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.
75f.

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. Cupim. 2. Dilema. 3. Construção. 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. 75f. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

Wellington Vieira Martins de Castro
Av. Engenheiro Francisco José Longo, 633. Apartamento 113
12.245-906 – 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.	Alan Turing	Presidente	-	ITA
Prof. Dr.	Jacques Waldmann	Orientador	-	ITA
Prof. Dr.	Linus Torwald		-	UXXX
Prof. Dr.	Richard Stallman		-	UYYY
Prof. Dr.	Donald Duck		-	DYSNEY
Prof. Dr.	Mickey Mouse		-	DISNEY

ITA

Aos amigos da Graduação e Pós-Graduação do ITA.

Agradecimentos

"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 de 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 desenvolve uma solução multiagente em ambiente simulado para o problema SLAM 2D. Para isso emprega o uso do Filtro de Informação Esparso, juntamente com outros algoritmos de navegação, associação de dados e de representação de mapas. As vantagens da solução distribuída do problema de SLAM, em relação ao problema original, é a divisão da carga de trabalho entre os agentes e a redundância de informação.

Atualmente cada agente é capaz de performar SLAM individualmente, o desenvolvimento se encontra na etapa de troca do mapa de *features* entre os agentes. Subsequente a essa etapa, será abordado o problema da exploração conjunta entre os agentes. Terminadas essas duas etapas de desenvolvimento, fechando o escopo delineado para o trabalho, se dará início à escrita da dissertação e também de artigo para submissão a coreferências e/ou periódicos.

Abstract

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CML	Concurrent Mapping and Localization
GPS	Global Positioning System
SLAM	Simultaneous Localization and Mapping
ROS	Robot Operating System
KF	Filtro de Kalman
EKF	Filtro de Kalman Extendido
EIF	Filtro de Informação Extendido
SEIF	Filtro de Informação Extendido Esparço
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging

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*
$p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0^{\mathrm{r}}\right)$	Distribuição do problema online SLAM
$p\left(\mathbf{x}_{0:t}^{\mathrm{r}}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0^{\mathrm{r}}\right)$	Distribuição do problema full SLAM
$\boldsymbol{\mu}_t$	Vetor média no instante t
$\overline{\boldsymbol{\mu}}_t$	Predição do vetor média 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
$\boldsymbol{g}_{\mathrm{r}}(\bullet, \bullet)$	Modelo de movimento do robô
$\boldsymbol{h}(\bullet, \bullet)$	Modelo de medida do sensor laser
$\boldsymbol{f}(\bullet, \bullet)$	Modelo de medida inverso do sensor laser
$\mathbb{E}\left[\bullet\right]$	Operador esperança
$\mathbf{0}_{m \times n}$	Matriz nula de dimensão $m \times n$
$\mathbf{I}_n$	Matriz identidade de dimensão n
$Cov\left(\bullet\right)$	Covariância
$\mathbf{M}$	Matriz de projeção

Sumário

1 Introdução

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*) 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 em (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 de 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 dotado de um sensor extrínseco, capaz de capturar medidas relacionadas ao ambiente e, um sensor intrínseco capaz de medir os comandos de controle executados, se deslocando. 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

- $\mathbf{z}_t$: o vetor de medidas no instante t
- $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{t-1}, \mathbf{z}_t\}$: 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:

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

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

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 terminologia utilizada, pelos pesquisadores, para cada uma dessas características. Parte da terminologia do problema de 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 resolve-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 contra partida, 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, medidas $\mathbf{z}$ 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 comumente denominados *Range* e *Bearing*. A componente *Range* é a distância do sensor até o objeto medido. Enquanto *Bearing* é a posição 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 o *Bearing*.

Então, de acordo com a presença/ausência dessas componentes os termos: *Range Only*, *Bearing Only* e *Range-Bearing* 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 posição angular é lida. E em *Range-Bearing* ambas as quantidades são lidas, em sistemas Range-Bearing SLAM são comumente utilizados sensores do tipo *LIDAR* (*Light Detection and Ranging*), que retornam uma nuvem de pontos onde cada ponto é descrito pela distância e posição angular em relação ao sensor.

Com um algoritmo capaz de estimar 1.1 ou 1.2 e sensores apropriados para alimenta-lo, um robô um robô é capaz de performar SLAM como foi apresentado até agora. Porém, ao mapear o ambiente, o robô pode explora-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. Quando a solução para o problema de SLAM também incorpora a geração de trajetórias, para que a exploração seja feita de forma autônoma

(ativa), é denominada SLAM Ativo.

Até o momento, o problema de SLAM foi tratado 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 único ponto de falha catastrófica, o nó central. Nessa arquitetura é geralmente mais simples manter consistência e consenso sobre o mapa global.

Em contra partida, 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 criar um sistema que consiste em um grupo de robôs capazes de mapear o ambiente onde estão inseridos, sem nenhuma infraestrutura de localização como GPS, de maneira ativa e descentralizada, preocupando-se com restrições de memória e processamento em ambiente simulado.

Portanto, além de produzir algoritmos que capacitem os robôs a resolverem o problema de 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. Pois ele provê um conjunto de bibliotecas e ferramentas pertinentes ao cenário de desenvolvimento em robótica, além de uma camada de comunicação comum utilizada pelos diferentes módulos do sistema (mapeamento, navegação, visão) trocarem informações.

Dessa forma, ao utilizar o ROS este trabalho se torna 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 simulador Gazebo (KOENIG; HOWARD, 2004)

tocar no assunto que o gazebo realiza simulacoes fidedignas de sensores, mass, friction, and numerous other physics variables dizer que ele oferece um controle muito grande sobre quase todos os aspectos da simulacao desde condicoes de luz até coeficientes de atrito textura, transparencia e cor

1.2 Motivação

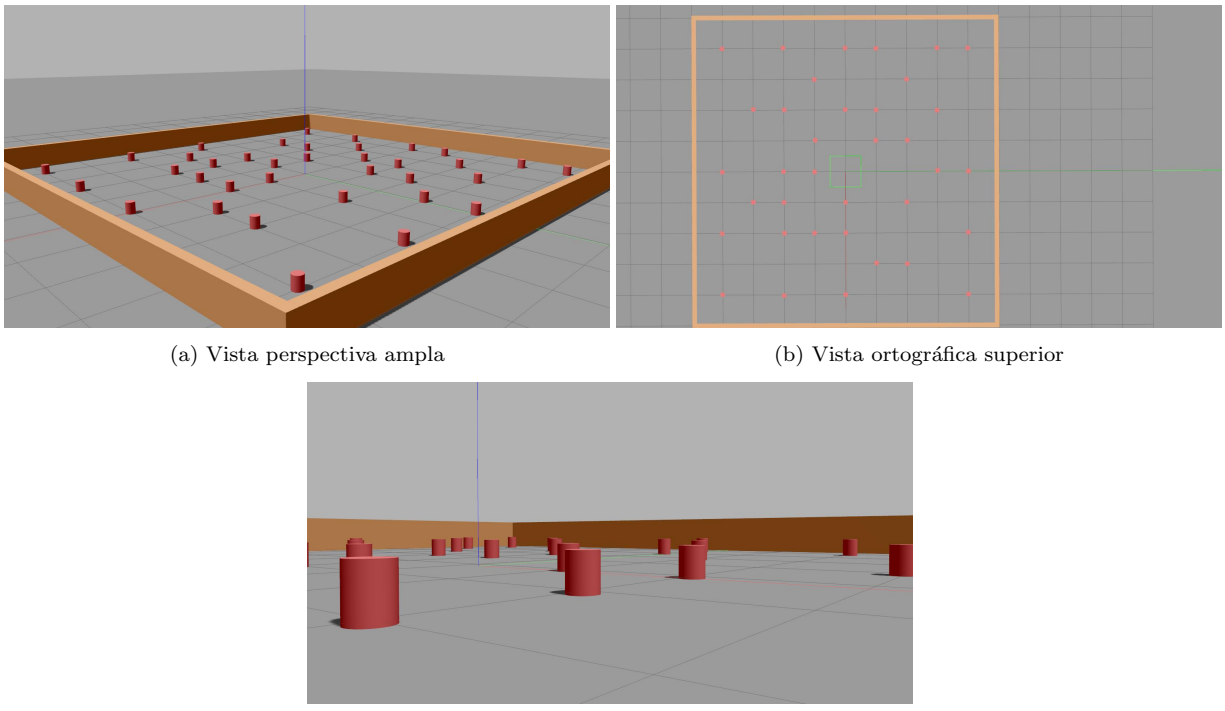
1.3 Estrutura de um sistema SLAM

1.4 Organização do trabalho

2 Visão Geral do Sistema

2.1 O ambiente

O ambiente consiste em uma espaço de $10m^2$ com diversos "postes" de formato cilíndrico de $16cm$ de diâmetro, delimitado por paredes, e está representado na Figura 2.1.



(a) Vista perspectiva ampla

(b) Vista ortográfica superior

(c) Vista perspectiva fechada

FIGURA 2.1 – Diferentes vistas do ambiente simulado

2.2 O robô

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

FIGURA 2.2 – Gêmeo digital do robô *Turtlebot 3*. Os circuitos eletrônicos do robô real não estão representados. O sensor LiDAR encontra-se no topo do robô.

2.2.1 Modelo do robô

Na Figura 2.3 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:

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

Onde 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 origem do sistema de coordenadas do robô, no sistema estático.

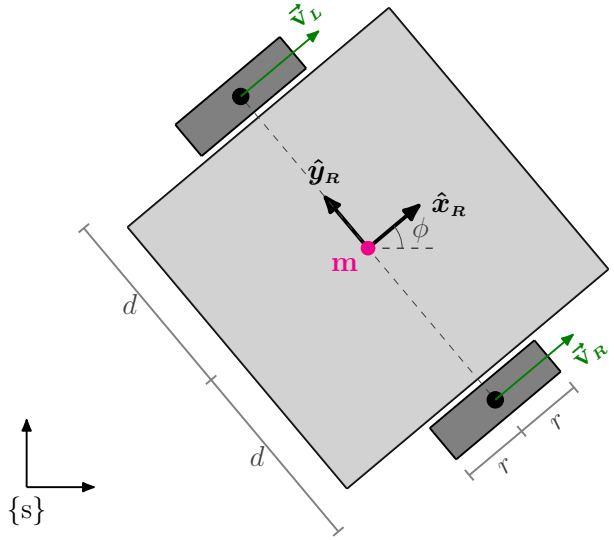


FIGURA 2.3 – Visão superior do esquemático de um robô diferencial. O eixo z está apontando para fora da folha. O sistema de coordenadas móvel do robô está posicionado no ponto médio do eixo das rodas (cinza escuro). Na imagem $\{s\}$ é um sistema de coordenadas estático.

