymbol{\Omega}_t^*]^{-1} = \begin{bmatrix} \boldsymbol{\Omega}_t^{-1} & \boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T \\ \mathbf{F}_X\boldsymbol{\Omega}_t^{-1} & \mathbf{F}_X\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T \end{bmatrix} \tag{3.42}$$

invertendo ambos os lados utilizando o lema da inversão na forma de blocos (Anexo A.2) temos, por fim, que a matriz de informação aumentada é dada por:

$$\boldsymbol{\Omega}_t^* = \begin{bmatrix} \boldsymbol{\Omega}_t + \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X & -\mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right) \\ -\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X & \left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1} \end{bmatrix} \tag{3.43}$$

o desenvolvimento detalhado do resultado acima está no Apêndice B.1.

Note que, ao contrário da matriz de covariância aumentada do EKF na Equação 3.21 que não altera os elementos já existentes no filtro, a matriz de informação aumentada altera alguns elementos existentes; mais especificamente, ela altera a informação do robô.

Isso pode ser concluído a partir da observação da forma da matrix $\mathbf{F}_X$, Equação 2.12, nula em todos elementos exceto pelo primeiro bloco 2×3. Para evidenciar essa conclusão,

a Equação 3.43 é reescrita abaixo:

$$\boldsymbol{\Omega}_t^* = \begin{bmatrix} \boldsymbol{\Omega}_{RR,t} + \mathbf{F}_R^T \left(\left[\mathbf{F}_Y^T \right]^{-1} \mathbf{Q}^{-1} \mathbf{F}_Y^{-1} \right) \mathbf{F}_R & \boldsymbol{\Omega}_{RM,t} & -\mathbf{F}_R^T \left(\left[\mathbf{F}_Y^T \right]^{-1} \mathbf{Q}^{-1} \mathbf{F}_Y^{-1} \right) \\ \boldsymbol{\Omega}_{RM,t}^T & \boldsymbol{\Omega}_{MM,t} & \mathbf{0}_{(n-3)\times 2} \\ -\left(\left[\mathbf{F}_Y^T \right]^{-1} \mathbf{Q}^{-1} \mathbf{F}_Y^{-1} \right) \mathbf{F}_R & \mathbf{0}_{2\times(n-3)} & \left[\mathbf{F}_Y^T \right]^{-1} \mathbf{Q}^{-1} \mathbf{F}_Y^{-1} \end{bmatrix} \tag{3.44}$$

A partir da matriz de informação e do vetor de estado aumentados, pode-se calcular o vetor de informação aumentado:

$$\begin{aligned} \boldsymbol{\xi}_t^* &= \boldsymbol{\Omega}_t^* \boldsymbol{\mu}_t^* \\ &= \begin{bmatrix} \left[\boldsymbol{\Omega}_t^{\mathrm{rr}} \right]^* & \left[\boldsymbol{\Omega}_t^{\mathrm{rm}} \right]^* \end{bmatrix} \boldsymbol{\mu}_t^* \\ & \qquad \boldsymbol{\xi}_t^{\mathrm{m}} \\ & \begin{bmatrix} \left[\boldsymbol{\Omega}_t^{\mathrm{rm^j}} \right]^* & \left[\boldsymbol{\Omega}_t^{\mathrm{m^j m^j}} \right]^* \end{bmatrix} \boldsymbol{\mu}_t^* \end{bmatrix} \end{aligned} \tag{3.45}$$

Assim como na matriz de informação, os componentes do vetor de informação correspondentes ao robô também são alterados, além dos elementos da nova *landmark*, é claro. De novo, é o contrário do que acontece no EKF, onde apenas os elementos da nova *landmark* são alterados no vetor de estados. A Figura 3.8 representa as porções alteradas e/ou adicionadas à matriz de informação e ao vetor de informação.

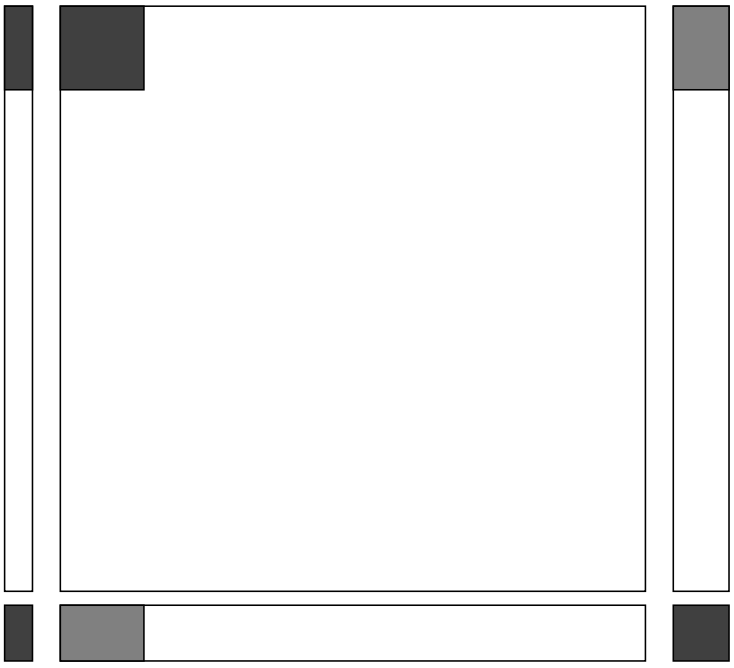


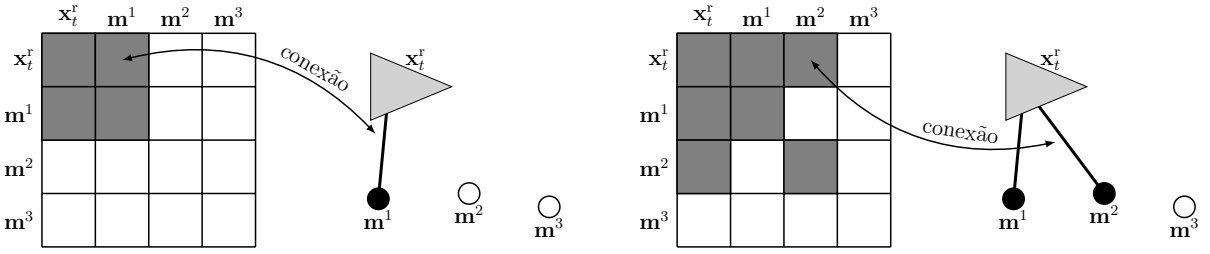
FIGURA 3.8 – Vetor e matriz de informação aumentados após inserção de nova *landmark*. As partes adicionadas, em cinza, correspondem à informação cruzada entre a nova landmark e o robô (cinza claro), e à média da nova *landmark* e do robô, e suas informações (cinza escuro).

Como pode ser observado na figura acima, a operação de inserção de *landmark* no SEIF possui complexidade constante $\mathcal{O}(1)$ em memória (o EKF possui complexidade linear); porém, assim como no EKF, a computação das novas porções possui complexidade $\mathcal{O}(n)$.

3.4.6 SEIF-SLAM: Esparsificação da matriz de informação

A esparsidade da matriz de informação foi utilizada como premissa em toda a discussão nas Seções 3.4.2, 3.4.3 e 3.4.4. Esta Seção mostrará como a operação de esparsificação é realizada, mas primeiro será ilustrado como a matriz de informação é preenchida nos passos de atualização e predição e como a esparsificação elimina a informação cruzada entre *landmarks*, mostrada na Figura 3.6c, tornando a matriz de informação esparsa.

Nas Figuras 3.9 até 3.13, é possível observar: a inserção de novas *landmarks* na matriz de informação no momento em que são observadas Figuras 3.9 e 3.11b; como as informações cruzadas entre *landmarks* surgem através do movimento do robô (passo de predição do filtro), Figuras 3.10b e 3.13b; a eliminação dessa informação cruzada entre a pose do robô e a posição de uma *landmark* através da esparsificação, Figura 3.12; como a eliminação de informação cruzada entre o robô e uma *landmark* impede a criação de novas conexões de movimento, Figura 3.13.

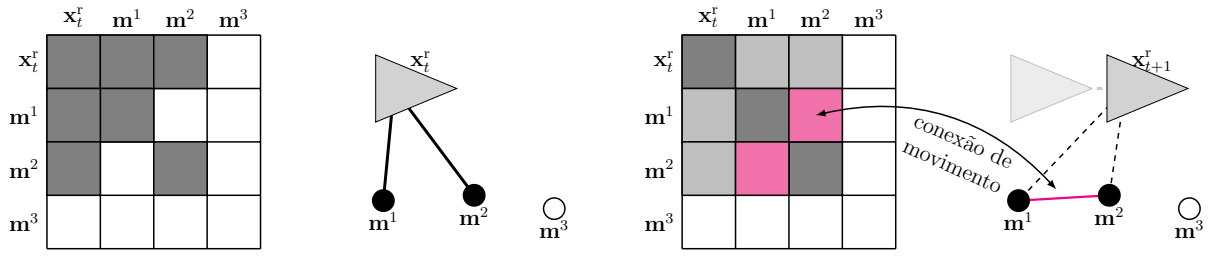


(a) *Landmark* $\mathbf{m}^1$ é observada (b) *Landmark* $\mathbf{m}^2$ também é observada

FIGURA 3.9 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks*, durante a observação das *landmarks* $\mathbf{m}^1$ e $\mathbf{m}^2$ no instante t. Os elementos não nulos da matriz de informação estão representados em cinza. Adaptado de (THRUN *et al.*, 2005, p. 389).

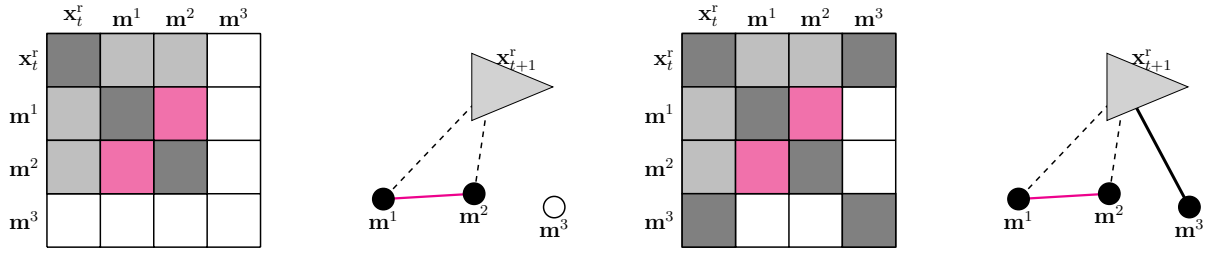
Através da sequência de esquemas apresentados acima é possível desenvolver a compreensão da construção da matriz de informação ao longo do tempo e como a esparsificação impede a criação de conexões entre *landmarks*. Como foi mostrado, a esparsificação consiste em "desativar"conexões entre *landmarks* e o robô, isto é, tornar a pose do robô condicionalmente independente da posição da *landmark* desativada. Neste exemplo, isso é obtido através da aproximação $p\left(\mathbf{x}_{t+1}^r \,|\, \mathbf{m}^1, \mathbf{m}^2\right) \approx p\left(\mathbf{x}_{t+1}^r \,|\, \mathbf{m}^2\right)$. Em distribuições multivariadas gaussianas, a independência condicional entre variáveis implica anular os elementos correspondentes da matriz de informação.

Para realizar a esparsificação da matriz de informação do SEIF-SLAM, vamos retomar os conjuntos $\mathbf{m}_t^+$ e $\mathbf{m}_t^-$ de *landmarks* ativas e inativas, respectivamente já definidos na



(a) Antes do movimento (b) Após o movimento

FIGURA 3.10 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks*, antes (esquerda) e depois (direita) do movimento do robô, entre os instantes t e $t+1$. A conexão de movimento gerada entre as *landmarks* $\mathbf{m}^1$ e $\mathbf{m}^2$ é mostrada em magenta, assim como os elementos correspondentes na matriz de informação. Os demais elementos não nulos estão representados em tons de cinza. Adaptado de (THRUN *et al.*, 2005, p. 389).



(a) (b) *Landmark* $\mathbf{m}^3$ é observada

FIGURA 3.11 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks* no instante $t+1$ durante a observação da *landmark* $\mathbf{m}^3$. Os elementos nulos da matriz de informação estão representados em branco. Adaptado de (THRUN *et al.*, 2005, p. 389).

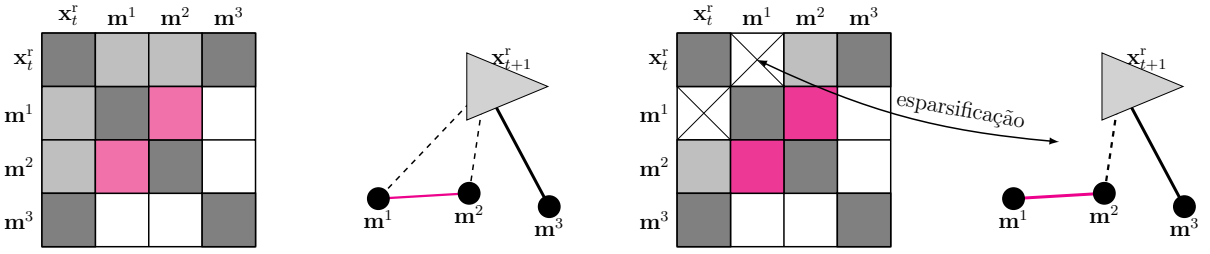
Seção 3.4.1. Também vamos subdividir $\mathbf{m}_t^+$ em:

$$\mathbf{m}_t^+ = \mathbf{m}_t^{++} \cup \mathbf{m}_t^{+-} \tag{3.46}$$

em que

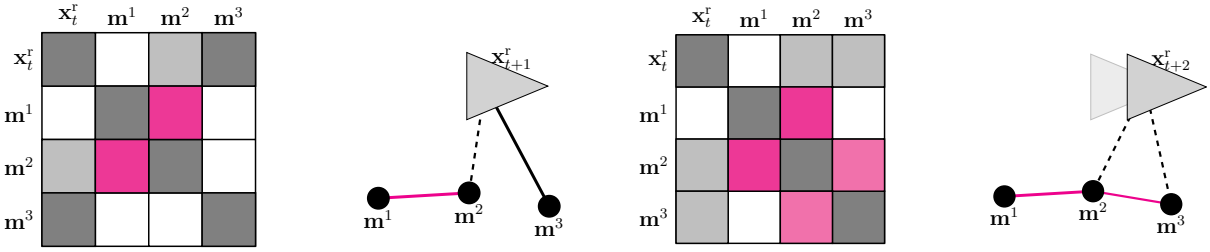
- $\mathbf{m}_t^{+-}$ é a porção do conjunto de *landmarks* ativas $\mathbf{m}_t^+$ que se tornarão inativas, ou seja, passarão a compor o conjunto $\mathbf{m}_t^-$ e terão suas informações cruzadas com o robô zeradas.
- $\mathbf{m}_t^{++}$ é a porção do conjunto de *landmarks* que continuarão ativas após a etapa de esparsificação.

A esparsificação consiste em quebrar a distribuição de probabilidade do problema SLAM aqui em 1.1 utilizando a regra do produto e aproximar a distribuição $p\left(\mathbf{x}_t^r \,|\, \mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^- = 0\right)$ pela marginal $p\left(\mathbf{x}_t^r \,|\, \mathbf{m}^{++}, \mathbf{m}^- = 0\right)$ (THRUN *et al.*, 2004a, Seção 3.3), como é mostrado



(a) Antes da esparsificação (b) Depois da esparsificação

FIGURA 3.12 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks* no instante $t+1$ antes e após a esparsificação da *landmark* $\mathbf{m}^1$. Os elementos nulos da matriz de informação estão representados em branco. Adaptado de (THRUN *et al.*, 2005, p. 389).



(a) Antes do movimento (b) Depois do movimento

FIGURA 3.13 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks* durante movimento entre os instantes t e $t+1$. Note que, devido à esparsificação da *landmark* $\mathbf{m}^1$, o movimento não gerou conexão de movimento entre as *landmarks* $\mathbf{m}^1$ e $\mathbf{m}^3$. Os elementos nulos da matriz de informação estão representados em branco.

abaixo:

$$
\begin{aligned}
p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) &= p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)\, p\left(\mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\
&= p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\
&= p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)\, p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\
&\approx p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}^{++}, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)\, p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\
&\approx \frac{p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)}{p\left(\mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)}\, p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)
\end{aligned}
\tag{3.47}
$$

A computação das matrizes de informações correspondentes às distribuições da última linha do desenvolvimento acima será com base na discussão em (THRUN *et al.*, 2005, p. 401) e na manipulação de distribuições gaussianas na forma canônica presentes no Anexo B. Primeiramente, vamos calcular a matriz $\mathbf{\Omega}_t^0$ correspondente a $p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m}^{+-}, \mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0\right)$; isso é feito extraindo os elementos correspondentes a todas as variáveis de estado exceto $\mathbf{m}^{-}$:

$$
\mathbf{\Omega}_t^0 = \mathbf{M}^T_{x_{\mathrm{r}}, m^{++}, m^{+-}} \mathbf{M}_{x_{\mathrm{r}}, m^{++}, m^{+-}}\, \mathbf{\Omega}_t\, \mathbf{M}^T_{x_{\mathrm{r}}, m^{++}, m^{+-}} \mathbf{M}_{x_{\mathrm{r}}, m^{++}, m^{+-}} \tag{3.48}
$$

A partir da matriz $\mathbf{\Omega}_t^0$ pode-se calcular as matrizes correspondentes a $p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$ e $p\left(\mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$, denominadas $\mathbf{\Omega}_t^1$ e $\mathbf{\Omega}_t^2$, respec-

tivamente:

$$\boldsymbol{\Omega}_t^1 = \boldsymbol{\Omega}_t^0 - \boldsymbol{\Omega}_t^0 \, \mathbf{M}_{m^{+-}}^T \left(\mathbf{M}_{m^{+-}} \, \boldsymbol{\Omega}_t^0 \, \mathbf{M}_{m^{+-}}^T \right)^{-1} \mathbf{M}_{m^{+-}} \, \boldsymbol{\Omega}_t^0 \tag{3.49}$$

$$\boldsymbol{\Omega}_t^2 = \boldsymbol{\Omega}_t^0 - \boldsymbol{\Omega}_t^0 \, \mathbf{M}_{x_\mathrm{r},m^{+-}}^T \left(\mathbf{M}_{x_\mathrm{r},m^{+-}} \, \boldsymbol{\Omega}_t^0 \, \mathbf{M}_{x_\mathrm{r},m^{+-}}^T \right)^{-1} \mathbf{M}_{x_\mathrm{r},m^{+-}} \, \boldsymbol{\Omega}_t^0 \tag{3.50}$$

Por fim, a matriz $\boldsymbol{\Omega}_t^3$ correspondente a $p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$ é calculada por:

$$\boldsymbol{\Omega}_t^3 = \boldsymbol{\Omega}_t - \boldsymbol{\Omega}_t \, \mathbf{M}_{x_\mathrm{r}}^T \left(\mathbf{M}_{x_\mathrm{r}} \, \boldsymbol{\Omega}_t \, \mathbf{M}_{x_\mathrm{r}}^T \right)^{-1} \mathbf{M}_{x_\mathrm{r}} \, \boldsymbol{\Omega}_t \tag{3.51}$$

De posse das matrizes $\boldsymbol{\Omega}_t^1$, $\boldsymbol{\Omega}_t^2$ e $\boldsymbol{\Omega}_t^3$, a matriz de informação esparsificada, $\tilde{\boldsymbol{\Omega}}_t$, correspondente a distribuição aproximada na Equação 3.47 é obtida:

$$\tilde{\boldsymbol{\Omega}}_t = \boldsymbol{\Omega}_t^1 - \boldsymbol{\Omega}_t^2 + \boldsymbol{\Omega}_t^3 \tag{3.52}$$

Após a esparsificação da matriz de informação, para manter a igualdade da Equação 3.22 repetida abaixo,

$$\boldsymbol{\xi} = \mathbf{P}^{-1} \boldsymbol{\mu} \tag{3.22 repetida}$$

é preciso recalcular o vetor de informação:

$$\begin{aligned} \tilde{\boldsymbol{\xi}}_t &= \tilde{\boldsymbol{\Omega}}_t \boldsymbol{\mu}_t \\ &= \left(\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_t + \tilde{\boldsymbol{\Omega}}_t \right) \boldsymbol{\mu}_t \\ &= \boldsymbol{\Omega}_t \boldsymbol{\mu}_t + \left(\tilde{\boldsymbol{\Omega}}_t - \boldsymbol{\Omega}_t \right) \boldsymbol{\mu}_t \\ &= \boldsymbol{\xi}_t + \left(\tilde{\boldsymbol{\Omega}}_t - \boldsymbol{\Omega}_t \right) \boldsymbol{\mu}_t \end{aligned} \tag{3.53}$$

As Equações 3.48 até 3.53 estão sumarizadas no Algoritmo 3.4, que trata da esparsificação da matriz de informação do SEIF utilizada como premissa para desenvolvimento das etapas anteriores.

