COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

MAPEAMENTO E RASTREAMENTO DE MÚLTIPLOS OBJETOS USANDO CONJUNTOS ALEATÓRIOS FINITOS

Kleberson Meireles de Lima

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Ramon Romankevicius Costa

Rio de Janeiro
Junho de 2022

MAPEAMENTO E RASTREAMENTO DE MÚLTIPLOS OBJETOS USANDO CONJUNTOS ALEATÓRIOS FINITOS

Kleberson Meireles de Lima

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Ramon Romankevicius Costa

Aprovada por: Prof. Ramon Romankevicius Costa
Prof. Fernando Cesar Lizarralde
Prof. Alessandro Jacoud Peixoto
Prof. Paulo César Pellanda
Prof. José Paulo Vilela Soares da Cunha

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2022

Lima, Kleberson Meireles de

Mapeamento e rastreamento de múltiplos objetos usando conjuntos aleatórios finitos/Kleberson Meireles de Lima. – Rio de Janeiro: UFRJ/COPPE, 2022.

XVI, 141 p.: il.; 29, 7cm.

Orientador: Ramon Romankevicius Costa

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2022.

Referências Bibliográficas: p. 96 – 113.

1. Random Finite Sets. 2. Multitarget tracking. 3. Cooperativo. 4. mapeamento. 5. Robótica. I. Costa, Ramon Romankevicius. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Agradecimentos

Aos meu pais, Anai e José, por minha vida e pelos ensinamentos que a fundamentam até hoje.

À minha amada, Ana, que, apesar do hiato, sempre me incentivou e apoiou para que eu pudesse alcançar mais um objetivo.

Ao meu orientador, pelo apoio e contribuições que tornaram este trabalho possível.

À banca, pelas contribuições que elevaram o nível desta tese.

Aos companheiros da COPPE/UFRJ, em especial, Diego Dias e Alessandro de Lima, pela amizade e ajuda.

Aos companheiros, Rui Rodrigues e Rubens Pailo, de Instituto de Pesquisas da Marinha, pela ajuda e discussões que contribuíram para a conclusão deste trabalho.

Aos companheiros, em particular, Lucas Wilke e Vanius Ferreira, de Diretoria de Engenharia Naval, pela ajuda e contribuições para este trabalho.

A todos que contribuíram direta ou indiretamente para que esta tese fosse possível.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MAPEAMENTO E RASTREAMENTO DE MÚLTIPLOS OBJETOS USANDO CONJUNTOS ALEATÓRIOS FINITOS

Kleberson Meireles de Lima

Junho/2022

Orientador: Ramon Romankevicius Costa

Programa: Engenharia Elétrica

Este trabalho apresenta métodos baseados em conjuntos aleatórios finitos para aplicação ou RFSs (do inglês, *Random Finite Sets*) nos problemas de rastreamento de múltiplos objetos e no mapeamento em robótica, o qual também faz parte do problema do *Simultaneous Localization And Mapping* (SLAM). Problemas de rastreamento de alvos ou objetos têm um papel central em diferentes aplicações, tais como sistemas de vigilância em redes de sensores, robótica móvel e veículos autônomos. Ao longo dos anos, tornou-se objeto de estudo da comunidade acadêmica e de grandes corporações de diversos setores, como transporte, defesa e segurança, logística e até mesmo no comércio eletrônico. Embora as técnicas convencionais tenham sido amplamente estudadas, o problema permanece em aberto. Além disso, as técnicas baseadas em RFSs apresentam-se como potenciais soluções para mapeamento ou rastreamento em larga escala. Nesse contexto, este estudo tem como objetivo propor técnicas baseadas em RFS para aplicação em sistemas complexos, tais como no mapeamento (ou SLAM) cooperativo ou grandes redes de sensores. Nesse sentido, esta tese propõe um esquema de fusão de dados para um sistema de vigilância marítima. Por meio de simulações, é ilustrado o desempenho do esquema proposto em um cenário realista. Os resultados obtidos indicam que problemas do filtro *Probability Hypothesis Density* (PHD) original foram superados, em particular, o problema da oclusão. Além disso, é proposto o uso do modelo computacional Gamma para a paralelização do filtro PHD baseado em mistura Gaussiana. O potencial de uso deste modelo é demonstrado por simulações utilizando o MATLAB *Parallel Computing Toolbox*, obtendo-se um aumento de velocidade significativo no processamento ao se comparar o algoritmo paralelizado com o sequencial.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

COOPERATIVE TRACKING AND MAPPING: A RANDOM FINITE SETS BASED APPROACH

Kleberson Meireles de Lima

June/2022

Advisor: Ramon Romankevicius Costa

Department: Electrical Engineering

This work presents Random Finite Sets (RFS) based methods for application to multitarget tracking and robotics mapping problems, which is part of the Simultaneous Localization And Mapping (SLAM) problem. The tracking problem is a key issue in applications such as surveillance systems in distributed sensor networks, mobile robotics, and autonomous vehicles. Over the years, it has become an object of study by the academic community and large corporations from various sectors, such as transport, defense and security, logistics, and even e-commerce. Although conventional techniques have been widely studied, the problem remains open, and those based on RFS present themselves as potential solutions for large-scale mapping or tracking. This study proposes techniques based on RFS for application in complex systems, such as cooperative mapping (or SLAM) or large sensor networks. In this sense, this thesis proposes a data fusion scheme for a maritime surveillance system. The proposed scheme's performance in a realistic scenario is illustrated through simulations. The results indicate that problems of the original Probability Hypothesis Density (PHD) filter were overcome, particularly the problem of occlusion. Furthermore, the Gamma computational model is proposed to parallelize the GM (Gaussian Mixture) PHD filter. The potential use of this model is demonstrated through simulations using MATLAB Parallel Computing Toolbox, obtaining a representative speed up in processing.

Sumário

Lista de Figuras x

Lista de Tabelas xiii

Lista de Abreviaturas xiv

1 Introdução 1

1.1 Introdução aos sistemas autônomos . . . 1
1.2 Breve histórico dos veículos autônomos . . . 2
1.3 O problema de mapeamento e localização em robótica móvel . . . 3
1.4 O problema de rastreamento de múltiplos alvos . . . 4
1.5 Revisão da literatura . . . 4
1.5.1 Mapeamento e localização em robótica . . . 5
1.5.2 *Multitarget tracking* . . . 9
1.6 Motivação . . . 11
1.7 Objetivo . . . 12
1.7.1 Publicações . . . 12
1.7.2 Contribuições . . . 13
1.8 Definição do problema . . . 13
1.8.1 Mapeamento e localização . . . 13
1.8.2 *Multitarget tracking* . . . 14
1.8.3 O problema cooperativo . . . 15
1.9 Organização deste texto . . . 16

2 Conjuntos Aleatórios Finitos 17

2.1 Conjuntos Aleatórios Finitos . . . 17
2.2 Por que usar RFSs? . . . 18
2.2.1 RFS x técnicas baseadas em vetores . . . 18
2.3 Conceitos-chave em FISST . . . 22
2.3.1 *Definição de conjunto aleatório finito* . . . 22
2.3.2 Funções de probabilidade e distribuições de um RFS . . . 23
2.3.3 Integral de conjunto . . . 25

2.3.4 Função intensidade . . . 27
2.3.5 Processo de Poisson em RFSs . . . 28
2.4 Filtro PHD . . . 29
2.4.1 Filtro RFS recursivo . . . 29
2.4.2 Filtro PHD recursivo . . . 31
2.4.3 Filtro PHD baseado em misturas Gaussianas . . . 32
2.4.4 Considerações finais . . . 38

3 Filtro PHD no problema de mapeamento em robótica 39
3.1 A taxonomia dos problemas de localização e mapeamento . . . 39
3.1.1 SLAM completo x em tempo real . . . 40
3.1.2 Localização global x local . . . 40
3.1.3 Ambientes estáticos x dinâmicos . . . 40
3.1.4 Representação topológica x métrica . . . 41
3.1.5 Localização passiva x ativa . . . 41
3.1.6 Multi-robôs x robô único . . . 41
3.1.7 Localização baseada em marcos x casamento de modelos . . . 41
3.2 Mapeamento "convencional" . . . 42
3.2.1 Mapas baseados em *occupancy grids* . . . 42
3.2.2 Mapas baseados em *Features* . . . 43
3.2.3 Localização e mapeamento simultâneos . . . 44
3.3 RFS aplicado ao mapeamento em robótica . . . 45
3.4 Simulações e resultados . . . 47
3.4.1 Cenário . . . 47
3.4.2 Modelos utilizados . . . 48
3.4.3 Parâmetros do filtro . . . 49
3.4.4 Resultados . . . 49
3.5 Considerações finais . . . 51

4 Filtro PHD em rastreamento e vigilância marítima distribuída 52
4.1 Introdução aos sistemas de rastreamento . . . 52
4.1.1 Sistemas de rastreamento e vigilância no ambiente marítimo . . . 55
4.2 Simulações . . . 57
4.2.1 Cenário simulado . . . 57
4.2.2 Modelos utilizados . . . 58
4.2.3 Parâmetros utilizados nos filtros . . . 60
4.2.4 Resultados obtidos . . . 61
4.3 Considerações finais . . . 66

5 Filtro PHD e computação paralela **68**
5.1 Motivações para um filtro PHD paralelizado 68
5.2 O modelo Gamma 70
5.2.1 Motivação para a paralelização Gamma 70
5.2.2 Conceitos fundamentais em Gamma 71
5.2.3 Gamma aplicado à extração de estados 73
5.3 Simulações e resultados 74
5.3.1 Paralelização na extração de estados 76
5.3.2 Paralelização no filtro GM PHD 77
5.4 Considerações finais 91

6 Considerações Finais **93**
6.1 Conclusões 93
6.2 Trabalhos futuros 94

Referências Bibliográficas **96**

A Publicações **114**

Lista de Figuras

2.1 Exemplos de um RFS de 0 a 3 elementos em um ambiente quadrado. Fonte: extraído de [1]. 18
2.2 Três trajetórias 19
2.3 Vetores das observações diferentes a depender da trajetória. Fonte: extraído de [1]. 20
2.4 Detecção perdida com sensor realístico. Fonte: extraído de [1]. 20
2.5 Função intensidade unidimensional. 28
2.6 Exemplo do passo de predição do filtro GM PHD. 35
2.7 Exemplo do passo de predição do filtro GM PHD. 36

3.1 Cenário simulado em $k = 1$. 47
3.2 Cenário simulado em $k = 330$. 50
3.3 Métrica OSPA para as simulações realizadas. 51

4.1 Arquitetura de sistema distribuído com fusão de dados centralizada. . 56
4.2 Cenário de vigilância utilizado nas simulações 58
4.3 Rastreamento com um sensor. 61
4.4 Rastreamento de dois alvos com trajetórias diferentes. 62
4.5 Alvo em oclusão 63
4.6 Resultado após a inserção de um segundo radar. 64
4.7 Solução de *tracking* para o esquema proposto. 65
4.8 O número de alvos estimados pelas soluções locais individualmente e pelo esquema proposto. 65
4.9 Métrica OSPA. 66

5.1 Exemplo de sequência de execução para o algoritmo proposto. 75
5.2 Execução paralela de um loop usando o comando parfor. 76
5.3 Comparação entre o tempo de processamento sequencial x paralelo. . 77
5.4 Exemplo de *feature-based* SLAM: as elipses vermelhas representam *landmarks* estimadas pelo veículo, à medida que se move ao longo da trajetória em azul. 78

5.5 Parte do código paralelizado neste trabalho.
Fonte: algoritmo apresentado em [2]. 79
5.6 Comparação entre o tempo de processamento sequencial x paralelo. . 80
5.7 Comparação entre o tempo de processamento sequencial x paralelo - destaque para o intervalo de $5 \leq J_{k|k-1} \leq 6 \times 10^5$. 81
5.8 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações. 81
5.9 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações - destaque para o intervalo de $40 \leq J_{k|k-1} \leq 200$. 82
5.10 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $J_{k|k-1} = 10$. 82
5.11 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $J_{k|k-1} = 10$ - destaque para o intervalo de $0 \leq |z_k| \leq 10000$. 83
5.12 Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D. 84
5.13 Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 60$. 84
5.14 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações. 85
5.15 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 80$. 85
5.16 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $|J_{k|k-1}| = 50$. 86
5.17 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $|J_{k|k-1}| = 50$ - destaque para o intervalo de $2 \leq z_k \leq 80$. 86
5.18 Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D. 87
5.19 Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 70$. 88
5.20 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações. 88
5.21 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 100$. 89

5.22 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $J_{k|k-1} = 50$. 89
5.23 Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $|J_{k|k-1}| = 50$ - destaque para o intervalo de $2 \leq z_k \leq 80$. 90

Lista de Tabelas

3.1 Estado do robô e objetos (*landmarks*) em $k = 0$. 48
3.2 Parâmetros utilizados pelo filtro GM PHD. 49

4.1 Sensores e Estação Central utilizados nas simulações. 57
4.2 Estado inicial e tipo de movimento dos alvos. 58
4.3 Parâmetros utilizados pelos filtros. 60

5.1 Comparação entre os casos simulados - ponto em que a paralelização se torna vantajosa. 91

Lista de Abreviaturas

ALV(s)	*Autonomous Land Vehicle(s)*, p. 2
AUV(s)	*Autonomous Underwater Vehicle(s)*, p. 1
C-SLAM	*cooperative*-SLAM, p. 7
C4ISR	*Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance*, p. 56
COTS	*off-the-shelf*, p. 56
CPHD	PHD Cardinalizado, p. 10
CPU	Central Processing Unit, p. 75
CUDA	Compute Unified Device Architecture, p. 69
DARPA	*Defense Advanced Research Projects Agency*, p. 1
DMM	*direct map merging*, p. 7
EK-PHD	*extended* Kalman PHD (EK-PHD), p. 5
EKF-SLAM	*Extended Kalman filter* SLAM, p. 5
EKF	filtro *Extended* Kalman, p. 9
FBRM	*Feature Based Robotic Mapping*, p. 14
FB	*Feature-Based*, p. 6
FISST	*Finite-set Statistics*, p. 9, 22
FastSLAM	*Fast Simultaneous Localization and Mapping*, p. 44
FoV	*Field of View*, p. 55
GAMMA	*General Abstract Model for Multiset Manipulation*, p. 69
GLMB	*Generalized Labeled Multi-Bernoulli*, p. 10

GM	Gaussian Mixture, p. vi
GM	Mistura Gaussiana, p. 10
GNN	*Global Nearest Neighbor*, p. 54
GPS	*Global Positioning System*, p. 6
GPU	Graphics Processing Unit, p. 69
IHM	Interface Homem-Máquina, p. 76
IMM	*Interactive Multiple-Model*, p. 9
IMM	*indirect map merging*, p. 7
IoT	Internet das coisas, p. 11
KF	Filtro de Kalman, p. 9
MHKF	Filtro Multi-Hipótese de Kalman, p. 9
MHT	*Multiple Hypothesis Tracker*, p. 5
MIMO	*Multiple-Input Multiple-Output*, p. 10
MIT	*Massachusetts Institute of Technology*, p. 2
MOT	*Multi-object tracking*, p. 15
MTM	*map transformation matrix*, p. 7
MTT	*multitarget tracking*, p. 4
MVSLAM	*Multi-Vehicle SLAM*, p. 8
ONR	*Office of Naval Research*, p. 2
OSPA	*Optimal Sub-Pattern Assignment*, p. 13
PF	Filtro de Partículas, p. 9
PHD	*Probability Hypothesis Density*, p. v, 5
PPP	*Poisson Point Process* , p. 28
RFS(s)	Random Finite Set(s), p. v, vi
RSS	*Received-Signal Strength*, p. 10

SAR	*Search And Rescue*, p. 55
SCUA	Sistema de Consciência Situacional por Aquisição de Informações Marítimas, p. 56
SC	*Single-Cluster*, p. 7
SLAM	*Simultaneous Localization And Mapping*, p. v
SMC	*Sequential Monte-Carlo*, p. 6
SOT	*Single Object Tracking*, p. 53
STT	*Single Object Tracking*, p. 53
SaaS	*Software as a Service*, p. 7
SisGAAz	Sistema de Gerenciamento da Amazônia Azul, p. 12
UK-PHD	*unscented* Kalman PHD, p. 5
VO	*Visual Odometry*, p. 5
fdp	função de densidade de probabilidade, p. 23
unif	distribuição uniforme, p. 24

Capítulo 1

Introdução

Tornou-se frequente, ao se abrir um portal de notícias na *internet*, encontrar manchetes sobre empresas que pretendem distribuir seus produtos vendidos *on-line* por meio de um *drone*, exploração de petróleo em águas profundas por meio de *AUVs* (*Autonomous Underwater Vehicles*), robôs utilizados em limpeza doméstica ou sobre sistemas de transportes que operam sem a interferência humana. A autonomia é o ponto comum entre elas.

1.1 Introdução aos sistemas autônomos

Em [3], [4] e [5] são encontradas diferentes definições para veículos autônomos terrestres, subaquáticos ou aéreos. Em comum, todas afirmam que esses sistemas possuem a ausência da interferência humana no cumprimento de seu propósito ou missão. Segundo Cheng [4], veículos autônomos são plataformas robóticas móveis dotadas de quatro tecnologias fundamentais:

- Percepção e modelagem do ambiente;
- Localização e mapeamento;
- Planejamento da trajetória e tomada de decisão;
- Controle de movimento.

Uma das principais motivações para o estudo de tecnologias associadas às plataformas terrestres é a grande quantidade de acidentes envolvendo veículos, de acordo com [4]. Dessa forma, poder-se-ia tornar os sistemas de transporte terrestres mais seguros. Além disso, a exploração espacial e aplicações militares também fomentam o desenvolvimento de veículos cada vez menos dependentes da interferência humana, como no caso do projeto "batalha não tripulada" ou do *Urban Challenge* financiados pela DARPA (*Defense Advanced Research Projects Agency*) [6–8]. Dentre os grupos

de tecnologias citados acima, destaca-se a aplicação dos chamados *Filtros Bayesianos* tanto para a percepção e modelagem do ambiente quanto para o problema de localização e mapeamento. Dentre as técnicas baseadas no *framework* Bayesiano, pode-se evidenciar as baseadas nos conjuntos aleatórios finitos (do inglês, *Random Finite Sets* ou simplesmente RFS), as quais são objeto deste estudo.

1.2 Breve histórico dos veículos autônomos

De acordo com [3], o primeiro passo para os veículos autônomos foi dado com o desenvolvimento do carro controlado por rádio na década de 1920, chamado *Linriccan Wonder*. Já na década de 1950, foi introduzido o conceito de rodovias inteligentes, as quais seriam dotadas de fios com padrões reconhecidos pelo veículo. Inicialmente, essa técnica foi empregada em laboratório e, em seguida, numa rodovia da cidade de *Lincoln* (Nebraska, EUA). Tal iniciativa teve a participação da *General Motors*, que introduziu a capacidade de controle automático de direção, aceleração e frenagem valendo-se de modelos avançados [3].

Em 1966, o Laboratório de Pesquisas em Sistemas de Transportes britânico lançou o primeiro veículo "sem motorista". Esse automóvel, um Citroen DS, era capaz de regular a velocidade até $130km/h$ sem desvios de direção numa estrada dotada de cabos magnéticos para guiagem.

Já década de 1980, a *Mercedez-benz* introduziu os sistemas de visão computacional para guiagem de um carro sem motorista. Para isso, foi utilizada uma van que chegou a alcançar $63km/h$ numa via sem pedestres e veículos [3].

Na década de 1990, existiram várias iniciativas de empresas, entidades governamentais e universidades para desenvolvimento de veículos autônomos e estradas inteligentes, como o Projeto *Prometheus* [9] e o *NHOA* [10, 11], nos Estados Unidos, e o Projeto Argo, na Itália. Nesse mesmo período, destacam-se as iniciativas em aplicação militar financiadas pela DARPA, segundo[3]. No campo das aplicações subaquáticas, salienta-se o desenvolvimento dos chamados robôs autônomos submarinos pelo MIT (*Massachusetts Institute of Technology*) financiados pela ONR (*Office of Naval Research*), órgão de fomento à pesquisa da Marinha dos EUA, por exemplo [12–14].

Nos anos 2000, entrou em operação na Holanda o primeiro sistema de transporte público autônomo: o *ParkShuttle*[15]. Além disso, o governo americano, por meio de seu exército, faz uso experimental de ALVs (*Autonomous Land Vehicles*) em operações de guerra como o Demo III [16].

1.3 O problema de mapeamento e localização em robótica móvel

Dada a classificação definida em [4], a localização e mapeamento constituem um dos blocos fundamentais no desenvolvimento de veículos ou robôs autônomos. Pois para realizar essas tarefas, o sistema autônomo necessita obter informações sobre o mundo ao seu redor para poder fornecer subsídios para os sistemas de guiagem do robô.

Quando o mapa já é conhecido e o robô necessita saber (ou estimar) onde se encontra, por meio de seus diversos sensores, tem-se o que é chamado de **localização**. A tarefa de construção de um modelo do ambiente é chamada de **mapeamento**. Uma vez que os processos de localização e mapeamento são solucionados simultaneamente, tem-se o que se chama SLAM (*Simultaneous Localization And Mapping*).

Segundo [17], a representação computacional, bem como das incertezas envolvidas, desses processos é crucial para a navegação autônoma.

De acordo com Stachniss et al. [18], o problema de SLAM surgiu no início da década de 1990 e até hoje se apresenta como um dos maiores desafios em robótica móvel e sistemas autônomos. Tradicionalmente, são utilizadas três técnicas para se resolver esse problema: filtros de Kalman, filtros de partículas e as baseadas em teoria dos grafos [19].

De acordo com Agunbiade e Zuva [20], pesquisadores têm estudado extensivamente esse problema computacional da navegação autônoma e proposto melhorias. Além disso, há uma constante transição dessa tecnologia para as indústrias. No entanto, ainda existem contratempos limitando a aceitação total da autonomia na navegação, apesar de pesquisas terem sido conduzidas nos últimos 30 anos. Tais dificuldades podem ser de natureza técnica (custo computacional elevado, processamento em tempo real, limitação de energia etc.), aspectos legais ou outros fatores. Para se ter ideia, mesmo a escolha do sensor adequado permanece como fator crítico para o desafio da autonomia a longo prazo, como trata [21].

Apesar das técnicas ditas vetoriais (por exemplo, o Filtro de Kalman) serem bem estabelecidas, elas apresentam uma dificuldade extra em aplicações reais: a associação de dados. De acordo com [19], trata-se de algo bastante complexo e deve ser solucionada à parte do problema de SLAM, entretanto, pode afetar diretamente o desempenho do algoritmo de localização e mapeamento. Logo, soluções alternativas, que lidem de maneira mais simples com a questão da associação de dados, são de interesse para os problemas de localização e mapeamento.

Nesse contexto, a teoria dos conjuntos aleatórios finitos (do inglês, *Random Finite Sets* - RFS) vem sendo estudada e, de acordo com [17], é uma ferramenta mais apropriada para a solução do problema de mapeamento (ou SLAM) quando

comparada às técnicas convencionais.

1.4 O problema de rastreamento de múltiplos alvos

De acordo com Bar-Shalom e Li [22], a estimação é o processo de selecionar um ponto de um espaço contínuo — a "melhor escolha" de acordo com alguns critérios; rastreamento é a estimativa do estado de um objeto em movimento com base em medições remotas, usando "N" sensores em locais fixos ou em plataformas móveis; e filtragem é a estimativa do estado atual de um sistema dinâmico — a razão para o uso da palavra "filtro" é que o processo para obter a "melhor estimativa" de dados ruidosos equivale a "filtrar" o ruído.

Em geral, o objetivo do *multitarget tracking* (MTT) é estimar conjuntamente, a cada tempo de observação, o número de alvos e suas trajetórias a partir de dados de sensores. Mesmo em um nível conceitual, o MTT é uma extensão não trivial do rastreamento de alvo único [23]. Esse problema tem dois objetivos: (1) estimar o número aleatório de alvos que estão presentes na área de interesse e (2) estimar o vetor de estado aleatório de cada alvo [24].

Nesse contexto, pelos mesmos motivos do caso do problema de mapeamento, as técnicas de MTT baseadas em RFS mostram-se vantajosas em relação às vetoriais, pois não há o problema derivado de associação de dados. O rastreamento em RFS usa uma coleção de vetores de estado, tratando os elementos como variáveis aleatórias, bem como o número de elementos nessa coleção em si.

1.5 Revisão da literatura

O presente trabalho tem como base o estudo das seguintes áreas: *multitarget tracking*, mapeamento e localização em robótica, RFS aplicado ao problema de mapeamento e localização, bem como o *multitarget tracking* em sistemas cooperativos (aqueles compostos por mais de um agente trabalhando colaborativamente).

Portanto, boa parte da revisão objetivou conhecer as principais técnicas utilizadas para mapeamento e localização em robótica baseadas no *framework bayesiano*. Tais técnicas são amplamente difundidas na literatura, as quais são ditas vetoriais. Também foi analisada a literatura que aplica RFS na solução do problema de mapeamento e localização. Por fim, o estado da arte em técnicas cooperativas é revisado.

Já em relação ao problema de rastreamento de múltiplos alvos (*multitarget tracking*), a mesma estratégia é utilizada. Inicialmente, são abordados os algoritmos vetoriais. Posteriormente, a busca na literatura sobre filtros RFS em sistema de *tracking* (incluindo-se os cooperativos).

1.5.1 Mapeamento e localização em robótica

Em [25], uma revisão histórica sobre o problema de SLAM é apresentada. Situa-se o leitor na origem do problema de SLAM, na conferência 1986 *IEEE Robotics and Automation Conference* realizada em São Francisco. Afirma-se nesse evento que métodos probabilísticos estavam sendo introduzidos nas áreas de robótica e inteligência artificial. Além disso, a formulação e estruturação para o problema de SLAM no contexto probabilístico são definidas. Por fim, algoritmos como o EKF-SLAM (*Extended Kalman filter* SLAM), FastSLAM e o filtro de partículas Rao-Blackwellized são definidos, bem como realizações *opensource* em diferentes linguagens de programação são indicadas. Já em [26], os temas de estado da arte, à época, para o SLAM são apresentados, dentre os quais é exposto o problema de *Multihypothesis Data Association* como um dos principais desafios para a área. Um estudo (e revisão) acerca de filtros estocásticos do ponto de vista do *framework bayesiano* é exibido em [27]. São definidos o filtro de Kalman e suas variações, além de filtros não paramétricos, como o de partículas, bem como versões baseadas em mistura gaussiana desses mesmos filtros. Também são apresentadas aplicações e referências em diversas áreas do conhecimento, inclusive no problema de localização em robótica móvel.

Em [28], realiza-se uma análise dos estudos e de temas em SLAM examinados no período de 2010 a 2016. Apresenta-se, por exemplo, o vSLAM, que utiliza técnicas de hodometria visual (VO, do inglês *Visual Odometry*) no processo de estimação de mapas 3D. Também são abordados diferentes métodos para mapas baseados em características (*Feature based maps*) combinados à VO. Nesse mesmo estudo, são apontados problemas ainda em aberto, como: a iniciação do mapa e o problema de escala para a determinação das referências absolutas.

O problema de rastreamento linear e não linear de alvos é apresentado em [2]. Como solução, propõe-se um novo algoritmo, formulado no contexto bayesiano e em *Random Finite Sets Statistics*. Para a tratabilidade computacional do problema, adota-se uma realização baseada em um filtro PHD (*Probability Hypothesis Density*), no qual se propaga recursivamente a mistura gaussiana. Para o caso não linear, duas variações do algoritmo baseado em mistura gaussiana PHD (GM-PHD) são apresentadas: o *extended* Kalman PHD (EK-PHD) e o *unscented* Kalman PHD (UK-PHD). O mesmo problema é tratado em [29], adicionalmente, o GM-PHD mostrou-se bem-sucedido em estimar o número correto de alvos e suas trajetórias em alta densidade de sinais espúrios ou falsos (*clutter*), além de apresentar melhor desempenho que o filtro MHT (*Multiple Hypothesis Tracker*), até então o mais utilizado na área de estimação de alvos.

Em [30], uma nova formulação baseada em RFS é apresentada, chamada cardina-

lizada. Essa nova e matematicamente rigorosa recursão *bayesiana* lida formalmente com os problemas de múltiplas medições geradas pelo alvo, o problema do campo de visão e vetor de estado do sensor, bem como com medições espúrias. Para o caso linear, tal formulação possui solução analítica fechada.

A aplicação em robótica dos conjuntos aleatórios finitos é tratada pela primeira vez em [31]. Esse trabalho apresenta uma nova formulação para o FB SLAM (*Feature-Based* SLAM), na qual o mapa é tratado como um conjunto de cardinalidade aleatória ao invés de ser visto como um vetor de dimensão desconhecida. Essa nova proposta foi baseada nas similaridades entre o problema do robô ao se deslocar em um ambiente desconhecido e o rastreamento de um radar em um ambiente no qual o número de alvos é desconhecido. Essa abordagem, até então desconhecida na comunidade de robótica, é apresentada como uma alternativa às formulações convencionais, pois evita o problema de associação de dados.

Já em [32], o filtro PHD é apresentado como derivado da função intensidade sendo aplicado em um mapa do tipo *occupancy grid*, baseado em localização (*location-based*) que carrega a informação dos espaços vazios e ocupados do mapa.

Em [17], [33] e [34], apresentam-se novas realizações para filtros baseados em RFS, como o SMC (*Sequential Monte-Carlo*) PHD e o filtro Rao–Blackwellized PHD *particle-based*. Nesses trabalhos, o problema de estimação é considerado independente do problema de localização. No subproblema de localização, é utilizado um filtro de partículas, aumentando o desempenho do algoritmo.

Na área de localização, em [35], a teoria dos RFSs é combinada com VO e, com uma única câmera, é aplicada na localização de um veículo *on-road*, auxiliando no seguimento de trajetória de automóvel. Trata-se da primeira aplicação em um ambiente real urbano, cenário caracterizado pela alta taxa de ocorrência de medições espúrias.

Já na área de mapeamento, em [36], aplica-se RFS na detecção de *landmarks* no tráfego real urbano. Trata-se da primeira aplicação nesse contexto. Para isso, utiliza-se uma plataforma *off-the-shelf* (Iphone4) para aquisição de sinais de imagem, giroscópio e GPS.

Em [37], são explorados os conceitos de mapeamento sob o contexto do *framework bayesiano* em conjunto com a teoria dos RFSs. Nesse trabalho, são apresentadas aplicações na geração de um mapa $3D$, com dados obtidos por meio de radar instalado em um robô móvel terrestre e no mapeamento de costa (detecção de rochas, embarcações e outras *features*), informações obtidas por intermédio de um caiaque adaptado.

Em [38], [39] e [40], são apresentadas aplicações do *Single-Cluster* PHD (SC PHD) no problema de SLAM de um veículo autônomo submarino em um ambiente controlado (tanque de testes). No primeiro trabalho, o SC PHD é comparado a

um filtro EKF e apresenta resultados superiores na estimação da localização. Em [39], apresenta-se o primeiro trabalho utilizado para estimação de *features* estáticas e dinâmicas em adição à localização do AUV. No último trabalho, os algoritmos RB PHD e o SC PHD são comparados, tendo o SC obtido resultados superiores na localização do veículo. Além disso, todos os algoritmos em RFS equiparam-se, ou superam, o FastSLAM (técnica mais utilizada) em relação ao custo computacional e desempenho.

Em [41], estuda-se o mapeamento cooperativo utilizando um grupo de AUVs. O problema é abordado de tal maneira que apenas um dos veículos é responsável pelo mapeamento e localização de cada um deles, esse chamado *master* e os demais de *slaves*. As estimativas são obtidas a partir da medição de todos os veículos e por meio de um filtro EKF. Apesar de a ideia ser voltada para AUVs, todos os testes foram realizados com robôs terrestres.

Lee et al. [42] expõem um levantamento de técnicas de fusão (*merge*) de mapas para *cooperative*-SLAM (C-SLAM). Nesse estudo, as técnicas propostas são classificadas em duas categorias: fusão direta (*direct map merging* - DMM) e fusão indireta (*indirect map merging* - IMM). DMM consiste em obter a matriz de transformação dos mapas (*map transformation matrix* - MTM) diretamente dos sensores de distância ou câmeras. Já o IMM consiste em obter a MTM por meio das partes comuns de cada mapa individual elaborado por cada robô. Após a apresentação dessas técnicas, são discutidas as vantagens e desvantagens de cada uma delas no contexto de precisão e de tempo de computação.

Diferentes trabalhos, como [43], [44], [45], [46] e [47], propõem novas técnicas para solucionar a fusão de mapas por intermédio de métodos diretos ou indiretos. Em comum, esses estudos focam na solução de *merge* e já recebem os mapas obtidos por meio de um filtro vetorial convencional (Kalman, EKF, UKF e partículas).

Em [48], dois métodos de fusão de mapas são avaliados para o problema de mapeamento cooperativo de AUVs. Por se tratar de uma comunicação submarina, um ponto crítico a ser considerado é a restrição no canal de comunicação. Nesse estudo, é proposto um algoritmo baseado em grafos responsáveis por gerar pacotes capazes de transmitir otimamente a informação. Por fim, o algoritmo é validado numa operação em ambiente real.

Em [49], a proposta de um *framework* baseado *cloud computing* para serviços em robótica é apresentada. Dentre os serviços disponibilizados está o de SLAM, o qual é implementado por meio do algoritmo FastSLAM. A ideia é que o mapa global possa ser compartilhado com outros (novos) robôs introduzidos no ambiente por meio de um modelo de software como serviço (SaaS - *Software as a Service*). Isso reduz a carga de exploração e construção de mapas para um novo robô e minimiza a necessidade de sensores adicionais.

Em [50], uma visão geral (arquiteturas e aplicações) sobre *cloud* em robótica é oferecida. Esse conceito tem sido bastante explorado nos problemas de SLAM, como é possível consultar em [51], [52], [53], [54] e [55].

O mapeamento cooperativo de *landmarks* (detecção e localização de alvos) com o uso de *Random Finite Sets* é abordado em [56]. Nessa tese, expõe-se a coleta de dados do ambiente dos robôs, que poderiam estar operando em ambiente de elevado risco ao ser humano, como inspeção de uma infraestrutura, exploração em minas, segurança e vigilância, monitoramento ambiental ou busca e salvamento. Os robôs comunicam-se com uma central de processamento, que realiza cálculos de estimativa e controle, ou agem de forma descentralizada. Por fim, simulações digitais validam a estrutura unificada de estimativa e controle.

Zhang, Buckl e Knoll [57] estudam o problema da localização cooperativa de múltiplos veículos a partir da formulação de um filtro SMC PHD. Nesse trabalho, em vez de uma arquitetura centralizada, uma solução descentralizada é investigada, na qual cada veículo é um centro de fusão de dados e cada um deles lida apenas com informações locais (apenas os vizinhos observados).

Uma arquitetura descentralizada, no mapeamento cooperativo utilizando RFS, também é abordada em [56]. Nesses trabalhos, são consideradas duas abordagens para a expansão do problema do SLAM com um único robô para o multi-veículo SLAM (MVSLAM). Na primeira delas, a posição inicial dos robôs é conhecida, enquanto na segunda é desconhecida. Em ambos os casos, utilizam-se RFS para representação dos mapas e um filtro PHD para a estimação das posições dos objetos. Já para a localização dos robôs, utiliza-se um filtro de partículas.

Em [58], o problema de localização e rastreamento de um time de futebol de robôs é analisado. Nesse trabalho, um algoritmo de rastreamento multi-alvos com RFS é utilizado para a solução do mapeamento global da equipe. Para isso, utiliza-se uma realização GM PHD para a construção dos mapas locais e uma etapa extra de *prune* & *merge* em cada um dos robôs.

Dames [59] propõe um algoritmo (utilizando o filtro GM PHD) distribuído de estimativa e controle, o qual permite que robôs móveis cooperativos procurem e rastreiem um número desconhecido de alvos. Esses alvos podem estar parados ou em movimento, e o número de alvos pode variar ao longo do tempo à medida que entram e saem da área de interesse. Trata-se de um problema clássico de *multitarget tracking* aplica à robótica móvel, destacando a relevância do filtro PHD para ambos os problemas.

Em [60], trata-se um cenário no qual um grupo de quadricópteros deve rastrear vários alvos estacionários em um ambiente desconhecido e limitado enquanto realizam uma *random walk*. Por meio de uma rede de comunicação, esses veículos podem transmitir suas estimativas das localizações dos alvos para outros quadricópteros,

essa estimativas são realizados com um filtro PHD.

1.5.2 *Multitarget tracking*

Muitos algoritmos e métodos para rastreamento de alvos estão disponíveis na literatura. Técnicas de filtragem recursiva sob uma estrutura de estimativa Bayesiana são comumente usadas para resolver o problema de rastreamento usando uma abordagem estocástica. Esses algoritmos e métodos incluem extensões do conhecido Filtro de Kalman (KF): O filtro *Extended* Kalman (EKF), filtragem *Interactive Multiple-Model* (IMM) e filtragem IMM de estrutura variável (VS-IMM), além de outras técnicas de filtragem, que incluem abordagens baseadas em grade, bem como técnicas não paramétricas [61].

O filtro de Kalman é o estimador ótimo para sistemas lineares estocásticos com ruído branco. Ao lidar com um problema de estimação não linear, essa técnica apresenta limitações de desempenho substanciais e nenhuma garantia de convergência. As variantes estendida e *unscented* contornam os problemas de não linearidade aplicando aproximações. Esses algoritmos são mais eficazes ao enfrentar problemas unimodais. O desempenho dos métodos baseados no Filtro de Kalman diminui significativamente quando aplicados a um problema multimodal, mesmo nos casos mais simples. É justamente o problema considerado em uma região de vigilância marítima: modelos não lineares e distribuições multimodais.

O problema de MTT é multimodal, ou seja, usualmente envolvem uma distribuição de probabilidade com mais de uma moda. Uma maneira estabelecida de superar as limitações dos métodos paramétricos é alcançada com o rastreamento baseado no Filtro Multi-Hipótese de Kalman (MHKF) ou no Filtro de Partículas (PF). O objetivo do MTT é estimar conjuntamente, a cada instante de observação, o número de alvos e suas trajetórias a partir de dados de sensores. Mesmo em um nível conceitual, o MTT é uma extensão não trivial do rastreamento de alvo único. De fato, o MTT é muito mais complexo tanto na teoria quanto na prática [23].

Embora essas técnicas de MTT vetoriais estejam bem estabelecidas, elas apresentam o problema da associação de dados como uma dificuldade extra, a qual pode ser até mais complexa. A falta de uma solução global ótima para estimar os estados dos alvos e a ausência de um filtro Bayesiano sem algum método heurístico intermediário exemplificam tal complexidade. Alternativamente, buscando lidar com essa fragilidade, Mahler [62] propõe um algoritmo MTT baseado na teoria dos RFS. A abordagem proposta unifica o problema de MTT em um único procedimento probabilístico, que contém detecção, correlação, rastreamento e classificação. O arcabouço matemático *Finite-set Statistics* (FISST) que suporta a abordagem RFS é detalhado em [63–65].

Apesar de todas as vantagens teóricas do RFS, o filtro de Bayes recursivo baseado em RFS, mesmo em problemas de alvo único, é tão desafiador computacionalmente que requer uma solução aproximada para torná-lo implementável. Com essa motivação, Mahler [66] apresenta o filtro PHD, que fornece o número de alvos esperados a partir da integração da função intensidade. Considerando sua natureza recursiva, ao propagar as estatísticas de momentos de primeira ordem do PHD, torna-se computacionalmente atraente. Em seu estudo do filtro PHD, Vo e Ma [67] propõem uma solução analítica para a recursão PHD de alvos com dinâmica linear. Clark e Bell [68] analisam a convergência para a aproximação *Sequential Monte Carlo* (SMC) usando um filtro de partículas, enquanto Clark e Vo [69] consideram as realizações da Mistura Gaussiana (GM) sob processos estocásticos lineares ou não lineares.

Existem diversas aplicações abordadas na literatura para os métodos mencionados que demonstram seu uso para rastreamento de vários alvos em um ambiente de alto nível de *clutter* (ou falsos alarmes) como [70–73] para SMC e [2, 29, 74–76] para implementações GM. Diferentes trabalhos [77–79] apresentam a versão Cardinalizada (CPHD), que é uma generalização da recursão PHD, propagando conjuntamente a intensidade posterior e a distribuição de cardinalidade posterior, enquanto a recursão PHD propaga apenas a intensidade. Mahler [80, 81] apresenta mais avanços nas aproximações de PHD.

Em relação às aplicações em sistemas de rastreamento em áreas de interesse marítimo, Pace [82] implementa o filtro SMC e GM PHD para um radar 3D aplicado a cenários aéreos e navais tridimensionais. Esse trabalho ilustra e compara o desempenho dos filtros na detecção, iniciação e finalização de trilhas com presença de elevado nível de *clutter*. Também relacionado ao mesmo tipo de aplicação, Do et al. [83] propõem um método para rastreamento online de múltiplos alvos para uma área naval usando um filtro *Generalized Labeled Multi-Bernoulli* (GLMB).

Em relação a sistema de *tracking* em redes de sensores distribuídos, Laneuville e Houssineau [84] abordam o problema de rastreamento multitarget com dados passivos, obtidos por câmeras geograficamente distribuídas usando um algoritmo baseado em mistura Gaussiana. Battistelli et al. [24] tratam do CPHD aplicado ao rastreamento distribuído de vários alvos em uma rede de sensores de nós heterogêneos e geograficamente dispersos com recursos de detecção, comunicação e processamento. Pailhas et al. [85] propõem uma rede de sonares MIMO (*Multiple-Input Multiple-Output*) para vigilância de área portuária, protegendo-a de intrusão submarina. Gulmezoglu et al. [86] investiga o uso de filtros GM PHD para rastreamento de várias pessoas usando uma rede de sensores de radar em um ambiente interno. Dias e Bruno [87] apresentam um novo algoritmo de filtro de partículas cooperativo para rastreamento de emissores usando medições de *Received-Signal Strength* (RSS) considerando o custo da comunicação.

A melhoria de desempenho desses algoritmos de rastreamento ganha importância significativa quando em sistemas de rastreamento cooperativo. Brancalion e Dias [88] apresentam uma abordagem conceitual para um sistema distribuído de fusão de dados, utilizado para vigilância marítima, com ênfase ao impacto na grande quantidade de dados no acréscimo de mais e mais nós à rede e na necessidade de melhor desempenho tanto nos nós quanto na comunicação. De maneira similar, Liu et al. [89] enfatizam a relevância e o impacto da Internet das coisas (IoT) e dos sensores embarcados, bem como da eficiência dos métodos de fusão de dados para veículos autônomos de superfície, os quais poderiam compor uma rede de vigilância cooperativa.

1.6 Motivação

Apesar de o problema de mapeamento e localização ter sido estudado ao longo das últimas décadas, possibilidades de contribuição permanecem em aberto. De acordo com [90], existem muitos algoritmos de SLAM para um robô operando isoladamente, os quais são eficazes e aplicados na prática. No entanto, a pesquisa sobre esse problema para um grupo de robôs torna-se mais relevante devido à expansão de seu uso em diferentes esferas, rápido desenvolvimento de *hardware* e evolução do *software*, tornando-se, portanto, uma promissora linha de pesquisa.

Também segundo [90], ainda não existe um algoritmo de navegação amplamente aceito, comprovado e confiável para grupos de robôs. Apesar de a comunidade científica estar interessada no tema, as contribuições empregadas não se tornam populares porque os autores não publicam os detalhes de implementação de suas abordagens (código-fonte) ou porque os próprios algoritmos não são elaborados o suficiente para serem uma solução universal.

Apesar das técnicas baseadas em *Random Finite Sets* (RFS) possuírem a ampla difusão na comunidade científica de estudo das tecnologias voltadas para radar e *tracking*, a aplicação em robótica é pouco estudada, quando comparadas às técnicas vetoriais, pela comunidade de robótica. Adicionalmente, o uso de RFS para robótica móvel é algo recente, o que é evidenciado pelo grande número de publicações recentes e pela grande quantidade de aplicações em que esse método seria mais eficiente e ainda não está implementado.

Além disso, uma das possíveis aplicações para as técnicas de SLAM em robótica cooperativa seria na área de segurança e defesa, tal qual em [91], [92] e [93]. Tais trabalhos abordam os desafios tecnológicos e operacionais que envolvem a natureza de operações de abordagem e patrulha costeiras e propõem soluções em lanchas autônomas. Aliado a isso, em 2018, deu-se início ao embrião do Sistema de Gerenciamento da Amazônia Azul (SisGAAz), o qual pretende monitorar a denominada

Amazônia Azul, uma área de $3,6$ milhões de quilômetros quadrados na costa brasileira. Esse sistema abrangerá inicialmente a Baía de Guanabara, região conhecida como uma das portas de entrada para armas e drogas no município do Rio de Janeiro. Nesse desenvolvimento inicial, é esperado um aporte em torno de 40 milhões de reais. Tal sistema poderia, no futuro, contar com lanchas autônomas integrando uma rede de sensores fixos (além dos embarcados), garantindo a vigilância constante e com segurança para os operadores humanos.

Por fim, surge o crescente interesse na otimização e melhoria de desempenho e do impacto da Internet das Coisas nas redes de sensores tanto de veículos autônomos quanto em sistemas de tracking. Essa demanda aborda o problema sob duas óticas. A primeira delas é a melhoria e um menor custo na comunicação entre os elementos da rede. Já a outra vertente, foca na melhoria dos algoritmos e esquemas de fusão de dados e no aumento de desempenho no processo dos nós, sendo essas abordagens objeto de estudo neste trabalho.

1.7 Objetivo

Este trabalho tem por fim propor melhorias de desempenho para sistemas cooperativos de rastreamento de múltiplos alvos em tempo real, bem como no mapeamento e na localização de múltiplos robôs por meio de técnicas baseadas *em Random Finite Sets*.

Nesse contexto, tem-se como objetivo específico contribuir para uma proposta de melhoria do desempenho computacional em um sistema cooperativo e, para isso:

- propor um esquema de fusão de dados para redes de sensores de sistemas de rastreamento e vigilância;
- avaliar por intermédio de simulações uma nova abordagem de paralelização para o filtro GM PHD.

1.7.1 Publicações

O presente trabalho resultou na publicação em congresso nacional especializado:

- LIMA, Kleberson Meireles de; COSTA, Ramon R.; PEREIRA-DIAS, Diego. **Aplicação de Conjuntos Aleatórios Finitos no Problema de Localização e Mapeamento de Robôs Móveis**. In: Anais do XXII Congresso Brasileiro de Automática: CBA2018. https://www.sba.org.br/open_journal_systems/index.php/cba/article/view/901/845.

Além disso, também foi publicado um trabalho em periódico internacional especializado:

- Lima, K. M. de, & Costa, R. R. (2022). **Cooperative-PHD Tracking Based on Distributed Sensors for Naval Surveillance Area**. In: Sensors (Vol. 22, Issue 3, p. 729). MDPI AG. https://doi.org/10.3390/s22030729.

1.7.2 Contribuições

Com base nas publicações realizadas e submetidas, pode-se indicar as seguintes contribuições deste trabalho:

(i) fomento e difusão das técnicas baseadas em RFS na área de robótica móvel, dado que [94], em nosso conhecimento, foi a primeira publicação dessa abordagem em congresso e/ou publicação brasileira;

(ii) novo esquema de fusão para uma rede de sensores dotada de uma estação central. Nesse esquema, inclui-se uma etapa adicional de "prune & merge" ao algoritmo original do filtro GM PHD. Com isso, é possível superar as limitações dos sensores e do algoritmo PHD original. O esquema proposto apresenta bons resultados, atestados pela métrica *Optimal Sub-Pattern Assignment* (OSPA) nas simulações realizadas. Além disso, a oclusão do alvo é superada, sendo uma das principais fraquezas do algoritmo GM PHD;

(iii) proposta de utilização do modelo paralelo computacional Gamma, como uma abordagem potencial para acelerar o processamento local em um sistema de rastreamento *multitarget* distribuído, ilustrando esse potencial usando ferramentas de computação paralela do MATLAB. Até onde sabemos, este é o primeiro trabalho que explora o modelo Gamma e a implementação paralela do GM PHD.

1.8 Definição do problema

1.8.1 Mapeamento e localização

Localização é o problema de descobrir onde objetos (tais como: obstáculos e outros veículos) estão, inclusive o próprio robô. O conhecimento disso é o cerne de qualquer interação física bem sucedida com o meio ambiente [95].

Segundo [96], **localização**, em **robótica móvel**, é o problema de se determinar a *pose* de um robô em relação a um mapa do ambiente.

A localização pode ser vista como um problema de transformação de coordenadas, com o mapa sendo o sistema de coordenadas global. Portanto, a localização é o processo de estabelecer uma relação entre as coordenadas globais e as locais do robô.

Contudo, de acordo com [96], em geral, a *pose* não pode ser medida diretamente. Ou seja, a maioria dos robôs não possui um sensor para medi-la. Logo, a *pose* deve ser inferida a partir de dados (por exemplo, por um sensor de distância). Por conseguinte, no problema de localização, o objetivo é estimar a posição do robô, dados: o mapa do ambiente M, as observações z_t e a ação de controle u_t.

Segundo [96], o **mapa** é uma lista de objetos com as respectivas localizações em um ambiente, o qual é definido formalmente por:

$$m = \{m_1, m_2, \ldots, m_N\}, \tag{1.1}$$

onde N é o número total de objetos no ambiente e cada m_i, com $i = 1 : N$, representa as propriedades ou características de um objeto. Há duas abordagens para a construção de mapas: as baseadas em características (*feature-based*) e as em localização (*location-based*). Em consonância com [96], mapas baseados em localização são volumétricos e carregam a informação não apenas sobre os espaços ocupados por objetos, mas também por espaços vazios. No caso planar $m_i = m_{x,y}$, o que evidencia tratar-se de uma coordenada (x, y). Já nas abordagens baseadas em características, o mapa define a localização de cada objeto e uma (ou mais) característica. Esse estudo concentra-se na abordagem baseada em características (*Feature Based Robotic Mapping* – FBRM). De acordo com [96], um vetor de características é definido por:

$$f(z_t) = \left\{f_t^1, f_t^2, ...\right\} = \left\{(r_t^1, \phi_t^1, s_t^1), (r_t^2, \phi_t^2, s_t^2), ...\right\}, \tag{1.2}$$

onde r é a distância para a origem de um sistema de coordenadas, ϕ é a orientação em relação à mesma origem e s é a assinatura, a qual pode ser um valor numérico (ex: para caracterizar uma cor) ou um valor multidimensional (ex: altura e cor).

No contexto de filtragem de Bayesiano, o problema de estimação pode ser definido como a distribuição de probabilidade condicional

$$p(m|z_{1:k}, x_{1:k}), \tag{1.3}$$

onde m é o mapa, $z_{1:k}$ são as observações ao longo da trajetória percorrida pelo robô e $x_{1:k}$ é a trajetória do robô.

Já o problema de SLAM, caracteriza-se por ser uma combinação dos dois problemas, portanto, estimar o mapa e a localização do robô simultaneamente.

1.8.2 *Multitarget tracking*

Como já apresentado anteriormente neste capítulo, o problema de MTT consiste em estimar recursivamente e conjuntamente o número de alvos e suas trajetórias

a partir de dados de sensores. Esse problema tem dois objetivos: o de estimar o número de alvos na área de interesse, bem como estimar o vetor de estado de cada um deles. O MTT ou MOT (*Multi-object tracking*) é caracterizado pelo seguinte:

- número de alvos, ou objetos, variante no tempo e desconhecido;
- detecções perdidas: quando um objeto não é detectado por um sensor, apesar de estar no campo de visão dele;
- falsos alarmes (ou *clutter*): detecções que não são causadas por qualquer objeto ou alvo.

Diferente das técnicas vetoriais, as quais dividem o problema em casos desacoplados de alvo único, o rastreamento RFS usa uma coleção de vetores de estado, tratando os elementos como variáveis aleatórias, bem como o número de elementos nessa coleção em si. Matematicamente, o objetivo é obter as funções de densidade de probabilidade conjunta (multitarget) *a posteriori*:

$$p(\mathbf{X}_k, \mathbf{Z}_k), \tag{1.4}$$

onde $\mathbf{X}_k$ e $\mathbf{Z}_k$ são, respectivamente, os conjuntos finitos aleatórios de estado e das observações no instante k. Esses conjuntos serão mais detalhados no capítulo seguinte.

1.8.3 O problema cooperativo

No mapeamento em robótica

De maneira geral, o problema é definido como: dado um conjunto de R robôs, obter o mapa combinado do ambiente a partir dos mapas locais m^i de cada um deles. Ou seja,

$$Cmap = merge(\{m^1, m^2, ..., m^R\}), \tag{1.5}$$

onde $Cmap$ é o mapa combinado, m^i é o mapa local de cada robô e *merge* é a operação definida como capaz de combinar tais mapas a partir de um certo limiar da função de intensidade previamente definido, conforme [58]. Não há uma definição formalizada matematicamente para a operação de *merge*, ela dependerá de cada caso ou algoritmo empregado na solução.

