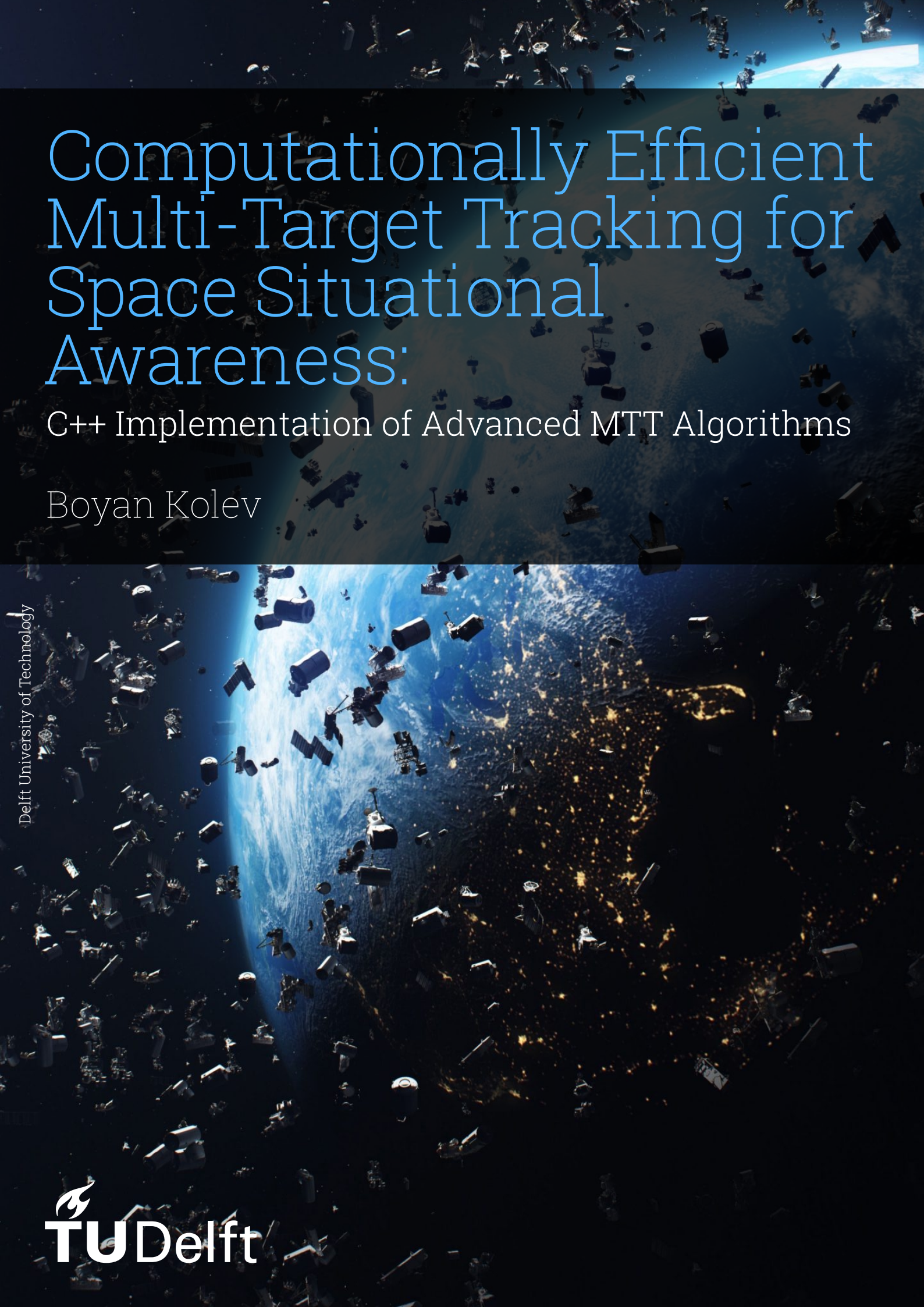
Computationally Efficient Multi-Target Tracking for Space Situational Awareness:

C++ Implementation of Advanced MTT Algorithms

Boyan Kolev



Delft University of Technology

TU Delft

Computationally Efficient Multi-Target Tracking for Space Situational Awareness:

C++ Implementation of Advanced MTT Algorithms

by

Boyan Kolev

Supervisor:	Dr. S. Gehly
Project Duration:	January 2025 - October 2025
Faculty:	Faculty of Aerospace Engineering, Delft
Student Track:	MSc Aerospace Engineering - Space Track
Student Number:	5243416

Cover: An illustration of space debris. (Photo: EUSPA)

TU Delft

Acknowledgments

I would like to express my deepest gratitude to all those who have supported me throughout the journey of writing this thesis.

First and foremost, I am sincerely thankful to my supervisor, Dr. Steve Gehly, for his invaluable guidance, encouragement, and patience. His expertise, insightful feedback, and constant support have been instrumental in shaping this research. I am particularly thankful for the weekly meetings that were held, during which I was able to have my questions answered and after which I always felt more confident in the next steps to undertake.

I also owe my heartfelt thanks to my parents, Maria Koleva and Ivo Kolev, whose unconditional love, support, and belief in me have been the foundation of my academic and personal growth. Their sacrifices, encouragement, and constant reminders to persevere have been a source of strength throughout this journey.

To my friends, thank you for your unwavering support, understanding, and motivation. Your encouragement, companionship, and the moments of laughter we shared helped me navigate the challenges and maintain balance during the most demanding times of this thesis.

Finally, I am grateful to everyone who, in one way or another, has influenced my academic journey. This thesis is not only the result of my efforts but also a reflection of the support, guidance, and encouragement I have received from all of you.

Boyan Kolev

Abstract

This research aims to advance the field of Space Situational Awareness (SSA) by optimizing multi-sensor multi-target tracking (MSMTT) algorithms within the Random Finite Set (RFS) framework. The project focuses on addressing the computational challenges posed by the increasing number of Resident Space Objects (RSOs) through the development of an efficient, scalable estimator capable of handling dense object environments and ambiguous data. Key components of this research include a detailed evaluation of existing RFS methods showing promise in the field of SSA, such as Labelled Multi-Bernoulli methods (LMB). By integrating and enhancing these techniques, the project aims to improve tracking accuracy and computational efficiency by implementing the filters using C++ source code, thereby supporting both current space operations and future mission planning.

Contents

Abstract ii

Nomenclature iv

List of Figures xi

List of Tables xii

List of Listings xiii

1 Introduction 1

2 Literature Review 3
2.1 Challenges in Space Situational Awareness . . . 3
2.2 Approaches Overview . . . 4
2.2.1 Data Fusion . . . 13
2.2.2 Assessment Metrics . . . 13

3 Research Question 15

4 Methodology 16
4.1 Finite Set Statistics . . . 16
4.2 Probability Hypothesis Density . . . 17
4.2.1 Theory . . . 18
4.2.2 Numerical Example . . . 21
4.3 Labelled Multi Bernoulli . . . 26
4.3.1 Theory . . . 26
4.3.2 Implementation Approach . . . 31
4.4 Poisson Labelled Multi Bernoulli . . . 36
4.5 Supporting Concepts and Methods . . . 41
4.5.1 Gaussian Mixture . . . 41
4.5.2 Kalman and Unscented Kalman Filter: Prediction & Update . . . 43
4.5.3 K-Shortest Paths and Murty's Algorithm for Hypothesis Management . . . 46
4.5.4 Model Description . . . 50
4.5.5 Parameter Tables . . . 53

5 Results & Discussion 55
5.1 PHD/LMB comparison . . . 55
5.1.1 Linear Model . . . 55
5.1.2 Non-Linear Model . . . 57
5.2 LMB/PLMB comparison . . . 59
5.3 Time Performance . . . 61
5.4 Implication for SSA . . . 62

6 Conclusion 64

References 66

A Literature Selection Methodology 72

B Code Structures and Pseudo-Code 75

C Original Research Proposal Plan 85
C.1 Experimental Set-Up Ideal . . . 88

Nomenclature

Abbreviations

Abbreviation	Definition
AA	Arithmetic Average
AR	Admissible Region
CAR	Constrained Admissible Region
CBMeMBer	Cardinality Balanced Multi-target Multi-Bernoulli
CPHD	Cardinalised Probability Hypothesis Density
DISP	Distinguishable and Independent Stochastic Populations
GA	Geometric Average
GEO	Geosynchronous Orbit
GLMB	Generalised Labelled Multi Bernoulli
GM	Gaussian Mixture
GTO	Geosynchronous Transfer Orbit
HISP	Hypothesised and Independent Stochastic Populations
JM CBMeMBer	Jump Markov Cardinality Balanced Multi-target Multi-Bernoulli
JPDA	Joint Probabilistic Data Association
LEO	Low Earth Orbit
MEO	Medium Earth Orbit
MeMBer	Multi-target Multi-Bernoulli
MHT	Multi-Hypothesis Tracking
MM CBMeMBer	Multi Model Cardinality Balanced Multi-target Multi-Bernoulli
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
(MS)MTT	(Multi-Sensor) Multi-Target Tracking
OR	Orbital Regime
OSPA	Optimal Subpattern Assignment
PAR	Probabilistic Admissible Region
PHD	Probability Hypothesis Density
PGFL	Probability Generating Functional
PMBM	Poisson Multi-Bernoulli Mixture
RFS	Random Finite Set
RSO	Resident Space Object
SMC CBMeMBer	Sequential Monte-Carlo Cardinality Balanced Multi-target Multi-Bernoulli
SSA	Space Situational Awareness

Symbols

Please note that all symbols are defined in the main text, and the following nomenclature table has been assembled as accurately as possible.