3.5 Associação de *landmarks*

Para calcular a inovação, $\mathbf{z}_t = \mathbf{y}^j - \boldsymbol{h}(\boldsymbol{\mu}_t)$, da *j-ésima* leitura, é necessário estabelecer a qual *landmark* do vetor de estados a leitura $\mathbf{y}^j$ corresponde. Porém, até o momento, tanto na atualização do EKF-SLAM quanto do SEIF-SLAM, essa correspondência foi dada como premissa. Mas um sistema SLAM precisa lidar com o problema da associação de medidas e *landmarks*, pois essas associações são desconhecidas e precisam ser feitas sempre que as *landmarks* não podem ser identificadas individualmente utilizando apenas a medida do sensor (LIU; THRUN, 2003, p. 4). Dessa forma é preciso diferenciar quando

Algoritmo 3.4 Etapa de esparsificação do SEIF-SLAM

function SEIF-SLAM-ESPARSIFICAÇÃO($\boldsymbol{\mu}_t, \boldsymbol{\xi}_t, \boldsymbol{\Omega}_t, \mathbf{m}^{++}, \mathbf{m}^{+-}$)

/* Defina as matrizes de projeção $\mathbf{M}_{x_\mathrm{t},m^{++},m^{+-}}$, $\mathbf{M}_{m^{+-}}, \mathbf{M}_{x_\mathrm{r},m^{+-}}$ e $\mathbf{M}_{x_\mathrm{r}}$ */

$$\boldsymbol{\Omega}_t^0 \leftarrow \mathbf{M}^T_{x_\mathrm{r},m^{++},m^{+-}} \mathbf{M}_{x_\mathrm{r},m^{++},m^{+-}} \, \boldsymbol{\Omega}_t \, \mathbf{M}^T_{x_\mathrm{r},m^{++},m^{+-}} \mathbf{M}_{x_\mathrm{r},m^{++},m^{+-}}$$

$$\tilde{\boldsymbol{\Omega}}_t \leftarrow \boldsymbol{\Omega}_t^0 - \boldsymbol{\Omega}_t^0 \, \mathbf{M}^T_{m^{+-}} \left(\mathbf{M}_{m^{+-}} \, \boldsymbol{\Omega}_t^0 \, \mathbf{M}^T_{m^{+-}} \right)^{-1} \mathbf{M}_{m^{+-}} \, \boldsymbol{\Omega}_t^0$$

$$- \left(\boldsymbol{\Omega}_t^0 - \boldsymbol{\Omega}_t^0 \, \mathbf{M}^T_{x_\mathrm{r},m^{+-}} \left(\mathbf{M}_{x_\mathrm{r},m^{+-}} \, \boldsymbol{\Omega}_t^0 \, \mathbf{M}^T_{x_\mathrm{r},m^{+-}} \right)^{-1} \mathbf{M}_{x_\mathrm{r},m^{+-}} \, \boldsymbol{\Omega}_t^0 \right)$$

$$+ \boldsymbol{\Omega}_t - \boldsymbol{\Omega}_t \, \mathbf{M}^T_{x_\mathrm{r}} \left(\mathbf{M}_{x_\mathrm{r}} \, \boldsymbol{\Omega}_t \, \mathbf{M}^T_{x_\mathrm{r}} \right)^{-1} \mathbf{M}_{x_\mathrm{r}} \, \boldsymbol{\Omega}_t$$

$$\tilde{\boldsymbol{\xi}}_t \leftarrow \boldsymbol{\xi}_t + \left(\tilde{\boldsymbol{\Omega}}_t - \boldsymbol{\Omega}_t \right) \boldsymbol{\mu}_t$$

return $\tilde{\boldsymbol{\xi}}_t, \tilde{\boldsymbol{\Omega}}_t$

end function

uma *landmark* é reobservada e quando é observada pela primeira vez e, além disso, associar a medida à *landmark* correta pois caso contrário o filtro irá divergir.

Portanto, nessa Seção é apresentado um método de associação entre as medidas lidas e as *landmarks* presentes no vetor de estados do sistema SLAM. Esa etapa é necessária em qualquer sistema SLAM, independente do "*back-end*" utilizado (EKF, EIF, Filtro de Partículas, etc); no entanto, a discussão aqui apresentada focará nas particularidades do SEIF.

Para calcular as correspondências foi utilizado o método guloso *Individual Compatibility Nearest Neighbor* (ICNN), Algoritmo 3.5; em linhas gerais, ele atribui a *landmark* com a menor distância de Mahalanobis (MCLACHLAN, 1999) à medida lida. E caso essa distância seja menor que um limiar $\chi^2_{2,\alpha}$, essa medida é associada à *landmark* candidata. Caso contrário, é considerado que essa medida pertence a uma *landmark* não observada antes, e então ela é inserida no vetor e matriz de informação como mostrado na Seção 3.8. Porém, essa nova *landmark* não é inserida imediatamente no filtro pois pode se tratar de um erro de leitura; na Seção 3.6 será mostrada a estratégia utilizada para mitigar esse problema e a partir de qual momento a *landmark* é inserida, de acordo com essa estratégia.

Como pode ser observado, o Algoritmo 3.5 utiliza as covariâncias das *landmarks*, denominadas por $\mathbf{P}_{0:\mathrm{M}}$, como entrada. Porém, no SEIF não é mantida uma matriz de covariância, mas sim a matriz de informação. Portanto, para utilizar a técnica acima é necessário "recuperar" as covariâncias das *landmarks* a partir da matriz de informação.

Algoritmo 3.5 Associação da medida $\mathbf{y}_t$ com *landmark* do vetor de estados

function TESTE-CORRESPONDÊNCIA-ICNN($\boldsymbol{\mu}_t, \mathbf{P}_{0:\mathrm{M}}, \mathbf{y}_t$)
  $m \leftarrow 0$
  $d \leftarrow \infty$
  $m_c \leftarrow \{\}$
  while $m < \mathrm{M}$ **do**
    $\Delta_m \leftarrow \mathbf{y}_t - \boldsymbol{h}^m(\boldsymbol{\mu}_t)$
    if $\Delta_m^T \Delta_m < d$ **then**
      $m_c \leftarrow \{m\}$
      $\pi_m \leftarrow \Delta_m^T \mathbf{P}_m^{-1} \Delta_m$
      $d \leftarrow \Delta_m^T \Delta_m$
    end if
    $m \leftarrow m + 1$
  end while
  if $\pi_m < \chi^2_{2,\alpha}$ **then**
    return m_c
  end if
  return $\{\}$
end function

3.5.1 Estimação da covariância de *landmarks* a partir da matriz de informação

Para utilizar a técnica de associação de dados, apresentada na Seção anterior, com o SEIF, uma técnica para calcular a covariância das *landmarks* se faz necessária. Num primeiro momento, basta inverter a matriz de informação e extrair os blocos da diagonal principal. Porém, dessa forma todo o esforço para manter as etapas do SEIF com complexidade constante seriam minados, pois a inversão de matrizes tem complexidade $\mathcal{O}\left(n^{2.376}\right)$ (COPPERSMITH; WINOGRAD, 1990).

A fim de tratar esse problema, (LIU; THRUN, 2003, p. 5) propõe calcular a covariância das *landmarks* candidatas à associação de maneira aproximada levando em consideração apenas um subconjunto de *landmarks*, denominado Cobertor de Markov (*Markov Blanket*) e simbolizado por $\mathbf{m}^+_{c_t}$, e condicionando todas as demais. Esse conjunto contém as *landmarks* ativas $\mathbf{m}^+$ e as vizinhas, *landmarks* conectadas na matriz de informação, da *landmark* candidata à associação. Um exemplo do Cobertor de Markov é ilustado na Figura 3.14.

Utilizando o resultado do condicionamento da distribuição gaussiana na forma de informação apresentado no Anexo B.2 e já utilizado na esparsificação, calcula-se a matriz

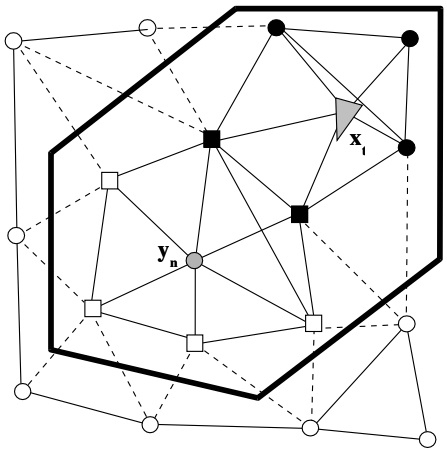


FIGURA 3.14 – Rede de conexões entre *landmarks* e o robô. A maioria das *landmarks* estão representadas por círculos, as *landmarks* vizinhas da *landmark* candidata a associação estão representadas por quadrados. As *landmarks* ativas estão coloridas de preto, a *landmark* candidata está colorida de cinza. O polígono de bordas grossas delimita o conjunto do Cobertor de Markov, $\mathbf{m}^+_{c_t}$. Neste caso há interseção entre o conjunto de *landmarks* vizinhas e ativas. Retirado de (THRUN *et al.*, 2005, p. 410).

de covariância aproximada $\tilde{\mathbf{P}}_{\mathrm{m^j}}$ da seguinte forma:

$$\tilde{\mathbf{P}}_{\mathrm{m^j}} = \mathbf{M}_{\mathrm{m^j}} \left(\mathbf{M}_{\mathbf{x}^{\mathrm{r}}_t, \mathrm{m^j}, \mathbf{m}^+_{c_t}} \, \boldsymbol{\Omega}_t \, \mathbf{M}^T_{\mathbf{x}^{\mathrm{r}}_t, \mathrm{m^j}, \mathbf{m}^+_{c_t}} \right)^{-1} \mathbf{M}^T_{\mathrm{m^j}} \tag{3.54}$$

Entretanto, ao contrário do que é mostrado na Figura 3.14, pode ser que não exista interseção entre o conjunto de *landmarks* vizinhas a candidata e o conjunto de *landmarks* ativas $\mathbf{m}^+$ dentro do Cobertor de Markov. Nesses casos, o cobertor deve ser aumentado de forma que englobe um "caminho" entre o robô e a *landmark* candidata. Para isso, a matriz de informação pode ser interpretada como um grafo e inserir as *landmarks* que integram o menor caminho entre o robô e a *landmark* candidata.

3.6 Estratégia para mitigar efeito de falsas detecções de *landmarks*

Um problema comum que ocorre na prática em sistemas SLAM é a detecção errônea e/ou falsa de *landmarks* gerando medidas erradas. Isso pode acontecer por erros no processamento dos dados do sensor, movimentos bruscos do robô, e em sistemas multiagentes um robô pode ser confundido com uma *landmark* dependendo dos tipos de *landmark* e do sensor utilizado. Esse último é particularmente comum no cenário desse trabalho, onde as *landmarks* e os robôs possuem formato cilíndrico e o sensor do tipo LiDAR fornece apenas informação espacial, sem cores ou texturas.

Falsas detecções podem levar a falsas associações de novas medidas, acarretando em erro de localização e consequente divergência na observação de novas *landmarks*. Para

atenuar esse problema e diminuir as chances de uma falsa *landmark* causar divergência dos filtros, é comum separar as *landmarks* em uma lista provisória e numa lista permanente.

A lista de *landmarks* permanentes é composta pelas *landmarks* já consolidadas pelo filtro, e influenciam na estimação da pose do robô. Já as *landmarks* provisórias são *landmarks* ainda não consolidadas; as leituras associadas a elas são utilizadas apenas para melhorar a estimativa de suas posições e não influenciam na estimação da pose do robô.

Para construir essas listas Saputra (2019) empregou o uso de um esquema de pontuação para as *landmarks* observadas. Esse esquema possui duas operações: atualização e degradação. A operação de atualização consiste em aumentar um ponto a pontuação $\mathcal{Q}$ da *landmark* sempre que ela é reobservada. Enquanto a degradação diminui dois pontos sempre que a *landmark* não é reobservada.

$$\mathcal{Q}_{t+1}\left(\mathbf{m}^j\right) = \begin{cases} \mathbf{m}^j \text{ é reobservada} & \implies \mathcal{Q}_t\left(\mathbf{m}^j\right) + 1 \text{ (atualização)} \\ \mathbf{m}^j \text{ não é reobservada} & \implies \mathcal{Q}_t\left(\mathbf{m}^j\right) - 2 \text{ (degradação)} \end{cases} \tag{3.55}$$

Toda *landmark* começa na lista provisória e conforme sua pontuação $\mathcal{Q}$ evolui no tempo ela pode ser promovida para a lista de *landmarks* permanentes, passando a ser utilizada na correção da pose do robô. Ou até mesmo removida no filtro caso deixe de ser observada sistematicamente. Uma *landmark* se torna permanente quando sua pontuação chega a 10, e é excluída quando pontua cinco pontos negativamente:

$$\begin{cases} \mathcal{Q}_t\left(\mathbf{m}^j\right) \geq 10 & \text{Promoção} \\ \mathcal{Q}_t\left(\mathbf{m}^j\right) \leq -5 & \text{Remoção} \end{cases} \tag{3.56}$$

Com essa abordagem foi possível erradicar a incorporação de falsas *landmarks* geradas tanto no cenário com um robô quanto no cenário com dois robôs. Além disso, como será visto na Seção 5.2, ao trocarem seus mapas os robôs enviam para o outro par apenas o vetor e matriz de informação correspondentes às *landmarks* consolidadas pelo filtro (i.e. permanentes).

3.7 Conclusão do capítulo

Neste Capítulo foram exploradas as particularidades da estimação de estado de sistemas SLAM em relação à estimação de estado de sistemas convencionais. Enquanto um sistema SLAM possui vetor de estados com tamanho dinâmico, o tamanho do vetor de medidas é variante no tempo, uma porção do estado é variante no tempo (a pose do robô) enquanto a outra é invariante (posição das *landmarks*), e a associação entre medidas e estados é desconhecida a *priori*. Um sistema convencional possui número de medidas fixo

e os componentes do estado têm mesmo tipo de variabilidade no tempo.

Também foram mostradas as adaptações necessárias para utilizar o Filtro de Kalman Estendido na estimação de estado de sistemas SLAM, caracterizando a técnica EKF-SLAM. Além disso, foi apresentado a técnica SEIF-SLAM e como ela é baseada na formulação dual do EKF, e como ela utiliza o conceito de *landmarks* ativas para manter a matriz de informação esparsa levando ao uso linear de memória e tempos de atualização e predição constantes. Mais adiante, na Seção 5.2 também será mostrado como a troca de informações entre sistemas SEIF-SLAM é natural devido a utilização da parametrização canônica do filtro.

Além disso foi abordada a técnica de associação de dados *Individual Compatibility Nearest Neighbor*, efetiva no ambiente deste trabalho. E por fim, foi tratado o problema prático da detecção de falsas *landmarks* e como atenuá-lo.

4 SLAM *Front-end*

No capítulo anterior foram abordadas as soluções EKF e SEIF, com maior foco neste último, do problema de estimação que deve ser resolvido por um sistema SLAM. Mas apenas resolver a estimação não é suficiente, existem outras capacidades "comportamentais" que um robô deve possuir para desempenhar a tarefa SLAM, como processamento de leituras dos sensores, construir representações úteis do ambiente e, a partir dessas representações, traçar trajetórias a fim de explorá-lo e aumentar seu conhecimento a respeito do mesmo. Esses aspectos são compreendidos pelo *front-end* como mostrado na Figura 1.1 e, nesse capítulo, serão abordadas as soluções para alguns deles.

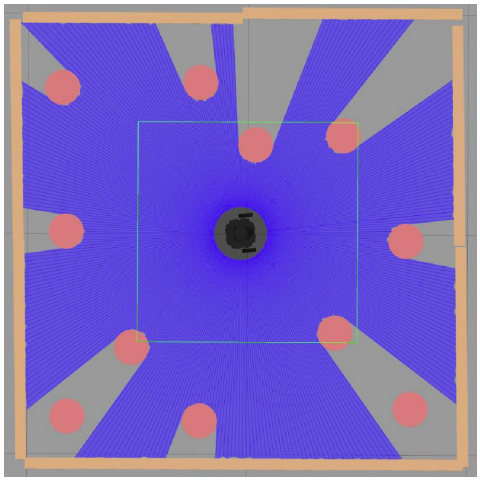
4.1 Dados do sensor laser

Essa Seção explica o formato dos dados brutos do sensor LiDAR e como eles são processados e transformados nos dados utilizados pelo modelo de medida descrito na Seção 2.2 (Medidas e o modelo de medida Distância-Azimute).

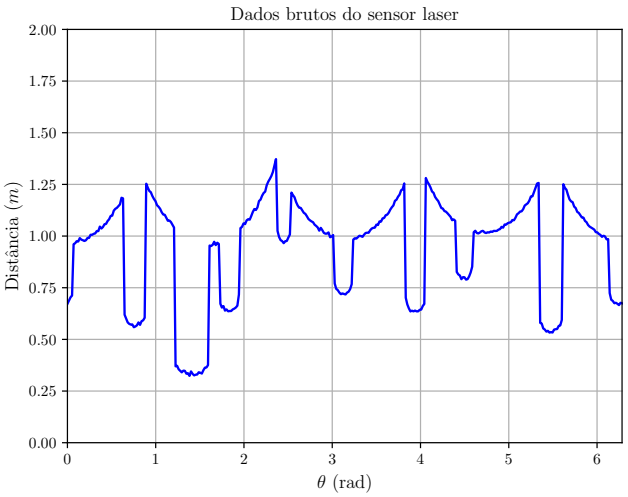
4.1.1 Dados brutos

Os dados brutos do sensor LiDAR utilizado, consistem numa sequência de distâncias $\{r^0, r^1, \ldots, r^N\}$. Esses valores são gerados pela reflexão de feixes de laser, emitidos pelo sensor, nas superfícies presentes no ambiente. Os feixes são disparados de maneira sequencial no sentido anti-horário, a partir do eixo x do sistema de coordenadas do sensor. Além disso, o sensor também fornece as posições angulares θ_0 do primeiro e θ_N do último feixe, e o incremento na posição angular $\Delta\theta$ entre o feixe k e o feixe $k+1$.

A Figura 4.1b apresenta os dados brutos obtidos em uma leitura feita no ambiente mostrado na Figura 6.1.



(a) Representação dos feixes laser (em azul) emitidos pelo sensor LiDAR com alcance máximo de 2 metros. Os círculos em rosa representam as landmarks.



(b) Interpolação linear das distâncias lidas pelo sensor.

FIGURA 4.1 – Visualização dos feixes laser emitidos pelo sensor LiDAR e a respectiva leitura gerada.

4.1.2 Processamento de dados

Para transformar os dados brutos da sequência de distâncias, apresentada na Figura 4.1b, nas medidas consumidas pelo modelo de medida descrito na Seção *Medidas e o modelo de medida Distância-Azimute*, é proposta uma sequência de passos para processar as medidas, conforme a Figura 4.2 que inicia com a detecção de picos no sinal de derivada apresentada por Brenner (2012) e se finda com o algoritmo de estimação de círculos a partir de pontos em coordenadas cartesianas (AL-SHARADQAH; CHERNOV, 2009, p. 903).

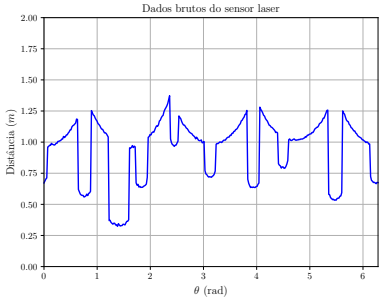
Para começar, é necessário extrair os pontos que correspondem às reflexões nas superfícies dos cilindros presentes no ambiente. Para isso, (BRENNER, 2012) observa que há uma variação brusca nas distâncias lidas pelo sensor quando os feixes são refletidos pelas superfícies dos cilindros. Ao analisar a derivada do sinal, representada na Figura 4.3, podemos notar que o intervalo de medidas correspondente às reflexões dos cilindros se encontram entre uma variação negativa seguida rapidamente de uma variação positiva na curva da derivada.

O próximo passo consiste em transformar o sinal para a representação em coordenadas cartesianas e segmentar em conjuntos os pontos nos intervalos entre picos e vales do sinal de derivada, como é mostrado na Figura 4.4. Então, cada um desses conjuntos é fornecido como entrada do Algoritmo C.1 (listado no Anexo C.1); o resultado são círculos estimados que melhor se ajustam a cada um desses conjuntos de pontos (Figura 4.5), juntamente com o erro médio quadrático da estimação (σ) de cada círculo.

São considerados apenas os círculos cujo erro médio quadrático $\sigma < \epsilon$ é inferior ao limiar $\epsilon = 10^{-3}$ metros e cujo raio seja superior a 6.5 cm. Essa última condição é especial-

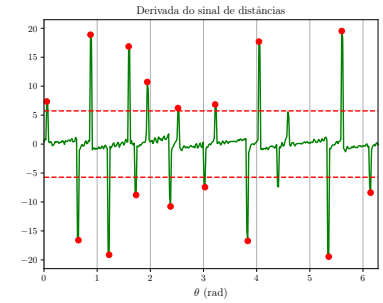
1

ENTRADA
Sequência de distâncias
lidas pelo sensor.



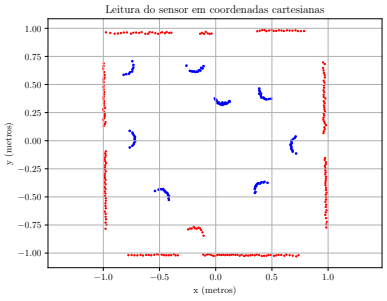
2

Cálculo da derivada do sinal
e extração de pontos de mínimo
e máximo fora dos valores limiares.



3

Segmentação dos pontos entre
os pontos de mínimo e máximo do
sinal de derivada, transformação
para coordenadas cartesianas.



4

SAÍDA
Estimação dos círculos que
descrevem os conjuntos de pontos,
com $r > 6.5cm$ e $\sigma < \epsilon$

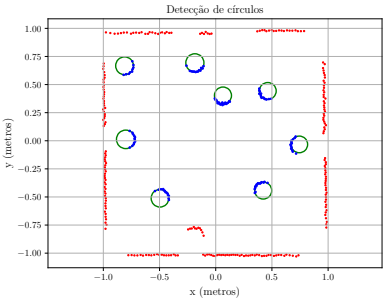


FIGURA 4.2 – Sequência de passos para processar os dados brutos do sensor LiDAR e transformá-los em dados úteis para o modelo de medida utilizado.

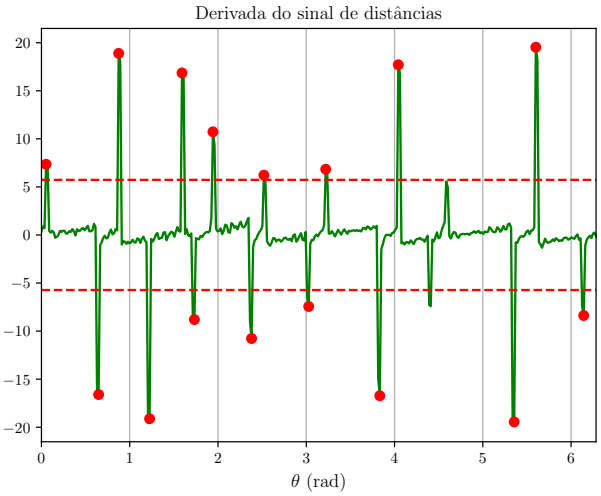


FIGURA 4.3 – Derivada central da sequência de distâncias representadas na Figura 4.1b. As linhas pontilhadas em vermelho representam os limiares a partir dos quais os picos são interpretados como início (negativo) ou fim (positivo) da superfície de um cilindro. Os picos destacados em vermelho ultrapassam as valores limiares.