Tal problema é conhecido como MVSLAM (do inglês, *Multi-Vehicle SLAM*), segundo [56].

No *multitarget tracking*

De maneira análoga, o problema de *multitarget tracking* cooperativo pode ser definido como:

$$D_{cooperativo} = merge(\{D(\mathbf{x})^1, D(\mathbf{x})^2, ..., D(\mathbf{x})^R\}), \tag{1.6}$$

onde R representa o número de sensores da rede de sensores distribuída e a função intensidade $D(\mathbf{x})^i$ representa a estimativa realizada independentemente pelo $i-th$ nó da rede. É importante destacar que a solução final (o *merge*) de forma distribuída e cooperativa é processada em uma estação central ou em todos os elementos da rede, a depender da arquitetura do sistema.

1.9 Organização deste texto

No Capítulo 2, a partir do problema de mapeamento em robótica, apresenta-se a abordagem baseada em *Random Finite Sets* (RFS). Após essa motivação, são apresentados a definição formal desses conjuntos e arcabouço matemático necessário para aplicação dessa abordagem no contexto de filtragem Bayesiana recursiva aplicada ao problema multi-objeto.

Então, já no Capítulo 3, é discutida a taxonomia do problema de localização e mapeamento, bem como o mapeamento e SLAM. A partir dessa contextualização inicial, é apresentado o filtro GM PHD aplicado ao problema de mapeamento em robótica, bem como as simulações e resultados para o filtro proposto.

No Capítulo 4, apresentam-se os conceitos relevantes em *multitarget tracking*, com foco na aplicação de vigilância em uma área marítima. Ainda nessa parte deste trabalho, é proposto um esquema de fusão dos dados para uma rede de sensores cooperativa aplicada numa área de vigilância marítima. Em seguida, o cenário simulado é descrito e os resultados são discutidos.

No Capítulo 5, apresentam-se se as motivações para a paralelização do filtro GM PHD, bem como o modelo computacional Gamma. A partir da apresentação desse modelo, o potencial na paralelização desse filtro é demonstrado por meio de simulações em diferentes casos.

O Capítulo 6 apresenta as conclusões deste trabalho e propostas de trabalhos futuros.

Por fim, o Apêndice A contém as publicações realizadas.

Capítulo 2

Conjuntos Aleatórios Finitos

Este capítulo visa apresentar, a partir do problema de mapeamento, a motivação para o uso de conjuntos aleatórios finitos ou, do inglês, *Random Finite Sets* (RFS), bem como o arcabouço matemático utilizado em conjunto com o *framework bayesiano*.

2.1 Conjuntos Aleatórios Finitos

De acordo com [17], as técnicas convencionais (baseadas em vetores) utilizadas em FBRM (*Feature Based Robotic Mapping*) e SLAM (*Simultaneous Localization and Mapping*) sofrem desvantagens quando aplicadas a situações realísticas. Isso acontece em ambientes nos quais um número *a priori* desconhecido de características, *landmarks* ou *targets* alvos deve ser estimado na presença de defeitos realistas dos sensores, como detecções não atendidas e falsos alarmes, além do problema desacoplado. Essas foram as principais motivações para o desenvolvimento dessa abordagem, a qual foi iniciada na comunidade de radar, fusão de dados e sistemas de *multitarget tracking*.

Uma variável RFS é uma variável aleatória que toma como valores conjuntos finitos [97]. É definido por uma distribuição discreta que caracteriza o número de elementos no conjunto e uma família de distribuições conjuntas que caracterizam a distribuição dos valores dos elementos, condicionada à cardinalidade [98]:

$$p(X) = p(|X| = n)p(X = \{x_1, ..., x_n\} \mid |X| = n). \tag{2.1}$$

Um exemplo de RFS pode ser visualizado na figura 2.1. Independente da ordem ou organização dos elementos dentro da caixa, os conjuntos de mesma cardinalidade são os mesmos. Além disso, a cardinalidade também é uma variável aleatória, uma vez que ela pode assumir diferentes valores entre 0 e 3.

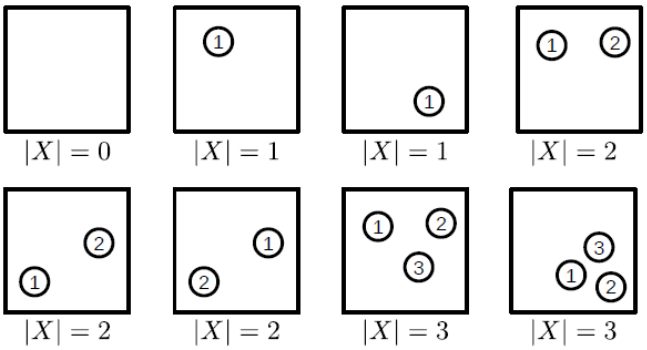

Figura 2.1: Exemplos de um RFS de 0 a 3 elementos em um ambiente quadrado. Fonte: extraído de [1].

2.2 Por que usar RFSs?

2.2.1 RFS x técnicas baseadas em vetores

Segundo [98], é preciso destacar as diferenças entre as técnicas baseadas em vetores e em conjuntos quando se considera os problemas de FBRM e SLAM, os quais podem ser entendidos como um problema *multi-target* quando os alvos são estáticos. Nesse tipo de problema a quantidade de características (*landmarks*) não é conhecida *a priori* e deve ser descoberta à medida que o robô explora o ambiente. Dois pontos-chave, que serão explicados a seguir, surgem nesse tipo de problema:

- Gerenciamento das características (*feature management*) - estimar a posição dos objetos/características no mapa;
- Associação de dados (*data association*) - atrelar as observações aos objetos/características.

Considerando o cenário hipotético, apresentado na figura 2.2, um robô atravessa os objetos m_1,..., m_7 ao longo de três trajetórias distintas X_1, X_2 e X_3.

Caso seja usado um vetor M para representar o mapa estimado, para cada trajetória teríamos $\widehat{M}_1 = [m_1\ m_2\ m_3\ m_4\ m_5\ m_6\ m_7]^T$, $\widehat{M}_2 = [m_4\ m_2\ m_3\ m_1\ m_5\ m_7\ m_6]^T$ e $\widehat{M}_3 = [m_6\ m_7\ m_5\ m_4\ m_3\ m_2\ m_1]^T$.

Como a ordem dos elementos num vetor é importante, como resultado seriam gerados três mapas diferentes. Destacando o fato de que o mapeamento seria dependente da trajetória.

Contudo, no sentido lógico, o mapeamento deveria ser independente da trajetória percorrida pelo veículo e, matematicamente, qualquer permutação nos elementos dos vetores seria uma representação válida. Por definição, a representação que captura todas as permutações dos elementos dentro de um vetor, e por conseguinte todas as características do mapa, é o conjunto finito $\widehat{\mathcal{M}} = \{m_1\ m_2\ m_3\ m_4\ m_5\ m_6\ m_7\}$ [17].

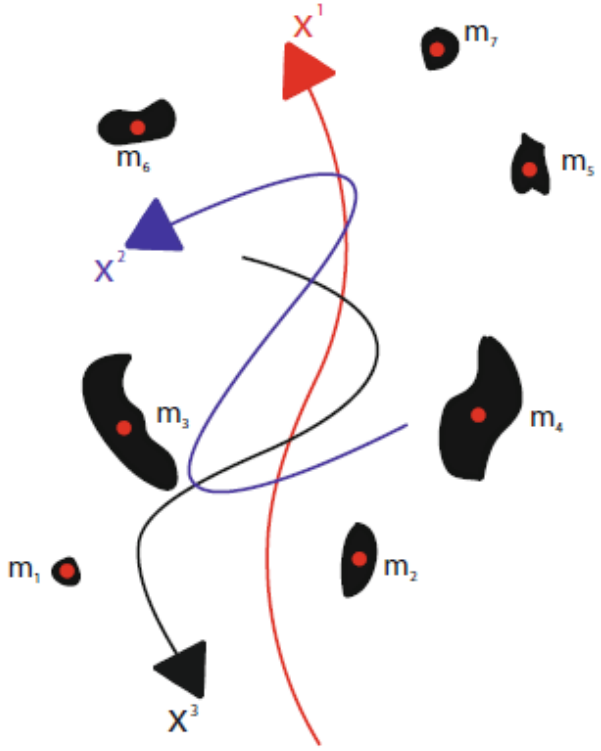

Figura 2.2: Três trajetórias realizadas por um robô em um cenário hipotético. Fonte: extraído de [1].

Outra motivação para o uso de RFS é o problema de associação entre as observações z e as respectivas características m, como pode ser visualizado na figura 2.3. Segundo [17], as técnicas de FBRM e SLAM baseadas em vetores exigem um passo de reordenação das observações antes que um filtro bayesiano (Kalman, EKF etc.) possa ser aplicado.

Por outro lado, utilizando-se de abordagem baseada em RFS, é possível fazer uma associação direta entre o conjunto $\mathcal{M}$ das características e o $\mathcal{Z}$ das observações.

Outro problema contornável pelo uso dos RFSs, quando comparado às técnicas baseadas em vetores, é o existente pelo uso de sensores não ideais associado ao crescimento do mapa por detecções falsas (*clutter*) ou objetos perdidos (fora do campo de detecção). Esse tipo de problema pode ser exemplificado a partir da figura 2.4.

Além das já citadas limitações das técnicas vetoriais, existem ainda o problema de aumento de dimensão no vetor de estado a cada vez que um novo alvo ou objeto é detectado. Supondo que as características m_1, m_2 e m_3 já tenham sido detectadas no instante $k-1$, em uma representação baseada em vetores, e que no instante k uma nova característica m_4 é detectada, ter-se-ia:

$$\widehat{M}_{k-1} = [m_1 \ m_2 \ m_3]^T$$

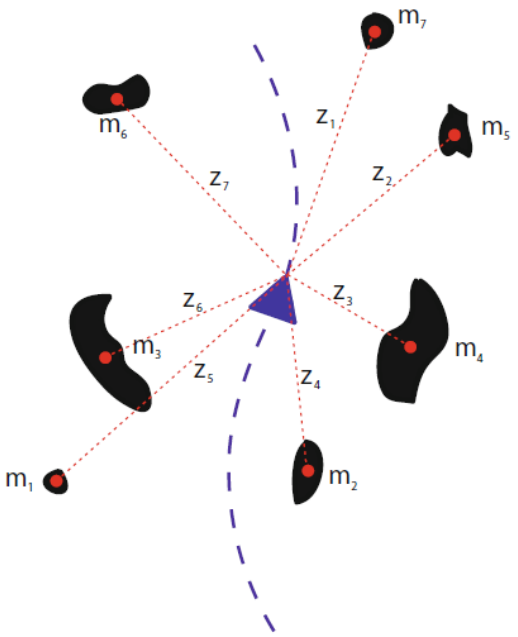

Figura 2.3: Vetores das observações diferentes a depender da trajetória.
Fonte: extraído de [1].

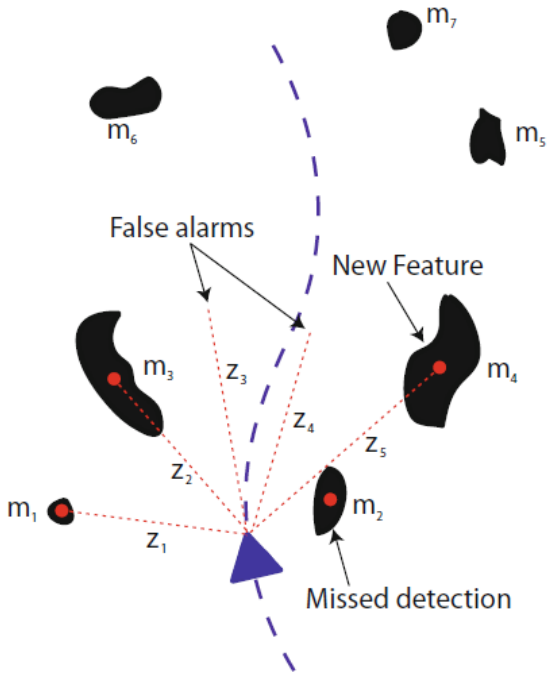

Figura 2.4: Detecção perdida com sensor realístico.Fonte: extraído de [1].

e

$$\widehat{M}_k = [m_1\ m_2\ m_3]^T \text{ “+” } m_4.$$

Segundo Mullane et al. [17], a operação de atribuição de m_4 não é clara do ponto de vista matemático. E, mesmo existindo técnicas que usem um vetor aumentado, com o uso de RFS bastaria usar a operação de união entre os conjuntos nos instantes $k-1$ e k:

$$\widehat{\mathcal{M}}_k = \{m_1\ m_2\ m_3\} \cup \{m_4\}.$$

Outro problema fundamental de qualquer FBRM ou SLAM é a necessidade de relacionar as observações ao estado estimado [17]

$$Z_k = h([m_1\ m_2\ m_3\ m_4], X_k) + \text{ruído}$$

onde $Z_k = [z_1\ z_2\ z_3\ z_4\ z_5]_k$ representa o vetor de observações no tempo k (aqui se usa o exemplo da figura 2.4), X_k é a pose do robô no instante k e h é o mapeamento (tipicamente não linear) entre a pose do robô e as observações. Esse exemplo mostra a dificuldade encontrada para mapear as quatro características (estados) com cinco observações. A observação extra é, possivelmente, devido a uma detecção espúria e uma característica foi “perdida”. Numa abordagem convencional, é indefinida a forma de integrar ao equacionamento esses dois “problemas”. É preciso um gerenciamento heurístico do mapa para remover estados em excesso e forçar o funcionamento do mapeamento.

Caso sejam utilizados conjuntos para representar as observações e mapa estimado, tem-se a seguinte relação:

$$\mathcal{Z}_k = \bigcup_{m \in \mathcal{M}_k} \mathcal{D}_k(m, X_k) \cup \mathcal{C}_k(X_k),$$

onde $\mathcal{D}_k(m, X_k)$ é o RFS das medições geradas por uma característica em m e dependente da pose X_k. Além disso, $\mathcal{C}_k(X_k)$ é o RFS das medições espúrias que poderia ser em função da pose. Portanto, $\mathcal{Z}_k = \{z_k^1, z_k^2, ..., z_k^{\mathfrak{z}_k}\}$ consiste em um número aleatório, $\mathfrak{z}_k$, de medições em que a ordem que aparecem não tem um significado físico com relação à estimativa das características no mapa. Consequentemente, de acordo com Mullane et al. [17], fica claro que, nas técnicas baseadas em vetores, a incerteza no número de características não é modelada, diferente de quando se usa RFS (a incerteza é algo intrínseco à técnica).

É importante destacar que as motivações aqui apresentadas, apesar de terem sido realizadas com exemplos na área de mapeamento, são válidas também em aplicações de detecção de objetos, vigilância etc. Em suma, as vantagens dos RFSs em relação às técnicas vetoriais são as seguintes:

- os conjuntos são invariantes em relação à ordem dos elementos;
- possuem facilidade de adicionar e/ou remover elementos;
- suprimem o problema de associação de dados desacoplados.

Apesar de todas as vantagens, é preciso aprender muitas coisas novas: RFS? conjunto de integrais? Distribuições? Modelos? Aproximações? Algoritmos? Métricas? Todas essas perguntas serão esclarecidas ao longo deste capítulo.

2.3 Conceitos-chave em FISST

Segundo Dames [98], apesar de todas as vantagens sobre as abordagens baseadas em vetores, com RFS há a dificuldade de se lidar com matemática não familiar para a maioria dos envolvidos com área de robótica. Portanto, para ser possível realizar uma inferência estatística com conjuntos, é preciso definir variáveis aleatórias adequadas e conseguir realizar operações como o cálculo de médias dessas variáveis.

2.3.1 *Definição de conjunto aleatório finito*

De acordo com Ristic, Beard e Fantacci [99], um RFS é um modelo probabilístico conveniente para a representação de múltiplos sistemas dinâmicos estocásticos (objetos ou características) e medições dos sensores. Em outras palavras, um RFS é uma variável aleatória cujos resultados possíveis são conjuntos com um número finito de elementos únicos. Isso é similar ao conceito convencional de variável aleatória, a qual é uma função que associa um número real a cada elemento do espaço amostral.

Suponha que no instante k existam n_k objetos com estados $x_{k,1}, ..., x_{k,n_k}$ assumindo valores do espaço de estados $\mathcal{X} \subseteq \mathcal{R}^{n_x}$. Tanto n_k quanto o número de estados individuais em $\mathcal{X}$ são variáveis aleatórias e variantes no tempo. O estado das múltiplas características, no instante k, é um **estado finito** definido por [99]:

$$\mathbf{X}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,n_k}\} \in \mathcal{F}(\mathcal{X}), \tag{2.2}$$

o qual pode ser modelado como um RFS em $\mathcal{X}$, em que $\mathcal{F}(\mathcal{X})$ é o conjunto finito formado por todos os subconjuntos de $\mathcal{X}$.

De maneira similar, para as observações teremos:

$$\mathbf{Z}_k = \{\mathbf{z}_{k,1}, \ldots, \mathbf{z}_{k,m_k}\} \in \mathcal{F}(\mathcal{Z}), \tag{2.3}$$

que pode ser modelado como um RFS sobre o estado de observações $\mathcal{Z} \subseteq \mathcal{R}^{n_z}$. Tanto a cardinalidade $|\mathbf{Z}_k|$ quanto os estados individuais em $\mathbf{Z}_k$ são randômicos e

$\mathcal{F}(\mathcal{Z})$ é o conjunto finito formado por todos os subconjuntos de $\mathcal{Z}$.

A depender do contexto, o estado $\mathbf{X}_k$ definido poderia ser um alvo, um pedestre ou um grupo de pedestres etc. Para fins de simplificação, será utilizado o termo "objeto" ao longo da apresentação dos conceitos.

Como já dito anteriormente, um RFS recebe valores $\mathbf{X} \in \mathcal{F}(\mathcal{X})$ e seus elementos $\mathbf{x}_k$ pertencem a $\mathbb{R}^{n_x}$. Portanto, um exemplo de realização, com elementos escalares, num instante k seria:

$$\mathbf{X} = \{\varnothing\} \rightarrow \text{nenhum objeto presente}$$
$$\mathbf{X} = \{x_1\} \rightarrow \text{um objeto, estado} x_1$$
$$\mathbf{X} = \{x_1, x_2\} \rightarrow \text{dois objetos, estados } x_1 \text{ e } x_2\,(x_1 \neq x_2).$$

Aqui se utiliza uma notação simplificada, omitindo-se o índice k tanto nos conjuntos quantos nos respectivos elementos. Ocasionalmente, essa mesma simplificação poderá aparecer no texto.

Por fim, é importante destacar que um RFS é totalmente caracterizado por sua distribuição de cardinalidade $\rho(n) = p(|X| = n)$, com $n \in \mathbb{N}$ e por uma família de distribuições conjuntas simétricas $p_n(\mathbf{x}_1, \ldots, \mathbf{x}_n) = p(X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \,|\, |X| = n)$, conforme a Equação 2.1.

2.3.2 Funções de probabilidade e distribuições de um RFS

De acordo com Mahler [100], uma função de conjunto é uma função de valor real $\Phi(T)$ definida nos subconjuntos mensuráveis T de Y, ou seja, $\Phi : Y \subseteq T \rightarrow \mathbb{R}$.

A função massa de probabilidade (ou simplesmente função de probabilidade) de um RFS $\mathbf{X}$ pertence ao espaço de estados $\mathcal{X}$ pode ser definida como [100]:

$$P_{\mathbf{X}}(S) = Pr(\mathbf{X} \in S), \tag{2.4}$$

onde P_r indica uma probabilidade e S é uma região, tal que $S \subseteq \mathbf{X}$.

Tal qual no caso escalar ou vetorial, a função de densidade de probabilidade (fdp) de um RFS, $\mathbf{X}$ é utilizada para descrever sua distribuição. A função de densidade de probabilidade $f_{\mathbf{X}}$ de um RFS $\mathbf{X}$ é definida tal que:

$$P_{\mathbf{X}}(S) = \int_S f_{\mathbf{X}}(\mathbf{x})dx. \tag{2.5}$$

Nesse contexto, a fdp de um RFS é uma função não negativa em conjuntos, que quando integrada é igual a um. Essas funções capturam tanto a distribuição sobre a cardinalidade quanto a distribuição sobre os elementos do conjunto (dada a

cardinalidade), que corresponde à fdp multiobjeto

$$f(\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}) = n!\, \rho(n) p_n(\mathbf{x}_1, ..., \mathbf{x}_n). \tag{2.6}$$

Dada a cardinalidade, é possível obter a função de densidade de probabilidade multiobjetos (simplesmente, fdp multiobjetos), denotada por $f(\mathbf{X})$, de um RFS $\mathbf{x}$, tal que [100]:

$$p_{\mathbf{X}}(\mathbf{X}) = f(\{\mathbf{x}_1, ..., \mathbf{x}_n\}) = \begin{cases} f(\mathbf{x}_1, ..., \mathbf{x}_n), & \text{se } |\{\mathbf{x}_1, ..., \mathbf{x}_n\}| = n; \\ 0, & \text{caso contrário.} \end{cases} \tag{2.7}$$

Como os conjuntos são invariáveis à ordem, as fdps também o são, por exemplo: $p_{\mathbf{X}}(\{\mathbf{x}_1, \mathbf{x}_2\}) = p_{\mathbf{X}}(\{\mathbf{x}_2, \mathbf{x}_1\})$

A seguir, são apresentados exemplos de distribuições, dadas as cardinalidades.

Exemplo 2.3.1 *RFS com um único elemento escalar.*

Se $x \in \mathbb{R}$, $x \sim \mathcal{N}(0,\, 1)$ e $\mathbf{X} = \{x\}$, temos:

$$p_{\mathbf{X}}(\mathbf{X}) = \begin{cases} \mathcal{N}(v; 0, 1), & \text{se } \mathbf{X} = v; \\ 0, & \text{se } |\mathbf{X}| \neq 1. \end{cases}$$

Então, $p_{\mathbf{X}}(\{0.3\}) = 0$ e $p_{\mathbf{X}}(\{0.2\}) = \mathcal{N}(0.2; 0, 1) \approx 0.391$.

Exemplo 2.3.2 *RFS com dois elementos escalares de distribuição uniforme.*

Se $x_1 \sim \mathbf{unif}(0,\, 1)$ e $x_2 \sim \mathbf{unif}(1,\, 2)$ são independentes, com $\mathbf{X} = \{x_1,\, x_2\}$, então:

$$p_{\mathbf{X}}(\mathbf{X}) = \begin{cases} p_1(v_1)p_2(v_2) + p_1(v_2)p_2(v_1), & \text{se } \mathbf{X} = \{v_1,\, v_2\}; \\ 0, & \text{se } |\mathbf{X}| \neq 2, \end{cases}$$

onde

$$p_1(x) = \begin{cases} 1, & \text{se } 0 < x < 1; \\ 0, & \text{caso contrário,} \end{cases}$$

e

$$p_2(x) = \begin{cases} 1, & \text{se } 1 < x < 2; \\ 0, & \text{caso contrário.} \end{cases}$$

Nesse caso, se $p_{\mathbf{X}}(\{1,25, 0.25\}) = p_1(1.25)p_2(0.25) + p_1(0.25)p_2(1.25) = 0 + 1 = 1$.

É importante destacar que se temos uma matriz ou vetor $X = \Pi[x_1, ..., x_n]$, podemos ordenar seus elementos de $n!$ formas diferentes. Portanto, se temos um

RFS $\mathbf{X} = \{x_1, ..., x_n\}$, ou seja, com mesmos elementos, temos a seguinte relação

$$n! p_X([x_1, ..., x_n]) = p_{\mathbf{X}}(\{x_1, ..., x_n\}) \text{ ou} \tag{2.8}$$

$$p_X([x_1, ..., x_n]) = \frac{1}{n!} p_{\mathbf{X}}(\{x_1, ..., x_n\}). \tag{2.9}$$

Essas relações são de particular interesse para as aproximações e deduções dos filtros recursivos.

2.3.3 Integral de conjunto

Para usar os benefícios das distribuições de probabilidade de um RFS, é preciso uma definição adequada de integrais.

Seja $f(\mathbf{X})$ uma função real sobre um conjunto, a integral de conjunto (do inglês, *set integral*) de $f(\mathbf{X})$ é definida por:

$$\int f(\mathbf{X}) \delta\mathbf{X} = \sum_{n=0}^{\infty} \frac{1}{n!} \int f(\{x_1, ..., x_n\}) dx_1 ... dx_n. \tag{2.10}$$

Essa integral caracteriza a soma sobre a cardinalidade de um conjunto, integrando todos os conjuntos possíveis de acordo com cardinalidade. Além disso, o termo $1/n!$ considera as permutações de um conjunto de tamanho n.

Exemplo 2.3.3 *Exemplo 2.3.1 revisitado: set integral de um RFS com um único elemento escalar.*

A integral de qualquer fdp multiobjeto deve ser igual a 1, portanto, para

$$p_{\mathbf{X}}(\mathbf{X}) = \begin{cases} \mathcal{N}(v; 0, 1), & \text{se } \mathbf{X} = \{v\}; \\ 0, & \text{se } |\mathbf{X}| \neq 1, \end{cases}$$

temos $\int p_{\mathbf{X}}(\mathbf{X}) \delta\mathbf{X} = \int f(p_{\mathbf{X}}(\{x_1\}) \delta\mathbf{X} = \int \mathcal{N}(x_1; 0, 1) dx_1 = 1$.

Exemplo 2.3.4 *Exemplo 2.3.2 revisitado: efeito do termo* $1/n!$.

Uma vez que

$$p_{\mathbf{X}}(\mathbf{X}) = \begin{cases} p_1(v_1)p_2(v_2) + p_1(v_2)p_2(v_1), & \text{se } \mathbf{X} = \{v_1, v_2\}; \\ 0, & \text{se } |\mathbf{X}| \neq 2, \end{cases}$$

temos

$$\begin{aligned}\int p_{\mathbf{X}}(\mathbf{X})\delta\mathbf{X} &= \sum_{n=0}^{\infty}\frac{1}{n!}\int p_{\mathbf{x}}(\{v_1,...,v_n\})dv_1...dv_n \\ &= \frac{1}{2!}\int p_{\mathbf{x}}(\{v_1,v_2\})dv_1dv_2 \\ &= \frac{1}{2}\int (p_1(v_1)p_2(v_2)+p_1(v_2)p_2(v_1))dv_1dv_2 \\ &= \frac{2}{2}\int p_1(v_1)dv_1\int p_2(v_2)dv_2 = 1.\end{aligned}$$

Valor esperado

Outra importante aplicação das integrais de conjunto é a obtenção do valor esperado. Para uma função $f : \mathcal{F}(\mathcal{X}) \to \mathbb{R}$ o valor esperado é definido por

$$\mathbb{E}[f(\mathbf{X})] = \int f(\mathbf{X})p_{\mathbf{X}}(\mathbf{X})\delta\mathbf{X}. \tag{2.11}$$

Uma aplicação do valor esperado é equação de Chapman–Kolmogorov, utilizado no passo de predição dos filtros Bayesianos, a qual será explorada mais adiante.

Distribuições de cardinalidade

A distribuição de cardinalidade de um RFS $\mathbf{X} \sim p_{\mathbf{X}}(\cdot)$ é definida por [100]

$$p_{\mathbf{X}} = Pr[|\mathbf{X}| = n] \tag{2.12}$$

$$= \int_{|\mathbf{X}|=n} p_{\mathbf{X}}(\mathbf{X})\delta\mathbf{X} \tag{2.13}$$

$$= \frac{1}{n!}\int p_{\mathbf{X}}(\{\mathbf{x}_1,...,\mathbf{x}_n\})d\mathbf{x}_1...d\mathbf{x}_n. \tag{2.14}$$

Exemplo 2.3.5 *Exemplo do cálculo de cardinalidade de um caso trivial.*

Se a distribuição de cardinalidade é dada por

$$\mathbf{X} \sim p_{\mathbf{X}}(\mathbf{X}) = \begin{cases}\mathcal{N}(x;0,1), & \text{se } \mathbf{X} = \{x\};\\ 0, & \text{se } |\mathbf{X}| \neq 1,\end{cases}$$

então

$$Pr[|\mathbf{X}| = n] = \frac{1}{n!}\int p_{\mathbf{X}}(\{\mathbf{x}_1, ..., \mathbf{x}_n\})d\mathbf{x}_1...d\mathbf{x}_n$$
$$= \begin{cases} \int \mathcal{N}(x; 0, 1)dx_1 = 1, & \text{se } \mathbf{X} = \{x\} \\ 0, & \text{se } |\mathbf{X}| \neq 1. \end{cases}$$

2.3.4 Função intensidade

A função de intensidade $D(\mathbf{x})$ (também conhecida como *probability hypothesis density* ou, simplesmente, PHD) é uma importante caracterização de um RFS $\mathbf{X}$ em $\mathcal{X}$, o qual é definido como seu momento estatístico de primeira ordem [99]. De acordo com Mahler [100], trata-se de uma função ordinária de densidade de um único objeto $\mathbf{x} \in \mathcal{X}$, portanto, de maneira intuitiva:

- o número $D(\mathbf{x})$ é a densidade de objetos em $\mathbf{x}$;
- $D(\mathbf{x})d\mathbf{x}$ é o número de objetos contidos de uma região infinitesimal $d\mathbf{x}$ centrada em $\mathbf{x}$;
- $D(\mathbf{x})$ não é uma fdp (a área sob a curva não é igual a 1).

Para definir a função PHD, é preciso primeiro definir a função *set Dirac delta* [99]:

$$\delta_{\mathbf{X}}(\mathbf{x}) = \sum_{\mathbf{w}\in\mathbf{X}} \delta_{\mathbf{w}}(\mathbf{x}), \tag{2.15}$$

onde $\delta_w(x)$ é a função delta de dirac *standard* concentrada em w.

Pode-se, então, definir a cardinalidade de um RFS como:

$$|\mathbf{X}| = \int_{\mathcal{X}} \delta_{\mathbf{X}} d\mathbf{x} \tag{2.16}$$

A função intensidade $D(x)$ pode ser definida de tal forma que a cardinalidade esperada de $\mathbf{X}$ sobre $\mathcal{X}$ é a integral:

$$\mathbb{E}\{|\mathbf{X}|\} = \int_{\mathcal{X}} D(\mathbf{x})d\mathbf{x}. \tag{2.17}$$

Lembrando que

$$\mathbb{E}\{|\mathbf{X}|\} \triangleq \int_{\mathcal{X}} |\mathbf{X}| f(\mathbf{X})\delta\mathbf{X}. \tag{2.18}$$

Combinando-se as Equações 2.17 e 2.18 (ver desenvolvimento em [99]), é possível definir

$$D(\mathbf{x}) \triangleq \{\delta_{\mathbf{X}}(\mathbf{x})\} = \int_{\mathcal{X}} \delta_{\mathbf{X}}(\mathbf{x}) f(\mathbf{X})d\mathbf{x}. \tag{2.19}$$

Com base em [98], o valor esperado da PHD corresponde ao número de características, alvos ou objetos (*landmarks*) de uma região, além de reforçar a ideia que a PHD não é uma distribuição de probabilidade. De acordo com [100], o número de objetos esperado numa região de interesse $\mathcal{X}$ é a integral da função intensidade:

$$N = \int_{\mathcal{X}} D(\mathbf{x}) d\mathbf{x}. \tag{2.20}$$

De acordo com [100], outra importante propriedade da função PHD é a da soma de intensidades. Sejam $\mathcal{X}_1, ..., \mathcal{X}_n$ RFSs independentes e $\mathcal{X}_1 = \mathcal{X}_1 \cup ... \cup \mathcal{X}_n$, então

$$D_{\mathbf{X}}(\mathbf{x}) = D_{\mathbf{X}_1}(\mathbf{x}) + ... + D_{\mathbf{X}_n}(\mathbf{x}). \tag{2.21}$$

Exemplo 2.3.6 *Exemplo de uma função intensidade unidimensional.*

Seja $D(x) = \mathcal{N}(x; 1, 1) + \mathcal{N}(x; 4, 1)$, apresentada na Figura 2.5, temos que

$$N = \int_{-\infty}^{+\infty} D(x)dx = \int_{-\infty}^{+\infty} \mathcal{N}(x; 1, 1)dx + \int_{-\infty}^{+\infty} \mathcal{N}(x; 1, 1)dx = 1 + 1 = 2.$$

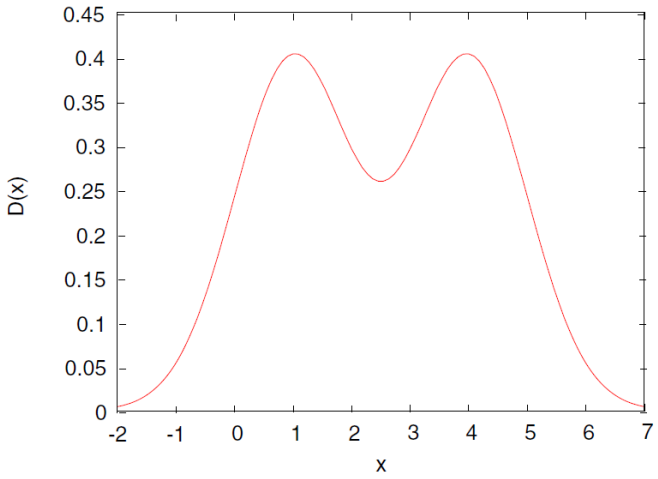

Figura 2.5: Função intensidade unidimensional.
Fonte: extraído de [1].

2.3.5 Processo de Poisson em RFSs

Para a utilização dos RFSs, é importante conhecer algumas das distribuições mais relevantes, bem como as respectivas propriedades e parâmetros. Segundo [100], o processo de Poisson (*Poisson Point Process* - PPP) é a família de RFSs central para a teoria dos filtros PHD, o qual é o objeto deste trabalho.

De acordo com [99], se cardinalidade de um RFS é independente identicamente distribuída (IID) com $\lambda > 0$

$$\rho(n) = \frac{e^{-\lambda}\lambda^n}{n!}, \text{ com } n = 0, 1, 2.... \tag{2.22}$$

então, é esse RFS é dito Poisson (ou PPP). A distribuição multiobjeto de um RFS $\mathbf{X}$ PPP é definida por [100]

$$p_{\mathbf{X}}(\mathbf{X}) = e^{-\lambda} \prod_{\mathbf{X} \in \mathcal{X}} D(\mathbf{x}), \tag{2.23}$$

onde $\lambda = \int D(x)dx$ é a taxa de Poisson e é a função intensidade $D(\mathbf{x})$. A Equação 2.23 pode ser reescrita como

$$p_{\mathbf{X}}(\{\mathbf{x}_1, ..., \mathbf{x}_n\}) = e^{-\lambda} \prod_{i=1}^{n} D(\mathbf{x}_i). \tag{2.24}$$

De acordo com [99], o RFS Poisson RFS é o único RFS que é completamente determinado por sua função de intensidade, sendo caracterizado por

$$D(\mathbf{x}) = \lambda p_{\mathbf{X}}(\mathbf{x}) \text{ e} \tag{2.25}$$

$$\mathbb{E}\{|\mathbf{X}|\} = \int D(\mathbf{x})d\mathbf{x} = \lambda. \tag{2.26}$$

Os processos de Poisson são comumente utilizados para modelar diferentes variáveis em MTT que exijam uma distribuição uniforme na região de interesse. Por exemplo, podem ser aplicadas para modelar o *clutter* e são amplamente usadas na modelagem de surgimento de novos alvos ou objetos no campo de visão de um sensor para o passo de predição do filtro PHD.

2.4 Filtro PHD

2.4.1 Filtro RFS recursivo

De acordo com Dames [98], é a abordagem mais básica usada para estimação com RFS, a qual atualiza a média recursivamente. Além disso, é o análogo do Filtro de Kalman, contudo, usa uma distribuição de Poisson.

Nessa abordagem, a estimação dos estados explora a intuição física do primeiro momento da função intensidade. Essa perspectiva é consistente com o equivalente de vários alvos do valor esperado — a expectativa de um RFS. Como dito anteriormente, a função PHD corresponde a uma densidade de massa, e a massa corresponde ao

valor esperado de alvos em alguma região do espaço de estados $S \subseteq \mathcal{X}$ [1]. Por esta razão, é possível vincular a função PHD e o framework bayesiano.

O teorema de Bayes permite calcular a probabilidade de um evento com base no conhecimento prévio das condições relacionadas ao evento e algumas novas evidências. Esse conceito é a base dos filtros ditos Bayesianos, os quais estimam recursivamente o estado atual de um sistema dinâmico de forma probabilística. Este processo consiste em duas etapas: predição e atualização (ou filtragem). Essas duas etapas são definidas de acordo com:

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1}, \tag{2.27}$$

e

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})}{\int p(\mathbf{z}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_k}, \tag{2.28}$$

onde $\mathbf{x}$ é o vetor de estado, e $\mathbf{z}$ é o vetor de observação, ambos no sentido convencional. As Equações 2.27 e 2.28 constituem o algoritmo recursivo do Teorema de Bayes. O filtro apresentado tem importância apenas conceitual. A integral na Equação 2.27, ela é tratável computacionalmente apenas para alguns casos discretos. No entanto, todos os filtros recursivos Bayesianos, como o Filtro de Kalman, são derivados dele.

De maneira análoga ao filtro Bayesiano, o filtro RFS Bayes recursivo tem como objetivo estimar a densidade posterior de um estado multiobjeto, representado pela variável RFS $\mathbf{X}_k$ [99]. Além disso, a evolução desse RFS é modelada como um processo de Markov.

Dados os RFSs do estado multiobjeto $\mathbf{X}_k$ e das observações $\mathbf{Z}_k$, no instante k, suponha que no instante $k-1$ a fdp a posteriori do estado multiobjeto $f_{k-1}(\mathbf{X}_{k-1}|\mathbf{Z}_{1:k-1})$ é conhecida. Aqui $\mathbf{Z}_{1:k-1} \equiv \mathbf{Z}_1, \ldots, \mathbf{Z}_{k-1}$ é a sequência de todas as medições anteriores. Então, respectivamente, as densidades multi-objeto dos passos de predição e filtragem são expressas da seguinte forma [99]:

$$f_{k|k-1}(\mathbf{X}_k|\mathbf{Z}_{1:k-1}) = \int \Pi_{k|k-1}(\mathbf{X}_k|\mathbf{X}') f_{k|k-1}(\mathbf{X}'|\mathbf{Z}_{1:k-1}) \delta\mathbf{X}', \tag{2.29}$$

$$f_k(\mathbf{X}_k|\mathbf{Z}_{1:k}) = \frac{\varphi_k(\mathbf{Z}_k|\mathbf{X}_k) f_{k|k-1}(\mathbf{X}_k|\mathbf{Z}_{1:k-1})}{\int \varphi_k(\mathbf{Z}_k|\mathbf{X}) f_{k|k-1}(\mathbf{X}|\mathbf{Z}_{1:k-1}) \delta\mathbf{X}}, \tag{2.30}$$

onde $\mathbf{X}'$ é o conjunto de alvos observados anteriormente, $\varphi_k(\mathbf{Z}_k|\mathbf{X}_k)$ é a função de verossimilhança das observações e $\Pi_{k|k-1}(\mathbf{X}_k|\mathbf{X}')$ é a densidade de transição de estados, a qual representa o modelo de movimento dos alvos ou objetos.

2.4.2 Filtro PHD recursivo

O filtro apresentado acima sofre de problemas de tratabilidade computacional das integrais tal qual o filtro de Bayes. Portanto, tem sua importância igualmente teórica. Visando suplantar tais limitações, Mahler [66] deriva um filtro Bayesiano recursivo utilizando a função intensidade, o qual é chamado filtro PHD. Essa técnica considera vários sensores, probabilidade não constante de detecção, alarmes falsos de Poisson, bem como o aparecimento, *spawning* e desaparecimento dos objetos. O PHD é uma aproximação de melhor ajuste da posterior *multitarget* e propaga o momento estatístico de primeira ordem da posterior *multitarget*. Para um processo de ponto de Poisson, conforme descrito anteriormente, as equações de predição e de atualização do filtro PHD são [97], respectivamente,

$$D_{k|k-1}(\mathbf{x}) = b_k(\mathbf{x}) + p_S \int \pi_{k|k-1}(\mathbf{x}|\mathbf{x}')D_{k-1}(\mathbf{x}')\mathrm{d}\mathbf{x}' \tag{2.31}$$

e

$$D_k(\mathbf{x}) = D_{k|k-1}(\mathbf{x}) \times \left[1 - p_D + \sum_{\mathbf{z}\in\mathbf{Z}_k} \frac{p_D g_k(\mathbf{z}|\mathbf{x})}{c_k(\mathbf{z}) + \int p_D g_k(\mathbf{z}|\mathbf{x})D_{k|k-1}(\mathbf{x})\mathrm{d}\mathbf{x}}\right], \tag{2.32}$$

onde $\mathbf{x}$ é o estado de um único objeto (um vetor aleatório), $\mathbf{z}$ é uma medida de um único objeto (um vetor aleatório), $b_k(\mathbf{x}\ )$ é intensidade dos *births* no instante k, p_S é a probabilidade de que um alvo ainda exista no momento k, e $\pi_{k|k-1}$ é a função de transição alvo, D_{k-1} é a intensidade anterior, D_k é a PHD a posteriori, p_D é a probabilidade de que um objeto seja detectado no momento k, $c_k(\mathbf{z})$ é o PHD do *clutter* no instante k, $g_k(\mathbf{z}|\mathbf{x})$ é o modelo de observação. Por simplicidade, as probabilidades de detecção p_S e p_D apresentadas aqui são independentes do estado.

A Equação 2.32 é interessante para uma interpretação intuitiva do passo de atualização [1]:

- Sem objetos no campo de visão do sensor: nesse caso, número de alvos ou objetos teriam $p_D = 0$ e, consequentemente, $D_k = D_{k|k-1} \times (1-0+0) = D_{k|k-1}$. Ou seja, sem novas observações, a estimativa seria a própria predição;

- Presença de objetos no campo de visão do sensor: considerando que um objeto no campo de visão do sensor tem $p_D \approx 1$ teríamos

$$D_k(\mathbf{x}) \approx \sum_{\mathbf{z}\in\mathbf{Z}_k} \frac{g_k(\mathbf{z}|\mathbf{x})}{c_k(\mathbf{z}) + \int p_D g_k(\mathbf{z}|\mathbf{x})D_{k|k-1}(\mathbf{x})\mathrm{d}\mathbf{x}}$$

  Portanto, a PHD da predição seria modificada pela soma dos termos dependentes das observações e do *clutter*;

- Robustez ao *clutter*: considerando que λ é grande e uniformemente distribuído numa região $|R|$, teríamos

$$p_D \frac{g_k(\mathbf{z}|\mathbf{x})}{\frac{\lambda}{|R|} + p_D g_k(\mathbf{z}|\mathbf{x}) D_{k|k-1}(\mathbf{x})} \approx p_D \frac{|R|}{\lambda} g_k(\mathbf{z}|\mathbf{x}) \approx 0,$$

uma vez que λ domina o numerador. Portanto, nessa região R, é provável que seja um falso alarme e não contribuiria significativamente para a posteriori. Por outro lado, se uma observação é originada de uma região de baixo *clutter*, é improvável que seja um falso alarme, portanto, $c_k(\mathbf{z}) \approx 0$, portanto,

$$p_D \frac{D_{k|k-1}(\mathbf{x}) g_k(\mathbf{z}|\mathbf{x})}{c_k(\mathbf{z}) + p_D g_k(\mathbf{z}|\mathbf{x}) D_{k|k-1}(\mathbf{x})} \approx p_D \frac{D_{k|k-1}(\mathbf{x}) g_k(\mathbf{z}|\mathbf{x})}{0 + p_D g_k(\mathbf{z}|\mathbf{x}) D_{k|k-1}(\mathbf{x})} \approx 1,$$

ou seja, contribuiria na atualização da intensidade.

2.4.3 Filtro PHD baseado em misturas Gaussianas

De acordo com [101], muitos modelos estatísticos envolvem uma distribuição de mistura finita de uma forma ou de outra. Tais modelos recebem particular atenção no campo da aprendizagem não supervisionada. Por exemplo, misturas Gaussianas podem ser utilizadas para representar a existência de *clusters*. Apesar de menos difundido, esse tipo de modelo também é utilizado na estimação Bayesiana, como no caso do (*Mixture Kalman Filter*) proposto em [102].

Uma mistura Gaussiana, para um filtro multiobjetos GM PHD, pode ser definida como uma soma ponderada [103]:

$$GM(x) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(x; m_k^{(i)}, P_k^{(i)}), \tag{2.33}$$

onde $w_k^{(i)}$, $m_k^{(i)}$ e $P_k^{(i)}$ são, respectivamente, o peso, a média e a matriz de covariância do i-ésimo componente da mistura composta por um total de J_k Gaussianas. No filtro GM PHD, todos os seus componentes (*births*, *spawning* etc.) são modelados como uma GM.

O filtro PHD possui duas aproximações clássicas. A primeira delas utiliza partículas, chamada *Sequential Monte Carlo* (SMC) [73, 104]. Já a abordagem GM, objeto deste estudo, é tida como menos custosa computacionalmente [2].

Aqui, será apresentada uma versão linear do filtro, podendo ser adaptada para o

caso não linear (ver [2]), na qual cada alvo ou objeto possui o modelo dinâmico [29]

$$f_{k|k-1}(\mathbf{x}_k|_{k-1}) = \mathcal{N}(\boldsymbol{x}; F_{k-1}\mathbf{x}_{k-1}, Q_{k-1}) \text{ e} \tag{2.34}$$

$$g_k(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{z}; H_k\mathbf{x}_{k-1}, R_k), \tag{2.35}$$

onde $\mathcal{N}(\cdot; \mathbf{m}, Q_{k-1})$ denota uma fdp Gaussiana de média m e matriz de covariância Q, F_{k-1} é matriz de transição de estado, Q_{k-1} é covariância do ruído do processo, H_k é matriz de observação e R_k é covariância do ruído da observação.

No caso mais geral, as probabilidades de sobrevivência $p_{S,k}(\mathbf{x})$ e de detecção dos objetos $p_{D,k}(\mathbf{x})$ são dependentes do estado, contudo, são consideradas constantes na maioria das aplicações encontradas na literatura. Os RFSs que modelam os nascimentos espontâneos dos alvos existentes e dos alvos gerados a partir dos já existentes são aproximados pelas misturas [2]

$$\gamma_k(\mathbf{x}) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(\mathbf{x}; m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}) \tag{2.36}$$

e

$$\beta_{k|k-1}(\mathbf{x}) = \sum_{j=1}^{J_{\beta,k}} w_{\beta,k}^{(j)} \mathcal{N}(\mathbf{x}; F_{\beta,k-1} m_{\beta,k}^{(j)} + d_{\beta,k}^{(j)}, P_{\beta,k}^{(j)}). \tag{2.37}$$

O termo $F_{\beta,k-1} m_{\beta,k}^{(j)} + d_{\beta,k}^{(j)}$ leva em consideração que a posição provável dos objetos gerados é a partir da posição a priori dos objetos já existentes acrescida de um fator d. Além disso, a distribuição para o *clutter* $c(\mathbf{z})$ é a de Poisson é dada por

$$c(\mathbf{z}) = \lambda_k v_k(\mathbf{z}), \tag{2.38}$$

onde λ_k é o parâmetro de Poisson que especifica o número esperado de falsos alarmes e v_k é a distribuição de probabilidade sobre o espaço de medição.

Predição

A cada instante k, a predição de função intensidade é realizada por meio da soma das misturas gaussianas a priori (instante $k-1$) com as misturas objetos sobreviventes e dos objetos gerados

$$D_{k|k-1}(\mathbf{x}) = D_{S,k|k-1}(\mathbf{x}) + \gamma_k(\mathbf{x}) + \beta_{k|k-1}(\mathbf{x}), \tag{2.39}$$

onde $D_{S,k|k-1}(\mathbf{x}) = p_{S,k}(\mathbf{x}) D_{k-1}(\mathbf{x})$. Mais especificamente, com $p_{S,k}$ constante, temos

$$D_{S,k|k-1}(\mathbf{x}) = p_{S,k} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}(\mathbf{x}; m_{S,k|k-1k}^{(j)}, P_{S,k|k-1}^{(j)}), \tag{2.40}$$

com

$$m_{S,k|k-1}^{(j)} = F_{k-1} m_{k-1}^{(j)} \text{ e} \tag{2.41}$$

$$P_{S,k|k-1}^{(j)} = Q_{k-1} + F_{k-1} P_{k-1}^{(j)} F_{k-1}^{T}. \tag{2.42}$$

Em relação aos objetos gerados, temos

$$w_{k-1}^{(j)} w_{\beta,k}^{(l)} \mathcal{N}(\mathbf{x}; m_{\beta,k|k-1k}^{(j,l)}, P_{\beta,k|k-1}^{(j,l)}), \tag{2.43}$$

com

$$m_{\beta,k|k-1}^{(j)} = F_{\beta,k-1}^{(l)} m_{k-1}^{(j)} + d_{\beta,k-1}^{(l)} \text{ e} \tag{2.44}$$

$$P_{\beta,k|k-1}^{(j)} = Q_{\beta,k-1}^{(l)} + F_{\beta,k-1}^{(l)} P_{\beta,k-1}^{(j)} (F_{\beta,k-1}^{(l)})^{T}. \tag{2.45}$$

Exemplo 2.4.1 *Exemplo do passo de predição de uma PHD unidimensional.*

Suponha que em certo instante $D_{k-1|k-1}(x) = 0.04\mathcal{N}(x; -2, 0.01) + 0.04\mathcal{N}(x; 2, 0.01)$, $p_s = 0.9$, $F_{k-1} = 1$ e $Q_{k-1} = 0.09$. Além disso, não há alvo gerado a partir da PHD a priori. Portanto, teríamos

$$D_{S,k|k-1}(\mathbf{x}) = 0.9 \times 0.04 \times (\mathcal{N}(x; -2, 0.09 + 0.01) + \mathcal{N}(x; 2, 0.09 + 0.01)).$$

Nesse mesmo instante, considerando dois nascimentos tal que

$$\gamma_k(\mathbf{x}) = 0.01 \times (\mathcal{N}(x; -1, 0.01) + \mathcal{N}(x; 1, 0.01)),$$

teríamos a predição de acordo com

$$\begin{aligned} D_{k|k-1}(\mathbf{x}) =& 0.9 \times 0.04 \times (\mathcal{N}(x; -2, 0.09 + 0.01) + \mathcal{N}(x; 2, 0.09 + 0.01)) \\ &+ 0.05 \times (\mathcal{N}(x; -1, 0.2) + \mathcal{N}(x; 1, 0.2)). \end{aligned}$$

Esse resultado é apresentado na Figura 2.6. Observa-se uma diferença entre a intensidade a priori $D_{k-1|k-1}(x)$ e a $D_{S,k|k-1}(x)$. Como p_s difere da unidade, ocorre uma diminuição do máximo das gaussianas, bem como o achatamento da curva, uma vez que a incerteza é aumentada durante a predição.

Atualização

Na etapa de atualização, o filtro estima os valores da função intensidade no instante k, com base nas observações obtidas nesse mesmo instante. Como modelo

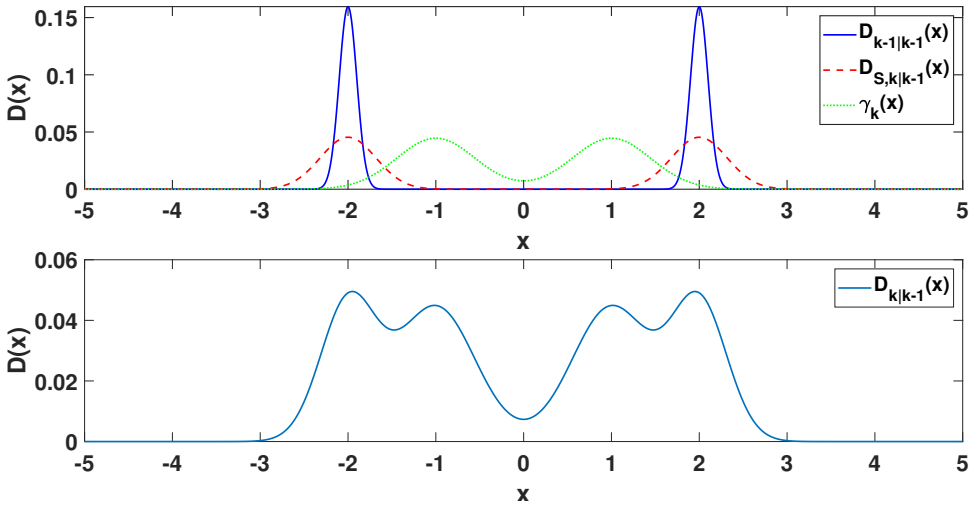

Figura 2.6: Exemplo do passo de predição do filtro GM PHD.

para as observações, utiliza-se

$$g_k(\mathbf{z}_k|\{x_k\}) = \begin{cases} p_{D,k}\mathcal{N}(z_k; H_k x_k, R_k), & \text{se } \mathbf{z}_k = \{z_k\}; \\ 1 - p_{D,k}, & \text{se } \mathbf{z}_k = \varnothing; \\ 0, & \text{se } |\mathbf{z}_k| > 1; \end{cases} \tag{2.46}$$

com $p_{D,k}$ e g_k linear.

