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.
38f.

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. 38f. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Wellington Vieira Martins de Castro
TITULO DO TRABALHO: SLAM distribuído envolvendo navegação, guiamento e fusão sensorial para reconstrução 2D.
TIPO DO TRABALHO/ANO: Dissertação / 2015

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 por motivarem tanto a criação deste template pelo Fábio Fagundes Silveira quanto por motivarem a mim e outras pessoas a atualizarem e aprimorarem este excelente trabalho.

Agradecimentos

Primeiramente, gostaria de agradecer ao Dr. Donald E. Knuth, por ter desenvolvido o TEX.

Ao Dr. Leslie Lamport, por ter criado o LATEX, facilitando muito a utilização do TEX, e assim, eu não ter que usar o Word.

Ao Prof. Dr. Meu Orientador, pela orientação e confiança depositada na realização deste trabalho.

Ao Dr. Nelson D'Ávilla, por emprestar seu nome a essa importante via de trânsito na cidade de São José dos Campos.

Ah, já estava esquecendo... agradeço também, mais uma vez ao TEX, por ele não possuir vírus de macro :-)

"If I have seen farther than others,
it is because I stood on the shoulders of giants."
— Sir Isaac Newton

Resumo

Aqui começa o resumo do referido trabalho. Não tenho a menor idéia do que colocar aqui. Sendo assim, vou inventar. Lá vai: Este trabalho apresenta uma metodologia de controle de posição das juntas passivas de um manipulador subatuado de uma maneira subótima. O termo subatuado se refere ao fato de que nem todas as juntas ou graus de liberdade do sistema são equipados com atuadores, o que ocorre na prática devido a falhas ou como resultado de projeto. As juntas passivas de manipuladores desse tipo são indiretamente controladas pelo movimento das juntas ativas usando as características de acoplamento da dinâmica de manipuladores. A utilização de redundância de atuação das juntas ativas permite a minimização de alguns critérios, como consumo de energia, por exemplo. Apesar da estrutura cinemática de manipuladores subatuados ser idêntica a do totalmente atuado, em geral suas caraterísticas dinâmicas diferem devido a presença de juntas passivas. Assim, apresentamos a modelagem dinâmica de um manipulador subatuado e o conceito de índice de acoplamento. Este índice é utilizado na sequência de controle ótimo do manipulador. A hipótese de que o número de juntas ativas seja maior que o número de passivas ($n_a > n_p$) permite o controle ótimo das juntas passivas, uma vez que na etapa de controle destas há mais entradas (torques nos atuadores das juntas ativas), que elementos a controlar (posição das juntas passivas).

Abstract

Well, the book is on the table. This work presents a control methodologie for the position of the passive joints of an underactuated manipulator in a suboptimal way. The term underactuated refers to the fact that not all the joints or degrees of freedom of the system are equipped with actuators, which occurs in practice due to failures or as design result. The passive joints of manipulators like this are indirectly controlled by the motion of the active joints using the dynamic coupling characteristics. The utilization of actuation redundancy of the active joints allows the minimization of some criteria, like energy consumption, for example. Although the kinematic structure of an underactuated manipulator is identical to that of a similar fully actuated one, in general their dynamic characteristics are different due to the presence of passive joints. Thus, we present the dynamic modelling of an underactuated manipulator and the concept of coulpling index. This index is used in the sequence of the optimal control of the manipulator.

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

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 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*
$p\left(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0\right)$	Distribuição do problema online SLAM
$p\left(\mathbf{x}_{0:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0\right)$	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
$\boldsymbol{g}_R(\bullet, \bullet)$	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 $m \times n$

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.