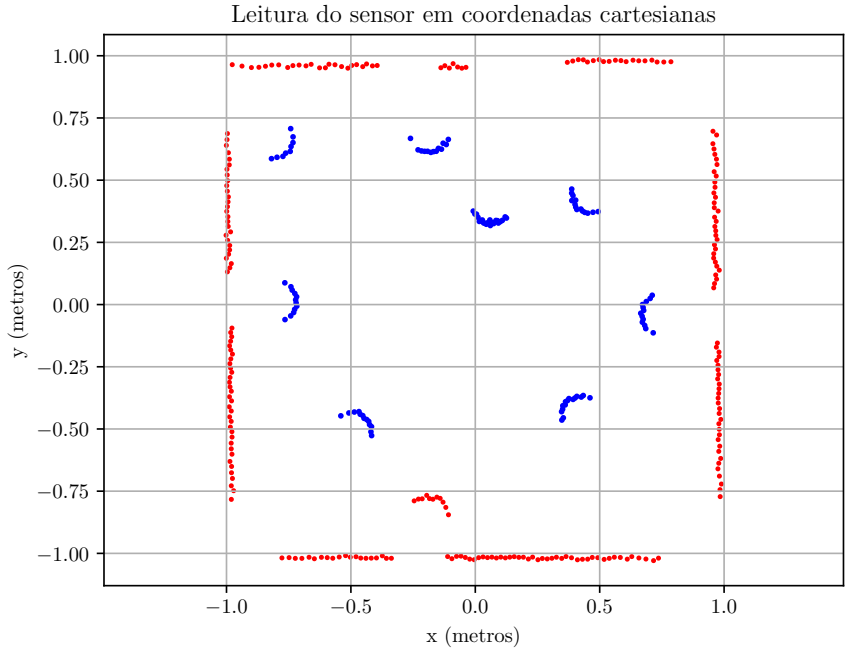


FIGURA 4.4 – Representação da leitura do sensor LiDAR em coordenadas cartesianas. Os pontos destacados em azul correspondem a reflexões das superfícies das *landmarks*.

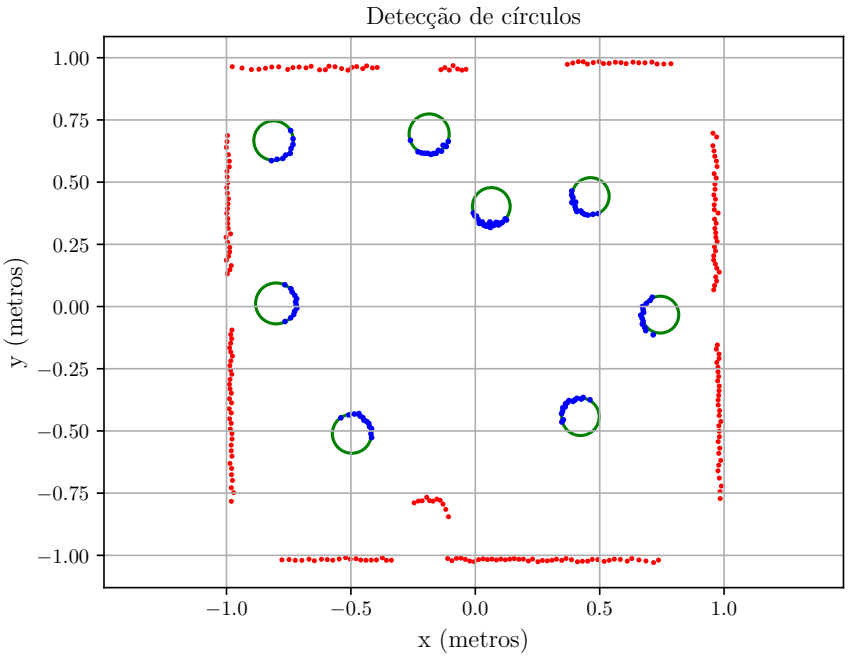


FIGURA 4.5 – Em azul, os conjuntos de pontos selecionados como pertencentes às superfícies das *landmarks*. Em verde, os círculos estimados para cada conjunto de pontos decorrentes das medidas obtidas do sensor.

mente útil no cenário multiagente, pois evita que um robô confunda o sensor LiDAR (que fica montado no topo do robô, conforme visto na Figura 6.2) do outro com uma *landmark*.

4.2 Mapa em grade

Até o momento o ambiente foi representado apenas como um conjunto de *landmarks*; embora essa representação seja muito útil para a estimação, abordada no Capitulo 3, ela não é adequada para representar outros aspectos do ambiente além das próprias *landmarks*. O robô não consegue representar a presença de outros objetos, paredes ou até mesmo outros agentes.

Uma forma de representação que trata desses problemas é a grade de ocupação. Grades de ocupação discretizam o espaço em um conjunto de células, onde cada célula armazena a probabilidade de estar ocupada ou não (ELFES, 1989). A Figura 4.6 ilustra a grade de ocupação construída a partir das observações a partir do centro da cena representada na Figura 4.1a.

Para utilizar grade de ocupação, são adotadas duas hipóteses simplificadoras: a primeira é que as células são independentes e o estado de ocupação de uma célula não diz nada sobre o estado de outra; a segunda é que o ambiente é estático. Embora essas duas premissas sejam falsas, grades de ocupação funcionam muito bem na prática, inclusive

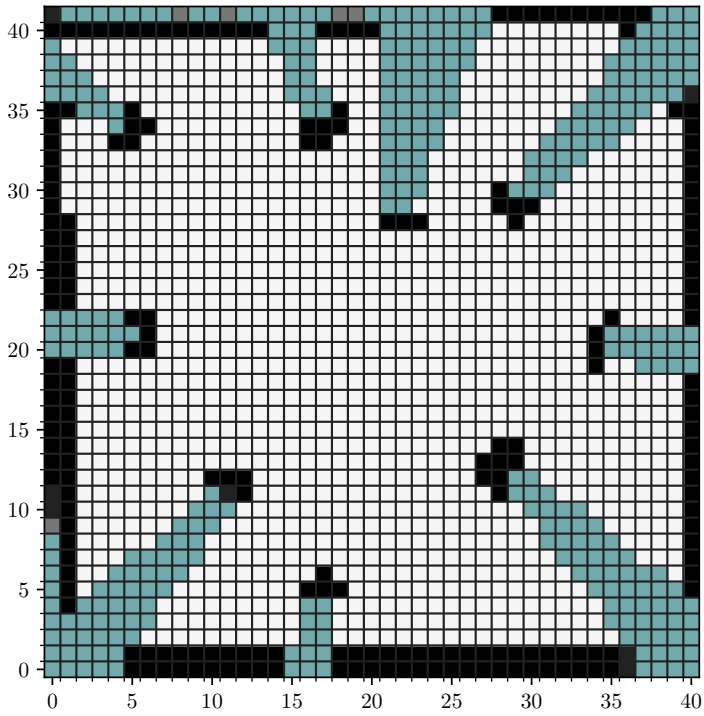


FIGURA 4.6 – Grade de ocupação produzida utilizando as leituras do sensor LiDAR. As células em azul representam porções do ambiente com estado desconhecido a partir da observação deste único ponto de vista. A cor das demais células refletem a probabilidade de estarem ocupadas, quanto mais escura maior a confiança do robô de que existe algum objeto no espaço delimitado pela célula. Cada célula é um quadrado com lado 5cm.

em ambientes dinâmicos. A primeira hipótese é falsa porque se uma célula representa a ocupação de parte do corpo de um objeto, é muito provável que suas vizinhas também estejam ocupadas por esse objeto. Já a segunda, porque há outros agentes se movimentando no ambiente.

Utilizando essas hipóteses, cada célula pode ser modelada como uma variável aleatória binária que pode assumir os estados livre ou ocupado, e cujo estado é constante no tempo (i.e. estático). Isso sugere a utilização do filtro de Bayes binário (THRUN *et al.*, 2005, p. 94) para estimar o estado das células da grade.

4.2.1 Construção da grade de ocupação

Para construir a grade de ocupação primeiro é necessário definir sua resolução, isto é, o tamanho de sua célula. Quanto menor a célula, mais memória é utilizada para representar o ambiente e melhor é a representação. A quantidade de células pode ser fixa, caso a área da grade seja definida inicialmente, ou dinâmica.

Uma vez que o ambiente está segmentado em células, utiliza-se o modelo de medida inverso $\boldsymbol{f}(\cdot,\cdot)$(Seção 2.2.2) para determinar a qual célula pertence a extremidade do feixe da leitura do LiDAR. Caso o comprimento do feixe seja menor que o alcance máximo

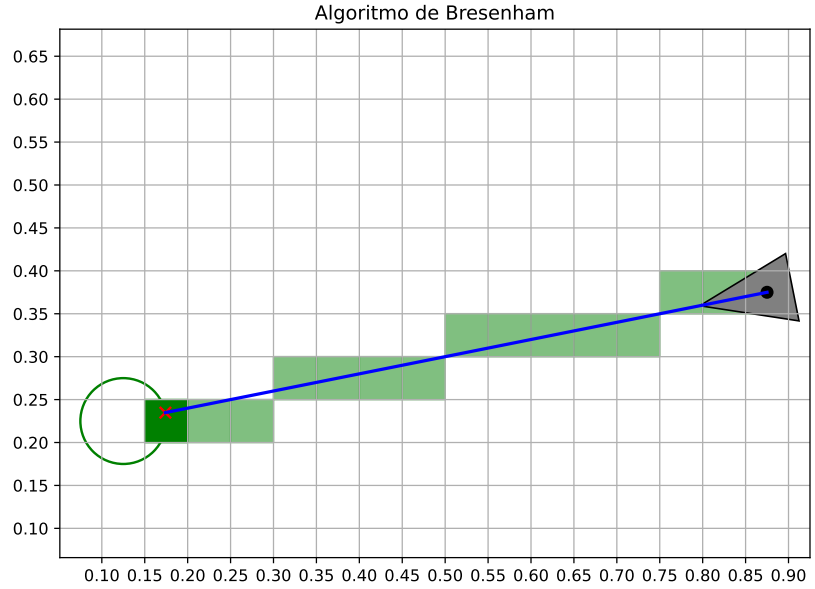


FIGURA 4.7 – Uso do algoritmo de Bresenham para determinar as células interceptadas pelo feixe (em azul) do sensor LiDAR. As células em verde claro, representam o espaço atravessado pelo feixe. A célula em verde escuro representam o espaço em que o feixe interceptou a superfície de um objeto. O robô é representado pelo triângulo cinza.

do sensor, aumenta a probabilidade de ocupação da célula que contém a extremidade do feixe, enquanto diminui as das células ao longo do feixe.

Para determinar quais células da grade são interceptadas pelos feixes do sensor LiDAR, é utilizado o algoritmo de desenho de linha de Bresenham (BRESENHAM, 1965). Inicialmente, esse algoritmo foi criado para desenhar linhas em *plotters* digitais. Aqui, ele é usado para "desenhar" o feixe do LiDAR, com origem no sensor e extremidade calculada pelo modelo de medida inverso, na grade de ocupação. As células no algoritmo de desenho são as células interceptadas pelo feixe, como é ilustrado na Figura 4.7.

Por fim, é preciso definir uma distribuição de probabilidade de ocupação das células para o modelo de medida inverso. Por conta da boa qualidade de sensores do tipo LiDAR, utilizou-se um modelo simplificado, ilustrado na Figura 4.8.

4.2.2 Representação em *log odds*

Implementações do filtro de Bayes binário são comumente realizadas na forma do logaritmo da razão de chances (*log odds ratio*) (THRUN *et al.*, 2005, p. 94). A razão de chances é a razão entre a probabilidade de ocorrência de um evento (a célula estar ocupada), dividida pela probabilidade da não ocorrência desse evento (a célula estar livre). Se $p\left(c_{ij} \mid \mathbf{y}_{1:t}\right)$ é a probabilidade da célula c_{ij} estar ocupada dada as medidas dos sensor

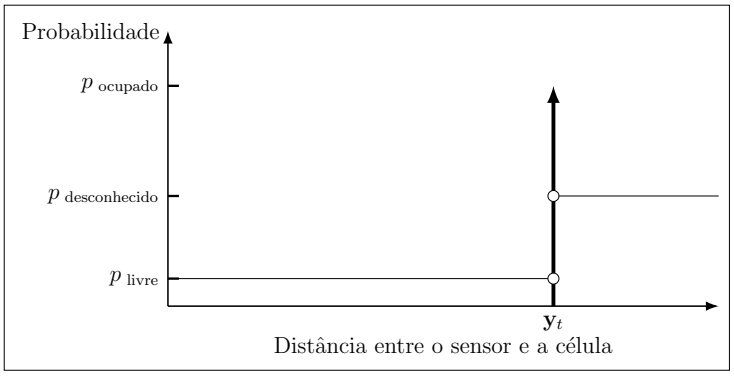


FIGURA 4.8 – Distribuição de probabilidade simplificada do modelo de medida inverso.

LiDAR até o instante t, então a sua razão de chances é:

$$\frac{p\left(c_{ij} \mid \mathbf{y}_{1:t}\right)}{p\left(\neg c_{ij} \mid \mathbf{y}_{1:t}\right)} = \frac{p\left(c_{ij} \mid \mathbf{y}_{1:t}\right)}{1 - p\left(c_{ij} \mid \mathbf{y}_{1:t}\right)} \tag{4.1}$$

A atualização recursiva da razão de chances da ocupação de uma célula é dada pelo produto de razões mostrado na Equação 4.2 abaixo (THRUN *et al.*, 2005, p. 96). A primeira razão corresponde à razão de chances de ocupação do modelo de medida inverso, cuja distribuição de probabilidade foi mostrada na Figura 4.8, o termo do meio é o termo recursivo, e o da direita representa algum conhecimento incial que se possa ter sobre o ambiente.

$$\frac{p\left(c_{ij} \mid \mathbf{y}_{1:t}\right)}{p\left(\neg c_{ij} \mid \mathbf{y}_{1:t}\right)} = \frac{p\left(c_{ij} \mid \mathbf{y}_{t}\right)}{1 - p\left(c_{ij} \mid \mathbf{y}_{t}\right)} \frac{p\left(c_{ij} \mid \mathbf{y}_{1:t-1}\right)}{1 - p\left(c_{ij} \mid \mathbf{y}_{1:t-1}\right)} \frac{1 - p\left(c_{ij}\right)}{p\left(c_{ij}\right)} \tag{4.2}$$

Definindo o logaritmo da razão de chances, também conhecido por logit, por:

$$l_t = \log\left(\frac{p\left(c_{ij} \mid \mathbf{y}_{1:t}\right)}{p\left(\neg c_{ij} \mid \mathbf{y}_{1:t}\right)}\right) \tag{4.3}$$

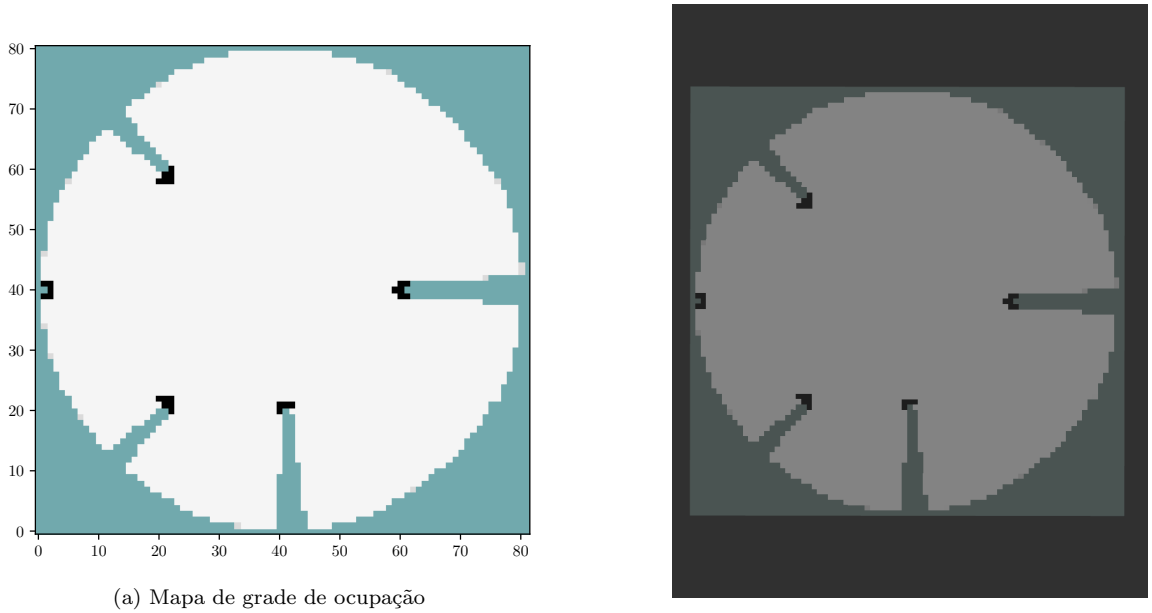
Aplicando a definição acima na Equação 4.2, ela pode ser reescrita na forma:

$$l_t = \log\left(\frac{\text{Modelo-de-medida-inverso}}{1 - \text{Modelo-de-medida-inverso}}\right) + l_{t-1} - l_0 \tag{4.4}$$

e a probabilidade de ocupação de uma célula pode ser recuperada utilizando a relação na Equação 4.5.

$$p = 1 - \frac{1}{1 + \exp(l_t)} \tag{4.5}$$

Como fica claro na Equação 4.4, a incorporação de novas leituras se resume a uma simples adição. Esse aspecto da representação em *log odds* será muito útil mais adiante quando os agentes trocarem suas grades de ocupação, pois, após o alinhamento entre as grades, a fusão entre elas também será uma simples soma.



(a) Mapa de grade de ocupação

(b) Visualização da grade de ocupação no software RVIZ

FIGURA 4.9 – Diferentes representações de uma mesma grade de ocupação.

4.3 Exploração Autônoma

Na Seção anterior foi mostrado como as grades de ocupação são utilizadas para representar ambiente e como essa representação é construída e mantida. Nessa seção será mostrado como a grade pode ser empregada para determinar regiões alvo para exploração dos agentes e como elas são úteis para evitar obstáculos.

A técnica de exploração utilizada nesse trabalho é uma simplificação da exploração orientada ao ganho de informação, denominada ganho binário. Ela consiste em guiar o robô para a região desconhecida mais próxima na grade de ocupação, e também é conhecida por *exploração baseada em fronteira*. Apesar de ser uma técnica simples, ela tende a funcionar bem na prática, pois sempre guia o robô em direção a terreno desconhecido (THRUN *et al.*, 2005, p. 584). Se o robô estivesse no centro da grade representada na Figura 4.9a, ele selecionaria para explorar a célula com ocupação desconhecida (azul) mais próxima e com uma distância de segurança mínima de uma célula ocupada.

Além de selecionar o novo ponto para onde deve seguir, o robô também precisa estabelecer um caminho livre de obstáculos que o leve de sua posição atual para a posição de destino. Para isso, o mapa de grade é transformado em um mapa de custo e o robô computa o caminho com menor custo até a célula de destino. O custo de uma célula é inversamente proporcional a sua distância para as células ocupadas mais próximas; o resultado pode ser visualizado como um "mapa de calor" na Figura 4.10.

Para obter o caminho com menor custo, o mapa de custo é modelado como um grafo com os custos das arestas iguais aos custos da células e então o menor caminho é obtido

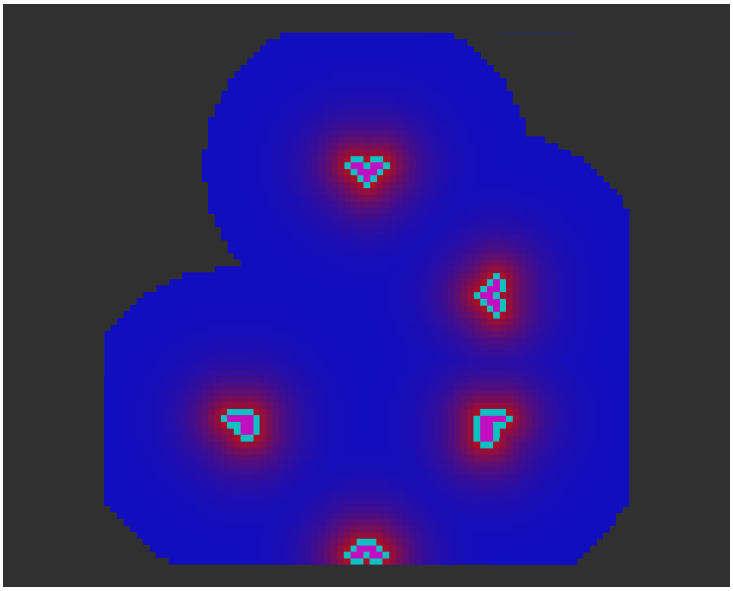


FIGURA 4.10 – Mapa de custo referente à grade de ocupação da Figura 4.9. As regiões quentes (vermelho) representam células de maior custo, enquanto as células frias (azul escuro) representam células de baixo custo.

utilizando o algoritmo de Dijkstra ou, sua variante com heurística, o algoritmo A*.

5 SLAM multiagente descentralizado

Até o momento foram tratados todos os aspectos importantes para que um robô resolva o problema SLAM ativamente: reconhecimento de *landmarks*, diferentes formas de representar o ambiente, exploração, estimação e um mecanismo para evitar que *landmarks* espúrias sejam incorporadas ao filtro causando divergência e falha. Porém, resta explorar um aspecto fundamental para sistema multiagentes: a troca de informações e a subsequente inclusão do conteúdo do mapa de um agente no(s) do(s) outro(s), e vice e versa.

A troca de informação par a par entre os agentes é o mecanismo que caracteriza um sistema SLAM multiagente descentralizado e que confere a ele redundância de informação e robustez frente a falhas de agentes. Além disso, também pode propiciar que os agentes dividam a carga de trabalho na exploração, concluindo a tarefa em menos tempo.

Essa troca pode acontecer em dois níveis: filtro e mapa de grade. Quando um robô incorpora as informações do filtro de seu par, ele pode aumentar a informação a respeito das *landmarks* já conhecidas por ele e que também foram observadas pelo seu par, além de poder adquirir informação de *landmarks* que ainda não foram observadas por ele, mas foram observadas pelo seu par. A matriz e vetor de informação de um é incorporado no do outro.

Similarmente, a troca dos mapas de grade de ocupação permite a mesma coisa do ponto de vista do mapa de grade. Durante a comunicação, um agente incorpora as probabilidades de ocupação, ou melhor o logaritmo das razões de chances (*log odds*), do mapa do outro ao seu próprio. Isso é muito simples por conta da natureza aditiva da representação em *log odds*, como mostrado na Equação 4.4.

No entanto, para que um robô incorpore os mapas (grade e *landmarks*) do outro aos seus próprios, é necessário antes que se determine a posição relativa entre eles para transformar os mapas que estão no sistema de referência local do outro para o seu próprio sistema de referência local, alinhando assim os elementos comuns que aparecem em ambos.

5.1 Computação da posição relativa entre agentes

Para determinar a posição relativa entre os robôs, utilizou-se a técnica de registro de nuvens de pontos, que consiste em estimar uma transformação $\mathcal{T}$ que, ao ser aplicada em uma nuvem de pontos *fonte*, melhor a alinhe com uma nuvem de pontos *alvo*. Um exemplo de registro de nuvens de pontos é mostrado na Figura 5.1, onde duas nuvens de um mesmo ambiente são alinhadas resultando em uma nuvem mais completa.