A função intensidade a posteriori $D_k(\mathbf{x})$ é a composição entre os objetos não detectados e os detectados [2, 29]

$$D_k(\mathbf{x}) = (1 - p_{D,k})D_{k|k-1} + \sum_{z \in \mathcal{Z}} D_{D,k}(\mathbf{x}, \mathbf{z}), \tag{2.47}$$

onde

$$D_{D,k}(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{J_{k|k-1}} w_k(j)\mathcal{N}(\mathbf{x}; m_{k|k-1k}^{(j)}, P_{k|k-1}^{(j)}), \tag{2.48}$$

$$w_k(j) = \frac{p_{D,k} w_{k|k-1}^{(j)} q_k^{(j)}(\mathbf{z})}{c_k(\mathbf{z}) + p_{D,k} \sum_{l=1}^{j_{k|k-1}} w_{k|k-1}^{(l)} q_k^{(l)}(\mathbf{z})}, \tag{2.49}$$

$$q_k^{(j)}(z) = \mathcal{N}(z; H_k m_{k|k-1}^{(j)}, R_k + H_k P_{k|k-1}^{(j)} H_k^T), \tag{2.50}$$

$$m_{k|k}^{(j)} = m_{k|k-1}^{(j)} + K_k^{(j)}(z - H_k m_{k|k-1}^{(j)}), \tag{2.51}$$

$$P_{k|k}^{(j)} = [I - K_k^{(j)} H_k] P_{k|k-1}^{(j)}, \tag{2.52}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)} H_k^T (H_k P_{k|k-1}^{(j)} H_k^T + R_k)^{-1}. \tag{2.53}$$

Notar que I refere-se à matriz Identidade e K_k ao ganho de Kalman. Adicionalmente,

é importante destacar que os componentes $q_k^{(j)}$ são atualizados conforme um Filtro de Kalman convencional. Portanto, para modelos não lineares, pode-se substituí-los por suas derivações *Extended* ou *Unscented*.

Outro ponto de destaque para esse filtro é que, inicialmente, considera todas as detecções no momento $k-1$ como objetos no momento k. Isso permite que objetos de baixa probabilidade de detecção sejam estimados de forma confiável [1].

Exemplo 2.4.2 *Exemplo do passo de atualização de uma PHD unidimensional - continuação Exemplo 2.4.1.*

Suponha que $z = \{-2, -1, 1\}$, $D_{k|k-1}$ é a função intensidade apresentada na Figura 2.6 e $p_D = 0.7$. Adicionalmente, $H_k = 1$, $R_k = 0.2^2$ e

$$c_k(z) = \begin{cases} 0,3 & \text{se } |x| \leq 5; \\ 0, & \text{caso contrário.} \end{cases}$$

O resultado do passo de atualização é apresentado na Figura 2.7. Ressalte-se que a PHD a posteriori possui 16 Gaussianas, contudo, apenas os elementos principais são visíveis.

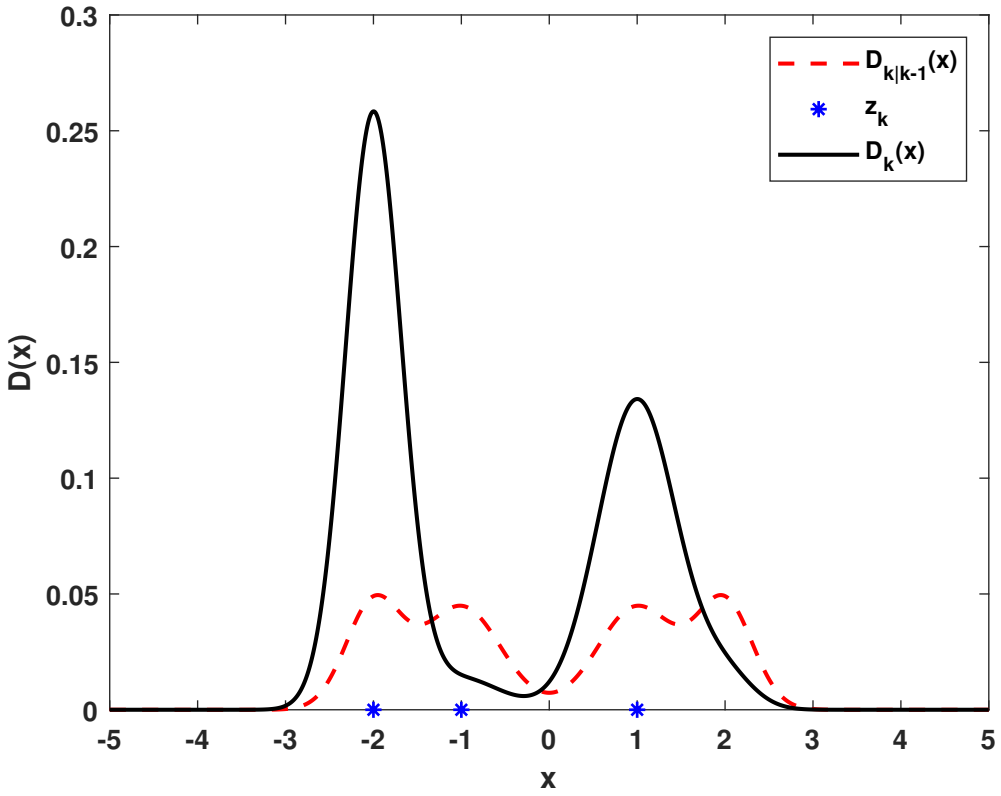

Figura 2.7: Exemplo do passo de predição do filtro GM PHD.

Prune, *Merge* e extração

Para implementações práticas, este algoritmo necessita de etapas complementares quando comparado à forma canônica da sequência Bayesiana. O filtro GM PHD sofre de problemas computacionais associados ao aumento sem limites do número de componentes gaussianos com o passar do tempo, de acordo com [2]. Os métodos de *Prune* & *Merge* visam superar essa limitação prática do filtro.

O *Prune* consiste em eliminar componentes da mistura gaussiana com valores abaixo de um limite T. Combinado com o primeiro, o método de *Merge* é aplicado para evitar o crescimento ilimitado. Este método consiste em agrupar componentes da GM em uma região definida pela distância de Mahalanobis[1] L e delimitada pelo limiar U de acordo com:

$$L := \left\{ i \in I = w_k^{(i)} > T | (m_k^{(i)} - m_k^{(j)})^T (P_k^{(i)})^{-1} (m_k^{(i)} - m_k^{(j)}) \leq U \right\}, \tag{2.54}$$

onde j se refere ao componente com maior peso no conjunto I. Essa distância mede o quanto alguns pontos estão distantes da média, refletindo a dispersão da amostra considerando a matriz de covariância. Uma interpretação intuitiva é que quanto mais os dados estiverem correlacionados, menor será a distância entre eles, formando um *cluster*.

A última etapa de uma aplicação prática do filtro GM PHD é a extração de estado, a qual elimina componentes GM com pesos fracos após a etapa de *Prune* & *Merge*. Uma alternativa melhor é selecionar as médias das Gaussianas que possuem pesos maiores que um *Limite de extração* E [2]. É importante observar que esta etapa não interfere no desempenho do filtro.

A estratégia proposta por Vo e Ma [2] é apresentado no Algoritmo 1.

Algorithm 1 StateExtratction$\left(\left\{ w_k^{(i)}, m_k^{(i)}, P_k^{(i)} \right\}_{i=1}^{J_k} \right)$

Set $\widehat{\mathbf{X}}_k := \varnothing$
for $i \leftarrow 1$ **to** J_k
  do { **if** $w_k^{(i)} > E$
    then for $i \leftarrow 1$ **to** $round(w_k)$
    do $\widehat{\mathbf{X}}_k := [\widehat{\mathbf{X}}_k, m_k^{(i)}]$
return $(\widehat{\mathbf{X}}_k)$

Métricas de desempenho do filtro

O conceito de erro entre uma grandeza de referência e seu valor estimado desempenha um papel fundamental em qualquer problema de filtragem [106]. Ao contrário

[1] Ver [105].

da ideia de distância perdida em sistemas de rastreamento de objeto único, como o erro entre o estado real e estimado, não há uma maneira direta de medir esse erro no caso de vários objetos. Como afirmado por Schuhmacher et al. [106], uma distância de erro multiobjeto satisfatória precisa capturar a "diferença" entre dois conjuntos de vetores, ou seja, o estado multiobjeto de referência e o multiobjeto estimado estado do objeto, de uma maneira matematicamente consistente, mas fisicamente significativa.

A métrica *Optimal Sub-Pattern Assignment* (OSPA) é a "distância" entre um conjunto de objetos rastreados e as posições verdadeiras conhecidas. Essa métrica contém duas medidas de erro entre esses conjuntos: componente de erro de localização (que considera a estimativa de estado) e componente de cardinalidade (uma referência para o número de alvos perdidos).

Para dois conjuntos finitos X e Y com as respectivas cardinalidades m e n, para $m \leq n$, a métrica OSPA é definida de acordo com [106]:

$$\overline{d}_p^{(c)}(X,Y) := \left(\frac{1}{n} \left(\min_{\pi \in \Pi} \sum_{i=1}^{m} (d^{(c)}(x_i, y_{\pi(i)})^p + c^p(n-m) \right) \right)^{\frac{1}{P}}, \tag{2.55}$$

onde $d^{(c)}(x,y) := min(c, d(x,y))$ é a distância de corte entre dois elementos de X e Y com $c > 0$ sendo o parâmetro de corte, Π_n representa o conjunto de permutações de comprimento m com elementos retirados de $\{1, 2, \ldots, n\}$ com $n \in \mathbb{N}$, qualquer elemento $\pi \in \Pi_n$ é uma sequência $\pi(1), ..., \pi(n)$ e $1 \leq p < \infty$.

O parâmetro de corte c determina o peso relativo do componente de erro de cardinalidade em relação ao componente de erro de distância base. Valores maiores de c tendem a enfatizar erros de cardinalidade. Já o parâmetro p, controla a penalidade atribuída às estimativas "outlier" (que não estão próximas das posições *truth*). Um valor mais alto de p aumenta a sensibilidade a valores *outliers* [107].

2.4.4 Considerações finais

Além da motivação para uso dos RFS, este capítulo abordou os principais conceitos necessários para aplicação dessa técnica no problema de *tracking* ou de mapeamento em robótica, desde a definição até aplicação de filtro PHD.

Capítulo 3

Filtro PHD no problema de mapeamento em robótica

A solução do problema de localização é um dos pontos-chave para o desenvolvimento de robôs. A partir da obtenção dessa informação, é possível fornecer os dados para o sistema de navegação do veículo. Com isso, uma trajetória é definida e seguida pela camada (ou módulo) de controle da trajetória.

Contudo, antes de se localizar, é preciso uma referência ou, de maneira abrangente, um mapa. Em muitos sistemas reais, os mapas não são completamente conhecidos ou são, até mesmo, desconhecidos.

Portanto, em robótica, temos dois problemas, localização e mapeamento, que quando combinados e solucionados de maneira conjunta são chamados SLAM (*Simultaneous Localization and Mapping*). Isto posto, esse capítulo objetiva apresentar os principais conceitos relacionados ao tema do ponto de vista *bayesiano*.

O objetivo deste capítulo é apresentar os principais conceitos de localização e mapeamento e, a partir disso, apresentar aplicação de abordagem baseada em RFS utilizada para o mapeamento em robótica.

3.1 A taxonomia dos problemas de localização e mapeamento

Segundo Thrun et al. [96], nem todo problema de localização é igualmente difícil. Para se entender a dificuldade envolvida nesse tipo de problema, é realizada uma classificação baseada nos casos mais recorrentes. Uma vez que no SLAM trata, em muitos casos, o mapeamento e a localização como problemas independentes, essa mesma divisão pode ser adotada.

A seguir, apresenta-se tal categorização com base no proposto por Thrun et al. [96] e Stachniss et al. [18].

3.1.1 SLAM completo x em tempo real

O problema de SLAM é dito completo (do inglês, *full*) é aquele no qual se quer determinar toda a trajetória realizada por um robô desde o instante inicial 0 [108]. Matematicamente, temos a distribuição de probabilidade

$$p(X_{0:k}|Z_{0:k}, U_{0:k}), \tag{3.1}$$

onde a $X_{1:k} = [x_0 \ ... \ x_k]$, $Z_{1:k} = [z_0 \ ... \ z_k]$, e $U_{1:k} = [u_0 \ ... \ u_k]$. Nesse caso, toda a informação disponível é utilizada.

Já o chamado SLAM em tempo real (do inglês, *online*), apenas a informação instantânea é de interesse e, para isso, utiliza-se a distribuição

$$p(X_k|Z_k, U_k). \tag{3.2}$$

Os algoritmos que tratam do problema *online* são comumente chamados filtros [18].

3.1.2 Localização global x local

Esse tipo de problema pode ser caracterizado pelo conhecimento acerca da localização no instante inicial. São três casos, num nível crescente de dificuldade, de acordo com [96]:

1. **Rastreamento de posição** (*position tracking*) – a posição inicial do robô é conhecida. A incerteza da posição é aproximada por uma distribuição unimodal. Trata-se de um problema local (incerteza confinada a uma região onde a *pose* é verdadeira).

2. **Localização global** (*global localization*) – a posição inicial não é conhecida. A aproximação por distribuições unimodais é inadequada.

3. **Problema do robô sequestrado** – durante a operação, o robô pode ser retirado da posição ao longo de seu percurso e transferido para outra posição aleatória. É uma variação do problema de localização global, porém ainda mais complexo, pois o robô pode acreditar estar em um ponto no qual ele não está. O uso prático desse tipo de problema é na verificação da habilidade do robô se recuperar de falhas em uma localização global.

3.1.3 Ambientes estáticos x dinâmicos

Com relação ao ambiente no qual o robô está inserido, pode-se dizer ser estático quando o único agente em movimento é o próprio robô. Já no dinâmico, outros

objetos podem movimentar-se ou mudar de posição com o tempo. O caso dinâmico pode ser mais realista, porém, mais complexo do que o primeiro.

3.1.4 Representação topológica x métrica

Algumas representações do ambiente são fundamentadas numa descrição qualitativa do ambiente, o qual é descrito por meio de relações entre os marcos (locais de interesse, importância). Esse mapas caíram em desuso em robótica e os métricos (baseados em coordenadas, distâncias) são os mais utilizados em SLAM, de acordo com Stachniss et al. [18].

3.1.5 Localização passiva x ativa

Na localização passiva, o sistema de localização do robô apenas observa o ambiente e não há atuação sobre o movimento do robô motivada por essa tarefa.

Já na localização ativa, o sistema de localização do robô pode, de acordo com a necessidade, utilizar-se do controle de movimento para auxiliar nessa tarefa. Tipicamente, possuem resultados melhores do que no caso passivo. Um exemplo de aplicação é na navegação costeira que tem como limitação depender de uma ação de controle sobre o robô. Outro exemplo típico é o processo de triangulação de uma AUV que utiliza sonares passivos, ver [109].

3.1.6 Multi-robôs x robô único

Quanto à quantidade de robôs empregados na solução, há a possibilidade de haver um sistema de localização baseado em apenas um robô, o qual com meios próprios determina sua localização no mapa.

Contudo, existem sistemas que empregam times de robôs, onde cada robô pode compartilhar dados sobre sua localização com os demais por um sistema de comunicação.

3.1.7 Localização baseada em marcos x casamento de modelos

Na localização baseada em marcos (*landmarks*), objetos do ambiente são utilizados como referencial para prover localização.

Já na localização por casamento de modelos (*model matching*), características geométricas são extraídas de imagens do ambiente e comparadas com um modelo do ambiente, possibilitando assim a verificação de erros de odometria [110], para exemplos, ver [111, 112].

3.2 Mapeamento "convencional"

Segundo Stachniss et al. [18], os trabalhos nessa área foram inicialmente concentrados em robôs que operavam em ambiente fechado (*indoor*). Daí a utilização inicial de mapas bidimensionais, dada a facilidade de representação dos obstáculos ou objetos presentes. Hoje há uma enorme gama de aplicações e diferentes formas de representar o mundo no qual o sistema autônomo está inserido, cada uma delas adequada à situação enfrentada. Diferentes formas para modelar o ambiente utilizadas nos problemas de mapeamento e SLAM pode ser consultadas em [18], [113] e [114].

Do ponto de vista *bayesiano*, o mapeamento é definido matematicamente como a distribuição

$$p(M|X_k, Z_k), \tag{3.3}$$

onde M é o mapa do ambiente, X_k é posição do veículo e Z_k a observação.

Essa distribuição será definida de acordo com o tipo de abordagem adotada. Historicamente, as principais formas de representação são: *occupancy gridds* e *feature-based*.

3.2.1 Mapas baseados em *occupancy grids*

Segundo Burgard et al. [115], os mapas dessa categoria, introduzidos nos anos 1980 por Moravec e Elfes[116], são uma abordagem popular e probabilística para representar o meio ambiente. A técnica consiste em dividir o ambiente numa grade e calcular a probabilidade de cada célula estar ocupada ou não.

Possuem a vantagem de não depender de algum recurso predefinido. Além disso, oferecem acesso em tempo constante às células e fornecem a capacidade de representar áreas desconhecidas (não observadas), que podem ser importantes, por exemplo, de acordo com Burgard et al. [115], em tarefas de exploração. Como principal limitação, podem apresentar erros significativos de arredondamento devido à discretização.

Para o caso $2D$, assume-se que o mapa é uma grade bidimensional de L células, denotas por $m_1, ..., m_L$. Dadas a sequência de observações e a das posições do robô, o objetivo é calcular a distribuição de probabilidade *a posteriori* $p(M|X_k, Z_k)$ considerando a hipótese de independência

$$p(M|x_{1:k}, z_{1:k}) = \prod_{l=1}^{L} p(M|x_{1:k}, z_{1:k}), \tag{3.4}$$

como forma de tornar o problema tratável computacionalmente.

Note que esta suposição é bastante forte. Basicamente, afirma que qualquer informação sobre a ocupação de uma célula não nos diz algo sobre as células vizinhas

[115]. Com isso, o problema resume-se a se determinar a ocupação de cada célula individualmente. Detalhes acerca do algoritmo podem ser encontrados em [117].

3.2.2 Mapas baseados em *Features*

Para ambientes com objetos (marcos ou alvos, também chamados na literatura inglesa de *landmarks* ou simplesmente *features*) localmente distinguíveis, os mapas *feature-based* têm sido amplamente utilizados. Assumi-se que a posição do robô é sempre conhecida e que resta, simplesmente, manter uma estimativa sobre as posições dos marcos individuais ao longo do tempo [115].

Como já explicitado neste trabalho, um mapa M, definido sob essa abordagem, consiste em uma coleção de *features*. O caso mais simples a ser considerado é quando o objeto é caracterizado apenas por suas coordenadas cartesianas (representado como um ponto), não se considerando nenhuma característica adicional (e.g. cor, geometria etc.).

Essa facilidade de representação, associada à condição da posição do robô ser considerada determinística, faz com que as distribuições gaussianas sejam as mais utilizadas, considerando-se o modelo de observação linear. Tal hipótese diminui a complexidade do algoritmo, sendo tomada como base para o FastSLAM [118], segundo Burgard et al. [115].

Matematicamente, temos

$$p(M|x_k, z_k) = \frac{p(z_k|x_k, M)p(M, X_k)}{p(x_k, z_k)}, \tag{3.5}$$

considerando $p(x_k) = 1$,

$$p(M|x_k, z_k) = \eta p(z_k|M)p(M), \tag{3.6}$$

sendo η um fator de normalização.

O modelo de predição é simplesmente

$$x_{map}(k|k-1) = x_{map}(k-1|k-1). \tag{3.7}$$

Inicialmente, o mapa está vazio, por isso, é necessário utilizar algum método para a adição de *features* recém-descobertos a ele. Para este fim, introduz-se uma função de inicialização de características Y que toma como argumentos o vetor de estado antigo e uma observação para um marco e retorna um novo vetor de estado mais longo com o novo recurso em seu final, considerando a utilização de abordagem vetorial.

Como exemplo, para o caso de medições de direção e distância, ter-se-ia a adição de nova característica dada por

$$\boldsymbol{x}_{new} = \begin{bmatrix} x_v + r\cos(\theta + \theta_v) \\ y_v + r\sin(\theta + \theta_v) \end{bmatrix}, \tag{3.8}$$

onde $[x_v\ y_v]^T$ corresponde à coordenada do veículo, θ_v à orientação do veículo, θ é o ângulo do marco em relação ao robô.

Com essa estratégia, também é preciso uma forma de alterar a matriz de covariância da distribuição $p(M|x_k, z_k)$. Possíveis estratégias e seus maiores detalhes podem ser consultados em [96], como no caso da adotada pelo filtro EKF.

3.2.3 Localização e mapeamento simultâneos

O SLAM, do ponto de vista *bayesiano*, consiste em determinar a distribuição *a posteriori* da pose do robô e do mapa dadas as medições e a ação de controle. Matematicamente,

$$p(x_k, M|z_k, u_k). \tag{3.9}$$

Esse problema é do tipo *chicken and egg*, pois um mapa é necessário para se determinar a localização e para a estimativa da *pose* é necessário um mapa.

O SLAM é fundamental para uma variedade de aplicações, como robôs em ambientes *indoor*, *outdoor* ou confinados, veículos aéreos e subaquáticos tripulados e autônomos.

Torna-se de difícil solução, uma vez que é preciso estimar tanto a posição do robô quanto o mapa, cuja relação entre as observações não é conhecida. Daí surge outra dificuldade extra, a associação de dados, que, segundo Stachniss et al. [18], na prática, é um dos problemas mais difíceis do mapeamento e localização simultâneos.

Ainda segundo Stachniss et al. [18], existem três paradigmas principais, dos quais as demais soluções derivam, os filtros EKF, de partículas e as baseadas em representações gráficas (grafos).

Um algoritmo que merece ser destacado é o FastSLAM (*Fast Simultaneous Localization and Mapping*). De acordo com Montemerlo e Thrun [119], o EKF tem duas sérias deficiências que o impedem de ser aplicado a ambientes grandes e reais: complexidade quadrática e sensibilidade a falhas na associação de dados. Ainda em consonância com [119], trata-se de uma abordagem alternativa baseada no Filtro de Partículas Rao-Blackwellized [120], que escala logaritmicamente o número de pontos de *landmarks* no mapa.

De acordo com [18], o FastSLAM utiliza partículas Rao-Blackwellized na etapa de predição e mantém na estimação (filtragem) o EKF. Maiores detalhes sobre o algoritmo, e suas diferentes versões, podem ser consultados em [121].

3.3 RFS aplicado ao mapeamento em robótica

Como mencionado anteriormente, em contraste com a representação vetorial, o RFS dos estados do mapa $\mathcal{M}_k$ pode encapsular de forma unificada as incertezas de cardinalidade e localização. Dessa forma, superando a associação de dados, a qual é um dos principais desafios no mapeamento, como já afirmado anteriormente.

Nessa abordagem, o mapa pode ser definido como o RFS $\mathcal{M}$ tal que

$$\mathcal{M}_k = \{\mathbf{m}_{k,1}, ..., \mathbf{m}_{k,n_k}\} \in \mathcal{F}(\mathcal{M}), \tag{3.10}$$

onde $\mathcal{F}(\mathcal{M})$ é o conjunto com todos os subconjuntos de $\mathcal{M} \subseteq \mathcal{R}^{n_m}$ e $m \in \mathbb{R}^f$, com f sendo o número de *features* utilizado.

O RFS para as observações é definido como [1]

$$\mathcal{Z}_k = \bigcup_{m \in \mathcal{M}_k} \mathcal{D}_k(m, X_k) \cup \mathcal{C}_k(X_k), \tag{3.11}$$

onde $\mathcal{D}_k$ é obtido a partir das medidas geradas pelas características (*features*) e $\mathcal{C}_k$ pelas medidas espúrias (*clutter*), em um instante k, que podem depender da posição $\mathcal{X}_k$ do robô.

De forma geral, $\mathcal{Z}_k$ pode representar uma série de parâmetros medidos, mas, no caso mais simples aqui descrito, é uma medida de distância e ângulo em relação ao sensor do robô. É importante ressaltar que a cardinalidade de $\mathcal{Z}_k$ geralmente difere da de $\mathcal{M}_k$ devido às incertezas nas medições, oclusões, medições espúrias e novos elementos entrando no campo de visão do robô. Supõe-se também que $\mathcal{D}_k$ e $\mathcal{C}_k$ são RFSs independentes [34].

O RFS de medidas gerado por uma *feature* m tem densidade de probabilidades igual a $p_D\left(m, \mathcal{X}_k\right) g_k(z|m, \mathcal{X}_k)$, onde o primeiro termo é a probabilidade de um sensor detectar uma *feature* m e o segundo é a probabilidade de uma *feature* m gerar uma medida z.

A habilidade de um sensor (ou um algoritmo de detecção de features) detectar um dado objeto pode ser altamente influenciada por sua posição relativa. Por exemplo, uma oclusão ou grande distância pode resultar em $p_D = 0$ [31].

A densidade de probabilidade de um sensor produzir uma medida $\mathcal{Z}_k$, dado um estado do robô (posição e orientação), $\mathcal{X}_k$, é dada pela convolução [1]:

$$g_k(\mathcal{Z}_k|\mathcal{X}_k, \mathcal{M}_k) = \sum_{\mathcal{W} \subseteq \mathcal{Z}_k} g_{\mathcal{D}}(\mathcal{W}|\mathcal{M}_k, \mathcal{X}_k) g_{\mathcal{C}}(\mathcal{Z}_k - \mathcal{W}), \tag{3.12}$$

onde $g_{\mathcal{C}}(\mathcal{Z}_k - \mathcal{W})$ é a densidade das medições espúrias e $g_{\mathcal{D}}$ é a densidade do RFS $\mathcal{D}_k$ das medições geradas pelas *features* em $\mathcal{M}_k$ (dado estado do veículo). O primeiro

termo incorpora as incertezas das medições e ruídos, o segundo termo modela os dados espúrios gerados pelo sensor que é, tipicamente, definido previamente. É reproduzida abaixo a forma geral da recursão de Bayes para mapeamento baseado em RFS:

$$p_{k|k-1}(\mathcal{M}_k, \mathcal{Z}_{0:k-1}, \mathcal{X}_k) = \int f_{\mathcal{M}}(\mathcal{M}_k|\mathcal{M}_{k-1}, \mathcal{X}_k) p_{k-1}(\mathcal{M}_k, \mathcal{Z}_{0:k-1}, \mathcal{X}_{0:k-1}) \delta\mathcal{M}_{k-1} \tag{3.13}$$

e

$$p_k(\mathcal{M}_k, \mathcal{Z}_{0:k}, \mathcal{X}_k) = \frac{g_k(\mathcal{Z}_k|\mathcal{X}_k, \mathcal{M}_k) p_{k|k-1}(\mathcal{M}_k, \mathcal{Z}_{0:k-1}, \mathcal{X}_k)}{\int g_k(\mathcal{Z}_k|\mathcal{X}_k, \mathcal{M}_k) p_{k|k-1}(\mathcal{M}_k, \mathcal{Z}_{0:k-1}, \mathcal{X}_k) \delta\mathcal{M}_k}, \tag{3.14}$$

sendo δ a indicação de integral de um conjunto e a integral é definida de acordo com a Equação 2.10. É importante ressaltar que essa formulação é equivalente à apresentada nas Equações 2.29 e 2.30, apenas adaptada ao caso onde o estado de interesse é um mapa.

Esse mapa baseado em *features* encapsula as incertezas inerentes no número de características introduzidas na detecção, medidas espúrias e movimentos do robô, assim como as incertezas de posição introduzidas pelo ruído de medição.

Como fica evidente pelas equações acima, a aplicação da forma geral da recursão de Bayes é matematicamente intratável, similar ao caso descrito no Capítulo 2. Uma aproximação baseada no filtro GM PHD é proposta em [34]. O filtro aqui utilizado é o definido na Tabela II dessa referência. O objetivo é propagar a intensidade υ_k do mapa no lugar de toda a densidade multidimensional

$$\upsilon_{k|k-1}(m|X_k) = \upsilon_{k-1}(m|X_{k-1}) + b_k(m|X_k), \tag{3.15}$$

onde b_k é o PHD do RFS das últimas *features* detectadas $\mathcal{B}(\mathcal{X}_k)$.

A correção do filtro PHD, que tem forma similar ao Filtro de Bayes usual, é dada por:

$$\upsilon_k(m|X_k) = \upsilon_{k|k-1}(m|X_k) \times \left[1 - p_D(m|X_k) + \sum_{z\in\mathcal{Z}_k} \frac{p_D(m|X_k) g_k(z|m, X_k)}{c_k(z|X_k) + \int_{\mathcal{M}_k} p_D(m|X_k)(\xi|X_k) g_k(z|\xi, X_k) \upsilon_{k|k-1}(\xi|X_k) d\xi}\right], \tag{3.16}$$

onde $\upsilon_k(m|X_k)$ é a predição para a função de intensidade, $p_D(m|X_k)$ a probabilidade de uma *feature* m ser detectada com o robô na posição X_k, $g_k(z|m, X_k)$ o modelo

de medição do sensor no instante k e $c_k(z|X_k)$ a intensidade de medições espúrias no instante k.

3.4 Simulações e resultados

Neste estudo, foi considerado apenas o problema de mapeamento, o que não invalidaria uma análise, pois mesmo no RFS SLAM, para um melhor desempenho computacional, há uma separação do problema. A partir da hipótese de independência entre os problemas, a localização é, normalmente, realizada por algum algoritmo baseado em Kalman ou filtro de partículas, conforme [99].

3.4.1 Cenário

Para as simulações, foram utilizados os dados do ambiente e dos sensores disponibilizados em [122]. Tal *framework* foi adaptado para receber o algoritmo implementado, o qual foi baseado nas referências [2], [29] e [17], bem como nos códigos disponibilizados em [123, 124] e [125].

O mapa *true* do ambiente, com escala em metros, pode ser visualizado, para o instante $k = 1$, na Figura 3.1.

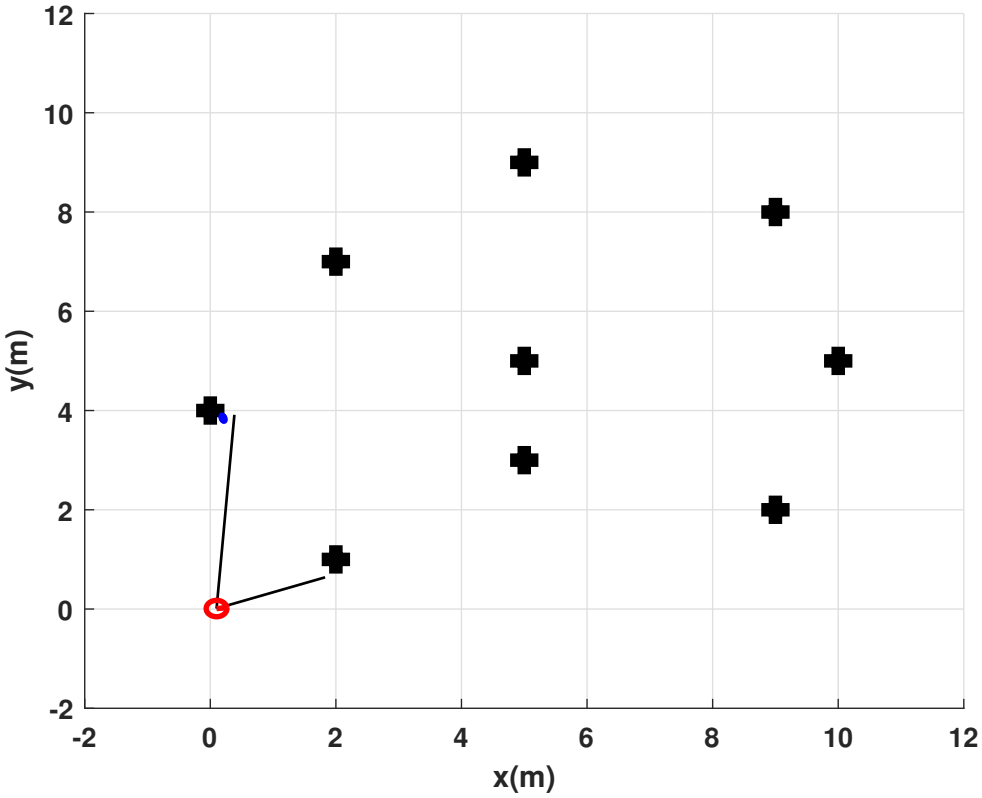

Figura 3.1: Cenário simulado em $k = 1$.

As *landmarks* estáticas, num total de 9, estão marcadas em preto e o robô em vermelho, as observações estão marcadas por pontos em azul e as linhas pretas entre o robô e as detecções são apenas uma representação para indicar cada observação dentro do FoV dos sensores. As coordenadas do robô e a posicionação de cada objeto é definida na Tabela 3.1

Tabela 3.1: Estado do robô e objetos (*landmarks*) em $k = 0$.

Entidade	x_0 **[m]**	y_0 **[m]**	**Orientação inicial** θ_0 **[Deg]**
Robô	0	0	0
Objeto 1	0	4	Não aplicável
Objeto 2	2	7	Não aplicável
Objeto 3	9	2	Não aplicável
Objeto 4	10	5	Não aplicável
Objeto 5	9	8	Não aplicável
Objeto 6	5	5	Não aplicável
Objeto 7	5	3	Não aplicável
Objeto 8	5	9	Não aplicável
Objeto 9	2	1	Não aplicável

3.4.2 Modelos utilizados

Neste cenário, o robô possui a seguinte equação dinâmica (modelo odométrico) [96]:

$$x_k = x_{k-1} + \delta_{trans} \cos(\theta_{k-1} + \delta_{rot1}), \tag{3.17}$$

$$y_k = y_{k-1} + \delta_{trans} \sin(\theta_{k-1} + \delta_{rot1}), \tag{3.18}$$

$$\theta_k = \theta_{k-1} + \delta_{rot1} + \delta_{rot2}, \tag{3.19}$$

onde $[x_k\, y_k\, \theta_k]^T$ corresponde à *pose* do robô no instante k, no intervalo $(k-1, k]$ o movimento é aproximado por uma rotação δ_{rot1}, seguido de uma translação δ_{trans} e uma segunda rotação δ_{rot2}.

Já o modelo de observação dos objetos é dada por [96]:

$$\mathbf{z}_k = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \theta_{k,robot} \end{pmatrix}, \tag{3.20}$$

onde $q = \delta^T \delta$,

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \overline{x}_{k,object} - \overline{x}_{k,robot} \\ \overline{y}_{k,object} - \overline{y}_{k,robot} \end{pmatrix} \tag{3.21}$$

e atan2 é a tangente inversa de quatro quadrantes (várias linguagens de programação fornecem uma implementação dessa função). Essa extensão da função arco tangente retorna valores entre $[-\pi, \pi]$ ao invés de valores entre $[-\pi/2, \pi/2]$.

3.4.3 Parâmetros do filtro

Nas simulações, foi implementado o filtro GM PHD com os parâmetros apresentados na Tabela 3.2, os quais foram obtidos por tentativa e erro.

Tabela 3.2: Parâmetros utilizados pelo filtro GM PHD.

Parâmetro	**Valor**
Alcance máximo do sensor	2.5 km
σ^2 para a distância	0.16 m
σ^2 para ângulo	$\pi/180°$
Probabilidade de sobrevivência (p_S)	1
Probabilidade de detecção (p_D)	0.98
Clutter	0.02×10^2
Prune threshold (T)	10^{-5}
Merge threshold (U)	4
Extraction threshold (E)	0.5

Para a estratégia de nascimento dos objetos, utiliza-se uma mistura gaussiana em que cada nova *landamrk* i, detectada no instante k, possui peso $w_{k,i} = 0.01$,

$$\overline{\mathbf{x}}_{k,i} = h^{-1}(\mathbf{z}_{k,i}, \mathbf{x}_{k,robot}),$$

e

$$P_{k,i} = M_{k,i}\, R_{k,i}\, M_{k,i}^T,$$

onde

$$M_{k,i} = jacobiano(\overline{\mathbf{x}}_{k,i}, \mathbf{x}_{k-1,robot})$$

e

$$R_{k,i} = \begin{pmatrix} 0.16^2 & 0 \\ 0 & (\pi/180)^2 \end{pmatrix}.$$

3.4.4 Resultados

Os resultados obtidos são apresentados por meio de um exemplo, ver Figura 3.2, que ilustra os objetos estimados no instante $k = 330$, o último passo da simulação realizada. Além disso, a métrica OSPA, com $c = 100$ e $p = 1$, é apresentada na Figura 3.3.

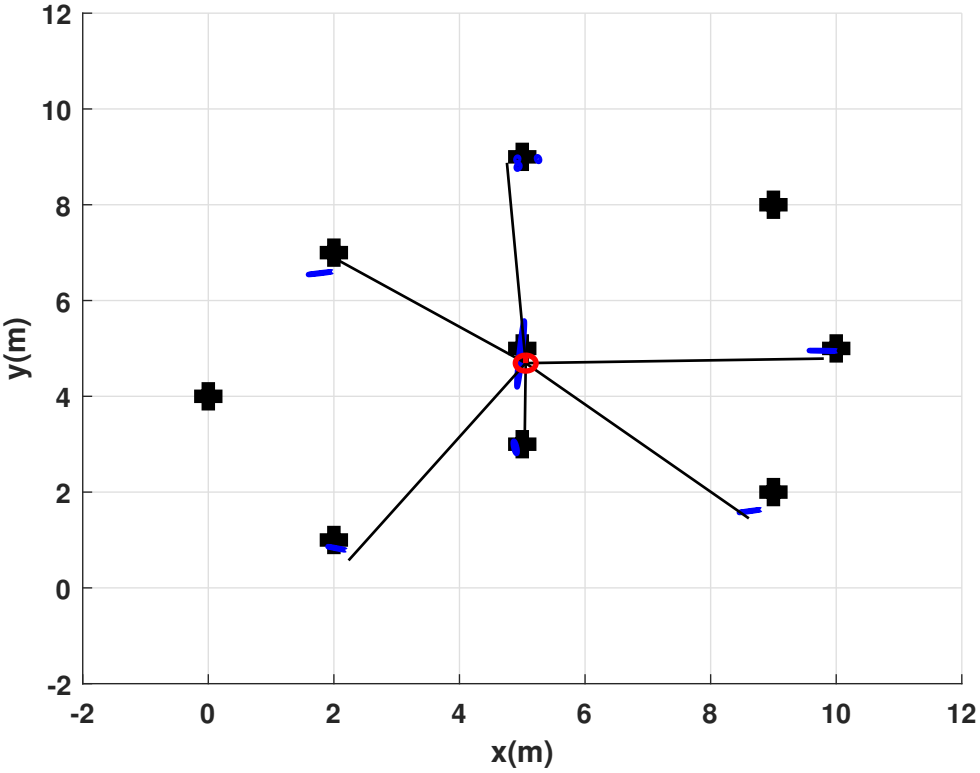

Figura 3.2: Cenário simulado em $k = 330$.

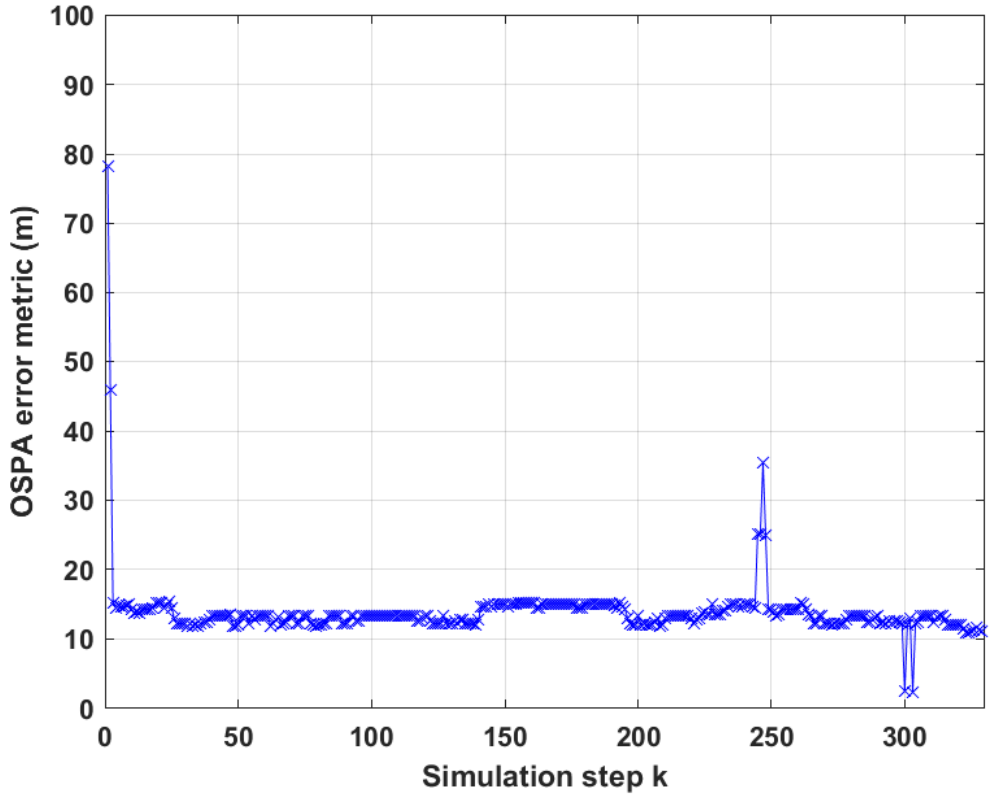

Figura 3.3: Métrica OSPA para as simulações realizadas.

A métrica OSPA indica que, após um transitório inicial, os alvos dentro do FoV dos sensores do robô são observados e estimados com um erro praticamente constante, exceto em torno de $k = 250$ e $k = 300$. Nessa primeira região, ocorre uma piora do desempenho do filtro, enquanto no segundo momento há uma sútil melhora, relembrando que quanto menor o valor da métrica, melhor é considerado o desempenho do filtro.

3.5 Considerações finais

Inicialmente, este capítulo retomou conceitos-chave no problema de SLAM. Além disso, definiram-se os diferentes tipos de mapas utilizados em robótica. A partir disso, foi proposto um filtro GM PHD para aplicação no problema de mapeamento em robótica baseado em *features*, apresentando-se os resultados por meio da métrica OSPA.

Com isso, pode-se aplicar técnicas baseadas em *Random Finte Sets* ao problema de mapeamento em robótica.

Capítulo 4

Filtro PHD em rastreamento e vigilância marítima distribuída

Este capítulo tem como objetivo apresentar a aplicação das técnicas baseadas em RFS no problema de rastreamento de múltiplos alvos (*multitarget tracking*) em uma área de vigilância marítima. Nesse contexto, é proposto um esquema de fusão dos dados obtidos localmente por cada nó da rede. Para isso, são apresentados conceitos iniciais relacionados ao problema de rastreamento de múltiplos alvos e sistemas de tracking, em particular, numa área de vigilância marítima. Em seguida, o cenário simulado é descrito e os resultados são discutidos.

4.1 Introdução aos sistemas de rastreamento

Existem diversas aplicações para os sistemas de tracking: veículos autônomos, aviação, substância em células etc. Os algoritmos utilizados nesses sistemas são desenvolvidos para responder diferentes questionamentos, tais quais:

- Quantos objetos existem?
- Onde eles estão e para onde vão?
- Possuem características ou propriedades importantes?
- Como esses objetos se comportam ao longo do tempo?

Somente para exemplificar a diversidade de aplicações dos sistemas de vigilância e desses algoritmos, poder-se-ia citar:

- sistemas de radares terrestres de vigilância em aeroportos [126, 127];
- grupo de pedestres por segurança ou qualquer outro motivo [128–131];
- rastreamento microfluídico das propriedades de células [132, 133];

- rastreamento de corpos rígidos (*extended targets*) [134, 135];
- veículos autônomos rastreando objetos (outros carros), faixas, pedestres etc [136–140].

Todos esses sistemas possuem o tema em comum: rastrear múltiplos alvos. Esse problema é chamado *multi-object tracking* (MOT) ou *multi-target tracking* (MTT), o qual é relevante para áreas como:

- robótica;
- vigilância;
- veículos autônomos;
- automação da produção;
- medicina;
- redes de sensores.

Historicamente, os algoritmos de MTT têm sua origem em aplicações de rastreamento e/ou acompanhamento de aviões com o uso de radares. Entretanto, o rastreamento de objetos em cenários urbanos, com o intuito de promover a compreensão (*situation awareness* ou consciência situacional) desse, torna-se cada vez mais importante, ainda mais com o advento e a evolução crescente das tecnologias em veículos autônomos (ou condução autônoma) [141].

Definições em *multitarget tracking*

O *Single Target Tracking* (STT) ou *Single Object Tracking* (SOT) é um problema de filtragem. SOT consiste em um cenário com um único alvo, cujo estado evolui ao longo do tempo e é apenas parcialmente observado por um sensor em intervalos discretos de tempo [142].

De acordo com [143], SOT pode ser definiodo como processamento sequencial das observações de sensores ruidosos para estimar o estado do objeto, o qual pode conter:

- posição;
- propriedades que descrevem seu movimento (velocidade, por exemplo);
- outras características de interesse.

Normalmente, o estado completo não pode ser medido diretamente.

Já o MTT ou MOT, segundo Granstrom [143], é uma extensão do SOT e pode ser definido como o processamento de várias medições obtidas de vários alvos. Portanto, o MOT pode ser definido como o processamento sequencial de medições de sensores ruidosos para determinar:

- o número de objetos dinâmicos; e
- o estado de cada objeto.

Percebe-se que, a partir, dessa definição, o MOT poderia ser encarado como diversos problemas SOT. Daí, surge outro problema, como associar às observações a cada um dos alvos? Esse é o problema da associação de dados.

Uma parte relevante da solução no MOT é resolver o problema de associação de dados, às vezes também chamado problema de correspondência. Associação de dados significa associar cada medição a uma das fontes geradoras de medição, ou seja, a um alvo ou a um falso alarme [143]. Existem as chamadas técnicas convencionais (ver [144]) para lidar com essa questão, tais como *Global Nearest Neighbor* (GNN) e *Joint Probabilistic Data Association* e *Multiple Hypothesis Tracking* (MHT). Uma abordagem mais apurada dessas técnicas fogem do escopo deste trabalho, uma vez que os filtros baseados em RFS não possuem uma associação de dados desacoplada, algo já abordado anteriormente.

Desafios em *multitarget tracking* (MTT)

Os principais desafios em MTT consistem em:

- o número de objetos desconhecidos no campo de visão do sensor (FoV, do inglês *Field of View*);
- o estado de cada objeto é desconhecido;
- os sensores são imperfeitos.

Pode-se enfatizar ainda que os objetos estão se movendo, possivelmente, de tal forma que:

- desaparecem, ou seja, deixam o FoV (*death*);
- novos objetos aparecem, ou seja, entram no FoV (*birth*);
- alvos dentro de FoV ocluem uns aos outros.

A imperfeição dos sensores leva aos seguintes problemas principais:

- detecção perdida: quando o objeto não é detectado pelo sensor;

- detecção falsa: uma observação que não é causada por um objeto. Também chamados alarmes falsos ou *clutter*.

4.1.1 Sistemas de rastreamento e vigilância no ambiente marítimo

De acordo com Nato [145], Vigilância Marítima é a observação sistemática de áreas marítimas de superfície e subsuperfície por todos os meios disponíveis para, principalmente, encontrar, identificar e determinar os movimentos de navios, submarinos e outros veículos, amigos ou inimigos, procedendo sobre ou sob a superfície dos mares e oceanos do mundo. Já o Sistema de Vigilância Marítima consiste em um sistema de coleta, reporte, correlação e apresentação de informações de apoio à decisão. Esse sistema dá apoio à vigilância marítima. Essa percepção dos elementos e eventos ambientais sobre o ambiente monitorado corresponde à consciência situacional da área de interesse.

Com base em [146], o MTT é uma tarefa difícil, porém, quando comparado a rastreamentos em vários campos ou cenários, o MTT Marítimo (*Maritime* MTT) é ainda mais desafiador. Essa complexidade relacionada ao ambiente marítimo está ligada à relação sinal-*clutter*, a qual dificulta e limita a detecção de alvos, levando a um elevado número de detecções imprecisas ou perdidas dos sensores (radares, por exemplo), bem como um considerável número de falsos alarmes.

Ainda segundo Chao e Yueji [146], construir um sistema de monitoramento marítimo, bem estruturado e poderoso, é uma medida importante para fortalecer a defesa nacional e manter a segurança marítima nacional. Nesse contexto, a importância do radar para a detecção nesse tipo de ambiente continua em ascensão, portanto, fortalecer a pesquisa em tecnologias de rastreamento para obter uma situação marítima mais precisa e eficiente tem um significado prático muito importante.

Dada a relevância desse tema e das dimensões da costa brasileira, em 2015, a Marinha do Brasil apresentou o programa estratégico denominado Sistema de Gerenciamento da Amazônia Azul (SisGAAz). Trata-se de um sistema de vigilância com o objetivo de monitorar a vigilância das Águas Jurisdicionais brasileiras e as áreas internacionais de responsabilidade das operações de Busca e Salvamento (SAR) [147].

O Sistema de Consciência Situacional por Aquisição de Informações Marítimas (SCUA) implementa o projeto-piloto do SisGAAz. Trata-se de um C4ISR (*Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance*) que integra dados de múltiplos sensores e de múltiplas fontes para avaliar a consciência situacional ou apoiar as atividades de tomada de decisão para a área

marítima de interesse. No contexto da consciência situacional, o rastreamento de múltiplos alvos é uma tarefa crucial.

A arquitetura desse sistema proposto é dita distribuída, ou seja, é composta por diversos nós, os quais compõem uma rede de sensores. Esses dispositivos podem ser fixos (estacionários) ou móveis (embarcados em navios ou lanchas). Essas arquiteturas distribuídas, ainda mais em redes de grandes dimensões como é a proposta do SisGAAz, possuem a capacidade de geração de dados em altas taxas proporcionada pela evolução das tecnologias de sensores para uso militar e exigem o uso de vários subsistemas existentes, sensores comerciais *off-the-shelf* (COTS) e de Internet das Coisas (IoT) devido à suas vantagens inerentes [88]. Para processar esses dados produzidos por diversas fontes, foram desenvolvidas técnicas de fusão de informações de múltiplos sensores distribuídas, onde as observações são processadas de forma distribuída [148–150].

Em arquiteturas distribuídas tais como a utilizada pelo SCUA (Figura 4.1), as estimativas locais usando as observações de processadores locais são feitas e então transmitidas para um centro de fusão onde a decisão final global ou estimativa é feita em termos de alguns critérios. Essa arquitetura multissensor distribuída tem muitas vantagens, como mais capacidade, confiabilidade, robustez e capacidade de sobrevivência do que a arquitetura centralizada [151]. As simulações apresentadas

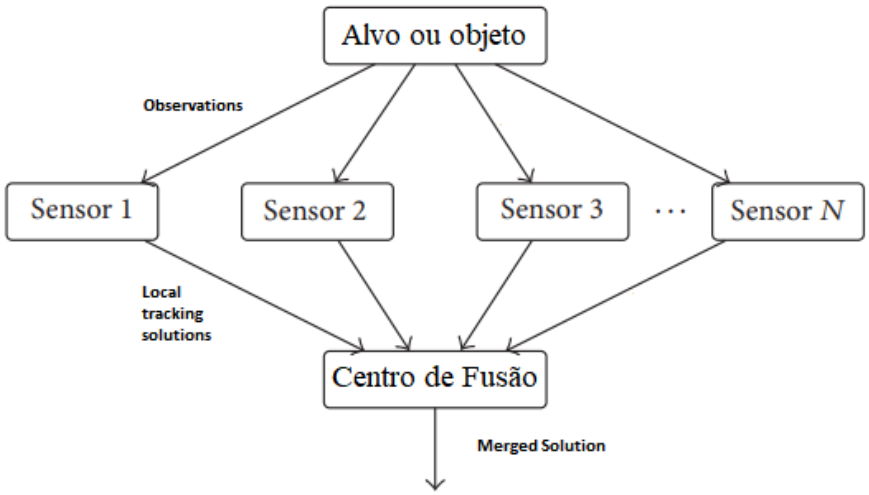

Figura 4.1: Arquitetura de sistema distribuído com fusão de dados centralizada.

neste capítulo utilizam esse mesmo esquema, o qual é ilustrado por uma rede de 2 sensores estacionários que realizam a solução local de MTT e uma estação central responsável pela fusão da informação oriunda dos sensores distribuídos.