Symbol	Definition	Unit
a	Semi-major axis length of the orbit	$[\mathrm{m}]$
a	Assignment of a measurement to a track (can be $\emptyset$ or O_i)	[–]
a_{max}	Maximum admissible semi-major axis defining the upper bound of the admissible region	$[\mathrm{m}]$
A_i	Optimal assignment ($i = 1$), suboptimal assignments ($i > 1$)	[–]
$B_{k\|k-1}(\boldsymbol{\zeta})$	RFS of targets spawned at time k from a target with previous state $\boldsymbol{\zeta}$	[–]
BT	Birth Threshold	[–]
C, C_{ij}, C^n	Cost matrix	[–]
$d^2(y_{k+1})$	Squared Mahalanobis distance of the measurement from the predicted track	[–]
$d_p^{(c)}(X, Y)$	p-th order OSPA metric with cut-off c between RFSs X and Y	[m]
e	Orbital eccentricity	[–]
e_{max}	Maximum admissible orbital eccentricity	[–]
f_{ij}	Likelihood of associating measurement O_i with track T_j	[–]
$f_{k\|k-1}(\mathbf{x}_k \mid \boldsymbol{\zeta})$	Single-target transition density: probability that a target at state $\boldsymbol{\zeta}$ at time $k-1$ moves to $\mathbf{x}_k$ at time k	[–]
$f_{k+1\|k+1}(X\|Z^{k+1})$	PDF that targets have state set X, given measurement history Z^{k+1}	[–]
G	Gating threshold	[–]
$G_X[h]$	PGFL of a multi-target RFS X, representing the functional form of its multi-target distribution	[–]
$g(\cdot)$	Measurement model	[–]
$g_k(\mathbf{z}_k \mid \mathbf{x}_k)$	Single-target measurement likelihood: probability density of observing $\mathbf{z}_k$ given target state $\mathbf{x}_k$	[–]
H	Measurement matrix	[–]
H_i	Hypothesis i	[–]
$h(\mathbf{x})$	Test function used in the PGFL of an RFS, mapping single-target state $\mathbf{x} \in \mathbb{X}$ to $[0, 1]$	[–]
h^X	Multi-object exponential of real-valued function h; $\prod_{\mathbf{x}\in X} h(\mathbf{x})$ for non-empty X, 1 otherwise	[–]
$\mathbf{h}$	Specific angular momentum vector $\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}}$	$[\mathrm{m}^2\,\mathrm{s}^{-1}]$
$\mathrm{Hyp}(n)$	Hypothesis n, i.e. one possible set of measurement-to-track assignments	[–]
I	Set of track labels	[–]
$j^{(n)}(\dots)$	nth-order Janossy density	[–]

Symbol	Definition	Unit
J_{max}	Maximum number of components in the multi-target filter	[–]
K_k	RFS of clutter received by the sensor at time k	[–]
$L(a)$	Likelihood of assignment a	[–]
ℓ	Label attached to a single target, defined as an ordered pair (k, i)	[–]
M	Number of most significant hypotheses retained after truncation	[–]
m_{birth}	Birth state mean vector	[m or m/s]
N_D	Number of detections within the validation gate	[–]
N_{max}	Maximum allowed number of hypotheses/components after capping	[–]
$\mathcal{N}^*$	Set of validated measurements within the gate	[–]
O_i	Measurement i	[–]
$p_{D,k}(\mathbf{x}_k)$	Probability of detection given target state $\mathbf{x}_k$ at time k	[–]
$p_f(\mathbf{z}^{(i)}; \bar{\mathbf{z}}^{(j)}, P_{zz}^{(j)})$	Measurement likelihood function evaluated for $\mathbf{z}^{(i)}$ given $\bar{\mathbf{z}}^{(j)}$ and $P_{zz}^{(j)}$	[–]
$p_{S,k}(\boldsymbol{\zeta})$	Probability that a target survives to time k given previous state $\boldsymbol{\zeta}$	[–]
$P_{zz}^{(j)}$	Predicted measurement covariance matrix for track j	[–]
P_D	Probability of detection	[–]
P_S	Probability of target survival	[–]
P_{birth}	Birth state covariance matrix	[m^2] or rad²
pdf_c	Clutter intensity (false alarm density)	[–]
p_G	Gating probability; probability that a valid measurement lies within the validation gate	[–]
$q_{D,G}(\mathbf{x}, \ell)$	Joint missed-detection probability accounting for detection and gating, $q_{D,G} = 1 - p_D(\mathbf{x}, \ell)\, p_G$	[–]
R	Measurement noise covariance matrix	[–]
r	Existence probability of a Bernoulli component	[–]
$\boldsymbol{r}$	Position vector of the RSO in the ECI frame	[m]
$\dot{\mathbf{r}}$	Velocity vector of the RSO in the ECI frame	[$\text{m}\,\text{s}^{-1}$]
S	Measurement covariance matrix	[–]
T_j	Track j	[–]
T_p	Pruning threshold for multi-target components	[–]
T_s	Sampling time of the motion model	[s]
U	Merging threshold for multi-target components	[–]
$v(\mathbf{x})$	Probability Hypothesis Density (PHD) of the corresponding unlabelled RFS	[–]
w	Weight	[–]
w_{birth}	Weight of a newborn target	[–]

Symbol	Definition	Unit
X	• Bernoulli random variable, or • Target state set	[–]
y_k	Measurement obtained at time step k	[–]
$\hat{y}_k$	Predicted measurement for a track at time k	[–]
$\tilde{y}$	Measurement residual vector	[–]
$\mathbf{z}^{(i)}$	Measurement vector corresponding to measurement i	[–]
$\bar{\mathbf{z}}^{(j)}$	Predicted measurement vector for track j	[–]
$\hat{x}_{k+1\|k}$	Predicted state of the track at time $k+1$ based on the previous estimate at time k	[–]
$\boldsymbol{a}_p$	Perturbing accelerations	$[\mathrm{m\,s^{-2}}]$
$\emptyset$	Null assignment (no measurement associated with a track)	[–]
α_i	Element of the discrete label space $\mathbb{L}$, uniquely identifying target i	[–]
$\mathbb{C}$	Indexing space of the δ-GLMB, defined as $\mathbb{F}(\mathbb{L}) \times \Xi$	[–]
$\mathbb{L}$	Discrete label space, $\mathbb{L} = \{\alpha_i : i \in \mathbb{N}\}$	[–]
$\mathbb{L}_k$	Label space of new targets born at time t_k, $\mathbb{L}_k = \{k\} \times \mathbb{N}$	[–]
$\mathbb{L}_{0:k}$	Label space of all targets up to time t_k (new births and survivals), defined recursively as $\mathbb{L}_{0:k} = \mathbb{L}_{k-1} \cup \mathbb{L}_k$	[–]
$\mathbb{N}$	Set of natural numbers	[–]
$\mathbb{R}$	Set of Real Numbers	[–]
$\mathbf{x}$	Single-target state vector	[–]
$\hat{\mathbf{x}}$	Estimated target state corresponding to the maximum of $p^{(\ell)}(\mathbf{x})$	[–]
$\mathbb{X}$	Single-target state space	[–]
$\mathbb{X} \times \mathbb{L}$	Joint state–label space defining the domain of a labelled RFS	[–]
$\hat{X}$	Set of confirmed tracks satisfying track-management thresholds	[–]
$1_Y(X)$	Inclusion function; equals 1 if $X \subseteq Y$, 0 otherwise	[–]
$\|X\|$	Cardinality (number of elements) of the set X	[–]
Y	Unlabelled version of labelled RFS X	[–]
$\mathcal{F}(\cdot)$	Collection of all finite subsets of the space $(\cdot)$	[–]
$\mathcal{F}(\mathbb{L}_+^{(i)})$	Collection of all finite subsets of the predicted label space $\mathbb{L}_+^{(i)}$	[–]
$\mathcal{G}^{(i)}$	Group (i) of tracks processed in parallel update	[–]
$\mathcal{L}(X)$	Set of labels of a labelled RFS X	[–]
$\mathcal{L}((\mathbf{x}, \ell))$	Label transformation mapping $(\mathbf{x}, \ell) \in \mathbb{X} \times \mathbb{L}$ to ℓ	[–]
$\mathcal{V}(X)$	Unlabelled version of labelled RFS X, i.e. $\mathcal{V}(X) = Y$	[–]

Symbol	Definition	Unit
$\mathcal{V}((\mathbf{x},\ell))$	State transformation mapping $(\mathbf{x},\ell)\in\mathbb{X}\times\mathbb{L}$ to $\mathbf{x}$	[–]
$\mathbf{B}_2$	Process noise input matrix for non-linear motion model	[–]
$\mathbf{Q}$	Process noise covariance matrix	[–]
$\mathbf{R}$	Measurement noise covariance matrix	[–]
α	Spread parameter controlling the dispersion of sigma points around the mean; typical range $10^{-3}\leq\alpha\leq 1$	[–]
α	Primary scaling factor for sigma points in the Unscented Kalman Filter	[–]
$\beta(\cdot)$	Belief mass function	[–]
β	Parameter to incorporate prior knowledge of the distribution; for Gaussian, $\beta=2$ is optimal	[–]
β	Parameter to incorporate prior knowledge of the distribution in the UKF; optimal for Gaussian is 2	[–]
$\beta_{k\|k-1}(\cdot\|\boldsymbol{\zeta})$	Intensity of the RFS $B_{k\|k-1}(\boldsymbol{\zeta})$ spawned at time k by a target with previous state $\boldsymbol{\zeta}$	[–]
β_{birth}	Birth tuning parameter	[–]
Γ_k	RFS of spontaneous birth at time k	[–]
$\gamma_k(\cdot)$	Intensity of the birth RFS Γ_k at time k	[–]
$\Delta(X)$	Distinct-label indicator; equals 1 if all labels in labelled RFS X are unique	[–]
$\Delta(\widetilde{X}^{(i)})$	Distinct-label indicator ensuring all labels in group (i) are unique	[–]
$\delta_Y(X)$	Generalised Kronecker delta function; equals 1 if $X=Y$, 0 otherwise	[–]
$\delta_I(L)$	Generalised Kronecker delta function; equals 1 if $L=I$, 0 otherwise	[–]
ε	Specific orbital energy of the RSO	[$\mathrm{J\,kg^{-1}}$]
$\eta^{(\theta)}_{Z^{(i)}}(\ell)$	Normalising constant for updated density $p^{(\theta)}(\mathbf{x},\ell\mid Z^{(i)})$	[–]
Θ_{I_+}	Space of association mappings $\theta: I_+\to\{0,1,\ldots,\|Z^{(i)}\|\}$ satisfying the one-to-one constraint	[–]
θ	Measurement-to-track association mapping for label set I_+	[–]
κ	Secondary scaling parameter for sigma points; often set to 0 or $3-n$	[–]
κ	Secondary scaling factor for sigma points in the UKF	[–]
$\kappa_k(\cdot)$	Intensity of clutter RFS K_k at time k	[–]
$\kappa(\mathbf{z})$	Intensity of clutter (Poisson false alarm process) at measurement $\mathbf{z}$	[–]
λ	False alarm density (average number of false measurements per unit volume)	[–]

