

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. Dr. John von Neumann

Pró-Reitor de Pós-Graduação

Campo Montenegro
São José dos Campos, SP - Brasil
2015

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, 2015.

53f.

Dissertação de Mestrado – Curso de Engenharia Eletrônica e Computação. Área de Sistemas e
Controle – Instituto Tecnológico de Aeronáutica, 2015. 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**. 2015. 53f. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Wellington Vieira Martins de Castro

TÍTULO DO TRABALHO: SLAM distribuído envolvendo navegação, guiamento e fusão sensorial para reconstrução 2D.

TIPO DO TRABALHO/ANO: Dissertação / 2015

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias desta dissertação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação pode ser reproduzida sem a autorização do autor.

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 reivindicar um pouco mais de terra firme.”

— T. H. HUXLEY

Resumo

Abstract

Lista de Figuras

FIGURA 2.1 – Diferentes vistas do ambiente simulado	21
FIGURA 2.2 – 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. . . .	22
FIGURA 2.3 – Esquemático do modelo de medida <i>Range-Bearing</i> . 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 <i>landmark</i> no mapa.	24
FIGURA 2.4 – Esquemático do modelo de medida <i>Range-Bearing</i> . O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. Os círculos representam <i>landmarks</i> , as conhecidas pelo robô (portanto presentes no vetor de estados), em cinza, e recém descobertas, em laranja. A p -ésima <i>landmark</i> acaba de ser encontrada pelo robô.	26
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 μ_R e sua autocovariância \mathbf{P}_{RR} (cinza escuro), e às covariâncias cruzadas, \mathbf{P}_{RM} e \mathbf{P}_{MR} , entre o robô e o mapa (cinza claro). Note que as partes correspondentes ao mapa, μ_M e \mathbf{P}_{MM} , permanecem inalteradas (branco). Adaptado de (SOLÀ, 2014, p. 10).	30

- 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 μ_R e à posição da *landmark* \mathbf{m}^j , e suas autocovariâncias \mathbf{P}_{RR} e $\mathbf{P}_{M^j M^j}$ (cinza escuro), e às covariâncias cruzadas, \mathbf{P}_{RM^j} e $\mathbf{P}_{M^j R}$, entre o robô e a *j*-ésima *landmark* (cinza claro). Adaptado de (SOLÀ, 2014, p. 8). 32
- 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). 33
- 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). 34
- FIGURA 3.5 – Representação das matrizes $\Psi_t, \lambda_t, \kappa_t$, 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. 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. 39
- FIGURA 4.1 – Visualização dos feixes laser emitidos pelo sensor LiDAR e a respectiva leitura gerada. 44

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 Espaço
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging

Lista de Símbolos

\mathbf{x}_t	Pose do robô no instante t
$\mathbf{x}_{0:t}$	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 do 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 <i>landmarks</i>
$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0)$	Distribuição do problema online SLAM
$p(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0)$	Distribuição do problema full SLAM
$\boldsymbol{\mu}_t$	Vetor média no instante t
$\bar{\boldsymbol{\mu}}_t$	Predição do vetor média no instante t
\mathbf{P}_t	Matriz de covariância no tempo t
$\bar{\mathbf{P}}_t$	Predição da matriz de covariância no tempo t
\mathbf{K}_t	Ganho de Kalman no tempo t
$g_R(\bullet, \bullet)$	Modelo de movimento do robô
$h(\bullet, \bullet)$	Modelo de medida do sensor laser
$f(\bullet, \bullet)$	Modelo de medida inverso do sensor laser
$\mathbb{E}[\bullet]$	Operador esperança
$\mathbf{0}_{m \times n}$	Matriz nula de dimensão $m \times n$
\mathbf{I}_n	Matriz identidade de dimensão n
$Cov(\bullet)$	Covariância

Sumário

1	INTRODUÇÃO	16
1.1	Objetivo	19
1.2	Motivação	20
1.3	Estrutura de um sistema SLAM	20
1.4	Organização do trabalho	20
2	VISÃO GERAL DO SISTEMA	21
2.1	O ambiente	21
2.2	Modelo do robô	21
2.3	Medidas e o modelo de medida <i>Range-Bearing</i>	23
2.3.1	Modelo de medida <i>Range-Bearing</i>	23
2.3.2	Modelo de medida inverso <i>Range-Bearing</i>	25
2.4	Simulação	26
2.5	Visualização	26
3	APLICAÇÃO DA ESTIMAÇÃO NO PROBLEMA SLAM (<i>Backend</i>) .	27
3.1	Filtro de Kalman Extendido	27
3.2	EKF-SLAM	28
3.2.1	EKF-SLAM: Predição (Movimento do robô)	29
3.2.2	EKF-SLAM: Atualização	30
3.2.3	EKF-SLAM: Inserção de <i>landmark</i> (aumento do vetor de estados) . .	32
3.3	Filtro de Informação Extendido (EIF)	34
3.4	SEIF-SLAM	35
3.4.1	SEIF-SLAM: Landmarks ativas e passivas	36

3.4.2	SEIF-SLAM: Passo de predição	36
3.4.3	SEIF-SLAM: Passo de atualização	40
3.4.4	SEIF-SLAM: Inserção de nova <i>landmark</i>	40
3.4.5	SEIF-SLAM: Esparsificação da matriz de informação	40
3.4.6	SEIF-SLAM: Recuperação da média	40
3.5	Associação de <i>landmarks</i>	41
3.6	Conclusão do capítulo	41
4	SLAM <i>Frontend</i>	42
4.1	Dados do sensor laser	42
4.1.1	Dados brutos	42
4.1.2	Processamento de dados	42
4.2	Mapa em grade	43
4.3	Exploração Autônoma	43
5	SLAM MULTIAGENTE DESCENTRALIZADO	46
5.1	Exploração distribuída	46
5.2	Cálculo da posição relativa entre agentes	46
5.3	Troca de Mapas	46
6	CONCLUSÃO	47
	REFERÊNCIAS	48
	APÊNDICE A – DESCRIÇÃO DETALHADA DE ALGORITMOS	49
A.1	Algoritmo EKF-SLAM	49
	APÊNDICE B – MATRIZES	52
B.1	Lema da Inversão. Fórmula de Sherman/Morrison	52

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.

Além disso, 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 : o vetor de estados descrevendo a pose do robô no instante t
- $\mathbf{x}_{1:t} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t\}$: histórico de poses do robô até o instante t
- \mathbf{u}_t : o vetor de controle executado pelo robô no instante t
- $\mathbf{u}_{1:t} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{t-1}, \mathbf{u}_t\}$: histórico de controles executados pelo robô até o instante t

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

$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0) \quad (1.1)$$

$$p(\mathbf{x}_{1:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0) \quad (1.2)$$

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 . 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, pertinente a este trabalho, do problema de SLAM é abordada a seguir.

Taxonomia do problema SLAM

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

Como mencionado anteriormente, há duas classes de problemas de SLAM: *online* SLAM e *full* SLAM, e, portanto, duas classes de algoritmos para resolvê-los. Os algoritmos *full* SLAM também são chamados de *offline* SLAM por serem normalmente utilizados

em etapas de pós processamento, como refinamento de mapas. Esses algoritmos exigem mais recursos, tanto de processamento quanto de memória para serem processados. Portanto, há grande dificuldade para utiliza-los embarcados nos agentes durante a etapa de exploração do ambiente.

Em 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 alimentá-lo, um robô é capaz de realizar SLAM como foi apresentado até agora. Porém, ao mapear o ambiente, o robô pode explorá-lo de maneira autônoma, ou, quando o cenário permite, ser controlado remotamente por um operador. Em cenários como a exploração de Marte tal controle é inviável. 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 (KONIG; 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.