O modelo de movimento do robô utilizado neste trabalho, 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 $]t-1, t]$.

$$\mathbf{x}_t = \boldsymbol{g}_{\mathrm{r}}(\mathbf{x}_{t-1}, \mathbf{u}_t)$$

$$= \begin{bmatrix} \phi_{t-1} \\ x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{cases} \begin{bmatrix} \alpha_t \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L} \left(\sin(\phi_{t-1} + \alpha_t) - \sin(\phi_{t-1})\right) \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L} \left(-\cos(\phi_{t-1} + \alpha_t) + \cos(\phi_{t-1})\right) \end{bmatrix}, \text{Se } u_t^L \neq u_t^R \\ \begin{bmatrix} 0 \\ r \cdot u_t^R \cos(\phi_{t-1}) \\ r \cdot u_t^R \sin(\phi_{t-1}) \end{bmatrix}, \text{Caso contrário} \end{cases} \tag{2.2}$$

Onde:

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

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 $\overline{\mathbf{x}}_{t-1}$, que será utilizada para linearizá-lo em técnicas como EKF e SEIF-SLAM, apresentadas mais adiante. O jacobiano do modelo $\boldsymbol{g}_{\mathrm{r}}(\bullet, \bullet)$ está na Equação 2.4.

$$\mathbf{G}_R = \mathbf{I}_3 + \begin{cases} \begin{bmatrix} 0 & 0 & 0 \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L} \left(\cos(\overline{\phi}_{t-1} + \overline{\alpha}_{t-1}) - \cos(\overline{\phi}_{t-1})\right) & 0 & 0 \\ d \cdot \dfrac{u_t^R + u_t^L}{u_t^R - u_t^L} \left(\sin(\overline{\phi}_{t-1} + \overline{\alpha}_{t-1}) - \sin(\overline{\phi}_{t-1})\right) & 0 & 0 \end{bmatrix}, \text{Se } u_t^L \neq u_t^R \\ \begin{bmatrix} 0 & 0 & 0 \\ -r \cdot u_t^R \sin(\overline{\phi}_{t-1}) & 0 & 0 \\ r \cdot u_t^R \cos(\overline{\phi}_{t-1}) & 0 & 0 \end{bmatrix}, \text{Caso contrário} \end{cases} \tag{2.4}$$

2.3 Medidas e o modelo de medida *Range-Bearing*

A medida gerada pelo sensor LiDAR, embarcado no TurtleBot, consiste em uma nuvem de pontos planar. Essa nuvem de pontos é processada e dela são extraídas estimativas dos centros dos cilindros presentes no ambiente. 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.

2.3.1 Modelo de medida *Range-Bearing*

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 2.4, 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

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

onde

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

Como o modelo de medida em 2.6 é não linear, é necessário lineariza-lo para utilizá-lo em soluções como EKF-SLAM, e seus derivados como SEIF-SLAM. Sua matriz jacobiana, $\mathbf{H}$, para a j-ésima *landmark* é 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.

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

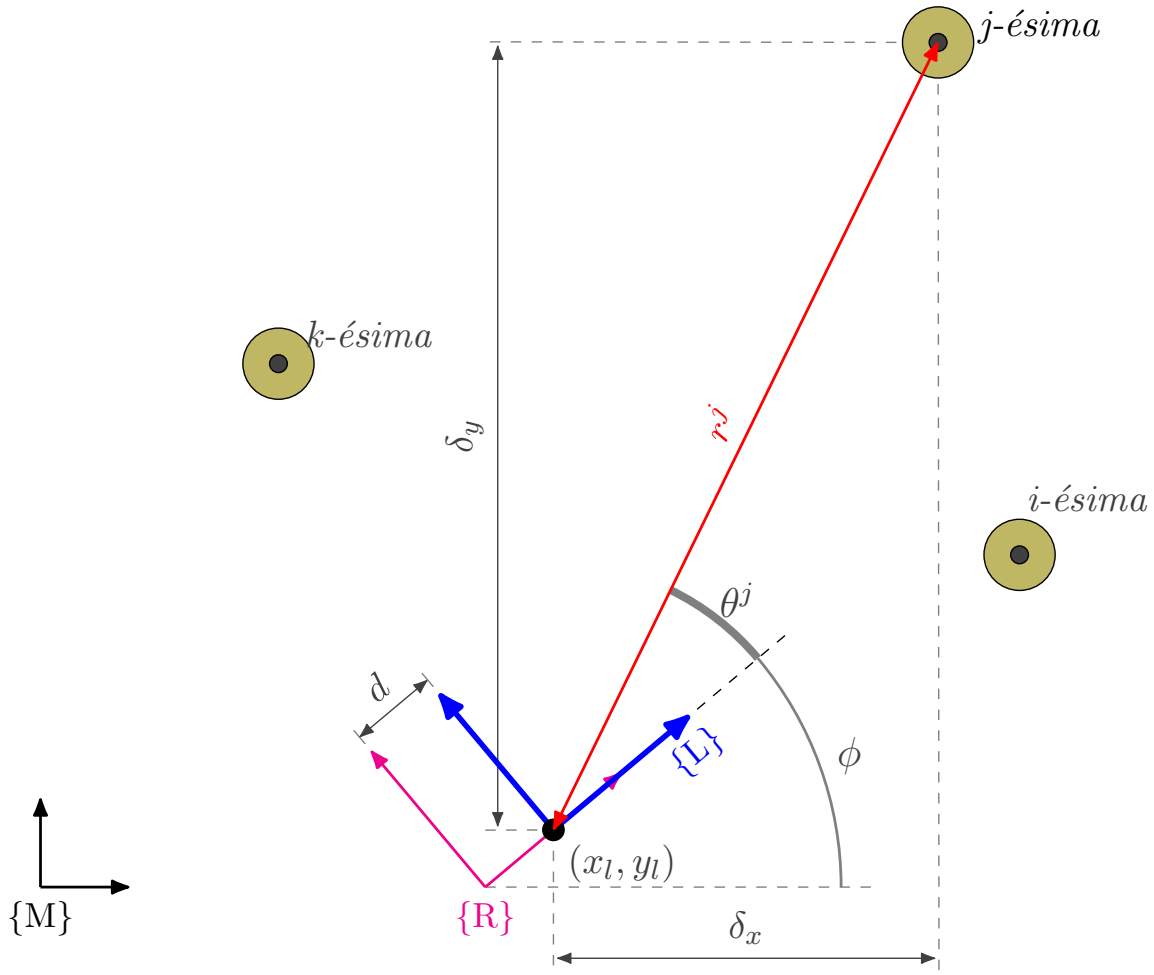


FIGURA 2.4 – Esquemático do modelo de medida *Range-Bearing*. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. A medida (r^j, θ^j) se refere à i-ésima *landmark* no mapa.

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

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

2.3.2 Modelo de medida inverso *Range-Bearing*

O modelo de medida descrito na Seção anterior é também conhecido como modelo de medida direto, ele calcula a medida que espera-se ler quando o sistema está em um dado estado. Mas também há o modelo de medida inverso, que calcula um 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 no vetor de estados, na Figura 2.5 está

representado o momento no qual o robô descobre a p-ésima *landmark* do ambiente.

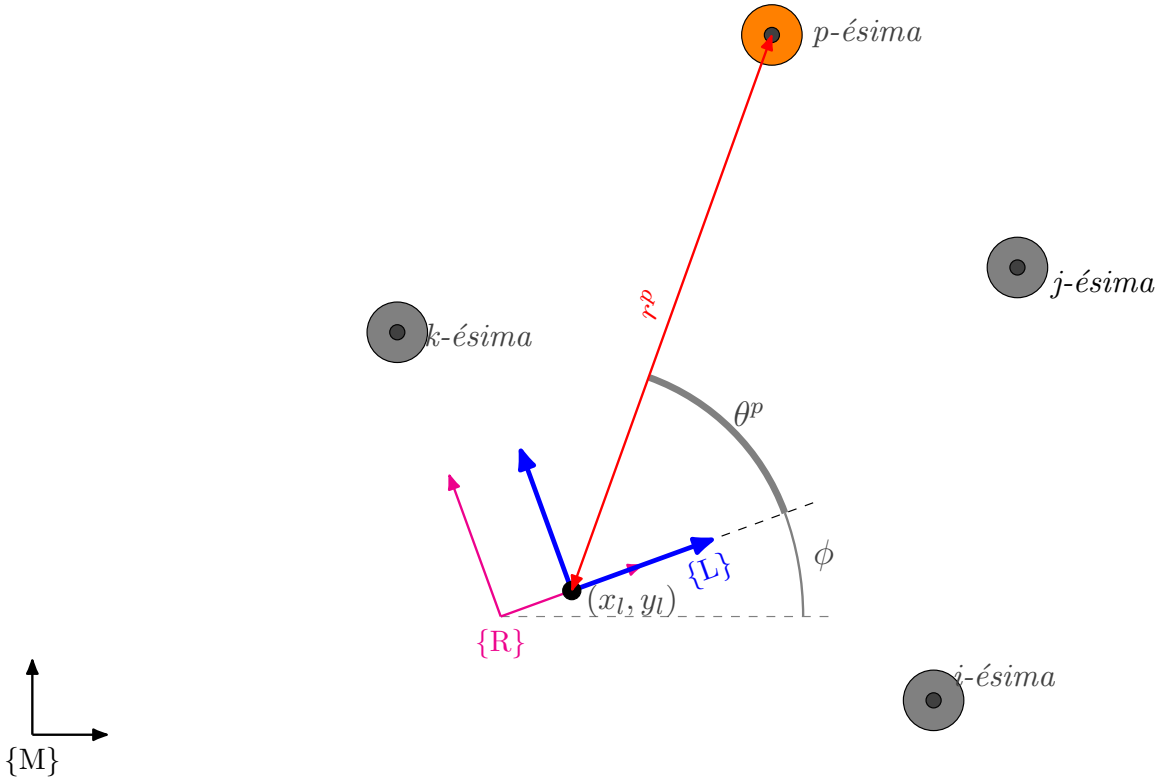


FIGURA 2.5 – Esquemático do modelo de medida *Range-Bearing*. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. Os círculos representam *landmarks*, as conhecidas pelo robô (portanto presentes no vetor de estados), em cinza, e recém descobertas, em laranja. A p-ésima *landmark* acaba de ser encontrada pelo robô.