Symbol	Definition	Unit
λ	Scaling factor for sigma point spread, defined as $\lambda = \alpha^2(n+\kappa) - n$	[–]
μ	Gravitational parameter	$[\mathrm{m}^3/\mathrm{s}^2]$
μ_E	Gravitational parameter of the Earth	$[\mathrm{m}^3\,\mathrm{s}^{-2}]$
$\nu(\cdot)$	Probability density function	[–]
Ξ	Discrete space representing the history of track-to-measurement associations	[–]
ξ	Realisation of the discrete association history Ξ	[–]
$\pi(X)$	PDF of a RFS X	[–]
$\pi(X)$	Multi-target posterior density of the labelled RFS X	[–]
$\rho(\cdot)$	Cardinality of a RFS	[–]
$\rho(n)$	Cardinality distribution of the δ-GLMB RFS	[–]
σ_w	Process noise standard deviation	$[\mathrm{m}/\mathrm{s}^2]$
σ_{vel}	Process noise standard deviation (velocity)	$[\mathrm{m}\,\mathrm{s}^{-1}]$
σ_ω	Process noise standard deviation (turn rate)	$[\mathrm{rad}\,\mathrm{s}^{-1}]$
σ_v	Measurement noise standard deviation	$[\mathrm{m}]$
σ_α	Measurement noise standard deviation (bearing)	$[\mathrm{rad}]$
σ_r	Measurement noise standard deviation (range)	$[\mathrm{m}]$
ϑ_u	Upper threshold on existence probability for track confirmation	[–]
ϑ_l	Lower threshold on existence probability for track maintenance	[–]
χ	Sigma Points	[–]
$\psi_{Z^{(i)}}(\mathbf{x}, \ell; \theta)$	Measurement likelihood function for track ℓ under association θ and measurements $Z^{(i)}$	[–]
Ω	Arbitrary space	[–]

List of Figures

2.1 Distribution of trackable space objects as a function of altitude in Earth orbit on February 27, 2013 courtesy of B.Jones et al. [35] . . . 4
2.2 Overview of different MTT methods (HISP conceptually related to RFS but not part of the family) . . . 5
2.3 Visual representation of a validation gate for a track and its associated measurements. $T_1 \equiv$ estimated mean state of target 1, dashed line represents an equiprobability curve (i.e. Gate) . . . 6
2.4 Visual representation of the GNN with data associations conflicts . . . 6
2.5 Assignment problem of measurements $O_i, \forall i \in \{1, 2\}$ to tracks $T_i, \forall i \in \{1, 2\}$ in a two measurements two tracks scenario with overlapping gates and measurements . . . 8
2.6 Bernoulli and Multi-Bernoulli Random Finite Set (RFS) representations in state space. . . 11

4.1 PHD simplified schematic . . . 18
4.2 Ground truth trajectories in xy plane . . . 22
4.3 PHD filter estimates and true tracks in xy coordinates for single run simulation with random seed = 0. . . . 22
4.4 Time series of the PHD filter estimates, ground truth tracks and measurements in cartesian coordinates for single run simulation with random seed = 0. . . . 23
4.5 Estimated vs. true cardinality for the PHD Filter for single run simulation with random seed = 0. . . . 23
4.6 OSPA metrics with parameters $c = 100$, $p = 1$ for single run simulation with timestep (k) = 1s and random seed = 0. . . . 24
4.7 Average estimated cardinality vs time over 1000 MC runs . . . 25
4.8 Average OSPA metrics with parameters $c = 100$, $p = 1$, over 1000 MC runs with time step (k) = 1s. . . . 25
4.9 LMB filter schematic overview [65] . . . 32
4.10 Example of partitioning into groups, with five tracks (red squares) and nine measurements (black stars), courtesy of [65] . . . 34
4.11 Illustrative example of the CAR/PAR approach for initialising target tacks, curtsey of Cament et al. [13]. . . . 38
4.12 PLMB filter schematic overview . . . 39
4.13 One-dimensional Gaussian mixture example showing three weighted Gaussian components (dashed lines) and their resulting mixture density (solid line). . . . 42
4.14 Two-dimensional Gaussian mixture example showing three components with elliptical covariance contours at 1σ and 2σ levels. Each component is represented by its mean (coloured dot) and covariance structure (dashed ellipses). . . . 42
4.15 Visual representation of the accuracy difference between the EKF and the UKF, figure courtesy of MATHWORKS[1] . . . 45
4.16 Boundary effects on sigma point representation . . . 46
4.17 Murty branching tree for simplified cost matrix example. . . . 49
4.18 Ground Truth Tracks for the Coordinated-Turn (CT) model . . . 52

5.1 Ground truth, LMB estimates and measurements for the linear model without gating (left) and with gating (right) . . . 55
5.2 OSPA metric Comparison between LMB estimates (left) and PHD estimates (right) . . . 56
5.3 Visual representation of the LMB tracks over time in Cartesian coordinates for LIN-model 56
5.4 LMB (left) cardinalities compared to PHD (right) cardinalities estimates . . . 57
5.5 LMB filter estimated cardinalities vs. true cardinalities showcasing the effect of gating on the results on the non-linear model . . . 57

5.6 LMB filter estimated OSPA analysis showcasing the effect of gating on the results . . . 58
5.7 Visual representation of the LMB tracks over time in Cartesian coordinates for CT-model with gating on, for the CT-model . . . 58
5.8 PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with constant birth threshold and active measurement gating . . . 59
5.9 PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with constant birth threshold and no measurement gating . . . 59
5.10 PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and no track existence probability bounding . . . 60
5.11 PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and track existence probability bounding . . . 60
5.12 PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the CT-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and no track existence probability bounding for the CT-model . . . 61
5.13 PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the CT-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and track existence probability bounding for the CT-model . . . 61

A.1 PRISMA selection flowchart . . . 73
A.2 Distribution of the selected sources for the thesis by publication year . . . 74

C.1 Thesis draft timeline for weekly hours distribution . . . 86

List of Tables

2.1 Murty's Assignment Algorithm Results for Optimal/Suboptimal Assignments (5-best assignments) . . . 7
2.2 Hypothesis Table MHT, num(T) = 2, num(O) = 2 . . . 8

4.1 Symbol legend for full PHD recursion . . . 19
4.2 Simplifying Assumptions . . . 19
4.4 Parameters of the Gaussian Mixture used for the illustrative examples. . . . 43
4.5 Motion Model Parameters (Linear / Non-Linear) . . . 53
4.6 Target Birth Parameters . . . 54
4.7 Algorithmic Parameters (Merging, Pruning, Capping, Detection, etc.) . . . 54
4.8 Unscented Kalman Filter (UKF) Parameters for Non-Linear Models . . . 54

5.1 Average CPU time performance of the LMB/PLMB filters for the non-linear coordinated turn model over 10 runs . . . 62

C.2 Main objectives description per phase and category - Literature (L), Code(C), Writing(W), Admin(A) - non-critical tasks marked with a star ★ . . . 87
C.3 Available computing resources during the project . . . 88

Listings

A.1 Search Query Scopus/IEEE 72
A.2 Search Query Scopus/IEEE 72

1

Introduction

Over the past several decades, the expansion of space activities has transformed near-Earth space into a critical global resource for various technological and scientific applications, including communication, navigation, and Earth observation. More recently, this growth has accelerated due to the NewSpace movement [50], which emphasises smaller, cheaper, and faster approaches to satellite development, particularly through the use of smallsats and CubeSats, as well as the deployment of large commercial constellations such as Starlink. This increase in space utilisation has been accompanied by significant growth in the number of both functional satellites and non-functional debris - collectively known as Resident Space Objects (RSOs), and includes over 34.000 objects larger than $10\,\mathrm{cm}$, more than 900.000 objects between $1\,\mathrm{cm}$ and $10\,\mathrm{cm}$, and over $120\,\mathrm{m}$illion objects ranging from $1\,\mathrm{mm}$ to $1\,\mathrm{cm}$ [32]. Furthermore, this increasing trend is expected to continue as more satellites are launched into space [55], which can create additional debris and increase the probability of collision. In turn, this debris poses a significant threat to the currently operational space assets and increases cost and risk for future missions [33]. Such collisions can trigger a self-sustaining process of further fragmentation, commonly referred to as the Kessler syndrome [37], or collisional cascade, where each collision generates debris that increases the likelihood of subsequent collisions.

To address these issues we turn to the field of (Multi-Sensor) Multi-Target Tracking (MS)MTT, which is a problem encountered in a wide range of disciplines including sonar/radar, computer vision, cell biology, vehicle perception [10, 17, 1, 31, 54] and of most significant interest to this study Space Situational Awareness (SSA) [19]. SSA is fundamentally about maintaining an up-to-date catalogue that accurately identifies the locations of objects and the associated uncertainties. This effort is crucial in mitigating the risks posed by orbital debris and involves characterising the debris field to effectively plan future missions and monitor current assets [28]. The common goal of multi-target tracking is estimating the trajectories of an unknown and time-varying number of objects based on sensor data, which is in addition corrupted by phenomena including observation noise, false alarms, missed detections, and data association uncertainty [4]. Moreover, for SSA additional challenges arise from not fully accounting/understanding the nature of the physical perturbations affecting the orbital trajectory, the limited number of sensors relative to the large number of RSOs, and the non-linearity of both the dynamical propagation and the measurement relationships [19, 28]. As such, combining these factors results in a highly demanding computational task, whose complexity grows exponentially with the number of tracked objects and measurements.

A wide range of approaches to solving MTT problems exist, including Bayesian and non-Bayesian estimation techniques, implemented in recursive, batch, box-particle, or fixed-lag processing schemes [59], with the most popular being the Joint Probabilistic Data Association (JPDA) filter, Multiple Hypothesis Tracking (MHT), and the most recently developed Random Finite Set (RFS) framework [13, 65, 75]. MHT and JPDA are formulated via data association, referring to the partitioning of the measurements into potential tracks and false alarms, followed by filtering, which refers to estimating the state of the target given its measurement history [75]. On the other hand, the distinguishing feature of the RFS approach is that it directly seeks both optimal and suboptimal estimates of the multitarget state,

i.e. instead of focusing on the data association problem [75], RFS methods directly try to estimate the overall state of all the targets at once. This shift in perspective simplifies the formulation of multitarget tracking and naturally accommodates scenarios in which the number of targets changes over time, such as when new targets appear or existing ones disappear. In addition, both the MHT and JPDA approaches have known limitations when it comes to SSA applications, respectively upscaling the problem becomes very computationally expensive when it comes to MHT [28, 65, 75], and the conventional JPDA filter can be error-prone, computationally expensive and requires modifications to accommodate for unknown and time-varying number of targets [70, 48]. As such it is not surprising that in recent years the trend has been evolving towards the use of the RFS framework approach with filters such as the (Cardinalised) Probability Hypothesis Density PHD/CPHD [28, 87], the Multi-Bernoulli filters [81, 46, 41, 44] , the Generalised Labelled Multi Bernoulli filters (GLMB) family [60, 82, 4, 35] and the relatively newer Poisson Multi Bernoulli Mixture (PMBM) and its derivatives [70, 86, 13, 26, 72]. Moreover, the Hypothesised and Independent Stochastic Populations (HISP) [19] filter, which follows from stochastic population frameworks, has recently been employed in SSA and exhibits similarities to the RFS framework. Therefore, this project aims to investigate which filters within the RFS framework can be optimised and up-scaled into an efficient estimator. This is achieved by evaluating performance based on accuracy and computational efficiency within the context of SSA, to facilitate the operations of current space assets and the planning of future space missions.