4.2 Simulações

Por questões de confidencialidade e acesso aos dados, as simulações aqui apresentadas são baseadas em observações obtidas por meio do MATLAB Sensor Fusion®, esses dados são utilizados para ilustrar os resultados do esquema de fusão proposto.

4.2.1 Cenário simulado

Neste cenário de simulação, a rede distribuída é composta por dois sensores estacionários (radares 2D), os quais cobrem um setor de azimute de 90 graus de uma baía. Esses radaress estão posicionados diagonalmente na área de vigilância em direções opostas, conforme Tabela 4.1. Cada um deles possui uma solução local responsável por rastrear os alvos em seus respectivos FoV e fornecer a distribuição de intensidade a posteriori. Os efeitos de latência e limitações de largura de banda na comunicação não são considerados nas simulações. Devido às relativamente pequenas distâncias envolvidas, o efeito da curvatura da Terra nas observações também não é considerado.

Este cenário foi definido de acordo com os principais aspectos: disponibilidade e facilidade de acesso aos dados. Não havia mais sensores disponíveis na coleta e análise de dados reais para preparar a simulação. Além disso, dados obtidos de sensores instalados em navios podem expor dados sensíveis de desempenho do meio. Dadas as regulamentações brasileiras, esses dados adicionais exigiriam um esforço significativo para consultar as autoridades competentes. Apesar desta limitação de dados, este cenário mostrou-se adequado para este trabalho.

Tabela 4.1: Sensores e Estação Central utilizados nas simulações.

Sensor	**Coordenadas da posição [km]**	**Setor pelo FoV [rad]**
Radar 1	$(7.38, -0.236)$	$[0, -\pi/2]$
Radar 2	$(-4, 9)$	$[-\pi/2, \pi/2]$
Estação Central (*Tower*)	$(0, 0)$	Não aplicável

Além disso, esse cenário simulado inclui embarcações tipicamente encontradas na área marítima de interesse (Baía de Guanabara). Foram definidos cinco navios no porto no setor sob vigilância. Dois deles estão navegando em movimento circular a 20 e 30 nós. Os outros estão viajando com um movimento retilíneo de 10, 12 e 6 nós. A Tabela 4.2 resume as informações sobre os estados iniciais das embarcações.

Tabela 4.2: Estado inicial e tipo de movimento dos alvos.

Embarcação	Velocidade inicial [knot]	Orientação inicial [º]	Posição inicial [km]	Tipo de movimento
1	20	130	$(3.15, 7.4)$	Circular com raio de 200 m
2	30	120	$(-0.5, 6)$	Circular com raio de 400 m
3	10	0	$(3.2, 1.3)$	Retilíneo
4	12	0	$(-1.5, 7)$	Retilíneo
5	6	90	$(-0.6, 0.7)$	Retilíneo

A Figura 4.2 resume e ilustra o cenário de vigilância descrito.

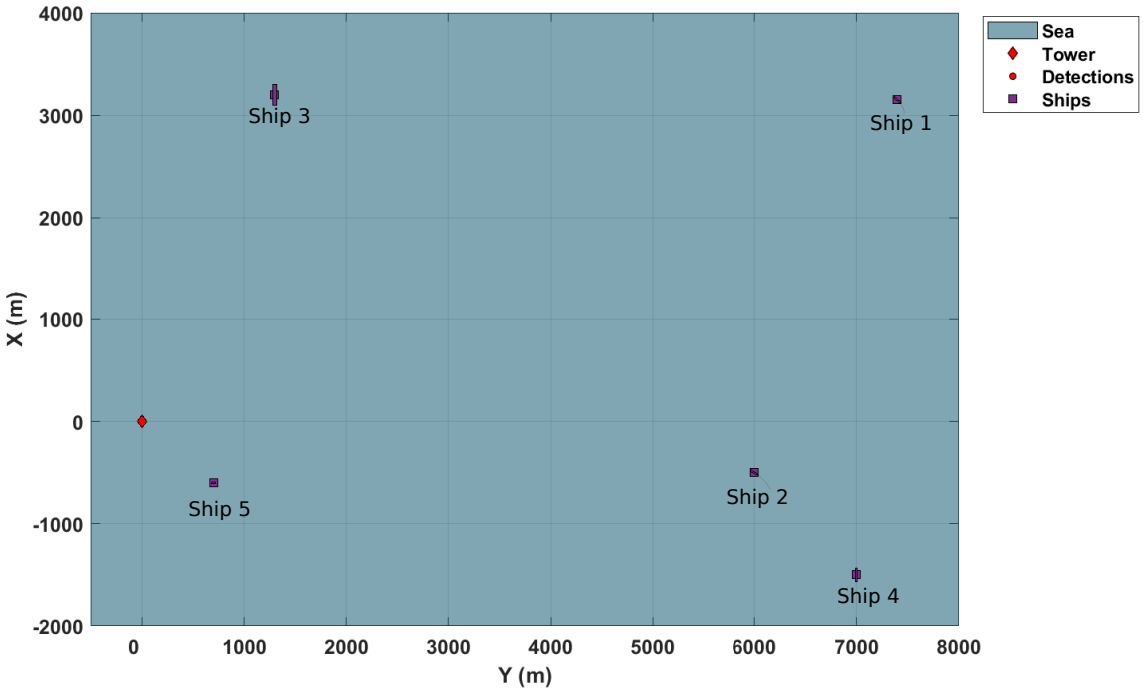

Figura 4.2: Cenário de vigilância utilizado nas simulações

4.2.2 Modelos utilizados

Modelo dinâmico dos alvos

O modelo de cada embarcação (*single-target model*) no cenário simulado é definido por:

$$\mathbf{x} = [x \ \dot{x} \ y \ \dot{y} \ \omega]^T, \tag{4.1}$$

onde (x, y) corresponde a coordenadas de posição, $(\dot{x}, \dot{y})$ corresponde a velocidades e ω é a taxa de giro constante.

Modelo dinâmico utilizado pelos filtros

Outro modelo dinâmico crucial é o utilizado pelo próprio filtro. Apesar de se tratar de um problema de rastreamento de múltiplos alvos, cada alvo precisa de um modelo de movimento para a etapa de atualização do filtro. Além disso, a escolha desse modelo está diretamente ligada ao desempenho do filtro.

O rastreamento de alvos com movimento de giro coordenado (*coordinated turn motion*) é altamente dependente dos modelos e algoritmos [152]. Como os alvos são possivelmente embarcações pequenas, rápidas e fáceis de manobrar, o modelo não linear de giro quase constante é a melhor escolha [153]. Para cada objeto, a dinâmica do estado é dada por [2]:

$$\mathbf{x}_k = F(\omega_{k-1})\mathbf{x}_{k-1} + Gw_{k-1} \tag{4.2}$$

$$\omega_k = \omega_{k-1} + \Delta u_{k-1}, \tag{4.3}$$

onde $\mathbf{x}_k = [x_k \ y_k \ \dot{x}_k \ \dot{y}_k]^T$ corresponde às coordenadas de posição e velocidades no instante k, ω é a taxa de variação de giro no tempo k, $w = [w_x \ w_y]^T$ é um ruído Gaussiano de média zero e matriz de covariância $\sigma_\omega \in \mathbb{R}^{2\times 2}$, u é uma variável aleatória com média zero e covariância σ_u,

$$F(\omega) = \begin{bmatrix} 1 & 0 & \frac{\sin\omega\Delta}{\omega} & -\frac{1-\cos\omega\Delta}{\omega} \\ 0 & 1 & \frac{1-\cos\omega\Delta}{\omega} & \frac{\sin\omega\Delta}{\omega} \\ 0 & 0 & \cos\omega\Delta & -\sin\omega\Delta \\ 0 & 0 & \sin\omega\Delta & \cos\omega\Delta \end{bmatrix},$$

$$G = \begin{bmatrix} \frac{\Delta^2}{2} & 0 \\ 0 & \frac{\Delta^2}{2} \\ \Delta & 0 \\ 0 & \Delta \end{bmatrix},$$

e Δ é o intervalo de amostragem.

Para cada alvo e sensor estacionário, o modelo de observação consiste em medições de distância e ângulo, de acordo com:

$$z_k = \begin{bmatrix} \sqrt{(x_k - x_s)^2 + (y_k - y_s)^2} \\ \text{atan2}\left(\frac{y_k - y_s}{x_k - x_s}\right) \end{bmatrix} + \epsilon_k, \tag{4.4}$$

onde (x_k, y_k) é a posição objeto no instante k, (x_s, y_s) é a posição do sensor e ϵ_k é o ruído Gaussiano de média zero do processo com matriz de covariância $R_k \in \mathbb{R}^{2\times 2}$.

4.2.3 Parâmetros utilizados nos filtros

Os filtros GM PHD utilizados nas simulações são os descritos nas Equações (2.31) e (2.32). Cada um deles permite estimar as posições das embarcações individualmente, conforme descrito anteriormente. Como o modelo preditivo é não linear, o filtro considera a versão EKF na etapa de atualização (ver [2]). A Tabela 4.3 apresenta os parâmetros de filtro usados nas simulações.

Além disso, não há *spawning* e o RFS de nascimento de alvos é Poisson, ou seja, é IID no FoV do sensor, com taxa de 10^{-5}. Os objetos são inicializados com um peso inicial unitário e uma matriz de covariância inicial

$$P_{birth} = \begin{bmatrix} 10^5 & 0 & 0 & 0 & 0 \\ 0 & 10^5 & 0 & 0 & 0 \\ 0 & 0 & 10^5 & 0 & 0 \\ 0 & 0 & 0 & 10^5 & 0 \\ 0 & 0 & 0 & 0 & 1000 \end{bmatrix}.$$

Tabela 4.3: Parâmetros utilizados pelos filtros.

Parâmetro	**Valor**
Alcance máximo do sensor	12 km
σ^2 para a distância	25 m
σ^2 para ângulo	0.5°
Probabilidade de sobrevivência (p_S)	0.99
Probabilidade de detecção (p_D)	0.9
Sensor *field-of-view* (FoV)	90°
Clutter	2×10^{-8}
Prune threshold (T)	10^{-6}
Merge threshold (U)	25
Extraction threshold (E)	0.8
Número máximo de componentes da GM	1000

Por fim, o RFS para os falsos alarmes segue o modelo uniforme de Poisson sobre o FoV do sensor, sendo a região de vigilância $[-\pi/2,\ \pi/2]$ rad $\times$ $[0,\ 12]$ km.

Cada radar estima independentemente as coordenadas dos alvos, essa informação precisa ser fundida. Em seguida, o processamento central simulado mescla essas distribuições, obtendo a solução final de forma distribuída e cooperativa. Da mesma forma que [58], o processo de *merge* é aquele definido na Equação 1.6. Nesse esquema de fusão proposto, a função intensidade de cada elemento da rede é fundida por meio

da clusterização baseada na distância de Mahalanobis, ou seja, um passo adicional de *merge*.

4.2.4 Resultados obtidos

Com base na métrica OSPA exibida na Figura 4.9, os resultados apresentados nas simulações são satisfatórios. Inicialmente, para ilustrar o desempenho alcançado, o sistema de rastreamento simulado utiliza apenas um radar, cujo FoV é representado pela região delimitada por linhas vermelhas (como nas demais figuras). O rastreador apresenta bons resultados, ou seja, consegue rastrear os alvos observados pelo sensor. As Figuras 4.3 e 4.4 ilustram esses resultados.

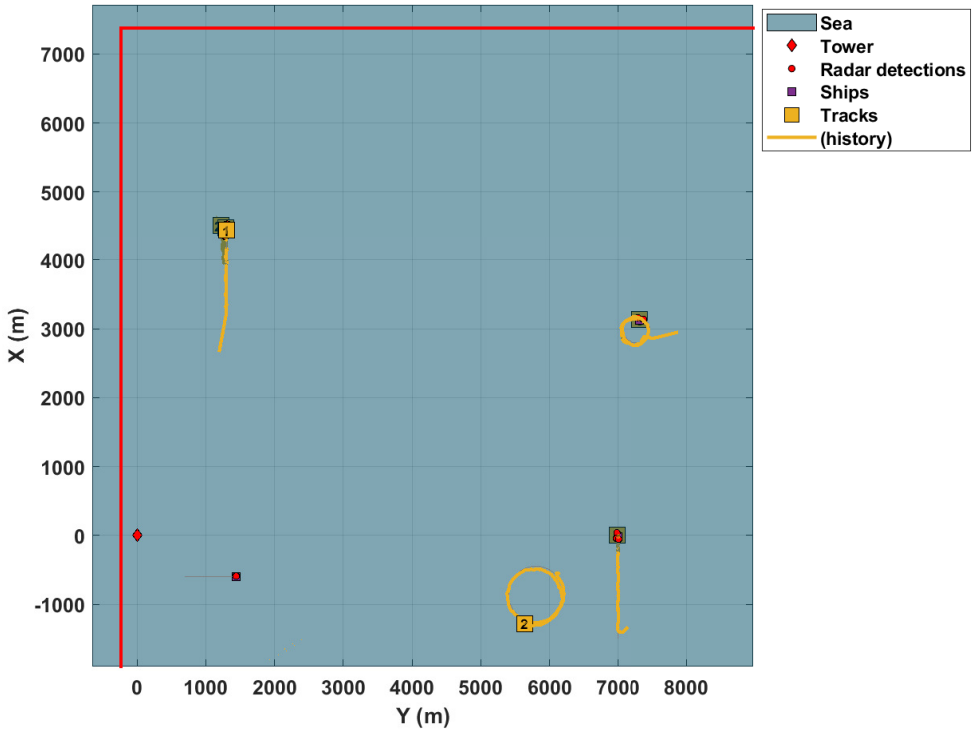

Figura 4.3: Rastreamento com um sensor.

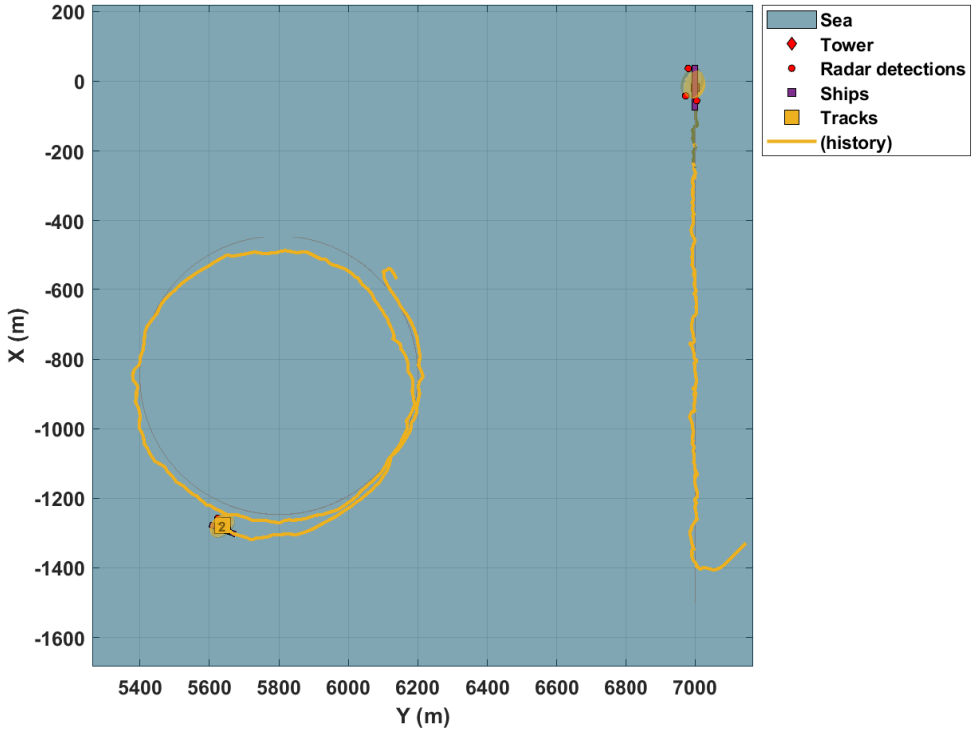

Figura 4.4: Rastreamento de dois alvos com trajetórias diferentes.

Como declarado anteriormente, esses figuras ilustram o desempenho do sistema de rastreamento com apenas um sensor. Além disso, mostram que os navios são rastreados independentemente de o movimento realizado ser linear ou circular, apesar de um transitório inicial. Ressalta-se que as embarcações apresentam diferenças significativas em suas velocidades e movimentos: simultaneamente, uma navega a 12 nós, enquanto a outra gira a 30 nós.

A Figura 4.4 apresenta uma visão ampliada de dois alvos com movimentos diferentes, um linear e outro circular. Essa figura reforça a capacidade de rastreamento do sistema em diferentes condições de movimento do alvo. Além disso, atesta a boa escolha do modelo dinâmico de taxa de giro para os alvos. Essa escolha é um fator essencial para o sucesso do sistema de rastreamento, conforme afirma [152].

Apesar da escolha bem sucedida para o modelo de predição, a Figura 4.5 ilustra um problema existente em sistemas reais, chamado oclusão de alvos. A Figura 4.5 exibe essa limitação ao se utilizar apenas um radar. A embarcação de maior dimensão oclui a menor, portanto, ela não é detectada pelo sistema de rastreamento. Tendo em vista que o sistema visa combater atividades ilegais, as quais poderiam ser realizadas na região por embarcações de pequeno porte, é imprescindível a utilização de sensores distribuídos por toda a área de vigilância. Este fato é ainda reforçado pelas grandes dimensões da área de vigilância.

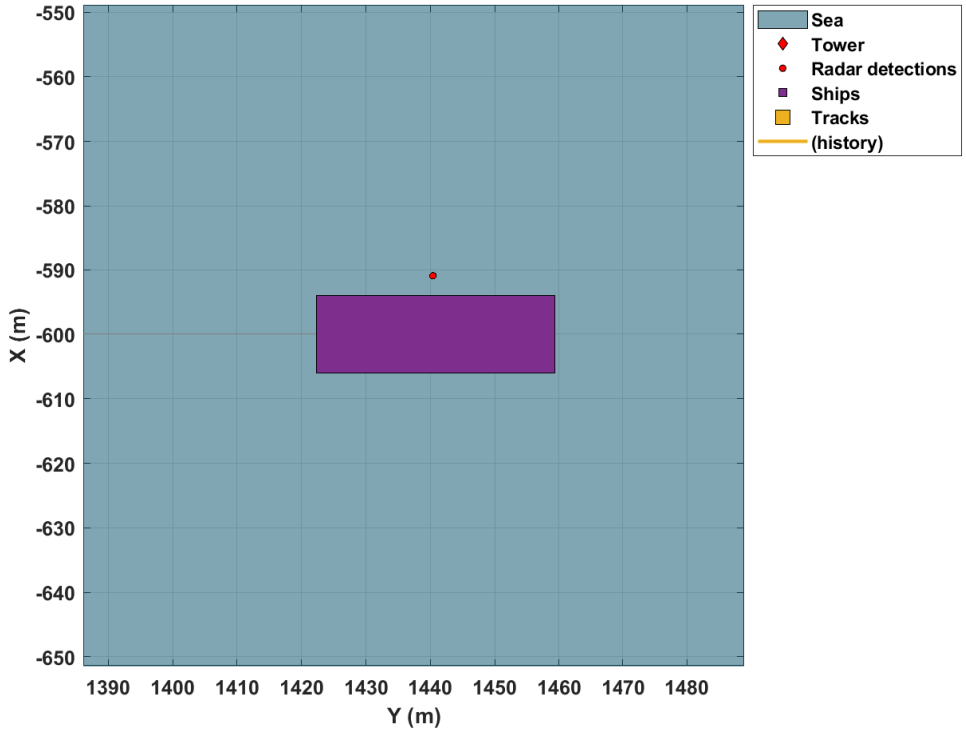

Figura 4.5: Alvo em oclusão

A inserção de um segundo radar evita a oclusão de alvos, conforme ilustrado na Figura 4.6.

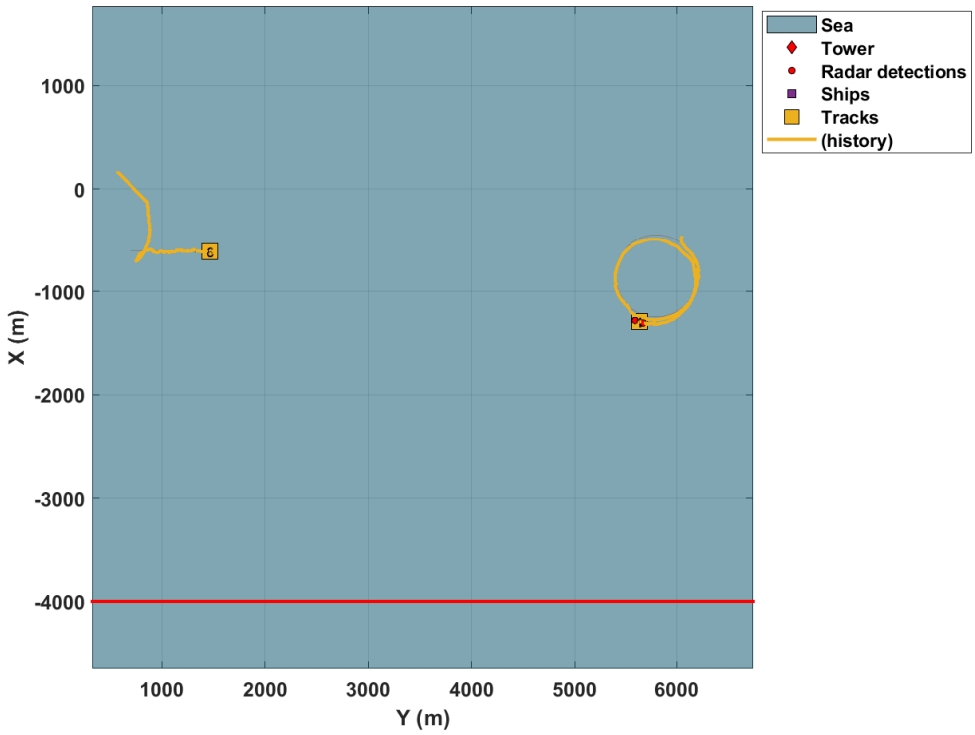

Figura 4.6: Resultado após a inserção de um segundo radar.

As Figuras 4.7 e 4.8 apresentam os resultados para o esquema de fusão proposto, ou seja, uma rede de sensores distribuído, com dois radares.

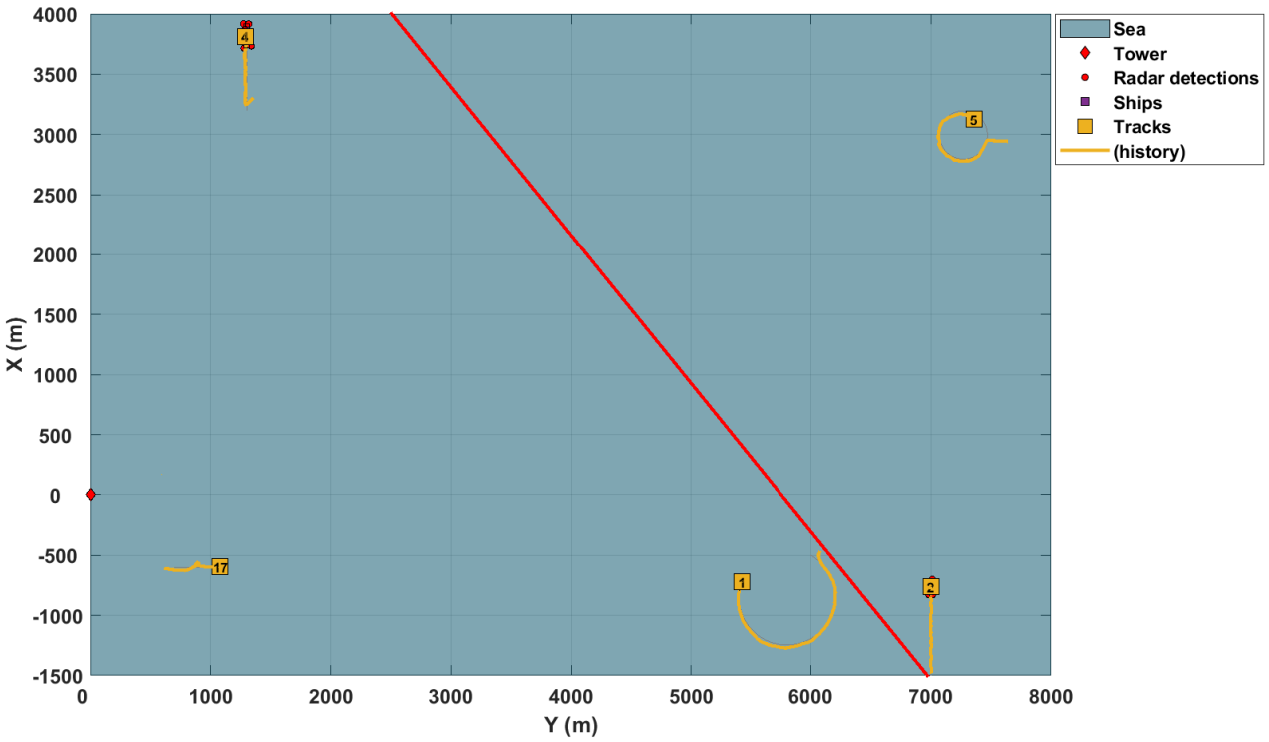

Figura 4.7: Solução de *tracking* para o esquema proposto.

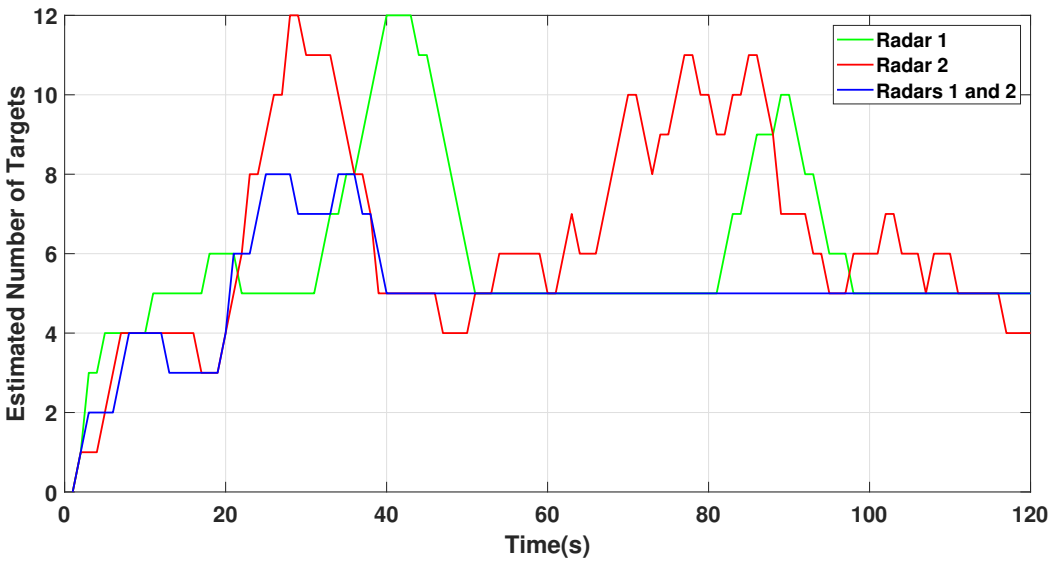

Figura 4.8: O número de alvos estimados pelas soluções locais individualmente e pelo esquema proposto.

É possível observar que a combinação das informações de cada solução local permite uma convergência mais rápida para o número de alvos reais. Além disso, é mais robusto, pois o aumento das fontes de dados (sensores) no sistema evita a

oclusão de alvos, comparando-se ao caso com apenas um radar.

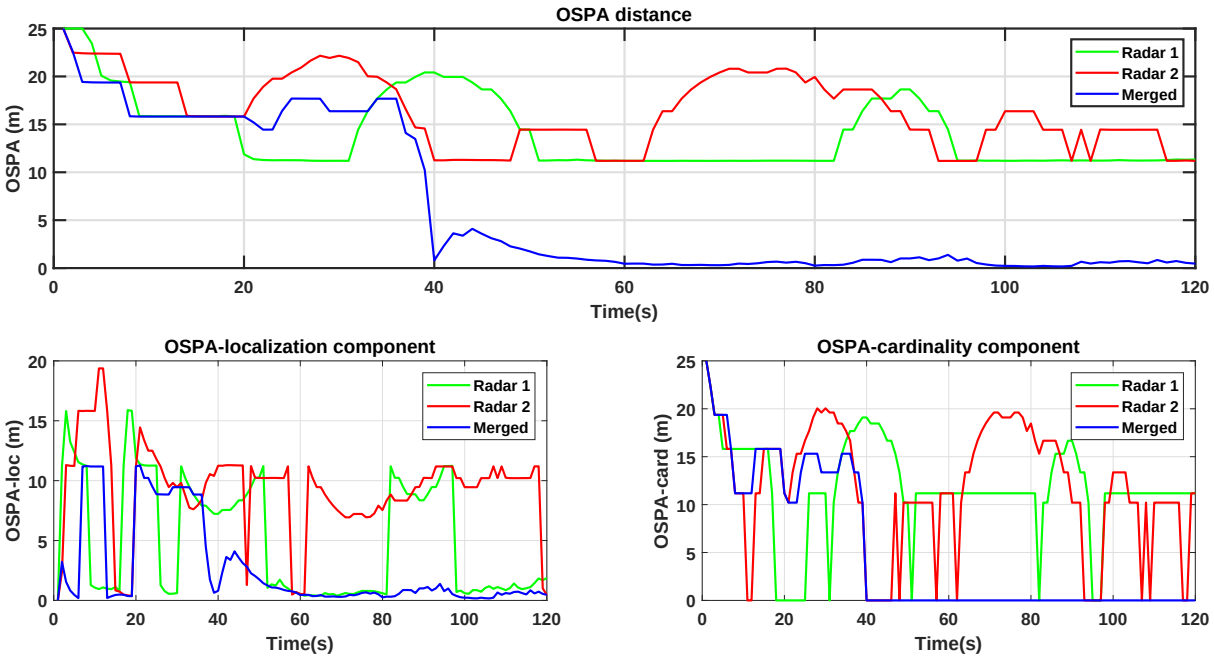

Figura 4.9: Métrica OSPA.

Por fim, a Figura 4.9 apresenta a métrica OSPA e seus componentes, corroborando as análises qualitativas baseadas nas figuras anteriores. A solução proposta é mais rápida e estável, como pode ser observado nos gráficos, principalmente, após o tempo de simulação de 40 s. Cabe ressaltar que quanto menor o valor da métrica, melhor o desempenho do filtro. Além disso, o resultado para o esquema proposto apresenta melhores resultados para componentes OSPA de localização e cardinalidade. O componente de localização está relacionado ao erro de posição, como pode ser observado na Figura 4.9, após 40 s, o valor tende a zero. Da mesma forma, no mesmo instante, o componente de cardinalidade torna-se zero.

4.3 Considerações finais

(As simulações ilustram o problema de oclusão de alvos quando apresentam diferentes geometrias e dimensões. A inserção de um novo sensor, e o consequente aumento da rede de sensores, evita a oclusão dos alvos e melhora o desempenho geral do sistema de rastreamento. Esses resultados reforçam a afirmação de Vo e Ma [2]: A eficiência e simplicidade na implementação da recursão do filtro GM PHD também sugerem uma possível aplicação para rastreamento em redes de sensores.

O algoritmo utilizado possui limitações de desempenho já demonstradas em outras pesquisas. A arquitetura proposta apresenta bons resultados ao aplicar uma

etapa adicional ao algoritmo original, a principal contribuição deste trabalho: um esquema de fusão de dados de baixa complexidade. Além disso, este esquema proposto apresenta resultados promissores, principalmente quando se trata do problema de oclusão do alvo. A questão da oclusão evidencia, ainda, para a aplicação, a necessidade de uma rede de sensores distribuídos, dada a missão do sistema e a extensão da área de interesse. Adicionalmente, é possível concluir que o aumento da área coberta, utilizando sensores de menor capacidade, também contribui para o aspecto econômico deste empreendimento modular.

Capítulo 5

Filtro PHD e computação paralela

Este capítulo tem como objetivo avaliar o uso de métodos de paralelização computacional combinados com filtro PHD. Essa classe de filtro surge como a base de todas as abordagens fundamentadas em RFS, tanto em problemas de localização & mapeamento simultâneo e mapeamento em robótica quanto no rastreamento de múltiplos alvos. Após uma motivação para uso de versões paralelizadas do filtro, apresenta-se o modelo computacional Gamma, o qual já vem sendo investigado para aplicações em sistemas de rastreamento. Esse modelo é baseado em reações químicas sendo introduzido para eliminar a sequencialidade dos cálculos executados, produzindo um paralelismo intrínseco em consequência de uma execução não determinística dessas computações. Por fim, demonstra-se o potencial para a paralelização do filtro GM PHD por intermédio de simulações.

5.1 Motivações para um filtro PHD paralelizado

Técnicas de filtragem baseadas em RFS, desempenham um papel de destaque em aplicações de rastreamento de múltiplos alvos e robótica no problema de mapeamento. O filtro PHD e suas variações representam a realização computacional primária dessa abordagem.

Apesar da ampla difusão dessa técnica, a investigação em aplicações de tempo real ainda é incipiente, principalmente, quando se trata de redes de sensores ou agentes (por exemplo, robôs) que colaboram na troca de dados. Existem duas abordagens relevantes para essa questão. A primeira estratégia preza por limitar os dados propagados na rede, enquanto a segunda em aumentar a velocidade de computação do algoritmo.

Em uma rede de sensores cooperativos, um filtro SMC PHD sequencial combinado a um algoritmo de baixo custo de comunicação baseado na disseminação aleatória de informações é proposto em [87]. As referências [154, 155] propõem uma abordagem de consenso de cardinalidade chamada prior-CC para uma filtragem PHD

com base em uma rede de sensores descentralizada com comunicação severamente restrita. Ao contrário das abordagens de comunicação de filtragem serial já existentes, o esquema proposto permite que a comunicação entre cada nó seja realizada em paralelo com o cálculo da solução local do filtro.

Há uma demanda crescente por poder computacional para a resolução de problemas complexos, antes considerados improváveis de serem solucionados. Por exemplo, os carros autônomos, que exigem grandes quantidades de computação para o processamento de dados de inúmeros sensores para tomar decisões em tempo real com base nessa informação. O paralelismo na execução, onde vários cálculos são realizados simultaneamente, tornou-se a maneira mais comum de fornecer níveis mais altos de poder computacional, aproveitando unidades computacionais de *hardware* massivamente paralelo [156].

A pesquisa em computação paralela é essencial para assegurar o progresso futuro dos computadores convencionais. Essa, no que lhe concerne, requer programas de *benchmark* para comparar plataformas, identificar gargalos de desempenho e avaliar as soluções potenciais [157].

De acordo com [158], a paralelização de filtros PHD não é uma questão que tenha sido explorada diretamente até 2016. Além disso, esse trabalho apresenta um *framework* para filtragem paralela baseado em processamento múltiplo, que paraleliza totalmente todas as etapas do filtro SMC PHD. De acordo com [159], embora não haja problema de associação de dados no filtro PHD, dificilmente pode ser implementado em tempo real em um processador serial. Esse artigo propõe um esquema de paralelização e implementação para o filtro SMC PHD na unidade de processamento gráfico (GPU) sob o *framework Compute Unified Device Architecture* (CUDA) [160]. Até onde é sabido, não há relato sobre a paralelização da versão do filtro PHD de mistura gaussiana (GM), apesar de existir um potencial para a paralelização de partes do filtro, como a etapa de atualização. Nessa etapa, são realizadas combinações entre a informação *a priori* e a informação observada, na qual cada execução dessa combinação independe das demais.

O GAMMA (*General Abstract Model for Multiset Manipulation*) foi proposto em 1986 para especificação de programas baseados na reescrita de multiconjuntos (*multisets*) paralelos. De acordo com [161], o modelo possui execução não determinística, pois seus elementos podem reagir livremente em modo paralelo, tornando desnecessária a recursão explícita. Essa abordagem conta com uma metáfora de reação química: a única estrutura de dados é o multiconjunto, cuja computação pode ser vista como uma sucessão de reações químicas consumindo elementos do multiconjunto e produzindo novos elementos de acordo com regras particulares [162]. Essa técnica já é utilizada em outros algoritmos de rastreamento para acelerar a computação em uma complexa rede de sensores [163–165].

É importante destacar que a melhoria de desempenho desses algoritmos de rastreamento ganha importância significativa quando em sistemas de rastreamento cooperativo como sistema de vigilância marítima, tal como apresentado em [88, 89, 103]. Essas referências enfatizam a relevância e o impacto da Internet das Coisas (IoT) em uma rede de sensores distribuídos. Nesse sentido, Gamma pode ser usado para explorar a velocidade local combinada com melhorias de comunicação paralela no contexto da computação fluida [166]. Versões paralelas (por exemplo, *dynamic dataflow* [167]) de programas imperativos ou funcionais podem ser definidas, mas nenhuma delas pode genuinamente modelar a total ausência de ordenação entre elementos que pode ser alcançada pelos programas em Gamma [168].

5.2 O modelo Gamma

5.2.1 Motivação para a paralelização Gamma

Computadores sequenciais estão se aproximando dos limites físicos de seu poder computacional, apesar do incrível ritmo de desenvolvimento da tecnologia de processadores. Em um sistema de computação sequencial, cada processador é aplicado para realizar uma operação por vez, ou seja, quando uma instrução é fornecida em um determinado momento, a próxima instrução deve aguardar a execução da primeira instrução.

Em vez disso, em uma computação de sistema paralelo, vários processadores cooperam para resolver um problema, reduzindo o tempo de computação porque operações podem ser executadas simultaneamente. Em outras palavras, divide-se o trabalho em operações sequenciais menores, as quais são realizadas concomitantemente.

É importante destacar a distinção entre paralelismo lógico e físico. Enquanto o paralelismo físico está relacionado à implementação, que corresponde à distribuição de tarefas em vários processadores, o paralelismo lógico é a possibilidade de descrever um programa como uma composição de várias tarefas independentes [169].

Modelos paralelos de computação são relevantes para a análise e projeto de algoritmos em conjunto com a avaliação de desempenho. Esses modelos devem ser independentes de *hardware* ou arquitetura. Como efeito, há muita diversidade neste campo. Algoritmos desenvolvidos adotando modelos paralelos devem ser implementáveis e executáveis em computadores com diferentes arquiteturas [170].

O modelo de computação baseado em reações químicas foi introduzido para eliminar a sequencialidade dos cálculos executados para resolver um determinado problema e assim fornecer um modelo de execução não determinístico [171]. O Gamma foi o primeiro desses, tendo sido proposto em 1986 [162], o qual fornece

paralelismo intrínseco como consequência dessa execução não determinística.

5.2.2 Conceitos fundamentais em Gamma

O modelo sequencial de computação baseado na arquitetura von Neumann sempre desempenhou um papel de liderança no design de muitas linguagens de programação criadas no passado, principalmente por dois motivos [168]:

- Modelos de execução sequencial forneceram uma forma adequada de abstração para algoritmos, atendendo a percepção intuitiva de um programa definido como receita para alcançar o resultado pretendido;
- As implementações foram realizadas em arquiteturas de processador único, refletindo essa visão sequencial abstrata.

Nesse contexto, a maioria das linguagens de programação são baseadas no paradigma imperativo, que supostamente engloba a arquitetura tradicional de von Neumann. Consequentemente, os programas contêm instruções sequenciais, que nem sempre são impostas pela lógica do problema em si, mas sim pelo modelo computacional.

As linguagens de programação funcionais representam uma tentativa de formalismo de alto nível não imperativo. No entanto, elas fazem uso massivo da recursão, tanto em estruturas de dados quanto em programas, o que acaba se tornando uma forma disfarçada de sequencialidade [172].

É fundamental destacar as questões relacionadas à separação entre o problema e a implementação. Seria outro fator complicador na programação paralela, pois várias linhas de controle sequenciais precisariam ser gerenciadas simultaneamente.

O modelo Gamma surge como um formalismo de alto nível para especificar programas de forma elegante e naturalmente paralela por meio do operador Γ e sua estrutura de dados: o *multiset* que pode ser do tipo real, character, integer, tuple, ou mesmo outro *multiset*.

Este modelo pode ser descrito como um transformador do *multiset*: a computação é uma sucessão da aplicação de regras que consomem seus elementos enquanto novos são produzidos no *multiset*. A computação termina quando nenhuma regra pode ser aplicada [172].

As operações básicas em *multisets* são as seguintes [169]:

- união: o número de ocorrências de um elemento em $M_1 + M_2$ é a soma de seus números de ocorrências em M_1 e M_2;
- diferença: o número de ocorrências de um elemento em $M_1 - M_2$ é a diferença entre seus números de ocorrências em M_1 e M_2 (se essa for maior ou igual a zero; caso contrário, é zero);

- produto: $M \times N$ é o produto cartesiano entre M e N;
- máximo: o número de ocorrências de um elemento em $M \cup N$ é o máximo de seus números de ocorrências em M e N;
- mínimo: o número de ocorrências de um elemento em $M \cap N$ é o mínimo de seus números de ocorrências em M e N;
- cardinal: $card(M)$ fornece a soma dos números de ocorrências de todos os elementos do multiconjunto M.

O conceito central em Gamma é o operador Γ. Essa estrutura de controle possibilita a execução de um programa de acordo com o modelo e pode ser formalmente definida por (adaptado de [172]):

$$\begin{aligned}
&\Gamma((R_1, A_1), ..., (R_m, A_m))(M) = \\
&\textbf{if } \forall\, i \in [1, m], \forall\, x_1, ..., x_n \in M, \neg R_i(x_i, ..., x_n) \\
&\textbf{then } M \\
&\textbf{else let } x_1, ..., x_n \in M, \text{let } i \in [1, m] \\
&\text{such that } R_i(x_i, ..., x_n) \text{ in} \\
&\Gamma((R_1, A_1), ..., (R_m, A_m))((M - \{x_1, ..., x_n\}) + A_i(x_1, ..., x_n))
\end{aligned} \tag{5.1}$$

Essa definição pode ser interpretada como segue. Os pares de funções (R_i, A_i) representam as reações aplicadas ao multiconjunto (M) e especificam como as reações foram usadas e suas condições de execução. A aplicação do par (R_i, A_i) em M resulta na substituição em M de um subconjunto de elementos $(x_1, ..., x_n)$ tal que $R_i(x_1, ..., x_n)$ é verdadeira para os elementos de $A_i(x_1, ..., x_n)$. Caso o elemento não satisfaça a condição de nenhuma reação, o resultado do cálculo é o próprio M inicial. Caso contrário, o resultado é o multiconjunto M subtraído do subconjunto de elementos $(x_1, ..., x_n)$ mais o subconjunto especificado pelos elementos $A_i(x_1, ..., x_n)$, ou seja, $((M - \{x_1, ..., x_n\}) + A_i(x_1, ..., x_n))$. Portanto, no caso de condições verdadeiras ocorrerem, há uma consequente transformação do multiconjunto existente.

É importante perceber que as reações ocorrem de forma não determinística. Portanto, as condições de reação podem ser satisfeitas simultaneamente e em qualquer ordem. Por esse motivo, a decisão de qual reação ocorrerá, uma vez que essas opções podem ser diversas e independentes, será não determinística e simultânea. Esse fato confere ao Gamma uma característica naturalmente paralela.

5.2.3 Gamma aplicado à extração de estados

Para demonstrar a aplicabilidade do paradigma Gamma ao filtro GM PHD em sistemas de rastreamento ou SLAM, esta seção deriva um exemplo de paralelização para a extração de estados apresentada no Algoritmo 1.

Para um multiconjunto definido para uma Mistura Gaussiana com componentes J_k, com i-ésimo elemento composto por $\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\}$, três condições de reação são derivadas de acordo com

$$\begin{aligned} &REDUCE = replace\left(\left\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\right\}\right) \\ &\text{by } NULL \\ &\text{if } w_k^{(i)} \leq 0.5; \end{aligned} \tag{5.2}$$

$$\begin{aligned} &EXPAND = replace\left(\left\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\right\}\right) \\ &\text{by } \left\{0, m_k^{(i)}, P_k^{(i)}\right\}, \left\{w_k^{(i)} - 1, m_k^{(i)}, P_k^{(i)}\right\} \\ &\text{if } w_k^{(i)} > 1.5; \end{aligned} \tag{5.3}$$

e

$$\begin{aligned} &ESTIMATE = \\ &replace\left(\left\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\right\}, \left\{w_k^{(j)}, m_k^{(j)}, P_k^{(j)}\right\}\right) \\ &\text{by } \left\{0, m_k^{(i)}, 0\right\} \cup \left\{0, m_k^{(j)}, 0\right\} \\ &\text{if } true. \end{aligned} \tag{5.4}$$

O algoritmo Gamma apresentado tem como multiconjunto inicial J_k Gaussianas para um dado instante k, representado por tuplas de três elementos: peso, média e a matriz de covariância. Nesse exemplo, o algoritmo possui essas três reações conectadas pelo operador sequencial ";", o que implica que uma reação só pode iniciar sua execução quando a anterior já tiver terminada. Essa sequencialidade é necessária para garantir o correto funcionamento do algoritmo, o que não prejudica a exploração do paralelismo, uma vez que ocorre no escopo interno de cada reação. Inicialmente, a reação REDUCE é realizada para remover do multiconjunto todas as Gaussianas (tuplas) cujo peso seja menor ou igual a 0.5, uma vez que tais Gaussianas não serão utilizadas para o estado final estimado. A reação definida pela Equação 5.2 seleciona qualquer tupla do multiconjunto e a remove ("by null") se a condição de reação $w_k^{(i)} \leq 0.5$ for satisfeita. Essa operação pode ser realizada em paralelo sem

impor nenhuma restrição de ordem ou sequencialidade neste processamento.

Após a reação REDUCE, inicia-se a execução da segunda reação definida pela Equação 5.3, que visa replicar as Gaussianas do multiconjunto em uma quantidade proporcional ao seu peso, preparando assim o banco de dados para a fase final do algoritmo, que precisa concatenar as médias gaussianas. Nesse caso, a função *replace* seleciona qualquer tupla do multiconjunto e a duplica em novas tuplas que diferem apenas em seu peso, sendo que uma recebe um peso igual a zero e a outra o original subtraído de uma unidade. Essa reação só ocorre se a tupla selecionada tiver um peso $w_k^{(i)} > 1.5$, o que garante que a replicação da tupla seja realizada em uma quantidade adequada com base em seu peso original. Como no caso da reação REDUCE, a transformação do multiconjunto durante a reação EXPAND pode ser processada totalmente em paralelo com a criação de réplicas ocorrendo para cada tupla existente.

Este método estima o estado de uma Mistura Gaussiana de forma similar àquela definida no Algoritmo 1. É importante ressaltar que todos os componentes GM já estão no *multiset*, devidamente replicados de acordo com seu peso original. Essa reação ESTIMATE, definida na Equação 5.4, ocorre selecionando quaisquer pares de tuplas do multiconjunto e substituindo-os por uma única tupla cujo segundo elemento (média) é a concatenação das médias das tuplas escolhidas. O primeiro e terceiro elementos são substituídos por 0 uma vez que o peso e a matriz de covariância não são usados para a extração do estado. Essa reação ocorre sem que nenhuma condição seja satisfeita, de modo que ao final do cálculo restará apenas uma única tupla, cujo segundo elemento será a concatenação final das médias das gaussianas. Como nas reações anteriores, ocorre o processamento de pares de tuplas, explorando todo o paralelismo possível para este tipo de cálculo. A Figura 5.1 apresenta como exemplo uma sequência de execução do algoritmo proposto.

5.3 Simulações e resultados

Apesar dos trabalhos já apresentados, aplicações práticas em Gamma carecem de investimentos nos ambientes de execução atualmente fornecidos [173]. O uso desse modelo como uma linguagem de programação real requer a disponibilidade de um compilador que consiga gerar um binário executável sobre alguma arquitetura existente. Por exemplo, em [174], apresenta-se uma implementação de compilador Gamma que consegue gerar códigos em C/CUDA para serem executados sobre plataformas paralelas e distribuídas de hardware, como um *cluster* de GPUs. Todavia, são poucos os trabalhos que buscam atacar esse problema, de modo que o acesso a um ambiente completo para se testar programas em Gamma é ainda bastante restrito. Por esse motivo, opta-se neste trabalho por exemplificar a modelagem de

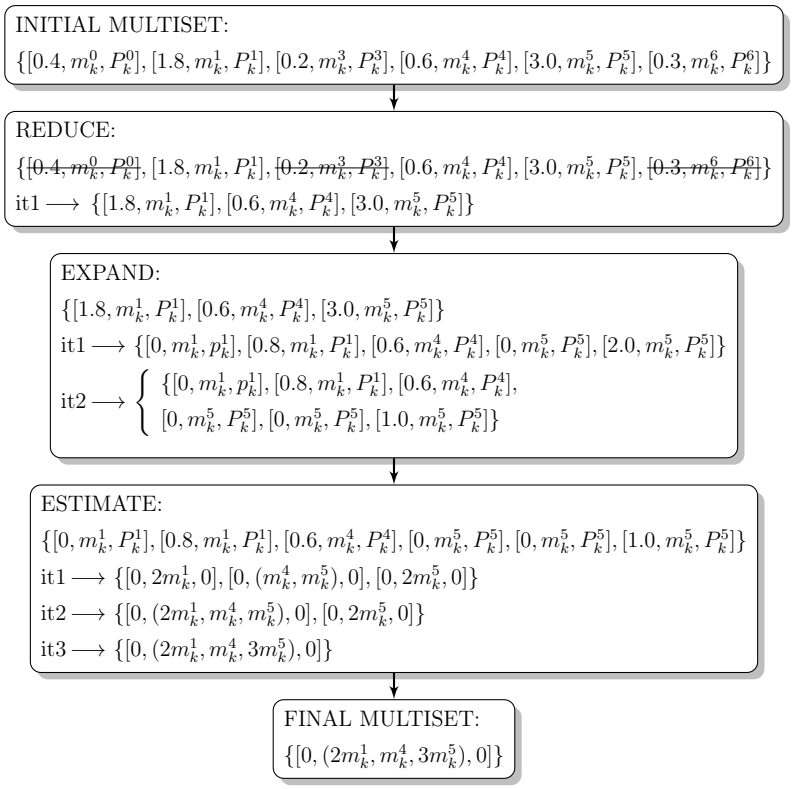

Figura 5.1: Exemplo de sequência de execução para o algoritmo proposto.

um trecho do algoritmo do filtro GM PHD em Gamma.

Adicionalmente, utiliza-se o *Parallel Computing Toolbox* do MATLAB para ilustrar o potencial do emprego de técnicas de paralelização no filtro GM PHD, cujos ganhos de desempenho poderão ser obtidos de maneira similar nas plataformas Gamma. Para as simulações, utiliza-se um PC com um processador Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz quad core.

Nessa ferramenta de processamento paralelo, cada *core* tem um *pool* associado, o qual é denominado *worker*, conforme ilustrado na Figura 5.2

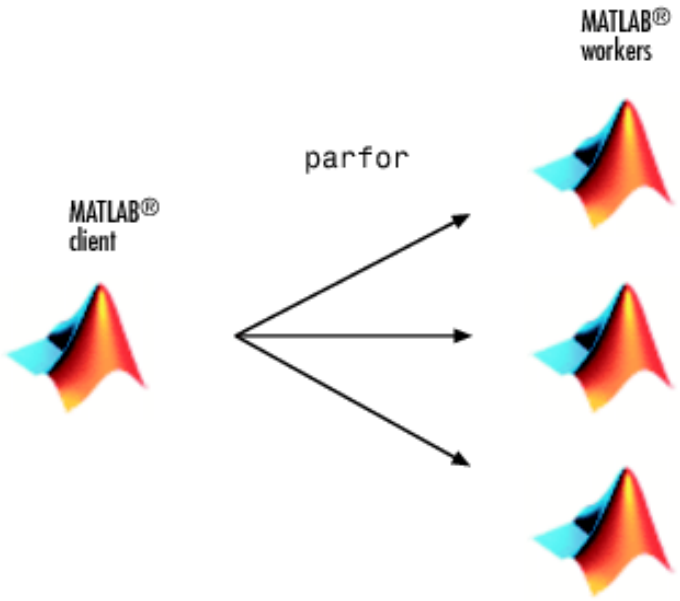

Figura 5.2: Execução paralela de um loop usando o comando parfor.
Fonte: disponível em Matlab help center.

5.3.1 Paralelização na extração de estados

Nessa seção, são apresentados os resultados da paralelização do Algoritmo 1. Esse algoritmo tem sua importância para o *plot* dos alvos ou objetos, seja numa Interface Homem-Máquina (IHM) ou para transmissão entre os elementos da rede, bem como para a tomada de decisão de um sistema de combate, por exemplo.