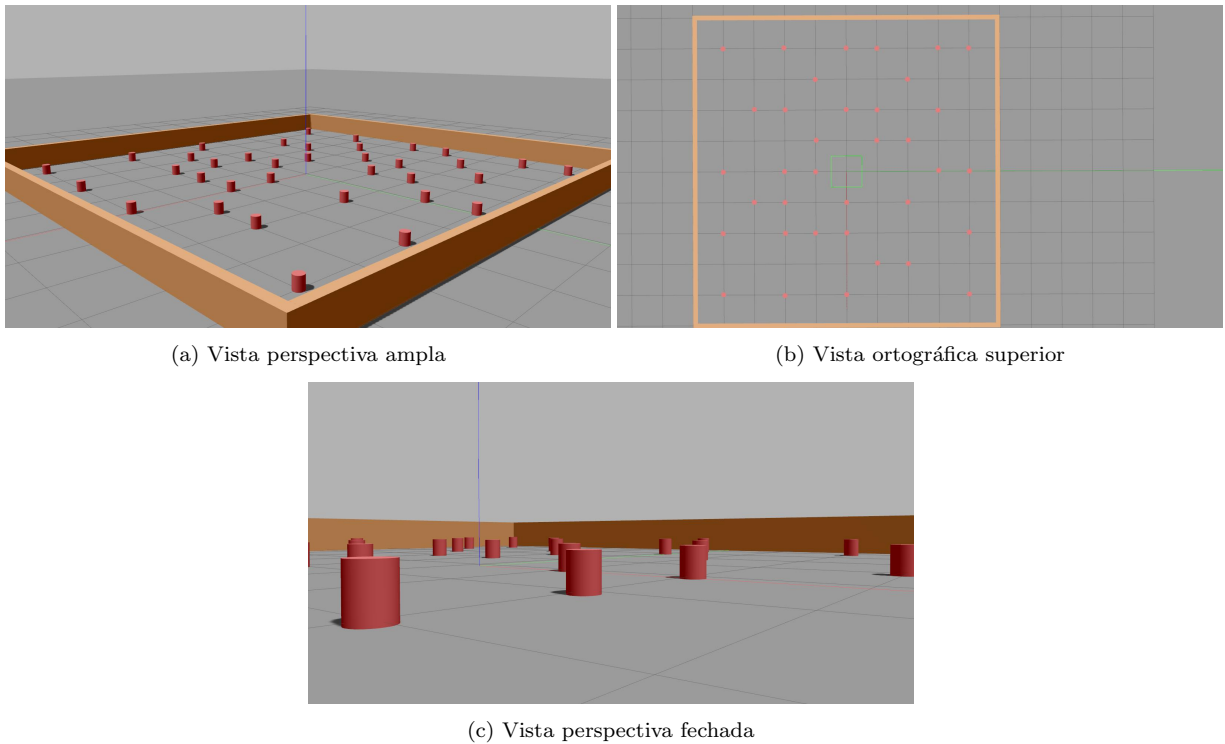


FIGURA 2.1 – Diferentes vistas do ambiente simulado

2.2 Modelo do 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. Na Figura 2.2 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:

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

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.

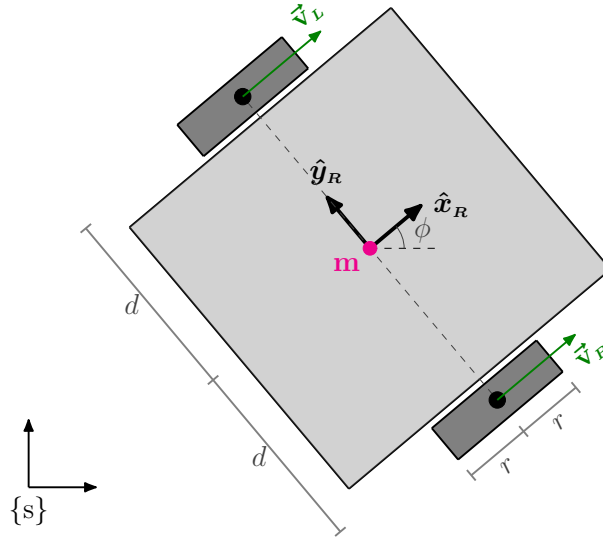


FIGURA 2.2 – Visão superior do esquemático de um robô diferencial. O eixo z está apontando para fora da folha. O sistema de coordenadas móvel do robô está posicionado no ponto médio do eixo das rodas (cinza escuro). Na imagem $\{s\}$ é um sistema de coordenadas estático.

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 $[t-1, t]$.

$$\begin{aligned} \mathbf{x}_t &= \mathbf{g}_R(\mathbf{x}_{t-1}, \mathbf{u}_t) \\ &= \begin{bmatrix} \phi_{t-1} \\ x_{t-1} \\ y_{t-1} \end{bmatrix} + \begin{cases} \begin{bmatrix} \alpha_t \\ d \cdot \frac{u_t^R + u_t^L}{u_t^R - u_t^L} (\sin(\phi_{t-1} + \alpha_t) - \sin(\phi_{t-1})) \\ d \cdot \frac{u_t^R + u_t^L}{u_t^R - u_t^L} (-\cos(\phi_{t-1} + \alpha_t) + \cos(\phi_{t-1})) \end{bmatrix}, & \text{Se } u_t^L \neq u_t^R \\ \begin{bmatrix} 0 \\ r \cdot u_t^R \cos(\phi_{t-1}) \\ r \cdot u_t^R \sin(\phi_{t-1}) \end{bmatrix}, & \text{Caso contrário} \end{cases} \quad (2.2) \end{aligned}$$

Onde:

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

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

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

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 2.3, 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:

$$\mathbf{x} = [\phi \ x \ y \ m_x^i \ m_y^i \ m_x^j \ m_y^j \ m_x^k \ m_y^k]^T \quad (2.5)$$