The structure of this report is as follows. First, an introduction to the problem in chapter 2 summarises the challenges, current methodologies, research, and gaps in the literature. The research questions then follow in chapter 3. Moreover, in chapter 4 a description of the methodology is provided, including more detailed theoretical background, implementation schemes and preliminary results. This is followed by a discussion and presentation of the obtained results in chapter 5. Finally, the conclusion on the progress so far and future plans for the project are provided in chapter 6.

2

Literature Review

Multi-Target Tracking is a challenge encountered across various fields, primarily aimed at tracking an unknown and time-varying number of objects based on sensor data corrupted by observation noise [4]. MTT can be tackled using different methods, the suitability of which depends on the specific context in which the problem is being addressed. This literature review is structured as follows: first, an in-depth discussion of the challenges in Space Situational Awareness is provided in section 2.1. This is followed by an overview of the main MTT methods, outlined in Figure 2.2 and discussed in section 2.2. Finally, an overview of the metrics used for assessment is presented in subsection 2.2.2.

2.1. Challenges in Space Situational Awareness

False alarms, misdetections and data association uncertainties are the common challenges in MTT [82], which are further accompanied by SSA specific complications including (i) the large number of RSOs that continuously increases, (ii) sparse number of (imperfect) sensors with limited fields of view and inaccuracies, (iii) the non-linearity of both the dynamical propagation and the measurement relationships [28, 13, 35].

Regarding statement (i), it should be noted that the increase in the number of objects naturally results in a greater number of potential events, i.e., observation and interaction instances such as detections, target births/deaths, and measurement-to-track associations that the estimator must handle. However, the primary indicator of complexity is not limited to the number of objects, but also to their density in both spatial and temporal dimensions, resulting in concepts such as clutter density and birth/death rates [28, 4]. In practice, higher target density also increases ambiguity in data association, which often becomes the most critical challenge in regions where many objects are closely spaced.

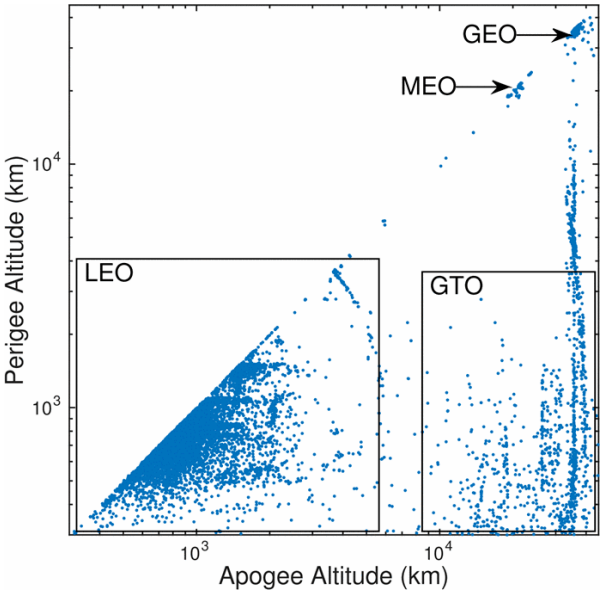


Figure 2.1: Distribution of trackable space objects as a function of altitude in Earth orbit on February 27, 2013 courtesy of B.Jones et al. [35]

On the other hand, the sensor's imperfections mentioned in statement (ii) are a source for the false detections and misdetections in any MTT field, but the SSA-specific issue comes from the sparse amount of sensors and their limited fields of view. This results in long gaps between measurements, increasing propagation uncertainties, complicating the generation of new tracks in the filter, intensifying the impact of the non-linear dynamics and affecting the validity of assumptions such as the Gaussian state uncertainty [28]. Moreover, the length of these gaps is dependent on the orbital regime of the RSO objects. For instance, objects in Low Earth Orbit (LEO) move quickly across the sky, which means they are only visible to a given sensor for short periods of time, creating gaps when they are below the horizon. In addition, because LEO objects complete many orbits per day, if each object is only observed once per night, there can be a large number of unobserved orbits between measurements. Both effects make measurements-to-track correlations more complex compared to higher-altitude regimes such as GEO [35], as well as increasing the difficulty for sensor scheduling. Moreover, the non-linearity of the dynamical propagation mentioned in statement (iii) is also dependent on the orbital regime (OR) of the RSOs with the magnitude of the perturbing accelerations[1] used to model the translational and rotational dynamics varying with the OR. Accurately modelling these perturbations, while maintaining a balance between sufficient accuracy and computational efficiency, is one of the challenges in SSA. Objects along the diagonal in Figure 2.1 are in a circular orbit, which includes the Medium Earth Orbit (MEO) and Geosynchronous Orbit (GEO) satellites, which tend to be easier to track compared to the LEO, and Geosynchronous Transfer Orbit (GTO) satellites, for which the perturbing accelerations are more complex to model or exhibit more significant variations in their orbit states [35]. To improve the sensor data quality, space-born sensors are emerging as a superior approach compared to ground-based observations since they are uninterrupted by daylight, and not affected by atmospheric phenomena such as diffraction, aberrations, turbulence and scattering [33]. However, this also creates the need for better sensor management schemes and multi-sensor data fusion strategies [32, 33, 87], which will further be discussed in section 2.2.

2.2. Approaches Overview

When it comes to MTT, various approaches have been developed through the years, starting with the Global Nearest Neighbour (GNN) [12] or efficient spatial searching algorithms associating tracks to measurements [73], to the modern approaches applying the Random Finite Sets framework. A general tree overview of the most relevant methods can be found in Figure 2.2. Even though they are compu-

[1] Including but not limited to: third body perturbations, atmospheric drag, solar radiation pressure, manoeuvres, mass distribution, attitude dependent effects

tationally efficient, methods such as the GNN cannot be applied to SSA due to their poor performance in situations where the clutter or target densities are high [28, 70]. On the other hand, more robust approaches to the problem are the Multiple Hypothesis Tracker (MHT) and the Joint Probabilistic Data Association (JPDA), both of which rely on data association, but are known to be NP-hard computationally i.e. scale exponentially with the number of measurements and targets. Despite this, improvements have been made and algorithms based on MHT have been demonstrated to handle scenarios with thousands of objects in the fields of cell biology and wildlife tracking [15, 9], which is still orders of magnitude below what is needed for SSA. Moreover, for JPDA, known limitations include computational efficiency, the handling of birth and death of targets [70], and decreased performance in the presence of densely spaced targets [28]. In recent years, S. Krishnaswamy [38] proposed a JPDA method via tensor compression for GEO MTT applications, performing more efficiently at the cost of accuracy, and even though ongoing research into JPDA methods for SSA exists, they are the exceptions. Indeed, in recent years when it comes to large-scale multi-object tracking, the preferred approaches and leading research occur within the RFS multi-sensor fusion algorithms framework, which is further subdivided into the PHD/CPHD filters [28, 87, 80, 58], the multi-Bernoulli filters [81, 46, 41, 44], LMB filters [4, 25, 35, 45, 60, 69, 82], hybrid Poisson Multi-Bernoulli filters [70, 86, 13, 26, 72], and HISP [19]. Before going over the strengths and limitations of these methods, a special mention to two pillars within the fields of RFS theory and MTT, R. Mahler [48] with the book *"Advances in statistical multisource-multitarget information fusion"* which was extensively used to understand the underlying mathematical concepts, and the Vo brothers with their numerous publications on the problem [31, 4, 75, 65, 35, 68, 66, 3, 5].

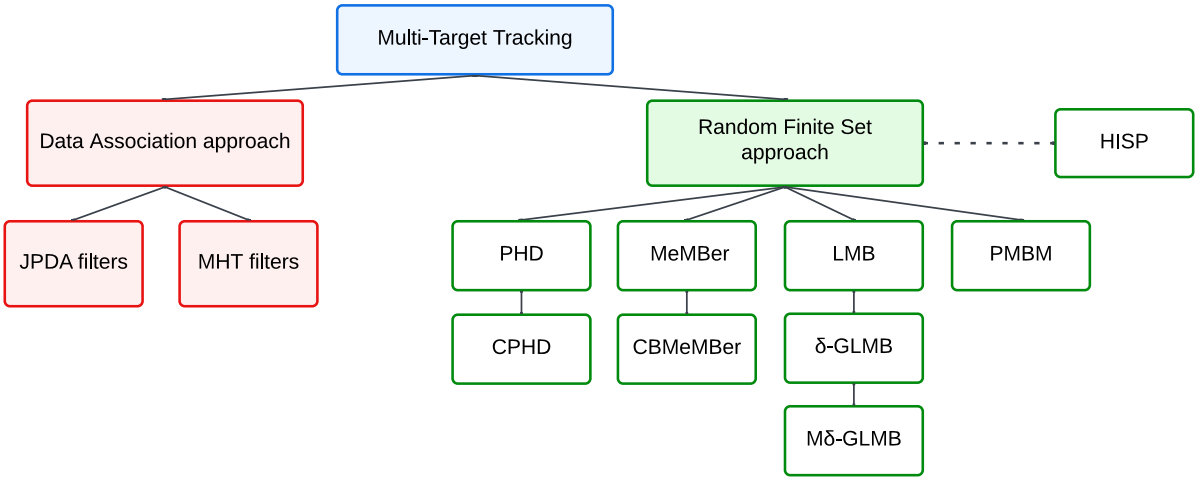


Figure 2.2: Overview of different MTT methods (HISP conceptually related to RFS but not part of the family)

GNN

The Global Nearest Neighbour (GNN) Method is one of the oldest and most straightforward recursive Bayesian multitarget filters, also known as the 2D assignment algorithm [12, 34]. It is a single-hypothesis assignment method: for each new set of measurements, the tracker attempts to assign each observation to an existing track or create a new track for unassigned measurements. This assignment is based on minimizing a global association distance, which measures how well a measurement matches a predicted target state, taking into account both position and uncertainty.

The term "global" distinguishes GNN from simple nearest-neighbor approaches. Instead of independently assigning each measurement to the closest track, GNN considers all possible measurement-to-track pairings simultaneously and selects the combination that minimizes the total association cost across all tracks. This step is essential in high-density scenarios, as it prevents conflicting assignments such as two tracks claiming the same measurement and ensures that the overall assignment across all tracks is optimal.