O modelo de medida inverso $\boldsymbol{f}(\bullet, \bullet)$ é descrito na Equação 2.11. Como pode ser observado, assim como o modelo de media "direto", o modelo de medida inverso é não linear, logo devemos lineariza-lo para utilizá-lo com o EKF-SLAM e seus algoritmos derivados. 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.

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

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

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

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

2.4 Simulação

2.5 Visualização

3 Aplicação da estimação no problema SLAM (*Backend*)

O famoso Filtro de Kalman (KF) é uma técnica de estimação ótima para sistemas lineares com ruídos gaussianos, nele a distribuição de probabilidade do estado estimado é representada por uma gaussiana, Eq. 3.1, parametrizada pelos momentos média e covariância. Ele foi desenvolvido simultaneamente em 1958 por Peter Swerling, e em 1960 por Rudolf Kalman (BONGARD, 2006, 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 série de Taylor, e a premissa de que os termos de ordem maior ou igual a dois são desprezíveis.

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

Essa técnica derivada do KF para sistemas não lineares é conhecida como Filtro de Kalman Extendido (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 de SLAM, estimando a pose do robô e a posição das *landmarks*, o que é conhecido como EKF-SLAM. Além disso, também serão abordadas técnicas decorrentes do EKF-SLAM como EIF-SLAM e SEIF-SLAM, sendo esta última a técnica de estimação utilizada neste trabalho.

3.1 Filtro de Kalman Extendido

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, Eq. 3.2; a segunda, chamada de modelo de medida, relaciona o estado do sistema com a medida esperada, gerada pelo sensor, Eq. 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 $\boldsymbol{g}(\bullet, \bullet)$, e o modelo de medida $\boldsymbol{h}(\bullet, \bullet)$ 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 $\boldsymbol{g}(\bullet, \bullet)$ e/ou $\boldsymbol{h}(\bullet, \bullet)$ 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. Onde $\mathbf{G}_t$ é o jacobiano do modelo do sistema no ponto $\boldsymbol{\mu}_{t-1}$, e $\mathbf{H}_t$ é o jacobiano do modelo de medida no ponto $\overline{\boldsymbol{\mu}}_t$.

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

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

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

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

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

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

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

Porém, a resolução do problema de 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, como o mapa são estimados, o vetor de estados é composto

[1] O leitor pode estanhar a Equação 3.10 do erro da estimativa. 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.

por ambos.

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

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, 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*:

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

Portanto, o passo de predição do vetor média 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 posições de memória da pose são modificadas 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 $\mathcal{O}(n)$ por conta do termo $\mathbf{G}_t^{\mathrm{r}}\mathbf{P}_{t-1}^{\mathrm{rm}}$, enquanto no caso geral do EKF onde o jacobiano $\mathbf{G}_t$ é denso, essa complexidade é $\mathcal{O}(n^3)$, que é a complexidade prática da multiplicação de matrizes $n \times n$. A Figura 3.1 ilustra as porções do vetor de estados, e da matriz de covariância, modificadas no passo de predição do EKF-SLAM.

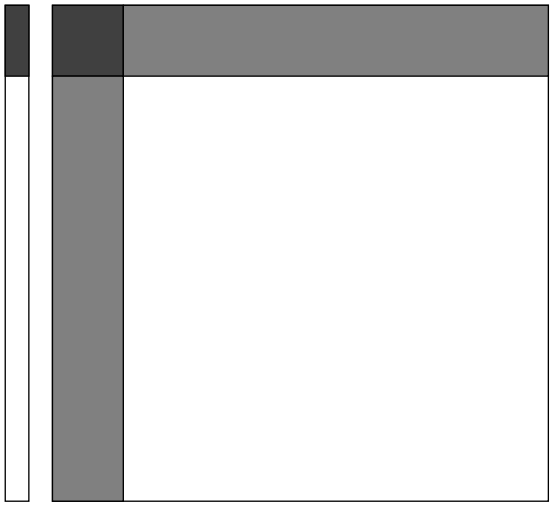


FIGURA 3.1 – Partes modificadas do vetor média e da matriz de covariância durante o movimento do robô. O vetor média é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As partes modificadas, em tons de cinza, correspondem ao estado do robô $\boldsymbol{\mu}_{\mathrm{r}}$ 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", e normalmente a inovação $\mathbf{z}_t$ é calculada 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, o cálculo da covariância da inovação pode ser obtido 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 da 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 dos cálculos da 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. Embora, aqui esse cálculo de complexidade constante deve ser repetido para cada observação presente no vetor de medidas, ou seja, no EKF-SLAM o cálculo da inovação, e de sua covariância possuem complexidade linear no número de medidas obtidas.

O cálculo do Ganho de 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 do cálculo do Ganho de Kalman, $\mathbf{K}_t^j$, é $\mathcal{O}(n)$ 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 no cálculo da inovação e de sua matriz de covariância.

A Figura 3.3 deixa claro que todos os elementos do vetor média e da matriz de covariâncias 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 próximas de zero. Esse tipo de correlação "fraca" será explorada pelo Filtro de Informação Extendido Esparso, a fim de obter-se um algoritmo de estimação mais eficiente.

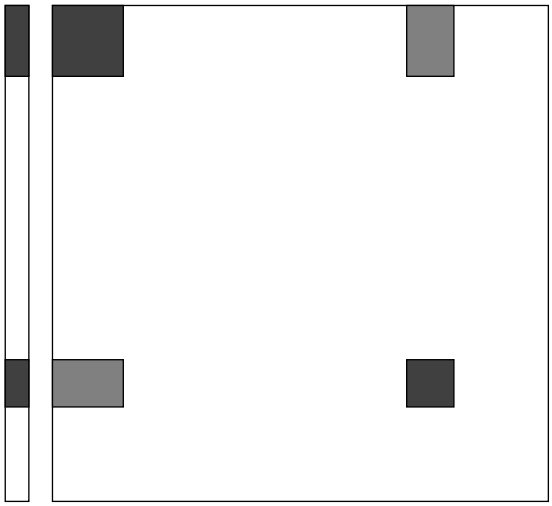


FIGURA 3.2 – Partes utilizadas do vetor média e da matriz de covariância durante o cálculo da inovação, quando uma *landmark* é observada. O vetor média é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As porções utilizadas, em tons de cinza, correspondem ao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

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 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 média e à matriz de covariância, aumentando a dimensão do sistema.

Para adicionar uma nova *landmark* no vetor de estado, será definida a função $\boldsymbol{\sigma}(\bullet, \bullet)$, ela gera um novo vetor de estados que é resultado da concatenação do vetor atual, com a posição da nova *landmark* calculada pelo modelo de medida inverso, descrito na Seção 2.3.2, 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 covariâncias. 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 de medição do sensor.

Inicializar a covariância da nova *landmark* com ∞ (ou números muito grandes), como

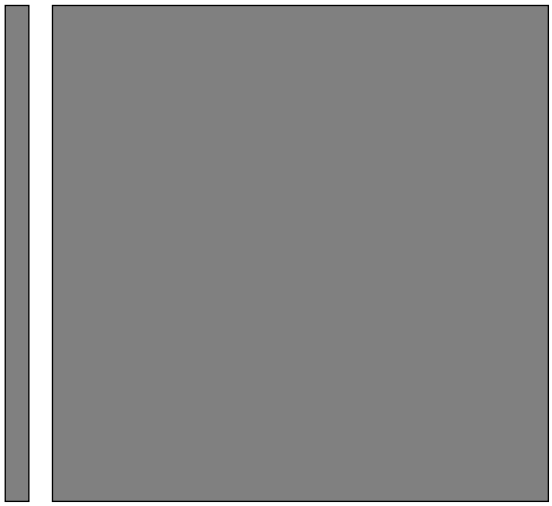


FIGURA 3.3 – O vetor média e a matriz de covariâncias são completamente atualizados durante a observação de uma *landmark*. Retirado de (SOLÀ, 2014, p. 8).

indicado em (BONGARD, 2006, p. 317), pode ser injusto. 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.

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

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*, concatenada as covariâncias cruzadas $\mathbf{P}_t \mathbf{F}_X^T$ e $\mathbf{F}_X \mathbf{P}_t$, e com 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 $\mathbf{P}_t \mathbf{F}_x$ pode ser reduzida de $\mathcal{O}(n^2)$ para $\mathcal{O}(n)$ se todos os cálculos inúteis forem ignorados, logo toda a operação de inserção de *landmark* tem custo $\mathcal{O}(n)$. A Figura 3.4 mostra o vetor média e a matriz de covariância com as novas inserções destacadas.

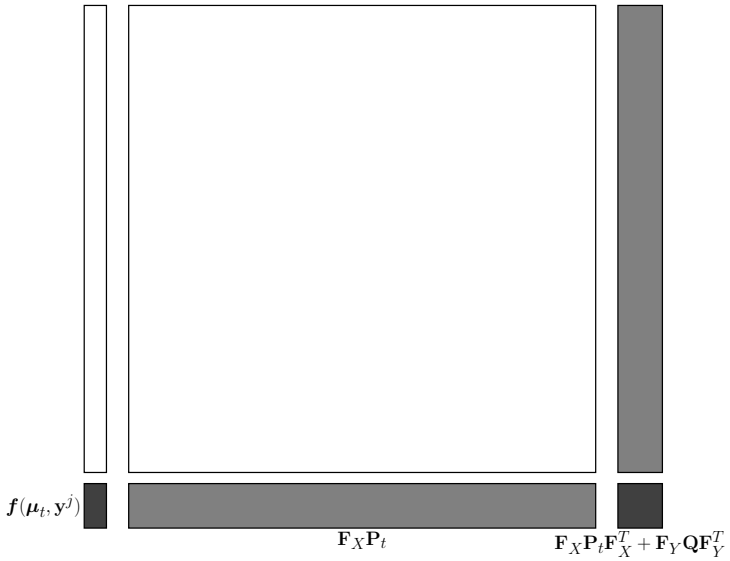


FIGURA 3.4 – Vetor média e matriz de covariância aumentados após inserção de nova *landmark*. As partes adicionadas, em cinza, correspondem às covariâncias cruzadas entre a nova landmark e o vetor de estados anterior (cinza claro), e à média da nova *landmark* e sua covariância (cinza escuro). Adaptado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

3.3 Filtro de Informação Extendido (EIF)

O Filtro de Kalman Extendido, apresentado na Seção 3.1 utiliza a parametrização de momentos (Eq. 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:

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

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

Com as parametrizações acima, o EKF pode ser reescrito na forma do Filtro de Informação Extendido:

$$\boldsymbol{\mu}_{t-1} = \boldsymbol{\Omega}_{t-1}^{-1}\boldsymbol{\xi}_{t-1} \tag{3.24}$$
$$\overline{\boldsymbol{\Omega}}_t = (\mathbf{G}_t\boldsymbol{\Omega}_{t-1}^{-1}\mathbf{G}_t^T + \mathbf{R}_t)^{-1} \tag{3.25}$$
$$\overline{\boldsymbol{\mu}}_t = \boldsymbol{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) \tag{3.26}$$
$$\overline{\boldsymbol{\xi}}_t = \overline{\boldsymbol{\Omega}}_t\,\overline{\boldsymbol{\mu}}_t \tag{3.27}$$
$$\boldsymbol{\Omega}_t = \overline{\boldsymbol{\Omega}}_t + \mathbf{H}_t^T\mathbf{Q}_t^{-1}\mathbf{H}_t \tag{3.28}$$
$$\boldsymbol{\xi}_t = \overline{\boldsymbol{\xi}}_t + \mathbf{H}_t^T\mathbf{Q}_t^{-1}\left[\mathbf{y}_t - \boldsymbol{h}(\overline{\boldsymbol{\mu}}_t) + \mathbf{H}_t\overline{\boldsymbol{\mu}}_t\right] \tag{3.29}$$

e a distribuição de probabilidade na Eq. 3.1 pode ser reescrita como:

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

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 quando comparado com o Filtro de Kalman, aqui basta definir $\boldsymbol{\Omega} = \mathbf{0}$, 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, onde a informação é coletada de maneira descentralizada (BONGARD, 2006, 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 $\mathcal{O}(n^3)$. 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 seja 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 (BONGARD, 2006, 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. Isso se deve à estrutura do problema SLAM, pois grande parte das correlações entre *landmarks* (portanto fora da diagonal principal) são propagadas pela incerteza na pose do robô, quando essas (as *landmarks*) são observadas por ele. Apenas *landmarks* dentro de uma mesma vizinhança são observadas juntas resultando em alta correlação.

Esse aspecto é explorado pelo Filtro de Informação Extendido Esparso (SEIF), por meio de aproximações o SEIF mantém a matriz de informação diagonalizada, aproximando os elementos 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 Extendido Esparso (SEIF), e como ele endereça 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 atingir essas façanhas, o SEIF mantém a matriz de informação com formato próximo ao de uma matriz diagonal, por meio do uso de *landmarks* ativas e passivas, 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 (BONGARD, 2006, Capítulo 12.4).

3.4.1 SEIF-SLAM: Landmarks ativas e passivas

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* passivas, 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 sí, como ocorre no EKF-SLAM e EIF-SLAM. Na verdade, aqui, elas são localmente correlacionadas com sua vizinhança. Onde vizinhança é definida 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 onde a inovação de uma *landmark* afeta todo o sistema.

O conjunto $\mathbf{m}_t^+$ contém as k últimas *landmarks* observadas até o instante t, onde k é o tamanho 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.

Nas próximas Seções, ficará claro que o tamanho definido 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

O passo de predição do SEIF-SLAM está condensado no Algoritmo 1, abaixo. As Seções que se seguem derivam os passos do algoritmo a partir das equações de predição do EIF, 3.25, 3.27 e 3.26. A esparsidade da matriz de informação é usada como premissa para garantir o tempo de execução constante, a esparsificação em sí será tratada mais adiante, por hora vamos assumir que a matriz é esparsa.

Algorithm 1 SEIF-SLAM passo de predição

function SEIF-SLAM-PREDICTION($\boldsymbol{\xi}_{t-1}, \boldsymbol{\mu}_{t-1}, \boldsymbol{\Omega}_{t-1}, \mathbf{u}_t$)
  $\boldsymbol{\Psi}_t \leftarrow \mathbf{M}_{x_\mathrm{r}}^T \left([\mathbf{G}_t^\mathrm{r}]^{-1} - \mathbf{I}_3 \right) \mathbf{M}_{x_\mathrm{r}}$
  $\boldsymbol{\lambda}_t \leftarrow \boldsymbol{\Psi}_t^T \boldsymbol{\Omega}_{t-1} + \boldsymbol{\Psi}_t^T \boldsymbol{\Omega}_{t-1} \boldsymbol{\Psi}_t + \boldsymbol{\Omega}_{t-1} \boldsymbol{\Psi}_t$
  $\boldsymbol{\Phi}_t \leftarrow \boldsymbol{\Omega}_{t-1} + \boldsymbol{\lambda}_t$
  $\boldsymbol{\kappa}_t \leftarrow \boldsymbol{\Phi}_t \mathbf{M}_{x_\mathrm{r}}^T \left(\mathbf{R}_{R,t}^{-1} + \mathbf{M}_{x_\mathrm{r}} \boldsymbol{\Phi}_t \mathbf{M}_{x_\mathrm{r}}^T \right)^{-1} \mathbf{M}_{x_\mathrm{r}} \boldsymbol{\Phi}_t$
  $\overline{\boldsymbol{\Omega}}_t \leftarrow \boldsymbol{\Phi}_t - \boldsymbol{\kappa}_t$
  $\overline{\boldsymbol{\xi}}_t \leftarrow (\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t) \boldsymbol{\mu}_{t-1} + \boldsymbol{\xi}_{t-1} + \overline{\boldsymbol{\Omega}}_t \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t}$
  $\overline{\boldsymbol{\mu}}_t \leftarrow \boldsymbol{\mu}_{t-1} + \mathbf{M}_{x_\mathrm{r}}^T \boldsymbol{\delta}_{r,t}$
  return $\overline{\boldsymbol{\xi}}_t, \overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\Omega}}_t$
end function

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

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

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

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

3.4.2.1 Predição da Matriz de Informação

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

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

Onde:

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

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

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

Para calcularmos

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

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

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

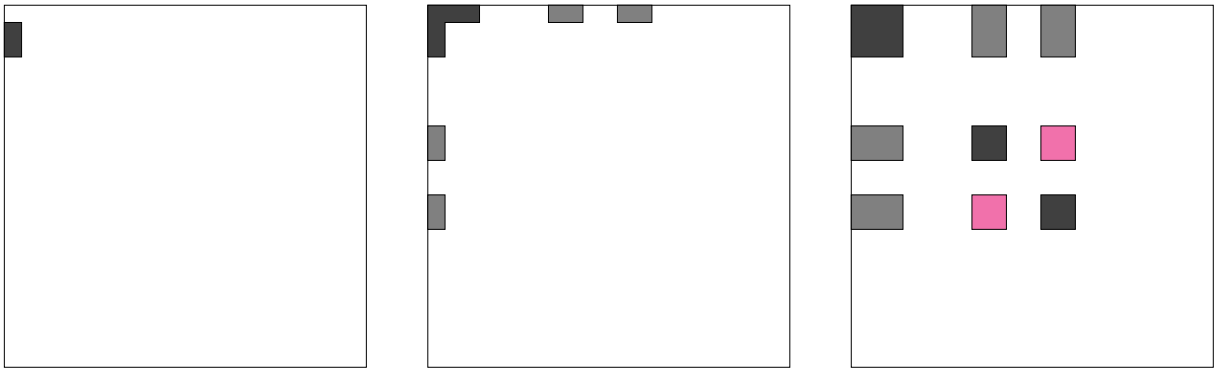
Note que $\boldsymbol{\Psi}_t$ é uma matriz de dimensão n, onde apenas os elementos do bloco superior esquerdo 3×3 , são diferentes de zero. Usando a Equação 3.37 na Equação 3.34 temos:

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

Como $\boldsymbol{\Psi}_t$ é esparsa e com quantidade de elementos não nulos constante, $\boldsymbol{\lambda}_t$ também será esparsa e com elementos não nulos constantes, 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, o cálculo de $\boldsymbol{\Phi}_t$ a partir de $\boldsymbol{\Omega}_{t-1}$ é constante, pois resulta da subtração dos

elementos não nulos de $\boldsymbol{\lambda}_t$, de $\boldsymbol{\Omega}_{t-1}$. Na Figura 3.5 são representadas as matrizes $\boldsymbol{\Psi}_t, \boldsymbol{\lambda}_t, \boldsymbol{\kappa}_t$ obtidas durante a execução do SEIF-SLAM com tamanho do conjunto de *landmarks* ativas igual a dois.



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

FIGURA 3.5 – 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 tamanho 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.

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 3.5 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. Primeiro, vamos reescrever o modelo de movimento na Equação Referênciaseq:motion-model, 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 $(\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t)\, \boldsymbol{\mu}_{t-1}$ contém um número determinado de elementos não nulos, e portanto é calculado em tempo constante. O produto $(\boldsymbol{\lambda}_t - \boldsymbol{\kappa}_t)\, \boldsymbol{\mu}_{t-1}$ resulta em uma matriz nula exceto pelo primeiro bloco 3×3, e ao multiplica-la pela matriz de informação predita, que também é esparsa, temos como resultado um vetor esparso (BONGARD, 2006, p. 398). Portanto, é necessário um número constante de operações, que independe do tamanho do mapa, para calcular o vetor de informação predito [2].

3.4.3 SEIF-SLAM: Recuperação da média

Outro desafio a ser resolvido no SEIF-SLAM, para manter o tempo de predição e/ou atualização constantes, é o cálculo do vetor média. Fundamental para as linearizações dos modelos de movimento e medida, e para no cálculo do vetor de informação na etapa de esparsificação. Na Equação 3.24, do Filtro de Informação Extendido, esse cálculo é feito através da inversão da matriz de informação, e, mesmo numa matriz esparsa, 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 (BONGARD, 2006, Cap. 12.6) mostra-se que o vetor média equivale ao ponto de máximo da quadrática

[2] Comumente a orientação do robô é normalizada para o intervalo $[-\pi, \pi[$ no vetor média, 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 $|\mathbf{m}^+|$.

na representação canônica da gaussiana Eq. 3.30, então o vetor média 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, Algoritmo 2, que apesar de não ser tão sofisticada quanto os algoritmos citados, se mostrou boa o suficiente para 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.

Algorithm 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 2 do algoritmo acima, 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 e o tamanho 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 Eq. 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 $\boldsymbol{h}(\bullet, \bullet)$, Eq. 2.6.

O algoritmo abaixo pressupõe que tais correspondências são conhecidas. Mais adiante na Seção 3.5 será abordado uma maneira de como essas correspondências são estabelecidas. Ao contrário dos outros passos, no passo de atualização a esparsidade da matriz de informação não influencia e/ou altera nada em relação ao Filtro de Informação Extendido. As equações de atualização nas linhas 3 e 4 abaixo, são as mesmas equações 3.28 e 3.29 do EIF.