Foram realizadas simulações do mesmo cenário utilizado no Capítulo 3. O resultado é ilustrado por meio da Figura 5.3, a qual compara o resultado entre o tempo de processamento entre extração no processamento sequencial e paralelo (com um *pool* de quatro *workers*). A primeira linha da figura apresenta os valores do tempo de processamento para cada caso, enquanto na segunda linha, é apresentado uma ampliação da imagem para permitir uma melhor visualização. Já a terceira linha, apresenta a diferença entre os tempos de processamento $t_{paralelo} - t_{sequencial}$.

Da observação da imagem, é possível perceber que, com os mesmos dados e condições (cenário, número de objetos etc.), o processamento paralelo é um pouco mais rápido que o sequencial na maior parte do tempo. Para o caso ilustrado, a diferença média entre esses tempos foi 0.2221ms, com as diferenças máxima e mínima de aproximadamente 164ms e -170ms (ponto em que o processamento sequencial foi mais rápido), respectivamente.

Nesse mesmo caso, o tempo médio de processamento no caso paralelo foi de 2.0344ms e no caso sequencial de 2.2565ms, levando a uma redução média de 10%. É importante frisar que mesmo o algoritmo de extração não sendo considerado custoso computacionalmente, a paralelização teve um bom resultado, a despeito do *overhead*

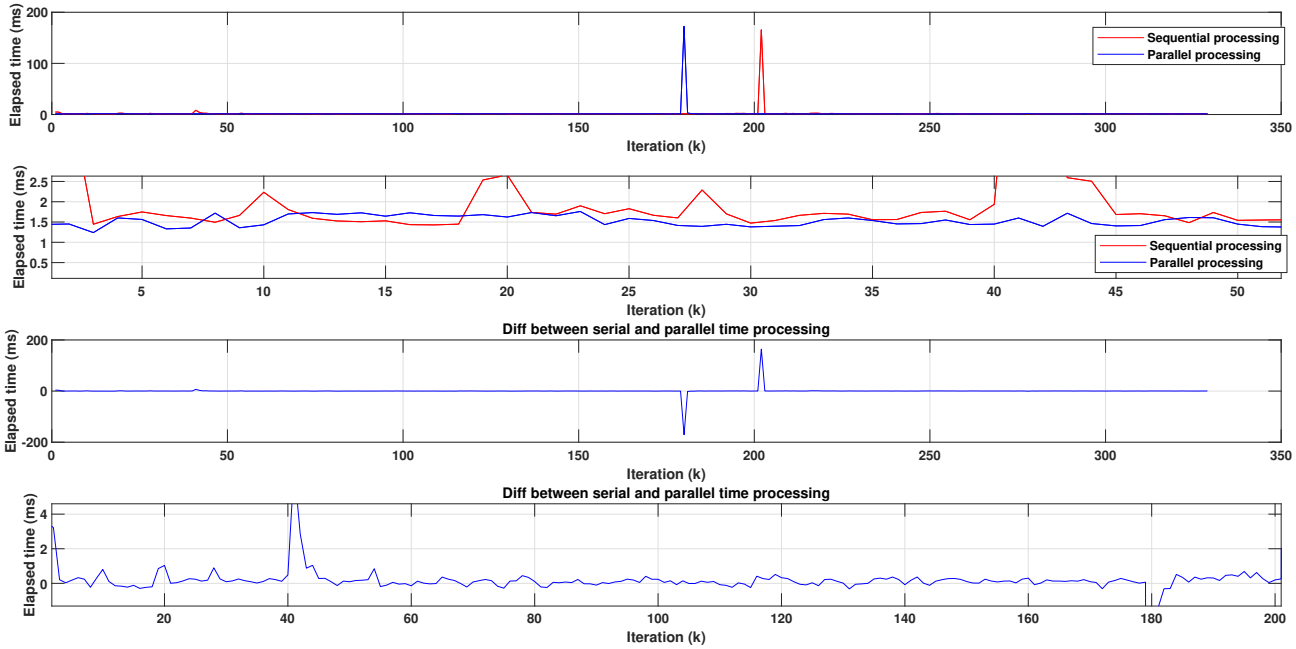

Figura 5.3: Comparação entre o tempo de processamento sequencial x paralelo.

de comunicação entre a aplicação *host* e o *pool* de paralelização.

Também é importante destacar que no cenário simulado existem apenas 8 objetos, portanto, é de se esperar um *speed up* ainda maior na aplicação em casos de maior complexidade, ou seja, com mais *landmarks*. Essa afirmação é corroborada por Lee [175], o qual afirma que o *update* pode criar quantidade significativa de componentes gaussianos, cada um consistindo de uma matriz de pesos, médias e covariâncias (relembrar que o número de componentes após a atualização do filtro PHD é de $J_{k|k-1} \times (1 + |z_k|)$). Para se ter uma noção dessa magnitude, apresenta-se a Figura 5.4, a qual apresenta um exemplo de um caso real de objetos identificados.

5.3.2 Paralelização no filtro GM PHD

Apesar da relevância da extração de estados para algumas aplicações, esse método não altera o desempenho do filtro PHD. Em consonância com [175], a maior complexidade ou custo computacional está associada ao passo de predição do filtro GM PHD. De fato, como o número de combinações entre a quantidade de observações $|z_k|$ e a função intensidade oriunda do passo de predição cresce, então o número de objetos em determinado cenário cresce.

Nesse contexto, define-se o *update* como a parte do filtro GM PHD com potencial para a paralelização. A Figura 5.5 destaca o trecho objeto da paralelização. É nesse

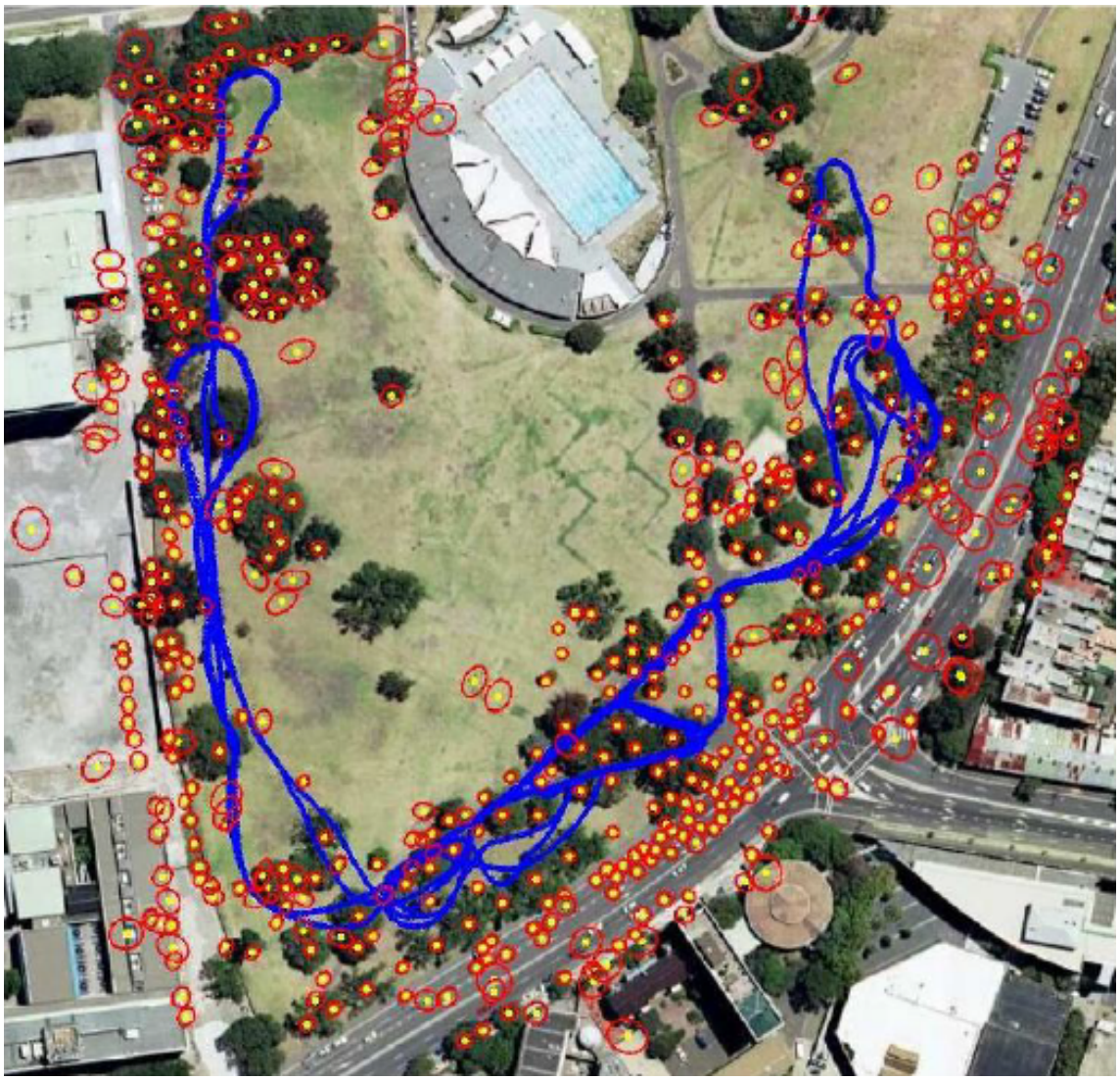

Figura 5.4: Exemplo de *feature-based* SLAM: as elipses vermelhas representam *landmarks* estimadas pelo veículo, à medida que se move ao longo da trajetória em azul. Fonte: extraído de [176].

loop que se realiza a combinação entre a informação *a priori* e a informação dos sensores.

(update)

for $j = 1, \ldots, J_{k|k-1}$

$w_k^{(j)} = (1 - p_{D,k}) w_{k|k-1}^{(j)}$,

$m_k^{(j)} = m_{k|k-1}^{(j)}, \quad P_k^{(j)} = P_{k|k-1}^{(j)}$.

end

$\ell := 0$.

for each $z \in Z_k$

$\ell := \ell + 1$.

for $j = 1, \ldots, J_{k|k-1}$

$w_k^{(\ell J_{k|k-1}+j)} = p_{D,k}\, w_{k|k-1}^{(j)} \mathcal{N}(z; \eta_{k|k-1}^{(j)}, S_k^{(j)})$.

$m_k^{(\ell J_{k|k-1}+j)} = m_{k|k-1}^{(j)} + K_k^{(j)}(z - \eta_{k|k-1}^{(j)})$,

$P_k^{(\ell J_{k|k-1}+j)} = P_{k|k}^{(j)}$,

end

$w_k^{(\ell J_{k|k-1}+j)} := \dfrac{w_k^{(\ell J_{k|k-1}+j)}}{\kappa_k(z) + \sum_{i=1}^{J_{k|k-1}} w_k^{(\ell J_{k|k-1}+i)}}$, for $j = 1, \ldots, J_{k|k-1}$.

end

$J_k = \ell J_{k|k-1} + J_{k|k-1}$.

output $\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\}_{i=1}^{J_k}$.

Figura 5.5: Parte do código paralelizado neste trabalho.
Fonte: algoritmo apresentado em [2].

Nesta seção, ilustra-se o potencial de paralelização do filtro a partir de três funções PHD: uma escalar, uma 2D e, por fim, uma 3D.

As simulações exploram três casos:

1. número de observações $|z_k|$ fixo, aumentado-se o número de componentes da intensidade $J_{k|k-1}$ - emula um aumento no número de alvos, provavelmente falsos alarmes;

2. aumento do número de componentes da intensidade $J_{k|k-1}$ e igual aumento do número de observações - emula um situação de elevado número de objetos no campo de visão do sensor e uma alta probabilidade de detecção;

3. aumento do número de observações $|z_k|$ com um número de componentes $J_{k|k-1}$ constante - emula uma situação do aumento abrupto de objetos, os quais poderiam ser causados por oclusões ou uma mudança de cenário (entrada de alvos no campo de visão de uma radar, por exemplo).

Nas simulações realizadas, são utilizados 4 *workers* para o processamento paralelo. Além disso, os resultados apresentados são as médias de 10 amostras, dessa forma, sendo possível uma melhor avaliação dos resultados e a minimização dos efeitos de não determinismo no processamento realizado pelos *workers*.

Paralelização no filtro GM PHD - caso escalar

Nesta parte do estudo, confrontam-se os resultados obtidos pelos processamentos sequencial e paralelo para uma função intensidade escalar, tal qual a apresentada nos Exemplos 2.4.1 e 2.4.2.

A Figura 5.6 apresenta o tempo de processamento considerando um número de observações fixo $|z_k| = 5$, aumentado-se o número de componentes da intensidade obtida na predição. Já na Figura 5.7, apresenta-se o mesmo resultado, com destaque para o intervalo inicial de $5 \leq J_{k|k-1} \leq 6 \times 10^5$.

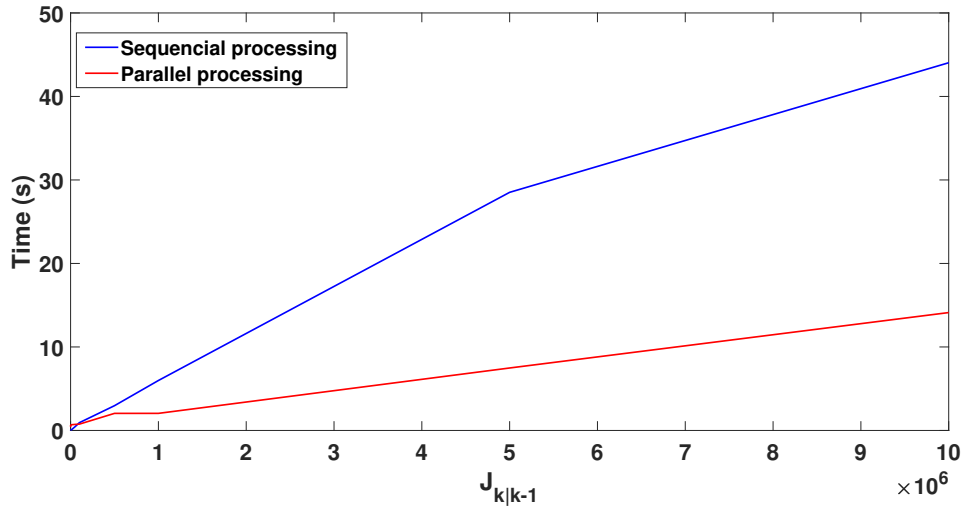

Figura 5.6: Comparação entre o tempo de processamento sequencial x paralelo.

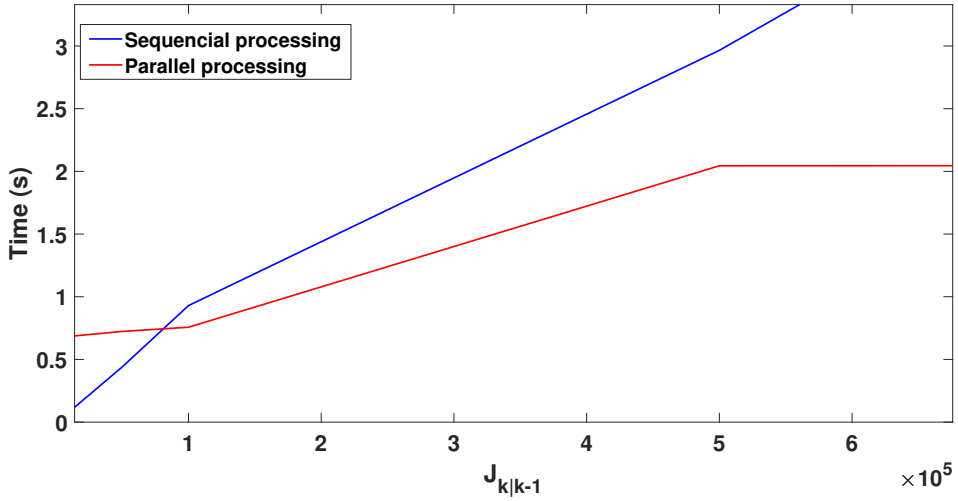

Figura 5.7: Comparação entre o tempo de processamento sequencial x paralelo - destaque para o intervalo de $5 \leq J_{k|k-1} \leq 6 \times 10^5$.

O segundo caso simulado é obtido aumentado-se o número de componentes da intensidade obtida na predição e aumentando-se de igual maneira o número de observações. Portanto, nesse caso é emulado um aumento do número de objetos com uma probabilidade de detecção próxima da unidade. Os resultados obtidos podem ser observados nas Figuras 5.8 e 5.9.

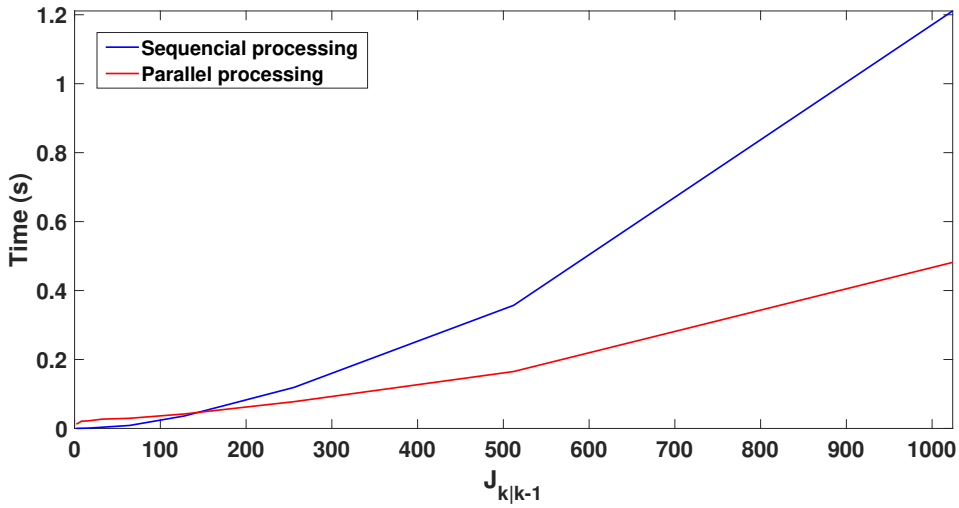

Figura 5.8: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações.

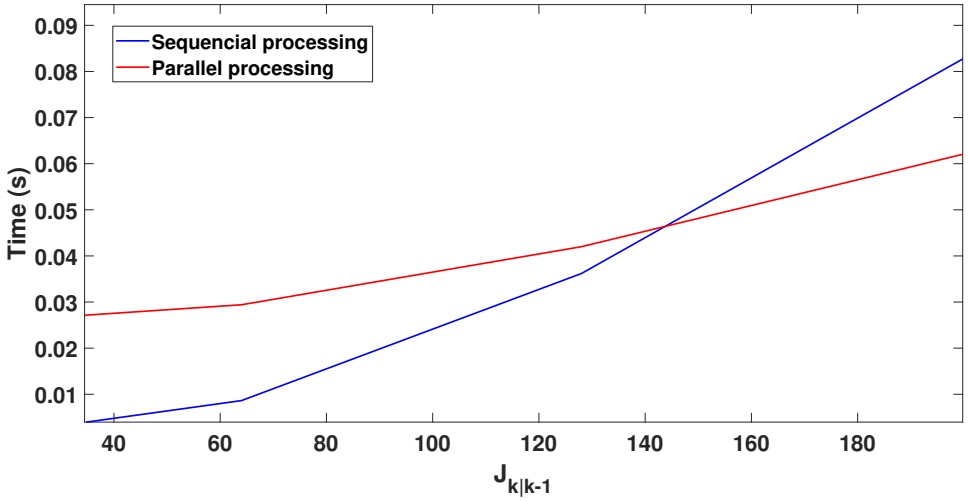

Figura 5.9: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações - destaque para o intervalo de $40 \leq J_{k|k-1} \leq 200$.

Como terceiro caso, simula-se o aumento do número de observações com um número fixo $J_{k|k-1} = 50$. Esse cenário emula uma medição com elevado nível de ruído, como o encontrado em sistemas submarinos com detecções por câmera, por exemplo, no caso de um AUV [39]. Os resultados obtidos são apresentados nas Figuras 5.10 e 5.11.

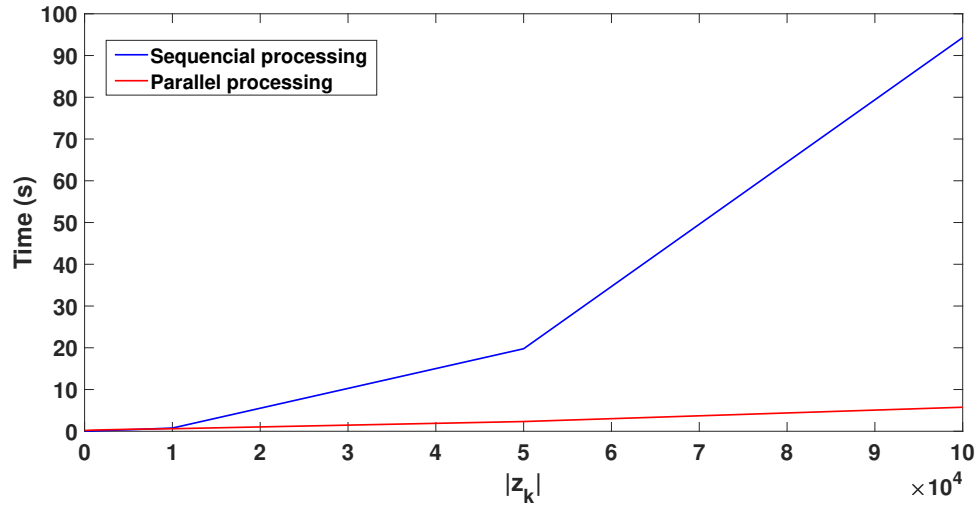

Figura 5.10: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $J_{k|k-1} = 10$.

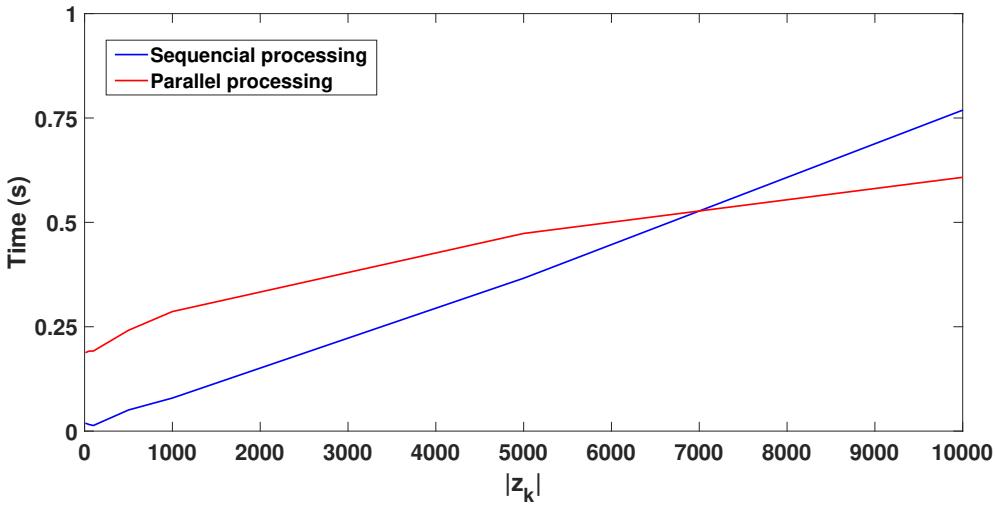

Figura 5.11: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $J_{k|k-1} = 10$ - destaque para o intervalo de $0 \leq |z_k| \leq 10000$.

Paralelização no filtro GM PHD - caso 2D

Nesta parte do estudo, confrontam-se os resultados obtidos pelos processamentos sequencial e paralelo para uma função intensidade definida para um objeto de estado $\mathbf{x} \in \mathbb{R}^2$, sendo a posição em um plano. Esse caso assemelha-se ao discutido no Capítulo 3.

A dinâmica de cada objeto é definida pelo modelo linear

$$\mathbf{x}_k = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + 0.09 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{w}_v, \tag{5.5}$$

e segundo o modelo de observação

$$\mathbf{z}_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + 0.04 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{w}_\epsilon, \tag{5.6}$$

onde os ruídos $\mathbf{w}_v$ e $\mathbf{w}_\epsilon \in \mathbb{R}^{2\times 1}$.

A Figura 5.12 apresenta o tempo de processamento considerando um número de observações fixo $|z_k| = 50$, aumentado-se o número de componentes da intensidade obtida na predição. Já na Figura 5.13, apresenta-se o mesmo resultado, com destaque para o intervalo inicial de $2 \leq J_{k|k-1} \leq 60$.

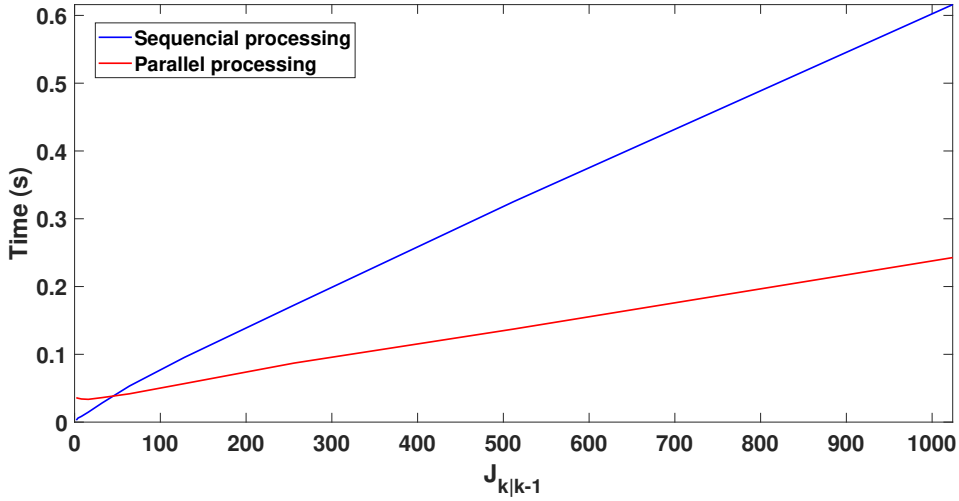

Figura 5.12: Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D.

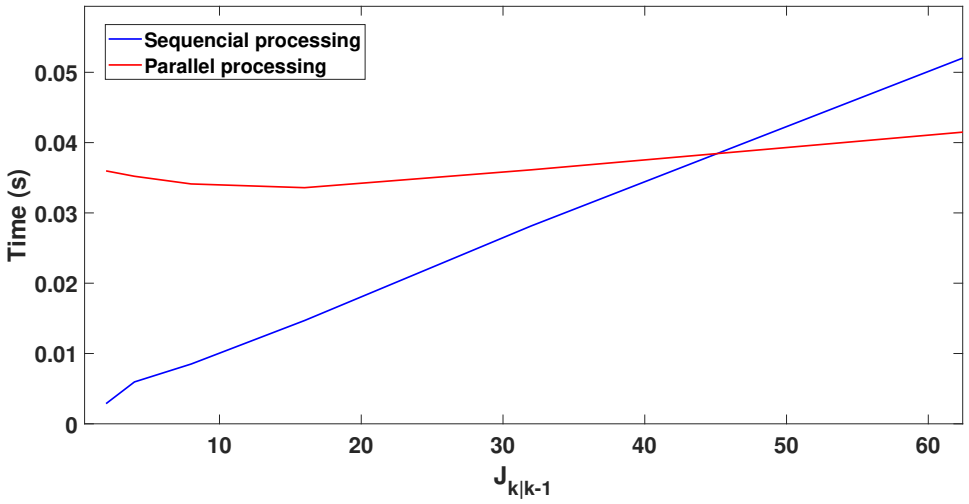

Figura 5.13: Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 60$.

O segundo caso simulado é obtido aumentado-se o número de componentes da intensidade obtida na predição e aumentando-se de igual maneira o número de observações. Portanto, nesse caso é emulado um aumento do número de objetos com uma probabilidade de detecção próxima da unidade. Os resultados obtidos podem ser observados nas Figuras 5.14 e 5.15.

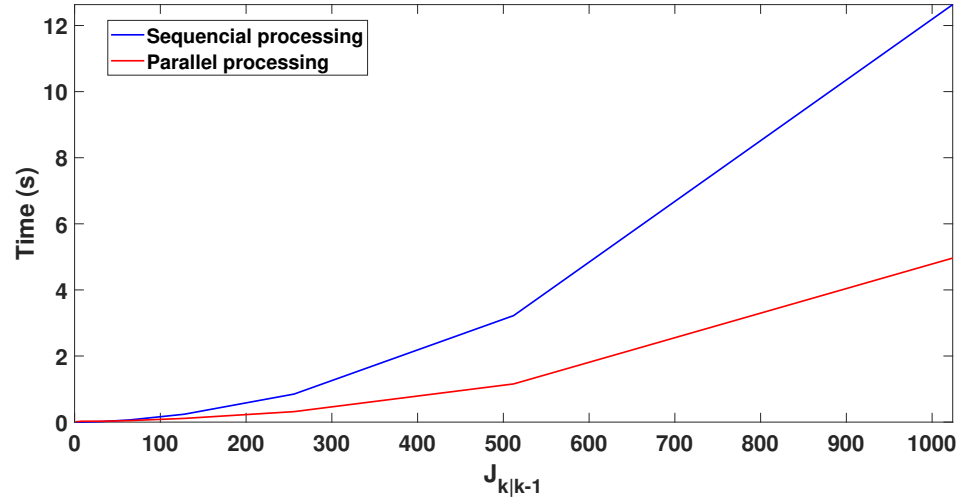

Figura 5.14: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações.

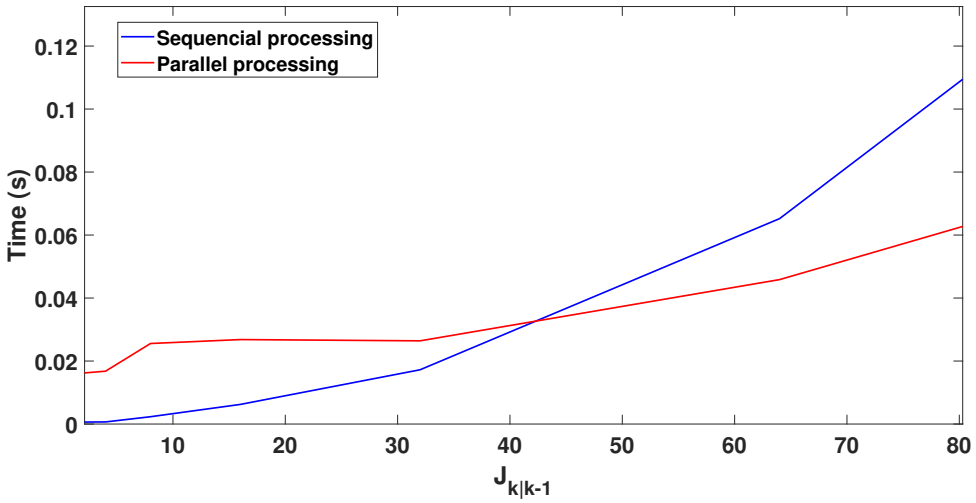

Figura 5.15: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 80$.

Como terceiro caso, simula-se o aumento do número de observações com um número fixo $J_{k|k-1} = 10$. Os resultados obtidos são apresentados nas Figuras 5.16 e 5.17.

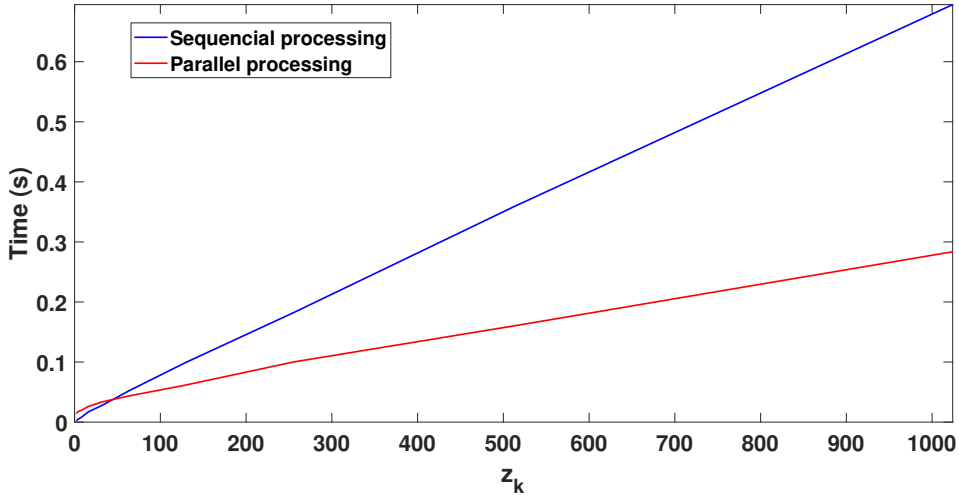

Figura 5.16: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $|J_{k|k-1}| = 50$.

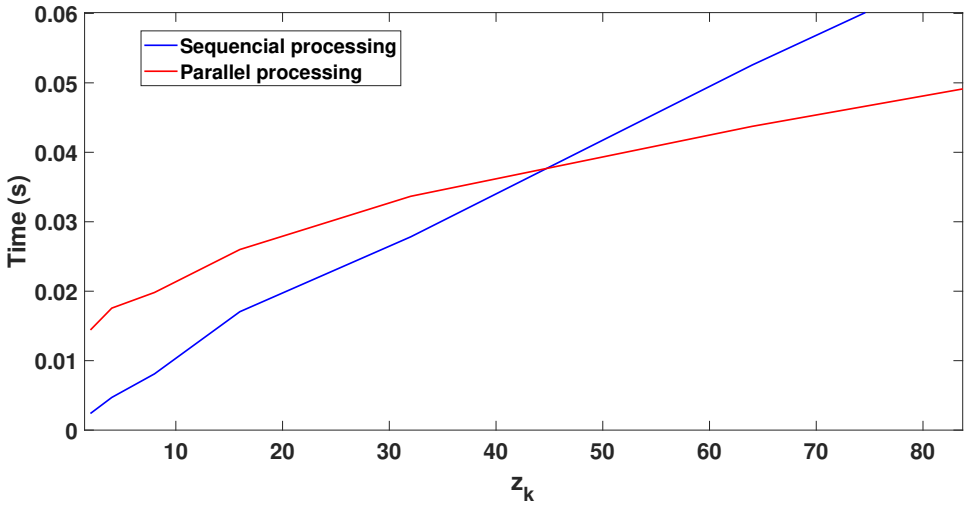

Figura 5.17: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $|J_{k|k-1}| = 50$ - destaque para o intervalo de $2 \leq z_k \leq 80$.

Paralelização no filtro GM PHD - caso 4D

Nesta parte do estudo, confrontam-se os resultados obtidos pelos processamentos sequencial e paralelo para uma função intensidade definida para um objeto de estado $\mathbf{x} \in \mathbb{R}^2$, sendo a posição em um plano. Esse caso assemelha-se ao discutido no Capítulo 4.

A dinâmica de cada objeto é definida pelo modelo linear

$$\mathbf{x}_k = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + \begin{pmatrix} 1/2 & 0 \\ 1 & 0 \\ 0 & 1/2 \\ 0 & 1 \end{pmatrix} \mathbf{w}_v, \tag{5.7}$$

e segundo o modelo de observação

$$\mathbf{z}_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{x}_{k-1} + 0.04 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \mathbf{w}_\epsilon, \tag{5.8}$$

onde o ruído $\mathbf{w}_v \in \mathbb{R}^{4\times1}$ e $\mathbf{w}_\epsilon \in \mathbb{R}^{2\times1}$.

A Figura 5.18 apresenta o tempo de processamento considerando um número de observações fixo $|z_k| = 50$, aumentado-se o número de componentes da intensidade obtida na predição. Já na Figura 5.19, apresenta-se o mesmo resultado, com destaque para o intervalo inicial de $2 \leq J_{k|k-1} \leq 70$.

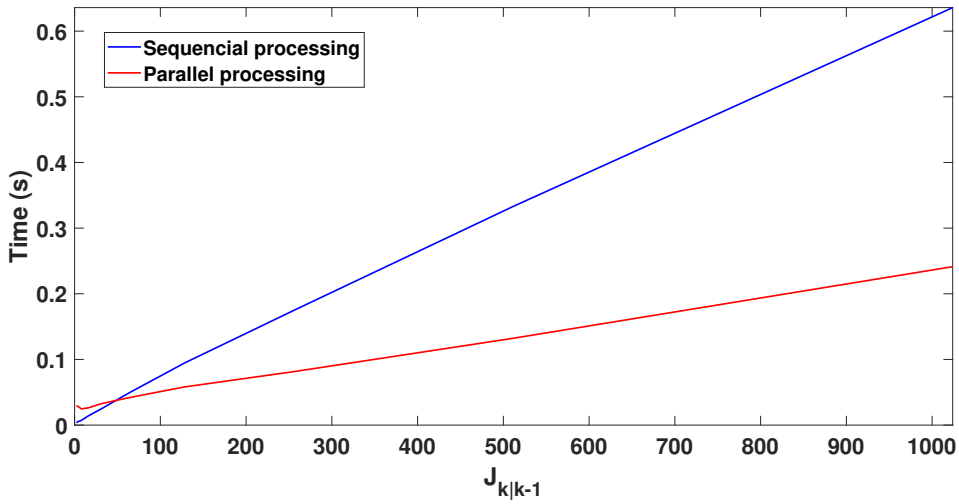

Figura 5.18: Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D.

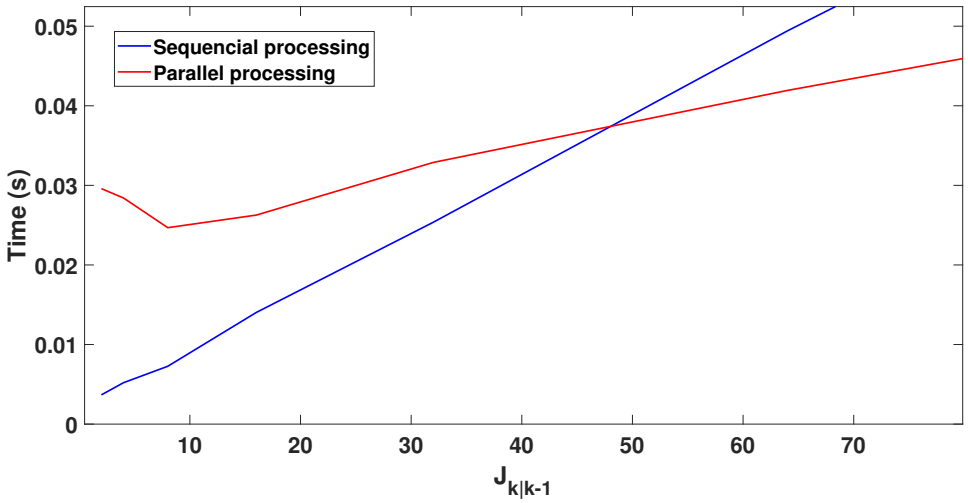

Figura 5.19: Comparação entre o tempo de processamento sequencial x paralelo com $|z_k| = 50$ - caso 2D - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 70$.

O segundo caso simulado é obtido aumentado-se o número de componentes da intensidade obtida na predição e aumentando-se de igual maneira o número de observações. Portanto, nesse caso é emulado um aumento do número de objetos com uma probabilidade de detecção próxima da unidade. Os resultados obtidos podem ser observados nas Figuras 5.20 e 5.21.

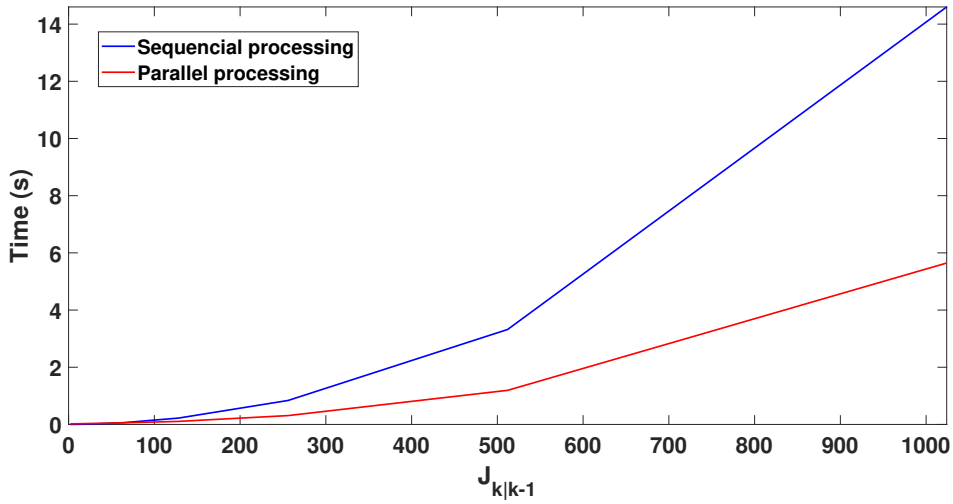

Figura 5.20: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações.

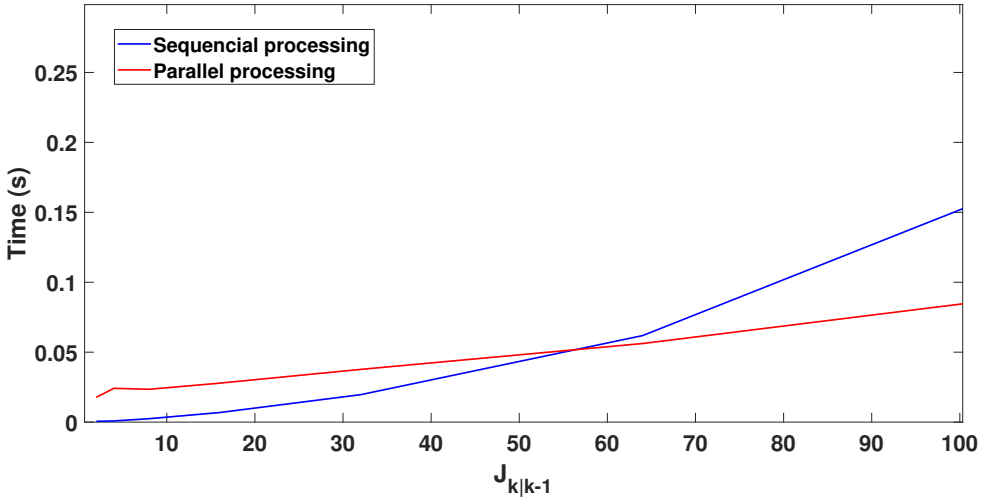

Figura 5.21: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de objetos e de observações - destaque para o intervalo de $2 \leq J_{k|k-1} \leq 100$.

Como terceiro caso, simula-se o aumento do número de observações com um número fixo $J_{k|k-1} = 10$. Os resultados obtidos são apresentados nas Figuras 5.22 e 5.23.

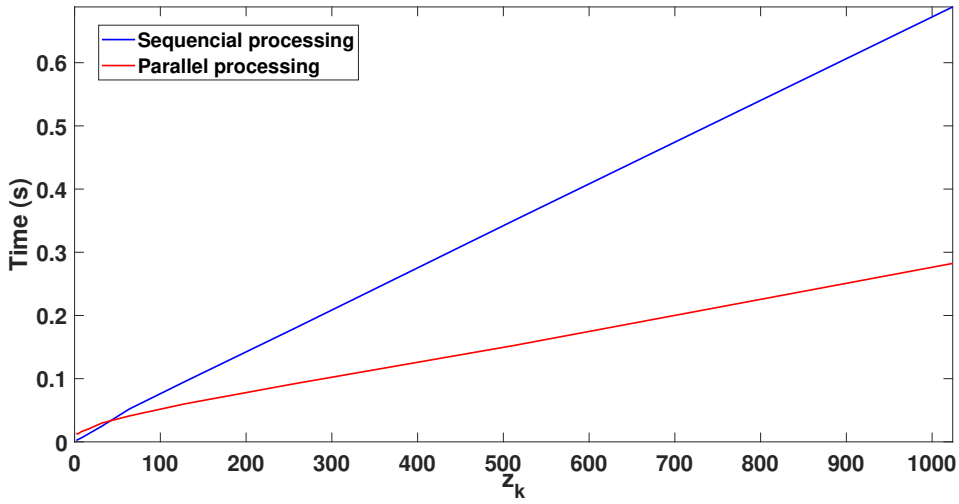

Figura 5.22: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $J_{k|k-1} = 50$.

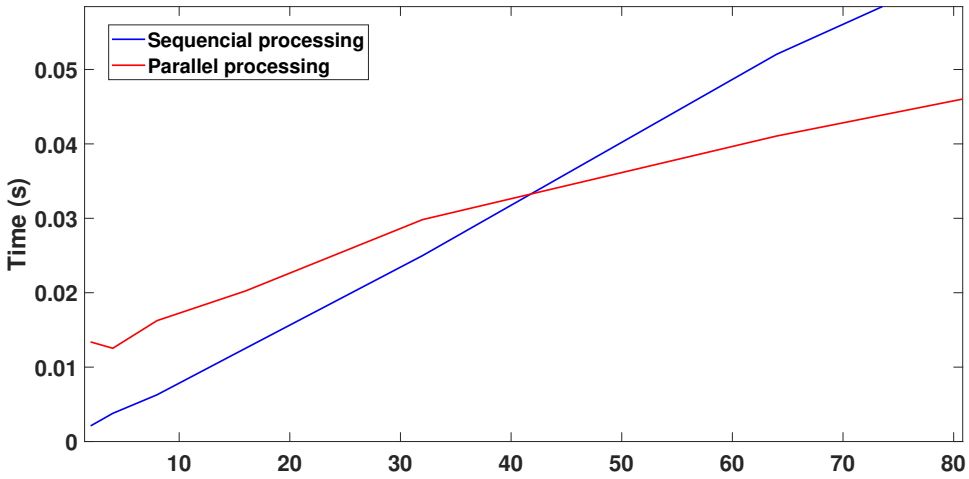

Figura 5.23: Comparação entre o tempo de processamento sequencial x paralelo com aumento do número de observações e $|J_{k|k-1}| = 50$ - destaque para o intervalo de $2 \leq z_k \leq 80$.

Análise dos resultados

A partir da observação dos gráficos apresentados para o caso escalar (Figuras 5.6 e 5.7, é possível constatar que o tempo de processamento cresce de maneira aproximadamente linear para o caso do processamento sequencial. Ao contrário do caso sequencial, no caso de processamento paralelo há uma fase que o tempo de processamento é praticamente constante. Essa constatação indica que, independente do aumento de $J_{k|k-1}$ ou $|z_k|$, o processamento permanece o mesmo. Além disso, pode-se concluir que essa diferença significativamente maior, inicialmente, no tempo de processamento paralelo é causada pelo *overhead* de comunicação com a aplicação cliente e os *pools* de processamento. Além disso, esse custo de paralelização só volta a aumentar a partir de uma determinada quantidade de dados, ou seja, há um custo mínimo para a paralelização. É importante destacar que a utilização de outras ferramentas de paralelização podem reduzir esse custo fixo mínimo quando comparado à ferramenta utilizada nas simulações.

Nos gráficos ampliados, também é possível visualizar aproximadamente o ponto no qual se torna vantajoso paralelizar o filtro. Para o caso exemplificado na Figura 5.9, a partir de 150 objetos no cenário de *tracking* ou de mapeamento, a melhora de desempenho na paralelização já supera o custo de comunicação envolvido. Já para o cenário apresentado na Figura 5.11, a partir de uma relação de objetos por falsos alarmes de aproximadamente 700, a paralelização também se tornaria vantajosa. Esses limites poderiam ser utilizados para lógicas de seleção do tipo "IF" para a utilização de um modo sequencial $\times$ paralelo, dessa forma, utilizando-se o

caso paralelo somente quando vantajoso.

É importante frisar que as simulações realizadas para o caso escalar possuem custo computacional reduzido quando comparadas com problemas de maior dimensão, dadas as operações matriciais envolvidas, principalmente, com o processo de atualização de Kalman. Portanto, comparando-se ao caso sequencial, melhores desempenhos ainda seriam possíveis em aplicações de maior complexidade.

De fato, a afirmação acima é atestada pelas simulações realizadas tanto para o problema em duas ou quatro dimensões. A Tabela 5.1 apresenta uma comparação entre cada caso simulado, a depender do tipo de PHD, quando a paralelização se torna vantajosa em relação à execução sequencial do algoritmo.

Tabela 5.1: Comparação entre os casos simulados - ponto em que a paralelização se torna vantajosa.

Caso	**Escalar**	**2D**	**4D**
1	$J_{k\,\vert\,k-1} \approx 1 \times 10^5$	$J_{k\,\vert\,k-1} \approx 45$	$J_{k\,\vert\,k-1} \approx 48$
2	$J_{k\,\vert\,k-1} \approx 140$	$J_{k\,\vert\,k-1} \approx 42$	$J_{k\,\vert\,k-1} \approx 56$
3	$z_k \approx 7000$	$z_k \approx 45$	$z_k \approx 42$

Adicionalmente, a partir da informação na Tabela 5.1 para os casos 2D e 4D, é possível observar que os valores em que à paralelização torna-se vantajosa são da mesma ordem de grandeza. Portanto, para as simulações realizadas, aumentando-se a dimensão do problema, não houveram ganhos significativos de velocidade de processamento da mesma forma que houve no aumento de dimensão do caso escalar para o 2D.

5.4 Considerações finais

Neste Capítulo, foi apresentado o modelo computacional baseado em reações chamado Gamma. Tal técnica vem sendo estudada e apresenta diversas potenciais aplicações em áreas como IoT, *fluid computing*, computação distribuída, computação aproximativa, redes de sensores e sistemas de *tracking*. Ainda nesse contexto, a aplicação desse modelo computacional foi exemplificada com uma proposta de algoritmo em Gamma para o problema de extração de estados, utilizado em conjunto com o filtro GM PHD.

Por fim, foram realizadas simulações para demonstrar o potencial do uso de computação paralela em conjunto com o filtro GM PHD. Nessas simulações, apesar do *overhead* de comunicação entre a aplicação *overhead* e os *pools*, responsáveis

por executar o paralelismo, ainda há um ganho de desempenho à medida que a complexidade do problema aumenta. Esse aumento de dimensão do problema é o esperado numa rede complexa de sensores distribuídos ou de IoTs.

Capítulo 6

Considerações Finais

6.1 Conclusões

Este trabalho aborda a aplicação de técnicas baseadas em *Random Finite Sets* nos problemas de *multi-object tracking* e de mapeamento em robótica, o qual é parte do problema de SLAM.

Com base na pesquisa bibliográfica realizada, conclui-se que o uso de RFS representa uma importante evolução em relação às metodologias vetoriais utilizadas para rastreamento, mapeamento e SLAM. As técnicas baseadas em conjuntos apresentam como principais vantagens a possibilidade de tratamento de erros de forma matematicamente rigorosa e a eliminação da etapa desacoplada de associação de dados entre medições e objetos já existentes a *priori*.

Contudo, mesmo sendo o método mais simples e rigoroso matematicamente, quando comparado ao vetorial, é inviável a implementação direta da recursão de Bayes em RFS devido à intratabilidade computacional das equações com integrais múltiplas no espaço dos conjuntos em que resulta. Portanto, é necessário recorrer a alguma aproximação que torne o problema aplicável computacionalmente.

Com base no estudo, foi possível concluir que o filtro PHD é amplamente utilizado como solução da intratabilidade dos filtros *bayesianos* em RFS. Tal solução é proposta como a mais simples de implementação para a recursão de Bayes, sem perder o controle sobre suas propriedades (estimação dos erros), ainda que não exista uma forma fechada para o caso geral.

Verificou-se que, assim como em uma grande parte das soluções para mapeamento e rastreamento convencionais, é praticamente impossível chegar-se a uma solução geral em que não se considere um determinado contexto e que se aplique em todos os casos.

Neste trabalho, foram apresentadas as motivações para o uso de RFS e alguns exemplos das ferramentas matemáticas utilizadas nessa abordagem com o máximo

de generalidade possível, para que o detalhamento não dificultasse a compreensão da técnica.

Foi possível concluir que o uso de RFS para robótica móvel é algo relativamente recente, uma vez que há pouca informação na literatura e pela grande quantidade de possíveis aplicações em que esse método seria mais eficiente e ainda não está implementado. Isso evidencia que o problema em tempo real permanece em aberto.

Este trabalho também apresenta um Sistema de Vigilância Marítima para o Litoral brasileiro, o qual possui uma rede complexa de sensores distribuídos e estações de processamento. Nesse contexto, apresenta-se um esquema de fusão para dados de rastreamento, de um número desconhecido de alvos, a partir de múltiplos sensores em uma área de vigilância em larga escala. Nesse cenário, o esquema de fusão combinado com o filtro PHD apresenta resultados promissores, por meio de simulações, mesmo para alvos com diferentes tamanhos, movimentos e velocidades. O esquema de fusão de dados proposto no Capítulo 4 melhora o desempenho geral do sistema de rastreamento e supera a oclusão dos alvos. Este fato evidencia ainda, para a aplicação, a necessidade de uma rede de sensores distribuídos, dada a missão do sistema e a extensão da área de interesse. Adicionalmente, é possível concluir que o aumento da área coberta utilizando sensores de menor capacidade também contribui para o aspecto econômico deste empreendimento modular.