A key preprocessing step in GNN (and for all MTT methods) is gating, which reduces the number of candidate measurements considered for each track. Gating defines a validation region around the predicted measurement using the track's predicted state and its uncertainty. This region is typically

ellipsoidal in measurement space and is defined using the Mahalanobis distance between the actual measurement y_{k+1} and the predicted measurement $\hat{y}_{k+1} = g(\hat{x}_{k+1|k})$:

$$d^2(y_{k+1}) = \tilde{y}^T S^{-1} \tilde{y} \leq G, \quad \tilde{y} = y_{k+1} - \hat{y}_{k+1}, \tag{2.1}$$

where:

- y_{k+1} is the actual measurement obtained at time step $k+1$,
- $\hat{y}_{k+1}$ is the predicted measurement for the track at time $k+1$, calculated using the measurement model $g(\cdot)$,
- $\hat{x}_{k+1|k}$ is the predicted state of the track at time $k+1$ based on the previous estimate at time k,
- $\tilde{y}$ is the measurement residual vector, representing the difference between the actual and predicted measurements,
- S is the predicted measurement covariance matrix, representing uncertainty in the predicted measurement. In the simplest linear Kalman filter case, S is computed as $S = HPH^T + R$, where P is the predicted state covariance matrix, H is the measurement matrix that maps the state space into the measurement space, and R is the measurement noise covariance matrix. A more thorough discussion of Kalman filters for linear and non-linear scenarios is provided in subsection 4.5.2.
- $d^2(y_{k+1})$ is the squared Mahalanobis distance of the measurement from the predicted track,
- G is the gating threshold, defining the boundary of the validation gate.

Only measurements satisfying $d^2(y_{k+1}) \leq G$ are considered for association, while measurements outside the gate are ignored. Increasing G enlarges the gate, allowing more candidate measurements at the cost of potentially more ambiguous associations.

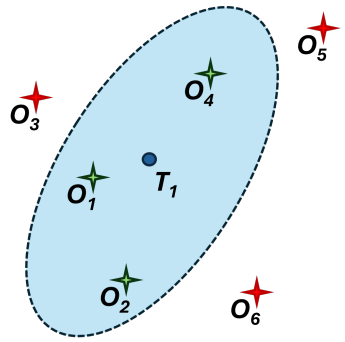


Figure 2.3: Visual representation of a validation gate for a track and its associated measurements. $T_1 \equiv$ estimated mean state of target 1, dashed line represents an equiprobability curve (i.e. Gate)

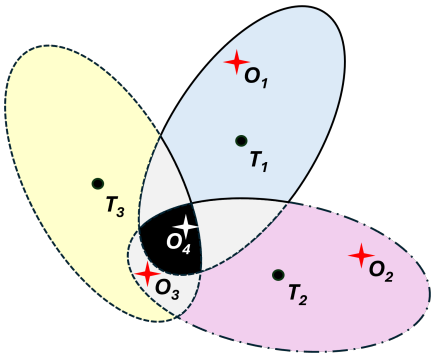


Figure 2.4: Visual representation of the GNN with data associations conflicts

In low-density scenarios, if no conflicts occur between tracks, the GNN assignment is straightforward: each track is simply assigned to its nearest neighbour within the gate. However, in high target or clutter density, as is common in SSA, multiple measurements may fall within a track's validation gate, or a single measurement may lie inside multiple track gates, increasing the risk of incorrect associations [28]. This is illustrated in Figure 2.3, where measurements $O_i, \forall i \in \{1, 2, 4\}$ are within the validation gate of T_1, which represents the predicted mean state of target 1. Measurements outside the gate, $O_i, \forall i \in \{3, 5, 6\}$, are precluded from association with T_1. A visual example and pseudo-code of gating are presented in chapter 4, although the basic approach follows the same principles described in this section. On the other hand, in case there are multiple targets, as in Figure 2.4, it is possible for a measurement to fall within the validation gate of multiple targets, as is the case of O_3, which is inside the validation gate of T_3, T_2 and O_4, which is in the validation gate of all three targets. To resolve this conflict, the tracker needs to solve a more complex data association problem by evaluating a cost matrix C_{ij}, which in this research is done using Murty's algorithm [56] in combination for the Hungarian method for optimal assignments [39] as further explained in section 4.5. Nonetheless, a simplified cost matrix structure may consist of a generalised statistical distance between observation i and track j,

such as the Mahalanobis distance, to which additional penalties can be added based on factors such as penalising tracks with larger prediction uncertainties. Using Figure 2.4 as template the following hypothetical cost matrix C can be produced,

$$C = \begin{array}{c} \begin{matrix} O_1 & O_2 & O_3 & O_4 \end{matrix} \\ \begin{bmatrix} \mathbf{7.1} & \times & \times & 4.3 \\ \times & \mathbf{5.2} & 8.4 & 5.1 \\ \times & \times & 6.6 & \mathbf{3.2} \end{bmatrix} \end{array} \begin{matrix} \\ T_1 \\ T_2 \\ T_3 \end{matrix}$$

where $\times$ denotes when a measurement i is not considered a possible assignment for track j, as it falls outside its validation gate. Moreover, for this example the optimal assignment $A_1 = \{(T_1, O_1), (T_2, O_2), (T_3, O_4)\}$ with a total cost of 15.5 is highlighted in bold and can be found using the Hungarian algorithm [39] (algorithm 6) for optimal assignments. It should be noted that for implementation purposes $\times$ can be assigned a very large values (e.g. 1000) or set to the numerical limit (i.e. $+\infty$) such that the total cost of solutions with $\times$ is never the optimal assignment. Furthermore, the GNN method will use the unassigned measurement O_3 to create a new tentative track T_4. On the other hand, for more advanced MTT methods or assignment problems it is possible that suboptimal solutions are required, i.e. the k next best possible assignment solutions. In order to compute these suboptimal assignments the Murty algorithm [56] (algorithm 7) can be used. After the single most likely association A_1 is computed using the Hungarian algorithm (or equivalent optimal assignment algorithms), Murty's algorithm computes the suboptimal solution by excluding possible individual assignments from the cost matrix one at a time and running the Hungarian algorithm for optimal assignments on the reduced problem. This process is repeated until a set number of possible assignments have been returned. For the simplified example the optimal/suboptimal solutions summarised in Table 2.1 are computed using Murty's algorithm [56] and in subsection 4.5.3 the step by step Hungarian algorithm process to compute A_1 is showcased.

Table 2.1: Murty's Assignment Algorithm Results for Optimal/Suboptimal Assignments (5-best assignments)

Rank i	**Assignment** A	**Total Cost**
1	$\{(T_1, O_1),\ (T_2, O_2),\ (T_3, O_4)\}$	15.5
2	$\{(T_1, O_4),\ (T_2, O_2),\ (T_3, O_3)\}$	16.1
3	$\{(T_1, O_1),\ (T_2, O_3),\ (T_3, O_4)\}$	18.7
4	$\{(T_1, O_1),\ (T_2, O_4),\ (T_3, O_3)\}$	18.8
5	$\{(T_1, O_1),\ (T_2, O_2),\ (T_3, O_3)\}$	18.9

MHT

MHT offers a more exhaustive approach to solving the multitarget estimation problem by employing all permutations of the measurement-to-track association, guaranteeing that the correct association is found at the expense of computational load [28]. In a measurement-oriented implementation as originally introduced by Reid [63], this computational burden originates from the creation of a new hypothesis of object-state-updates for each permutation, while considering all possible object-to-track associations. Furthermore, not only are all of the hypotheses propagated to the next measurement time, but new ones are also generated, such that the procedure can become computationally demanding in case the number of objects is large, as is the case for SSA. Nonetheless, this guarantees that one hypothesis will contain the correct associations at all times with an optimal estimate, given that all permutations are kept. An illustrative example of how the hypothesis table and associated probabilities can be generated for a two-measurements, two-tracks scenario is provided in Figure 2.5 and Table 2.2.

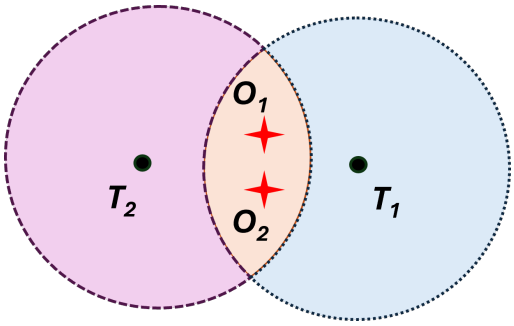


Figure 2.5: Assignment problem of measurements $O_i, \forall i \in \{1,2\}$ to tracks $T_i, \forall i \in \{1,2\}$ in a two measurements two tracks scenario with overlapping gates and measurements

Table 2.2: Hypothesis Table MHT, num(T) = 2, num(O) = 2

Hyp	$\mathbf{T_1}$	$\mathbf{T_2}$	L(Hyp)
1	$\emptyset$	$\emptyset$	$(1-p_D)^2\lambda^2$
2	$\emptyset$	O_1	$f_{12} \cdot p_D \cdot (1-p_D)\lambda$
3	$\emptyset$	O_2	$f_{22} \cdot p_D \cdot (1-p_D)\lambda$
4	O_1	$\emptyset$	$f_{11} \cdot p_D \cdot (1-p_D)\lambda$
5	O_1	O_2	$f_{11} \cdot f_{22} \cdot p_D^2$
6	O_2	$\emptyset$	$f_{21} \cdot p_D \cdot (1-p_D)\lambda$
7	O_2	O_1	$f_{21} \cdot f_{12} \cdot p_D^2$

In Table 2.2, each row represents one hypothesis, i.e. the assignment of the measurements O_i (including the no measurement assignment $\emptyset$) for all tracks T_j. On the other hand, each column represents possible assignments of measurements for one track. As such, Hyp(2) : [$\emptyset$ O_1] assigns no measurements to track 1 and assigns measurement 1 to track 2. Moreover, computing the probability that a given hypothesis is correct is performed by first using Eqs.(2.2) - (2.3), in which the likelihood of each assignment a of measurement O_i to track T_j is determined [28], and further multiplying these likelihoods within the same row and normalising.

$$L(a) = \begin{cases} (1-p_D)\lambda^{N_D} & \text{if } a \in \emptyset \\ f_{ij} \cdot p_D \cdot \lambda^{(N_D-1)} & \text{if } a \in \{O_i\},\ i \in \mathcal{N}^* \end{cases} \tag{2.2}$$

$$f_{ij} = p_f\left(\mathbf{z}^{(i)}; \bar{\mathbf{z}}^{(j)}, P_{zz}^{(j)}\right) \tag{2.3}$$

Moreover, it can be seen that in the scenario where $a \equiv \emptyset$ i.e. no measurement is assigned to a track j, $L(\emptyset)$ is a function of the probability of detection p_D, the false alarm density λ and the number of detections within the gate N_D. On the other hand, in case a measurement is assigned to a track, $L(a)$ is in addition a function of the measurement likelihood f_{ij}, which in turn depends on measurement $\mathbf{z}^{(i)}$, predicted measurement $\bar{\mathbf{z}}^{(j)}$, and uncertainty matrix $P_{zz}^{(j)}$.