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

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

onde

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

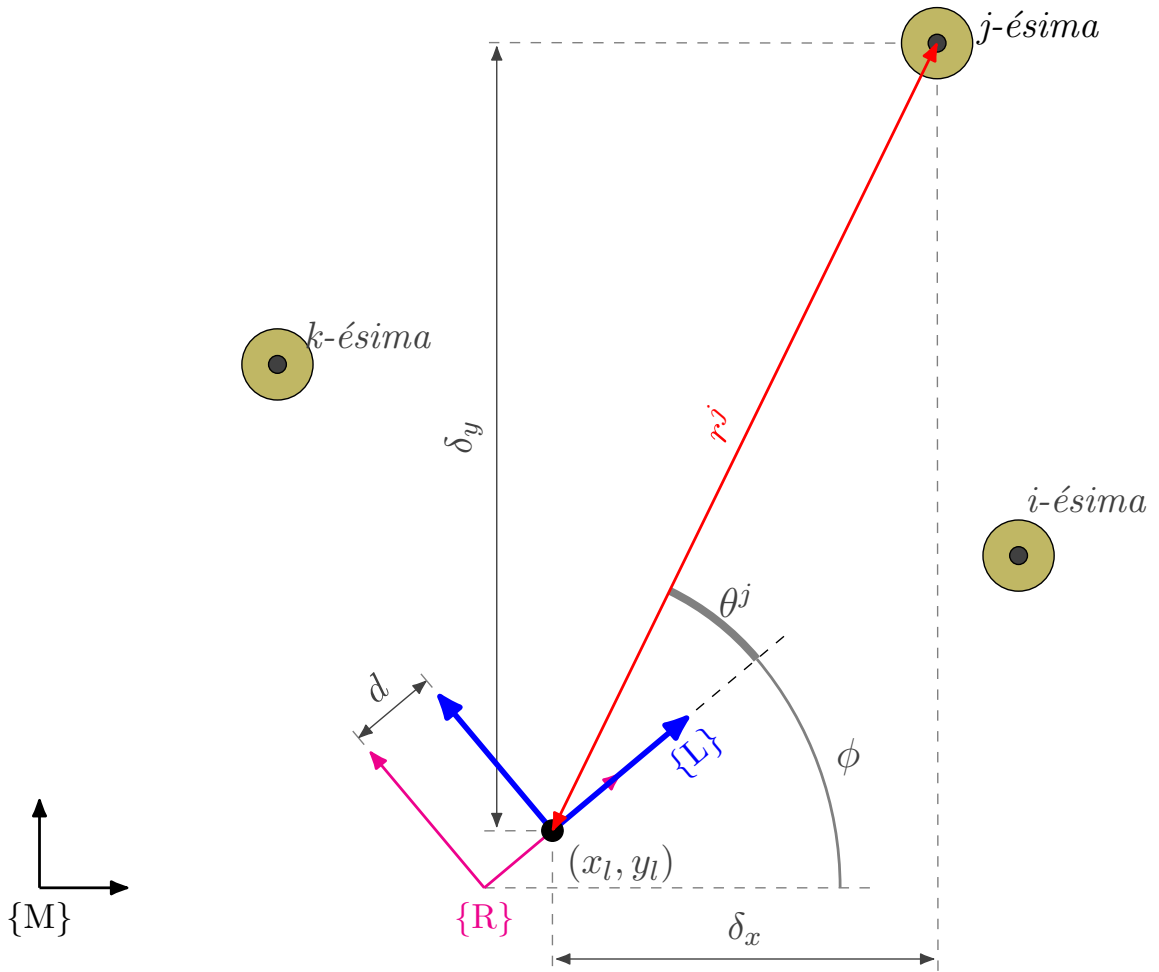


FIGURA 2.3 – Esquemático do modelo de medida *Range-Bearing*. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. A medida (r^j, θ^j) se refere à i -ésima *landmark* no mapa.

Como o modelo de medida em 2.6 é não linear, é necessário linearizá-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}_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.

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

$$\mathbf{H}_M^j = \begin{bmatrix} \frac{\delta_x}{r^j} & \frac{\delta_y}{r^j} \\ \frac{-\delta_y}{[r^j]^2} & \frac{\delta_x}{[r^j]^2} \end{bmatrix} \quad (2.9)$$

$$\mathbf{H}^j(\mathbf{x}) = \begin{bmatrix} \mathbf{H}_R^j & \mathbf{0} & \cdots & \mathbf{H}_M^j & \cdots & \mathbf{0} \end{bmatrix} \quad (2.10)$$

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

O modelo de medida inverso $\mathbf{f}(\bullet, \bullet)$ 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.

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

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

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

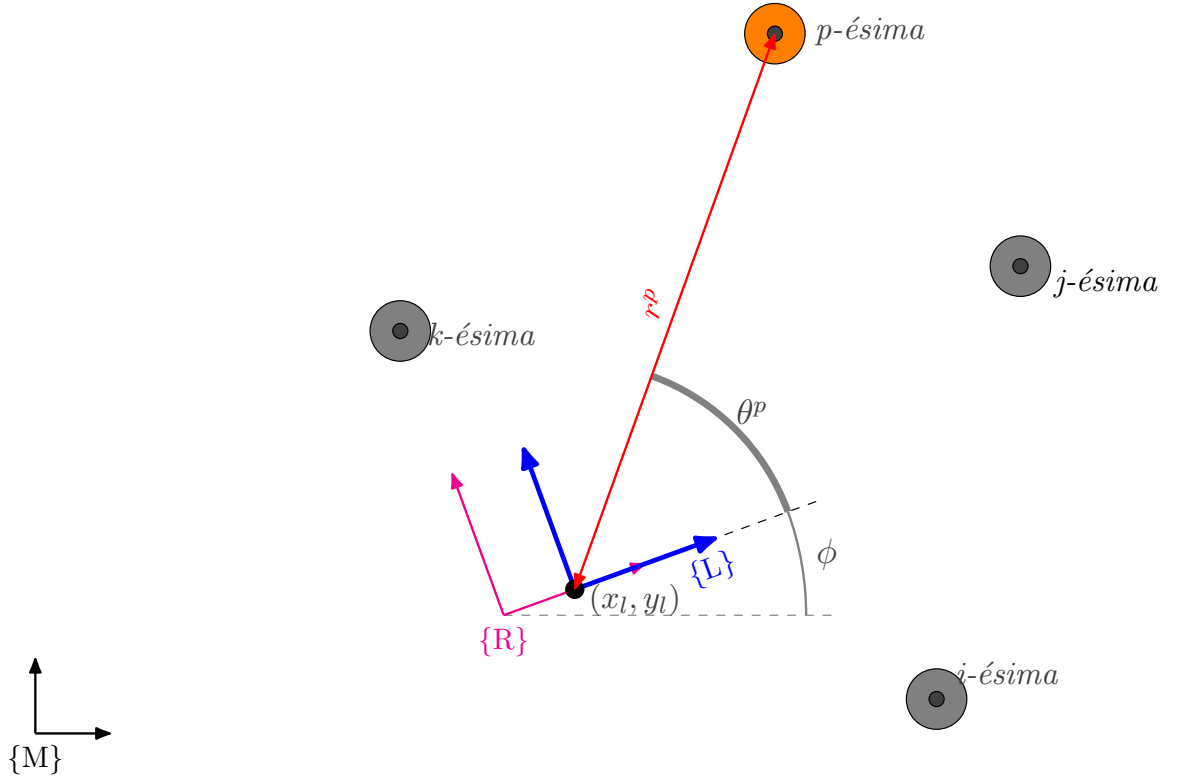


FIGURA 2.4 – Esquemático do modelo de medida *Range-Bearing*. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. Os círculos representam *landmarks*, as conhecidas pelo robô (portanto presentes no vetor de estados), em cinza, e recém descobertas, em laranja. A p -ésima *landmark* acaba de ser encontrada pelo robô.

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_X & \mathbf{F}_Y \end{bmatrix} \quad (2.14)$$

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, 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.

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. 3.1; a segunda, chamada de modelo de medida, relaciona o estado do sistema com a medida esperada,

gerada pelo sensor, Eq. 3.2.

$$\mathbf{x}_t = \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \quad (3.1)$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \boldsymbol{\delta}_t \quad (3.2)$$

Como o modelo do sistema $\mathbf{g}(\bullet, \bullet)$, e o modelo de medida $\mathbf{h}(\bullet, \bullet)$ 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 $\mathbf{g}(\bullet, \bullet)$ e/ou $\mathbf{h}(\bullet, \bullet)$ não são lineares, o EKF pode ser utilizado para estimar o estado do sistema.

As Equações de 3.3 até 3.9 definem o EKF¹ 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 $\bar{\boldsymbol{\mu}}_t$.

$$\bar{\boldsymbol{\mu}}_t = \mathbf{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) \quad (3.3)$$

$$\bar{\mathbf{P}}_t = \mathbf{G}_t \mathbf{P}_{t-1} \mathbf{G}_t^T + \mathbf{R}_t \quad (3.4)$$

$$\mathbf{z}_t = \mathbf{y}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t) \quad (3.5)$$

$$\mathbf{Z}_t = \mathbf{H}_t \bar{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{Q}_t \quad (3.6)$$

$$\mathbf{K}_t = \bar{\mathbf{P}}_t \mathbf{H}_t^T \mathbf{Z}_t^{-1} \quad (3.7)$$

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t \mathbf{z}_t \quad (3.8)$$

$$\mathbf{P}_t = \bar{\mathbf{P}}_t - \mathbf{K}_t \mathbf{Z}_t \mathbf{K}_t^T \quad (3.9)$$

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 por ambos.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_M \end{bmatrix} \quad (3.10)$$

¹O leitor pode estranhar a Equação 3.9 do erro da estimativa. Normalmente ela é escrita na forma $\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\mathbf{P}}_t$, porém de acordo com (LEWIS *et al.*, 2017, p. 73), a forma em 3.9 é uma alternativa melhor na presença de erros de arredondamento, e é frequentemente utilizada em implementações de software.