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
- $\mathbf{x}_{1:t} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t\}$: histórico de poses do robô até o instante t
- $\mathbf{u}_t$: o vetor de controle executado pelo robô no instante t
- $\mathbf{u}_{0:t} = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{t-1}, \mathbf{u}_t\}$: histórico de controles executados pelo robô até o instante t

- $\mathbf{z}_t$: o vetor de medidas no instante t
- $\mathbf{z}_{0:t} = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{t-1}, \mathbf{z}_t\}$: 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:

$$p\left(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{0:t}, \mathbf{x}_0\right) \tag{1.1}$$

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

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 $\mathbf{x}_t$. 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 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 Organização do trabalho

2 Visão Geral do Sistema

2.1 O ambiente

2.2 Modelo do robô

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, quanto o sistema está no estado $\mathbf{x}_t$. Na Figura 2.1, 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:

$$\mathbf{x} = \begin{bmatrix} \phi & x & y & m_x^i & m_y^i & m_x^j & m_y^j & m_x^k & m_y^k \end{bmatrix}^T \tag{2.1}$$

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

$$\boldsymbol{h}^j(\mathbf{x}) = \begin{bmatrix} r^j \\ \theta^j \end{bmatrix} = \begin{bmatrix} \sqrt{(m_x^j - x_l)^2 + (m_y^j - y_l)^2} \\ \arctan\left(\dfrac{m_y^j - y_l}{m_x^j - x_l}\right) - \phi \end{bmatrix} \tag{2.2}$$

onde

$$\begin{cases} x_l = x + d\cos\phi \\ y_l = y + d\sin\phi \end{cases} \tag{2.3}$$