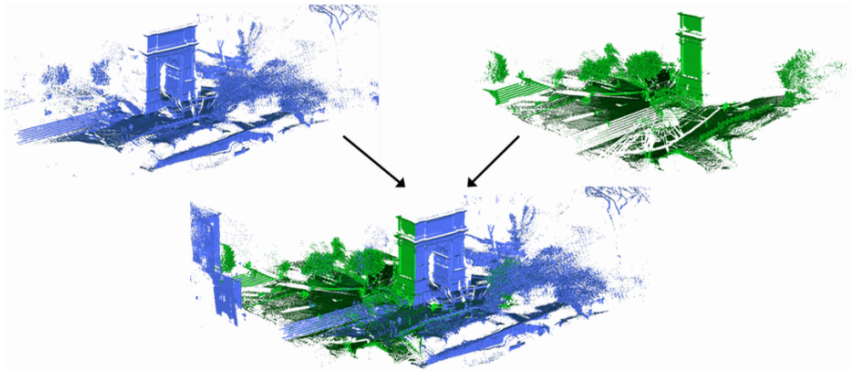


FIGURA 5.1 – Exemplo de registro de par de nuvem de pontos. As nuvens representam porções diferentes de uma mesma cena e possuem elementos em comum.

Com isso, ao utilizar como nuvem de pontos o mapa e *landmarks* de cada robô e restringir que a transformação $\mathcal{T}$ seja uma transformação de corpo rígido $\mathcal{T}_r$, é possível determinar a posição relativa entre eles. A Transformação $\mathcal{T}_r$ é definida por um ângulo de rotação e um vetor de translação entre dois sistemas de referência:

$$\mathcal{T}_r = \begin{bmatrix} \theta & d_x & d_y \end{bmatrix}^T \tag{5.1}$$

A premissa para utilizar essa técnica é que, ao explorar o mesmo ambiente, o mapa criado por cada robô possui elementos (*landmarks*) em comum com os mapas dos demais e, portanto, é possível identificar e estabelecer relação entre as *landmarks* em comum para determinar a transformação $\mathcal{T}_r$ que melhor as alinhem.

Porém, como a única informação que caracteriza uma *landmark* é a sua posição no sistema de referência local, fica difícil determinar se ela também foi observada por outros e, portanto, está em seu mapa. Essa dificuldade se deve à utilização do sensor LiDAR, pois em soluções que usam câmeras RGB ou de RGB-D, as texturas e cores podem ser utilizadas para identificar pontos de interesse.

Para superar essa limitação, utilizou-se um descritor de características para descrever uma *landmark* a partir de sua vizinhança, dessa forma criando uma "impressão digital" que se repetiria no outro mapa, possibilitando identificar quando uma mesma *landmark* foi observada pelo outro par. Então foi desenvolvido um descritor de características inspirado no descritor *Spin Image* (JOHNSON; HEBERT, 1999) para o caso planar do mapa de

landmarks. A Figura 5.2 ilustra o descritor. Ele define uma vizinhança em torno da *landmark* descrita e a divide em setores. Cada setor possui uma posição correspondente no vetor de características; a cada posição do vetor corresponde a quantidade de *landmarks* vizinhas à *landmark* descrita em seu setor correspondente.

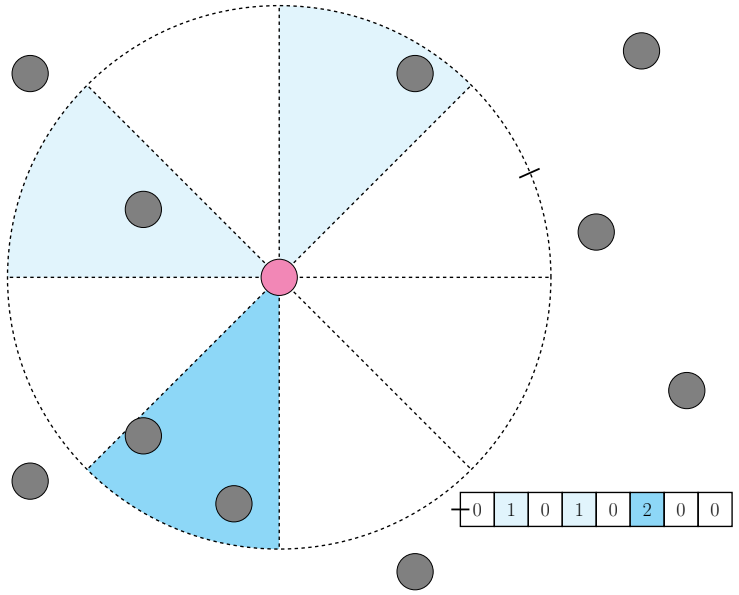


FIGURA 5.2 – Ilustração do descritor de característica utilizado para relacionar *landmarks* entre os mapas dos agente. As *landmarks* estão representadas por círculos em cinza, a *landmark* descrita é representada em magenta. O vetor descritor está representado no canto inferior direito. Cada posição do vetor corresponde a um setor da circunferência tracejada. Os setores que possuem *landmarks* vizinhas estão realçados em tons de azul.

Depois que o vetor de características de cada *landmark* em ambos os mapas é criado, são estabelecidas correspondências entre pares de vetores de descrição (um de cada mapa) utilizando como critério a distância euclidiana. Então, utiliza-se o algoritmo *Random Sample Consensus* (RANSAC) (FISCHLER; BOLLES, 1981) para estimar a melhor transformação que alinha os dois mapas.

Aplicando a transformação obtida no mapa de *landmarks* do outro robô, o primeiro avalia quantas *landmarks* a transformação alinha corretamente, e se há alguma *landmark* com alinhamento indefinido. Um alinhamento é tido como correto quando a distância d entre as *landmarks* é menor que um limiar δ_{near} e como indefinido se a distância está entre os limiares $\delta_{\text{near}} \leq d < \delta_{far}$. A transformação é então aceita como correta se não existirem alinhamentos indefinidos e a quantidade de alinhamentos corretos é maior ou igual a quatro.

5.2 Troca de Mapas

Dada a transformação da posição relativa entre o par de robôs (j, k), obtida pelo registro dos mapas de *landmarks* dos robôs j e k, resta incorporar os mapas de um no

outro. As próximas duas Seções abordam a fusão dos vetores e matrizes de informação de ambos os agentes, e de seus mapas de ocupação. Ambas as fusões exploram o caráter aditivo de suas respectivas representações.

5.2.1 Troca do vetor e matriz de informação

A exposição que segue nessa Seção segue de (THRUN *et al.*, 2005, Seção 12.11).

Para que um robô incorpore os vetores de estado e informação e a matriz de informação um do outro, é necessário antes mudar o sistema de referência de um para o outro. Em posse da transformação de mudança do sistema de referência do robô j para o sistema de referência do robô k, $\mathcal{T}_r^{j\to k}$,

$$\mathcal{T}_r^{j\to k} = \begin{pmatrix} \theta^{j\to k} & d_x^{j\to k} & d_y^{j\to k} \end{pmatrix} \tag{5.2}$$

onde:

- $\theta^{j\to k}$ é o ângulo de rotação necessário para alinhar os eixos do sistema de referência local do robô k com os eixos do sistema de referência do robô j
- $d_x^{j\to k}$ descolamento no eixo x que leva a componente x da origem do sistema de referência do robô k para a origem do sistema de referência do robô j
- $d_y^{j\to k}$ descolamento no eixo y que leva a componente y da origem do sistema de referência do robô k para a origem do sistema de referência do robô j

A posição da i-ésima *landmark* do vetor de estados do robô j é reescrita no sistema de referência do robô k da seguinte forma:

$$\mathbf{m}_i^{j\to k} = \underbrace{\begin{bmatrix} d_x^{j\to k} \\ d_y^{j\to k} \end{bmatrix}}_{\boldsymbol{\delta}_m} + \underbrace{\begin{bmatrix} \cos(\theta^{j\to k}) & \sin(\theta^{j\to k}) \\ -\sin(\theta^{j\to k}) & \cos(\theta^{j\to k}) \end{bmatrix}}_{\mathbf{A}_m} \begin{bmatrix} m_x^i \\ m_y^i \end{bmatrix} \tag{5.3}$$

Já a pose do robô j no sistema de referência do robô k é dada por:

$$\mathbf{x}_r^{j\to k} = \underbrace{\begin{bmatrix} \theta^{j\to k} \\ d_x^{j\to k} \\ d_y^{j\to k} \end{bmatrix}}_{\boldsymbol{\delta}_r} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta^{j\to k}) & \sin(\theta^{j\to k}) \\ 0 & -\sin(\theta^{j\to k}) & \cos(\theta^{j\to k}) \end{bmatrix}}_{\mathbf{A}_r} \begin{bmatrix} \theta^j \\ x^j \\ y^j \end{bmatrix} \tag{5.4}$$

Definindo o vetor $\boldsymbol{\Delta}$ e a matrix $\mathbf{A}$ como:

$$\boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\delta}_r & \boldsymbol{\delta}_m & \dots & \boldsymbol{\delta}_m \end{bmatrix}^T \tag{5.5}$$

e

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_r & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_m & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_m \end{bmatrix} \tag{5.6}$$

pode-se reescrever a mudança de referência do vetor de estados do robô j para o sistema de referência do robô k, como:

$$\mathbf{x}^{j\to k} = \boldsymbol{\Delta} + \mathbf{A}\,\mathbf{x}^{j} \tag{5.7}$$

Com o resultado, a mudança de sistema de referência do robô j para o do robô k, mostrado na Equação 5.7, e usando como hipótese que não há ganho ou perda de informação ao mudar o sistema de referência dos vetores e matrizes do robô j para o robô k, ou seja que

$$p\left(\mathbf{x}_t^{j\to k} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = p\left(\mathbf{x}_t^{j} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \tag{5.8}$$

chegamos aos valores de $\boldsymbol{\xi}^{j\to k}$ (vetor de informação) e $\boldsymbol{\Omega}^{j\to k}$ (matriz de informação) através do seguinte desenvolvimento:

$$\begin{aligned}
p\left(\mathbf{x}_t^{j\to k} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) &= \eta \exp\left(-\frac{1}{2}\left(\mathbf{x}^{j\to k}\right)^T \boldsymbol{\Omega}^{j\to k}\,\mathbf{x}^{j\to k} + \left(\mathbf{x}^{j\to k}\right)^T \boldsymbol{\xi}^{j\to k}\right) \\
&= \eta \exp\left(-\frac{1}{2}\left(\boldsymbol{\Delta} + \mathbf{A}\,\mathbf{x}^{j}\right)^T \boldsymbol{\Omega}^{j\to k}\,\left(\boldsymbol{\Delta} + \mathbf{A}\,\mathbf{x}^{j}\right) + \left(\boldsymbol{\Delta} + \mathbf{A}\,\mathbf{x}^{j}\right)^T \boldsymbol{\xi}^{j\to k}\right) \\
&= \begin{aligned}[t] &\eta \exp\Bigg(\underbrace{-\frac{1}{2}\boldsymbol{\Delta}^T\boldsymbol{\Omega}^{j\to k}\boldsymbol{\Delta}}_{\text{constante}} - \frac{1}{2}\boldsymbol{\Delta}^T\boldsymbol{\Omega}^{j\to k}\mathbf{A}\mathbf{x}^{j} - \frac{1}{2}\left[\mathbf{x}^{j}\right]^T \mathbf{A}^T\boldsymbol{\Omega}^{j\to k}\boldsymbol{\Delta} \\ &\quad - \frac{1}{2}\left[\mathbf{x}^{j}\right]^T \mathbf{A}^T\boldsymbol{\Omega}^{j\to k}\mathbf{A}\mathbf{x}^{j} + \underbrace{\boldsymbol{\Delta}^T\boldsymbol{\xi}^{j\to k}}_{\text{constante}} + \left[\mathbf{x}^{j}\right]^T \mathbf{A}^T\boldsymbol{\xi}^{j\to k}\Bigg)\end{aligned} \\
&= \eta^* \exp\left(-\frac{1}{2}\left[\mathbf{x}^{j}\right]^T \left(\mathbf{A}^T\boldsymbol{\Omega}^{j\to k}\mathbf{A}\right)\mathbf{x}^{j} - \left[\mathbf{x}^{j}\right]^T \mathbf{A}^T\boldsymbol{\Omega}^{j\to k}\boldsymbol{\Delta} + \left[\mathbf{x}^{j}\right]^T \mathbf{A}^T\boldsymbol{\xi}^{j\to k}\right) \\
&= \begin{aligned}[t] &\eta^* \exp\Bigg(-\frac{1}{2}\left[\mathbf{x}^{j}\right]^T \underbrace{\left(\mathbf{A}^T\boldsymbol{\Omega}^{j\to k}\mathbf{A}\right)}_{\boldsymbol{\Omega}^{j}}\mathbf{x}^{j} \\ &\quad + \left[\mathbf{x}^{j}\right]^T \underbrace{\left(-\mathbf{A}^T\boldsymbol{\Omega}^{j\to k}\boldsymbol{\Delta} + \mathbf{A}^T\boldsymbol{\xi}^{j\to k}\right)}_{\boldsymbol{\xi}^{j}}\Bigg)\end{aligned}
\end{aligned} \tag{5.9}$$

em que a cor azul é utilizada para mostrar ao leitor a correspondência de termos entre as linhas. Na Equação 5.9 foi utilizada para mostrar como os termos constantes (invariantes no tempo) saem do expoente e são incorporados à constante de normalização.

Utilizando a equivalência da Equação 5.8, têm-se:

$$\boldsymbol{\Omega}^{j} = \mathbf{A}^{T}\boldsymbol{\Omega}^{j\to k}\mathbf{A} \tag{5.10}$$

$$\boldsymbol{\xi}^{j} = -\mathbf{A}^{T}\boldsymbol{\Omega}^{j\to k}\boldsymbol{\Delta} + \mathbf{A}^{T}\boldsymbol{\xi}^{j\to k} \tag{5.11}$$

e como $\mathbf{A}$ é ortogonal ($\mathbf{A}^{-1} = \mathbf{A}^{T}$), temos que $\boldsymbol{\xi}^{j\to k}$ e $\boldsymbol{\Omega}^{j\to k}$ são obtidos da seguinte forma:

$$\boldsymbol{\Omega}^{j\to k} = \mathbf{A}\boldsymbol{\Omega}^{j}\mathbf{A}^{T} \tag{5.12}$$

$$\boldsymbol{\xi}^{j\to k} = \mathbf{A}\left(\boldsymbol{\xi}^{j} + \boldsymbol{\Omega}^{j}\mathbf{A}^{T}\boldsymbol{\Delta}\right) \tag{5.13}$$

Com o vetor de informação e a matriz de informação do robô j reescritos no sistema de referência do robô k, Equações 5.13 e 5.12 respectivamente, criamos as seguintes grandezas aumentadas:

$$\boldsymbol{\xi}_{t}^{*} = \begin{bmatrix} \left[\boldsymbol{\xi}_{t}^{k}\right]^{T} & \left[\boldsymbol{\xi}_{t}^{j\to k}\right]^{T} \end{bmatrix}^{T} \tag{5.14}$$

$$\boldsymbol{\Omega}_{t}^{*} = \begin{bmatrix} \boldsymbol{\Omega}_{t}^{k} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_{t}^{j\to k} \end{bmatrix} \tag{5.15}$$

O vetor e a matriz de informação aumentados são simplesmente o vetor e matriz do robô k concatenados com o vetor e matriz de informação do robô j transformados para o sistema de referência do robô k. Portanto, há mais de uma linha/bloco que possui informação relacionada a uma mesma *landmark* que foi observada por ambos os robôs separadamente, então resta ainda fundir linhas e blocos que dizem respeito à mesma *landmark*. Para isso, basta somá-los; no exemplo abaixo, supõe-se que os blocos 2 e 4 do vetor e matriz de informação correspondem ao mesmo objeto:

$$\begin{bmatrix} \boldsymbol{\Omega}_{11} & \boldsymbol{\Omega}_{12} & \boldsymbol{\Omega}_{13} & \boldsymbol{\Omega}_{14} \\ \boldsymbol{\Omega}_{21} & \boldsymbol{\Omega}_{22} & \boldsymbol{\Omega}_{23} & \boldsymbol{\Omega}_{24} \\ \boldsymbol{\Omega}_{31} & \boldsymbol{\Omega}_{32} & \boldsymbol{\Omega}_{33} & \boldsymbol{\Omega}_{34} \\ \boldsymbol{\Omega}_{41} & \boldsymbol{\Omega}_{42} & \boldsymbol{\Omega}_{43} & \boldsymbol{\Omega}_{44} \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{\Omega}_{11} & \boldsymbol{\Omega}_{12} + \boldsymbol{\Omega}_{14} & \boldsymbol{\Omega}_{13} \\ \boldsymbol{\Omega}_{21} + \boldsymbol{\Omega}_{41} & \boldsymbol{\Omega}_{22} + \boldsymbol{\Omega}_{42} + \boldsymbol{\Omega}_{24} + \boldsymbol{\Omega}_{44} & \boldsymbol{\Omega}_{23} + \boldsymbol{\Omega}_{43} \\ \boldsymbol{\Omega}_{31} & \boldsymbol{\Omega}_{32} + \boldsymbol{\Omega}_{34} & \boldsymbol{\Omega}_{33} \end{bmatrix} \tag{5.16}$$

$$\begin{bmatrix} \boldsymbol{\xi}_{1} \\ \boldsymbol{\xi}_{2} \\ \boldsymbol{\xi}_{3} \\ \boldsymbol{\xi}_{4} \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{\xi}_{1} \\ \boldsymbol{\xi}_{2} + \boldsymbol{\xi}_{4} \\ \boldsymbol{\xi}_{3} \end{bmatrix} \tag{5.17}$$

Por último, depois da fusão dos blocos da matriz e do vetor de informação, é necessário ajustar os elementos correspondentes do vetor média. Para isso foi utilizado o método descrito no Algoritmo 3.2, mas ao invés da otimização ser realizada num conjunto de *landmarks* observadas, ela é realizada no conjunto de *landmarks* que foram observadas

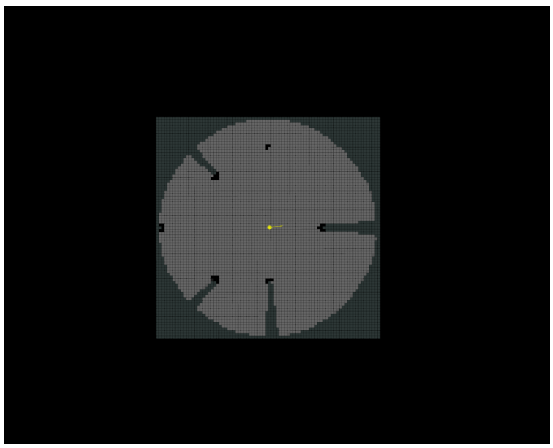
por ambos os robôs do par e, portanto, tiveram seus elementos fundidos.

5.2.2 Troca de grades de ocupação

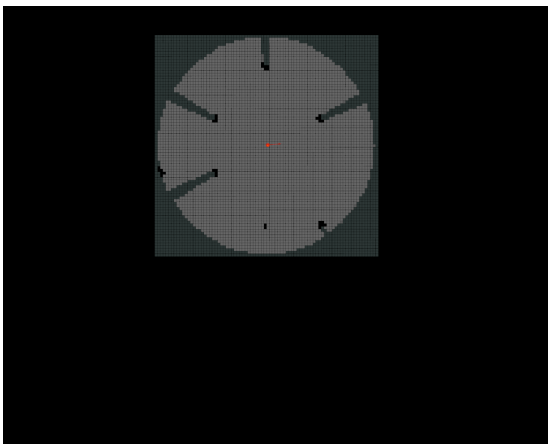
Similarmente a troca do vetor e matriz de informação, as células da grade de ocupação de um robô são projetadas na do outro por meio da transformação $\mathcal{T}_r^{j\to k}$. O logaritmo da razão de chance da i-ésima célula da grade de ocupação do robô k é atualizada da seguinte forma:

$$l_i^k = l_i^{j\to k} + l_i^k - l_0^k \tag{5.18}$$

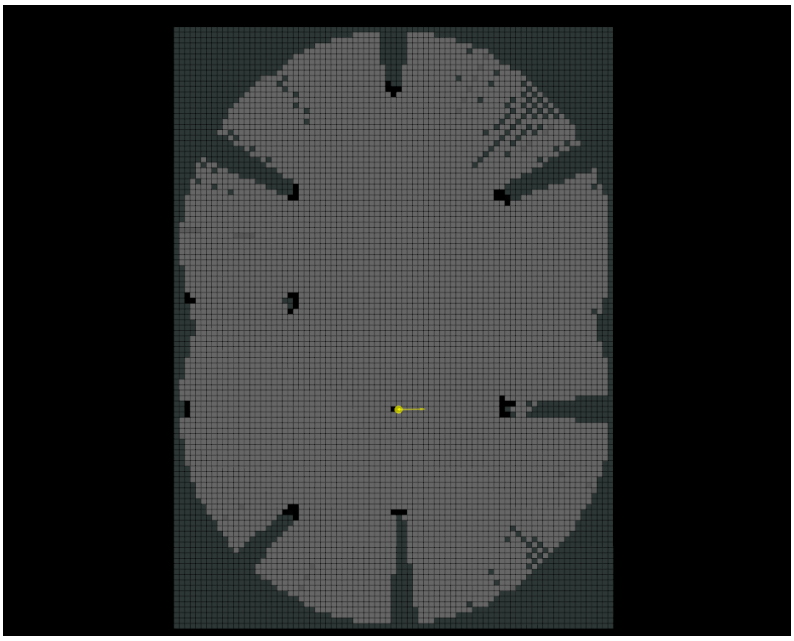
A Figura 5.3 ilustra a fusão das grades de ocupação de um par de robôs. É possível observar que um robô passa a ter informações de células que não podem ser observadas do seu ponto de vista, mas que foram observadas pelo outro robô.



(a) Mapa de grade do robô 1. O robô está no centro da grade em amarelo.



(b) Mapa de grande do robô 2. O robô está no centro da grade em vermelho.



(c) Mapa de grade do robô 1 após incorporar o mapa de grade do robô 2.

FIGURA 5.3 – As imagens menores representam os mapas de grade individuais de cada robô. A imagem maior representa o mapa de grade do primeiro robô após receber o mapa de grade do segundo. Note que algumas células com probabilidade de ocupação 0.5 (nem ocupada e nem desocupada) pelo robô 1 passam a ter probabilidade de ocupação diferente de 0.5, pois são observadas do ponto de vista do robô 2.

6 Metodologia

Neste capítulo é disponibilizada toda a implementação do trabalho assim como as configurações do ambiente computacional em que foi desenvolvida. Também são explicitados bibliotecas e demais *softwares* utilizados que, juntamente com o *software* desenvolvido, compõem o sistema investigado.