O vetor \mathbf{x}_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*.

$$\mathbf{x}_M = \begin{bmatrix} m_x^1 \\ m_y^1 \\ \vdots \\ m_x^n \\ m_y^n \end{bmatrix} \quad (3.11)$$

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}_R é alterada pela entrada \mathbf{u} , logo o modelo do sistema consiste apenas no modelo de movimento do robô \mathbf{g}_R concatenado com as posições das *landmarks*:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{g}_R(\mathbf{x}_{R,t}, \mathbf{u}_t) \\ \mathbf{x}_M \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_{R,t} \\ \mathbf{0} \end{bmatrix} \quad (3.12)$$

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

$$\bar{\boldsymbol{\mu}}_t = \begin{bmatrix} \mathbf{g}_R(\boldsymbol{\mu}_{R,t-1}, \mathbf{u}_t) \\ \boldsymbol{\mu}_M \end{bmatrix} \quad (3.13)$$

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 3.12 possui forma esparsa:

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (3.14)$$

Então, a Equação do erro de predição do EKF, em 3.4, torna-se:

$$\begin{aligned}
 \bar{\mathbf{P}}_t &= \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{P}_{t-1} \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{R}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\
 &= \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{RR,t-1} & \mathbf{P}_{RM,t-1} \\ \mathbf{P}_{MR,t-1} & \mathbf{P}_{MM} \end{bmatrix} \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{R}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (3.15) \\
 &= \begin{bmatrix} \mathbf{G}_{R,t} \mathbf{P}_{RR,t-1} \mathbf{G}_{R,t}^T + \mathbf{R}_{R,t} & \mathbf{G}_{R,t} \mathbf{P}_{RM,t-1} \\ \mathbf{P}_{MR,t-1} \mathbf{G}_{R,t}^T & \mathbf{P}_{MM} \end{bmatrix}
 \end{aligned}$$

A complexidade dessa operação é da ordem de $\mathcal{O}(n)$ por conta do termo $\mathbf{G}_{R,t} \mathbf{P}_{RM,t-1}$, 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.

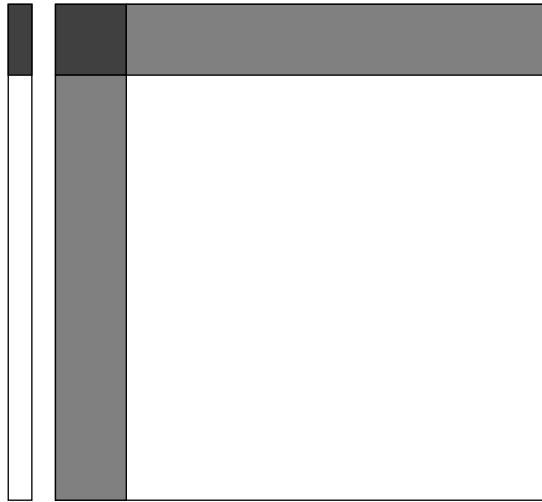


FIGURA 3.1 – Partes modificadas do vetor média e da matriz de covariância durante o movimento do robô. O vetor média é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As partes modificadas, em tons de cinza, correspondem ao estado do robô $\boldsymbol{\mu}_R$ e sua autocovariância \mathbf{P}_{RR} (cinza escuro), e às covariâncias cruzadas, \mathbf{P}_{RM} e \mathbf{P}_{MR} , entre o robô e o mapa (cinza claro). Note que as partes correspondentes ao mapa, $\boldsymbol{\mu}_M$ e \mathbf{P}_{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 .

$$\mathbf{z}_t^j = \mathbf{y}_t^j - \mathbf{h}^j(\bar{\boldsymbol{\mu}}_t) \quad (3.16)$$

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:

$$\mathbf{Z}_t^j = \begin{bmatrix} \mathbf{H}_R^j & \mathbf{H}_M^j \end{bmatrix} \begin{bmatrix} \bar{\mathbf{P}}_{RR} & \bar{\mathbf{P}}_{RM_j} \\ \bar{\mathbf{P}}_{RM_j}^T & \bar{\mathbf{P}}_{M_j M_j} \end{bmatrix} \begin{bmatrix} \mathbf{H}_R^j \\ \mathbf{H}_M^j \end{bmatrix} + \mathbf{Q}_t \quad (3.17)$$

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 3.17, 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 3.8 e 3.9, 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 3.8 e 3.9, 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.

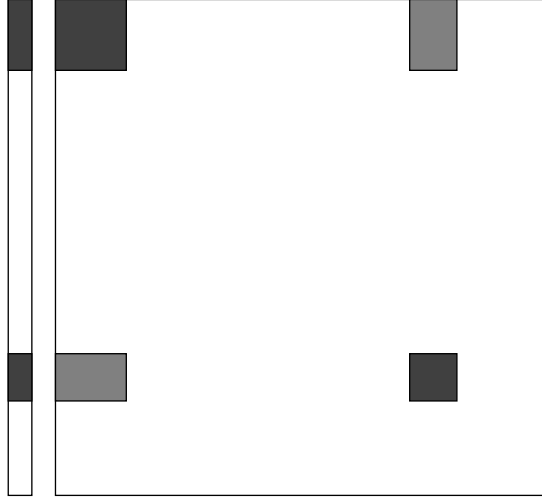


FIGURA 3.2 – Partes utilizadas do vetor média e da matriz de covariância durante o cálculo da inovação, quando uma *landmark* é observada. O vetor média é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As porções utilizadas, em tons de cinza, correspondem ao estado do robô μ_R e à posição da *landmark* \mathbf{m}^j , e suas autocovariâncias \mathbf{P}_{RR} e $\mathbf{P}_{M^j M^j}$ (cinza escuro), e às covariâncias cruzadas, \mathbf{P}_{RM^j} e $\mathbf{P}_{M^j R}$, 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}_M , e portanto não é possível calcular a inovação na Equação 3.16. 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 $\sigma(\bullet, \bullet)$, 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 .

$$\sigma(\mathbf{x}_t, \mathbf{y}^j) = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{f}(\mathbf{x}_t, \mathbf{y}^j) \end{bmatrix} \quad (3.18)$$

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 medição do sensor.

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

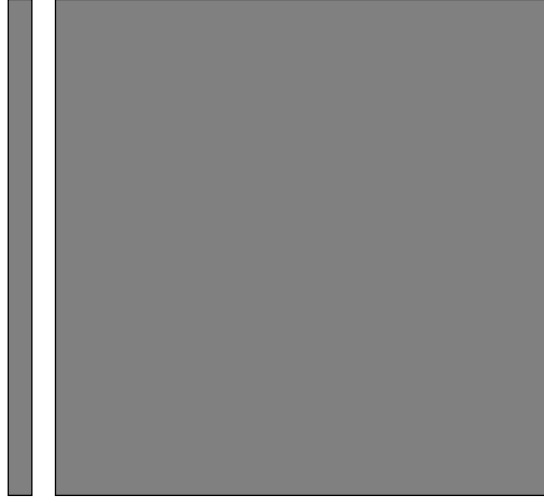


FIGURA 3.3 – O vetor média e a matriz de covariâncias são completamente atualizados durante a observação de uma *landmark*. Retirado de (SOLÀ, 2014, p. 8).

indicado em (BONGARD, 2006, p. 317), pode ser injusto. Portanto devemos calcular o erro, α , da nova estimativa do vetor aumentado, de maneira análoga à forma como é feita nos passos de predição e atualização do EKF.