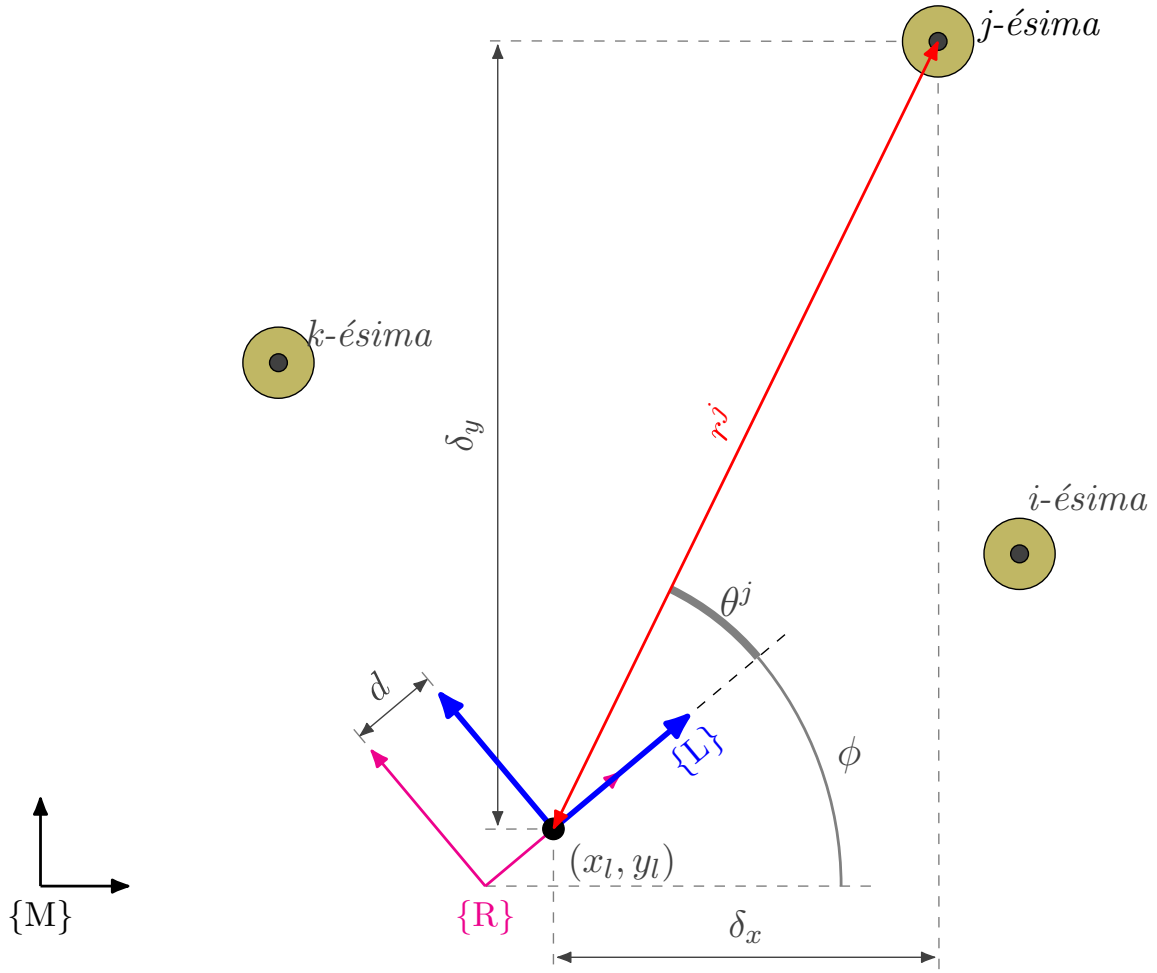

FIGURA 2.1 – Esquemático do modelo de medida *Range-Bearing*. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. A medida (r^j, θ^j) se refere à i-ésima *landmark* no mapa.

Como o modelo de medida em 2.2 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 2.6, 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.

$$\mathbf{H}_R^j = \begin{bmatrix} \frac{d}{r^j}\left(\delta_x \sin\phi - \delta_y \cos\phi\right) & \frac{-\delta_x}{r^j} & \frac{-\delta_y}{r^j} \\ -\left(\frac{d}{[r^j]^2}\left(\delta_y \sin\phi + \delta_x \cos\phi\right) + 1\right) & \frac{\delta_y}{[r^j]^2} & \frac{-\delta_x}{[r^j]^2} \end{bmatrix} \tag{2.4}$$

$$\mathbf{H}_M^j = \begin{bmatrix} \frac{\delta_x}{r^j} & \frac{\delta_y}{r^j} \\ \frac{-\delta_y}{[r^j]^2} & \frac{\delta_x}{[r^j]^2} \end{bmatrix} \tag{2.5}$$

$$\mathbf{H}^j(\mathbf{x}) = \begin{bmatrix} \mathbf{H}_R^j & \mathbf{0} & \cdots & \mathbf{H}_M^j & \cdots & \mathbf{0} \end{bmatrix} \tag{2.6}$$

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 da meios para que suas posições sejam incorporadas no vetor de estados, na Figura 2.2 está representado o momento no qual o robô descobre a p-ésima *landmark* do ambiente.