Hence, for each hypothesis, the tracks are updated and propagated to the next epoch, where a hypothesis generates a new hypothesis table, for which the probabilities are computed anew and normalised using the hypothesis that spawned it. This generation of new hypotheses can grow exponentially with the number of objects to be tracked, requiring significant computational resources. In practice, however, various techniques can be employed to remove unlikely hypotheses, set thresholds, improve computational efficiency, and prevent the number of hypotheses from growing unreasonably large. Indeed, when the number of objects is large, the measurement-oriented approach by Reid [63] can make the problem computationally demanding, and although advances have been made to reduce the number of hypotheses [17] or employ track-oriented MHT [34, 11], it is still estimated that for SSA and MTT, more suitable approaches exist.

JPDA

While the GNN method makes a rigid assignment of a detection to a track, the JPDA method applies a soft assignment so that detections within the validation gate of a track can all make weighted contributions to the track based on their probability of association [22]. Furthermore, similarly to the MHT, the JPDA approach produces the same hypothesis table and probabilities, but further merges the hypotheses for a weighted measurement update, thus propagating a single set of object states per epoch.

For instance, returning to the hypothesis Table 2.2, to perform the weighted measurement update, the sum of probabilities of all hypotheses having a matching measurement to track association is gathered. As such if we consider H_4 and H_5 which both assign O_1 to T_1, the total probability of this assignment

therefore becomes $p_{11} = p(H_4) + p(H_5)$. Obtaining the weighted residuals used to update the tracks in the correction step:

$$\mathbf{y}_k^{(i,j)} = \mathbf{z}_k^{(i)} - \bar{\mathbf{z}}_k^{(j)}$$
$$\mathbf{y}_k^{(j)} = \sum_{i=1}^{m} p_{ij} \mathbf{y}_k^{(i,j)}$$

Another way of understanding what is happening is considering that the JPDA requires one more step compared to the GNN as since T_2 and T_1 conflict for the assignment of O_1, to calculate the associated probability p_{11} we must consider the joint probability that T_2 is not assigned to O_1 i.e. T_2 is assigned to O_2 or is unassigned. As such if another measurement O_3 was in the gate of T_2, we would need to consider in addition that T_2 is assigned to O_3, a new hypothesis H_8 : [O_1 O_3] or simply $p_{11}^{\star} = p_{11} + p(H_8)$.

Hence, JPDA is less resource-intensive than MHT due to its hypothesis management, which reduces the hypothesis table to a single set of object states to propagate per epoch. Nonetheless, because it merges information from measurements not necessarily generated by each target, JPDA produces less accurate estimates than MHT in regions with densely spaced targets, such as in SSA [28]. Consequently, conventional JPDA filters can be error-prone in such high-density scenarios [70, 48] and may still become computationally expensive, although generally less so than MHT [70, 48]. Recent advancements by Krishnaswamy [38] use tensor compression methods to represent a low-dimensional approximation of the measurements, achieving a speedup at the cost of some accuracy loss. However, the author noted that realistic data association scenarios for GEO MTT remain challenging due to the ”exponential scaling of JPDA itself.” Therefore, JPDA methods are not further investigated in this research.

PHD/CPHD

The classical Probability Hypothesis Density (PHD) method proposed by Mahler [48], and based on RFS, estimates the intensity function (first-order moment) of the multi-target posterior distribution within the Bayesian framework. Unlike the previously discussed data association based methods such as GNN, MHT, or JPDA, the PHD filter simultaneously estimates both the number of targets and their states. It represents the multi-target state as a RFS i.e. a random size set of random vectors corresponding to object states, and operates on unlabelled targets, updating a spatial density rather than producing explicit tracks. In the same way that the Kalman filter is a moment approximation to the single-target Bayes filter, the PHD is the first moment approximation to the multitarget Bayes filter. It effectively addresses challenges such as sensor inaccuracies, dynamic target variations and unknown & time varying number of targets [28, 87]. Nonetheless, implementing this filter without simplifications is a difficult and computationally complex task due to its multidimensional integrals. Two possible simplifications were introduced by Vo et al. - approximating the PHD as a Gaussian Mixture (GM) yielding the classical GM-PHD filter [74] or implementing the filter using the sequential Monte Carlo (SMC) method yielding the classical SMC-PHD filter [76]. The GM-PHD method is preferred over the SMC-PHD method since it has the advantage of lower computational cost and simpler state extraction. However, using the classical GM-PHD approach has drawbacks, including the production of a large number of redundant, invalid likelihood functions in dense clutter environments, and the expected number of targets varies significantly from one epoch to the next, resulting in reduced computational efficiency [28, 48, 80].

A solution to improve the GM-PHD filter was proposed by Vo et al. [57], introducing a tracking label, extracting the target track by assigning a label to each Gaussian component. However, in a dense clutter environment, this leads to the emergence of many Gaussian components from unidentified sources, hence increasing the computational complexity of the label GM-PHD in the subsequent prediction and update steps. As such, Wang et al. [80] introduced improvements to the steps for the Gaussian components with the same label, greatly reducing the computational burden and error in dense clutter environments. Similarly, Zhu et al. [87] observed that the classical GM-PHD filter mistakenly identifies stars passed by targets as actual targets, further exacerbating the existing target quantity estimation error of classical PHD filters in space-borne sensor scenarios. The authors proposed an improved GM-PHD filtering algorithm, which removes stars and noise based on spatio-temporal pipeline filtering techniques [79].

On the other hand, another solution to address the high variability in the expected number of targets in the classical GM-PHD is to propagate and update the cardinality distribution in addition to the PHD function, yielding the Cardinalized Probability Hypothesis Density (CPHD) filter, for which the classical version is described in [48]. A centralised system, multi-sensor GM-CPHD with measurement-based birth model, was implemented by Gehly [28], and was successful in tracking a small number of geostationary objects in the presence of missed detections and sparse clutter. Nonetheless, one of the identified drawbacks of the CPHD filter was the so-called 'spooky' effect [23], in which missed detections cause the probability mass to shift from undetected to detected objects at a distance. Although clustering techniques exist to alleviate the problem, it suggests that better approaches based on multi-Bernoulli filters exist for SSA [28].

Moreover, one of the drawbacks of the centralised system assumption is that, although it is known to provide an optimal solution, it requires partitioning the sensor measurements into disjoint subsets to manage the combinatorial complexity of multi-target associations [28, 58, 44, 25]. In a centralised GM-CPHD filter, each measurement could potentially originate from any target or clutter, and the filter must compute the joint likelihood over all possible associations. Without partitioning, the number of possible measurement-to-target assignments grows combinatorially with the number of targets and measurements, making the problem intractable. Partitioning into subsets reduces this complexity by allowing the filter to process smaller groups of measurements independently, but even then, the total computational cost remains high for large numbers of targets, sensors, or measurements. This computational burden motivates the exploration of distributed implementations, which can achieve similar performance with lower resource requirements [58, 43].

On the other hand, distributed systems have a lower computational load and have shown promising results, as demonstrated by a distributed GM-CPHD filter implementation by Park et al. [58]. Similarly, a distributed network PHD filter can be implemented using the process described in [43].

Multi Bernoulli

Before introducing the multi-Bernoulli (MB) filter, it is helpful to recall the underlying distributions. A Bernoulli distribution models a binary random variable, representing whether a single target exists or not, with existence probability r (i.e., target exists with probability r and does not exist with probability $1-r$). Extending this to multiple independent targets yields the Multi-Bernoulli (MB) distribution, which is a collection of independent Bernoulli components, each with its own existence probability r_i and state distribution $p_i(x)$. In tracking terms, the MB distribution provides a convenient way to approximate the multi-target posterior by maintaining a set of Bernoulli components, each corresponding to a potential target. Figure 2.6 illustrate the Bernoulli and Multi-Bernoulli representations with a short summary of what is computed to get to the results below

Bernoulli distribution. A Bernoulli random variable has only two outcomes:

$$X = \begin{cases} 1 & \text{(target exists) with probability } r, \\ 0 & \text{(no target) with probability } 1-r. \end{cases}$$

Example with $r = 0.7$:

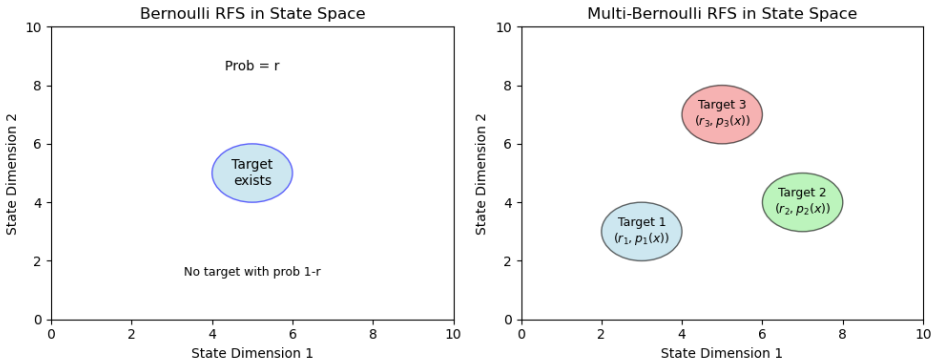
$$P(X=1) = r = 0.7, \qquad P(X=0) = 1-r = 0.3.$$

Multi-Bernoulli distribution (two targets). Consider two independent Bernoulli components with existence probabilities $r_1 = 0.8$ and $r_2 = 0.4$. The joint probability mass function (PMF) is obtained as the product of the independent Bernoulli probabilities:

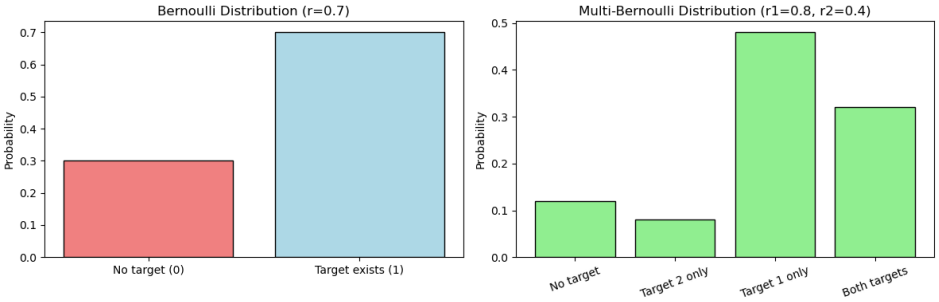
$$\begin{aligned} P(\varnothing) &= (1-r_1)(1-r_2) = 0.2 \times 0.6 = 0.12, \\ P(\{\mathsf{target}_2\}) &= (1-r_1)r_2 = 0.2 \times 0.4 = 0.08, \\ P(\{\mathsf{target}_1\}) &= r_1(1-r_2) = 0.8 \times 0.6 = 0.48, \\ P(\{\mathsf{target}_1, \mathsf{target}_2\}) &= r_1 r_2 = 0.8 \times 0.4 = 0.32. \end{aligned}$$

These four probabilities sum to 1, as required. The expected cardinality is

$$\mathbb{E}[N] = r_1 + r_2 = 1.2.$$



(a) Bernoulli RFS in state space. A single Bernoulli component represents the existence of one target with probability r, or no target with probability $1 - r$.