Também foi possível observar que a combinação de métodos de computação paralela com o filtro PHD pode trazer ganhos às aplicações em tempo real. Tal conclusão foi obtida por meio da simulação e da comparação dos resultados obtidos nos processamentos sequencial e paralelo.

6.2 Trabalhos futuros

Para trabalhos futuros na área de vigilância, um dos focos seria investigar diferentes filtros baseados em RFS, bem como as capacidades dos algoritmos *extended* e diferentes tipos de sensores, como sonares passivos estacionários ou radares de bordo.

Além disso, os modelos computacionais utilizados para paralelismo e multiprocessamento devem ser investigados. Em particular, aqueles baseados em modelos Gamma, uma vez que as aplicações de fusão de dados SCUA já utilizam esta abordagem. Esses trabalhos futuros poderão atestar, corroborar os resultados empíricos obtidos nas simulações, e contribuir para a utilização desses algoritmos em um novo paradigma computacional.

Ainda em relação ao processamento paralelo, poder-se-ia investigar a paralelização de outros filtros baseados em RFS, incluindo-se versões *extended*, e paralelização de algoritmos de detecção e clusterização. Esses últimos, como uma alternativa ao

merge tradicional, os quais podem ser mais robustos em aplicações com um significativo número de dados, tais como o sistema de *tracking* em tempo real para veículos autônomos, numa rede de sensores ou robôs baseados em Internet das Coisas.

Por fim, sugere-se a investigação de técnicas de compressão e supressão de dados como forma de melhorar aplicações de tempo real em redes de sensores, levando-se em consideração a relação entre uso da banda de comunicação pelos nós e a qualidade da estimação efetuada cooperativamente.

Referências Bibliográficas

[1] MULLANE, J., VO, B.-N., ADAMS, M., et al. *Random Finite Sets for Robot Mapping and SLAM: New Concepts in Autonomous Robotic Map Representations*, v. 72, *Springer Tracts in Advanced Robotics*. Berlin Heidelberg, Springer-Verlag, 2011.

[2] VO, B., MA, W. "The Gaussian Mixture Probability Hypothesis Density Filter", *IEEE Transactions Signal Processing*, v. 54, n. 11, pp. 4091–4104, September 2006. doi: 10.1109/TSP.2006.881190.

[3] BIMBRAW, K. "Autonomous Cars: Past, Present and Future". In: *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2015)*, pp. pag. 191–198, 2015.

[4] CHENG, H. *Autonomous Intelligent Vehicles: Theory, Algorithms, and Implementation*. London, UK, Springer, 2011.

[5] KACHROO, P., WADOO, S. *Autonomous Underwater Vehicles: Modeling, Control Design and Simulation*. 4th ed. Portland, OR, CRC, jan. 2010. ISBN: 9781315218038. doi: 10.1201/b10463.

[6] HIRSCHBERG, M. "To Boldly Go Where No Unmanned Aircraft Has Gone Before: A Half-Century of DARPA's Contributions to Unmanned Aircraft". In: *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, jan 2010. doi: 10.2514/6.2010-158.

[7] TADJDEH, Y. "DARPA Pursuing Technologies to Help Troops ID Enemies", *National Defense*, v. 100, n. 750, pp. 42–45, 2016. Disponível em: <https://www.jstor.org/stable/27021240>.

[8] Buehler, M., Iagnemma, K., Singh, S. (Eds.). *The DARPA Urban Challenge*. GER, Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-03991-1.

[9] XIE, M., TRASSOUDAINE, L., ALIZON, J., et al. "Active and intelligent sensing of road obstacles: Application to the European Eureka-

PROMETHEUS project". In: *4th International Conference on Computer Vision*. IEEE Computer Society Press, 1993. doi: 10.1109/iccv.1993.378154.

[10] THORPE, C., HERBERT, M., KANADE, T., et al. "Toward autonomous driving: the CMU Navlab. I. Perception", *IEEE Expert*, v. 6, n. 4, pp. 31–42, aug 1991. doi: 10.1109/64.85919.

[11] POMERLEAU, D. A. "Knowledge-Based Training of Artificial Neural Networks for Autonomous Robot Driving". In: *Robot Learning*, Springer US, pp. 19–43, USA, 1993. doi: 10.1007/978-1-4615-3184-5_2.

[12] CURCIO, J., LEONARD, J., VAGANAY, J., et al. "Experiments in Moving Baseline Navigation using Autonomous Surface Craft". In: *Proceedings of OCEANS 2005 MTS/IEEE*. IEEE, 2005. doi: 10.1109/oceans.2005.1639839.

[13] BAHR, A., LEONARD, J. J., FALLON, M. F. "Cooperative Localization for Autonomous Underwater Vehicles", *The International Journal of Robotics Research*, v. 28, n. 6, pp. 714–728, may 2009. doi: 10.1177/0278364908100561.

[14] SCHMIDT, H. *GOATS-2000 Multi-AUV Cooperative Behavior Multi-scaleEnvironmentsal Assesment*. Relatório técnico, Department of Ocean Engineering, Massachusetts Institute of Technology, Cambridge, MA, 02139, 2001.

[15] HEYNINGEN, E. V. "Automated for the people: a new automated people mover has been developed in the netherlands that will use unmanned vehicle technology and a navigation system to ferry passengers around terminals", *Traffic technology international. Annual review, SURFACE SYSTEMS INC*, pp. 260–263, 1998.

[16] SHOEMAKER, C., BORNSTEIN, J. "The Demo III UGV program: a testbed for autonomous navigation research". In: *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intelligent Systems and Semiotics (ISAS) (Cat. No.98CH36262)*. IEEE, 1998. doi: 10.1109/isic.1998.713784.

[17] MULLANE, J., VO, B., ADAMS, M., et al. "Mobile Robotics in a Random Finite Set Framework". In: *Lecture Notes in Computer Science*,

Springer Berlin Heidelberg, pp. 519–528, Berlin, 2011. doi: 10.1007/978-3-642-21524-7_64.

[18] STACHNISS, C., LEONARD, J. J., THRUN, S. "Simultaneous Localization and Mapping". In: *Springer Handbook of Robotics*, cap. 46, pp. 1153–1176, Cham, Springer International Publishing, 2016.

[19] SICILIANO, B., KHATIB, O. *Springer Handbook of Robotics*. 2nd ed. Berlin, Heidelberg, Springer, 2016. Chapter 46.

[20] AGUNBIADE, O., ZUVA, T. "Simultaneous Localization and Mapping in Application to Autonomous Robot". In: *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*. IEEE, dec 2018. doi: 10.1109/iconic.2018.8601094.

[21] ZAFFAR, M., EHSAN, S., STOLKIN, R., et al. "Sensors, SLAM and Long-term Autonomy: A Review", *ArXiv*, 2018.

[22] YAAKOV BAR-SHALOM, X. RONG LI, T. K. *Estimation with Applications to Tracking and Navigation*. USA, John Wiley Sons, 2001.

[23] VO, B., MALLICK, M., BAR-SHALOM, Y., et al. "Multitarget Tracking", *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 1–15, 2015.

[24] BATTISTELLI, G., CHISCI, L., FANTACCI, C., et al. "Consensus CPHD Filter for Distributed Multitarget Tracking", *IEEE Journal of Selected Topics in Signal Processing*, v. 7, n. 3, pp. 508–520, jun 2013. doi: 10.1109/jstsp.2013.2250911.

[25] DURRANT-WHYTE, H., BAILEY, T. "Simultaneous localization and mapping: part I", *IEEE Robotics Automation Magazine*, v. 13, n. 2, pp. 99–110, jun. 2006.

[26] BAILEY, T., .DURRANT-WHYTE, H. "Simultaneous localization and mapping (SLAM): part II", *IEEE Robotics Automation Magazine*, v. 13, n. 3, pp. 108–117, ago. 2006.

[27] CHEN, Z. *Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond*. Relatório técnico, McMaster University, 2003.

[28] TAKETOMI, T., UCHIYAMA, H., IKEDA, S. "Visual SLAM algorithms: a survey from 2010 to 2016", *IPSJ Transactions on Computer Vision and Applications*, v. 9, n. 1, pp. 16, jun. 2017.

[29] CLARK, D., PANTA, K., VO, B.-N. "The GM-PHD Filter Multiple Target Tracker". In: *Proceedings of 9th International Conference on Information Fusion*, 2006. doi: 10.1109/ICIF.2006.301809.

[30] VO, B.-T., VO, B.-N., CANTONI, A. "Bayesian Filtering With Random Finite Set Observations", *IEEE Transactions on Signal Processing*, v. 56, n. 4, pp. 1313–1326, apr 2008.

[31] MULLANE, J., VO, B., ADAMS, M. D., et al. "A random set formulation for Bayesian SLAM". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[32] ERDINC, O., WILLETT, P., BAR-SHALOM, Y. "The Bin-Occupancy Filter and Its Connection to the PHD Filters", *IEEE Transactions on Signal Processing*, 2009. doi: 10.1109/TSP.2009.2025816.

[33] KALYAN, B., LEE, K., WIJESOMA, W. "FISST-SLAM: Finite Set Statistical Approach to Simultaneous Localization and Mapping", *Int. J. Rob. Res.*, v. 29, n. 10, pp. 1251–1262, set. 2010.

[34] MULLANE, J., VO, B.-N., ADAMS, M. D., et al. "A Random-Finite-Set Approach to Bayesian SLAM", *IEEE Transactions on Robotics*, v. 27, n. 2, pp. 268–282, apr 2011. doi: 10.1109/tro.2010.2101370.

[35] ZHANG, F., STAHLE, H., GASCHLER, A., et al. "Single Camera Visual Odometry Based on Random Finite Set Statistics". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[36] ZHANG, F., STAHLE, H., CHEN, C., et al. "A Lane Marking Extraction Approach based on Random Finite Set Statistics". In: *IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[37] ADAMS, M., VO, B., MAHLER, R., et al. "SLAM gets a PHD: new concepts in map estimation", *IEEE Robotics & Automation Magazine*, 2014.

[38] NAGAPPA, S., PALOMERAS, N., LEE, C. S., et al. "Single cluster PHD SLAM: Application to autonomous underwater vehicles using stereo vision". In: *2013 MTS/IEEE OCEANS - Bergen*. IEEE, jun 2013. doi: 10.1109/oceans-bergen.2013.6608107.

[39] LEE, C., CLARK, D., SALVI, J. "SLAM With Dynamic Targets via Single-Cluster PHD Filtering", *IEEE Journal of Selected Topics in Signal Processing*, v. 7, n. 3, pp. 543–552, jun. 2013.

[40] LEE, C., NAGAPPA, S., PALOMERAS, N., et al. "SLAM with SC-PHD Filters: An Underwater Vehicle Application", *IEEE Robotics & Automation Magazine*, jun. 2014. doi: 10.1109/MRA.2014.2310132.

[41] WALTER, M., LEONARD, J. "An experimental investigations of cooperative SLAM". In: *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[42] LEE, H., LEE, S.-H., LEE, T.-S., et al. "A survey of map merging techniques for cooperative-SLAM". In: *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2012.

[43] GARCEA, A., ZHU, J., OPDENBOSCH, V., et al. "Robust Map Alignment for Cooperative Visual SLAM". In: *25th IEEE International Conference on Image Processing (ICIP)*, 2018.

[44] RUAN, J.AND SHIH, S., TSAI, M., FANG, Y., et al. "Cooperative Visual Simultaneous Localization and Mapping by Ordering Keyframes Similarity". In: *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2018.

[45] WANIEK, N., BIEDERMANN, J., CONRADT, J. "Cooperative SLAM on small mobile robots". In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015.

[46] TRUJILLO, J., MUNGUIA, R. ANDGUERRA, E., GRAU, A. "Cooperative Monocular-Based SLAM for Multi-UAV Systems in GPS-Denied Environments", *MDPI: Sensors*, 2018.

[47] DEMIM, F., NEMRA, A., LOUADJ, K., et al. "Cooperative SLAM for multiple UGVs navigation using SVSF filter", *Journal for Control, Measurement, Electronics, Computing and Communications*, v. 58, n. 1, 2017.

[48] PAULL, L., HUANG, G., SETO, M., et al. "Communication-constrained multi-AUV cooperative SLAM". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[49] ARUMUGAM, R., ENTI, V. R., BINGBING, L., et al. "DAvinCi: A cloud computing framework for service robots". In: *2010 IEEE International Conference on Robotics and Automation*, 2010.

[50] HU, G., TAY, W. P., WEN, Y. "Cloud robotics: architecture, challenges and applications", *IEEE Network*, v. 26, n. 3, pp. 21–28, maio 2012.

[51] RIAZUELO, L., CIVERA, J., MONTIEL, J. "C2TAM: A Cloud framework for cooperative tracking and mapping", *Robotics and Autonomous Systems*, v. 62, n. 4, pp. 401–413, 2014. doi: https://doi.org/10.1016/j.robot.2013.11.007.

[52] BENAVIDEZ, P., MUPPIDI, M., RAD, P., et al. "Cloud-based realtime robotic Visual SLAM". In: *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, 2015.

[53] K., L., ĆESIĆ J., I., M., et al. "Cooperative Cloud SLAM on Matrix Lie Groups". In: A., O., A., S., L., M., et al. (Eds.), *ROBOT 2017. Advances in Intelligent Systems and Computing*, v. 693. ROBOT 2017: Third Iberian Robotics Conference, Springer, Cham, 2017.

[54] ALI, S. S., HAMMAD, A., ELDIEN, A. S. T. "FastSLAM 2.0 tracking and mapping as a Cloud Robotics service", *Computers & Electrical Engineering*, v. 69, pp. 412–421, jul. 2018. doi: 10.1016/j.compeleceng.2017.11.012.

[55] ZHANG, P., WANG, H., DING, B., et al. "Cloud-Based Framework for Scalable and Real-Time Multi-Robot SLAM". In: *IEEE International Conference on Web Services (ICWS)*, 2018.

[56] PETRÍK, V. *Multi-Vehicle Random Finite Set SLAM*. Tese de Mestrado, CZECH TECHNICAL UNIVERSITY IN PRAGUE – Faculty of Electrical Engineering – Department of Cybernetics, 2014.

[57] ZHANG, F., BUCKL, C., KNOLL, A. "Multiple Vehicle Cooperative Localization with Spatial Registration Based on a Probability Hypothesis Density Filter", *MDPI: Sensors*, v. 14, n. 1, pp. 995–1009, jan. 2014.

[58] CANO, P., RUIZ-DEL SOLAR, J. "Robust Tracking of Soccer Robots Using Random Finite Sets", *IEEE Intelligent Systems*, v. 32, n. 6, pp. 22–29, November 2017. doi: 0.1109/MIS.2017.4531220.

[59] DAMES, P. M. "Distributed multi-target search and tracking using the PHD filter", *Autonomous Robots*, v. 44, n. 3-4, pp. 673–689, feb 2019. doi: 10.1007/s10514-019-09840-9.

[60] SHIRSAT, A., BERMAN, S. "Decentralized Multi-target Tracking with Multiple Quadrotors using a PHD Filter". In: *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, jan 2021. doi: 10.2514/6.2021-1583.

[61] CARTHEL, C., CORALUPPI, S., GRIGNAN, P. "Multisensor tracking and fusion for maritime surveillance". In: *2007 10th International Conference on Information Fusion*, pp. 1 – 6, August 2007.

[62] MAHLER, R. P. S. "Random-set approach to data fusion". In: Sadjadi, F. A. (Ed.), *Automatic Object Recognition IV*. SPIE, jul 1994. doi: 10.1117/12.181026.

[63] MAHLER, R. ""Statistics 101"for multisensor, multitarget data fusion", *IEEE Aerospace and Electronic Systems Magazine*, v. 19, n. 1, pp. 53–64, jan 2004. doi: 10.1109/maes.2004.1263231.

[64] MAHLER, R. P. S. *Statistical Multisource Multitarget Information Fusion*. Boston, MA, USA, Artech House, Inc., 2007.

[65] GOODMAN, I. R., MAHLER, R. P. S., NGUYEN, H. T. *Mathematics of Data Fusion*. USA, Kluwer Academic Publishers, 1997. doi: 10.1007/978-94-015-8929-1.

[66] MAHLER, R. "Multitarget bayes filtering via first-order multitarget moments", *IEEE Transactions on Aerospace and Electronic Systems*, v. 39, n. 4, pp. 1152–1178, oct 2003. doi: 10.1109/taes.2003.1261119.

[67] VO, B.-N., MA, W.-K. "A closed-form solution for the probability hypothesis density filter". In: *2005 7th International Conference on Information Fusion*. IEEE, 2005. doi: 10.1109/icif.2005.1591948.

[68] CLARK, D., BELL, J. "Convergence results for the particle PHD filter", *IEEE Transactions on Signal Processing*, v. 54, n. 7, pp. 2652–2661, jul 2006. doi: 10.1109/tsp.2006.874845.

[69] CLARK, D., VO, B.-N. "Convergence Analysis of the Gaussian Mixture PHD Filter", *IEEE Transactions on Signal Processing*, v. 55, n. 4, pp. 1204–1212, apr 2007. doi: 10.1109/tsp.2006.888886.

[70] CLARK, D., VO, B.-T., VO, B.-N. "Gaussian Particle Implementations of Probability Hypothesis Density Filters". In: *2007 IEEE Aerospace Conference*. IEEE, 2007. doi: 10.1109/aero.2007.353049.

[71] CLARK, D., BELL, J. "Multi-target state estimation and track continuity for the particle PHD filter", *IEEE Transactions on Aerospace and Electronic Systems*, v. 43, n. 4, pp. 1441–1453, oct 2007. doi: 10.1109/taes.2007.4441750.

[72] LIN, L., BAR-SHALOM, Y., KIRUBARAJAN, T. "Track labeling and PHD filter for multitarget tracking", *IEEE Transactions on Aerospace and Electronic Systems*, v. 42, n. 3, pp. 778–795, jul 2006. doi: 10.1109/taes.2006.248213.

[73] ZAJIC, T., MAHLER, R. P. S. "Particle-systems implementation of the PHD multitarget-tracking filter". In: Kadar, I. (Ed.), *Signal Processing, Sensor Fusion, and Target Recognition XII*. SPIE, aug 2003. doi: 10.1117/12.488533.

[74] CLARK, D., GODSILL, S. "Group Target Tracking with the Gaussian Mixture Probability Hypothesis Density Filter". In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE, 2007. doi: 10.1109/issnip.2007.4496835.

[75] HAO, Y., MENG, F., ZHOU, W., et al. "Gaussian mixture probability hypothesis density filter algorithm for multi-target tracking". In: *2009 IEEE International Conference on Communications Technology and Applications*. IEEE, oct 2009. doi: 10.1109/iccomta.2009.5349103.

[76] BAISA, N. L., WALLACE, A. "Development of a N-type GM-PHD filter for multiple target, multiple type visual tracking", *Journal of Visual Communication and Image Representation*, v. 59, pp. 257–271, feb 2019. doi: 10.1016/j.jvcir.2019.01.026.

[77] MAHLER, R. "PHD filters of higher order in target number", *IEEE Transactions on Aerospace and Electronic Systems*, v. 43, n. 4, pp. 1523–1543, oct 2007. doi: 10.1109/taes.2007.4441756.

[78] NANNURU, S., BLOUIN, S., COATES, M., et al. "Multisensor CPHD filter", *IEEE Transactions on Aerospace and Electronic Systems*, v. 52, n. 4, pp. 1834–1854, 2016. doi: 10.1109/TAES.2016.150265.

[79] VO, B.-T., VO, B.-N., CANTONI, A. "Analytic Implementations of the Cardinalized Probability Hypothesis Density Filter", *Signal Processing, IEEE Transactions on*, v. 55, pp. 3553 – 3567, August 2007.

[80] MAHLER, R. "A brief survey of advances in random-set fusion". In: *2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, oct 2015. doi: 10.1109/iccais.2015.7338726.

[81] MAHLER, R. "Exact Closed-Form Multitarget Bayes Filters", *Sensors*, v. 19, n. 12, pp. 2818, jun 2019. doi: 10.3390/s19122818.

[82] PACE, M. "Comparison of PHD based filters for the tracking of 3D aerial and naval scenarios". In: *2010 IEEE Radar Conference*. IEEE, 2010. doi: 10.1109/radar.2010.5494574.

[83] DO, C.-T., NGUYEN, T. T. D., LIU, W. "Tracking Multiple Marine Ships via Multiple Sensors with Unknown Backgrounds", *Sensors*, v. 19, n. 22, pp. 5025, nov 2019. doi: 10.3390/s19225025.

[84] LANEUVILLE, D., HOUSSINEAU, J. "Passive multi target tracking with GM-PHD filter". In: *2010 13th International Conference on Information Fusion*. IEEE, jul 2010. doi: 10.1109/icif.2010.5711954.

[85] PAILHAS, Y., HOUSSINEAU, J., PETILLOT, Y. R., et al. "Tracking with MIMO sonar systems: applications to harbour surveillance", *IET Radar, Sonar & Navigation*, v. 11, n. 4, pp. 629–639, apr 2017. doi: 10.1049/iet-rsn.2016.0080.

[86] GULMEZOGLU, B., GULDOGAN, M. B., GEZICI, S. "Multiperson Tracking With a Network of Ultrawideband Radar Sensors Based on Gaussian Mixture PHD Filters", *IEEE Sensors Journal*, v. 15, n. 4, pp. 2227–2237, apr 2015. doi: 10.1109/jsen.2014.2372312.

[87] DIAS, S. S., BRUNO, M. G. S. "Cooperative Target Tracking Using Decentralized Particle Filtering and RSS Sensors", *IEEE Transactions on Signal Processing*, v. 61, n. 14, pp. 3632–3646, jul 2013. doi: 10.1109/tsp.2013.2262276.

[88] BRANCALION, J. F. B., DIAS, S. S. "An IoT Inspired Distributed Data Fusion Architecture for Coastal Surveillance Applications". In: *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE, jul 2020. doi: 10.23919/fusion45008.2020.9190591.

[89] LIU, R. W., GUO, Y., NIE, J., et al. "Intelligent Edge-Enabled Efficient Multi-Source Data Fusion for Autonomous Surface Vehicles in Maritime Internet of Things", *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2022. doi: 10.1109/tgcn.2022.3158004.

[90] KUZMIN, M. "Review, Classification and Comparison of the Existing SLAM Methods for Groups of Robots". In: *PROCEEDING OF THE 22ND CONFERENCE OF FRUCT ASSOCIATION*, pp. 115 –120, Jyvaskyla, Finland, maio 2018.

[91] SONNENBURG, C. R., WOOLSEY, C. A. "Modeling, Identification and Control of an Unmanned Surface Vehicle", *Journal of Field Robotics*, v. 30, n. 3, pp. 371–398, 2013. doi: 10.1002/rob.21452.

[92] LIU, Z., ZHANG, Y., YU, X., et al. "Unmanned surface vehicles: An overview of developments and challenges", *Annual Reviews in Control*, v. 41, n. 71–93, 2016.

[93] ELKINS, L., SELLERS, D., MONACH, W. "The Autonomous Maritime Navigation (AMN) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles", *Journal of field robotics*, 2010.

[94] LIMA, K. M. D., COSTA, R. R., PEREIRA-DIAS, D. "Conjuntos Aleatórios Finitos no Problema de Localização e Mapeamento de Robôs Móveis". In: *Anais do XXII Congresso Brasileiro de Automática*, Vitória, ES, Brasil, 2018.

[95] RUSSEL, S. J., NORVIG, P. *Artificial Intelligence: a modern approach*. Lake Street Upper Saddle River, NJ, United States, Prentice Hall, 2010.

[96] THRUN, S., BURGARD, W., FOX, D. *Probabilistic Robotics*. Intelligente Robotics and Autonomous Agent. Cambridge, Mass., MIT Press, August 2005.

[97] RISTIC, B. *Particle Filters for Random Set Models*. USA, Springer New York, 2013. doi: 10.1007/978-1-4614-6316-0.

[98] DAMES, P. M. *Multi-Robot Active Information Gathering Using Random Finite*. Tese de Doutorado, University of Pennsylvania, 2015.

[99] RISTIC, B., BEARD, M., FANTACCI, C. "An Overview of Particle Methods for Random Finite Set Models", *Information Fusion*, v. 31, n. C, pp. 110–126, September 2016.

[100] MAHLER, R. P. S. *Advances in statistical multisource-multitarget information fusion*. Boston, USA, Artech House, Inc., 2014.

[101] FRÜHWIRTH-SCHNATTER, S. *Finite mixture and Markov switching models*. Berlin, Germany, Springer, 2006.

[102] CHEN, R., LIU, J. S. "Mixture Kalman filters", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, v. 62, n. 3, pp. 493–508, aug 2000. doi: 10.1111/1467-9868.00246.

[103] DE LIMA, K. M., COSTA, R. R. "Cooperative-PHD Tracking Based on Distributed Sensors for Naval Surveillance Area", *Sensors*, v. 22, n. 3, pp. 729, jan 2022. doi: 10.3390/s22030729.

[104] SIDENBLADH, H. "Multi-target particle filtering for the probability hypothesis density". In: *Sixth International Conference of Information Fusion, 2003. Proceedings of the*. IEEE, 2003. doi: 10.1109/icif.2003.177321.

[105] MAHALANOBIS, P. "On the Generalized Distance in Statistics", *Proceedings of the National Institute of Science of India*, v. 2, n. 1, pp. 49–55, 1936.

[106] SCHUHMACHER, D., VO, B.-T., VO, B.-N. "A Consistent Metric for Performance Evaluation of Multi-Object Filters", *IEEE Transactions on Signal Processing*, v. 56, n. 8, pp. 3447–3457, aug 2008. doi: 10.1109/tsp.2008.920469.

[107] RISTIC, B., VO, B.-N., CLARK, D., et al. "A Metric for Performance Evaluation of Multi-Target Tracking Algorithms", *IEEE Transactions on Signal Processing*, v. 59, n. 7, pp. 3452–3457, jul 2011. doi: 10.1109/tsp.2011.2140111.

[108] BRESSON, G., ALSAYED, Z., YU, L., et al. "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving", *IEEE Transactions on Intelligent Vehicles*, v. 2, n. 3, pp. 194–220, set. 2017. ISSN: 2379-8904. doi: 10.1109/TIV.2017.2749181.

[109] PAULL, L., SAEEDI, S., SETO, M., et al. "AUV Navigation and Localization: A Review", *IEEE Journal of Oceanic Engineering*, v. 39, n. 1, pp. 131–149, jan 2014. doi: 10.1109/joe.2013.2278891.

[110] THRUN, S. "Bayesian Landmark Learning for Mobile Robot Localization", *Machine Learning*, v. 33, n. 1, pp. 41–76, 1998. doi: 10.1023/a:1007554531242.

[111] SIM, R., DUDEK, G. "Mobile robot localization from learned landmarks". In: *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*. IEEE, 1998. doi: 10.1109/iros.1998.727439.

[112] SANTOS, M. M. D., GIACOMO, G. G. D., DREWS, P. L., et al. "Matching Color Aerial Images and Underwater Sonar Images Using Deep Learning for Underwater Localization", *IEEE Robotics and Automation Letters*, v. 5, n. 4, pp. 6365–6370, oct 2020. doi: 10.1109/lra.2020.3013852.

[113] PUENTE, I., GONZÁLEZ-JORGE, H., MARTÍNEZ-SÁNCHEZ, J., et al. "Review of mobile mapping and surveying technologies", *Measurement*, v. 46, n. 7, pp. 2127 – 2145, 2013. ISSN: 0263-2241.

[114] FILLIAT, D., MEYER, J.-A. "Map-based navigation in mobile robots:: I. A review of localization strategies", *Cognitive Systems Research*, v. 4, n. 4, pp. 243 – 282, 2003. ISSN: 1389-0417.

[115] BURGARD, W., HEBERT, M., BENNEWITZ, M. "World Modeling". In: *Springer Handbook of Robotics*, cap. 45, pp. 1135–1152, Cham, Springer International Publishing, 2016.

[116] MORAVEC, H., ELFES, A. "High resolution maps from wide angle sonar". In: *Proceedings IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers, 1985. doi: 10.1109/robot.1985.1087316.

[117] ELFES, A. "Using Occupancy Grids for Mobile Robot Perception and Navigation", *Computer*, v. 22, pp. 46–57, jun. 1989.

[118] MONTEMERLO, M., THRUN, S., KOLLER, D., et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Alberta, Canada, 2002.

[119] MONTEMERLO, M., THRUN, S. "Simultaneous localization and mapping with unknown data association using fastSLAM". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, v. 2, pp. 1985–1991. IEEE, set. 2003.

[120] MURPHY, K., RUSSELL, S. "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks". In: *Sequential Monte Carlo Methods in Practice*, Springer, pp. 499–515, New York, NY, USA, 2001. doi: 10.1007/978-1-4757-3437-9_24.

[121] KURT-YAVUZ, Z., YAVUZ, S. "A comparison of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM algorithms". In: *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*. IEEE, jun 2012. doi: 10.1109/ines.2012.6249866.

[122] STACHNISS, C. "Codes - Framework to SLAM", Disponível em <https://goo.gl/cgMCFu>. Acesso em 22 de marcco de 2018., 2018.

[123] VO, B. T. "Codes for MATLAB". 2017. Disponível em <http://ba-tuong.vo-au.com/codes.html>. Acesso em 29 de janeiro de 2017.

[124] CLARKE, B. "Gaussian Mixture Probability Hypothesis Density Filter (GM-PHD)". 2017. Disponível em <https://goo.gl/Rj5hVo>. Acesso em 29 de janeiro de 2017.

[125] SOLA, J. "SLAM Toolbox for Matlab." Disponível em <https://github.com/joansola/slamtb>. Acesso em 1 de fevereiro de 2018., 2018.

[126] O'DONNELL, R., MUEHE, C. "Automated Tracking for Aircraft Surveillance Radar Systems", *IEEE Transactions on Aerospace and Electronic Systems*, v. AES-15, n. 4, pp. 508–517, jul 1979. doi: 10.1109/taes.1979.308735.

[127] BESADA, J., MOLINA, J., GARCIA, J., et al. "Aircraft identification integrated into an airport surface surveillance video system", *Machine Vision and Applications*, v. 15, n. 3, jul 2004. doi: 10.1007/s00138-004-0135-8.

[128] KALYAN, B., LEE, K., WIJESOMA, S., et al. "A random finite set based detection and tracking using 3D LIDAR in dynamic environments". In: *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, oct 2010. doi: 10.1109/icsmc.2010.5641985.

[129] EDMAN, V., ANDERSSON, M., GRANSTRÖM, K., et al. "Pedestrian group tracking using the GM-PHD filter". In: *21st European Signal Processing Conference (EUSIPCO 2013)*, pp. 1–5, 2013. ISBN: 9780992862602.

[130] AHMED, I., JEON, G. "A real-time person tracking system based on SiamMask network for intelligent video surveillance", *Journal of Real-Time Image Processing*, v. 18, n. 5, pp. 1803–1814, jul 2021. doi: 10.1007/s11554-021-01144-5.

[131] JIMÉNEZ-BRAVO, D. M., MURCIEGO, Á. L., MENDES, A. S., et al. "Multi-object tracking in traffic environments: A systematic literature review", *Neurocomputing*, v. 494, pp. 43–55, jul 2022. doi: 10.1016/j.neucom.2022.04.087.

[132] JUANG, R. R., LEVCHENKO, A., BURLINA, P. "Tracking cell motion using GM-PHD". In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, jun 2009. doi: 10.1109/isbi.2009.5193262.

[133] WOOD, T. M., YATES, C. A., WILKINSON, D. A., et al. "Simplified Multi-target Tracking Using the PHD Filter for Microscopic Video Data", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, n. 5, pp. 702–713, may 2012. doi: 10.1109/tcsvt.2011.2177937.

[134] GRANSTROM, K., LUNDQUIST, C., ORGUNER, O. "Extended Target Tracking using a Gaussian-Mixture PHD Filter", *IEEE Transactions on Aerospace and Electronic Systems*, v. 48, n. 4, pp. 3268–3286, oct 2012. doi: 10.1109/taes.2012.6324703.

[135] GRANSTRÖM, K., NATALE, A., BRACA, P., et al. "PHD Extended Target Tracking Using an Incoherent X-band Radar: Preliminary Real-World Experimental Results". In: *FUSION 2014 - 17th International Conference on Information Fusion*, 2014.

[136] SIDENBLADH, H., WIRKANDER, S.-L. "Tracking Random Sets of Vehicles in Terrain". In: *2003 Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, jun 2003. doi: 10.1109/cvprw.2003.10097.

[137] ZHANG, F., CLARKE, D., KNOLL, A. "Vehicle detection based on LiDAR and camera fusion". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, oct 2014. doi: 10.1109/itsc.2014.6957925.

[138] GRANSTRÖM, K., REUTER, S., MEISSNER, D., et al. "A multiple model PHD approach to tracking of cars under an assumed rectangular shape". In: *17th International Conference on Information Fusion (FUSION)*, pp. 1–8, 2014.

[139] XIAO, J., XIONG, W., YAO, Y., et al. "Lane Detection Algorithm Based on Road Structure and Extended Kalman Filter", *International Journal of Digital Crime and Forensics*, v. 12, n. 2, pp. 1–20, apr 2020. doi: 10.4018/ijdcf.2020040101.

[140] TÖRŐ, O., BÉCSI, T., GÁSPÁR, P. "PHD Filter for Object Tracking in Road Traffic Applications Considering Varying Detectability", *Sensors*, v. 21, n. 2, pp. 472, jan 2021. doi: 10.3390/s21020472.

[141] LUO, W., XING, J., MILAN, A., et al. "Multiple object tracking: A literature review", *Artificial Intelligence*, v. 293, pp. 103448, apr 2021. doi: 10.1016/j.artint.2020.103448.

[142] PANTA, K. *Multi-Target Tracking Using 1st Moment of Random FiniteSets*. Tese de Doutorado, Department of Electrical and Electronic Engineering, The University of Melbourne, Australia, 2007.

[143] GRANSTRÖM, K. *Extended target tracking using PHD filters*. Tese de Doutorado, Department of Electrical EngineeringLinköping University, Linköping, Sweden, 2012.

[144] BAR-SHALOM, Y., LI, X. *Multitarget-multisensor Tracking: Principles and Techniques*. US, YBS, 1995.

[145] NATO. *STANAG 3680 - NATO Glossary of Terms and Definitions*. Relatório técnico, NATO, 1998.

[146] CHAO, L., YUEJI, W. "Review of multi-target tracking technology for marine radar", *Journal of Radars*, v. 10, n. 1, pp. 100–115, 2021.

[147] NAVY, B. "Strategic Programs". https://www.marinha.mil.br/programas-estrategicos, 2020. Disponível em: <https://www.marinha.mil.br/programas-estrategicos>. Accessed: 2020-04-02.

[148] VISWANATHAN, R., VARSHNEY, P. "Distributed detection with multiple sensors I. Fundamentals", *Proceedings of the IEEE*, v. 85, n. 1, pp. 54–63, 1997. doi: 10.1109/5.554208.

[149] BLUM, R., KASSAM, S., POOR, H. "Distributed detection with multiple sensors I. Advanced topics", *Proceedings of the IEEE*, v. 85, n. 1, pp. 64–79, 1997. doi: 10.1109/5.554209.

[150] DA, K., LI, T., ZHU, Y., et al. "Recent advances in multisensor multitarget tracking using random finite set", *Frontiers of Information Technology and Electronic Engineering*, v. 22, n. 1, pp. 5–24, jan 2021. doi: 10.1631/fitee.2000266.

[151] ZHAO, S., ZHOU, J. "A Fault-Tolerant Detection Fusion Strategy for Distributed Multisensor Systems", *International Journal of Distributed Sensor Networks*, v. 12, n. 2, pp. 8613149, feb 2016. doi: 10.1155/2016/8613149.

[152] YUAN, X., LIAN, F., HAN, C. "Models and Algorithms for Tracking Target with Coordinated Turn Motion", *Mathematical Problems in Engineering*, v. 2014, pp. 1–10, 2014. doi: 10.1155/2014/649276.

[153] LI, X. R., JILKOV, V. "Survey of maneuvering target tracking . Part I: dynamic models", *IEEE Transactions on Aerospace and Electronic Systems*, v. 39, n. 4, pp. 1333–1364, oct 2003. doi: 10.1109/taes.2003.1261132.

[154] LI, T., HLAWATSCH, F., DJURIC, P. M. "Cardinality-Consensus-Based PHD Filtering for Distributed Multitarget Tracking", *IEEE Signal Processing Letters*, v. 26, n. 1, pp. 49–53, jan 2019. doi: 10.1109/lsp.2018.2878064.

[155] LI, T., MALLICK, M., PAN, Q. "A Parallel Filtering-Communication-Based Cardinality Consensus Approach for Real-Time Distributed PHD Filtering", *IEEE Sensors Journal*, v. 20, n. 22, pp. 13824–13832, nov 2020. doi: 10.1109/jsen.2020.3004068.

[156] CICCOZZI, F., ADDAZI, L., ASADOLLAH, S. A., et al. "A Comprehensive Exploration of Languages for Parallel Computing", *ACM Computing Surveys*, v. 55, n. 2, pp. 1–39, mar 2022. doi: 10.1145/3485008.

[157] MUSTAFA, D. "A Survey of Performance Tuning Techniques and Tools for Parallel Applications", *IEEE Access*, v. 10, pp. 15036–15055, 2022. doi: 10.1109/access.2022.3147846.

[158] LI, T., SUN, S., BOLIĆ, M., et al. "Algorithm design for parallel implementation of the SMC-PHD filter", *Signal Processing*, v. 119, pp. 115–127, feb 2016. doi: 10.1016/j.sigpro.2015.07.013.

[159] GAO, L., TANG, X., WEI, P. "Real-Time Implementation of Particle-PHD Filter Based on GPU". In: *2014 IEEE 17th International Conference on Computational Science and Engineering*. IEEE, dec 2014. doi: 10.1109/cse.2014.308.

[160] GARLAND, M., GRAND, S. L., NICKOLLS, J., et al. "Parallel Computing Experiences with CUDA", *IEEE Micro*, v. 28, n. 4, pp. 13–27, jul 2008. doi: 10.1109/mm.2008.57.

[161] MELLO, R. R., ARAÚJO, L. S., ALVES, T. A. O., et al. "Gamma — General Abstract Model for Multiset mAnipulation and dynamic dataflow model: An equivalence study", *Concurrency and Computation: Practice and Experience*, v. 33, n. 11, jan 2021. doi: 10.1002/cpe.6176.

[162] BANÂTRE, J.-P., LE MÉTAYER, D. *A New Computational Model and Its Discipline of Programming*. In: Rapport de Recherche 566, INRIA, France, 1986.

[163] BARBOSA, R. *Fusão de alvos utilizando grafos em ambientes de múltiplos sensores*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2012.

[164] LIVERNET, F., CUILLIERE, O. "Tracking based on graph theory applied to pairs of plots". In: *2010 IEEE Radar Conference*, pp. 90–95. IEEE, 2010.

[165] MELLO, R. R., DE ALMEIDA, R. H. P., FRANCA, F. M., et al. "A Parallel Implementation of Data Fusion Algorithm Using Gamma". In: *2015 International Symposium on Computer Architecture and High Performance Computing Workshop (SBAC-PADW)*. IEEE, oct 2015. doi: 10.1109/sbac-padw.2015.25.

[166] DE MELLO JUNIOR, R. R., DE ARAÚJO, L. S., DUTRA, D. L. C., et al. "Fluid computing". In: *Proceedings of the 10th Euro-American Conference on Telematics and Information Systems*. ACM, nov 2020. doi: 10.1145/3401895.3401932.

[167] ARVIND, CULLER, D. E. "Dataflow Architectures", *Annual Review of Computer Science*, v. 1, n. 1, pp. 225–253, jun 1986. doi: 10.1146/annurev.cs.01.060186.001301.

[168] BANĂTRE, J.-P., FRADET, P., MÉTAYER, D. L. "Gamma and the Chemical Reaction Model: Fifteen Years After". In: *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 17–44, Berlin, Germany, 2001. doi: 10.1007/3-540-45523-x_2.

[169] BANÂTRE, J.-P., MÉTAYER, D. L. "Programming by multiset transformation", *Communications of the ACM*, v. 36, n. 1, pp. 98–111, jan 1993. doi: 10.1145/151233.151242.

[170] KALE, V. "Parallel computing architectures and APIs : IoT big data stream processing". cap. Parallel Computing Models, pp. 99–113, New York, NY, USA, CRC Press, 2019.

[171] TEICAN, S. C., ACIOBANITEI, I., PURA, M.-L. "Using Set Rewriting to Implement the Chemical Reaction Computation Model". In: *2018 International Conference on Communications (COMM)*. IEEE, jun 2018. doi: 10.1109/iccomm.2018.8484749.

[172] BANÂTRE, J.-P., MÉTAYER, D. L. "The gamma model and its discipline of programming", *Science of Computer Programming*, v. 15, n. 1, pp. 55–77, nov 1990. doi: 10.1016/0167-6423(90)90044-e.

[173] DE MELLO JUNIOR, R. R. *Explorando a equivalência entre uma máquina química abstrata e computação dataflow.* Tese de Doutorado, Programa de Pós-graduação em Engenharia de Sistemas, COPPE - UFRJ, 2022.

[174] DE ALMEIDA, R. H. P., MELLO, R. R., PAILLARD, G. A. L., et al. "A GPU-based implementation for the gamma multiset rewriting paradigm". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, apr 2016. doi: 10.1145/2851613.2851719.

[175] LEE, C. S. *Simultaneous Localization and Mapping Using Single Cluster Probability Hypothesis Density Filters.* Tese de Doutorado, Universitat de Girona, ESP, 2014.

[176] AULINAS, J., LLADO, X., SALVI, J., et al. "SLAM base Selective Submap Joining for the Victoria Park Dataset", *IFAC Proceedings Volumes (IFAC-PapersOnline)*, v. 7, 09 2010.

Apêndice A

Publicações

APLICAÇÃO DE CONJUNTOS ALEATÓRIOS FINITOS NO PROBLEMA DE LOCALIZAÇÃO E MAPEAMENTO DE ROBÔS MÓVEIS

KLEBERSON M. LIMA[*], DIEGO PEREIRA-DIAS[*], RAMON R. COSTA[*]

[*]*Universidade Federal do Rio de Janeiro (UFRJ)*
Programa de Engenharia Elétrica (PEE/COPPE/UFRJ)
Rio de Janeiro, Rio de Janeiro, Brasil

Emails: klebersonmlima@outlook.com, dpd@metalmat.ufrj.br, ramon@coep.ufrj.br

Abstract— This paper presents an overview on Random Finite Set (RFS) applied to the problem of mapping in robotics. A solution is presented for the RFS-based mapping problem. Initially, this type of approach is presented and the use of its advantages over vector-based techniques is motivated. Conflicts that occur in construction maps are shown when using conventional approaches, such as: maps dependent on the trajectory traversed and the problem of data association. Once justified the use of RFS in the problem of mapping and location of robots, the mathematical framework involved is presented and the main concepts pertinent to this formalism are defined. Afterwards, the RFS is presented within the Bayesian framework, the PHD filter. Finally, through simulations, an application to obtain a map of the environment using the Probability Hypothesis Density (PHD) filter is shown.

Keywords— Robotics, mapping, Random Finite Sets.

Resumo— Este trabalho apresenta uma visão geral sobre Conjuntos Aleatórios Finitos (do inglês *Random Finite Sets*) aplicado ao problema de mapeamento em robótica. Além disso, aplica-se uma solução baseada em RFS para o problema de mapeamento FBRM (*Feature-Based Robot Mapping*). Inicialmente, faz-se uma apresentação desse tipo de abordagem e motiva-se a utilização dessa técnica através de suas vantagens em relação às vetoriais. São mostrados conflitos que acontecem na construção de mapas, quando do uso das abordagens convencionais, tais como: mapas dependentes da trajetória percorrida e o problema da associação de dados. Uma vez que justificado o uso de RFS no problema de mapeamento e localização de robôs, apresenta-se o arcabouço matemático envolvido e define-se os principais conceitos pertinentes a esse formalismo. Em seguida, apresenta-se o RFS dentro do *framework* Bayesiano, o filtro PHD, do inglês *Probability Hypothesis Density* . Por fim, através de simulações, mostra-se uma aplicação para obtenção de um mapa do ambiente com uso do filtro PHD.

Palavras-chave— robótica, mapeamento, Random Finite Sets.

1 Introdução

A importância da robótica na nossa vida cotidiana tem crescido nos últimos anos. Tornou-se frequente, ao se abrir um portal de notícias na *internet*, encontrar manchetes sobre empresas que desejam distribuir seus produtos vendidos *on-line* por meio de um *drone*, robôs utilizados em limpeza doméstica ou sobre sistemas de transportes que operam sem a interferência humana. A autonomia é o ponto comum entre elas.

Quando o mapa já é conhecido e o robô necessita saber (ou estimar) onde se encontra, através de seus diversos sensores, tem-se o que é chamado de **localização**. Uma vez que os processos de localização e mapeamento são solucionados simultaneamente, tem-se o que se chama **SLAM** (*Simultaneous Localization And Mapping*). Segundo (Mullane et al., 2011a), a representação computacional, e incerteza, desses processos é crucial para a navegação autônoma.

Para realizar essas tarefas, o sistema autônomo necessita obter informações sobre o mundo ao seu redor para que possa fornecer subsídios para os sistemas de guiagem do robô. A tarefa de construção de um modelo do ambiente é chamada de **mapeamento**.

O problema de SLAM surgiu na década de 90 e até hoje apresenta-se como um dos maiores desafios em robótica móvel e sistemas autônomos. Tradicionalmente, são utilizadas três técnicas para se resolver esse problema: filtros de Kalman, filtros de partículas e as baseadas em teoria dos grafos (Siciliano and Khatib, 2016).s

As técnicas baseadas em filtros de Kalman e de partículas são as mais amplamente exploradas e aplicadas (Zhang et al., 2017). Apesar dessas técnicas serem bem estabelecidas, elas apresentam uma dificuldade extra em aplicações reais: a associação de dados. De acordo com Siciliano and Khatib (2016), esta é uma dificuldade bastante complexa que deve ser tratada à parte do problema de SLAM, mas que pode afetar diretamente o desempenho da solução. Logo, soluções alternativas, que lidem de maneira mais simples com a essa questão da associação de dados, são de interesse para os problemas de localização e mapeamento.

Nesse contexto, a teoria dos conjuntos aleatórios finitos (do inglês, *Random Finite Sets* - RFS) vem sendo estudada e, de acordo com (Mullane et al., 2011a), é uma ferramenta mais apropriada para a solução do problema de mapeamento (ou SLAM) quando comparada às técnicas convencionais.

Este trabalho apresenta um estudo sobre a aplicação da técnica RFS no problema de mapeamento. A contribuição deste trabalho é fomentar o estudo das técnicas baseadas em RFS em nossa comunidade, através de um caso em ambiente simulado. O trabalho está organizado da seguinte forma: na Seção 2 resume-se o problema de localização, na Seção 3 apresenta-se o ferramental teórico dos RFS, na Seção 4 descrevem-se os resultados obtidos e na Seção 5 são expostas as conclusões e comentários finais.

2 Formulação do Problema

Localização é o problema de descobrir onde os objetos estão, inclusive o próprio robô. O conhecimento disso é o cerne de qualquer interação física bem sucedida com o meio ambiente (Russel and Norvig, 2010).

Segundo (Thrun et al., 2005), **localização**, em **robótica móvel**, é o problema de se determinar a *pose* de um robô em relação a um mapa do ambiente. Para o caso de um robô móvel que se move no plano, temos que a *pose*, em um dado instante t, é definida pelas coordenadas cartesianas e a orientação da parte frontal, ou seja, $x_t = \begin{bmatrix} x & y & \theta \end{bmatrix}^\top$.

A localização pode ser vista como um problema de transformação de coordenadas, com o mapa sendo o sistema de coordenadas global. Portanto, a localização é o processo de estabelecer uma relação entre as coordenadas globais e as locais do robô. Contudo, de acordo com (Thrun et al., 2005), em geral a pose não pode ser medida diretamente. Ou seja, a maioria dos robôs não possuem um sensor para medi-la. Logo, a pose deve ser inferida a partir de dados (por exemplo, por meio de um sensor de distância).

Por conseguinte, no problema de localização, o objetivo é estimar a posição do robô dados: o mapa do ambiente M, as observações z_t e a ação de controle u_t.

Segundo (Thrun et al., 2005), o **mapa** é uma lista de objetos com as respectivas localizações em um ambiente e é definido formalmente por:

$$m = \{m_1, m_2, \ldots, m_N\}, \tag{1}$$

onde N é o número total de objetos no ambiente e cada m_i, com $i = 1:N$, representa as propriedades ou características de um objeto.

Há duas abordagens para a construção de mapas: as baseadas em características (*feature-based*) e as em localização (*location-based*). Em consonância com (Thrun et al., 2005), mapas baseados em localização são volumétricos e carregam a informação não apenas sobre os espaços ocupados por objetos mas também por espaços vazios. No caso planar $m_i = m_{x,y}$, o que evidencia tratar-se de uma coordenada (x, y). Já nas abordagens baseadas em características, o mapa define a localização de cada objeto e uma (ou mais) característica. Esse estudo concentra-se na abordagem baseada em características (*Feature Based Robotic Mapping* – FBRM).

De acordo com (Thrun et al., 2005), um vetor de características é definido por:

$$f(z_t) = \left\{ f_t^1, f_t^2, \ldots \right\} = \left\{ (r_t^1, \phi_t^1, s_t^1), (r_t^2, \phi_t^2, s_t^2), \ldots \right\},$$

onde r é a distância para a origem de um sistema de coordenadas, ϕ é a orientação em relação à mesma origem e s é a assinatura, a qual pode ser um valor numérico (ex: para caracterizar uma cor) ou um valor multidimensional (ex: altura e cor).

Resumidamente, o problema de mapeamento caracteriza-se por estimar o mapa m através da distribuição de probabilidade $p(m \mid z_t, x_t)$. Já o problema de SLAM, caracteriza-se por ser uma combinação dos dois problemas, portanto, estimar o mapa e a localização do robô simultaneamente.

3 Conjuntos Aleatórios Finitos

De acordo com (Mullane et al., 2011a), as técnicas convencionais (baseadas em vetores) utilizadas em FBRM e SLAM sofrem dificuldades quando aplicadas em situações realísticas. Isso acontece em ambientes nos quais um número a priori desconhecido de características deve ser estimado e, na presença de defeitos realistas dos sensores, como detecções não atendidas e falsos alarmes.

Um RFS é uma variável aleatória que toma valores como conjuntos finitos. É definido por uma distribuição discreta que caracteriza o número de elementos no conjunto e uma família de distribuições conjuntas que caracterizam a distribuição dos valores dos elementos, condicionada à cardinalidade (Dames, 2015):

$$p(X) = p(|X| = n)\, p(X = \{x_1, x_2, \ldots, x_n\} \mid |X| = n). \tag{2}$$

Um exemplo de RFS pode ser visualizado na Figura 1.

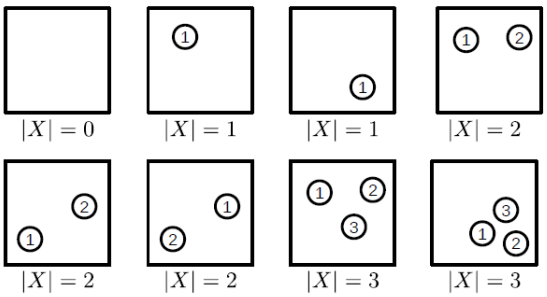

Figura 1: Exemplos de um RFS de 0 a 3 elementos em um ambiente quadrado.
Fonte: Extraído de (Mullane et al., 2011a).

O problema de rastreamento multi-alvos utilizando RFS já é amplamente difundido na comunidade de radares e, recentemente, vem sendo aplicado em robótica móvel, de acordo com (Dames, 2015).

RFS x Técnicas Baseadas em Vetores

Segundo (Dames, 2015), é preciso destacar as diferenças entre as técnicas baseadas em vetores e em conjuntos quando se considera os problemas de FBRM e SLAM (o qual pode ser entendido como um problema *multi-target* quando os alvos são estáticos). Nesse tipo de problema, a quantidade de características (*landmarks*) não é conhecida *a priori* e deve ser descoberta à medida que o robô explora o ambiente. Dois pontos principais, que serão explicados a seguir, surgem nesse tipo de problema:

- Gerenciamento das características (*feature management*) - estimar a posição dos objetos/características no mapa;
- Associação de dados (*data association*) - atrelar as observações aos objetos/características.