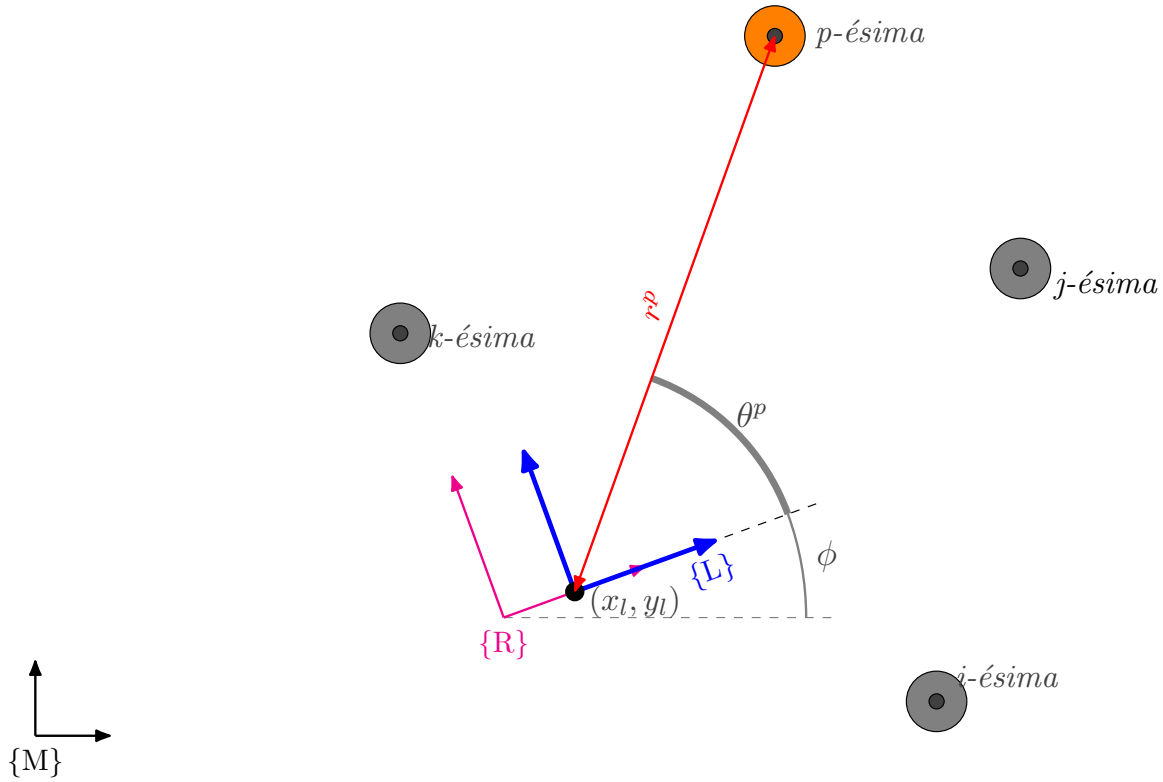

FIGURA 2.2 – Esquemático do modelo de medida *Range-Bearing*. O sistema de coordenadas do robô é representado em magenta, e o sistema de coordenadas do sensor em azul. Os círculos representam *landmarks*, as conhecidas pelo robô (portanto presentes no vetor de estados), em cinza, e recém descobertas, 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 2.7. 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 2.10 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 2.8 e 2.9, respectivamente.

$$\boldsymbol{f}(\mathbf{x}, \mathbf{y}^p) = \begin{bmatrix} m_x^p \\ m_y^p \end{bmatrix} = \begin{bmatrix} x_l + r^p \cos(\phi + \theta^p) \\ y_l + r^p \sin(\phi + \theta^p) \end{bmatrix} \tag{2.7}$$

$$\mathbf{F}_X = \begin{bmatrix} \begin{bmatrix} -d\sin\phi - r^p \sin(\phi + \theta^p) & 1 & 0 \\ d\cos\phi + r^p \cos(\phi + \theta^p) & 0 & 1 \end{bmatrix} & \mathbf{0}_{2\times n-3} \end{bmatrix} \tag{2.8}$$

$$\mathbf{F}_Y = \begin{bmatrix} \cos(\phi + \theta^p) & -r^p \sin(\phi + \theta^p) \\ \sin(\phi + \theta^p) & r^p \cos(\phi + \theta^p) \end{bmatrix} \tag{2.9}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_X & \mathbf{F}_Y \end{bmatrix} \tag{2.10}$$

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 este 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.

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

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

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 3.3 até 3.9 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 $\overline{\boldsymbol{\mu}}_t$.

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

$$\overline{\mathbf{P}}_t = \mathbf{G}_t \mathbf{P}_{t-1} \mathbf{G}_t^T + \mathbf{R}_t \tag{3.4}$$

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

$$\mathbf{Z}_t = \mathbf{H}_t \overline{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{Q}_t \tag{3.6}$$

$$\mathbf{K}_t = \overline{\mathbf{P}}_t \mathbf{H}_t^T \mathbf{Z}_t^{-1} \tag{3.7}$$

$$\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + \mathbf{K}_t \mathbf{z}_t \tag{3.8}$$

$$\mathbf{P}_t = \overline{\mathbf{P}}_t - \mathbf{K}_t \mathbf{Z}_t \mathbf{K}_t^T \tag{3.9}$$

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.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_M \end{bmatrix} \tag{3.10}$$

[1] O leitor pode estanhar a Equação 3.9 do erro da estimativa. Normalmente ela é escrita na forma $\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\overline{\mathbf{P}}_t$, 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*.

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

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 do modelo de movimento do robô $\boldsymbol{g}_R$ concatenado com as posições das *landmarks*:

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

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

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

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:

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

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

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

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.