(b) Multi-Bernoulli RFS in state space. Several independent Bernoulli components represent different potential targets, each with its own existence probability r_i and spatial distribution $p_i(x)$.

Figure 2.6: Bernoulli and Multi-Bernoulli Random Finite Set (RFS) representations in state space.

Thus after the introduction of the PHD and CPHD filters, Mahler [48] proposed a multi-target multi-Bernoulli (MeMBer) filter, which approximates the multi-target posterior probability density function by calculating the updated multi-Bernoulli RFS of measurements and the missed targets, i.e. instead of propagating moments, the MB filter propagates the parameters of an MB distribution that approximate the posterior multi-target density by maintaining several Bernoulli components over time. Thus, it has the same complexity as the PHD scaling linearly with the number of measurements [44]. Furthermore, similarly to the PHD/CPHD relation, the MeMBer filter suffers from overestimating the cardinality and as such the Cardinality Balanced MeMBer filter was introduced [48]. Wang et al. [81] implemented an updated version of the CBMeMber based on Gaussian Mixtures and in a comparative study against the PHD/CPHD/MeMBer found that CBMeMBer achieves a better tracking performance, but has shortcomings when it comes to measurement innovation that are neglected when the probability of missed detections is large enough. Moreover, the missed detection target information is required when calculating the Bernoulli RFS of updated measurements. Additionally in an attempt to reduce the computational complexity, several variations of the filter's implementation have been developed, such as the standard jump Markov (JM) or the Rao-Blackwellized JM CMBMBer filters, which have been compared and analysed by Li [41], attempting to reduce computational complexity. Moreover, in the classical sense, the CBMeMBer filter is a single-sensor multi-target tracking method and has been extended to a multi-sensor system. Liu et al. [46] introduced a track association and fusion method for multi-sensor based on the CBMeMBer filter, and have shown that the performance is comparable to the equivalent multi-sensor implementations of the PHD filter, but the proposed method is insensitive to the observation noise and clutter rate, and marginally affected by the probability of detection. Nonetheless, it

oversimplifies the problem of target birth, which in real scenarios is unknown, but was assumed as prior knowledge.

GLMB/LMB

The Generalized Labeled Multi-Bernoulli (GLMB) filter builds on the Multi-Bernoulli concept using a conjugate prior for labeled RFSs. Labeled Random Finite Set (RFS) filters extend the concept of standard RFS filters by explicitly assigning unique labels to each target, allowing the filter to maintain identity over time. This labeling provides a natural framework for estimating target trajectories, rather than just the number and states of targets, which is the focus of PHD, CPHD, and multi-Bernoulli filters. Thus, unlike PHD, CPHD, and CBMeMBer filters, which only propagate target existence and state distributions, the GLMB maintains a mixture representation, where each component corresponds to a possible data association history. These explicit data associations are the main source of computational complexity in the δ-GLMB filter [20]. A detailed theoretical background of Labelled RFS theory is further discussed in subsection 4.3.1.

Reuter et al. [65] introduced the labeled Multi-Bernoulli (LMB) filter as a generalization of the multi-Bernoulli filter, combining the advantages of formal track estimation and avoidance of the 'spooky effect' seen in CPHD. The LMB can also be viewed as a computationally efficient approximation to the δ-GLMB filter. Later developments include Gibbs sampling implementations [64] and the Marginalized δ-GLMB (Mδ-GLMB) filter [20], which reduce complexity while preserving key statistics such as cardinality and PHD. Recent work by Beard et al. [4] extends GLMB methods to large-scale multi-object tracking scenarios.

Hybrid Poisson Multi Bernoulli

The Poisson Multi-Bernoulli Mixture (PMBM) filter family is one of the more recently developed when it comes to RFS filters, and represents the multi-target posterior as a union of a Poisson point process representing all undetected targets and a multi-Bernoulli mixture which considers all the data association hypotheses, and can be regarded as a generalised model of the δ-GLMB density [70, 86]. Furthermore, the PMBM filter can achieve the same tracking performance as the δ-GLMB filter, but with a lower computational load as it propagates significantly fewer hypotheses [86]. Fernández et al. [26] provided the PMBM filter's derivation and implementation and established the connection to the δ-GLMB filter. The PMBM filter family is a good candidate to handle the needs of SSA, with Cament et al. [13] implementing the Poisson Labelled Multi-Bernoulli (PLMB) filter for space-debris tracking in LEO orbit. The PLMB filter approximates the PMBM filter, propagating a single LMB distribution, whereas the PMBM propagates multiple MB distributions. It has been shown that the PLMB filter has similar MTT performance compared to the LMB/PMBM filters, but with higher computational efficiency [14]. The method introduced by Cament et al. [13] can further be developed by changing the multi-sensor strategy from iterative multi-target update by each sensor based on the Loopy Belief Propagation (LBP) algorithm, to a more advanced strategy such as Gibbs sampling, additional considerations could involve changing the single state distribution to better fit the RSO distributions and introducing a more complex state extractor. Even more recently, Zhao et al. [86] introduced a PMBM-based MTT under measurement merging with a GM-based implementation, and Gibbs sampling for efficient data association. However, this work can extend to multi-sensor cases and robust algorithm cases with partially unknown parameters.

HISP

The Hypothesised and Independent Stochastic Populations (HISP) filter stems from the framework of stochastic populations, which shares similarities with the RFS framework. The HISP filter is a principled approximation of the Distinguishable ISP (DISP) filter, maintaining linear complexity in the number of tracks and observations by simplifying the data-update step [19]. This property makes HISP an attractive, scalable solution for SSA. Delande et al. [19] present a practical implementation of the algorithm with promising results, although no explicit comparison with other methods is provided.

The current implementation is sequential, but the algorithm is amenable to parallelization, which could further improve computational efficiency. Additionally, the propagator could be enhanced by incorporating better modelling of perturbing accelerations.

2.2.1. Data Fusion

The main advantages of distributed networks are their increased security, redundancy and scalability. Data fusion methods are usually separated in three levels - low level i.e. fusing raw-data from sensors, Intermediate or feature level fusion i.e integration of features derived from multiple raw data source (or within the same raw data) viable for RSO detection, and high level or decision fusion i.e. combination of decisions/scores from multiple sensors [33]. A global overview of data fusion techniques can be found in [32], but the average consensus methods have been widely used in MTT for their scalability and flexibility [58, 44, 43, 18]. Among average consensus techniques, the linear arithmetic average (AA) and the log-linear geometric average (GA) are the most used, and a discussion on their applicability, pros & cons has been compiled by K. Da et al. [18], with the following observations:

- AA fusion is more effective at preventing misdetections but less capable of eliminating false alarms, whereas GA fusion is better at suppressing false alarms, but is more susceptible to misdetections.
- GA fusion of Poisson/Bernoulli/Identically independent distributed cluster - multi-target probability distributions tends to underestimate the number of targets, whereas AA fusion consistently maintains an unbiased cardinality. Furthermore, GA has been observed to experience a delay in detecting newborn targets.
- GA fusion is more complicated to implement and compute, and requires an approximation which is valid only when the Gaussian components are well separated. On the other hand, AA is easier to implement and can be done in parallel since it only requires union and re-weighting operations.

Hence, GA performs better in high clutter density and high false alarm rates, and AA performs better in low detection probability and closely spaced targets.

Nonetheless, consensus fusion has one major drawback in the specific field of SSA. For Earth and space-based tracking, the sensors typically only view a small portion of the target scene, with estimates often coming from a single sensor at a time. In consensus data fusion, such estimates are downweighted, as consensus-based approaches tend to favour data originating from multiple or all sensors, as it is designed at its core for scenarios where all sensors observe all targets. In cases with a limited field of view (FoV) or with missed detections, differences in the local posteriors can arise, resulting in actual objects being excluded from the fused multi-target estimate. Recently Gehly et. al. [27] and Battistelli et. al. [42] have proposed a complementary fusion approach based on Generalised Covariance Intersection fusion, respectively using the labelled CPHD filter and the LMB filter for sensors with different FoVs to counteract this issue.

2.2.2. Assessment Metrics

First, it is important to mention that for a quantity to be considered a metric (representing miss-distance) it must fulfil four properties:

P1. Non-Negativity i.e. $d(x, y) \geq 0$

P2. Symmetry i.e. $d(x, y) = d(y, x)$

P3. Identity i.e. $d(x, y) = 0 \Longleftrightarrow x = y$

P4. Triangle inequality i.e. $d(x, y) \leq d(x, z) + d(z, y)$

As such, the first well-established, mathematically consistent and physically meaningful metric for performance evaluation of multi-object filters is the Optimal Subpattern Assignment (OSPA), which Schuhmacher introduced [68], by generalising the concept of miss-distance (i.e. error between estimated and true state) to MTT. This metric has become standard practice and is used extensively throughout MTT literature. Nonetheless, OSPA presents some minor drawbacks such as:

- A single parameter determines both the penalty given to cardinality errors and the threshold at which physical distances in the state space are cut off; as such, separate control, which might sometimes be desirable, is not possible.
- In case of an empty point pattern, OSPA is insensitive to the cardinality of the nonempty point pattern.
- Phenomena such as track switching and fragmentation are not penalised consistently.

From that point onwards, many variations of OSPA have been introduced. For instance, OSPA for tracks (OSPA-T) and a method for its approximation were introduced by Ristik [66]. OSPA-T is used for target tracking where a metric on the space of finite sets of tracks[2] (typically of unequal length in time) is needed. However, T. Vu [78] discusses some of the limitations of the approximate OSPA-T, such that it no longer satisfies the axioms of a metric, specifically P4, the triangle inequality. The authors then present an alternative OSPA-MT method which alleviates these drawbacks, but the new metric becomes computationally demanding for practical problems. One of the newer developments when it comes to the OSPA metrics variants is the OSPA$^{(2)}$ method for which preliminary results were published by Beard et al. [3, 5]. The authors then published the complete mathematical details in [4]. The differentiating factor of OSPA$^{(2)}$ is that it is constructed from the base 'distance between' elements of the sets, where this 'base distance' itself is also constructed via OSPA, rather than from the distance between tracks as in the OSPA method. Thus, both the OSPA and OSPA$^{(2)}$ metrics measure the cardinality and precision of sets of targets. Still, only OSPA$^{(2)}$ is designed to take labels into account [13], and its implementation efficiency has been shown in [4]. Finally, it is worth mentioning the GOSPA metric introduced by Svensson et al. [62], which has the main difference from the OSPA metric in that it penalises cardinality errors differently by not normalising with the cardinality of the largest set, enabling the optimisation over assignments rather than permutations. This allows the introduction of penalties for localisation errors for detected targets and errors due to missed/false targets.