$$\begin{aligned}
 \alpha &= \mathbf{x}_t^* - \mu_t^* \\
 &= \begin{bmatrix} \mathbf{x}_t \\ \mathbf{f}(\mathbf{x}_t, \mathbf{y}) \end{bmatrix} - \begin{bmatrix} \mu_t \\ \mathbf{f}(\mu_t, \mathbf{y}^j) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_t - \mu_t \\ \mathbf{f}(\mathbf{x}_t, \mathbf{y}) - \mathbf{f}(\mu_t, \mathbf{y}^j) \end{bmatrix} \\
 &= \begin{bmatrix} \eta_t \\ \cancel{\mathbf{f}(\mu_t, \mathbf{y}^j)} + \mathbf{F}_X \eta_t + \mathbf{F}_Y \delta - \cancel{\mathbf{f}(\mu_t, \mathbf{y}^j)} \end{bmatrix} \\
 &= \begin{bmatrix} \eta_t \\ \mathbf{F}_X \eta_t + \mathbf{F}_Y \delta \end{bmatrix} \quad \text{Onde } \eta_t = \mathbf{x}_t - \mu_t \text{ e } \delta = \mathbf{y} - \mathbf{y}^j
 \end{aligned} \tag{3.19}$$

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

$$\begin{aligned}
 \mathbf{P}_t^* &= \mathbb{E} [\alpha \alpha^T] \\
 &= \begin{bmatrix} \mathbf{P}_t & \mathbf{P}_t \mathbf{F}_X^T \\ \mathbf{P}_t \mathbf{F}_X & \mathbf{F}_X \mathbf{P}_t \mathbf{F}_X^T + \mathbf{F}_Y \mathbf{Q} \mathbf{F}_Y^T \end{bmatrix}
 \end{aligned} \tag{3.20}$$

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.

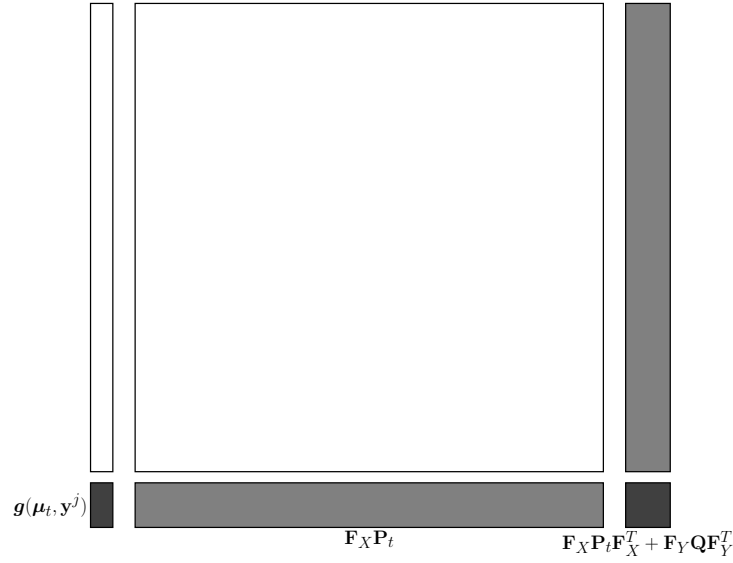


FIGURA 3.4 – Vetor média e matriz de covariância aumentados após inserção de nova *landmark*. As partes adicionadas, em cinza, correspondem às covariâncias cruzadas entre a nova *landmark* e o vetor de estados anterior (cinza claro), e à média da nova *landmark* e sua covariância (cinza escuro). Adaptado 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 para representar a distribuição de probabilidade gaussiana. Já o Filtro de Informação utiliza a chamada representação canônica, composta pelo vetor de informação, ξ , e pela matriz de informação, Ω . Definidos a seguir:

$$\xi = \mathbf{P}^{-1} \mu \quad (3.21)$$

$$\Omega = \mathbf{P}^{-1} \quad (3.22)$$

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

$$\mu_{t-1} = \Omega_{t-1}^{-1} \xi_{t-1} \quad (3.23)$$

$$\bar{\Omega}_t = (\mathbf{G}_t \Omega_{t-1}^{-1} \mathbf{G}_t^T + \mathbf{R}_t)^{-1} \quad (3.24)$$

$$\bar{\mu}_t = g(\mu_{t-1}, \mathbf{u}_t) \quad (3.25)$$

$$\bar{\xi}_t = \bar{\Omega}_t \bar{\mu}_t \quad (3.26)$$

$$\Omega_t = \bar{\Omega}_t + \mathbf{H}_t^T \mathbf{Q}_t^{-1} \mathbf{H}_t \quad (3.27)$$

$$\xi_t = \bar{\xi}_t + \mathbf{H}_t^T \mathbf{Q}_t^{-1} [\mathbf{y}_t - h(\bar{\mu}_t) + \mathbf{H}_t \bar{\mu}_t] \quad (3.28)$$

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 $\mathbf{\Omega} = \mathbf{0}$, enquanto no KF é necessário utilizar valores muito grandes na matriz de covariância. Outro aspecto interessante do IF é sua naturalidade para sistemas multi robôs, 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 3.23 e, a predição da matriz de informação em 3.24, 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 Esperso (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 Esperso (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 3.23, 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 $Cov(\mathbf{m}^j, \mathbf{m}^k) \neq \mathbf{0}, \forall \{j, k\}$, 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 3.29. O conjunto \mathbf{m}_t^+ é composto pelas *landmarks* ativas, que estão “ligadas” ao robô no tempo t , ou seja, $Cov(\mathbf{x}_{R,t}, \mathbf{m}_t^+) \neq 0$. Já o conjunto \mathbf{m}_t^- é formado pelas *landmarks* passivas, que não estão correlacionadas com a pose atual do robô, ou seja, $Cov(\mathbf{x}_{R,t}, \mathbf{m}_t^-) = 0$.

$$\begin{cases} \mathbf{m}_t^+ \cup \mathbf{m}_t^- &= \mathbf{x}_M \\ \mathbf{m}_t^+ \cap \mathbf{m}_t^- &= \emptyset \end{cases} \quad (3.29)$$

Uma das consequências desse esquema, é que as *landmarks* não são globalmente correlacionadas entre si, como ocorre no EKF-SLAM e EIF-SLAM. Na verdade, aqui, elas são localmente correlacionadas com sua vizinhança. 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, 3.24, 3.26 e 3.25. 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 ela é esparsa.

TODO:

- nota sobre a cópia da matriz de informação
- nota sobre correção da componente de orientação

Algorithm 1 SEIF-SLAM passo de predição

```

1: function SEIF-SLAM-PREDICTION( $\xi_{t-1}, \mu_{t-1}, \Omega_{t-1}, \mathbf{u}_t$ )
2:    $\Psi_t \leftarrow \mathbf{M}_{x_r}^T (\mathbf{G}_{R,t}^{-1} - \mathbf{I}_3) \mathbf{M}_{x_r}$ 
3:    $\lambda_t \leftarrow \Psi_t^T \Omega_{t-1} + \Psi_t^T \Omega_{t-1} \Psi_t + \Omega_{t-1} \Psi_t$ 
4:    $\Phi_t \leftarrow \Omega_{t-1} + \lambda_t$ 
5:    $\kappa_t \leftarrow \Phi_t - \Phi_t \mathbf{M}_{x_r}^T (\mathbf{R}_{R,t}^{-1} + \mathbf{M}_{x_r} \Phi_t \mathbf{M}_{x_r}^T)^{-1} \mathbf{M}_{x_r} \Phi_t$ 
6:    $\bar{\Omega}_t \leftarrow \Phi_t - \kappa_t$ 
7:    $\bar{\xi}_t \leftarrow (\lambda_t - \kappa_t) \mu_{t-1} + \xi_{t-1} + \bar{\Omega}_t \mathbf{M}_{x_r}^T \delta_{r,t}$ 
8:    $\bar{\mu}_t \leftarrow \mu_{t-1} + \mathbf{M}_{x_r}^T \delta_{r,t}$ 
9:   return  $\bar{\xi}_t, \bar{\mu}_t, \bar{\Omega}_t$ 
10: end function
    
```