Algorithm 3 Etapa de atualização do SEIF-SLAM com associação conhecida

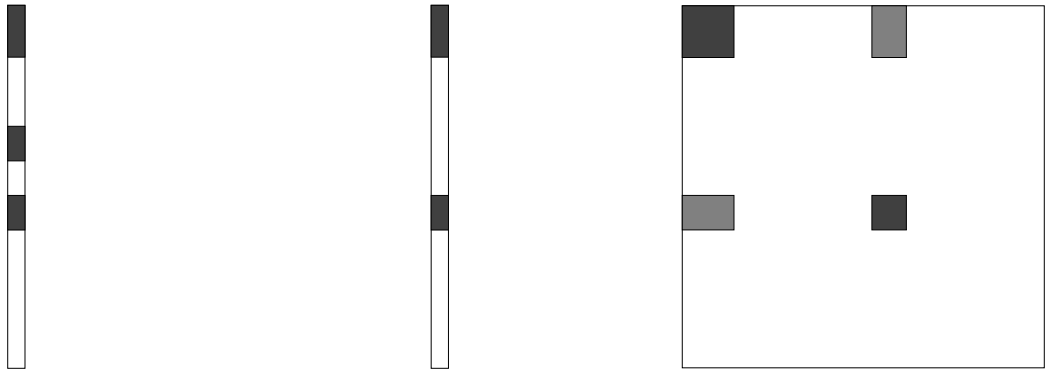
1: **function** SEIF-SLAM-ATUALIZAÇÃO($\overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\xi}}_t, \overline{\boldsymbol{\Omega}}_t, \mathbf{y}, j =$ índice da *landmark*)
2: $\quad \mathbf{z} \leftarrow \mathbf{y} - \boldsymbol{h}^j(\overline{\boldsymbol{\mu}}_t)$
3: $\quad \boldsymbol{\Omega}_t \leftarrow \overline{\boldsymbol{\Omega}}_t + \mathbf{H}_t^T \mathbf{Q}_t^{-1} \mathbf{H}_t$
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 3.6. O SEIF herda o caráter "local"da atualização do EIF, e difere do EKF onde a atualização altera todos os elementos do vetor média 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 com 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. Utilizando o resultado do



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

FIGURA 3.6 – 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 média, 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.

aumento da matriz de covariâncias, repetido a seguir:

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

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:

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

Onde, pelo lema da inversão na forma de blocos (Apêndice **??**) temos:

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

$$\begin{aligned}\mathbf{Y} &= -\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T\mathbf{U} \\ &= \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\end{aligned} \tag{3.44}$$

$$\begin{aligned}\mathbf{Z} &= -\mathbf{U}\mathbf{F}_X\boldsymbol{\Omega}_t^{-1}\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1} \\ &= -\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X\end{aligned} \tag{3.45}$$

$$\begin{aligned}\mathbf{X} &= \left[\boldsymbol{\Omega}_t^{-1}\right]^{-1} + \left[\boldsymbol{\Omega}_t^{-1}\right]^{-1}\boldsymbol{\Omega}_t^{-1}\mathbf{F}_X^T\,\mathbf{U}\,\mathbf{F}_X\boldsymbol{\Omega}_t^{-1}\left[\boldsymbol{\Omega}_t^{-1}\right]^{-1} \\ &= \boldsymbol{\Omega}_t + \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X\end{aligned} \tag{3.46}$$

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

$$\boldsymbol{\Omega}_t^* = \begin{bmatrix}\boldsymbol{\Omega}_t + \mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X & -\mathbf{F}_X^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right) \\ -\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_X & \left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\end{bmatrix} \tag{3.47}$$

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

Isso pode ser concluído a partir da observação da forma da matrix $\mathbf{F}_X$, Eq. 2.12, nula em todos elementos exceto pelo primeiro bloco 2×3. Para deixar isso em evidência a Eq. 3.47 é reescrita abaixo:

$$\boldsymbol{\Omega}_t^* = \begin{bmatrix}\boldsymbol{\Omega}_{RR,t} + \mathbf{F}_R^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_R & \boldsymbol{\Omega}_{RM,t} & -\mathbf{F}_R^T\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right) \\ \boldsymbol{\Omega}_{RM,t}^T & \boldsymbol{\Omega}_{MM,t} & \mathbf{0}_{n-3\times 2} \\ -\left(\left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\right)\mathbf{F}_R & \mathbf{0}_{2\times n-3} & \left[\mathbf{F}_Y^T\right]^{-1}\mathbf{Q}^{-1}\mathbf{F}_Y^{-1}\end{bmatrix} \tag{3.48}$$

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

vetor de informação aumentado:

$$\begin{aligned}\boldsymbol{\xi}_t^* &= \boldsymbol{\Omega}_t^* \boldsymbol{\mu}_t^* \\ &= \begin{bmatrix} \begin{bmatrix} [\boldsymbol{\Omega}_t^{\mathrm{rr}}]^* & [\boldsymbol{\Omega}_t^{\mathrm{rm}}]^* \end{bmatrix} \boldsymbol{\mu}_t^* \\ \boldsymbol{\xi}_t^{\mathrm{m}} \\ \begin{bmatrix} \left[\boldsymbol{\Omega}_t^{\mathrm{rm^j}}\right]^* & \left[\boldsymbol{\Omega}_t^{\mathrm{m^j m^j}}\right]^* \end{bmatrix} \boldsymbol{\mu}_t^* \end{bmatrix}\end{aligned} \tag{3.49}$$

Assim como na matriz de informação, os elementos 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 3.7 representa as porções alteradas e/ou adicionadas à matriz de informação e ao vetor de informação.

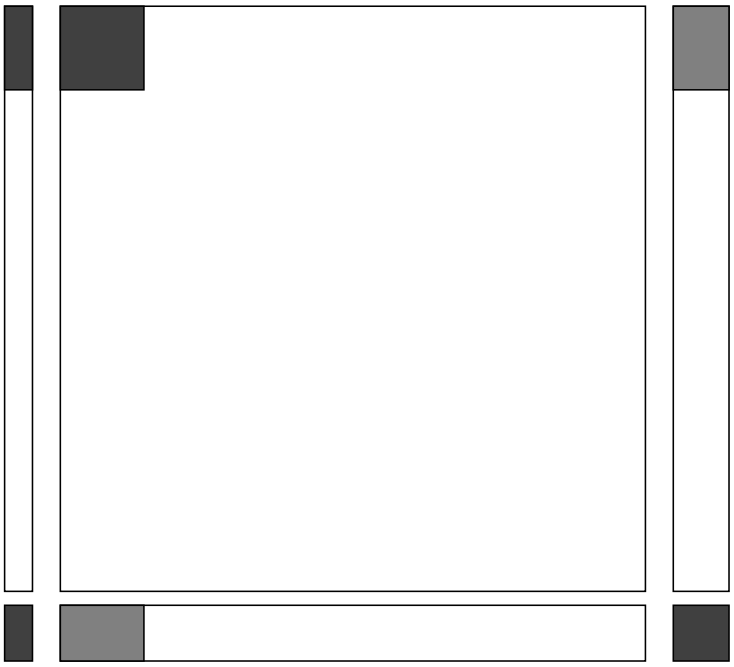


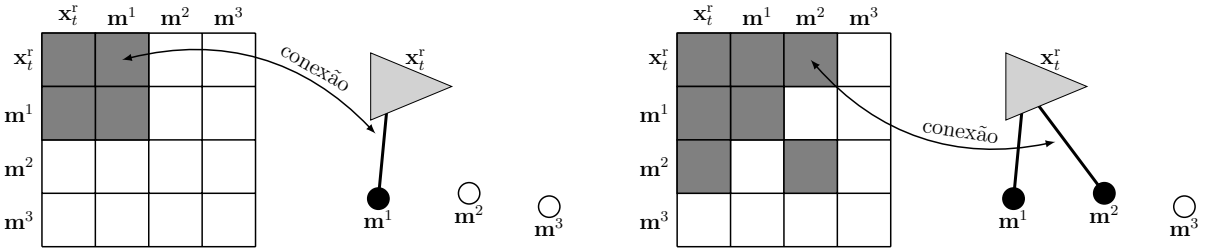
FIGURA 3.7 – Vetor e matriz de informação aumentados após inserção de nova *landmark*. As partes adicionadas, em cinza, correspondem à informação cruzada entre a nova landmark e o robô (cinza claro), e à média da nova *landmark* e do robô, e suas informações (cinza escuro).

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 o cálculo das novas porções possui complexidade $\mathcal{O}(n)$.

3.4.6 SEIF-SLAM: Esparsificação da matriz de informação

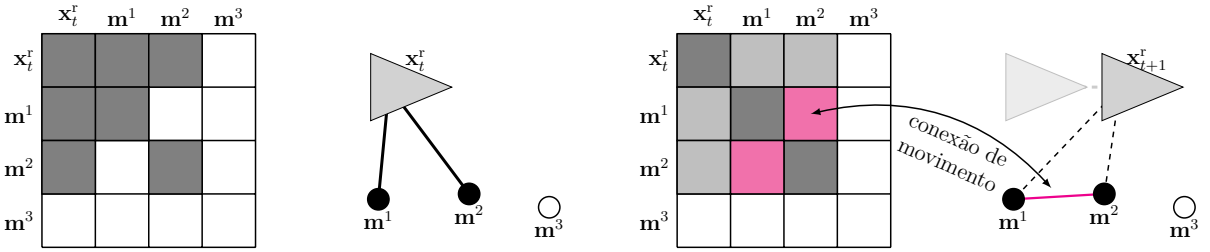
A esparsidade da matriz de informação foi utilizada como premissa em toda a discussão nas Seções 3.4.2, 3.4.3 e 3.4.4. Esta Seção mostrará como a operação de esparsificação é realizada, mas primeiro será ilustrado como a matriz de de informação é povoada nos passos de atualização e predição e como a esparsificação elimina a informação cruzada entre *landmarks*, mostrada na Figura 3.5c, tornando a matriz de informação esparsa.

Ao longo das Figuras 3.8 até 3.12 é possível observar: A inserção de novas *landmarks* na matriz de informação no momento em que são observadas Figuras 3.8 e 3.10b; Como as informações cruzadas entre *landmarks* surgem através do movimento do robô (passo de predição do filtro), Figuras 3.9b e 3.12b; A eliminação dessa informação cruzada entre a pose do robô e a posição de uma *landmark* através da esparsificação, Figura 3.11; Como a eliminação de informação cruzada entre o robô e uma *landmark* impede a criação de novas conexões de movimento, Figura 3.12.