Observa-se, na Figura 2, o cenário hipotético em que um robô percorre três trajetórias distintas X_1, X_2 e X_3 nas vizinhanças dos objetos $m_1, \ldots, m_7$.

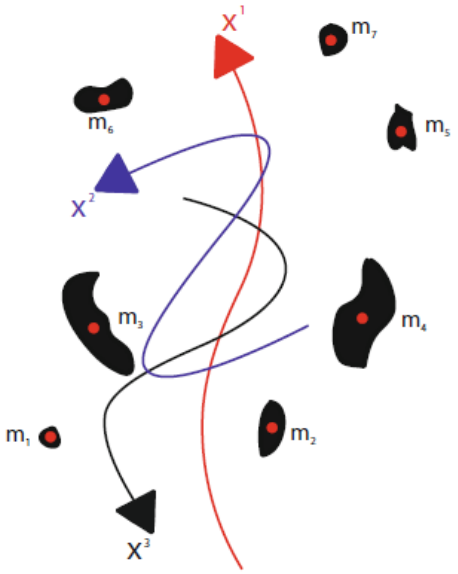

Figura 2: Três trajetórias realizadas por um robô em um cenário hipotético.
Fonte: Extraído de Mullane et al. (2011a).

Caso seja usado um vetor M para representar o mapa estimado, para cada trajetória, teríamos $\widehat{M}_1 = [m_1 \ m_2 \ m_3 \ m_4 \ m_5 \ m_6 \ m_7]^T$, $\widehat{M}_2 = [m_4 \ m_2 \ m_3 \ m_1 \ m_5 \ m_7 \ m_6]^T$ e $\widehat{M}_3 = [m_6 \ m_7 \ m_5 \ m_4 \ m_3 \ m_2 \ m_1]^T$. Como a ordem dos elementos num vetor é importante, ter-se-ia três mapas diferentes, destacando-se o fato de que o mapeamento seria dependente da trajetória.

Contudo, no sentido lógico, o mapeamento deveria ser independente da trajetória percorrida pelo veículo e, matematicamente, qualquer permutação nos elementos dos vetores seria uma representação válida. Por definição, a representação que captura todas as permutações dos elementos dentro de um vetor (e, por conseguinte, todas as características do mapa) é o conjunto finito $\widehat{\mathscr{M}} = \{m_1 \ m_2 \ m_3 \ m_4 \ m_5 \ m_6 \ m_7\}$ (Mullane et al., 2011a).

Uma outra motivação para o uso de RFS é o problema de associação entre as observações z e as respectivas características m, como pode ser visualizado na Figura 3. As técnicas de FBRM e SLAM baseadas em vetores exigem um passo de reordenação das observações antes que um filtro Bayesiano (Kalman, EKF, partículas etc.) possa ser aplicado (Mullane et al., 2011a), .

Por outro lado, utilizando-se de abordagem baseada em RFS, é possível fazer uma associação direta entre o conjunto $\mathscr{M}$ das características e o $\mathscr{Z}$ das observações.

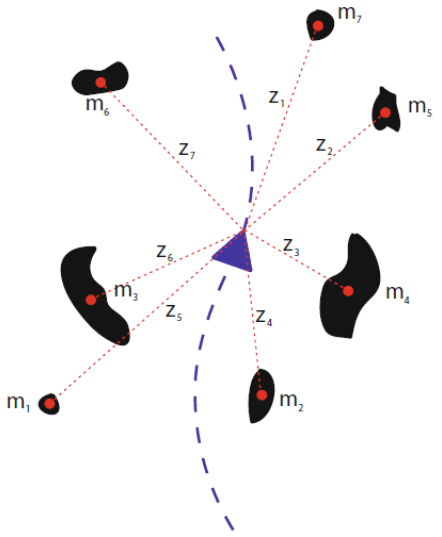

Figura 3: Vetores das observações diferentes a depender da trajetória.
Fonte: Extraído de Mullane et al. (2011a).

Um outro problema contornável pelo uso dos RFSs, quando comparado às técnicas baseadas em vetores, é o devido ao uso de sensores não ideais associado ao problema de crescimento do mapa por detecções falsas ou objetos perdidos (fora do campo de detecção). Esse tipo de problema pode ser exemplificado a partir da Figura 4.

Supondo que as características m_1, m_2 e m_3 já tenham sido detectadas no instante $k-1$, em uma representação baseada em vetores, e que no instante k uma nova característica m_4 é detectada, ter-se-ia:

$$\widehat{M}_{k-1} = [m_1 \ m_2 \ m_3]^T \tag{3}$$

e

$$\widehat{M}_k = [m_1 \ m_2 \ m_3]^T \text{ “+” } m_4. \tag{4}$$

Segundo (Mullane et al., 2011a), a operação de atribuição de m_4 não é clara do ponto de vista matemático. E, mesmo existindo técnicas que usem um vetor aumentado, com o uso de RFS, bastaria usar a operação de união entre os conjuntos nos instantes $k-1$ e k:

$$\widehat{\mathscr{M}}_k = \{m_1 \ m_2 \ m_3\} \cup \{m_4\}. \tag{5}$$

Um outro problema fundamental de qualquer FBRM ou SLAM é a necessidade de se relacionar as observações ao estado estimado (Mullane et al., 2011a).

$$Z_k = h([m_1 \ m_2 \ m_3 \ m_4], X_k) + \text{ruído} \tag{6a}$$

$$\text{i.e.} [z_1 \ z_2 \ z_3 \ z_4 \ z_5] = h([m_1 \ m_2 \ m_3 \ m_4], X_k) + \text{ruído}, \tag{6b}$$

onde Z_k representa o vetor de observações no tempo k (aqui se usa o exemplo da Figura 4), X_k é a **pose** do

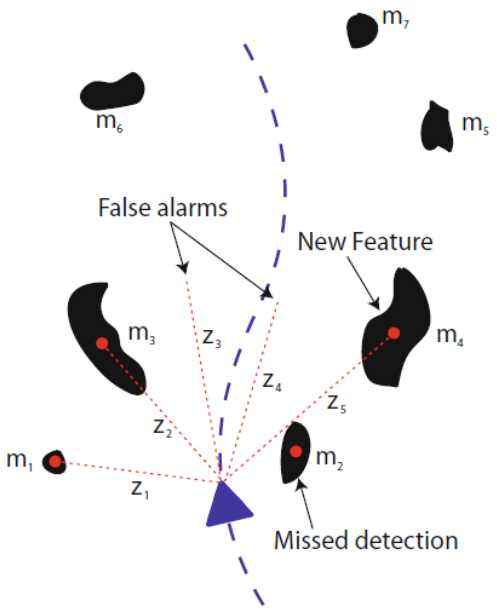

Figura 4: Detecção perdida com sensor realista (não ideal).
Fonte: Extraído de Mullane et al. (2011a).

robô no instante k e h é o mapeamento (tipicamente não linear) entre a pose do robô e as observações. Esse exemplo mostra a dificuldade encontrada para mapear as quatro características (estados) com cinco observações. A observação extra é, possivelmente, devido a uma detecção espúria e a uma característica que foi "perdida". Numa abordagem convencional, é indefinida a forma como integrar ao equacionamento esses dois "problemas". É preciso um gerenciamento heurístico do mapa para remover estados em excesso e forçar o funcionamento do mapeamento.

Caso sejam utilizados conjuntos para representar as observações e mapa estimado, tem-se a seguinte relação:

$$\mathscr{Z}_k = \bigcup_{m \in \mathscr{M}_k} \mathscr{D}_k(m, X_k) \cup \mathscr{C}_k(X_k), \tag{7}$$

onde $\mathscr{D}_k(m, X_k)$ é o RFS das medições geradas por uma característica em m e dependente da pose X_k. Além disso, $\mathscr{C}_k(X_k)$ é o RFS das medições espúrias que poderia ser em função da pose. Portanto, $\mathscr{Z}_k = \{z_k^1, z_k^2, ..., z_k^{zk}\}$ consiste de um número aleatório, zk, de medições cuja ordem em que aparecem não tem um significado físico com relação à estimativa das características no mapa. Então, de acordo com (Mullane et al., 2011a), fica claro que, nas técnicas baseadas em vetores, a incerteza no número de características não é modelada diferente de quando se usa RFS (a incerteza é algo intrínseco à técnica).

4 Conceitos Fundamentais em RFS

Segundo (Dames, 2015), apesar de todas as vantagens sobre as abordagens baseadas em vetores, com RFS há a dificuldade de tratar com matemática não familiar para a maioria dos profissionais e acadêmicos que lidam com robótica. Portanto, para ser possível realizar uma inferência estatística com conjuntos, é preciso definir variáveis aleatórias adequadas e ser capaz de realizar operações como o cálculo de médias dessas variáveis.

4.1 *Random Finite Set*

De acordo com (Ristic et al., 2016), um RFS é um modelo probabilístico conveniente para a representação de múltiplos sistemas dinâmicos estocástico (objetos ou características) e medições dos sensores.

Suponha que, no instante k, exista n_k objetos com estados $x_{k,1}, \ldots, x_{k,n_k}$ assumindo valores do espaço de estados $\mathscr{X} \subseteq \mathscr{R}^{n_k}$. Tanto n_k quanto o número de estados individuais em $\mathscr{X}$ são variáveis aleatórias e variantes no tempo. O estado das múltiplas características, no instante k, é um **estado finito** definido por:

$$\mathbf{X}_k = \{x_{k,1}, \ldots, x_{k,n_k}\} \in \mathscr{F}(\mathscr{X}), \tag{8}$$

o qual pode ser modelado como um RFS em $\mathscr{X}$. Onde $\mathscr{F}(\mathscr{X})$ é o conjunto finito de todos os subconjuntos de $\mathscr{X}$.

De maneira similar, para as observações teremos:

$$\mathbf{Z}_k = \{z_{k,1}, \ldots, z_{k,m_k}\} \in \mathscr{F}(\mathscr{Z}),$$

pode ser modelado como um RFS sobre o estado de observações $\mathscr{Z} \subseteq \mathscr{R}^{m_k}$. Tanto a cardinalidade $|\mathbf{Z}_k|$ quanto os estados individuais em $\mathbf{Z}_k$ são aleatórios e $\mathscr{F}(\mathscr{Z})$ é o conjunto de todos os subconjuntos de $\mathscr{Z}$.

Um RFS é totalmente caracterizado por sua distribuição de cardinalidade $\rho(n) = p(|X| = n)$, com $n \in \mathbb{N}$ e por uma família de distribuições conjuntas simétricas $p_n(x_1, ..., x_n) = p(X = \{x_1, ..., x_n\} \,||X| = n)$ (vide (2)).

4.2 *Set integral*

Seja $f(\mathbf{X})$ uma função real sobre um conjunto, a *set integral* de $f(\mathbf{X})$ é definida por:

$$\int f(\mathbf{X}) \delta \mathbf{X} = \sum_{n=0}^{\infty} \frac{1}{n!} \int f(\{x_1, ..., x_n\}) \mathrm{d}x_1 \ldots \mathrm{d}x_n. \tag{9}$$

Essa integral caracteriza a soma sobre a cardinalidade de um conjunto, integrando todos os conjuntos possíveis de acordo com cardinalidade. Além disso, o termo $1/n!$ leva em conta as permutações de um conjunto de tamanho n.

O interesse é quando $f(\mathbf{X})$ é um distribuição de probabilidade!

4.3 *Distribuições de Probabilidade Sobre um RFS*

Considerando a hipótese de que os elementos do RFS são independentes e identicamente distribuídos

(mesma distribuição de probabilidade), a probabilidade de um RFS é dada por:

$$p(\mathbf{X}) = |X|!p(|X|)\prod_{x\in X} p(X), \tag{10}$$

onde $|\cdot|$ é o operador cardinalidade, $|X|!$ é o número de permutações dos elementos do conjunto, $p(|X|)$ é a distribuição de cardinalidade e $p(X)$ é a probabilidade de uma característica (*landmark*) ter o estado x. Para $p(X)$, (10) ser uma distribuição de probabilidade válida, é necessário que a *set integral* seja unitária, a verificação pode ser obtida em (Ristic et al., 2016).

4.4 Probability Hypothesis Density (PHD)

A função intensidade (ou simplesmente PHD) é importante na caracterização do primeiro momento estatístico de uma RFS, segundo (Ristic et al., 2016). Inicialmente, é preciso definir a função *Set Dirac Delta*:

$$\delta_X(X) = \sum_{w\in X} \delta_w(x), \tag{11}$$

onde $\delta_w(x)$ é a função delta de Dirac *standard* concentrada em w. Pode-se, então, definir a cardinalidade de um RFS como:

$$|X| = \int_{\mathscr{X}} \delta_X \, \mathrm{d}x \tag{12}$$

A função intensidade $D(x)$ pode ser definida de tal forma que a cardinalidade esperada de $\mathbf{X}$ sobre $\mathscr{X}$ é a integral:

$$\{|X|\} = \int_{\mathscr{X}} D(X) \, \mathrm{d}x. \tag{13}$$

Desenvolvendo a integral é possível obter (Ristic et al., 2016),

$$D_X(x) = \{\delta_X(X)\} = \int_{\mathscr{X}} \delta_X(x) f(X) \, \mathrm{d}x. \tag{14}$$

O valor esperado da PHD corresponde ao número de características (*landmarks*) de uma região e a PHD não se trata de uma distribuição de probabilidade (a área sob a curva não é igual a 1) (Dames, 2015).

4.5 Filtro PHD

A abordagem mais básica usada para estimação com RFS é a qual atualiza a média recursivamente (Dames, 2015). Além disso, é o análogo do Filtro de Kalman, contudo, usa uma distribuição de Poisson, que é definida por

$$p(X) = e^{-\lambda} \prod_{x\in\mathscr{X}} v(x), \tag{15}$$

onde $\lambda = \int v(x)dx$.

Mostra-se o problema do ponto de vista *bayesiano*, o qual possui formulação similar aos filtros bayesianos para o caso vetorial. Ou seja, o problema é resolvido recursivamente e é dividido em duas partes: predição (através do modelo de transição de estados) e filtragem (fusão da informação obtida através das observações ou sensores).

4.6 Mapeamento Baseado em RFS

Como mencionado anteriormente, em contraste com a representação vetorial, o RFS dos estados do mapa $\mathscr{M}_k$ pode encapsular de forma unificada as incertezas de cardinalidade e localização. Assim como o RFS de medidas $\mathscr{Z}_k$ pode encapsular as incertezas de medida e valor.

Como realizado em uma seção anterior deste estudo, o RFS é definido de acordo com a equação (6b). Nessa equação, há um conjunto $\mathscr{D}_k$ obtido a partir das medidas geradas pelas características (*features*) e outra $\mathscr{C}_k$ pelas medidas espúrias, em um instante k, que podem depender da posição $\mathscr{X}_k$ do robô.

De forma geral, $\mathscr{Z}_k$ pode representar uma série de parâmetros medidos, mas, no caso mais simples aqui descrito, é uma medida de distância e ângulo em relação ao sensor do robô. É importante ressaltar que a cardinalidade de $\mathscr{Z}_k$ geralmente difere da de $\mathscr{M}_k$ devido às incertezas nas medições, oclusões, medições espúrias e novos elementos entrando no campo de visão do robô $\mathscr{D}_k$ e $\mathscr{C}_k$ são assumidos como RFS independentes.

O RFS de medidas gerado por uma *feature m* tem densidade de probabilidades igual a $P_{\mathscr{D}}(m,\mathscr{X}_k)g_k(z\,|\,m,\mathscr{X}_k)$, onde o primeiro termo é a probabilidade de um sensor detectar uma *feature m* e o segundo é a probabilidade de uma *feature m* gerar uma medida z.

A habilidade de um sensor (ou um algoritmo de detecção de features) detectar um dado objeto pode ser altamente influenciada por sua posição relativa. Por exemplo, uma oclusão ou grande distância pode resultar em $P_{\mathscr{D}} = 0$.

A densidade de probabilidade de um sensor produzir uma medida $\mathscr{Z}_k$, dado um estado do robô (posição e orientação), $\mathscr{X}_k$, é dada pela convolução:

$$g_k(\mathscr{Z}_k|\mathscr{X}_k,\mathscr{M}_k) = \sum_{\mathscr{W}\subseteq\mathscr{Z}_k} g_{\mathbb{D}}(\mathscr{W}|\mathscr{M}_k,\mathscr{X}_k)g_{\mathscr{C}}(\mathscr{Z}_k-\mathscr{W}), \tag{16}$$

onde $g_{\mathscr{C}}(\mathscr{Z}_k-\mathscr{W})$ é a densidade das medições espúrias e $g_{\mathbb{D}}$ é a densidade do RFS $\mathscr{D}_k$ das medições geradas pelas *features* em $\mathscr{M}_k$ (dado estado do veículo). O primeiro termo incorpora as incertezas das medições e ruídos, o segundo termo modela os dados espúrios gerados pelo sensor e é tipicamente definido previamente. É reproduzida abaixo a forma geral da recursão de Bayes para mapeamento baseado em RFS:

$$p_{k|k-1}(\mathscr{M}_k,\mathscr{Z}_{0:k-1},\mathscr{X}_k) = \int f_{\mathscr{M}}(\mathscr{M}_k|\mathscr{M}_{k-1},\mathscr{X}_k)p_{k-1}(\mathscr{M}_k,\mathscr{Z}_{0:k-1},\mathscr{X}_{0:k-1})\delta\mathscr{M}_{k-1} \tag{17}$$

e

$$p_k(\mathscr{M}_k,\mathscr{Z}_{0:k},\mathscr{X}_k) = \frac{g_k(\mathscr{Z}_k|\mathscr{X}_k,\mathscr{M}_k)p_{k|k-1}(\mathscr{M}_k,\mathscr{Z}_{0:k-1},\mathscr{X}_k)}{\int g_k(\mathscr{Z}_k|\mathscr{X}_k,\mathscr{M}_k)p_{k|k-1}(\mathscr{M}_k,\mathscr{Z}_{0:k-1},\mathscr{X}_k)\delta\mathscr{M}_k}, \tag{18}$$

sendo δ a indicação de integral de um conjunto e a integral é definida de acordo com a equação (9).

Esse mapa baseado em *features* encapsula as incertezas inerentes no número de características introduzidas na detecção, medidas espúrias e movimentos do robô, assim como as incertezas de posição introduzidas pelo ruído de medição.

Como fica evidente pelas equações acima, a aplicação da forma geral da recursão de Bayes é matematicamente intratável. Uma aproximação baseada em filtro PHD é proposta em (Mullane et al., 2011b). O objetivo é propagar a intensidade υ_k do mapa no lugar de toda a densidade multidimensional (vide (18)).

$$\upsilon_{k|k-1}(m|X_k) = \upsilon_{k-1}(m|X_{k-1}) + b_k(m|X_k), \qquad (19)$$

onde b_k é o PHD do RFS das últimas *features* detectadas $\mathscr{B}(\mathscr{X}_k)$.

A correção do filtro PHD, que tem forma similar ao Filtro de Bayes usual, é dada por:

$$\upsilon_k(m|X_k) = \upsilon_{k|k-1}(m|X_k) \times \left[[1 - P_{\mathscr{D}}(m|X_k) + \sum_{z \in \mathscr{Z}_k} \frac{P_{\mathscr{D}} g_k(z|m,X_k)}{c_k(z|X_k) + \int P_{\mathscr{D}}(\xi|X_k) g_k(z|\xi,X_k) \upsilon_{k|k-1}(\xi|X_k) d\xi} \right], \qquad (20)$$

onde $\upsilon_k(m|X_k)$ é a predição para a função de intensidade, $P_{\mathscr{D}}(m|X_k)$ a probabilidade de uma *feature* m ser detectada com o robô na posição X_k, $g_k(z|m,X_k)$ o modelo de medição do sensor no instante k e $c_k(z|X_k)$ a intensidade de medições espúrias no instante k.

5 Simulação e Resultados

Para a simulação dos resultados, foram utilizados como base os dados do ambiente e dos sensores disponibilizados para download em (Stachniss, 2018). Tal *framework* foi adaptado para receber o algoritmo implementado, o qual foi baseado nas referências (VO and MA, 2006), (Clark et al., 2006) e (Mullane et al., 2011a), bem como nos códigos disponibilizados em (Vo; Clarke) e (Sola, 2018). O mapa *true* do ambiente, com escala em metros, pode ser visualizado na Figura 5. As *landmarks* estão marcadas em azul, caminhos que não devem ser cruzados em preto e a trajetória deseja para o robô está em vermelho.

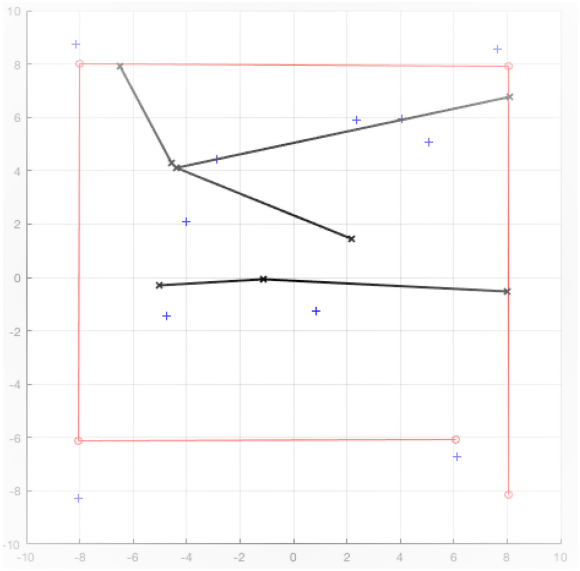

Figura 5: Mapa do ambiente com as posições reais das *features* em azul e trajetória planejada para o robô em vermelho.

O resultado obtido, após o seguimento da trajetória pelo robô, pode ser observado na Figura 6. A trajetória estimada é apresentada em azul, a trajetória executada pelo veículo em vermelho. Já as *features* estimadas estão em verde, sendo os círculos azuis as *features true*.

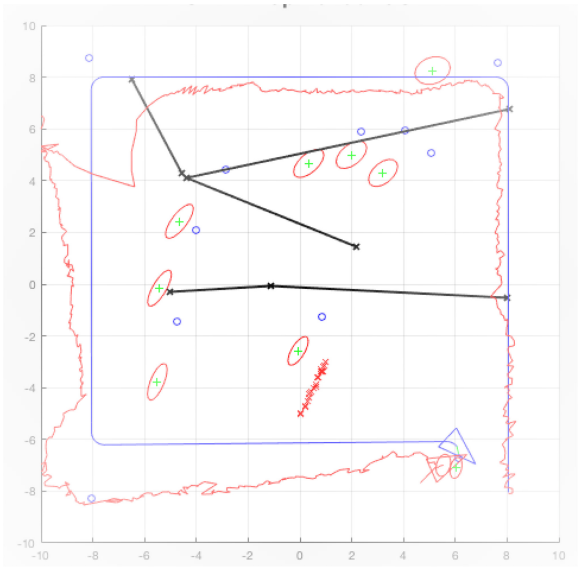

Figura 6: Posições estimadas das *features* x posições reais.

Neste estudo foi considerado apenas o problema de mapeamento. Mesmo no RFS SLAM, para um melhor desempenho computacional, há uma separação do problema , sendo a localização normalmente realizada por algum algoritmo baseado em Kalman ou filtro de partículas, conforme (Ristic et al., 2016).

6 Conclusão

Inicialmente, pode-se concluir que o uso de *Random Finte Sets* é adequado à aplicação no problema de mapeamento em robótica. Tal resultado já era esperado, dado que fundamentalmente foi pensado para o problema de *multi-target tracking* oriundo da comunidade de radares.

Também é possível verificar que, apesar de todas as vantagens apresentadas, o arcabouço matemático diferente do usual, do utilizado pela comunidade e estudiosos em robótica, pode tornar-se um entrave.

Uma vez superada tal dificuldade, a aplicação do filtro PHD apresenta resultados consistentes e há uma associação direta entre a integral da função intensidade e número de *features* ou *landmarks* existente no ambiente. Além disso, torna-se uma tarefa simples estimar a posição desses marcos, já que os valores de máximo locais da mistura Gaussiana correspondem à posição deles. E, em posse dessa informação, é possível obter-se o vetor das médias e, portanto, as coordenadas x e y de cada *feature*.

Dessa forma, a principal colaboração desse trabalho é fomentar o estudo das técnicas baseadas em RFS em nossa comunidade. Além disso, este trabalho abre a possibilidade para aplicação de RFS em problemas reais nos robôs pertencentes ao Laboratório de Controle (LabCON/PEE) da Universidade Federal do Rio de Janeiro. Portanto, espera-se apresentar trabalhos futuros dessa forma, além de uma evolução para o caso mais complexo SLAM e elaboração de um ambiente simulado via ROS.

Referências

D. Clark, K. Panta, and B.-n. Vo. The gm-phd filter multiple target tracker. In *Proceedings of 9th International Conference on Information Fusion*, 2006. doi: 10.1109/ICIF.2006.301809.

B. Clarke. Gaussian mixture probability hypothesis density filter (gm-phd). Disponível em <`https://goo.gl/Rj5hVo`>. Acesso em 29 de janeiro de 2017.

P. M. Dames. *Multi-Robot Active Information Gathering Using Random Finite*. PhD thesis, Philip M. Dames, 2015.

J. Mullane, B.-N. Vo, M. Adams, and B.-T. Vo. Mobile robotics in a random finite set framework. In *Lecture Notes in Computer Science*, pages 519–528. Springer Nature, 2011a. doi: 10.1007/978-3-642-21524-7_64.

J. Mullane, B.-N. Vo, M. D. Adams, and B.-T. Vo. A random-finite-set approach to bayesian SLAM. *IEEE Transactions on Robotics*, 27(2):268–282, apr 2011b. doi: 10.1109/tro.2010.2101370.

B. Ristic, M. Beard, and C. Fantacci. An overview of particle methods for random finite set models. *Information Fusion*, 31(C):110–126, September 2016.

S. J. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, 2nd edition, 2016. Chapter 46.

J. Sola. Slam toolbox for matlab. Disponível em <`https://github.com/joansola/slamtb`>. Acesso em 1 de fevereiro de 2018., 2018.

C. Stachniss. Codes - framework to slam. Disponível em <`https://goo.gl/cgMCFu`>. Acesso em 22 de marcco de 2018., 2018.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligente Robotics and Autonomous Agent. MIT Press, August 2005.

B. Vo. Codes for matlab. Disponível em <`http://ba-tuong.vo-au.com/codes.html`>. Acesso em 29 de janeiro de 2017.

B. VO and W. MA. The gaussian mixture probability hypothesis density filter. *IEEE Transactions Signal Processing*, 54(11):4091–4104, September 2006. doi: 10.1109/TSP.2006.881190.

F. Zhang, S. Li, S. Yuan, E. Sun, and L. Zhao. Algorithms analysis of mobile robot slam based on kalman and particle filter. In *Proceedings of 9th International Conference on Modelling, Identification and Control (ICMIC)*, Kunming, China, July 2017.

sensors

MDPI

Communication

Cooperative-PHD Tracking Based on Distributed Sensors for Naval Surveillance Area

Kleberson Meireles de Lima * and Ramon Romankevicius Costa

PEE/COPPE—Department of Electrical Engineering, Federal University of Rio de Janeiro, Cidade Universitária, Centro de Tecnologia, Bloco H, Rio de Janeiro 21941-972, RJ, Brazil; ramon@coep.ufrj.br
* Correspondence: klebersonmlima@outlook.com

Abstract: Brazil has an extensive coastline and Exclusive Economic Zone (EEZ) area, which are of high economic and strategic importance. A Maritime Surveillance System becomes necessary to provide information and data to proper authorities, and target tracking is crucial. This paper focuses on a multitarget tracking application to a large-scale maritime surveillance system. The system is composed of a sensor network distributed over an area of interest. Due to the limited computational capabilities of nodes, the sensors send their tracking data to a central station, which is responsible for gathering and processing information obtained by the distributed components. The local Multitarget Tracking (MTT) algorithm employs a random finite set approach, which adopts a Gaussian mixture Probability Hypothesis Density (PHD) filter. The proposed data fusion scheme, utilized in the central station, consists of an additional step of prune & merge to the original GM PHD filter algorithm, which is the main contribution of this work. Through simulations, this study illustrates the performance of the proposed algorithm with a network composed of two stationary sensors providing the data. This solution yields a better tracking performance when compared to individual trackers, which is attested by the Optimal Subpattern Assignment (OSPA) metric and its location and cardinality components. The presented results illustrate the overall performance improvement attained by the proposed solution. Moreover, they also stress the need to resort to a distributed sensor network to tackle real problems related to extended targets.

Keywords: surveillance systems; tracking; PHD filter

Citation: Lima , K.M.d.; Costa, R.R. Cooperative-PHD Tracking Based on Distributed Sensors for Naval Surveillance Area. *Sensors* **2022**, *22*, 729. https://doi.org/10.3390/s22030729

Academic Editors: Andrzej Stateczny

Received: 13 November 2021
Accepted: 22 December 2021
Published: 19 January 2022

1. Introduction

In 2015, the Brazilian Navy presented the strategic program, so-called the Management System of Amazônia Azul (SisGAAz). Its main objective is to monitor the surveillance of Brazilian jurisdictional waters and the international areas of responsibility for Search and Rescue (SAR) operations [1].

The SCUA (freely translated to the Unified Situational Awareness System) implements the SisGAAz's first step in the gradual development of this Brazilian program. This system is a C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance) that integrates multisensor and multisource data to either evaluate situational awareness or support decision-making activities for the maritime area of interest. In the context of situational awareness, multitarget tracking is a crucial task.

As for the problem of tracking multiple targets with an unknown number of targets, random finite sets have attracted significant attention since they were introduced in the 1990s by Mahler [2]. Since this new approach was introduced, several filters for its implementation have been proposed. In the 2000s, real-data implementations were emerging in areas as diverse as underwater active acoustics and air-to-ground GMTI (Ground Moving Target Indicator) detection and tracking [3].

Although the Probability Hypothesis Density (PHD) filter is already widely known, real-time implementation in systems that demand high performance and availability remains an issue. Over the years, several works have proposed ways to overcome these

concerns. According to Gao et al. [4], although there is no data association problem in this filter, it can hardly be real-time implemented on a serial processor.

Different approaches have tried to overcome processing limitations or speed up processing. Zheng et al. [5] have proposed an efficient event-driven data particle PHD filter for real-time multi-target tracking. In order to reduce processing, the proposed method distinguishes the survival measurements, spontaneous birth measurements, and clutters for weight computation, which are steps with high computational complexity. With the same objective, Li et al. [6] present an algorithmic framework for parallel SMC-PHD filtering based on multiple processing. This algorithm fully parallels all Sequential Monte Carlo (SMC) steps.

Few papers address the parallelization and optimization of the implementation of SMC PHD filters. Li et al. [7] has proposed a novel Cardinality Consensus (CC) scheme to PHD filter for multitarget tracking in decentralized sensor networks with severely constrained communication. This scheme has a parallel filtering-communication mode as it performs communication parallel to filtering operations, which leads to data exchanges that require only a tiny amount of time. Li et al. [8] present details of GM PHD on the CC scheme, which is the first distributed GM filter based on the parallel filtering-communication mode, according to the authors.

In distributed multitargeting tracking applications, the fusion process of data received from sensors is another concern. Li et al. [9] propose a suitable real-time data fusion algorithm called Generalized Covariance Intersection (GCI). They, see [10], present another solution for this issue. This work introduces Fixed-nodes PHD (FN-PHD) fusion, which can achieve similar performance as PHD, but at much lower communication and computational costs. Firstly, radars track targets by a PHD filter. Then the target state closest to the fusion node is chosen, and all states among radars based on GCI are predicted and fused.

The SCUA applies parallelism to solve complex problems concerning data fusion in real time. The Brazilian Navy system has already implemented the PPTS (Pair of Plots in Two Stages) algorithm. This approach is a graph-based method proposed in [11], which is PP (Pair of Plot [12]) based. In 2015, Mello et al. proposed the implementation based on the model Gamma [13]. The proposed solution was the first parallel implementation of the PPTS method, which employed three Gamma models, where two of them exploited the resources of a parallel hardware environment (using MPI protocol and GPU).

This manuscript deals with the application of the GM PHD algorithm in a distributed sensor network, which can be: Radars or cameras installed along the coast or area of interest; radars on patrol or larger ships; and sensors installed in smaller boats. In such a system, unlike those presented in other applications [14–16], each node has its processing and tracking solution. Furthermore, through a communication link, they send their tracking data to a central station responsible for gathering and processing the information obtained by the distributed components.

The main contribution of this work is to propose an additional step of 'prune & merge' to the original GM PHD filter algorithm. This approach includes an additional step expecting to overcome the limitations of sensors and the original PHD algorithm. The proposed scheme presented good results attested by the OSPA metric in the simulations carried out. Furthermore, the target occlusion is overcome, which is one of the main weaknesses of the GM PHD algorithm.

The paper is organized as follows: Section 2 presents related works; Section 3 defines the MTT problem, presents the mathematical framework for RFS-based MTT, and presents the PHD filter definition; Section 4 illustrates a distributed network tracking system application, describes methods, presents the simulated scenarios and modeling, and analyzes the results; and Section 5 exposes the conclusions.

2. Related Works

Many algorithms and methods for tracking targets are available in the literature. Recursive filtering techniques under a Bayesian estimation framework are commonly used to solve the tracking problem using a stochastic approach. These algorithms and methods

include extensions of the well-known Kalman Filter (KF): The Extended Kalman filter (EKF), Interactive Multiple-Model (IMM) filtering, and Variable-Structure IMM (VS-IMM) filtering, besides others filtering techniques, that include grid-based approaches, as well as nonparametric particle techniques [17].

The Kalman filter is the optimal estimator for stochastic white noise linear systems. When dealing with a nonlinear estimation problem, this technique has substantial performance limitations and no convergence guarantee. Extended and unscented KF variants circumvent the nonlinearity issues applying approximations. These algorithms are most effective when facing unimodal problems. The performance of Kalman Filter-based methods significantly decreases when applied to a multimodal problem, even in the most uncomplicated cases. It is precisely the problem considered in a maritime surveillance region: Monlinear models and multimodal distributions.

The Multitarget Tracking (MTT) problem is multimodal. An established way to overcome parametric methods limitations is achieved with the Multi-Hypothesis Kalman Filter (MHKF) or Particle Filter (PF)-based tracking. The objective of MTT is to jointly estimate, at each observation time, the number of targets and their trajectories from sensor data. Even at a conceptual level, MTT is a nontrivial extension of single-target tracking. Indeed, MTT is far more complex in both theory and practice [18]. According to [19], if the assumptions of the Kalman filter or grid-based filters hold, then no other algorithm can out-perform them. However, the assumptions in typical real scenarios do not hold, and approximate techniques are essentially employed. As mentioned by [20], the problems found in these applications favor and justify the use of MTT algorithms. The study by Jinan and Raveendran [21] presents a particle realization applied to MTT using the coordinate turn model to characterize targets dynamics in a two-dimensional maneuver representation. The survey by Wang et al. [22] introduces some advances in algorithms for multitarget tracking.

While the Kalman Filter is concerned with tracking a unique object under a single target problem, there are complementary steps of data association and management that compose multitarget tracking. Conventional MTT techniques typically employ divide-and-conquer approaches that partition a multitarget problem into a family of parallel single-target problems [23]. Given that each KF instance corresponds to a unique target, the data association issue arises as a consequence. If a wrong association occurs (i.e., some observation is associated with a wrong track), the system cannot recover from this error. When dealing with crowded scenes, it is not straightforward to assign an observation to a particular track [20]. One strategy to deal with this problem in Multiple-Hypothesis Tracking (MHT) is to delay data association decisions by keeping multiple hypotheses active until data association ambiguities are solved [24]. Data association is a process in which each measurement is hypothesized to have been originated from a known target, a new target, or clutter. This process is the vital point of MTT, and it becomes more complex with multiple targets, where the tracks compete for measurement [25]. For an introduction to conventional algorithms of data association in MTT, see [26]. In [27], the data association problem is treated as an optimization problem, and two methods are presented, one using a neural network and the other, an evolutionary algorithm.

Although these MTT techniques are well established, they present the problem of data association as an extra and even more complex difficulty. The lack of an optimal global solution for estimating target states and the absence of a Bayesian filter without some intermediate heuristic method exemplify such complexity. Alternatively, seeking to cope with this brittleness, Mahler [2] proposes an MTT algorithm based on the theory of Random Finite Sets (RFS). The indicated approach unifies into a single probabilistic procedure that contains detection, correlation, tracking, and classification. Finite-set Statistics (FISST) is the mathematical framework that supports the RFS approach, see [23,28,29] for a more detailed review.

Notwithstanding all RFS theoretical advantages, the RFS-based recursive Bayes filter, even in single-target problems, is so computationally challenging that it requires an approximate solution to make it implementable. For this purpose, Mahler [30] presents the PHD filter, which gives the number of expected targets when integrated over a region

in target state space. Considering its recursive nature, by propagating PHD's first-order moment statistics, it becomes computationally attractive. In their study of the PHD filter, Vo and Ma [31] propose an analytical solution for the PHD recursion of targets with linear dynamics. Clark and Bell [32] analyze convergence for Sequential Monte Carlo (SMC) approximation using a particle filter, while Clark and Vo [33] consider the Gaussian Mixture (GM) realizations under linear or nonlinear stochastic processes.

There are significant applications in the literature for the mentioned methods that demonstrate their use to multitarget tracking under a high clutter environment as [34–37] for SMC, and [38–42] for GM implementations. Different works [43–45] present the Cardinalized version (CPHD), which is a generalization of the PHD recursion, propagating jointly the posterior intensity and the posterior cardinality distribution, whilst PHD recursion propagates only the intensity. Mahler [46,47] introduces more advances in PHD approximations.

Regarding naval applications, Pace [14] implements SMC and GM PHD filter for a 3D radar applied to realistic three-dimensional aerial and naval scenarios. This work illustrates and compares their performance in detecting, initiating, and terminating tracks with clutter presence. Also related to the same type of application, Do et al. [16] propose a method for online tracking multiple targets for a naval area using a Generalized Labeled Multi-Bernoulli (GLMB) filter.

There is some research on surveillance and distributed sensors based on PHD filtering. However, few papers deal with maritime surveillance and distributed processing. Laneuville and Houssineau [48] consider the problem of multitarget tracking with passive data, obtained by geographically-distributed cameras using a GM-based algorithm. Battistelli et al. [49] treat CPHD applied to distributed multitarget tracking over a sensor network of heterogeneous and geographically-dispersed nodes with sensing, communication, and processing capabilities. Pailhas et al. [15] attempt Multiple-Input Multiple-Output (MIMO) sonar systems for area surveillance, especially to a harbor environment, with a restricted area located close to a traffic area, protecting it from underwater intrusion. Gulmezoglu et al. [50] investigate the use of GM PHD filters for multiple people tracking using a network of radar sensors in an indoor environment. Dias and Bruno [51] introduce a new cooperative particle filter algorithm for tracking emitters using Received-Signal Strength (RSS) measurements considering the communication's cost. They propose two different solutions for reducing communication overhead with a modest degradation in performance. Concerning maritime surveillance, these cited applications have in common the following aspects: A central sensor fusion, stationary sensors, and no local PHD computation.

3. Background

This section provides minimal background to readers unfamiliar with the problem of tracking multiple targets, Bayesian filters, or the RFS approach.

3.1. Problem Statement

To formulate the tracking problem under a RFS framework, it is necessary to define some fundamental issues. According to Bar-Shalom and Li [52], estimation is the process of selecting a point from a continuous space—the "best choice" in line with some criteria; tracking is the state estimation of a moving object based on remote measurements, using one or more sensors at fixed locations or on moving platforms; and filtering is the current state estimation of a dynamic system—the reason for the use of the word "filter" is that the process for obtaining the "best estimate" from noisy data amounts to "filtering out" the noise.

In general, the objective of MTT is to jointly estimate, at each observation time, the number of targets and their trajectories from sensor data. Even at a conceptual level, MTT is a non-trivial extension of single-target tracking [18]. This problem has a two-fold objective: (1) Estimate the random number of targets that are present in the area of interest and (2) estimate the random state vector of each target [49].

Different from traditional MTT vector-based techniques that divide the problem into decoupled single-target problems, RFS tracking uses a state vector collection, treating the elements as random variables as well as the number of elements in that collection itself. Mathematically, the objective is to obtain the posterior multitarget joint probability density functions:

$$p(\mathbf{X}_k, \mathbf{Z}_k), \tag{1}$$

where $\mathbf{X}_k$ and $\mathbf{Z}_k$ are, respectively, the state and observations random finite sets, which are defined in the next section, at the instant k.

3.2. RFS Fundamentals

This section presents concepts and mathematical tools necessary to contextualize RFS-based tracking and Bayesian filtering. For a more detailed review, see [23,29].

A random finite set is a convenient probabilistic model for the representation of multiple stochastic dynamic systems (objects) and sensor measurements. Suppose, at the discrete-time k, there are n_k objects with states $\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,n_k}$ taking values in the state space $\mathcal{X} \subseteq \mathcal{R}^{n_x}$. Both the number of dynamic objects, n_k and their individual states in $\mathcal{X}$ are random and time-varying. The multi-object state at instant k, represented by a finite set.

$$\mathbf{X}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,n_k}\} \in \mathcal{F}(\mathcal{X}), \tag{2}$$

can be modeled as a random finite set on $\mathcal{X}$. Here $\mathcal{F}(\mathcal{X})$ is the set of finite subsets of $\mathcal{X}$ [53].

Analogously, suppose that $\mathbf{Z}_k$ is a measurement set from an observation process, which contains m_k elements reported at time k, then:

$$\mathbf{Z}_k = \{\mathbf{z}_{k,1}, \ldots, \mathbf{z}_{k,m_k}\} \in \mathcal{F}(\mathcal{Z}), \tag{3}$$

is the RFS model on the observation space $\mathcal{Z} \subseteq \mathcal{R}^{n_z}$.

The cardinality (number of elements) and the individual state for a random finite collection are random variables that take values as unordered finite sets. These RFS characteristics provide this technique with the capacity to perform data association automatically and multitarget state estimation jointly.

The cardinality is a random variable and is modeled by a discrete distribution [54], according to:

$$\rho(n) = Pr\{|X| = n\}, \tag{4}$$

where $n \in \mathbb{N}$, and Pr denotes the probability.

A random finite set is a finite-set valued random variable. Thereof, it has the usual probabilistic descriptors of a random variable in the sense of Finite Set Statistics (FISST), such as the Probability Density Function (PDF) and its statistical moments. The PDF of an RFS variable $\mathbf{X}$ is denoted $f(\mathbf{X})$ and uniquely defined by the cardinality $\rho(n)$ and symmetric joint distributions $f_n(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Mathematically, the FISST PDF definition is [54]:

$$f(\mathbf{X}) = f(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) = n! \cdot \rho(n) \cdot f_n(\mathbf{x}_1, \ldots, \mathbf{x}_n). \tag{5}$$

The probability distribution of the cardinality itself is obtained according to:

$$\rho(n) = \frac{1}{n!} \int f(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) \mathrm{d}x_1 \ldots \mathrm{d}x_n. \tag{6}$$

A central concept in FISST is the *set integral*. Given $f(\mathbf{X})$ is a random variables distribution over a random set, the *set integral* of $f(\mathbf{X})$ is defined by:

$$\int f(\mathbf{X}) \delta\mathbf{X} = \sum_{n=0}^{\infty} \frac{1}{n!} \int f(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) \mathrm{d}x_1 \ldots \mathrm{d}x_n. \tag{7}$$

This integral characterizes the sum over the set cardinality, integrating all possible sets according to the number of elements. Besides, the term $1/n!$ considers the permutations of a set of size n.

The *intensity function* (also known as the probability hypothesis density or PHD) of an RFS X is defined as its first statistical moment [54]:

$$D(\mathbf{x}) = \mathbb{E}\{\delta_{\mathbf{X}}(\mathbf{x})\} = \int \delta_{\mathbf{X}}(\mathbf{x}) f(\mathbf{X})\, \delta\mathbf{X}, \tag{8}$$

where $\delta_{\mathbf{w}}(\mathbf{x})$ is the standard Dirac delta function concentrated at $\mathbf{w}$ and $\delta_{\mathbf{X}}(\mathbf{x})$ is the set Dirac delta function, which is defined according to:

$$\delta_{\mathbf{X}}(\mathbf{x}) = \sum_{\mathbf{w}\in\mathbf{X}} \delta_{\mathbf{w}}(\mathbf{x}). \tag{9}$$

It is important to indicate that the PHD is not a probability density. It is uniquely characterized by the following property. Given a region S of single-target state space, the integral is defined by:

$$\hat{\mathfrak{m}} = \int_S D(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{10}$$

and is the expected number of targets in S, i.e., equal to unity. In particular, if S is the entire state space (the multitarget state space), then the integral is the total expected number of targets in the scene [23]. An intuitive interpretation of this function considers it as the density of the expected number of targets occurring at a given point. Furthermore, the mass is seen as the density's integral over the volume of space. Consequently, it is the expected number of targets. Additionally, the peaks of the intensity function indicate locations with a relatively high concentration of the expected number of targets, in other words, locations with a high probability of feature existence [55].

The Poisson RFS is a vital probability distribution type for this study. It is the unique RFS completely specified by its intensity function. Its name comes from the Poisson point process. If the RFS is Poisson, i.e., the number of points is Poisson distributed and the points themselves are IID (Independently and Identically Distributed), then the probability density of $\mathbf{X}$ can be constructed exactly from the PHD [55]. Summarily, its cardinality distribution, multi-object PDF, and intensity function follow, respectively, Equations (11)–(13).

$$\rho(n) = \frac{e^{-\lambda}\lambda^n}{n!},\ n = 0, 1, 2, \ldots \tag{11}$$

where $\lambda > 0$ is the expected cardinality of $\mathbf{X}$.

$$f(\mathbf{X}) = e^{-\lambda} \prod_{\mathbf{x}\in\mathbf{X}} \lambda\, p(\mathbf{x}). \tag{12}$$

$$D(\mathbf{X}) = \lambda\, p(\mathbf{x}). \tag{13}$$

In this context, $p(\mathbf{x})$ refers to the Probability Density Function (PDF) of a random variable $\mathbf{x}$, which can be distributed as Gaussian or Uniform, for example.

3.3. PHD Filter Definition

A simple approach to set-based estimation is to exploit the physical intuition of the first moment of an RFS, known as its PHD or intensity function [55]. This approach is consistent with the multitarget equivalent of expected value—the expectation of an RFS. As stated previously, the PHD function corresponds to a mass density, and the mass corresponds to the expected value of targets in some state-space region $S \subseteq \mathcal{X}$. For this reason, it is possible to link the PHD function and Bayesian framework.

In order to obtain a definition for the PHD function, there are two intuitive ways. The first one would be to obtain the PHD as a mathematical expectation of a point process

density, as previously presented in Equation (8). The second one would be to treat the PHD as the limit of an occupancy grid probability.

As stated by Erdinc et al. [56], the surveillance region can be partitioned into bins, and it is assumed that each bin has the same volume. Furthermore, these bins are sufficiently small so that each bin is potentially occupied by at most one target [56]. Based on this, if m_i denote the centre and $B(m_i)$ the region defined by the boundaries of the ith grid cell, then the expected number of features over the region $S_J = \bigcup_{i \in J} B(\mathbf{x}_i)$ is given by:

$$\mathbb{E}\{|\mathcal{X} \cap S_J|\} = \sum_{i \in J} P^{occ}(B(\mathbf{x}_i)) = \sum_{i \in J} D(\mathbf{x}_i)\lambda(B(\mathbf{x}_i)), \tag{14}$$

where $\mathcal{X} \cap S_J$ is the intersection between the state space $\mathcal{X}$ and surveillance area S_J, $P^{occ}(B(\mathbf{x}_i))$ denote the occupancy probability, $\lambda(B(\mathbf{x}_i))$ is the area of the ith grid cell, and:

$$D(\mathbf{x}_i) = \frac{P^{occ}(B(\mathbf{x}_i))}{\lambda(B(\mathbf{x}_i))} \tag{15}$$

is the intensity function. As the grid cell area tends to zero (or for an infinitesimally small cell), $B(\mathbf{x}_i) \to \mathrm{d}\mathbf{x}_i$, the sum then becomes an integral, and the expected number of features in S becomes:

$$\mathbb{E}\{|\mathcal{X} \cap S_J|\} = \int_S D(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{16}$$

$D(\mathbf{x})$ defines the PHD and it can be interpreted as the occupancy probability density at the point $\mathbf{x}$ [55]. The main objective for MTT RFS-based is propagating the intensity function to estimate targets state recursively based on the Bayesian framework.

Bayes theorem allows computing the probability of an event based on prior knowledge of conditions related to the event and some new evidence. This concept is the basis of Bayesian filters, which estimate the current state of a dynamic system in a probabilistic way. This process consists of two steps: Prediction and update, according to:

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1})\mathrm{d}\mathbf{x}_{k-1}, \tag{17}$$

and

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k \mid \mathbf{x}_k)p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})}{\int p(\mathbf{z}_k \mid \mathbf{x}_k)p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})\mathrm{d}\mathbf{x}_k}, \tag{18}$$

where $\mathbf{x}$ is the state vector, and $\mathbf{z}$ is the observation vector, both in the conventional sense. Equations (17) and (18) are the recursive algorithm presentation of the Bayes Theorem.

Because of the Markov assumption, the posterior probability of the state is only determined based on the prior distribution, i.e., it is conditionally independent of the other earlier states. This premise is the basis for recursive estimation.

The presented filter has only conceptual importance. Because of the integral in Equation (17), it is computationally tractable only for a few discrete cases. However, all Bayesian recursive filters, such as Kalman Filter, are derived from it.

The recursive RFS filter is defined by Equations (19) and (20), similarly to the Bayesian filter one. Given the multitarget state $\mathbf{X}_k$, and measurement $\mathbf{Z}_k$ in the random finite set framework, suppose that at time $k-1$ the posterior FISST PDF of the multi-object state, $f_{k-1}(\mathbf{X}_{k-1}|\mathbf{Z}_{1:k-1})$ is known. Here $\mathbf{Z}_{1:k-1} \equiv \mathbf{Z}_1, \ldots, \mathbf{Z}_{k-1}$ is the sequence of all previous measurements. Then, respectively, the predicted and updated posterior multi-object densities is expressed as follows [53]:

$$f_{k|k-1}(\mathbf{X}_k|\mathbf{Z}_{1:k-1}) = \int \Pi_{k|k-1}(\mathbf{X}_k|\mathbf{X}')f_{k|k-1}(\mathbf{X}'|\mathbf{Z}_{1:k-1})\delta\mathbf{X}', \tag{19}$$

$$f_k(\mathbf{X}_k|\mathbf{Z}_{1:k}) = \frac{\varphi_k(\mathbf{Z}_k|\mathbf{X}_k)f_{k|k-1}(\mathbf{X}_k|\mathbf{Z}_{1:k-1})}{\int \varphi_k(\mathbf{Z}_k|\mathbf{X})f_{k|k-1}(\mathbf{X}|\mathbf{Z}_{1:k-1})\delta\mathbf{X}}, \tag{20}$$

where $\mathbf{X}'$ is the set of targets previously observed, $\varphi_k(\mathbf{Z}_k|\mathbf{X}_k)$ is the observation likelihood function, and $\Pi_{k|k-1}(\mathbf{X}_k|\mathbf{X}')$ is the FISST transitional density (representing the multitarget motion model in the RFS sense). The posterior density is the estimate of a set of points from the multi-object filter at each time step. It is a target state estimate collection (unlabeled and unordered). Similar to the integral present in the Bayes filter, computing the exact multi-object posterior density is numerically intractable because of the set integrals. All the practical algorithms are approximations, and the PHD filter is one of these algorithms.

Mahler [30] derives a recursive Bayes filter for a PHD filter that accounts for multiple sensors, nonconstant probability of detection, Poisson false alarms, as well as the appearance, spawning, and disappearance of targets. The PHD is a best-fit approximation of the multitarget posterior in an information-theoretic sense and propagates the first-order statistical moment of the multitarget posterior.

For a Poisson point process, as described previously, the prediction and update recursive filter equations are [54]:

$$D_{k|k-1}(\mathbf{x}) = b_k(\mathbf{x}) + p_S \int \pi_{k|k-1}(\mathbf{x}|\mathbf{x}')D_{k-1}(\mathbf{x}')\mathrm{d}\mathbf{x}', \tag{21}$$

$$D_k(\mathbf{x}) = D_{k|k-1}(\mathbf{x}) \times \left[1 - p_D + \sum_{\mathbf{z}\in\mathbf{Z}_k} \frac{p_D g_k(\mathbf{z}|\mathbf{x})}{c_k(\mathbf{z}) + \int p_D g_k(\mathbf{z}|\mathbf{x})D_{k|k-1}(\mathbf{x})\mathrm{d}\mathbf{x}}\right], \tag{22}$$

respectively, where $\mathbf{x}$ is the state of a single object (a random vector), $\mathbf{z}$ is a measurement of a single object (a random vector), $b_k(\mathbf{x})$ is the PHD of the births at instant k, p_S is the probability that a target still exists at time k, and $\pi_{k|k-1}$ is the target transition function, D_{k-1} is the prior intensity, D_k is the posterior intensity, p_D is the probability that a target is detected at time k, $c_k(\mathbf{z})$ is the clutter PHD at time k, $g_k(\mathbf{z}|\mathbf{x})$ is the target observation model. For simplicity, the probabilities of detection p_S and p_D presented here are independent of the state.