6.1 *Software*

Todo o software desenvolvido neste trabalho, utilizado nas investigações e na experimentação, está disponibilizado no repositório <https://github.com/wvmcastro/slam> sob licença de uso GPLv3 que permite a todos utilizá-lo e modificá-lo livremente.

Toda a teoria apresentada nos capítulos 3, 4 e 5 foi implementada; a única exceção é a exploração autônoma apresentada na Seção 4.3, na qual foram usados utilitários da biblioteca de navegação (MARDER-EPPSTEIN *et al.*, 2010) presente no Sistema Operacional de Robôs (ROS).

Grande parte da implementação foi feita na linguagem C++ por conta de seu alto desempenho e também pela série de verificações estáticas realizadas por seu compilador, o que evita uma enorme quantidade de erros de execução. Para garantir o consumo reduzido de memória viabilizado pelo SEIF através do emprego de matrizes esparsas, foi utilizada versão 3.3.7 da biblioteca de álgebra linear Eigen (JACOB; GUENNEBAUD, 2018) disponível para a linguagem de programação C++. As implementações de visualizações de gráficos foram realizadas na linguagem Python versão 3, utilizando a biblioteca Matplotlib (HUNTER, 2007).

6.2 Ambiente computacional

As Seções 6.2.1 e 6.2.2 descrevem em detalhes o *hardware* e *software* utilizados durante o desenvolvimento da pesquisa com o objetivo de guiar e/ou servir de ponto de partida para futuras pesquisas em temas relacionados a este trabalho.

6.2.1 *Hardware*

Tanto para o desenvolvimento quanto para a execução de experimentos foi utilizado um *laptop* da marca Dell modelo G7 com as seguintes configurações:

- Processador: Intel® Core™ i7-8750H CPU @ 2.20GHz × 12
- Placa de vídeo: nVidia GP106M (GeForce GTX 1060 *Mobile*)
- Memória RAM:
  - Capacidade: 16 gigabytes
  - Modelo: DDR4
  - Velocidade: 2400 MT/s
- Memória não volátil:
  - Disco rígido (*hard drive*): TOSHIBA MQ02ABF1, capacidade de 1 terabyte
  - Unidade de estado sólido (SSD): Micron 1100 SATA M.2, capacidade de 256 gigabytes

6.2.2 *Software*

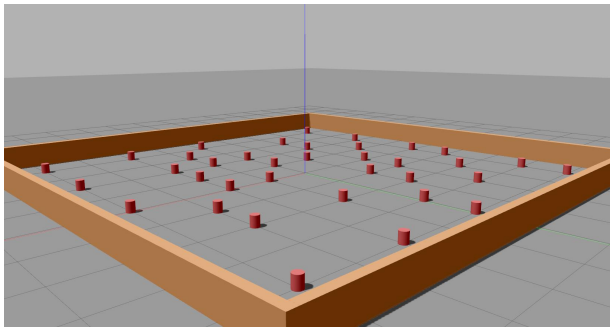
O *hardware* descrito na Seção 6.2.1 foi equipado com os seguintes *softwares*:

- Sistema Operacional: Linux, distribuição Ubuntu 20.04 LTS
- *Robot Operating System (ROS)* 1, versão *Noetic Ninjemys*
- Simulador Gazebo clássico versão 11

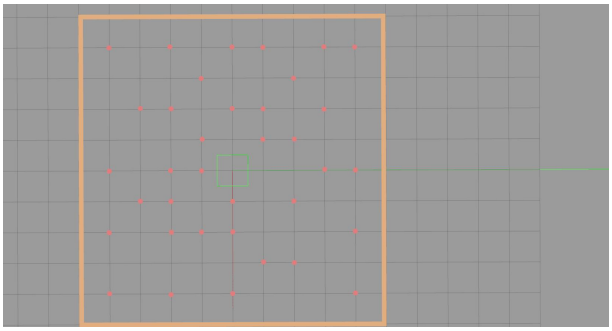
6.3 Simulação

6.3.1 O ambiente

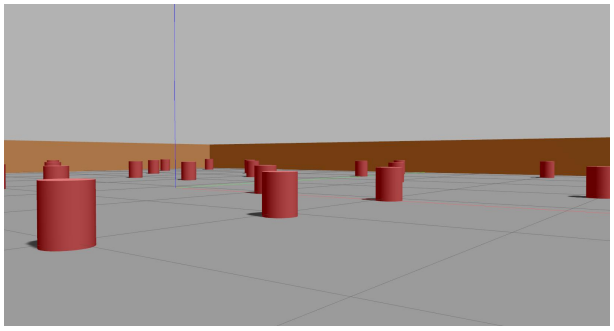
O ambiente consiste em uma espaço de $100m^2$ delimitado por paredes e diversos "postes" de formato cilíndrico de $16cm$ de diâmetro, e está representado na Figura 6.1.



(a) Vista perspectiva ampla



(b) Vista ortográfica superior



(c) Vista perspectiva fechada

FIGURA 6.1 – Diferentes vistas do ambiente simulado

6.3.2 O robô

Neste trabalho foi utilizado o modelo do robô *Turtlebot 3* (ROBOTIS, 2021) exibido na Figura 6.2, que é um robô de acionamento diferencial equipado com *encoder* de rodas, uma IMU e um sensor laser do tipo LiDAR.

FIGURA 6.2 – Modelo do robô *Turtlebot 3*. O sensor LiDAR encontra-se no topo do robô.

7 Experimentos e resultados

Nesta Seção são apresentados experimentos repetidos a exaustão para avaliar o desempenho médio esperado do sistema. E também outros experimentos preliminares.

7.1 Comparação SEIF vs Odometria

Este primeiro experimento visa medir a capacidade do SEIF para estimar a pose do robô enquanto ele explora o ambiente de maneira autônoma, mediante comparação da estimativa com a odometria das rodas. Além disso, compara-se a estimativa do SEIF variando a cardinalidade do conjunto de *landmarks* ativas $\{\mathbf{m}^+\}$. O critério de comparação utilizado foi o Erro Absoluto Integral (IAE) (Equação 7.1) da posição estimada $\hat{\mathbf{x}}$, em relação à pose real $\mathbf{x}$ colhida do simulador. Para repetir o experimento com outro tamanho de $\{\mathbf{m}^+\}$ foram gravadas a sequência de entradas $\mathbf{u}_{1:t}$ (posição angular das rodas) e a sequência de leituras $\mathbf{y}_{1:t}$ do sensor LiDAR.

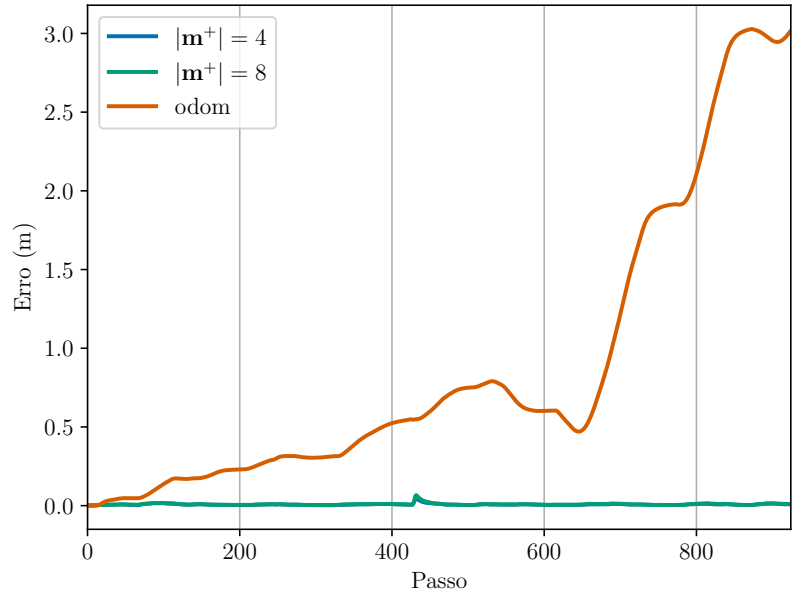
$$\text{IAE}(\hat{\mathbf{x}}, \mathbf{x}) = \sum \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2 \tag{7.1}$$

A Tabela 7.1 lista os valores obtidos para o IAE e a Figura 7.1 ilustra a evolução temporal do erro da estimativa de posição do robô. Além da posição, também foi avaliado o erro na estimação da orientação, ilustrado na Figura 7.2. Em ambas as Figuras, os valores obtidos pelo SEIF são mostrados separadamente, pois aparecem sobrepostos na escala do erro da odometria.

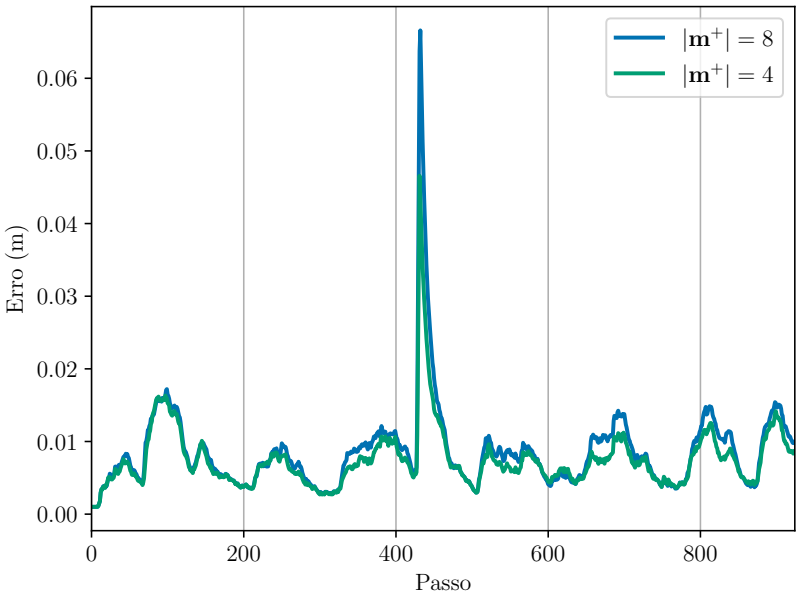
Como esperado, ambas as configurações do SEIF estimaram de maneira satisfatória tanto a posição quanto a orientação do robô quando comparadas com a estimação da

TABELA 7.1 – Erro Integral Absoluto da trajetória do robô para diferentes estimadores

Fonte da estimativa	IAE (m)
Odometria	846.01
SEIF $\mathbf{m}^+ = 4$	**7.01**
SEIF $\mathbf{m}^+ = 8$	8.04

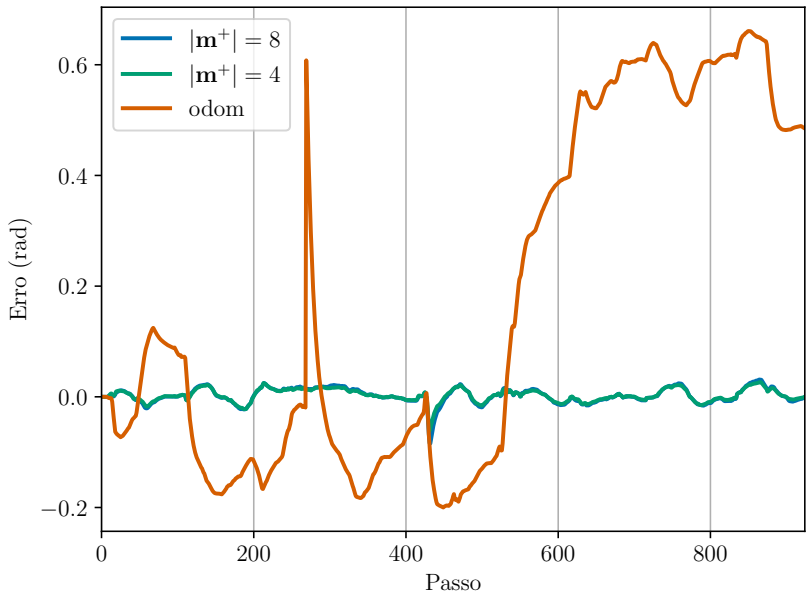


(a) Evolução temporal do erro da estimativa de posição pela odometria e duas configurações do SEIF com $|\mathbf{m}^+| = 4$ e $|\mathbf{m}^+| = 8$.

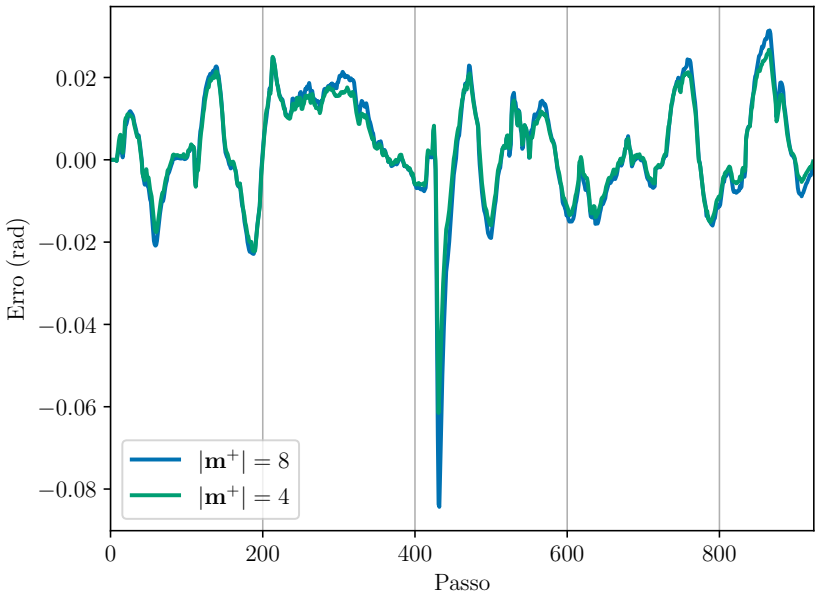


(b) Evolução temporal do erro da estimativa de posição pelas duas configurações do SEIF com $|\mathbf{m}^+| = 4$ e $|\mathbf{m}^+| = 8$.

FIGURA 7.1 – Erros de estimação da posição do robô durante exploração parcial do ambiente da Figura 6.1



(a) Evolução temporal do erro da estimativa de orientação do robô pela odometria e pelas duas configurações do SEIF com $|\mathbf{m}^+| = 4$ e $|\mathbf{m}^+| = 8$.



(b) Evolução temporal do erro da estimativa da orientação do robô pelas duas configurações do SEIF com $|\mathbf{m}^+| = 4$ e $|\mathbf{m}^+| = 8$.

FIGURA 7.2 – Erros de estimação da orientação do robô durante exploração parcial do ambiente da Figura 6.1

TABELA 7.2 – Quantidade de elementos não nulos da matriz de informação esparsa para diferentes configurações do SEIF-SLAM

Cardinalidade do conjunto $\{\mathbf{m}^+\}$	Elementos não nulos da matriz de informação	Economia em relação ao EKF-SLAM[1]
2	1721	69.40%
4	1879	66.60%
8	2021	64.07%

odometria. Porém, nota-se que a configuração com a cardinalidade do conjunto $\{\mathbf{m}^+\}$ menor estimou melhor tanto a posição quanto a orientação, o que vai contra o consenso geral e a observação de Thrun *et al.* (2005, p. 408) de que quanto maior o conjunto $\mathbf{m}^+$ melhor é a estimativa. A causa dessa divergência não foi entendida até o momento da escrita deste texto; além disso, precisa ser verificada de maneira sistemática através de repetidas simulações.

7.2 Comparações sobre uso de memória

Neste experimento foram comparados o uso de memória de diferentes configurações do SEIF num mapa com 36 *landmarks* com o uso que a abordagem mais clássica EKF-SLAM utilizaria. Nesse ensaio o robô foi teleoperado, e para garantir a reprodução exata do mesmo cenário as entradas e leituras foram novamente gravadas para alimentar as diferentes configurações do SEIF.

As quantidades de elementos não nulos da matriz de informação de cada parametrização estão compiladas na Tabela 7.2 e um comparativo com a quantidade de memória que seria utilizado pelo EKF-SLAM, com a mesma quantidade de *landmarks*, é ilustrado na Figura 7.3. Por fim, as matrizes de informação são mostradas na Figura 7.4 para ilustrar sua estrutura e quantidade de elementos nulos, que não ocupam memória quando utilizada uma estrutura de dados adequada para a representação de matrizes esparsas.

O perfil de uso de memória observado mostrou-se como esperado; quanto maior a cardinalidade do conjunto de *landmarks* ativas $\mathbf{m}^+$, maior o consumo de memória pelo SEIF-SLAM. Além disso, todas as parametrizações usam muito menos memória quando comparados com o EKF-SLAM.

[1]Embora esses valores correspondam ao número de elementos nulos em relação ao tamanho total da matriz de informação, eles não representam necessariamente economia de memória. Para que a matriz esparsa seja representada é necessário utilizar memória extra para organizar o espaço de armazenamento dos elementos. Porém, essa quantidade tende a ser irrisória quando comparada com a quantidade total de elementos de uma matriz densa. Na estrutura de dados utilizada nesse trabalho, a quantidade de memória utilizada para armazenar a matriz esparsa é da ordem de $\mathcal{O}(2n + N)$, onde n é a quantidade de elementos não nulos e N é a dimensão da matriz. Porém, esses valores podem variar dependendo da técnica utilizada para armazenar a matriz.

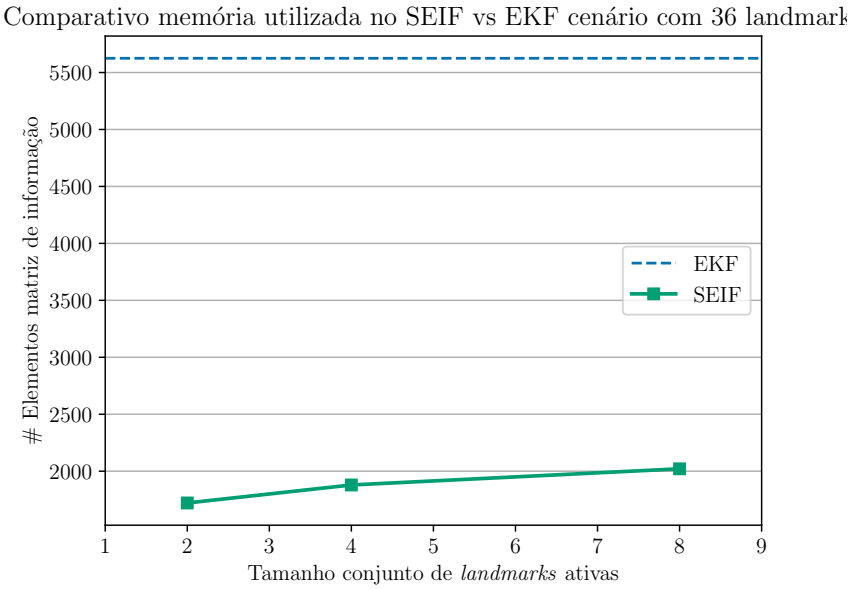


FIGURA 7.3 – Uso de memória para diferentes parametrizações do SEIF-SLAM com diferentes tamanhos (2, 4, 8) do conjunto de *landmarks* ativas. A linha tracejada representa a quantidade de elementos que a matriz do EKF-SLAM teria para o mesmo cenário.

7.3 Desempenho de mapeamento de um único agente

Para avaliar o desempenho de mapeamento de dois ou mais agentes resolvendo o problema SLAM de maneira distribuída e descentralizada, primeiro é necessário entender o desempenho de um único agente resolvendo a mesma tarefa para assim possuirmos uma base de comparação.

Para isso um único agente foi submetido, repetidamente, à tarefa de mapeamento do ambiente apresentado na Figura 6.1. Cada uma dessas simulações poderia assumir um dos seguintes resultados:

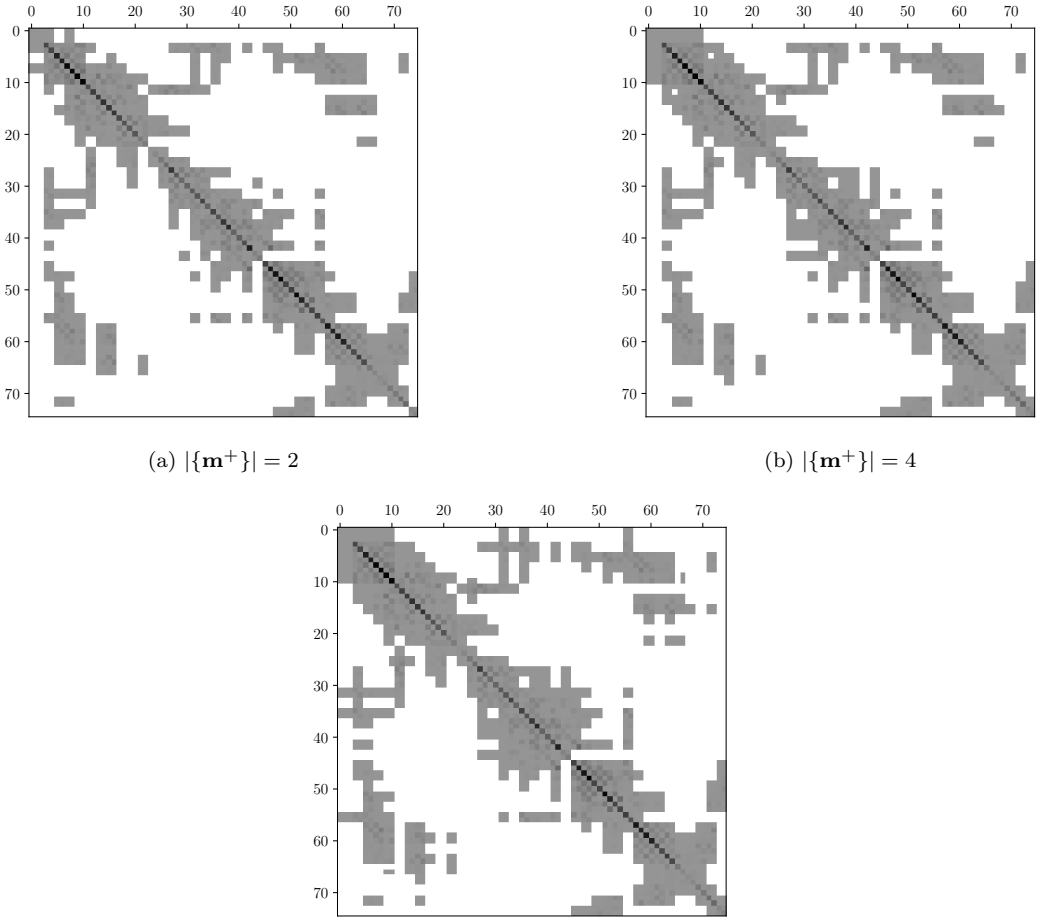
1. **Sucesso**: o robô conseguiu mapear 90% ou mais da área sem divergência na estimação de sua posição dentro de um tempo limite estipulado.