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

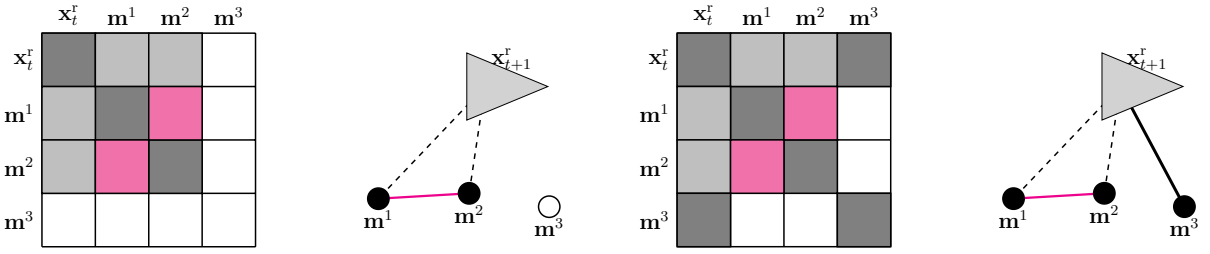
FIGURA 3.8 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks*, durante a observação das *landmarks* $\mathbf{m}^1$ e $\mathbf{m}^2$ no instante t. Os elementos não nulos da matriz de informação estão representados em cinza. Adaptado de (BONGARD, 2006, p. 389).



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

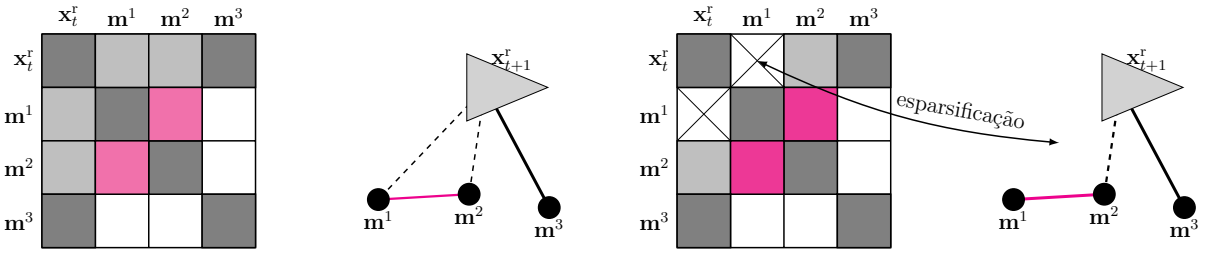
FIGURA 3.9 – Matriz de informação (representada pela grade) ao lado do esquemático do robô e *landmarks*, antes (esquerda) e depois (direita) do movimento do robô, entre os instantes t e $t+1$. A conexão de movimento gerada entre as *landmarks* $\mathbf{m}^1$ e $\mathbf{m}^2$ é mostrada em magenta, assim como os elementos correspondentes na matriz de informação. Os demais elementos não nulos estão representados em tons de cinza. Adaptado de (BONGARD, 2006, p. 389).

Através da sequência de esquemas apresentados acima foi possível desenvolver uma noção de como a matriz de informação é construída ao longo do tempo e, como a esparsificação impede a criação de conexões entre *landmarks*. Como foi mostrado, a esparsificação



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

FIGURA 3.10 – 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 (BONGARD, 2006, p. 389).



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

FIGURA 3.11 – 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 (BONGARD, 2006, p. 389).

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 nulidade entre os elementos correspondentes da matriz de informação.

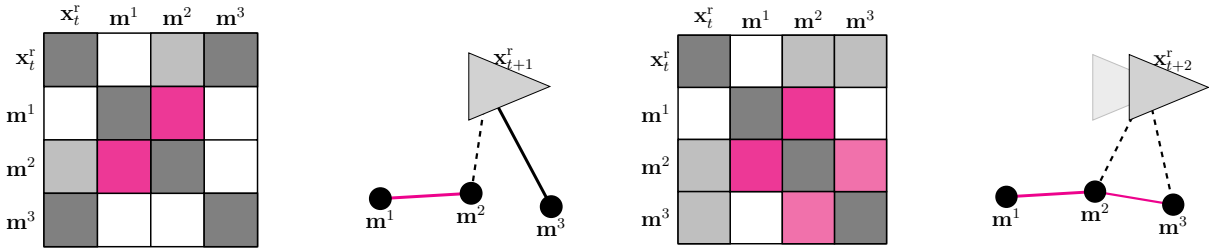
Para realizar a esparsificação da matriz de informação do SEIF-SLAM, vamos retomar os conjuntos de *landmarks* $\mathbf{m}_t^+$ e $\mathbf{m}_t^-$ definidos na Seção 3.4.1. Também vamos subdividir $\mathbf{m}_t^+$ em:

$$\mathbf{m}_t^+ = \mathbf{m}_t^{++} \cup \mathbf{m}_t^{+-} \tag{3.50}$$

onde

- $\mathbf{m}_t^{+-}$ é a porção do conjunto de *landmarks* ativas $\mathbf{m}_t^+$ que serão tornadas passivas, ou seja, passarão a compor o conjunto $\mathbf{m}_t^-$ e terá 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 em 1.1 utilizando a regra do produto e aproximar a distribuição $p\left(\mathbf{x}_t^r \,|\, \mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^- = 0\right)$



(a) Antes do movimento

(b) Depois do movimento

FIGURA 3.12 – 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 *landamrks* $\mathbf{m}^1$ e $\mathbf{m}^3$. Os elementos nulos da matriz de informação estão representados em branco.

pela marginal $p\left(\mathbf{x}_t^{\mathrm{r}} \,|\, \mathbf{m}^{++}, \mathbf{m}^{-} = 0\right)$, como é mostrado abaixo:

$$\begin{aligned} p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) &= p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\ &= p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\ &= p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\ &\approx p\left(\mathbf{x}_t^{\mathrm{r}}, \,|\, \mathbf{m}^{++}, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \\ &\approx \frac{p\left(\mathbf{x}_t^{\mathrm{r}}, \mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)}{p\left(\mathbf{m}^{++} \,|\, \mathbf{m}^{-} = 0, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)} p\left(\mathbf{m}^{+-}, \mathbf{m}^{++}, \mathbf{m}^{-} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \end{aligned} \tag{3.51}$$

Os cálculos das matrizes de informações correspondentes às distribuições da última linha acima do desenvolvimento acima serão feitos com base na discussão em (BONGARD, 2006, p. 401) e, dos resultados de manipulação de distribuições gaussianas na forma canônica presentes no Apêndice 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 $\mathbf{m}^{-}$:

$$\mathbf{\Omega}_t^0 = \mathbf{M}_{x_{\mathrm{r}}, m^{++}, m^{+-}}^T \mathbf{M}_{x_{\mathrm{r}}, m^{++}, m^{+-}} \, \mathbf{\Omega}_t \, \mathbf{M}_{x_{\mathrm{r}}, m^{++}, m^{+-}}^T \mathbf{M}_{x_{\mathrm{r}}, m^{++}, m^{+-}} \tag{3.52}$$

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 $\mathbf{\Omega}_t^2$, respectivamente:

$$\mathbf{\Omega}_t^1 = \mathbf{\Omega}_t^0 - \mathbf{\Omega}_t^0 \, \mathbf{M}_{m^{+-}}^T \left(\mathbf{M}_{m^{+-}} \, \mathbf{\Omega}_t^0 \, \mathbf{M}_{m^{+-}}^T\right)^{-1} \mathbf{M}_{m^{+-}} \, \mathbf{\Omega}_t^0 \tag{3.53}$$

$$\mathbf{\Omega}_t^2 = \mathbf{\Omega}_t^0 - \mathbf{\Omega}_t^0 \, \mathbf{M}_{x_{\mathrm{r}}, m^{+-}}^T \left(\mathbf{M}_{x_{\mathrm{r}}, m^{+-}} \, \mathbf{\Omega}_t^0 \, \mathbf{M}_{x_{\mathrm{r}}, m^{+-}}^T\right)^{-1} \mathbf{M}_{x_{\mathrm{r}}, m^{+-}} \, \mathbf{\Omega}_t^0 \tag{3.54}$$

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

$$\mathbf{\Omega}_t^3 = \mathbf{\Omega}_t - \mathbf{\Omega}_t \, \mathbf{M}_{x_{\mathrm{r}}}^T \left(\mathbf{M}_{x_{\mathrm{r}}} \, \mathbf{\Omega}_t \, \mathbf{M}_{x_{\mathrm{r}}}^T\right)^{-1} \mathbf{M}_{x_{\mathrm{r}}} \, \mathbf{\Omega}_t \tag{3.55}$$

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 Eq. 3.51 é obtida:

$$\tilde{\boldsymbol{\Omega}}_t = \boldsymbol{\Omega}_t^1 - \boldsymbol{\Omega}_t^2 + \boldsymbol{\Omega}_t^3 \tag{3.56}$$

Após a esparsificação da matriz de informação, para manter a equivalência abaixo,

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

é preciso recalcular o vetor de informação:

$$\begin{aligned}
\tilde{\boldsymbol{\xi}}_t &= \tilde{\boldsymbol{\Omega}}_t \boldsymbol{\mu}_t \\
&= \left(\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_t + \tilde{\boldsymbol{\Omega}}_t\right) \boldsymbol{\mu}_t \\
&= \boldsymbol{\Omega}_t \boldsymbol{\mu}_t + \left(\tilde{\boldsymbol{\Omega}}_t - \boldsymbol{\Omega}_t\right) \boldsymbol{\mu}_t \\
&= \boldsymbol{\xi}_t + \left(\tilde{\boldsymbol{\Omega}}_t - \boldsymbol{\Omega}_t\right) \boldsymbol{\mu}_t
\end{aligned} \tag{3.57}$$

As Equações 3.52 até 3.57 estão sumarizadas no Algoritmo 4, que trata da esparsificação da matriz de informação do SEIF utilizada como premissa para no desenvolvimento das etapas anteriores.

Algorithm 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

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). Além disso, é preciso diferenciar quando entre os momentos que uma *landmark* é re-observada e quando é observada pela primeira vez.

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 independentemente do "*backend*"utilizado (EKF, EIF, Filtro de Partículas, etc), no entanto a discussão aqui apresentada focará nas particularidades do SEIF.