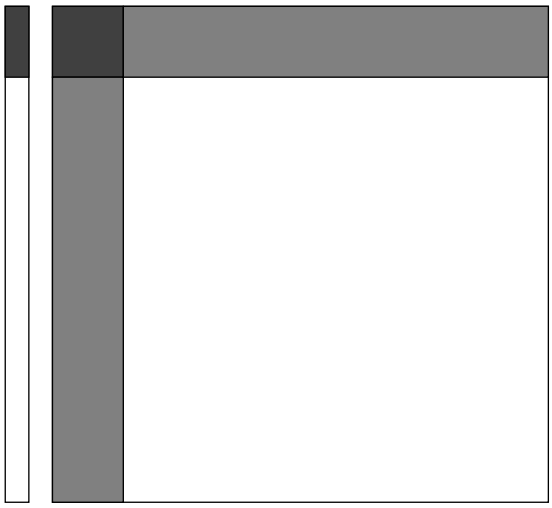

FIGURA 3.1 – Partes modificadas do vetor média e da matriz de covariância durante o movimento do robô. O vetor média é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As partes modificadas, em tons de cinza, correspondem ao estado do robô $\boldsymbol{\mu}_R$ 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 $\mathbf{z}_t^j$.

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

Além disso, como o jacobiano do modelo de medida na Equação 2.6 é esparso, o cálculo da covariância da inovação pode ser obtido por:

$$\mathbf{Z}_t^j = \begin{bmatrix} \mathbf{H}_R^j & \mathbf{H}_M^j \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_{RR} & \overline{\mathbf{P}}_{RM_j} \\ \overline{\mathbf{P}}_{RM_j}^T & \overline{\mathbf{P}}_{M_jM_j} \end{bmatrix} \begin{bmatrix} \mathbf{H}_R^j \\ \mathbf{H}_M^j \end{bmatrix} + \mathbf{Q}_t \tag{3.17}$$

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, claro, aqui esse cálculo de complexidade constante deva ser realizado 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 2.6. 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 da do vetor de estados e da matriz de covariância 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.

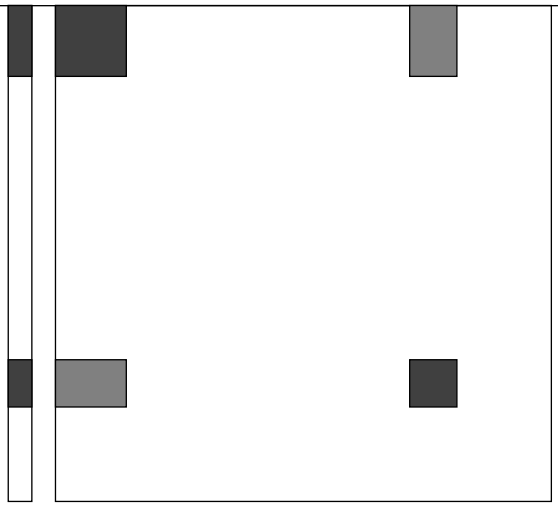

FIGURA 3.2 – Partes utilizadas do vetor média e da matriz de covariância durante o cálculo da inovação, quando uma *landmark* é observada. O vetor média é representado pela barra na esquerda, e a matriz de covariância pelo quadrado na direita. As porções utilizadas, em tons de cinza, correspondem ao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

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

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 de medição do sensor.

Inicializar a covariância da nova *landmark* com ∞ (ou números muito grandes), como indicado em (BONGARD, 2006, p. 317), pode ser injusto. Portanto devemos calcular o

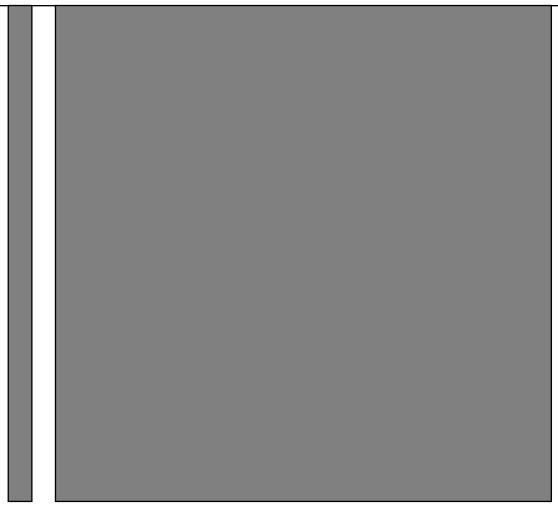

FIGURA 3.3 – O vetor média e a matriz de covariâncias são completamente atualizados durante a observação de uma *landmark*. Retirado de (SOLÀ, 2014, p. 8).