2. **Falha por tempo limite excedido**: O robô não conseguiu mapear ao menos 90% da área antes do limite de tempo (foram dados 17 minutos)[2].

3. **Falha por divergência**: Quando ao longo da exploração a posição estimada pelo SEIF diferiu em mais de um metro da posição real do robô.

Para medir o desempenho de mapeamento do robô, computa-se a área coberta por ele ao longo do tempo. A área coberta é definida como a soma das áreas de todas as células

[2]Esse tempo foi escolhido após sucessivas observações do robô desempenhando SLAM no ambiente utilizado; em nenhuma dessas observações o robô levou mais do que 10 minutos; então foi definido o valor de 17 minutos considerando-se uma margem de segurança



(a) $|\{\mathbf{m}^+\}| = 2$

(b) $|\{\mathbf{m}^+\}| = 4$

(c) $|\{\mathbf{m}^+\}| = 8$

FIGURA 7.4 – Representação das matrizes de informação correspondentes aos experimentos da Tabela 7.2. Os elementos nulos são representados pela cor branca, e os não nulos em escala de cinza. Quanto mais escuro o tom de cinza, maior a magnitude do valor representado.

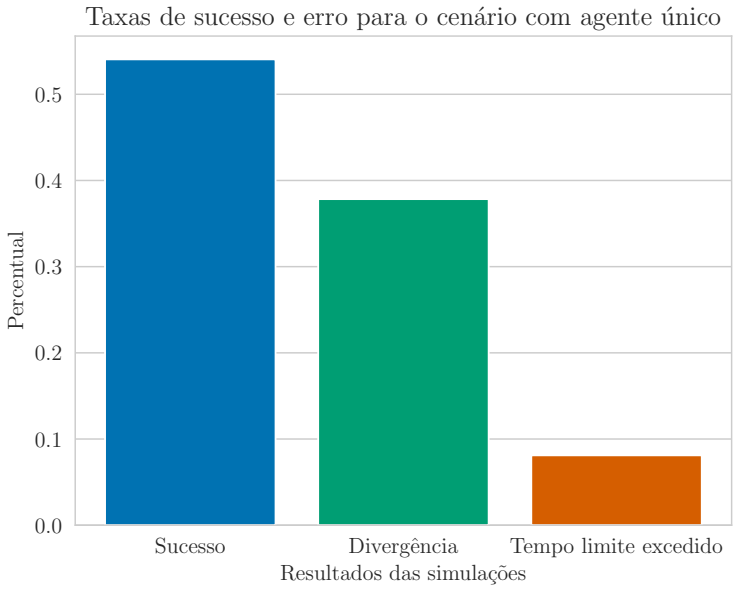


FIGURA 7.5 – Taxas de sucesso e falha por tempo limite excedido ou divergência para o cenário com um único agente.

da grade de ocupação que possuem probabilidade de ocupação diferente de 50%.

No total foram realizadas 111 (cento e onze[3]) simulações variando-se aleatoriamente (seguindo uma distribuição uniforme) a posição inicial do agente. Em 60 foi obtido sucesso, 9 falharam por tempo limite excedido e 42 falharam por divergência. O comparativo pode ser observado na Figura 7.5.

Com o objetivo de obter-se o desempenho médio do sistema com um único agente foi colhido o tempo de mapeamento de cada simulação em que o robô obteve sucesso, esses dados se encontram nas tabelas C.1 e C.2 no Apêndice C.1. A distribuição dos tempos de mapeamento registrados encontra-se ilustrada na Figura 7.6; o tempo de mapeamento médio foi de 429 segundos com desvio padrão de 58 segundos. A Figura 7.7 apresenta o caminho percorrido por quatro realizações do mapeamento por um único agente.

Durante a realização destes experimentos, criou-se a impressão de que agentes com posição inicial longe das extremidades do ambiente obtinham melhor tempo de mapeamento. Para analisar essa hipótese foi feita a Figura 7.8, que relaciona a posição inicial do agente com seu tempo de mapeamento através da cor.

A Figura confirma essa tendência pois nela é possível notar que a maior parte dos pontos azul claros (representando menor tempo de mapeamento) se encontram longe das

[3]O objetivo era realizar ao menos 100 simulações, pois de acordo com Paula *et al.* (2014) é um bom número base de repetições, sendo que mais repetições podem ser executadas de conforme a necessidade imposta pelo valor de desvio-padrão desejado do estimador. Porém, como as simulações foram executadas em *loop* e não eram supervisionadas em alguns períodos, acabou-se executando algumas extras até que fossem paradas; resolvemos então utilizar todos os dados disponíveis.

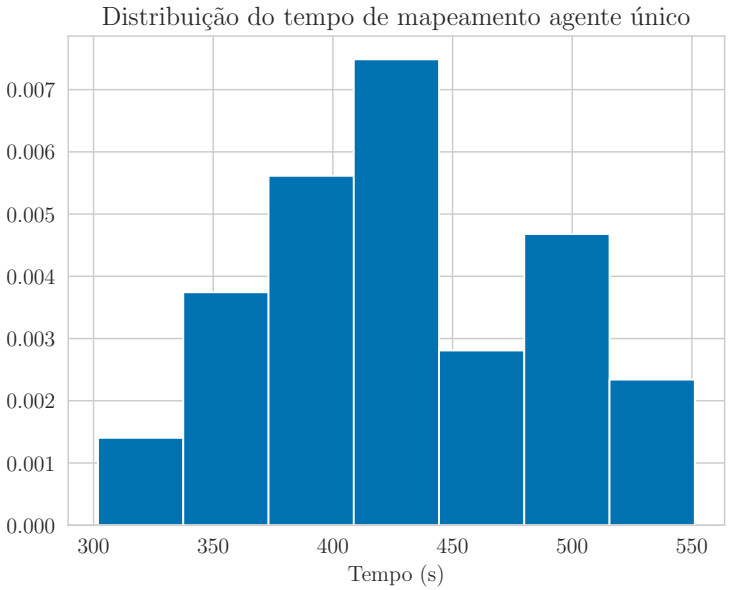


FIGURA 7.6 – Histograma de distribuição dos tempos de mapeamento das simulações com um único agente.

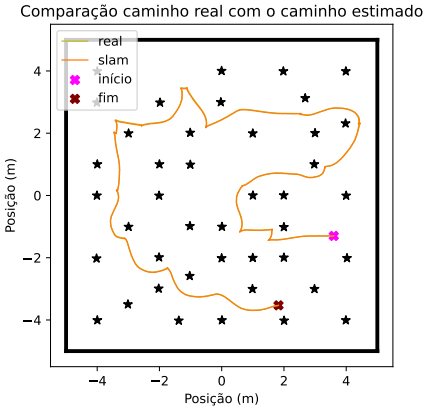
bordas, enquanto há poucos pontos avermelhados (maior tempo de mapeamento) na região central do mapa.

7.4 Mapeamento conjunto e descentralizado

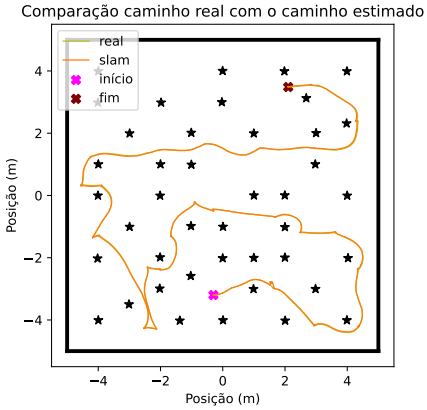
Esse experimento avalia o cenário com dois robôs desempenhando SLAM e trocando mapas entre si como mostrado na Seção 5.2. Como tanto o vetor e a matriz de informação quanto as grades de ocupação são compartilhados entre os agentes, esse experimento é dividido em duas partes.

Na primeira, o objetivo é medir a capacidade dos agentes de trocar os vetores e matrizes de informação entre si (Seção 5.2.1) no cenário ideal onde a transformação entre seus sistemas de referências é exata e conhecida; para isso são fornecidas as poses iniciais de cada agente. Já na segunda parte, os agentes não conhecem suas poses iniciais e usam o registro de nuvem de pontos para calcular as transformações.

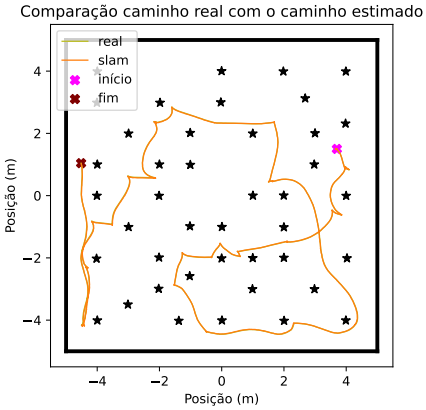
Nos cenários multiagente, a comunicação ocorre sempre que dois robôs chegam a uma distância de dois metros um do outro e há um intervalo mínimo de 60 segundos entre duas comunicações consecutivas de um mesmo par de robôs. Erros de comunicação ou degeneração de mensagens não são simulados.



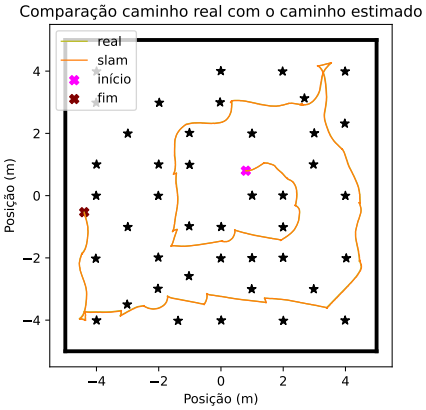
(a) Mapeamento em 302 segundos



(b) Mapeamento em 427 segundos



(c) Mapeamento em 429 segundos



(d) Mapeamento em 551 segundos

FIGURA 7.7 – Caminhos percorridos em diferentes realizações do mapeamento com um único agente em que (a) melhor tempo de mapeamento; (b) realização com tempo levemente inferior ao tempo médio; (c) realização com tempo levemente superior ao tempo médio; (d) pior tempo de mapeamento. Os caminhos real e estimado aparecem sobrepostos na escala da figura. As estrelas representam os pontos de referência (*landmarks*) do ambiente.

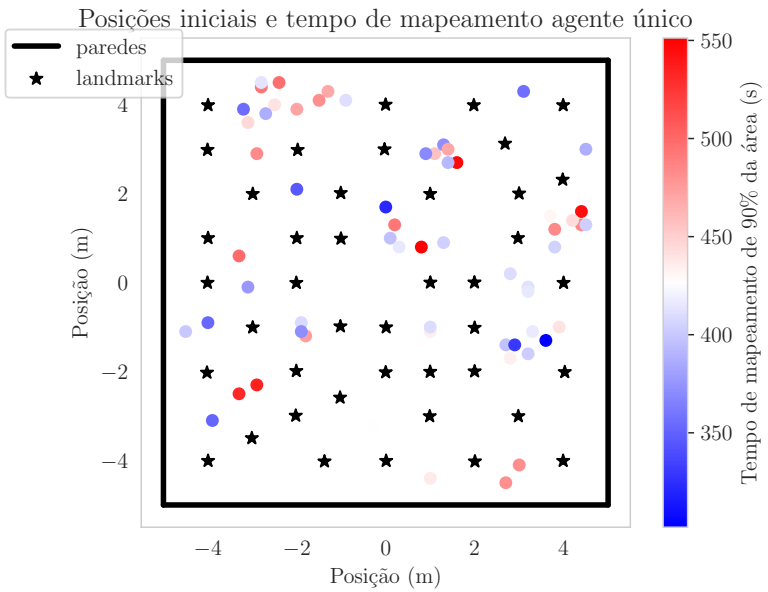


FIGURA 7.8 – Posição inicial e tempo de mapeamento das simulações com um único agente. Cada partícula representa uma posição inicial, o tempo é expressado pela sua cor.

7.4.1 Pose inicial conhecida

Além do ganho de redundância, sistemas multiagentes também podem apresentar ganho de eficiência ao dividir a carga de trabalho entre os robôs. Para analisar o desempenho de mapeamento com dois agentes, foram realizadas diversas simulações variando a posição inicial dos agentes, nos moldes do experimento anterior com um único agente.

Neste experimento foram realizadas 126 simulações, pois assim como no experimento anterior o objetivo era realizar aos menos 100 repetições conforme os passos apresentados em (PAULA *et al.*, 2014, p. 52); destas, em 46 os agentes obtiveram sucesso ao mapear o ambiente sem que a estimação de nenhum divergisse, 22 falharam por tempo limite excedido e 58 falharam por erro de estimação (divergência), o comparativo está na Figura 7.9. Os dados das realizações que obtiveram sucesso estão registrados nas tabelas C.3, C.4, C.5 e C.6 no Apêndice C.2.

Constata-se então que houve aumento de falhas tanto por divergência quanto por tempo limite excedido em comparação com o cenário com um único agente. O aumento das falhas por tempo limite excedido pode ser explicado pelo aumento de cenários em que a biblioteca de navegação do ROS fica presa. Isso ocorre quando um robô fica a uma distância de um obstáculo abaixo de um limiar de segurança e para de se movimentar para evitar uma colisão, sendo que esse problema é mais comum em ambientes dinâmicos (ZHENG, 2021). Já o aumento de falhas por divergência do filtro ocorre pois a chance de um *ou* outro agente falhar é maior do que a de um único agente.

A distribuição do tempo de mapeamento das simulações bem-sucedidas com dois agen-

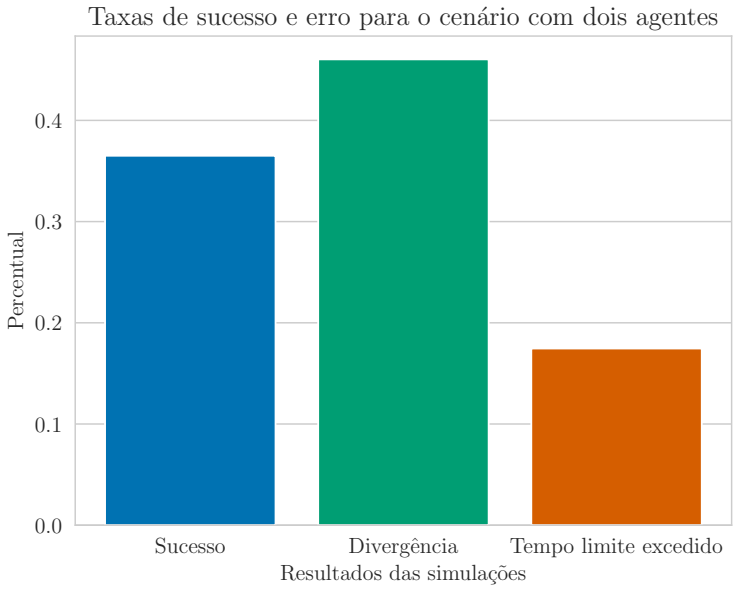


FIGURA 7.9 – Taxas de sucesso e falha por tempo limite excedido ou divergência para o cenário com dois agentes.

tes se encontra na Figura 7.10; o tempo de mapeamento médio foi de 321 segundos com desvio padrão de 62 segundos. Esse tempo representa 75% do tempo de mapeamento médio de um único agente, mostrado na Figura 7.6. A relação das posições iniciais, de ambos os robôs em todas as realizações, com o tempo de mapeamento pode ser observada na Figura 7.11. Na Figura 7.12 são também apresentados, como no experimento anterior, os caminhos percorridos pelos robôs em quatro realizações das simulações com dois agentes.

Embora com o dobro de agentes, o sistema não obteve o dobro do desempenho no tempo de exploração. Isso pode ser explicado por dois motivos: o primeiro é que não há uma política de exploração conjunta acordada entre os agentes; cada um explora o ambiente com base nas fronteiras presentes nos seus mapas individuais sem levar em consideração a posição de seu par em relação a essas fronteiras; o segundo motivo é que a navegação se torna mais complexa, pois agora um agente deve se desviar do outro durante a exploração e também não fazem isso de maneira sincronizada e/ou combinada.

7.4.2 Pose inicial desconhecida

Nesse último experimento, as poses iniciais dos agentes não foram fornecidas, como no experimento anterior. Dessa forma para incorporarem um o mapa do outro é necessário que calculem a transformação entre seus mapas de *landmarks* utilizando a técnica de registro de nuvem de pontos discutida na Seção 5.1. A Figura 7.13 mostra duas simulações do sistema SLAM com dois robôs; entre elas variou-se apenas as posições iniciais dos agentes.

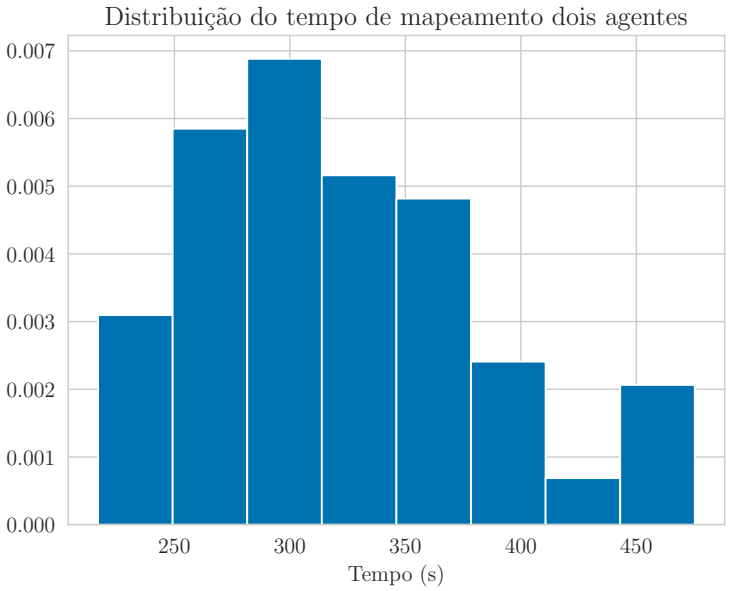


FIGURA 7.10 – Histograma de distribuição dos tempos de mapeamento das simulações com dois agentes.

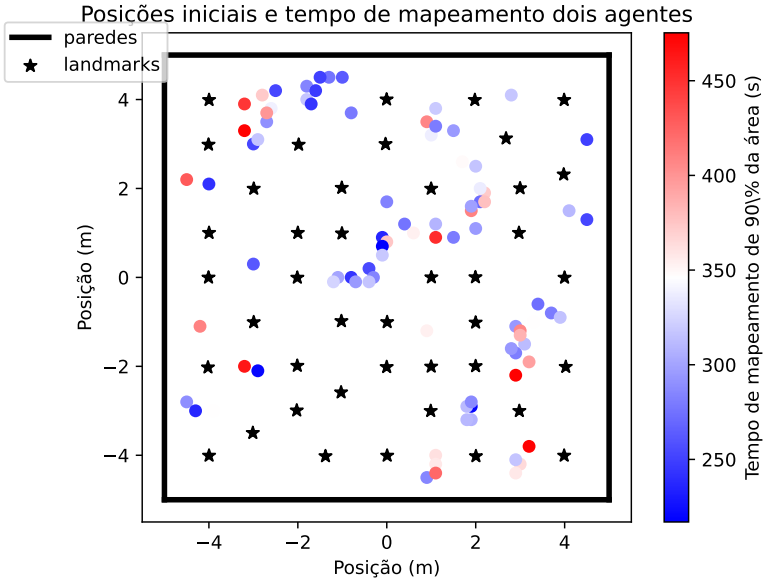
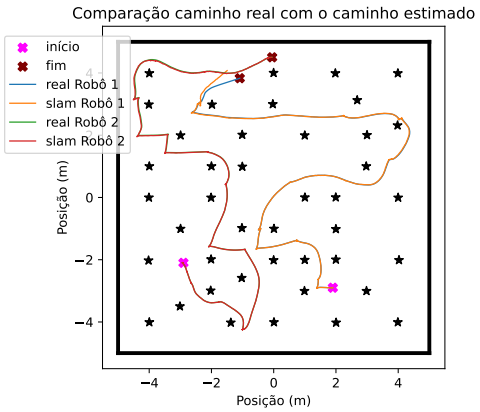
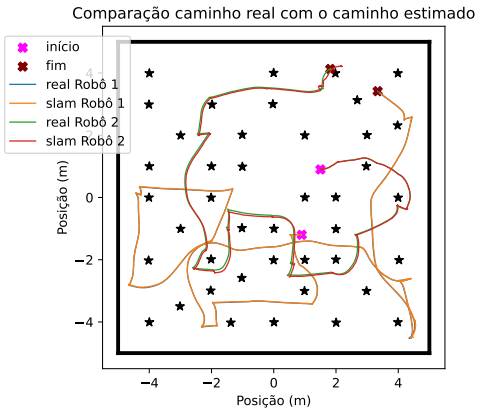


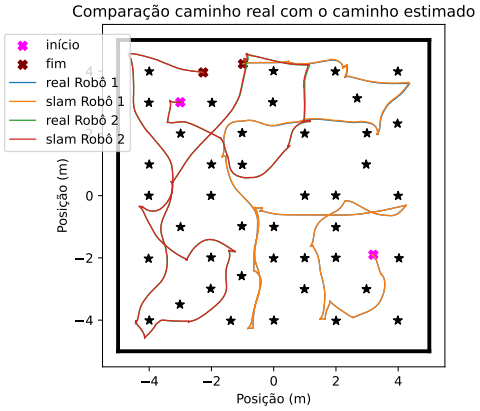
FIGURA 7.11 – Posição inicial e tempo de mapeamento de ambos os robôs nas simulações com dois agentes. Cada partícula representa uma posição inicial, o tempo é expressado pela sua cor.



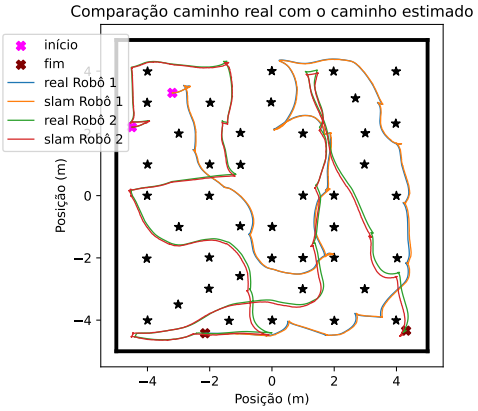
(a) Mapeamento médio em 220 segundos



(b) Mapeamento médio em 317 segundos



(c) Mapeamento médio em 324 segundos



(d) Mapeamento médio em 451 segundos

FIGURA 7.12 – Caminhos percorridos em diferentes realizações do mapeamento com dois agentes em que (a) melhor tempo médio de mapeamento; (b) realização com tempo médio levemente inferior ao tempo médio geral; (c) realização com tempo médio levemente superior ao tempo médio geral; (d) pior tempo de mapeamento. Os caminhos real e estimado podem aparecer sobrepostos na escala da figura. As estrelas representam os pontos de referência (*landmarks*) do ambiente.