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

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (3.14 \text{ repetida})$$

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

$$\begin{aligned}
 \mathbf{R}_t &= \begin{bmatrix} \mathbf{R}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{bmatrix}^T \mathbf{R}_{R,t} \begin{bmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{bmatrix} \\
 &= \mathbf{M}_{x_r}^T \mathbf{R}_{R,t} \mathbf{M}_{x_r}
 \end{aligned} \quad (3.30)$$

3.4.2.1 Predição da Matriz de Informação

Primeiro, vamos reescrever a Equação 3.24 em termos de Φ_t e, $\mathbf{R}_{R,t}$:

$$\bar{\Omega}_t = (\Phi_t^{-1} + \mathbf{M}_{x_r}^T \mathbf{R}_{R,t} \mathbf{M}_{x_r})^{-1} \quad (3.31)$$

Onde:

$$\begin{aligned}\Phi_t &= (\mathbf{G}_t \Omega_{t-1}^{-1} \mathbf{G}_t^T)^{-1} \\ &= [\mathbf{G}_t^T]^{-1} \Omega_{t-1} [\mathbf{G}_t]^{-1}\end{aligned}\quad (3.32)$$

Aplicando o lema da inversão (Apêndice B.1) em 3.31, temos:

$$\begin{aligned}\bar{\Omega}_t &= \Phi_t - \Phi_t \mathbf{M}_{x_r}^T (\mathbf{R}_{R,t}^{-1} + \mathbf{M}_{x_r} \Phi_t \mathbf{M}_{x_r}^T)^{-1} \mathbf{M}_{x_r} \Phi_t \\ &= \Phi_t - \kappa_t\end{aligned}\quad (3.33)$$

Para calcularmos

$$\kappa_t = \Phi_t - \Phi_t \mathbf{M}_{x_r}^T (\mathbf{R}_{R,t}^{-1} + \mathbf{M}_{x_r} \Phi_t \mathbf{M}_{x_r}^T)^{-1} \mathbf{M}_{x_r} \Phi_t \quad (3.34)$$

em tempo constante, temos que calcular Φ_t em tempo constante a partir de Ω_{t-1} . Para isso, vamos representar \mathbf{G}_t como na Equação 3.14, e calcular sua inversa:

$$\begin{aligned}\mathbf{G}_t^{-1} &= \begin{bmatrix} \mathbf{G}_{R,t} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{G}_{R,t}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (\text{inversão de matriz bloco diagonal}) \\ &= \mathbf{I}_n + \begin{bmatrix} \mathbf{G}_{R,t}^{-1} - \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \mathbf{I}_n + \overbrace{\mathbf{M}_{x_r}^T (\mathbf{G}_{R,t}^{-1} - \mathbf{I}_3) \mathbf{M}_{x_r}}^{\Psi_t} \\ &= \mathbf{I} + \Psi_t\end{aligned}\quad (3.35)$$

Note que Ψ_t é uma matriz de dimensão n , onde apenas os elementos do bloco superior esquerdo 3×3 , são diferentes de zero. Usando a Equação 3.35 na Equação 3.32 temos:

$$\begin{aligned}\Phi_t &= [\mathbf{G}_t^T]^{-1} \Omega_{t-1} [\mathbf{G}_t]^{-1} \\ &= (\mathbf{I} + \Psi_t^T) \Omega_{t-1} (\mathbf{I} + \Psi_t) \\ &= \Omega_{t-1} + \underbrace{\Psi_t^T \Omega_{t-1} + \Psi_t^T \Omega_{t-1} \Psi_t + \Omega_{t-1} \Psi_t}_{\lambda_t} \\ &= \Omega_{t-1} + \lambda_t\end{aligned}\quad (3.36)$$

Como Ψ_t é esparsa e com quantidade de elementos não nulos constante, λ_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 Φ_t a partir de Ω_{t-1} é constante, pois resulta da subtração dos elementos não nulos de λ_t , de Ω_{t-1} . Na Figura 3.5 são representadas as matrizes Ψ_t , λ_t , κ_t obtidas durante a execução do SEIF-SLAM com tamanho do conjunto de *landmarks* ativas igual a dois.

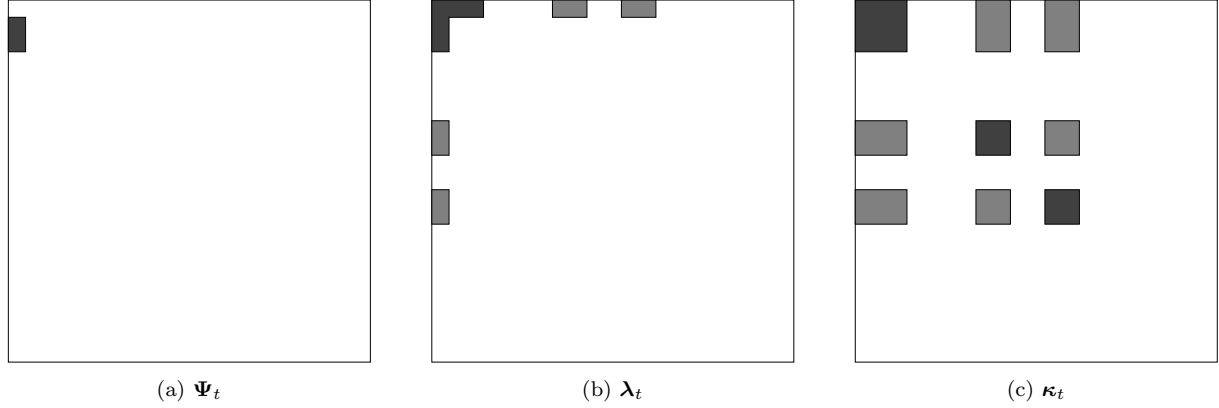


FIGURA 3.5 – Representação das matrizes Ψ_t , λ_t , κ_t , necessárias para calcular a matriz de informação predita durante o movimento do robô. Os elementos nulos são representados em branco, e os não nulos em cinza. As matrizes acima pertencem a um sistema SEIF-SLAM de um robô diferencial e sensor laser 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.