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.

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

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

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

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 2.8 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.

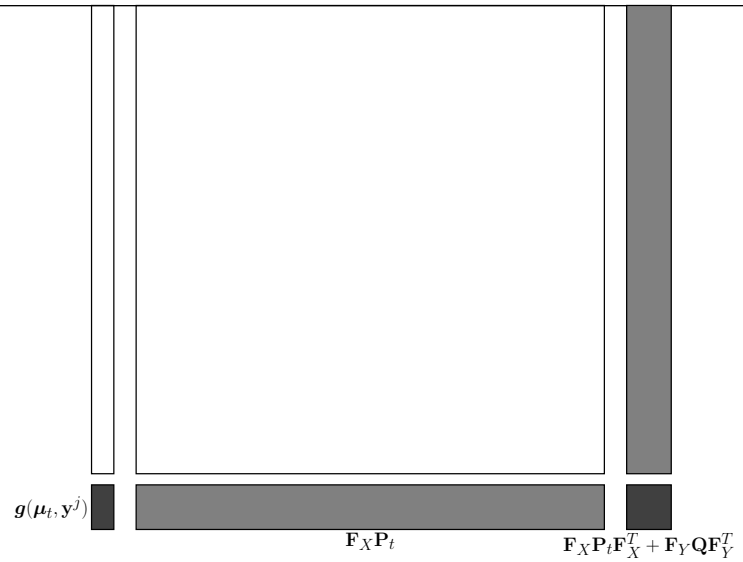

FIGURA 3.4 – Vetor média e matriz de covariância aumentados após inserção de nova *landmark*. As partes adicionadas, em cinza, correspondem às covariâncias cruzadas entre a nova landmark e o vetor de estados anterior (cinza claro), e à média da nova *landmark* e sua covariância (cinza escuro). Adaptado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

3.2.4 EKF-SLAM: Algoritmo

Algorithm 1 Etapa de predição do EKF-SLAM

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

$$\overline{\boldsymbol{\mu}}_t \leftarrow \begin{bmatrix} \boldsymbol{g}_R(\boldsymbol{\mu}_{R,t-1}, \mathbf{u}_t) \\ \boldsymbol{\mu}_M \end{bmatrix}$$

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

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

$$\overline{\mathbf{P}}_t \leftarrow \begin{bmatrix} \overline{\mathbf{P}}_{RR,t} & \overline{\mathbf{P}}_{RM,t} \\ \overline{\mathbf{P}}_{RM,t}^T & \mathbf{P}_{MM} \end{bmatrix}$$

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

end function

3.3 Filtro de Informação Extendido (EIF)

3.4 SEIF-SLAM

3.5 Conclusão do capítulo

Algorithm 2 Etapa de atualização do EKF-SLAM

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

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

$$\mathbf{Z} \leftarrow \begin{bmatrix} \mathbf{H}_R^j & \mathbf{H}_M^j \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_{RR} & \overline{\mathbf{P}}_{RM_j} \\ \overline{\mathbf{P}}_{RM_j}^T & \overline{\mathbf{P}}_{M_jM_j} \end{bmatrix} \begin{bmatrix} \mathbf{H}_R^j \\ \mathbf{H}_M^j \end{bmatrix} + \mathbf{Q}_t$$

$$\mathbf{K} \leftarrow \begin{bmatrix} \mathbf{P}_{RR,t} & \mathbf{P}_{RM^j,t} \\ \mathbf{P}_{MR,t} & \mathbf{P}_{MM^j,t} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \mathbf{H}_R^j \end{bmatrix}^T \\ \begin{bmatrix} \mathbf{H}_M^j \end{bmatrix}^T \end{bmatrix} \mathbf{Z}^{-1}$$

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

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

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

end function

Algorithm 3 Etapa de inserção de nova *landmark* do EKF-SLAM

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

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

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

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

end function

Algorithm 4 EKF-SLAM

function EKF-SLAM($\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t, \mathbf{y}^{1:k}, \mathbf{c}^{1:k}$)
  $\overline{\boldsymbol{\mu}}_t, \overline{\mathbf{P}}_t \leftarrow$ EKF-SLAM-Predição($\boldsymbol{\mu}_{t-1}, \mathbf{P}_{t-1}, \mathbf{u}_t$)
  $\mathcal{L}^* \leftarrow \{\}$ ▷ Conjunto de novas landmarks
  for ($\mathbf{y}^i \in \mathbf{y}^{1:k}$) **do**
    $j \leftarrow \mathbf{c}^i$
    if landmark j está no mapa $\mathbf{x}_M$ **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