For computational implementations, see [38,39] for approximations based on Gaussian Mixture and [37,57] for particle methods.

This work implements a Gaussian Mixture realization. The GM is a weighted sum of Gaussians, according to:

$$\mathrm{GM}(x) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(x; m_k^{(i)}, P_k^{(i)}), \tag{23}$$

where $w_k^{(i)}$ is the weight, $m_k^{(i)}$ is the mean, and $P_k^{(i)}$ is the covariance matrix of the ith component from total J_k.

For practical implementations, this algorithm needs complementary steps when compared to the canonical form of the Bayesian sequence. The GM PHD filter suffers from computational problems associated with the increasing number of Gaussian components as time progresses. This issue implies that the number of components in the posterior intensities increases without bound [38].

The prune & merge methods are the two most implemented methods to solve these issues in practical implementations. The first one consists of reaping components of the Gaussian mixture with values below a limit of T. Combined with the pruning operation, the merge method is applied to prevent unbounded growth. This method consists of clustering GM components in a region defined by the Mahalanobis distance L and delimited by threshold U according to:

$$L := \left\{ i \in I = w_k^{(i)} > T | (m_k^{(i)} - m_k^{(j)})^T (P_k^{(i)})^{-1} (m_k^{(i)} - m_k^{(j)}) \leq U \right\}, \tag{24}$$

where j refers to the component with the highest weight in set I. This distance measures how some points are distant from the mean, reflecting the sample dispersion considering the covariance matrix. An intuitive interpretation is that the more the data is correlated, the shorter the distance between them will be, forming a cluster.

The last step is the state extraction, which eliminates GM components with weak weights after the prune & merge step. A better alternative is to select the means of the Gaussians that have weights greater than an *Extraction threshold E* [38]. It is important to note that this step does not interfere with the filter performance.

Algorithm 1 presents the relationship between the filter equations and the execution steps for the GM PHD approximation .

Algorithm 1 $GMPHD(D(\mathbf{x}_{k-1}))$

for Every instant k
do
- Prediction based on Equation (21)
- Update based on Equation (22)
- Prune and Merge based on Equation (24)
- **comment:** Extraction based on:
- Set $\widehat{\mathbf{X}}_k := \varnothing$
- **for** $i \leftarrow 1$ **to** J_k
  - **do** **if** $w_k^{(i)} > E$
    - **then** $\widehat{\mathbf{X}}_k := [\widehat{\mathbf{X}}_k, m_k^{(i)}]$

return $(D(\mathbf{x}_k), \widehat{\mathbf{X}}_k)$

3.4. Optimal Subpattern Assignment (OSPA) Metric

The concept of error between a reference quantity and its estimated value plays a fundamental role in any filtering problem [58]. Unlike the idea of miss-distance in single-object tracking systems, such as the error between actual and estimated state, there is no direct way to measure this error in the multi-object case. As stated by Schuhmacher et al. [58], a satisfactory multi-object miss-distance needs to capture the "difference" between two sets of vectors, namely the reference multi-object state and the estimated multi-object state, in a mathematically consistent yet physically meaningful manner.

In short, the OSPA metric is the "distance" between a set of tracks and the known truths. This metric contains two error measures between those sets: Localization error component (accounts for state estimation) and cardinality error component (a benchmark for the number of missed targets). There is a third component out of the scope of this work called the labeling error component, which accounts for an incorrect assignment.

For two finite sets X and Y with respective m and n cardinalities, for $m \leq n$, the OSPA distance metric is defined according to [58]:

$$\overline{d}_p^{(c)}(X,Y) := \left(\frac{1}{n} \left(\min_{\pi \in \Pi} \sum_{i=1}^{m} (d^{(c)}(x_i, y_{\pi(i)})^p + c^p(n-m) \right) \right)^{\frac{1}{p}}, \tag{25}$$

where $d^{(c)}(x,y) := min(c, d(x,y))$ is the cutoff distance between two elements of X and Y with $c > 0$ being the cutoff parameter, Π_n represents the set of permutations of length m with elements taken from $\{1, 2, \ldots, n\}$, and $1 \leq p < \infty$.

The cutoff parameter c determines the relative weighting of the cardinality error component against the base distance error component. Larger values of c tend to emphasize cardinality errors and vice versa. The order parameter p controls the penalty assigned to "outlier" estimates (which are not close to the ground truth tracks). A higher value of p increases sensitivity to outliers [59].

4. Application for Tracking Maritime Surveillance

The SCUA integrates multisensor and multisource data to evaluate situational awareness and support decision-making activities for the maritime area of interest. Both types of sensors, shipboard or stationary, can belong to the distributed network. Therefore, different devices such as cameras, radars, and sonars can integrate the system.

For small ships, computers (sometimes IoT devices or tablets) have a limited processing capacity. System components exchange data via communication links such as 4G, LoRaWan, or satellite. As a result of processing capacity and bandwidth limitations, the tracker should have a low computational load. Allied to this, it should produce good results in a low clutter environment with a high probability of detection.

Each network member has a local application, which runs SCUA with several features and services, with the tracker being one of those. SCUA has parallel processes and computing (based on OpenCL), depending on the hardware used in the node. The parallelization method is not concerning in this manuscript.

Figure 1 presents the implementation scheme for PHD filtering in local trackers and the data fusion in Central Station (C&C), for the case illustrated in this work. The simulations presented in this section use this same scheme.

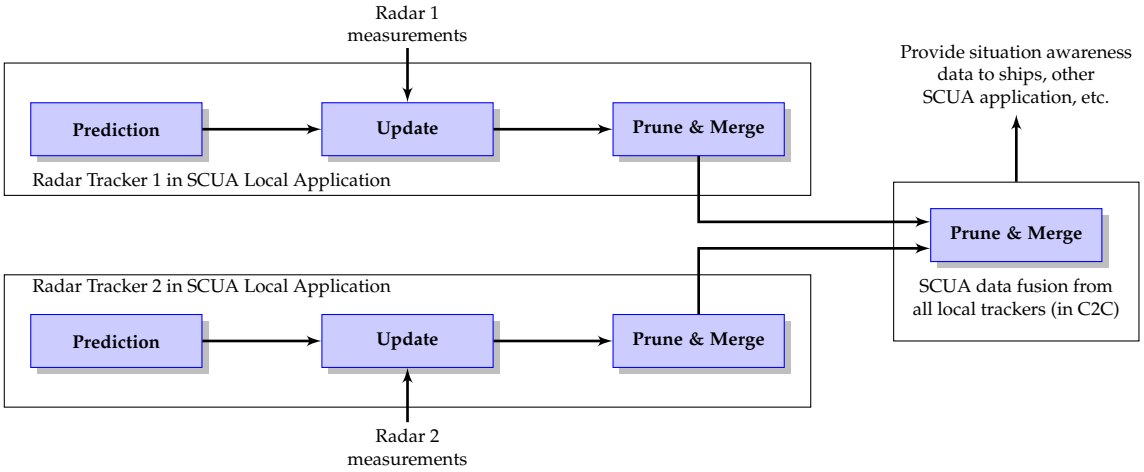

Figure 1. Probability Hypothesis Density (PHD) filtering and data fusion implementation scheme (example of two sensors available).

4.1. Surveillance Scenario

Due to confidentiality issues, only simulations based on Matlab Sensor Fusion® objects are used to illustrate the results of the application. In this simulation scenario, the distributed network is composed of two stationary sensors (2D radars). Both radars stare into the harbor, covering a 90-degree azimuth sector. The sensors are diagonally positioned in the surveillance area in opposite directions, according to Table 1. Each of them has a tracker responsible for monitoring targets in their respective FoV and providing the posterior intensity distribution.

This scenario was defined according to the main aspects: Availability and ease of access to data. There were no more sensors available at the collection and analysis of actual data to prepare the simulation. Furthermore, data obtained from sensors installed on ships could expose sensitive performance data. Given Brazilian regulations, this additional data would require significant effort to consult competent authorities. Despite this data limitation, this scenario proved appropriate for this work.

Table 1. Radars and Central Station in simulation.

Entity	Coordinates [km]	Angular Sector Covered by FoV [rad]
Radar 1	$(7.38, -0.236)$	$[0, -\pi/2]$
Radar 2	$(-4, 9)$	$[-\pi/2, \pi/2]$
Central Station (Tower)	$(0, 0)$	Not Applicable

The effects of latency and bandwidth limitations in communication are not considered in the simulations. Due to the distances involved, the effect of the earth's curvature in the observations is also not considered.

Additionally, there are vessels typically found in the maritime area of interest in simulation. There are five vessels in the harbor within the surveillance sector. Two of them are turning at 20 and 30 knots. The others are traveling with a constant heading of 10, 12, and 6 knots. Table 2 resumes information concerning the initial states of the ships.

Table 2. Ships initial states and type of movement.

Ship	Initial Speed [knot]	Initial Orientation [Deg]	Initial Coordinates [km]	Type of Movement
1	20	130	$(3.15, 7.4)$	Circular with radius 200 m
2	30	120	$(-0.5, 6)$	Circular with radius 400 m
3	10	0	$(3.2, 1.3)$	Constant heading
4	12	0	$(-1.5, 7)$	Constant heading
5	6	90	$(-0.6, 0.7)$	Constant heading

Figure 2 illustrates the surveillance scenario described here.

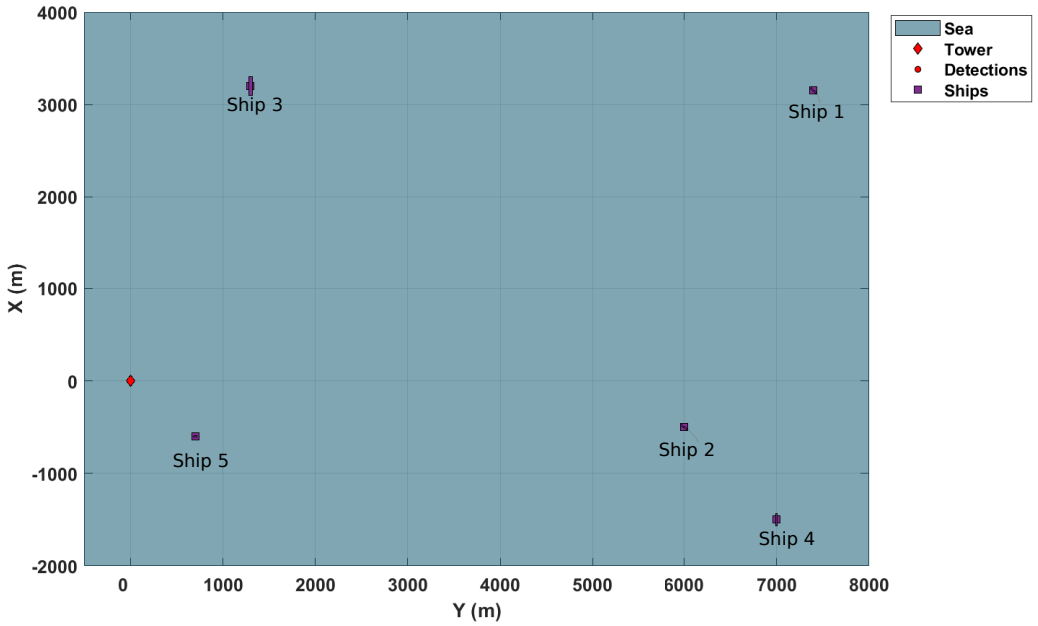

Figure 2. Maritime surveillance scenario.

4.2. Dynamic Modeling

Each of these five vessels in the simulation scenario has a kinematic state vector denoted by:

$$\mathbf{x} = [x\ \dot{x}\ y\ \dot{y}\ \omega]^T, \tag{26}$$

where (x, y) corresponds to position coordinates, $(\dot{x}, \dot{y})$ corresponds to velocities, ω is the constant turn-rate, and T denotes the transpose operation.

Another crucial dynamic model is the one used by the filter itself. Despite the problem of multitarget tracking, the dynamic model of a single target remains fundamental to establish the representative dynamics of each individualized target. Therefore, each target needs a motion model for the filter prediction step. The choice of this model is directly linked to the filter performance.

Tracking targets with coordinated turn motion is highly dependent on the models and algorithms [60]. Since the targets are possibly small, fast, and easy-to-maneuver vessels,

the nonlinear nearly constant turn model is the best choice [61]. For a single target, the state dynamics is given by [38]:

$$\mathbf{x}_k = F(\omega_{k-1})\mathbf{x}_{k-1} + Gw_{k-1} \tag{27}$$

$$\omega_k = \omega_{k-1} + \Delta u_{k-1}, \tag{28}$$

where $\mathbf{x}_k = [x_k\ y_k\ \dot{x}_k\ \dot{y}_k]^T$ corresponds to $2D$ position and velocities coordinates at instant k, ω stands for the turn rate in time k, $w = [w_x\ w_y]^T$ is a zero-mean Gaussian process noise with $\sigma_\omega \in \mathbb{R}^{2\times 2}$ covariance, u is a random variable with zero-mean, σ_u is the covariance,

$$F(\omega) = \begin{bmatrix} 1 & 0 & \frac{\sin\omega\Delta}{\omega} & -\frac{1-\cos\omega\Delta}{\omega} \\ 0 & 1 & \frac{1-\cos\omega\Delta}{\omega} & \frac{\sin\omega\Delta}{\omega} \\ 0 & 0 & \cos\omega\Delta & -\sin\omega\Delta \\ 0 & 0 & \sin\omega\Delta & \cos\omega\Delta \end{bmatrix},$$

$$G = \begin{bmatrix} \frac{\Delta^2}{2} & 0 \\ 0 & \frac{\Delta^2}{2} \\ \Delta & 0 \\ 0 & \Delta \end{bmatrix},$$

and Δ is the sample time.

For each target and stationary sensor, the observation model consists of azimuth and range measurements, according to:

$$z_k = \begin{bmatrix} \sqrt{(x_k - x_s)^2 + (y_k - y_s)^2} \\ \arctan\left(\frac{y_k - y_s}{x_k - x_s}\right) \end{bmatrix} + \epsilon_k \tag{29}$$

where (x_k, y_k) is the target position at instant k, (x_s, y_s) is the sensor position, and ϵ_k is a zero-mean Gaussian process noise with $R_k \in \mathbb{R}^{2\times 2}$ covariance.

4.3. Filter Parameters

The Gaussian Mixture PHD filters used in the simulations are that described in Equations (21) and (22). Each of them enables us to estimate the positions of the ships individually, as described before. Since the predictive model is nonlinear, the filter is the EKF version (see [38] for details about the Jacobians). Table 3 presents the filter parameters used in simulations.

Table 3. Filter parameters used for tracking targets.

Filter Parameter	Value
Sensor maximum range	12 km
Distance resolution noise	25 m
Azimuth resolution noise	0.5°
Probability of Survival (p_S)	0.99
Probability of detection (p_D)	0.9
Sensor field-of-view (FoV)	90°
Clutter density	2×10^{-8}
Prune threshold (T)	10^{-6}
Merge threshold (U)	25
Extraction threshold (E)	0.8
Maximum number of components	1000

For simulation purposes, there is no spawning, and the spontaneous birth RFS is Poisson, i.e., added uniformly inside the sensor's field of view, with a rate of 10^{-5}. The births have a unitary initial weight, and an initial covariance matrix:

$$P_{birth} = \begin{bmatrix} 10^5 & 0 & 0 & 0 & 0 \\ 0 & 10^5 & 0 & 0 & 0 \\ 0 & 0 & 10^5 & 0 & 0 \\ 0 & 0 & 0 & 10^5 & 0 \\ 0 & 0 & 0 & 0 & 1000 \end{bmatrix}.$$

Finally, the clutter RFS follows the uniform Poisson model over the sensor FoV, the surveillance region being $[-\pi/2,\ \pi/2]$ rad $\times$ $[0,\ 12]$ km.

Since each radar independently estimates the coordinates of the targets, this information needs to be gathered. Then, the simulated central processing merges these distributions, obtaining the final solution in a distributed and cooperative way. Similarly to [62], the merging process is:

$$\text{Cooperative Tracking} = merge(\left\{D(\mathbf{x})^1, D(\mathbf{x})^2, \ldots, D(\mathbf{x})^R\right\}), \tag{30}$$

where $D(\mathbf{x})^i$ is the intensity estimated by ith distributed sensor, and R is the number of distributed sensors. Merge operation is a clustering technique. For RFS applications, the algorithms based on Mahalanobis distance are the most common method, which is detailed in [38].

Figure 3 resumes, in a block diagram, all the steps of the simulation setup (surveillance scenario, models, and filter parameters).

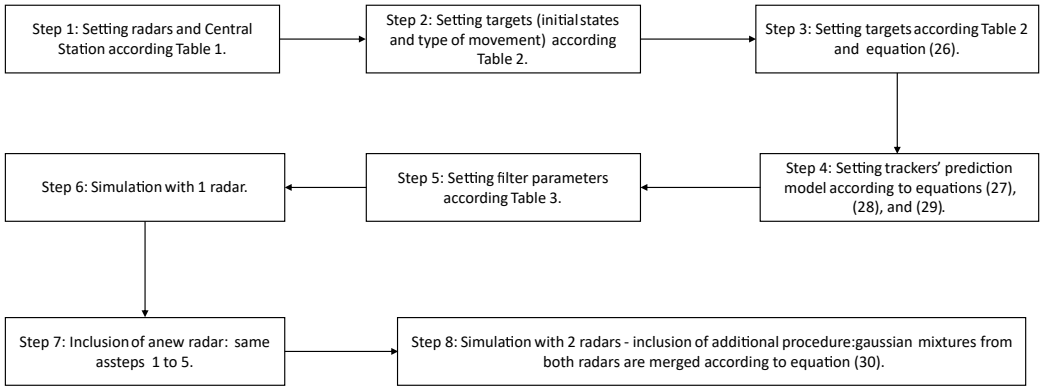

Figure 3. Block diagram of the proposed scheme.

4.4. Results and Discussions

Based on OSPA metrics shown in Figure 10, the results presented in the simulations are satisfactory. Initially, to illustrate the performance achieved, the simulated tracking system uses only one radar, whose FoV is represented by the region delimited by red lines (as in the other figures). The tracker exhibits good results, i.e., it can track the targets observed by the sensor. Figures 4 and 5 present these results.

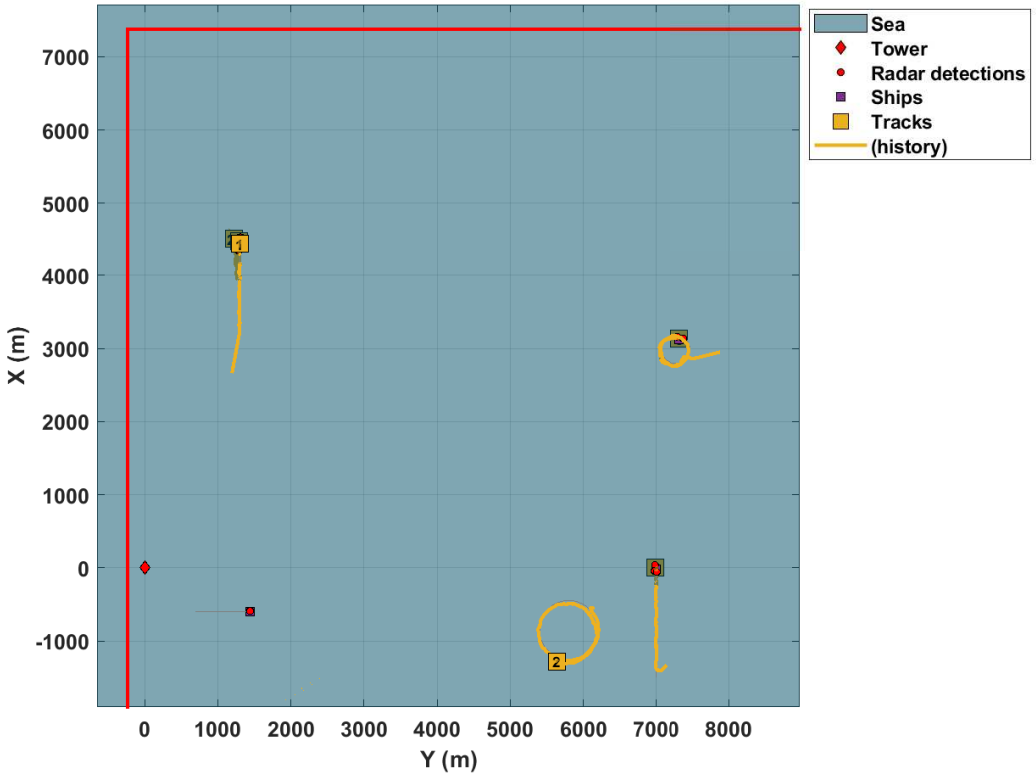

Figure 4. Tracking with one sensor.

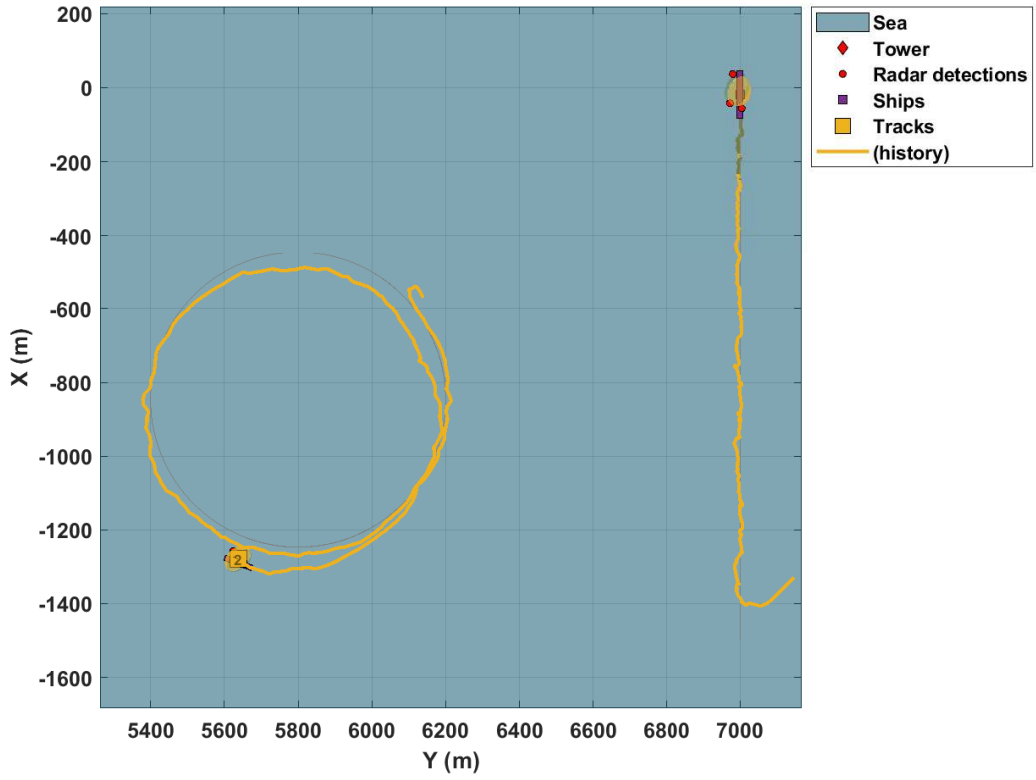

Figure 5. Tracking of two targets with different trajectories.

As stated before, these figures demonstrate the tracking system's performance with just one sensor. In addition, these figures show that the ships are tracked regardless of whether the movement performed is linear or circular, despite an initial transitory. It is noteworthy that the vessels present significant differences in their speed and movements. At the same time, one navigates at 12 knots, the other spins at 30 knots.

Figure 5 is an enlarged view of two targets with different movements, one linear and the other circular. This figure reinforces the tracking capacity of the system in different conditions of movement of the target. Furthermore, it attests the good choice of the dynamic turn-rate model for the targets. This choice is an essential factor for the success of the tracking system, as stated by [60].

Despite the successful choice for the prediction model, the figure illustrates an existing problem in real-life systems called target occlusion. Figure 6 shows this weakness when using only one radar. The larger vessel occludes the smaller one. Therefore, it is not detected by the tracking system. Given that the application aims to combat illegal activities usually carried out in the region by small vessels, it is essential to use sensors distributed throughout the surveillance area. This fact is further reinforced by the large dimensions of the surveillance area.

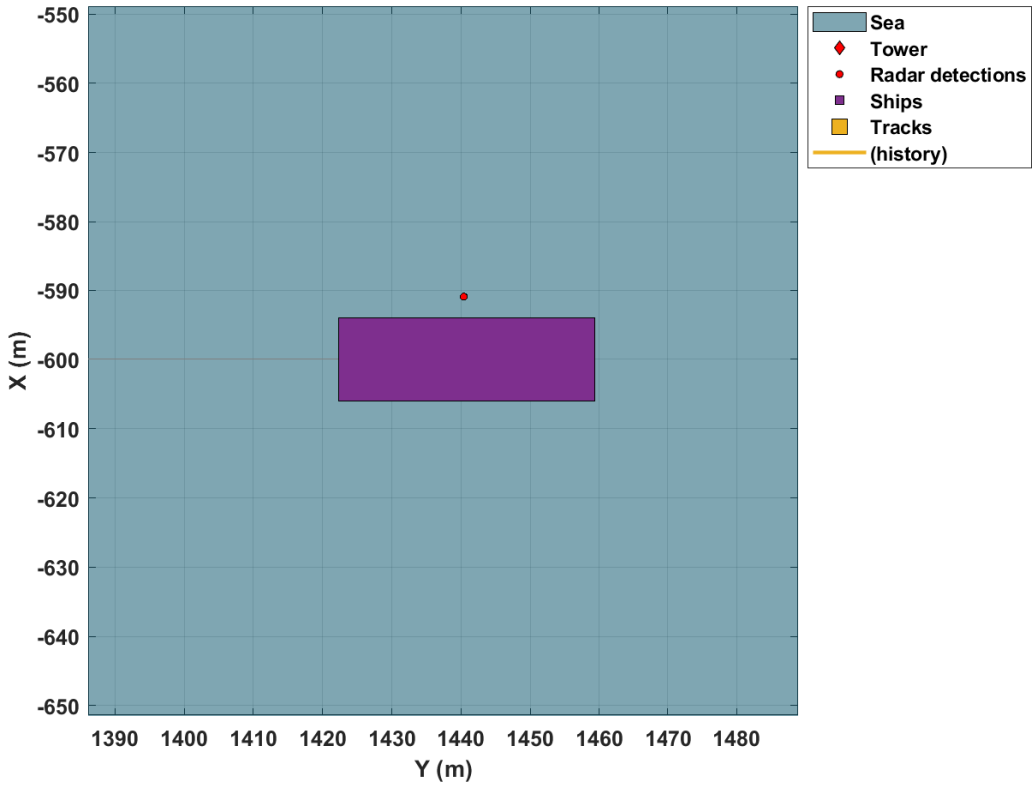

Figure 6. Target in occlusion.

Inserting a second radar prevents targets occlusion, as illustrated by Figure 7.

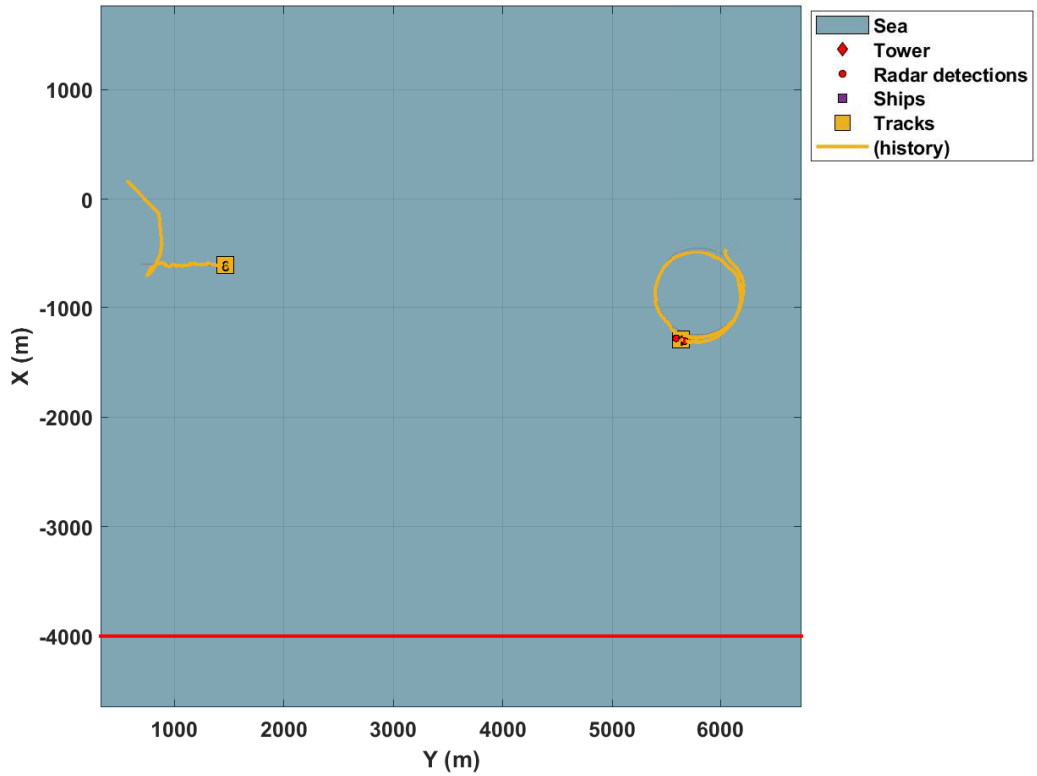

Figure 7. Second radar in the surveillance area.

Figures 8 and 9 present the results with combined trackers, i.e., two radars.

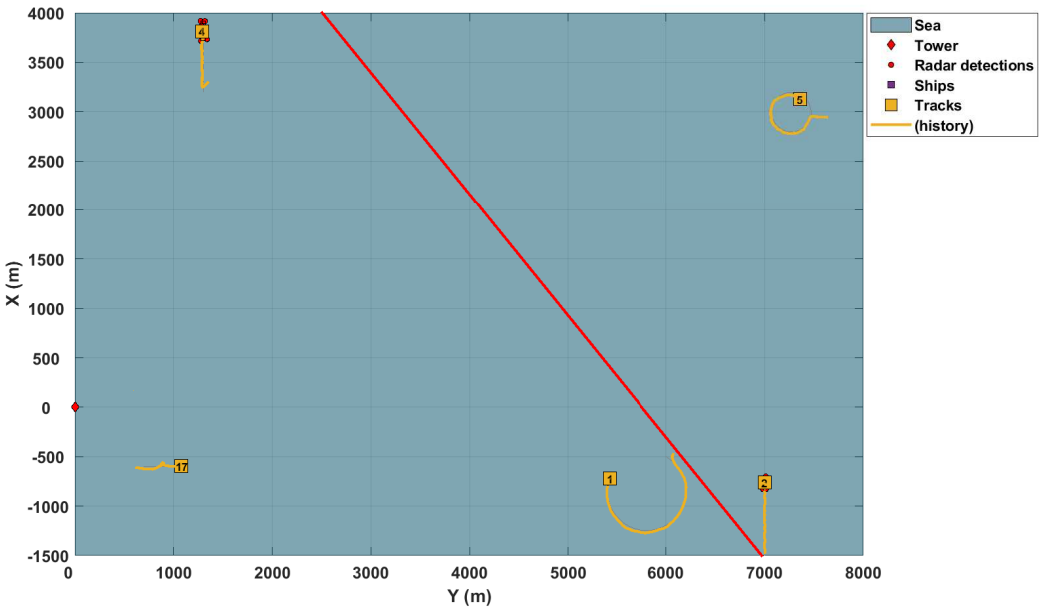

Figure 8. Complete tracking with merged PHDs.

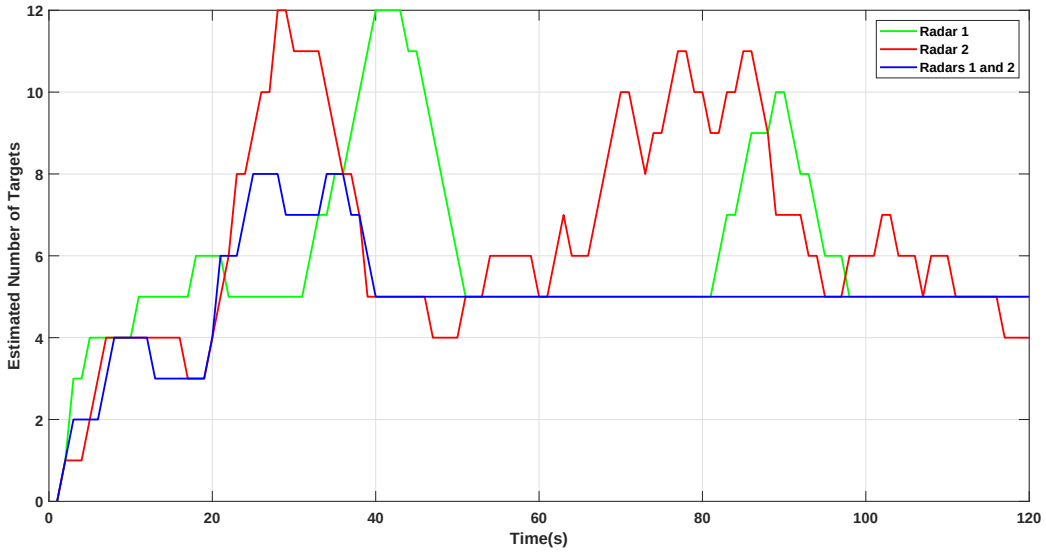

Figure 9. The number of estimated targets by individual trackers, and merged.

It is possible to observe that combining information from each tracker allows a faster convergence to the number of actual targets. In addition, it is more stable since the increase in data sources (sensors) in the system avoids targets occlusion compared to using only one radar.

Finally, Figure 10 presents the OSPA metric and its components, which corroborates the analyses based on the previous figures. The merged tracker is faster and more stable, as can be observed in the plots, in particular after a period of 40 s. The lower the metric value, the better the filter's performance. Furthermore, the combined filter with data from both radars shows better results for localization and cardinality OSPA components. The location component is related to the position error, as can be seen in Figure 10, after 40 s, the value tends to zero. Similarly, at the same instant, the cardinality component becomes zero.

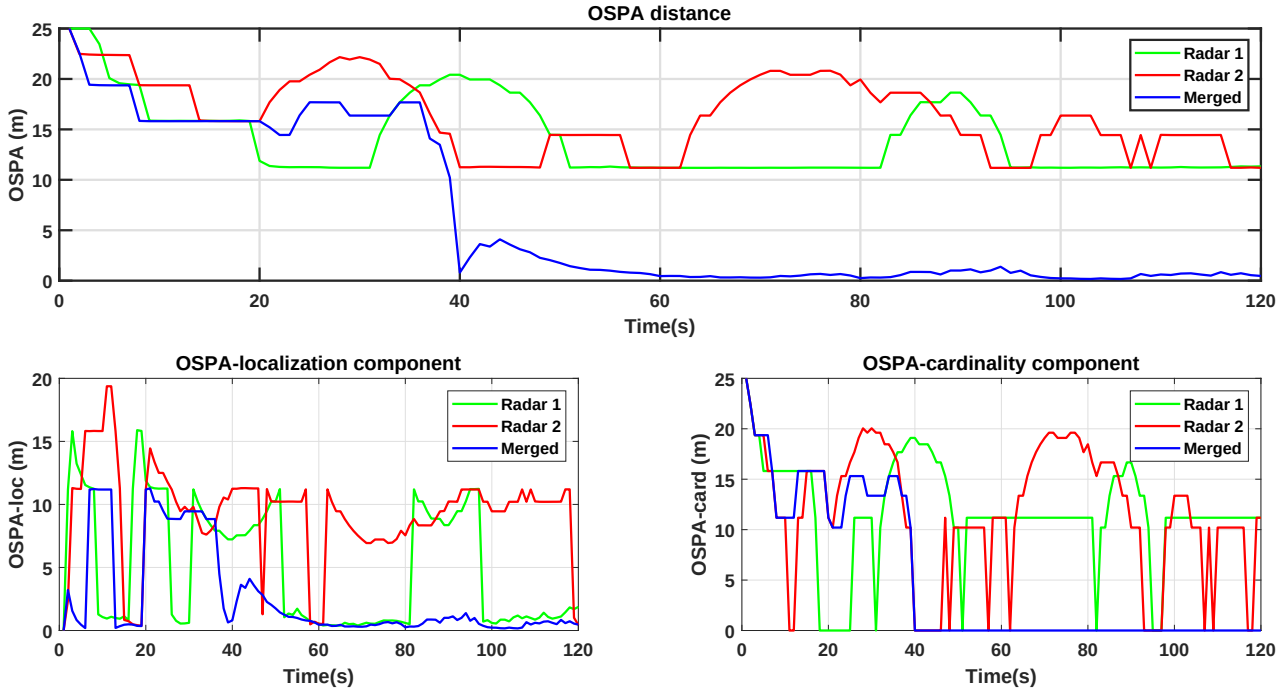

Figure 10. OSPA metrics.

The simulations illustrate the problem of target occlusion when they present different geometries and dimensions. Inserting a new sensor and the consequent increase in the sensor network prevents targets occlusion and improves the overall performance of the tracking system. These results reinforce the statement by Vo and Ma [38]: The efficiency and simplicity in the implementation of the Gaussian mixture PHD recursion also suggest a possible application to tracking in sensor networks.

The algorithm has performance limitations already demonstrated in other researches. The proposed architecture presents good results when applying an additional step to the original algorithm, which is the main contribution of this work: A low-complexity data fusion scheme. Additionally, this proposed scheme presents promising results, especially when dealing with the target occlusion problem.

5. Conclusions

This paper introduces a Maritime Surveillance System for the Brazilian Coast. The system has a complex network of distributed sensors and processing stations. The demanding requirements for its operation are one of the motivations for the present work. In this context, this work presents an algorithm to track an unknown number of marine targets from multiple sensors in a large-scale surveillance area. A merged version of the Gaussian Mixture PHD filter is applied to a realistic multitarget tracking scenario. The proposed filter shows promising results through simulations, even for targets with different sizes, movements, and speeds. The proposed data fusion scheme improves the overall performance of the tracking system and overcomes targets' occlusion. This fact further evidences, for the application, the need for a network of distributed sensors, given the system's mission and the extension of the area of interest. Additionally, it is possible to conclude that increasing the covered area using sensors of a smaller capacity also contributes to the economic aspect of this modular undertaking.

For future work, one of the focuses would be investigating different RFS-based filters, the capabilities of the extended algorithms, and different sensor types such as stationary passive sonars or shipboard radars. Moreover, the computational models used for parallelism and multiprocessing should be investigated. In particular, those based on Gamma models, once SCUA data fusion applications already use this approach. These future works will attest, corroborate the empirical results obtained in the simulations, and contribute to using these algorithms in a new computational paradigm.

Author Contributions: Conceptualization, K.M.d.L. and R.R.C.; methodology, K.M.d.L.; software, K.M.d.L.; validation, K.M.d.L. and R.R.C.; writing–original draft preparation, K.M.d.L.; writing–review and editing, K.M.d.L. and R.R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Plano Estratégico da Marinha. Strategic Programs. Available online: https://www.marinha.mil.br/programas-estrategicos (accessed on 2 April 2020).
2. Mahler, R.P.S. Random-set approach to data fusion. In Proceedings of the SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing, Orlando, FL, USA, 4–8 April 1994. [CrossRef]
3. Maher, R. A survey of PHD filter and CPHD filter implementations. In Proceedings of the Defense and Security Symposium, Orlando, FL, USA, 9–13 April 2007. [CrossRef]
4. Gao, L.; Tang, X.; Wei, P. Real-Time Implementation of Particle-PHD Filter Based on GPU. In Proceedings of the 2014 IEEE 17th International Conference on Computational Science and Engineering, Chengdu, China, 19–21 December 2014. [CrossRef]
5. Zheng, Y.; Shi, Z.; Lu, R.; Hong, S.; Shen, X. An Efficient Data-Driven Particle PHD Filter for Multitarget Tracking. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2318–2326. [CrossRef]
6. Li, T.; Sun, S.; Bolić, M.; Corchado, J.M. Algorithm design for parallel implementation of the SMC-PHD filter. *Signal Process.* **2016**, *119*, 115–127. [CrossRef]

7. Li, T.; Hlawatsch, F.; Djuric, P.M. Cardinality-Consensus-Based PHD Filtering for Distributed Multitarget Tracking. *IEEE Signal Process. Lett.* **2019**, *26*, 49–53. [CrossRef]
8. Li, T.; Mallick, M.; Pan, Q. A Parallel Filtering-Communication-Based Cardinality Consensus Approach for Real-Time Distributed PHD Filtering. *IEEE Sens. J.* **2020**, *20*, 13824–13832. [CrossRef]
9. Li, G.; Yi, W.; Kong, L. A real-time fusion algorithm for asynchronous sensor system with PHD filter. In Proceedings of the 2017 International Conference on Control, Automation and Information Sciences (ICCAIS), Chiang Mai, Thailand, 31 October–1 November 2017. [CrossRef]
10. Li, G.; Yi, W.; Jiang, M.; Kong, L. Distributed fusion with PHD filter for multi-target tracking in asynchronous radar system. In Proceedings of the 2017 IEEE Radar Conference (RadarConf), Seattle, WA, USA, 8–12 May 2017. [CrossRef]
11. Barbosa, R. Fusão de Alvos Utilizando Grafos em Ambientes de Múltiplos Sensores. Ph.D. Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, 2012.
12. Livernet, F.; Cuilliere, O. Tracking based on graph theory applied to pairs of plots. In Proceedings of the 2010 IEEE Radar Conference, Arlington, VA, USA, 10–14 May 2010; pp. 90–95.
13. Mello, R.R.; de Almeida, R.H.P.; Franca, F.M.; Paillard, G.A. A Parallel Implementation of Data Fusion Algorithm Using Gamma. In Proceedings of the 2015 International Symposium on Computer Architecture and High Performance Computing Workshop (SBAC-PADW), Florianopolis, Brazil, 18–21 October 2015. [CrossRef]
14. Pace, M. Comparison of PHD based filters for the tracking of 3D aerial and naval scenarios. In Proceedings of the 2010 IEEE Radar Conference, Arlington, VA, USA, 10–14 May 2010. [CrossRef]
15. Pailhas, Y.; Houssineau, J.; Petillot, Y.R.; Clark, D.E. Tracking with MIMO sonar systems: Applications to harbour surveillance. *IET Radar Sonar Nav.* **2017**, *11*, 629–639. [CrossRef]
16. Do, C.T.; Nguyen, T.T.D.; Liu, W. Tracking Multiple Marine Ships via Multiple Sensors with Unknown Backgrounds. *Sensors* **2019**, *19*, 5025. [CrossRef]
17. Carthel, C.; Coraluppi, S.; Grignan, P. Multisensor tracking and fusion for maritime surveillance. In Proceedings of the 2007 10th International Conference on Information Fusion, Québec, QC, Canada, 9–12 July 2007.
18. Vo, B.; Mallick, M.; Bar-shalom, Y.; Coraluppi, S.; Osborne, R., III; Mahler, R.; Vo, B. Multitarget Tracking. *Wiley Encycl. Electr. Electron. Eng.* **2015**, *2015*, 1–15.
19. Arulampalam, M.; Maskell, S.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **2002**, *50*, 174–188. [CrossRef]
20. Bazzani, L.; Bloisi, D.D.; Murino, V. A Comparison of Multi Hypothesis Kalman Filter and Particle Filter for Multi-Target Tracking. CVPR 2009. 2009. Available online: https://www.researchgate.net/publication/228373867 (accessed on 3 April 2020).
21. Jinan, R.; Raveendran, T. Particle Filters for Multiple Target Tracking. *Procedia Technol.* **2016**, *24*, 980–987. [CrossRef]
22. Wang, X.; Li, T.; Sun, S.; Corchado, J. A Survey of Recent Advances in Particle Filters and Remaining Challenges for Multitarget Tracking. *Sensors* **2017**, *17*, 2707. [CrossRef]
23. Mahler, R.P.S. *Statistical Multisource Multitarget Information Fusion*; Artech House, Inc.: Boston, MA, USA, 2007.
24. Kim, C.; Li, F.; Ciptadi, A.; Rehg, J.M. Multiple Hypothesis Tracking Revisited. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4696–4704.
25. Wang, L.Y.; Zhao, Y.J.; Wang, W.X. Study on Theory of Multi-target Tracking and Data Association Algorithms in Phased Array Radar. In Proceedings of the 2006 6th International Conference on ITS Telecommunications, Chengdu, China, 21–23 June 2006; pp. 1232–1235. [CrossRef]
26. Liggins, M., II; Hall, D.; Llinas, J., Eds. *Handbook of Multisensor Data Fusion: Theory and Practice*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2009.
27. Zaveri, M.A.; Merchant, S.N.; Desai, U.B. Data Association for Multiple Target Tracking: An Optimization Approach. In *Neural Information Processing*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 187–192. [CrossRef]
28. Mahler, R. "Statistics 101" for multisensor, multitarget data fusion. *IEEE Aerosp. Electron. Syst. Mag.* **2004**, *19*, 53–64. [CrossRef]
29. Goodman, I.R.; Mahler, R.P.S.; Nguyen, H.T. *Mathematics of Data Fusion*; Springer: Dordrecht, The Netherlands, 1997. [CrossRef]
30. Mahler, R. Multitarget bayes filtering via first-order multitarget moments. *IEEE Trans. Aerosp. Electron. Syst.* **2003**, *39*, 1152–1178. [CrossRef]
31. Vo, B.N.; Ma, W.K. A closed-form solution for the probability hypothesis density filter. In Proceedings of the 2005 7th International Conference on Information Fusion, Philadelphia, PA, USA, 25–28 July 2005. [CrossRef]
32. Clark, D.; Bell, J. Convergence results for the particle PHD filter. *IEEE Trans. Signal Process.* **2006**, *54*, 2652–2661. [CrossRef]
33. Clark, D.; Vo, B.N. Convergence Analysis of the Gaussian Mixture PHD Filter. *IEEE Trans. Signal Process.* **2007**, *55*, 1204–1212. [CrossRef]
34. Clark, D.; Vo, B.T.; Vo, B.N. Gaussian Particle Implementations of Probability Hypothesis Density Filters. In Proceedings of the 2007 IEEE Aerospace Conference, Big Sky, MT, USA, 3–10 March 2007. [CrossRef]
35. Clark, D.; Bell, J. Multi-target state estimation and track continuity for the particle PHD filter. *IEEE Trans. Aerosp. Electron. Syst.* **2007**, *43*, 1441–1453. [CrossRef]
36. Lin, L.; Bar-Shalom, Y.; Kirubarajan, T. Track labeling and PHD filter for multitarget tracking. *IEEE Trans. Aerosp. Electron. Syst.* **2006**, *42*, 778–795. [CrossRef]

37. Zajic, T.; Mahler, R.P.S. Particle-systems implementation of the PHD multitarget-tracking filter. In Proceedings of the Signal Processing, Sensor Fusion, and Target Recognition XII, Orlando, FL, USA, 21 April 2003. [CrossRef]
38. VO, B.; MA, W. The Gaussian Mixture Probability Hypothesis Density Filter. *IEEE Trans. Signal Process.* **2006**, *54*, 4091–4104. [CrossRef]
39. Clark, D.; Panta, K.; Vo, B.N. The GM-PHD Filter Multiple Target Tracker. In Proceedings of the 9th International Conference on Information Fusion, Florence, Italy, 10–13 July 2006. [CrossRef]
40. Clark, D.; Godsill, S. Group Target Tracking with the Gaussian Mixture Probability Hypothesis Density Filter. In Proceedings of the 2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information, Melbourne, Australia, 3–6 December 2007. [CrossRef]
41. Hao, Y.; Meng, F.; Zhou, W.; Sun, F.; Hu, A. Gaussian mixture probability hypothesis density filter algorithm for multi-target tracking. In Proceedings of the 2009 IEEE International Conference on Communications Technology and Applications, Beijing, China, 16–18 October 2009. [CrossRef]
42. Baisa, N.L.; Wallace, A. Development of a N-type GM-PHD filter for multiple target, multiple type visual tracking. *J. Vis. Commun. Image Represent.* **2019**, *59*, 257–271. [CrossRef]
43. Mahler, R. PHD filters of higher order in target number. *IEEE Trans. Aerosp. Electron. Syst.* **2007**, *43*, 1523–1543. [CrossRef]
44. Nannuru, S.; Blouin, S.; Coates, M.; Rabbat, M. Multisensor CPHD filter. *IEEE Trans. Aerosp. Electron. Syst.* **2016**, *52*, 1834–1854. [CrossRef]
45. Vo, B.T.; Vo, B.N.; Cantoni, A. Analytic Implementations of the Cardinalized Probability Hypothesis Density Filter. *IEEE Trans. Signal Process.* **2007**, *55*, 3553 – 3567. [CrossRef]
46. Mahler, R. A brief survey of advances in random-set fusion. In Proceedings of the 2015 International Conference on Control, Automation and Information Sciences, Changshu, China, 29–31 October 2015. [CrossRef]
47. Mahler, R. Exact Closed-Form Multitarget Bayes Filters. *Sensors* **2019**, *19*, 2818. [CrossRef]
48. Laneuville, D.; Houssineau, J. Passive multi target tracking with GM-PHD filter. In Proceedings of the 2010 13th International Conference on Information Fusion, Edinburgh, UK, 26–29 July 2010. [CrossRef]
49. Battistelli, G.; Chisci, L.; Fantacci, C.; Farina, A.; Graziano, A. Consensus CPHD Filter for Distributed Multitarget Tracking. *IEEE J. Sel. Top. Signal Process.* **2013**, *7*, 508–520. [CrossRef]
50. Gulmezoglu, B.; Guldogan, M.B.; Gezici, S. Multiperson Tracking with a Network of Ultrawideband Radar Sensors Based on Gaussian Mixture PHD Filters. *IEEE Sens. J.* **2015**, *15*, 2227–2237. [CrossRef]
51. Dias, S.S.; Bruno, M.G.S. Cooperative Target Tracking Using Decentralized Particle Filtering and RSS Sensors. *IEEE Trans. Signal Process.* **2013**, *61*, 3632–3646. [CrossRef]
52. Bar-Shalom, Y.; Li, X.R.; Kirubarajan, T. *Estimation with Applications to Tracking and Navigation*; John Wiley Sons: Hoboken, NJ, USA, 2001.
53. Ristic, B.; Beard, M.; Fantacci, C. An Overview of Particle Methods for Random Finite Set Models. *Inf. Fusion* **2016**, *31*, 110–126. [CrossRef]
54. Ristic, B. *Particle Filters for Random Set Models*; Springer: New York, NY, USA, 2013. [CrossRef]
55. Mullane, J.; Vo, B.N.; Adams, M.; Vo, B.T. *Random Finite Sets for Robot Mapping and SLAM: New Concepts in Autonomous Robotic Map Representations*; Springer Tracts in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2011; Volume 72.
56. Erdinc, O.; Willett, P.; Bar-Shalom, Y. The Bin-Occupancy Filter and Its Connection to the PHD Filters. *IEEE Trans. Signal Process.* **2009**, *57*, 4232–4246. [CrossRef]
57. Sidenbladh, H. Multi-target particle filtering for the probability hypothesis density. In Proceedings of the Sixth International Conference of Information Fusion, Cairns, Australia, 8–11 July 2003. [CrossRef]
58. Schuhmacher, D.; Vo, B.T.; Vo, B.N. A Consistent Metric for Performance Evaluation of Multi-Object Filters. *IEEE Trans. Signal Process.* **2008**, *56*, 3447–3457. [CrossRef]
59. Ristic, B.; Vo, B.N.; Clark, D.; Vo, B.T. A Metric for Performance Evaluation of Multi-Target Tracking Algorithms. *IEEE Trans. Signal Process.* **2011**, *59*, 3452–3457. [CrossRef]
60. Yuan, X.; Lian, F.; Han, C. Models and Algorithms for Tracking Target with Coordinated Turn Motion. *Math. Probl. Eng.* **2014**, *2014*, 1–10. [CrossRef]
61. Li, X.R.; Jilkov, V. Survey of maneuvering target tracking . Part I: Dynamic models. *IEEE Trans. Aerosp. Electron. Syst.* **2003**, *39*, 1333–1364. [CrossRef]
62. Cano, P.; Ruiz-del Solar, J. Robust Tracking of Soccer Robots Using Random Finite Sets. *IEEE Intell. Syst.* **2017**, *32*, 22–29. [CrossRef]