Para calcular as correspondências foi utilizada o método guloso *Individual Compatibility Nearest Neighbor* (ICNN), Algoritmo 5, em linhas gerais ele atribui a *landmark* cuja distribuição de probabilidade possui a menor distância de Mahalanobis (MCLACHLAN, 1999) para a medida lida. E caso essa distância seja menor que um limiar $\chi^2_{2,\alpha}$, é considerado que essa medida 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 3.7.

Algorithm 5 Associação da medida $\mathbf{y}_t$ com *landmark* do vetor de estados

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

Como pode ser observado acima, o Algoritmo utiliza as covariâncias das *landmarks*, denominadas por $\mathbf{P}_{0:\mathrm{M}}$, como entrada. Porém, no SEIF não é mantida uma matriz de covariâncias, 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.

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 $\mathcal{O}(2.376)$ (COPPERSMITH; WINOGRAD, 1987).

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 3.13.

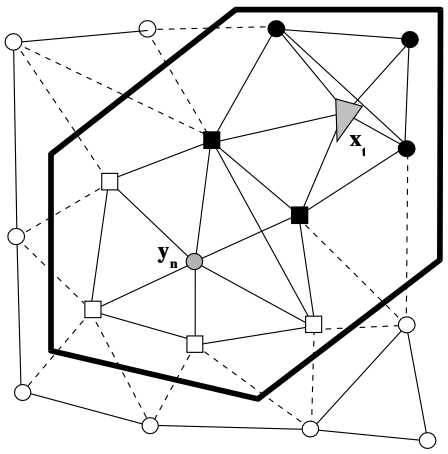


FIGURA 3.13 – Rede de conexões entre *landmarks* e o robô. A maioria das *landmarks* estão representadas por círculos, as *landmarks* vizinhas da *landmark* candidata a associação estão representadas por quadrados. As *landmarks* ativas estão coloridas de preto, a *landmark* candidata está colorida de 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 (BONGARD, 2006, p. 410).

Utilizando o resultado do condicionamento da distribuição gaussiana na forma de informação apresentado no Apêndice **??** e já utilizado na esparsificação, calcula-se a matriz de covariância aproximada $\tilde{\mathbf{P}}_{\mathrm{m^j}}$ da seguinte forma:

$$\tilde{\mathbf{P}}_{\mathrm{m^j}} = \mathbf{M}_{\mathrm{m^j}} \left(\mathbf{M}_{\mathbf{x}^{\mathrm{r}}_t, \mathrm{m^j}, \mathbf{m}^{+}_{c_t}} \, \boldsymbol{\Omega}_t \, \mathbf{M}^{T}_{\mathbf{x}^{\mathrm{r}}_t, \mathrm{m^j}, \mathbf{m}^{+}_{c_t}} \right)^{-1} \mathbf{M}^{T}_{\mathrm{m^j}} \tag{3.58}$$

Entretanto ao contrário do que é mostrado na Figura 3.13 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 sua posição 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 re-observada. Enquanto a degradação diminui dois pontos sempre que a *landmark* não é re-observada.

$$\mathcal{Q}_{t+1}\left(\mathbf{m}^j\right) = \begin{cases} \mathbf{m}^j \text{ é re-observada} & \implies \mathcal{Q}_t\left(\mathbf{m}^j\right) + 1 \text{ (atualização)} \\ \mathbf{m}^j \text{ não é re-observada} & \implies \mathcal{Q}_t\left(\mathbf{m}^j\right) - 2 \text{ (degradação)} \end{cases} \tag{3.59}$$

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 a 10, e é

excluída quando pontua cinco pontos negativamente:

$$\begin{cases} \mathcal{Q}_t\left(\mathbf{m}^j\right) \geq 10 & \text{Promoção} \\ \mathcal{Q}_t\left(\mathbf{m}^j\right) \leq -5 & \text{Remoção} \end{cases} \tag{3.60}$$

Com essa abordagem foi possível erradicar a incorporação de falsas *landmarks* geradas tanto no cenário com um robô quanto no cenário dois robôs. Além disso, como será visto na próxima 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 a 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 estados têm mesmo tipo de variabilidade no tempo.

Também foram mostradas as adaptações necessárias para utilizar o Filtro de Kalman Extendido 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*, que se mostrou boa o suficiente para as configurações de ambiente utilizadas neste trabalho. E por fim, foi tratado o problema prático da detecção de falsas *landmarks* e evitá-lo.

4 SLAM *Frontend*

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 2.3 (Medidas e o modelo de medida *Range-Bearing*).

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 2.1.

4.1.2 Processamento de dados

Para transformar o dado bruto, a sequência de distâncias na Figura 4.1b, nas medidas consumidas pelo modelo de medida descrito na Seção *Medidas e o modelo de medida Range-Bearing*, é proposto um *pipeline* de processamento de dados, Figura 4.2,cujo último estágio é 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 à reflexões nas superfícies dos cilindros presentes no ambiente. Para isso, observa-se 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 à reflexões dos cilindros se encontram entre uma

variação positiva seguida rapidamente de uma variação negativa na curva da derivada.

O próximo passo consiste em transformar o sinal para a representação em coordenadas cartesianas, e segmentar os conjuntos de pontos correspondentes aos intervalos entre picos e vales do sinal de derivada computado anteriormente, como é mostrado na Figura 4.4. Então, cada um desses conjuntos é fornecido como entrada do Algoritmo 10 (listado no Anexo C.1), o resultado são círculos que melhor explicam esses conjuntos de pontos (Figura 4.5), juntamente com o erro médio quadrático entre cada conjunto de pontos e seu círculo.

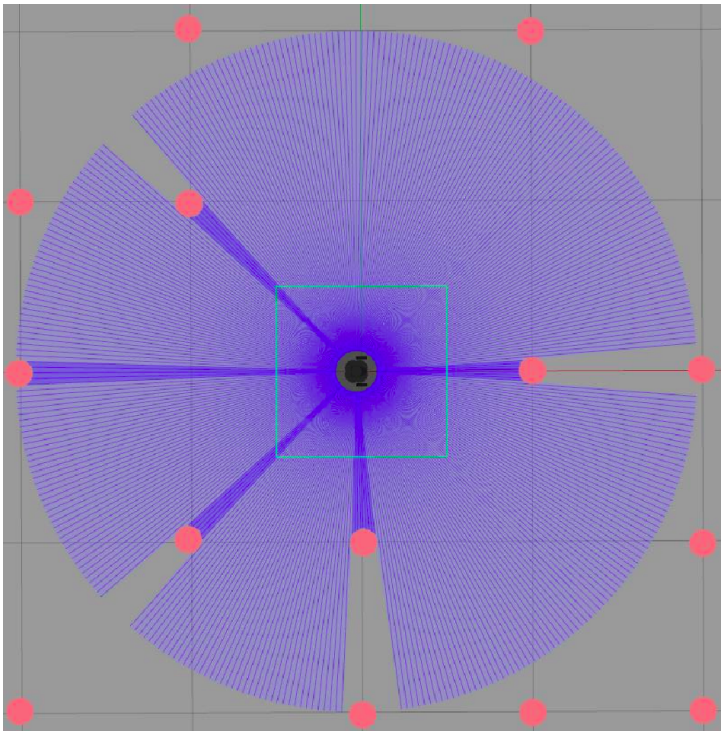
Por fim são considerados apenas os círculos cujo erro médio quadrático para seus pontos correspondentes é inferior a um limiar $\epsilon = 10^{-3}$ e cujo raio seja superior a 6cm. Essa última condição é especialmente útil no cenário multiagente, pois evita de que um robô confunda o sensor LiDAR (que fica montado no topo do robô, conforme visto na Figura 2.2) do outro com uma *landmark*.

4.2 Mapa em grade

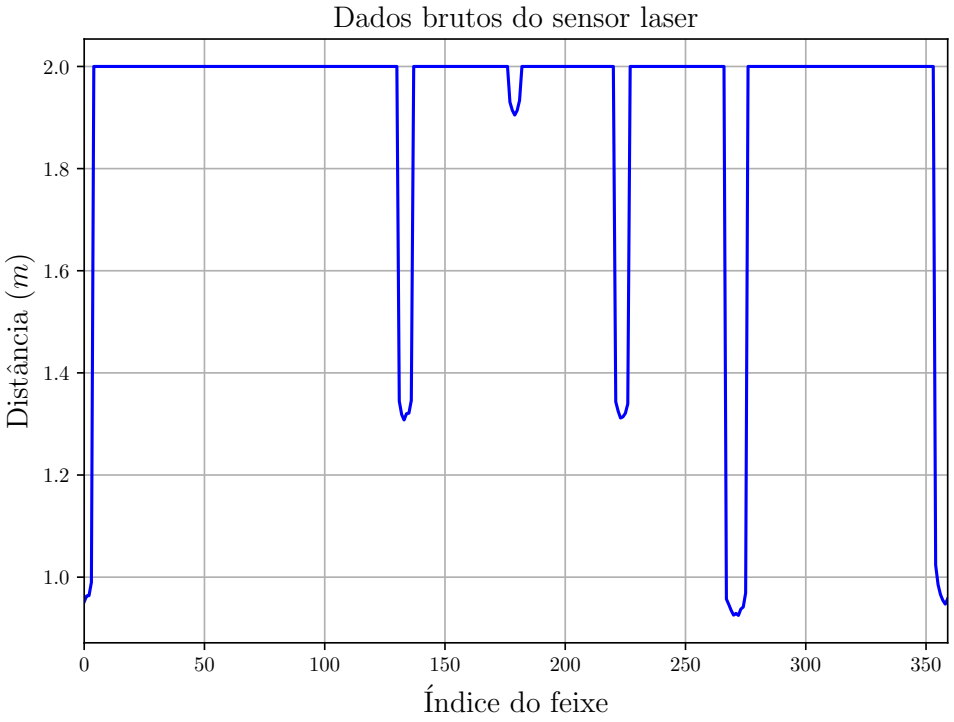
teste

4.3 Exploração Autônoma

teste



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

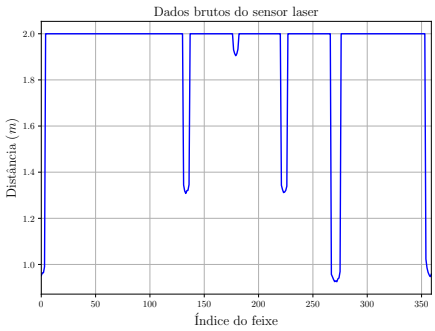


(b) Interpolação linear das distâncias lidas pelo sensor.