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 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 3.26, possa ser realizada em tempo constante no SEIF-SLAM. Primeiro, vamos reescrever o modelo de movimento na Equação Referênciaseq:motion-model, como:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \delta_t \quad (3.37)$$

Partindo da Equação 3.26 e utilizando 3.37 acima, temos:

$$\begin{aligned}
 \bar{\xi}_t &= \bar{\Omega}_t \bar{\mu}_t \\
 &= \bar{\Omega}_t (\mu_{t-1} + M_{x_r}^T \delta_{r,t}) \\
 &= \bar{\Omega}_t (\Omega_{t-1}^{-1} \xi_{t-1} + M_{x_r}^T \delta_{r,t}) \\
 &= \bar{\Omega}_t \Omega_{t-1}^{-1} \xi_{t-1} + \bar{\Omega}_t M_{x_r}^T \delta_{r,t} \\
 &= \left(\bar{\Omega}_t + \underbrace{\Omega_{t-1} - \Omega_{t-1}}_0 + \underbrace{\Phi_t - \Phi_t}_0 \right) \Omega_{t-1}^{-1} \xi_{t-1} + \bar{\Omega}_t M_{x_r}^T \delta_{r,t} \\
 &= \left(\underbrace{\bar{\Omega}_t - \Phi_t}_{-\kappa_t} + \Omega_{t-1} + \underbrace{\Phi_t - \Omega_{t-1}}_{\lambda_t} \right) \Omega_{t-1}^{-1} \xi_{t-1} + \bar{\Omega}_t M_{x_r}^T \delta_{r,t} \\
 &= (\lambda_t - \kappa_t + \Omega_{t-1}) \Omega_{t-1}^{-1} \xi_{t-1} + \bar{\Omega}_t M_{x_r}^T \delta_{r,t} \\
 &= (\lambda_t - \kappa_t) \Omega_{t-1}^{-1} \xi_{t-1} + \Omega_{t-1} \Omega_{t-1}^{-1} \xi_{t-1} + \bar{\Omega}_t M_{x_r}^T \delta_{r,t} \\
 &= (\lambda_t - \kappa_t) \mu_{t-1} + \xi_{t-1} + \bar{\Omega}_t M_{x_r}^T \delta_{r,t}
 \end{aligned} \tag{3.38}$$

Como λ_t e κ_t são ambas esparsas, o produto $(\lambda_t - \kappa_t) \mu_{t-1}$ contém um número determinado de elementos não nulos, e portanto é calculado em tempo constante. O produto $(\lambda_t - \kappa_t) \mu_{t-1}$ resulta em uma matriz nula exceto pelo primeiro bloco 3×3 , e ao multiplicá-la pela matriz de informação predita, que também é esparsa, temos como resultado um vetor esparsa (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.

3.4.3 SEIF-SLAM: Passo de atualização

3.4.4 SEIF-SLAM: Inserção de nova *landmark*

3.4.5 SEIF-SLAM: Esparsificação da matriz de informação

3.4.6 SEIF-SLAM: Recuperação da média

A recuperação da média, na Equação 3.23, implica na inversão da matriz de informação, e, mesmo numa matriz esparsa, essa operação não é constante. Portanto, no SEIF esse passo é executado de uma forma completamente diferente, ele é modelado como um problema de otimização.

...

3.5 Associação de *landmarks*

3.6 Conclusão do capítulo

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 $\{r^0, r^1, \dots, r^N\}$. 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 $k + 1$ e o feixe k .

A Figura 4.1b apresenta os dados brutos obtidos em uma leitura feita no ambiente mostrado na Figura 2.1.

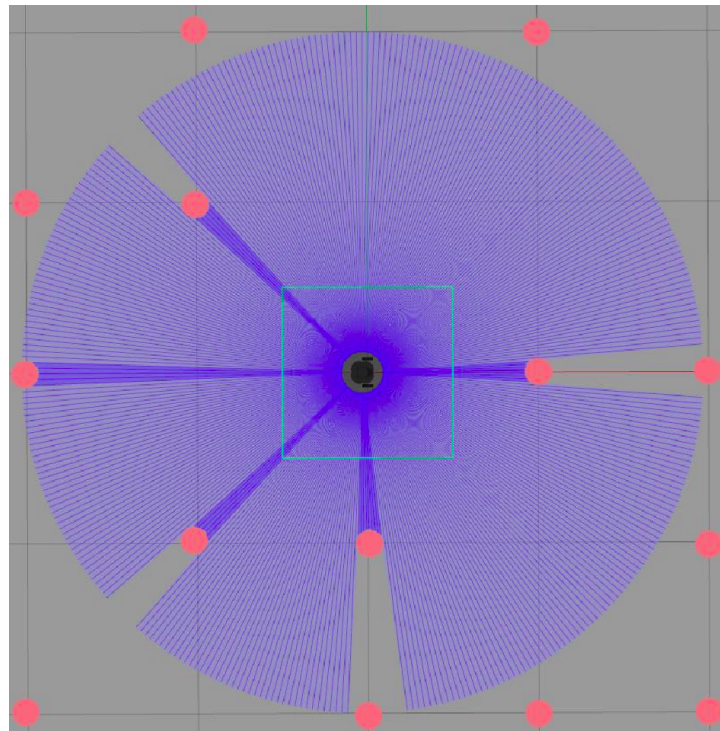
4.1.2 Processamento de dados

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 utilizado o algoritmo de estimação de círculos a partir de pontos em coordenadas cartesianas, discutido em (AL-SHARADQAH; CHERNOV, 2009, p. 903).

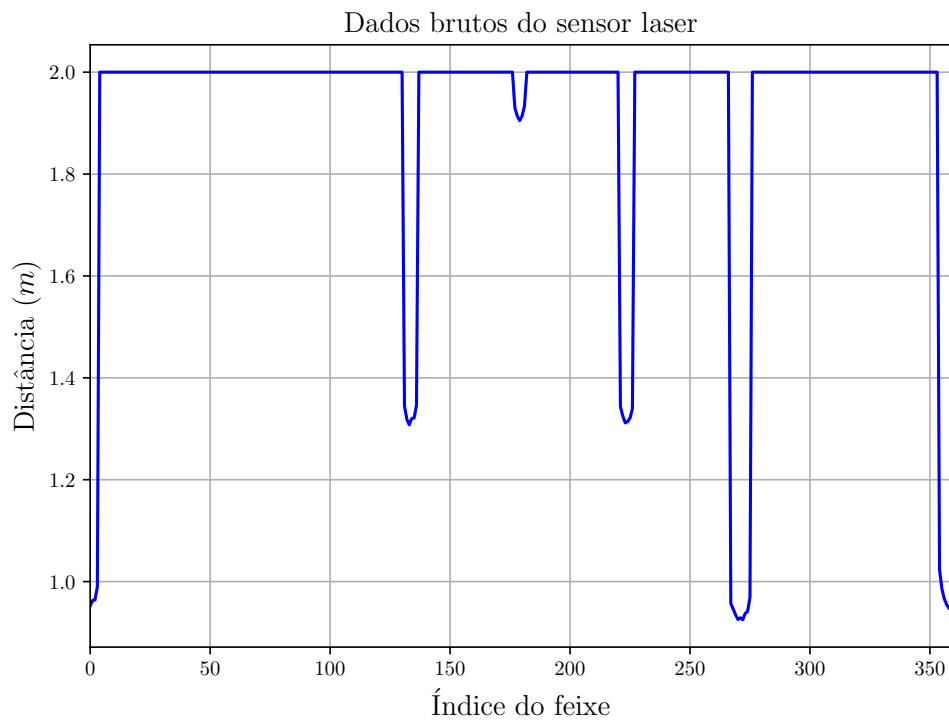
Porém antes é necessário extrair os pontos que correspondem à reflexões nas superfícies dos cilindros presentes no ambiente. Para isso, observa-se que há uma variação brusca nas distâncias lidas pelo sensor quando os feixes são refletidos pelas superfícies dos cilindros. Ao analisar a derivada do sinal, representada na Figura ??, 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.

4.2 Mapa em grade

4.3 Exploração Autônoma

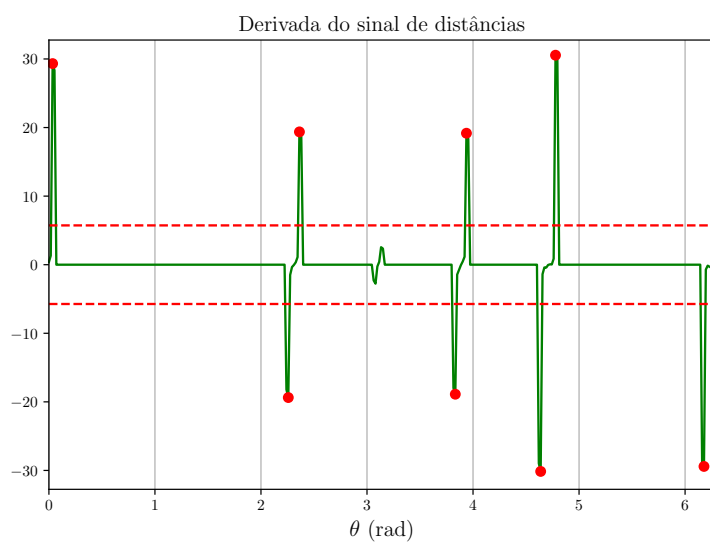


(a) Representação dos feixes laser (em azul) emitidos pelo sensor LiDAR com alcance máximo de 2 metros. Os círculos em rosa representam as landmarks.