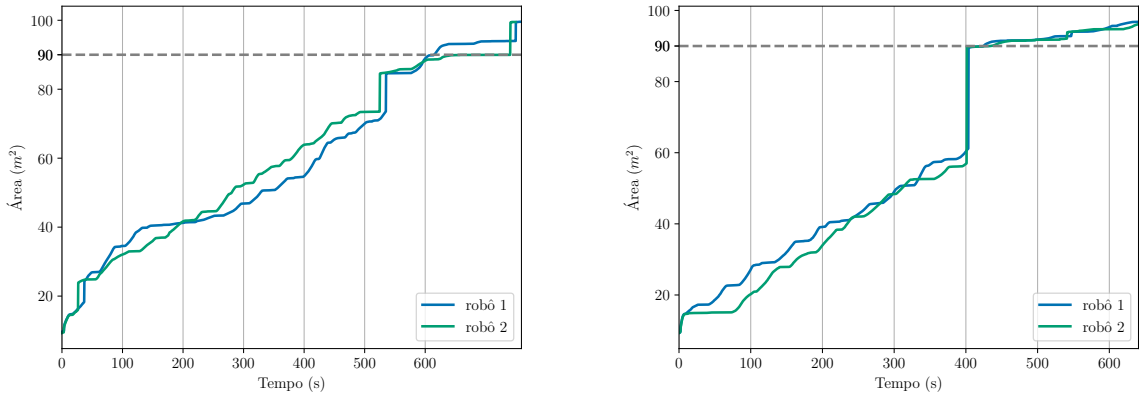


FIGURA 7.13 – Diferentes execuções do mapeamento com dois agentes com pose inicial desconhecida. Entre as execuções variou-se a pose inicial dos agentes. As curvas da Figura da direta estão defasadas, pois o segundo agente foi colocado no ambiente alguns instantes depois que o primeiro. Isso ocorreu pois neste experimento especificamente o processo de inicialização da simulação foi feito manualmente, o que provocou diferença no tempo em que os agente foram colocados no ambiente.

Além dos problemas de sincronia entre os agentes, já citados no experimento anterior, neste exemplo também foram observados problemas com a estimação da transformação entre os mapas de *landmarks* dos agentes. Enquanto no experimento anterior a troca de mapas entre os robôs sempre ocorre com sucesso, pois eles sabem suas poses iniciais e, portanto, sabem relacionar o mapa de um com o do outro, neste experimento nem toda aproximação dos robôs resultou em troca dos mapas, pois muitas vezes eles não conseguiram estimar a transformação.

Este experimento também teve que ser repetido mais de uma vez, pois em muitas execuções os robôs estimavam a transformação errada, incorporando a informação do outro robô de maneira errada e provocando divergência do filtro. Desse modo, a técnica de registro de nuvem de pontos utilizada não se mostrou robusta para estimar a transformação entre os mapas dos robôs. Por conta da alta taxa de falha na estimação da transformação relativa, que leva à divergência, não foram feitas realizações repetidas neste experimento para avaliar o desempenho.

8 Conclusão

Neste trabalho foi desenvolvido um sistema SLAM multiagente distribuído e descentralizado, no qual todo o processamento é feito pelos agentes sem a presença de uma figura reguladora central, utilizando o Filtro de Informação Estendido Esparso e explorando seu baixo uso de memória e tempo de processamento constante (desconsiderando-se associação de dados). Para seu desenvolvimento utilizou-se o *framework* ROS e o simulador Gazebo.

Essas duas ferramentas contam com ampla base de usuários e portanto há muito conteúdo em comunidades e fóruns que pode ser consultado ao longo do desenvolvimento para sanar dúvidas de integração e funcionamento. Além disso, como são ferramentas de código livre, podem ser utilizadas sem a necessidade de adquirir licenças; esse fato, somado à disponibilização integral da implementação realizada neste trabalho[1], potencializa a possibilidade de reuso e melhoramento deste por outros.

Outras contribuições são as deduções e ilustrações que foram produzidas para preencher lacunas dos livros e artigos consultados ao longo do desenvolvimento deste trabalho. Sendo as mais importantes: (I) A Seção 3.2.3 que apresenta dedução da covariância de uma nova *landmark* no EKF-SLAM. (II) A Figura 3.6, que ilustra a esparsidade das matrizes intermediárias utilizadas na etapa de predição do SEIF-SLAM. (III) A Seção 3.8 análoga ao que foi feito para o EKF-SLAM, traz a dedução da informação de uma nova *landmark* no SEIF-SLAM. (IV) A combinação da sequência de passos no processamento da leitura do sensor LiDAR, apresentada na Seção 4.1.2. (V) Errata da equação de transformação de sistema de referência do vetor e matriz de informação do SEIF apresentada em (THRUN *et al.*, 2005, Seção. 12.11.2), e aqui deduzida na Equação 5.9.

8.1 Trabalhos futuros

- **Investigação de resultados**: É preciso confirmar, através da repetição sistemática de simulações, a degradação no desempenho de estimação do SEIF ao aumentar a cardinalidade do conjunto de *landmarks* ativas, apresentada na Seção 7.1. Caso

[1]Implementação disponível em: <https://github.com/wvmcastro/slam>

confirmado, é necessário revisar a implementação para procurar possíveis erros já que esse resultado destoa da literatura e compreender quais circunstâncias causam esse efeito.

- **Coordenação entre agentes**: A falta de coordenação entre os agentes limitou o ganho de eficiência esperado de um sistema com mais de um robô, como discutido na Seção 7.4.1. Dessa forma, um trabalho futuro deve focar na coordenação entre agentes com o objetivo de dividir a carga de trabalho.

- **Troca de mapas**: Em relação à troca de mapas, há espaço para melhorias na técnica de registro de nuvem de pontos utilizada, possibilitando maior taxa de sucesso na troca de mapas no cenário onde a pose inicial dos agentes não é fornecida, como debatido na Seção 7.4.2.

- **Recuperação após falsa associação**: Outro aspecto que merece ser contemplado num trabalho futuro é a recuperação após uma falsa associação de medida e *landmark* e/ou a utilização de uma técnica de associação de dados mais robusta. Apesar da lista de *landmarks* provisórias ter atenuado esse problema, não foi suficiente para resolvê-lo, como mostram as taxas de falha por divergência das simulações apresentadas nos experimentos das Seções 7.3 e 7.4.1.

Referências

AL-SHARADQAH, A.; CHERNOV, N. Error analysis for circle fitting algorithms. **Electronic Journal of Statistics**, Institute of Mathematical Statistics and Bernoulli Society, v. 3, p. 886–911, 2009. 60

BRENNER, C. **Slam Lectures, SLAM A 06**. 2012. Disponível em: <https://www.youtube.com/playlistlist=PLpUPoM7Rgzi_7YWn14Va2FODh7LzADBSm>. Acesso em: 02/05/2022. 60

BRESENHAM, J. E. Algorithm for computer control of a digital plotter. **IBM Systems Journal**, IBM, v. 4, n. 1, p. 25–30, 1965. 65

CADENA, C.; CARLONE, L.; CARRILLO, H.; LATIF, Y.; SCARAMUZZA, D.; NEIRA, J.; REID, I.; LEONARD, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. **IEEE Transactions on Robotics**, IEEE, v. 32, n. 6, p. 1309–1332, 2016. 21, 23

COPPERSMITH, D.; WINOGRAD, S. Matrix multiplication via arithmetic progressions. **Journal of Symbolic Computation**, v. 9, n. 3, p. 251–280, 1990. ISSN 0747-7171. Computational algebraic complexity editorial. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0747717108800132>. 33, 55

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. **IEEE Robotics & Automation Magazine**, IEEE, v. 13, n. 2, p. 99–110, 2006. 18

DURRANT-WHYTE, H.; RYE, D.; NEBOT, E. Localization of autonomous guided vehicles. **Robotics Research**, Springer, p. 613–625, 1996. 18

ELFES, A. Using occupancy grids for mobile robot perception and navigation. **Computer**, IEEE, v. 22, n. 6, p. 46–57, 1989. 63

ELWIN, M. L. **Circle Fitting**. 2021. Disponível em: <https://web.archive.org/web/20230131094934/https://nu-msr.github.io/navigation_site/lectures/circle_fit.html>. Acesso em: 31/01/2023. 114

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Communications of the ACM**, ACM New York, NY, USA, v. 24, n. 6, p. 381–395, 1981. 71

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE Computer Society, v. 9, n. 03, p. 90–95, 2007. 77

JACOB, B.; GUENNEBAUD, G. **Eigen library version 3.3.7**. 2018. Disponível em: <https://eigen.tuxfamily.org/>. Acesso em: 01/14/2023. 77

JOHNSON, A. E.; HEBERT, M. Using spin images for efficient object recognition in cluttered 3d scenes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 21, n. 5, p. 433–449, 1999. 70

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: **2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)**. [S.l.: s.n.], 2004. v. 3, p. 2149–2154 vol.3. 22

LEWIS, F. L.; XIE, L.; POPA, D. **Optimal and Robust Estimation: With an Introduction to Stochastic Control Theory**. [S.l.]: CRC press, 2017. 31

LIU, Y.; THRUN, S. Results for outdoor-slam using sparse extended information filters. In: **2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)**. [S.l.: s.n.], 2003. v. 1, p. 1227–1233 vol.1. 53, 55

MARDER-EPPSTEIN, E.; BERGER, E.; FOOTE, T.; GERKEY, B.; KONOLIGE, K. The office marathon: Robust navigation in an indoor office environment. In: **2010 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 2010. p. 300–307. 77

MCLACHLAN, G. J. Mahalanobis distance. **Resonance**, v. 4, n. 6, p. 20–26, 1999. 54

PAULA, R. R. d. *et al.* Método de monte carlo e aplicações. 2014. 86, 89

QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. *et al.* Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA Workshop on Open Source Software**. [S.l.], 2009. v. 3, n. 3.2, p. 5. 22

ROBOTIS. **TurtleBot 3**. 2021. Disponível em: <https://github.com/ROBOTIS-GIT/turtlebot3>. 79

SAEEDI, S.; TRENTINI, M.; SETO, M.; LI, H. Multiple-robot simultaneous localization and mapping: A review. **Journal of Field Robotics**, Wiley Online Library, v. 33, n. 1, p. 3–46, 2016. 18, 21

SAPUTRA, R. P. Implementation 2d ekf-based simultaneous localisation and mapping for mobile robot. **arXiv preprint arXiv:1905.06529**, 2019. 57

SMITH, R. C.; CHEESEMAN, P. On the representation and estimation of spatial uncertainty. **The International Journal of Robotics Research**, Sage Publications Sage CA: Thousand Oaks, CA, v. 5, n. 4, p. 56–68, 1986. 30

SOLÀ, J. **Simulataneous localization and mapping with the extended Kalman filter. A very quick guide... with Matlab code!** 2014. Disponível em: <https://www.iri.upc.edu/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf>. Acesso em: 02/05/2022. 34, 35, 36, 38

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics**. MIT Press, 2005. (Intelligent Robotics and Autonomous Agents series). ISBN 9780262303804. Disponível em: <https://books.google.com.br/books?id=wjM3AgAAQBAJ>. 30, 36, 38, 39, 40, 45, 50, 51, 52, 56, 64, 65, 66, 67, 72, 83, 94, 111, 113

THRUN, S.; KOLLER, D.; GHAHRAMANI, Z.; DURRANT-WHYTE, H.; NG, A. Y. Simultaneous mapping and localization with sparse extended information filters: Theory and initial results. In: **Algorithmic Foundations of Robotics V**. [S.l.]: Springer, 2004. p. 363–380. 39, 51

THRUN, S.; LIU, Y.; KOLLER, D.; NG, A. Y.; GHAHRAMANI, Z.; DURRANT-WHYTE, H. Simultaneous localization and mapping with sparse extended information filters. **The International Journal of Robotics Research**, SAGE Publications, v. 23, n. 7-8, p. 693–716, 2004. 30, 40, 46

WALTER, M. R.; EUSTICE, R. M.; LEONARD, J. J. Exactly sparse extended information filters for feature-based slam. **The International Journal of Robotics Research**, Sage Publications Sage UK: London, England, v. 26, n. 4, p. 335–359, 2007. 41

ZHENG, K. Ros navigation tuning guide. **Robot Operating System (ROS) The Complete Reference (Volume 6)**, Springer, p. 197–226, 2021. 89

Apêndice A - Descrição detalhada de algoritmos

A.1 Algoritmo EKF-SLAM

As três operações principais do EKF-SLAM (predição, atualização e inserção de novas *landmarks*) estão descritas nos Algoritmos A.1, A.2 e A.3, respectivamente. Por fim, o Algoritmo A.4 descreve o EKF-SLAM completo, composto pelas três operações. A correção na orientação, para que ela permaneça dentro do intervalo $[-\pi, \pi]$, é omitida tanto no Algoritmo A.1 quanto no A.2.

Algoritmo A.1 Etapa de predição do EKF-SLAM

1: **function** EKF-SLAM-PREDIÇÃO($\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t$)

2: $\quad \overline{\boldsymbol{\mu}}_t \leftarrow \begin{bmatrix} \boldsymbol{g}_{\mathrm{r}}(\boldsymbol{\mu}_{\mathrm{r},t-1}, \mathbf{u}_t) \\ \boldsymbol{\mu}_{\mathrm{m}} \end{bmatrix}$

3: $\quad \overline{\mathbf{P}}_{RR,t} \leftarrow \mathbf{G}_{R,t}\mathbf{P}_{RR,t-1}\mathbf{G}_{R,t}^T$

4: $\quad \overline{\mathbf{P}}_{RM,t} \leftarrow \mathbf{G}_{R,t}\mathbf{P}_{RM,t-1}$

5: $\quad \overline{\mathbf{P}}_t \leftarrow \begin{bmatrix} \overline{\mathbf{P}}_{RR,t} & \overline{\mathbf{P}}_{RM,t} \\ \overline{\mathbf{P}}_{RM,t}^T & \mathbf{P}_{\mathrm{mm}} \end{bmatrix}$

6: $\quad$ **return** $\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t$

7: **end function**

Algoritmo A.2 Etapa de atualização do EKF-SLAM

function EKF-SLAM-ATUALIZAÇÃO($\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t, \mathbf{y}, j =$ índice da *landmark*)

$\mathbf{z} \leftarrow \mathbf{y} - \boldsymbol{h}^j(\overline{\boldsymbol{\mu}}_t)$

$\mathbf{Z} \leftarrow \begin{bmatrix} \mathbf{H}^j_{\mathrm{r}} & \mathbf{H}^j_{\mathrm{m}} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_{\mathrm{rr}} & \overline{\mathbf{P}}_{\mathrm{rm^j}} \\ \overline{\mathbf{P}}^T_{\mathrm{rm^j}} & \overline{\mathbf{P}}_{\mathrm{m^jm^j}} \end{bmatrix} \begin{bmatrix} \mathbf{H}^j_{\mathrm{r}} \\ \mathbf{H}^j_{\mathrm{m}} \end{bmatrix} + \mathbf{Q}_t$

$\mathbf{K} \leftarrow \begin{bmatrix} \mathbf{P}_{RR,t} & \mathbf{P}_{RM^j,t} \\ \mathbf{P}_{MR,t} & \mathbf{P}_{MM^j,t} \end{bmatrix} \begin{bmatrix} [\mathbf{H}^j_{\mathrm{r}}]^T \\ [\mathbf{H}^j_{\mathrm{m}}]^T \end{bmatrix} \mathbf{Z}^{-1}$

$\boldsymbol{\mu}_t \leftarrow \mathbf{K}\mathbf{z}$

$\mathbf{P}_t \leftarrow \overline{\mathbf{P}}_t - \mathbf{K}\mathbf{Z}\mathbf{K}^T$

return $\boldsymbol{\mu}_t, \mathbf{P}_t$

end function

Algoritmo A.3 Etapa de inserção de nova *landmark* do EKF-SLAM

function EKF-SLAM-INSERÇÃO-NOVA-LANDMARK($\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t, \mathbf{y}, j$)

$\overline{\boldsymbol{\mu}}^*_t \leftarrow \begin{bmatrix} \overline{\boldsymbol{\mu}}_t \\ \boldsymbol{f}(\overline{\boldsymbol{\mu}}_t, \mathbf{y}) \end{bmatrix}$

$\overline{\mathbf{P}}^*_t \leftarrow \begin{bmatrix} \overline{\mathbf{P}}_t & \overline{\mathbf{P}}_t\mathbf{F}_X^T \\ \overline{\mathbf{P}}_t\mathbf{F}_X & \mathbf{F}_X\overline{\mathbf{P}}_t\mathbf{F}_X^T + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T \end{bmatrix}$

return $\overline{\boldsymbol{\mu}}^*_t, \overline{\mathbf{P}}^*_t$

end function

Algoritmo A.4 EKF-SLAM

function EKF-SLAM($\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t, \mathbf{y}^{1:k}, \mathbf{c}^{1:k}$)
  $\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t \leftarrow$ EKF-SLAM-Predição($\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t$)
  $\mathcal{L}^* \leftarrow \{\}$ ▷ Conjunto de novas landmarks
  for ($\mathbf{y}^i \in \mathbf{y}^{1:k}$) **do**
    $j \leftarrow \mathbf{c}^i$
    if landmark j está no mapa $\mathbf{x}_\mathrm{m}$ **then**
      $\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t \leftarrow$ EKF-SLAM-Atualização($\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t, \mathbf{y}^i, j$)
    else
      $\mathcal{L}^* \leftarrow \mathcal{L}^* + \{(j, \mathbf{y}^i)\}$
    end if
  end for
  for $\mathcal{L}^i \in \mathcal{L}^*$ **do**
    $j, \mathbf{y}^i \leftarrow \mathcal{L}^i$
    $\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t \leftarrow$ EKF-SLAM-Inserção-Nova-Landmark($\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t, \mathbf{y}^i, j$)
  end for
  $\boldsymbol{\mu}_t, \mathbf{P}_t \leftarrow \overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t$
  return $\boldsymbol{\mu}_t, \mathbf{P}_t$
end function

A.2 Complexidade da multiplicação ingênua de matrizes

O Algoritmo A.5 realiza o produto "ingênuo", ou seja, sem nenhum tipo de otimização, de duas matrizes de dimensões $m \times n$ e $n \times p$. Como é evidenciado pelos três laços de repetição aninhados, entre as linhas 3-9, a complexidade assintótica do algoritmo é $\mathcal{O}(mnp)$. Cabe ressaltar que, caso as matrizes tenham alguma dimensão conhecida, ou seja, constante, essa dimensão deixa de ser considerada na notação assintótica pois passa a ser uma constante do problema, deixando de ser uma variável de entrada. Porém, isso não quer dizer que essa dimensão não tenha tamanho relevante para a complexidade da computação.

Algoritmo A.5 Produto ingênuo de matrizes

```
1:  function PRODUTO-INGÊNUO(A_{m×n}, B_{n×p})
2:      M_{m×p} ← 0_{m×n}
3:      for i ∈ {0, 1, ..., m} do
4:          for j ∈ {0, 1, ..., p} do
5:              for k ∈ {0, 1, ..., n} do
6:                  M_{i,j} ← M_{i,j} + A_{i,k} · B_{k,j}
7:              end for
8:          end for
9:      end for
10:     return M
11: end function
```

Apêndice B - Desenvolvimento de Resultados

B.1 Matriz de informação aumentada a partir da matriz de covariância

Para derivar a inserção de *landmark* no EIF/SEIF, vamos partir dos resultados da inserção de *landmark* no EKF apresentados na Seção 3.2.3. Por comodidade, repetiremos a Equação 3.21 com o resultado do aumento da matriz de covariância:

$$\mathbf{P}_t^* = \begin{bmatrix} \mathbf{P}_t & \mathbf{P}_t\mathbf{F}_X^T \\ \mathbf{F}_X\mathbf{P}_t & \mathbf{F}_X\mathbf{P}_t\mathbf{F}_X^T + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T \end{bmatrix} \qquad \text{(3.21 repetida)}$$

e aplicando a equivalência $\mathbf{\Omega}^{-1} = \mathbf{P}$, pode-se reescrever a expressão acima como:

$$[\mathbf{\Omega}_t^*]^{-1} = \begin{bmatrix} \mathbf{\Omega}_t^{-1} & \mathbf{\Omega}_t^{-1}\mathbf{F}_X^T \\ \mathbf{F}_X\mathbf{\Omega}_t^{-1} & \mathbf{F}_X\mathbf{\Omega}_t^{-1}\mathbf{F}_X^T + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T \end{bmatrix} \qquad \text{(B.1)}$$

Invertendo ambos os lados:

$$\mathbf{\Omega}_t^* = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{Z} & \mathbf{U} \end{bmatrix}$$

Pelo lema da inversão na forma de blocos (Anexo A.2) temos:

$$\begin{aligned}
\mathbf{U} &= \left[\mathbf{F}_X\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T - \mathbf{F}_X\cancel{\boldsymbol{\Omega}_t^{-1}\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}}\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T\right]^{-1} \\
&= \left[\cancel{\mathbf{F}_X\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T} + \mathbf{F}_Y\mathbf{Q}\mathbf{F}_Y^T - \cancel{\mathbf{F}_X\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T}\right]^{-1} \\
&= \left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}
\end{aligned} \tag{B.2}$$

$$\begin{aligned}
\mathbf{Y} &= -\cancel{\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}}\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T\mathbf{U} \\
&= \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)
\end{aligned} \tag{B.3}$$

$$\begin{aligned}
\mathbf{Z} &= -\mathbf{U}\mathbf{F}_X\cancel{\boldsymbol{\Omega}_t^{-1}\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}} \\
&= -\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X
\end{aligned} \tag{B.4}$$

$$\begin{aligned}
\mathbf{X} &= \left[\boldsymbol{\Omega}_t^{-1}\right]^{-1} + \cancel{\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}}\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T\,\mathbf{U}\,\mathbf{F}_X\cancel{\boldsymbol{\Omega}_t^{-1}\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}} \\
&= \boldsymbol{\Omega}_t + \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X
\end{aligned} \tag{B.5}$$

Por fim, temos que a matriz de informação aumentada é dada por:

$$\boldsymbol{\Omega}_t^* = \begin{bmatrix}
\boldsymbol{\Omega}_t + \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X & -\mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right) \\
-\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X & \left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}
\end{bmatrix} \tag{B.6}$$

Apêndice C - Dados experimentais

C.1 Resultados de mapeamento bem-sucedido com agente único

As tabelas C.1 e C.2 registram a posição inicial, tempo de mapeamento de 90% da área total do ambiente representado na Figura 6.1, o erro integral absoluto (IAE) da posição e da orientação para cada uma das simulações nas quais o robô conseguiu cumprir a tarefa com sucesso.