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

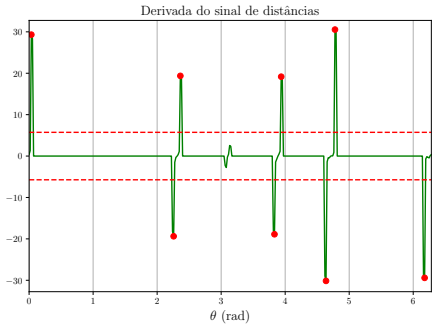
1

ENTRADA
Sequência de distâncias
lidas pelo sensor.



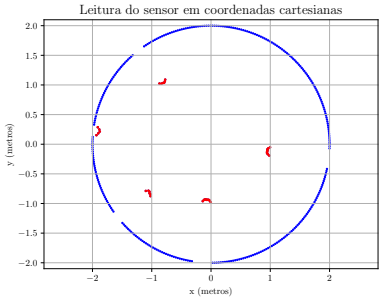
2

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. E transformação
para coordenadas cartesianas.



4

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

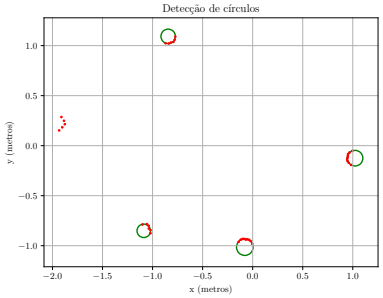


FIGURA 4.2 – Sequência de passos para processar os dados brutos do sensor LiDAR e transforma-los em dados úteis para o modelo de medida utilizado.

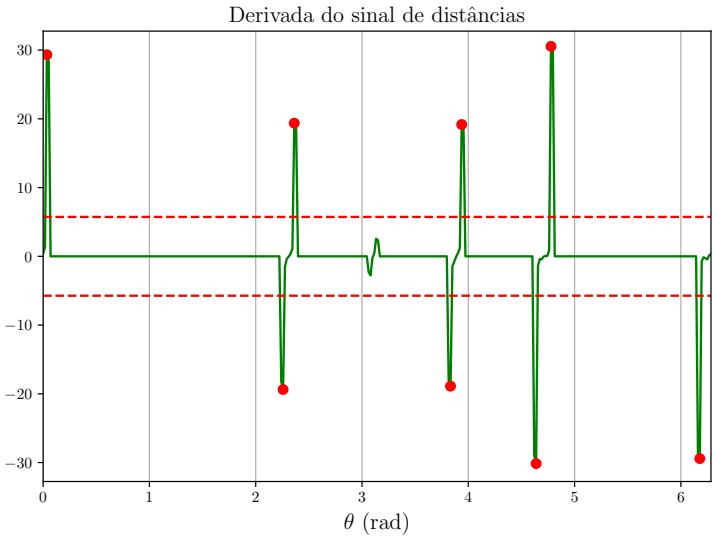


FIGURA 4.3 – Derivada central da sequência de distâncias representadas na Figura 4.1b. As linhas pontilhadas em vermelho representam os limiares a partir dos quais os picos são interpretados como início ou fim da superfície de um cilindro. Os picos destacados em vermelho ultrapassam as valores limiares.

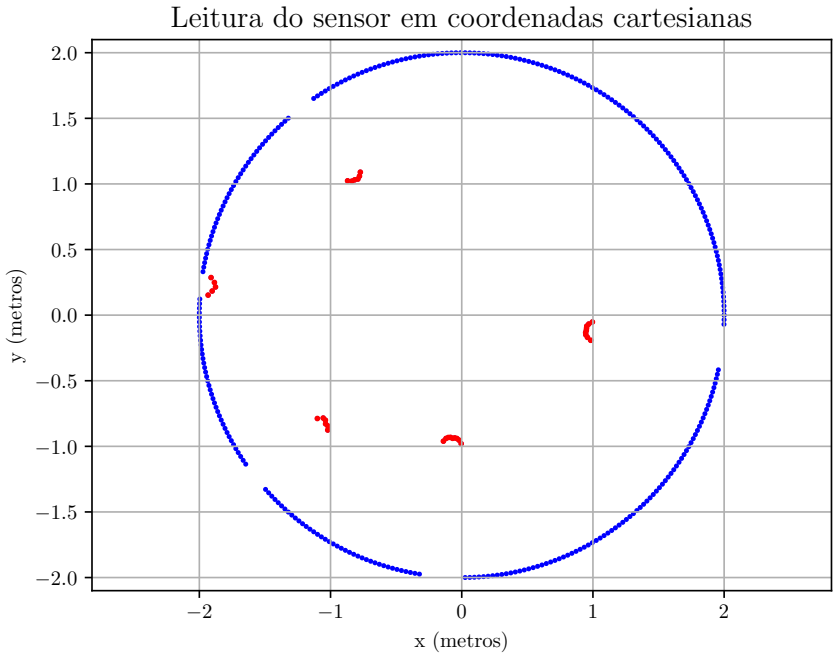


FIGURA 4.4 – Representação da leitura do sensor LiDAR em coordenadas cartesianas. Os pontos destacados em vermelho correspondem a reflexões das superfícies das *landmarks*.

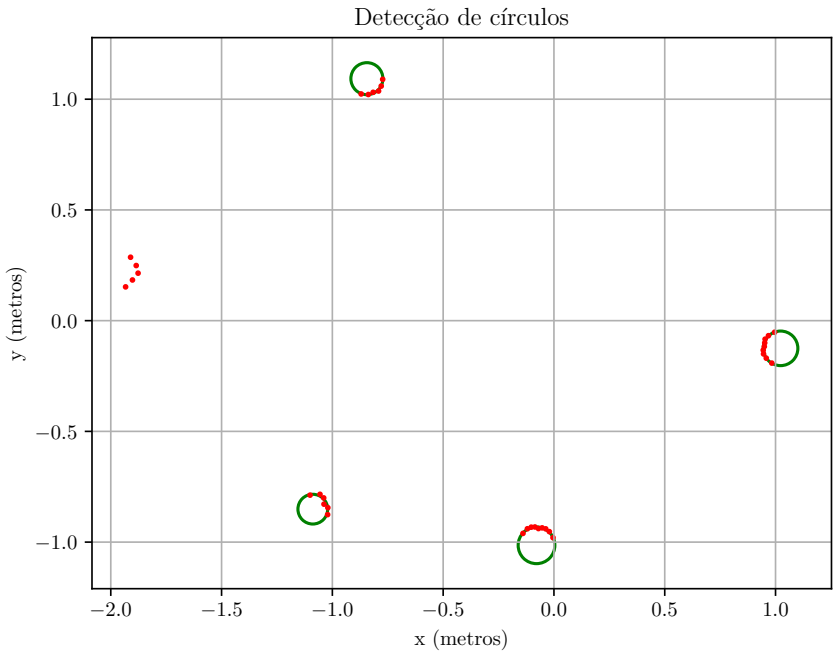


FIGURA 4.5 – Em vermelho, os conjuntos de pontos selecionados como pertencentes à superfícies das *landmarks*. Em verde, os círculos estimados para cada conjunto.

5 SLAM multiagente descentralizado

5.1 Cálculo da posição relativa entre agentes

5.2 Troca de Mapas

6 Conclusão

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.

BONGARD, J. **Probabilistic robotics. sebastian thrun, wolfram burgard, and dieter fox.(2006, mit press.) 647 pages**. [*S.l.*]: MIT Press, 2006.

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.

COPPERSMITH, D.; WINOGRAD, S. Matrix multiplication via arithmetic progressions. *In*: **Proceedings of the nineteenth annual ACM symposium on Theory of computing**. **Proceedings** [...]. [*S.l.: s.n.*], 1987. p. 1–6.

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

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

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. *In*: **IEEE/RSJ International Conference on Intelligent Robots and Systems**. **Proceedings** [...]. Sendai, Japan: [*s.n.*], 2004. p. 2149–2154.

LEWIS, F. L.; XIE, L.; POPA, D. **Optimal and robust estimation: with an introduction to stochastic control theory**. [*S.l.*]: CRC press, 2017.

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

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

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**. **Proceedings** [...]. [*S.l.: s.n.*], 2009. v. 3, n. 3.2, p. 5.

ROBOTIS. **TurtleBot 3**. 2021. Available at: https://github.com/ROBOTIS-GIT/turtlebot3.

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.

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

SOLÀ, J. **Simulataneous localization and mapping with the extended Kalman filter. A very quick guide... with Matlab code!** 2014. Available at: https://www.iri.upc.edu-/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf. Accessed on: 02/05/2022.

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 6, 7 e 8, respectivamente. Por fim, o Algoritmo 9 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 6 quanto no 7.

Algorithm 6 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**

Algorithm 7 Etapa de atualização do EKF-SLAM

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

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

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

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

$\boldsymbol{\mu}_t \leftarrow \mathbf{Kz}$

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

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

end function

Algorithm 8 Etapa de inserção de nova *landmark* do EKF-SLAM

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

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

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

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

end function

Algorithm 9 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

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 (BONGARD, 2006, 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:*

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

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:

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

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 (BONGARD, 2006, 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 ajuste de círculos

O Algoritmo 10 é uma adaptação das notas de aula do professor Matthew L. Elwin da disciplina ME495 (*Sensing, Navigation, and Machine Learning for Robotics*) ministrada na Universidade Northwestern.

Algorithm 10

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	25 de março de 2015	DCTA/ITA/DM-018/2015	75

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:

Cupim; Cimento; Estruturas

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Cupim; Dilema; Construção

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

ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Aeronáutica e Mecânica. Área de Sistemas Aeroespaciais e Mecatrônica. Orientador: Prof. Dr. Adalberto Santos Dupont. Coorientadora: Profª. Drª. Doralice Serra. Defesa em 05/03/2015. Publicada em 25/03/2015.

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 de 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 desenvolve uma solução multiagente em ambiente simulado para o problema SLAM 2D. Para isso emprega o uso do Filtro de Informação Esparso, juntamente com outros algoritmos de navegação, associação de dados e de representação de mapas. As vantagens da solução distribuída do problema de SLAM, em relação ao problema original, é a divisão da carga de trabalho entre os agentes e a redundância de informação.

Atualmente cada agente é capaz de performar SLAM individualmente, o desenvolvimento se encontra na etapa de troca do mapa de *features* entre os agentes. Subsequente a essa etapa, será abordado o problema da exploração conjunta entre os agentes. Terminadas essas duas etapas de desenvolvimento, fechando o escopo delineado para o trabalho, se dará início à escrita da dissertação e também de artigo para submissão a coreferências e/ou periódicos.

12. GRAU DE SIGILO:

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