4 Conclusão

Neste trabalho realizou-se o projeto de uma metodologia de controle subótimo redundante da junta passiva de um manipulador com três graus de liberdade instantaneamente. Para este propósito usou-se nas formulações o vetor gradiente de uma função escalar que estima o acoplamento entre a junta passiva e as ativas desse manipulador. Aqui a redundância foi usada da melhor maneira possível sem focalizar o efeito global. Portanto, este método deve ser denominado de *controle ótimo local por redundância*. A principal vantagem dessa formulação é a computação em tempo real, que é necessária para o controle do manipulador experimental. Além disso esse método pode ser usado com diferentes tipos de controladores, uma vez que as alterações são feitas nas equações dinâmicas do manipulador.

A consequência direta observada nessa formulação é a redução dos torques na fase de controle da junta passiva, e consequente redução da energia elétrica gasta. Isso ocorre devido ao fato de que ao longo da trajetória do manipulador o índice de acoplamento de torque tende a ser maximizado, e portanto, menor é o torque necessário nos atuadores para se conseguir o posicionamento da junta passiva do manipulador.

Outros resultados indiretos obtidos são: um movimento mais uniforme e suave do manipulador e um tempo de acomodação menor tanto no posicionamento da junta passiva quanto das ativas, conforme podemos obervar nos gráficos de desempenho dos resultados apresentados. Isso ocorre porque a maximização do acoplamento entre as juntas facilita o controle. Assim ocorrem menos picos de torque, e como as juntas ativas tem "menos trabalho" para posicionar a passiva estas se movem menos na direçao contrária ao movimento daquelas, diminuindo assim as velocidades alcançadas e os tempos de posicionamento.

Uma extensão deste trabalho pode ser a implementação de um *controle ótimo global por redundância* da junta passiva do manipulador. Para isto pode-se fazer o planejamento *off-line* da trajetória das juntas de modo a minimizar a energia consumida. Alguns estudos foram feitos nesse sentido, usando o Princípio Mínimo de Pontryagin, mas sem resultados satisfatórios até o momento.

Referências

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.

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. **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 - Tópicos de Dilema Linear

A.1 Notas sobre detalhes de implementação

notas sobre implementação vao aqui

Anexo A - Exemplo de um Primeiro Anexo

A.1 Uma Seção do Primeiro Anexo

Algum texto na primeira seção do primeiro anexo.

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	38

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:

Aqui começa o resumo do referido trabalho. Não tenho a menor idéia do que colocar aqui. Sendo assim, vou inventar. Lá vai: Este trabalho apresenta uma metodologia de controle de posição das juntas passivas de um manipulador subatuado de uma maneira subótima. O termo subatuado se refere ao fato de que nem todas as juntas ou graus de liberdade do sistema são equipados com atuadores, o que ocorre na prática devido a falhas ou como resultado de projeto. As juntas passivas de manipuladores desse tipo são indiretamente controladas pelo movimento das juntas ativas usando as características de acoplamento da dinâmica de manipuladores. A utilização de redundância de atuação das juntas ativas permite a minimização de alguns critérios, como consumo de energia, por exemplo. Apesar da estrutura cinemática de manipuladores subatuados ser idêntica a do totalmente atuado, em geral suas caraterísticas dinâmicas diferem devido a presença de juntas passivas. Assim, apresentamos a modelagem dinâmica de um manipulador subatuado e o conceito de índice de acoplamento. Este índice é utilizado na sequência de controle ótimo do manipulador. A hipótese de que o número de juntas ativas seja maior que o número de passivas ($n_a > n_p$) permite o controle ótimo das juntas passivas, uma vez que na etapa de controle destas há mais entradas (torques nos atuadores das juntas ativas), que elementos a controlar (posição das juntas passivas).

12. GRAU DE SIGILO:

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