(b) Interpolação linear das distâncias lidas pelo sensor.

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



5 SLAM multiagente descentralizado

5.1 Exploração distribuída

5.2 Cálculo da posição relativa entre agentes

5.3 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.
- 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.
- 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.
- SOLÀ, J. **Simultaneous 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 2, 3 e 4, respectivamente. Por fim, o Algoritmo 5 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 2 quanto no 3.

Algorithm 2 Etapa de predição do EKF-SLAM

```
1: function EKF-SLAM-PREDIÇÃO( $\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t$ )
2:    $\bar{\boldsymbol{\mu}}_t \leftarrow \begin{bmatrix} \mathbf{g}_R(\boldsymbol{\mu}_{R,t-1}, \mathbf{u}_t) \\ \boldsymbol{\mu}_M \end{bmatrix}$ 
3:    $\bar{\mathbf{P}}_{RR,t} \leftarrow \mathbf{G}_{R,t} \mathbf{P}_{RR,t-1} \mathbf{G}_{R,t}^T$ 
4:    $\bar{\mathbf{P}}_{RM,t} \leftarrow \mathbf{G}_{R,t} \mathbf{P}_{RM,t-1}$ 
5:    $\bar{\mathbf{P}}_t \leftarrow \begin{bmatrix} \bar{\mathbf{P}}_{RR,t} & \bar{\mathbf{P}}_{RM,t} \\ \bar{\mathbf{P}}_{RM,t}^T & \mathbf{P}_{MM} \end{bmatrix}$ 
6:   return  $\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t$ 
7: end function
```

Algorithm 3 Etapa de atualização do EKF-SLAM

```

1: function EKF-SLAM-ATUALIZAÇÃO( $\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t, \mathbf{y}, j = \text{índice da landmark}$ )
2:    $\mathbf{z} \leftarrow \mathbf{y} - \mathbf{h}^j(\bar{\boldsymbol{\mu}}_t)$ 
3:    $\mathbf{Z} \leftarrow \begin{bmatrix} \mathbf{H}_R^j & \mathbf{H}_M^j \end{bmatrix} \begin{bmatrix} \bar{\mathbf{P}}_{RR} & \bar{\mathbf{P}}_{RM_j} \\ \bar{\mathbf{P}}_{RM_j}^T & \bar{\mathbf{P}}_{M_j M_j} \end{bmatrix} \begin{bmatrix} \mathbf{H}_R^j \\ \mathbf{H}_M^j \end{bmatrix} + \mathbf{Q}_t$ 
4:    $\mathbf{K} \leftarrow \begin{bmatrix} \mathbf{P}_{RR,t} & \mathbf{P}_{RM^j,t} \\ \mathbf{P}_{MR,t} & \mathbf{P}_{MM^j,t} \end{bmatrix} \begin{bmatrix} [\mathbf{H}_R^j]^T \\ [\mathbf{H}_M^j]^T \end{bmatrix} \mathbf{Z}^{-1}$ 
5:    $\boldsymbol{\mu}_t \leftarrow \mathbf{K}\mathbf{z}$ 
6:    $\mathbf{P}_t \leftarrow \bar{\mathbf{P}}_t - \mathbf{K}\mathbf{Z}\mathbf{K}^T$ 
7:   return  $\boldsymbol{\mu}_t, \mathbf{P}_t$ 
8: end function

```

Algorithm 4 Etapa de inserção de nova *landmark* do EKF-SLAM

```

1: function EKF-SLAM-INSERÇÃO-NOVA-LANDMARK( $\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t, \mathbf{y}, j$ )
2:    $\bar{\boldsymbol{\mu}}_t^* \leftarrow \begin{bmatrix} \bar{\boldsymbol{\mu}}_t \\ \mathbf{f}(\bar{\boldsymbol{\mu}}_t, \mathbf{y}) \end{bmatrix}$ 
3:    $\bar{\mathbf{P}}_t^* \leftarrow \begin{bmatrix} \bar{\mathbf{P}}_t & \bar{\mathbf{P}}_t \mathbf{F}_X^T \\ \bar{\mathbf{P}}_t \mathbf{F}_X & \mathbf{F}_X \bar{\mathbf{P}}_t \mathbf{F}_X^T + \mathbf{F}_Y \mathbf{Q} \mathbf{F}_Y^T \end{bmatrix}$ 
4:   return  $\bar{\boldsymbol{\mu}}_t^*, \bar{\mathbf{P}}_t^*$ 
5: end function

```

Algorithm 5 EKF-SLAM

```

1: function EKF-SLAM( $\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t, \mathbf{y}^{1:k}, \mathbf{c}^{1:k}$ )
2:    $\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t \leftarrow \text{EKF-SLAM-PREDIÇÃO}(\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t)$ 
3:    $\mathcal{L}^* \leftarrow \{\}$  ▷ Conjunto de novas landmarks
4:   for ( $\mathbf{y}^i \in \mathbf{y}^{1:k}$ ) do
5:      $j \leftarrow \mathbf{c}^i$ 
6:     if landmark  $j$  está no mapa  $\mathbf{x}_M$  then
7:        $\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t \leftarrow \text{EKF-SLAM-ATUALIZAÇÃO}(\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t, \mathbf{y}^i, j)$ 
8:     else
9:        $\mathcal{L}^* \leftarrow \mathcal{L}^* + \{(j, \mathbf{y}^i)\}$ 
10:    end if
11:  end for
12:  for  $\mathcal{L}^i \in \mathcal{L}^*$  do
13:     $j, \mathbf{y}^i \leftarrow \mathcal{L}^i$ 
14:     $\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t \leftarrow \text{EKF-SLAM-INSERÇÃO-NOVA-LANDMARK}(\bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t, \mathbf{y}^i, j)$ 
15:  end for
16:   $\boldsymbol{\mu}_t, \mathbf{P}_t \leftarrow \bar{\boldsymbol{\mu}}_t, \bar{\mathbf{P}}_t$ 
17:  return  $\boldsymbol{\mu}_t, \mathbf{P}_t$ 
18: end function

```

Apêndice B - Matrizes

B.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 matrizes quadradas invertíveis \mathbf{R} e \mathbf{Q} e qualquer matriz \mathbf{P} com dimensões apropriadas, o seguinte é verdadeiro:*

$$(\mathbf{R} + \mathbf{P}\mathbf{Q}\mathbf{P}^T)^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{P}(\mathbf{Q}^{-1} + \mathbf{P}^T\mathbf{R}^{-1}\mathbf{P})^{-1}\mathbf{P}^T\mathbf{R}^{-1} \quad (\text{B.1})$$

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

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

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

Isso é mostrado através de uma série de manipulações:

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

□

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO DM	2. DATA 25 de março de 2015	3. DOCUMENTO Nº DCTA/ITA/DM-018/2015	4. Nº DE PÁGINAS 53
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: Prof ^{ra} . Dr ^a . Doralice Serra. Defesa em 05/03/2015. Publicada em 25/03/2015.			
11. RESUMO:			
12. GRAU DE SIGILO: (X) OSTENSIVO () RESERVADO () SECRETO			