TABELA C.1 – Resultados de mapeamento de cenário com um único agente

# Realização	Posição inicial (m)	Tempo mapeamento 90% da área (s)	IAE de posição (m)	IAE de orientação (rad)
1	(2.7, -1.4)	397	38.7	53.2
2	(1.1, 2.9)	456	102.7	279.7
3	(3.0, -4.1)	482	82.6	105
4	(1.3, 3.1)	371	42.4	39.1
5	(-2.0, 2.1)	345	114.2	182.5
6	(1.0, -1.1)	435	94.5	180
7	(-0.9, 4.1)	410	85.7	122.1
8	(-3.1, 3.6)	445	112.5	290
9	(3.9, -1.0)	441	123.3	230
10	(-3.3, 0.6)	498	148.6	273.8
11	(-4.0, -0.9)	353	118	138.5
12	(0.9, 2.9)	370	79.8	112.6
13	(1.6, 2.7)	546	158.2	182.3
14	(-4.5, -1.1)	399	53.3	52.5
15	(3.2, -0.1)	412	72.7	114.1
16	(0.2, 1.3)	493	148	244.6
17	(-3.1, -0.1)	376	50.9	52.2
18	(-3.2, 3.9)	355	93.8	109.9

TABELA C.2 – Resultados de mapeamento de cenário com um único agente (Continuação)

# Realização	Posição inicial (m)	Tempo mapeamento 90% da área (s)	IAE de posição (m)	IAE de orientação (rad)
19	(4.4, 1.3)	486	213.6	252.8
20	(2.8, 0.2)	410	29.9	32.9
21	(3.1, 4.3)	357	16.5	15.6
22	(0.0, 1.7)	322	24.3	26.5
23	(3.8, 1.2)	485	25.6	22.2
24	(3.2, -0.2)	415	28.3	27
25	(-1.9, -0.9)	408	32.2	23.6
26	(0.1, 1.0)	396	16.8	16.3
27	(-1.3, 4.3)	469	25.2	21.5
28	(-2.5, 4.0)	440	246.3	109.3
29	(-2.4, 4.5)	497	177.7	221.9
30	(-1.5, 4.1)	482	204.9	188.8
31	(3.7, 1.5)	431	27.8	24.1
32	(4.5, 3.0)	387	48	13.1
33	(4.4, 1.6)	544	56.9	60.6
34	(3.6, -1.3)	302	29.5	18.8
35	(1.4, 3.0)	470	51.2	24.4
36	(-1.8, -1.2)	474	25.4	27.3
37	(4.2, 1.4)	443	251.5	63.7
38	(3.8, 0.8)	405	15.2	17.5
39	(-2.8, 4.4)	494	61.9	29
40	(3.3, -1.1)	416	16.6	23.6
41	(2.9, -1.4)	330	41.5	31.8
42	(-2.0, 3.9)	471	207.8	57.2
43	(2.7, -4.5)	482	69.1	56.1
44	(-2.9, 2.9)	484	238.3	89.7
45	(2.8, -1.7)	434	40.8	37.6
46	(-0.3, -3.2)	427	45.7	25.5
47	(0.8, 0.8)	551	31.6	21.7
48	(-3.9, -3.1)	351	13.8	19.7
49	(4.5, 1.3)	403	70.3	20.6
50	(-2.7, 3.8)	387	25.8	13
51	(-1.9, -1.1)	371	17.7	16.6
52	(1.3, 0.9)	403	21.5	24.7
53	(-3.3, -2.5)	531	31.4	22.8
54	(1.4, 2.7)	392	38.1	26.5
55	(3.2, -1.6)	401	26.3	15.2
56	(-2.8, 4.5)	413	21.7	21.3
57	(1.0, -1.0)	410	27.2	16.9
58	(1.0, -4.4)	437	26.4	33.3
59	(-2.9, -2.3)	535	51.8	46.8
60	(0.3, 0.8)	416	27.9	22.1

TABELA C.3 – Posição inicial e tempo de mapeamento para o cenário com dois agentes

Realização	Posição inicial (m)		Tempo mapeamento de 90% da área (s)	
	Robô 1	Robô 2	Robô 1	Robô 2
1	(-1.7, 3.9)	(2.2, 1.9)	346	373
2	(-4.2, -1.1)	(1.9, 1.5)	409	409
3	(3.3, -1.0)	(-2.9, -2.1)	346	334
4	(-1.8, 4.0)	(-2.8, 4.1)	317	373
5	(-1.8, 4.3)	(-1.1, 0.0)	284	300
6	(-4.5, -2.8)	(-1.6, 4.2)	289	345
7	(2.9, -1.7)	(-1.7, 3.9)	281	244
8	(1.0, 3.2)	(3.0, -4.2)	338	365
9	(-2.5, 4.2)	(4.5, 1.3)	252	253
10	(3.1, -1.5)	(2.1, 1.7)	312	272
12	(3.7, -0.8)	(0.9, -4.5)	280	285
13	(0.9, -1.2)	(1.5, 0.9)	354	280
14	(0.4, 1.2)	(3.4, -0.6)	276	273
15	(1.9, -2.9)	(-2.9, -2.1)	220	221
16	(-0.8, 3.7)	(-2.7, 3.5)	279	290
17	(1.9, -3.2)	(-1.2, -0.1)	307	324
18	(-4.3, -3.0)	(-0.1, 0.9)	237	237
19	(-1.0, 4.5)	(-3.0, 0.3)	262	262
20	(2.0, 2.5)	(-2.6, 3.8)	316	339

C.2 Resultados de mapeamento bem-sucedido com dois agentes

As Tabelas C.3 e C.4 registram as posições iniciais e os tempos de mapeamento de 90% da área do ambiente representado na Figura 6.1 para o cenário com dois agentes com pose inicial conhecida. Enquanto as Tabelas C.5 e C.6 apresentam os dados de erro absoluto integral da posição e da orientação.

[1] *outlier*

TABELA C.4 – Posição inicial e tempo de mapeamento para o cenário com dois agentes (continuação)

Realização	Posição inicial (m)		Tempo mapeamento de 90% da área (s)	
	Robô 1	Robô 2	Robô 1	Robô 2
21	(0.0, 0.8)	(-2.7, 3.7)	375	399
22	(-3.2, 3.9)	(1.1, 0.9)	447	452
23	(3.2, -1.9)	(-3.0, 3.0)	394	253
24	(4.1, 1.5)	(1.8, -3.2)	310	311
25	(3.2, -3.8)	(1.1, -4.0)	473	362
26	(-1.3, 4.5)	(-0.4, 0.2)	276	255
27	(-0.8, 0.0)	(-4.0, 2.1)	244	242
28	(1.7, 2.6)	(0.9, 3.5)	349	407
29	(2.1, 2.0)	(1.1, 3.8)	336	319
30	(1.8, -2.9)	(1.1, 3.4)	311	280
31	(1.1, -4.2)	(2.2, 1.7)	358	378
32	(-3.9, -3.0)	(2.9, -2.2)	347	475
33	(1.9, -2.8)	(2.9, -1.1)	287	295
34	(1.1, 1.2)	(1.9, 1.6)	308	298
35	(-0.1, 0.7)	(-1.6, 4.2)	217	246
36	(2.8, 4.1)	(3.0, -1.2)	317	407
37	(-3.2, -2.0)	(-0.1, 0.5)	463	318
38	(3.9, -0.9)	(1.5, 3.3)	316	296
39	(1.1, -4.4)	(3.0, -1.3)	422	385
40	(2.0, 1.1)	(-0.7, -0.1)	296	295
41	(0.0, 1.7)	(-0.3, 0.0)	283	284
42	(0.6, 1.0)	(2.9, -4.4)	354	358
43	(-0.4, -0.1)	(-2.9, 3.1)	316	317
44	(-3.2, 3.3)	(-4.5, 2.2)	474	429
45	(2.8, -1.6)	(2.9, -4.1)	295	316
46	(4.5, 3.1)	(-1.5, 4.5)	250	250
47[1]	(1.9, -3.0)	(-2.1, 4.1)	962	962

TABELA C.5 – Erro absoluto integral da posição e da orientação para o cenário com dois agentes

Realização	IAE de posição (m)		IAE de orientação (rad)	
	Robô 1	Robô 2	Robô 1	Robô 2
1	34.4	37.6	65.8	22.1
2	54.8	42.6	77.1	66
3	43.4	42.9	62.5	46.5
4	114.4	115.5	76.8	73.4
5	36.6	24	47.9	34.3
6	49.5	56.8	104.3	165
7	35	42.9	67.6	42
8	37.3	36.1	41.3	43.6
9	25.7	7.1	8.9	7.3
10	62.4	48.4	29.1	76.5
12	36.7	26.9	15.8	33.2
13	nan	206.1	38.4	211.6
14	8.3	14.6	8	8.8
15	27.6	16.2	47.2	15.9
16	53.9	62.1	47.8	75.7
17	95.3	19.6	91	26
18	14.3	72.3	16.5	77.9
19	22.8	16.7	18.6	23.9
20	32.3	52.3	38.9	40.2

TABELA C.6 – Erro absoluto integral da posição e da orientação para o cenário com dois agentes (continuação)

Realização	IAE de posição (m)		IAE de orientação (rad)	
	Robô 1	Robô 2	Robô 1	Robô 2
21	22.4	15	17.7	16.5
22	72.4	43.4	36.6	25.2
23	43.5	45.4	94.8	74.3
24	183	241	144.9	157
25	65.3	31	42.1	51.6
26	61.1	21.5	47.5	19.5
27	27.2	27	54	21.8
28	15.3	15.2	12.7	13.3
29	381.8	86.6	128.9	42.4
30	49.6	14.9	14.3	11.8
31	15.3	12.9	17.1	12
32	97.3	56.2	243.4	38.9
33	50.7	43	101.1	72.6
34	21.8	13	13.3	15.7
35	41.8	24.1	44.5	40.6
36	91.4	147	240.8	190.4
37	69.7	27.9	30.3	37.6
38	26.5	58.8	63	50.2
39	8.6	9.2	9.7	15
40	9.2	11.9	7.8	10.4
41	30	35.5	65.8	44.8
42	15.8	17.2	16.6	16.3
43	50.1	32.5	104.2	68.2
44	77.2	218.5	44.1	48.6
45	36.9	74.9	51.4	44.9
46	30.8	17.1	8.6	7.1
47[1]	149	909.8	127.6	412.8

Anexo A - Matrizes

A.1 Lema da Inversão. Fórmula de Sherman/Morrison

A fórmula de Sherman/Morrison, também conhecida como lema da inversão especializado, é definido a seguir, retirado de (THRUN *et al.*, 2005, p. 50).

Lemma 1. *Para qualquer matrizez quadradas invertíveis* $\mathbf{R}$ *e* $\mathbf{Q}$ *e qualquer matriz* $\mathbf{P}$ *com dimensões apropriadas, o seguinte é verdadeiro:*

$$\left(\mathbf{R}+\mathbf{P}\mathbf{Q}\mathbf{P}^{T}\right)^{-\mathbf{1}}=\mathbf{R}^{-\mathbf{1}}-\mathbf{R}^{-\mathbf{1}}\mathbf{P}\left(\mathbf{Q}^{-\mathbf{1}}+\mathbf{P}^{T}\mathbf{R}^{-\mathbf{1}}\mathbf{P}\right)^{-\mathbf{1}}\mathbf{P}^{T}\mathbf{R}^{-\mathbf{1}} \tag{A.1}$$

assumindo que todas as matrizes acima podem ser invertidas como definido na premissa.

Demonstração. Defina $\mathbf{\Psi}=\left(\mathbf{Q}^{-\mathbf{1}}+\mathbf{P}^{T}\mathbf{R}^{-\mathbf{1}}\mathbf{P}\right)^{-\mathbf{1}}$. É suficiente mostrar que:

$$\left(\mathbf{R}^{-\mathbf{1}}-\mathbf{R}^{-\mathbf{1}}\mathbf{P}\mathbf{\Psi}\mathbf{P}^{T}\mathbf{R}^{-\mathbf{1}}\right)\left(\mathbf{R}+\mathbf{P}\mathbf{Q}\mathbf{P}^{T}\right)=\mathbf{I}$$

Isso é mostrado através de uma série de manipulações:

$$
\begin{aligned}
&= \mathbf{R^{-1}R} + \mathbf{R^{-1}PQP^T} - \mathbf{R^{-1}P\Psi P^T R^{-1}R} - \mathbf{R^{-1}P\Psi P^T R^{-1}PQP^T} \\
&= \mathbf{I} + \mathbf{R^{-1}PQP^T} - \mathbf{R^{-1}P\Psi P^T I} - \mathbf{R^{-1}P\Psi P^T R^{-1}PQP^T} \\
&= \mathbf{I} + \mathbf{R^{-1}PQP^T} - \mathbf{R^{-1}P\Psi P^T} - \mathbf{R^{-1}P\Psi P^T R^{-1}PQP^T} \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{\Psi P^T} - \mathbf{\Psi P^T R^{-1}PQP^T}\right] \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{\Psi Q^{-1}QP^T} - \mathbf{\Psi P^T R^{-1}PQP^T}\right] \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{\Psi Q^{-1}QP^T} - \mathbf{\Psi P^T R^{-1}PQP^T}\right] \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{\Psi}\left(\mathbf{Q^{-1}} - \mathbf{P^T R^{-1}P}\right)\mathbf{QP^T}\right] \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{\Psi}\left(\mathbf{Q^{-1}} - \mathbf{P^T R^{-1}P}\right)\mathbf{QP^T}\right] \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{\Psi\Psi^{-1}QP^T}\right] \\
&= \mathbf{I} + \mathbf{R^{-1}P}\left[\mathbf{QP^T} - \mathbf{QP^T}\right] \\
&= \mathbf{I}
\end{aligned}
\tag{A.2}
$$

□

A.2 Inversão na forma de blocos

Seja a matriz $(m+n)\times(m+n)$, $\mathbf{M}$, particionada na seguinte forma de blocos:

$$
\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tag{A.3}
$$

onde as matrizes $\mathbf{A}_{m\times m}$ e $\mathbf{D}_{n\times n}$ são invertíveis, então temos que $\mathbf{M}^{-1}$ é dado por:

$$
\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}\right)^{-1}\mathbf{CA}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}\right)^{-1} \\ -\left(\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}\right)^{-1}\mathbf{CA}^{-1} & \left(\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}\right)^{-1} \end{bmatrix} \tag{A.4}
$$

Anexo B - Manipulações da distribuição de probabilidade gaussiana multivariada na forma canônica

Os resultados exibidos nas próximas Seções foram retirados de (THRUN *et al.*, 2005, p. 358-359), onde são demonstrados. Seja a distribuição de probabilidade $p(\mathbf{x}, \mathbf{y})$ sobre os vetores aleatórios $\mathbf{X}$ e $\mathbf{Y}$ uma gaussiana representada na forma canônica (informação).

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\Omega}_{\mathrm{xx}} & \boldsymbol{\Omega}_{\mathrm{xy}} \\ \boldsymbol{\Omega}_{\mathrm{yx}} & \boldsymbol{\Omega}_{\mathrm{yy}} \end{bmatrix} \tag{B.1}$$

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_{\mathrm{x}} \\ \boldsymbol{\xi}_{\mathrm{y}} \end{bmatrix} \tag{B.2}$$

B.1 Marginalização

Se $\boldsymbol{\Omega}_{\mathrm{yy}}$ é invertível, a marginal $p(\mathbf{x})$ é também uma gaussiana representada por:

$$\overline{\boldsymbol{\Omega}}_{\mathrm{xx}} = \boldsymbol{\Omega}_{\mathrm{xx}} - \boldsymbol{\Omega}_{\mathrm{xy}} \boldsymbol{\Omega}_{\mathrm{yy}}^{-1} \boldsymbol{\Omega}_{\mathrm{yx}} \tag{B.3}$$

$$\overline{\boldsymbol{\xi}}_{\mathrm{x}} = \boldsymbol{\xi}_{\mathrm{x}} - \boldsymbol{\Omega}_{\mathrm{xy}} \boldsymbol{\Omega}_{\mathrm{yy}}^{-1} \boldsymbol{\xi}_{\mathrm{y}} \tag{B.4}$$

B.2 Condicionamento

A condicional $p(\mathbf{x} \,|\, \mathbf{y})$ também é uma gaussiana representada por:

$$\overline{\boldsymbol{\Omega}}_{\mathrm{x}\,|\,\mathrm{y}} = \boldsymbol{\Omega}_{\mathrm{xx}} \tag{B.5}$$

$$\overline{\boldsymbol{\xi}}_{\mathrm{x}\,|\,\mathrm{y}} = \boldsymbol{\xi}_{\mathrm{x}} - \boldsymbol{\Omega}_{\mathrm{xy}}\, y \tag{B.6}$$

Anexo C - Algoritmos

C.1 Algoritmo de ajuste de círculos

O Algoritmo C.1 é uma adaptação das notas de aula do professor Matthew L. Elwin da disciplina ME495 (*Sensing, Navigation, and Machine Learning for Robotics*) (ELWIN, 2021) ministrada na Universidade Northwestern.

Algoritmo C.1

function AJUSTE-CÍRCULO($x_{0:n}, y_{0:n}$)
  if $n+1 < 4$ **then**
    return $\{\emptyset\}$
  end if
  $\hat{x} \leftarrow \frac{1}{n+1}\sum x_i$
  $\hat{y} \leftarrow \frac{1}{n+1}\sum y_i$
  $z_{1:n} \leftarrow \{\emptyset\}$
  for x_i, y_i **do**
    $x_i \leftarrow x_i - \hat{x}$
    $y_i \leftarrow y_i - \hat{y}$
    $z_i \leftarrow x_i^2 + y_i^2$
  end for
  $\mathbf{Z} \leftarrow \begin{bmatrix} z_0 & x_0 & y_0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ z_{n+1} & x_{n+1} & y_{n+1} & 1 \end{bmatrix}$
  $\mathbf{U\Sigma V}^T \leftarrow \text{SVD}(\mathbf{Z})$
  if $\sigma_{44} \leq 10^{-4}$ **then**
    $\mathbf{a} \leftarrow \begin{bmatrix} \sigma_{14} & \sigma_{24} & \sigma_{34} & \sigma_{44} \end{bmatrix}^T$
  else
    $\hat{z} \leftarrow \frac{1}{n+1}\sum z_i$
    $\mathbf{H}^{-1} \leftarrow \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & -2\hat{z} \end{bmatrix}$
    $\mathbf{Y} \leftarrow \mathbf{V\Sigma V}^T$
    Encontre os autovetores e autovalores de $\mathbf{Q} \leftarrow \mathbf{YH}^{-1}\mathbf{Y}$
    Seja $\mathbf{a}_*$ o autovetor correspondente ao menor autovalor de $\mathbf{Q}$
    $\mathbf{a} \leftarrow \mathbf{Y}^{-1}\mathbf{a}_*$
  end if
  $c_x \leftarrow -\frac{a_2}{2a_1} + \hat{x}$
  $c_y \leftarrow -\frac{a_3}{2a_1} + \hat{y}$
  $r^2 \leftarrow \frac{a_2^2 + a_3^2 - 4a_1a_4}{4a_1^2}$
  $\sigma^2 \leftarrow \frac{1}{n+1}\sum\left((x_i - c_x)^2 + (y_i - c_y)^2 - r^2\right)^2$
  return $(c_x, c_y), \sqrt{r^2}, \sqrt{\sigma^2}$
end function

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO Nº	4. Nº DE PÁGINAS
DM	26 de janeiro de 2023	DCTA/ITA/DM-167/2022	115

5. TÍTULO E SUBTÍTULO:

SLAM distribuído envolvendo navegação, guiamento e fusão sensorial para reconstrução 2D

6. AUTOR(ES):

Wellington Vieira Martins de Castro

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

SLAM; Mapeamento; Localização; Robótica Móvel; Filtro de Informação

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Robôs; Dinâmica de robôs; Localização e mapeamento simultâneo; Sensores; Robótica; Computação; Controle.

10. APRESENTAÇÃO: (**X**) **Nacional** ( ) **Internacional**

ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Sistemas e Controle. Orientador: Prof. Dr. Jacques Waldmann. Defesa em 09/12/2022. Publicada em 2022.

11. RESUMO:

O problema de Localização e Mapeamento Simultâneos, conhecido pela sigla SLAM, pergunta se é possível para um robô ser colocado em um ambiente desconhecido a priori, e incrementalmente construir um mapa deste ambiente enquanto simultaneamente se localiza neste mapa sem a necessidade de infraestrutura de localização como GPS. A solução do problema SLAM é fundamental para a robótica móvel autônoma. Entretanto, apesar de já solucionado, não é uma tarefa trivial tanto do ponto de vista teórico como do ponto de vista da implementação. Dependendo da dinâmica do robô, sensores utilizados, recurso computacional disponível, necessidade de navegação e guiamento, a solução pode se tornar mais ou menos complexa. Este trabalho investiga uma solução multiagente distribuída e descentralizada em ambiente simulado para o problema SLAM 2D. Para isso emprega o uso do Filtro de Informação Estendido Esparso, juntamente com outros algoritmos de navegação, associação de dados e de representação de mapas. As vantagens da solução distribuída do problema SLAM, em relação ao problema original, são a divisão da carga de trabalho entre os agentes e a redundância de informação. O objetivo deste trabalho é simular um grupo de robôs que mapeiem o ambiente onde estão inseridos, de maneira ativa e descentralizada, considerando restrições de memória e processamento. Portanto, além de produzir algoritmos que capacitem os robôs a resolverem o problema SLAM Ativo Descentralizado e Distribuído, é preciso criar uma infraestrutura de software na qual o ambiente e os agentes serão simulados. Para isso utilizou-se o Sistema Operacional de Robô, ROS do inglês *Robot Operating System*, que provê um conjunto de bibliotecas e ferramentas pertinentes ao desenvolvimento em robótica, além de uma camada de comunicação utilizada pelos diferentes módulos do sistema (mapeamento, navegação, visão) para trocarem informações. Para a simulação do ambiente, sensores e agentes, utilizou-se o simulador Gazebo. A solução desenvolvida apresenta uso de memória significativamente reduzido, mais de 50%, em relação à abordagem clássica EKF-SLAM. Além disso, apesar de não possuir coordenação entre os agentes, observaram-se ganhos no tempo de exploração no cenário com dois agentes em comparação com um único agente.

12. GRAU DE SIGILO:

(**X**) **OSTENSIVO** ( ) **RESERVADO** ( ) **SECRETO**