On the other hand, around the same time as the original OSPA paper [68] was published, K. Bernardin [8] introduced two methods of assessing multi-object tracking based on the CLEAR MOT heuristics - Multiple Object Tracking Accuracy (MOTA) 'metric' giving the accuracy in tracking targets by taking into account label switching, miss detections and false alarms, and the Multiple Object Tracking Precision (MOTP) giving the estimates target location errors when correctly detected. Even though these quantities have been used to assess performance in computer vision [61] and to a lesser extent for SSA [13], J. Bento [7] proceeds to show that MOTA is not a metric and specific conditions for MOTP need to be imposed to ensure it is one. Furthermore, in [7], two new metrics built upon MOTA/MOTP were introduced. However, it has been shown not to be suitable for large-scale problems, thus not applicable to SSA.

Identified Gaps

From the literature, most of the implementations were based on MATLAB, which is not open-source, or Python, which is computationally less efficient than compiled languages. Implementing the relevant methods in C++ allows for offsetting both issues at once. Furthermore, the presented literature implements estimators that consider using the same filter independently of the size of the RSO. Still, implementing different filters within the same estimator tailored for different RSO sizes/types can be an interesting approach for SSA. Finally, the Hybrid Poisson Multi-Bernoulli filters family and HISP sections have shown promising results for SSA applications. Still, both lack either an extension of the proposed idea to multi-sensor / distributed networks, a proper comparison analysis with more established methods such as the PHD/CPHD/GLMB, or provide opportunities for improvements in their implementation.

[2] With one track defined as a labelled temporal sequence

3

Research Question

Given the escalating complexity and computational demands of Multi-Sensor Multi-Target Tracking for Space Situational Awareness, where the need to monitor an increasing population of satellites and debris is critical the main research question of this thesis is:

Which methods within the Random Finite Set framework can be optimised, scaled up efficiently and combined to handle the increased computational demands of the distributed network multi-sensor multi-target problem for space situational awareness?

To tackle this question we further sub-divide it into the following research guiding questions:

RQ-I. Which RFS methods perform the best under the simplifying assumption of central network single-sensor MTT problem based on OSPA and computational time metrics?

RQ-II. How do these methods perform on multi-sensor centralised/distributed network MSMTT problem based on OSPA and computational time metrics?

RQ-III. How do the estimators perform on a simplified SSA model?

4

Methodology

This chapter provides a more in-depth review of the theoretical background and multi-target techniques which will be used for this research. Additionally, some implementation techniques and code structures are provided.

For the remainder of this chapter, the assumed equations of motion are of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{w}, t) \tag{4.1}$$

where $\mathbf{x}$ is the state vector and $\mathbf{w}$ is a zero mean white noise process. Furthermore, the measurement relationship is given by

$$\mathbf{z}_k = g(\mathbf{x}_k, t_k) + \epsilon_k, \quad \mathbf{x}_k \equiv \mathbf{x}(t_k) \tag{4.2}$$

where ϵ_k is a zero mean measurement noise vector. Moreover, for the orbital debris problem, both the dynamics and measurement relationships are generally non-linear.

4.1. Finite Set Statistics

The approach taken in this research places finite set statistics (FISST) at its core; as such, a broad introduction to what they represent will be provided, based on the whole, in-depth derivations and discussions advanced by Mahler [48]. FISST was introduced to address the main drawbacks of conventional multitarget filters, which rely on single-target filters to propagate and update object states, resulting in the requirement to associate individual measurements with specific targets —a data association task that is the most computationally expensive in these methods. On the other hand, the distinguishing feature of FISST is that it applies multitarget statistics to process a set of measurements and to update a set of object states, directly seeking the optimal and suboptimal estimates of the multitarget state, disregarding the data association problem [75].

The core element of FISST is the Random Finite Set, which represents an order-independent set of random vectors used to describe sets of object states or measurement vectors. As such we can define the targets state set X, and the measurement sets Z^{k+1}, i.e.

$$X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \quad \text{with } n \geq 0\text{; and} \tag{4.3}$$

$$Z^{k+1} : Z_1, \ldots, Z_{k+1} \qquad \text{where } Z_k = \{z_1(t_k), ..., z_m(t_k)\} \quad m \geq 0 \tag{4.4}$$

is the time history of measurement sets at t_{k+1}. Thus, in place of the list of measurement-updated association hypotheses, the RFS approach uses a Multitarget Probability Density Function (M)PDF

$$f_{k+1|k+1}(X|Z^{k+1}) \tag{4.5}$$

defined on the finite sets from Eqs. (4.3) - (4.4). As such, $f_{k+1|k+1}(X|Z^{k+1})$ represents the probability (density) that the targets have state set X, given measurement history Z^{k+1}. Moreover, since $n \in \mathbb{N}$

the following scenarios are possible:

$$X = \begin{cases} \emptyset & \text{if} \quad \text{no targets are present} \\ \{\mathbf{x}_1\} & \text{if} \quad \text{one target with state } \mathbf{x}_1 \text{ is present} \\ \{\mathbf{x}_1, \mathbf{x}_2\} & \text{if} \quad \text{two targets with states } \mathbf{x}_1 \neq \mathbf{x}_2 \text{ are present} \\ \vdots & \vdots \quad \vdots \\ \end{cases} \tag{4.6}$$

To illustrate the three core elements of FISST i.e. the multitarget PDF, belief-mass function, and set integral —an example based on the state RFS X is provided. First, we assume that the multitarget PDF $p(X) \geq 0$ exists. Then the belief-mass function β can be defined as the probability that X is in some space Ω such that

$$\beta_X(\Omega) = \mathsf{Pr}(X \in \Omega) = \int_\Omega p(X) dX \tag{4.7}$$

For which we define the set integral as a sum of integrals, where each term accounts for a different possible number of targets n. This is the case because the RFS contains a discrete, unordered number of continuous state vectors.

$$\begin{aligned} \int_\Omega p(X) dX &= \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\Omega \times \ldots \times \Omega} j^{(n)} \left(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \right) d\mathbf{x}_1 \cdots d\mathbf{x}_n \\ &= p(\emptyset) + \int_\Omega j^{(1)}(\{\mathbf{x}_1\}) d\mathbf{x}_1 + \frac{1}{2} \int_{\Omega \times \Omega} j^{(2)} \left(\{\mathbf{x}_1, \mathbf{x}_2\} \right) d\mathbf{x}_1 d\mathbf{x}_2 + \ldots \end{aligned} \tag{4.8}$$

where $j^{(n)}(\ldots)$ represents the spatial distribution of n targets known as the n-th order Janossy density. Thus, it can now be defined

$$\int p(X) dX = 1 \tag{4.9}$$

from which the cardinality distribution $p(n)$ can be defined as

$$p(n) = \mathsf{Pr}(|X| = n) = \frac{1}{n!} \int j^{(n)}(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) \mathbf{x}_1 \cdots d\mathbf{x}_n \tag{4.10}$$

which is the probability that there are exactly n targets in $X \equiv |X| = n$, meaning that each term of the set integral corresponds to the cardinality distribution for that number of targets yielding

$$\int p(X) dX = p(\emptyset) + p(1) + p(2) + \cdots = 1 \tag{4.11}$$

Lastly, we introduce the concept of statistical moments. For single-target statistics, the first moment of a PDF is the mean. Similarly, it is possible to extend this concept where, in the multitarget case, the first moment is the probability hypothesis density function defined by

$$\nu(\mathbf{x}) = \int p(\{\mathbf{x}\} \cup X) dX \tag{4.12}$$

4.2. Probability Hypothesis Density

Some general background for the PHD filter has already been provided in section 2.2. In this section, the focus will primarily be on introducing the base formulas and methodology for the simulation and presenting the results of a simple 2D benchmark test case.

4.2.1. Theory

The PHD represents a target density in some region of single target space Ω. It propagates the posterior intensity, a statistical moment of the posterior multiple target state [74]. As such by integrating Eq. (4.12) over Ω we can obtain the number of targets N in Ω, as the integral of ν over any region Ω gives the expected number of elements of X that are in Ω.

$$\hat{N} = \int |X \cap \Omega| \, P(dX) = \int_{\Omega} \nu(\mathbf{x}) d\mathbf{x} \tag{4.13}$$

In Equation 4.13, the notation $P(dX)$ represents the probability measure over all possible realizations of the RFS X [48]. Each realization X may contain a different number of targets with varying states, and the measure assigns a probability to each configuration. The set integral

$$\int |X \cap \Omega| \, P(dX)$$

thus computes the expected number of targets in the region Ω, averaging the random variable $|X \cap \Omega|$ over all possible sets X. Intuitively, $|X \cap \Omega|$ simply counts the number of targets from the set X that lie within the region Ω.

When a multitarget probability density function $p(X)$ exists, the measure can be expressed in density form as $P(dX) = p(X)\, dX$, where dX denotes the set integral element over all finite subsets of the single-target state space. Using this representation, the expectation can equivalently be written in terms of the PHD $\nu(\mathbf{x})$ as

$$\hat{N} = \int_{\Omega} \nu(\mathbf{x}) \, d\mathbf{x},$$

providing a practical method to compute the expected number of targets as the first-order statistical moment of the multitarget state. Moreover, the PHD process by its nature can be summarised as a two-step predictor/corrector as depicted in Figure 4.1 [48].

					predictor		corrector			
Bayes:	$\cdots$	$\rightarrow$	$p_{k-1}(X_{k-1} \mid Z_{1:k-1})$	$\rightarrow$	$p_{k\|k-1}(X_k \mid Z_{1:k-1})$	$\rightarrow$	$p_k(X_k \mid Z_{1:k})$	$\rightarrow$	$\cdots$
			$\downarrow$		$\downarrow$		$\downarrow$		
PHD:	$\cdots$	$\rightarrow$	$\nu_{k-1}(\mathbf{x}_{k-1})$	$\rightarrow$	$\nu_{k\|k-1}(\mathbf{x}_k)$	$\rightarrow$	$\nu_k(\mathbf{x}_k)$	$\rightarrow$	$\cdots$

Figure 4.1: PHD simplified schematic

The full PHD recursion is provided in Eqs. (4.14) - (4.15), with symbol explanations in Table 4.1.

$$\begin{aligned} \nu_{k|k-1}(\mathbf{x}) = & \int p_{S,k}(\boldsymbol{\zeta}) f_{k|k-1}(\mathbf{x}_k \mid \boldsymbol{\zeta}) \, \nu_{k-1}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\ & + \int \beta_{k|k-1}(\mathbf{x_k}|\boldsymbol{\zeta}) \nu_{k-1}(\boldsymbol{\zeta}) \, d\boldsymbol{\zeta} \\ & + \gamma_k(\mathbf{x_k}) \end{aligned} \tag{4.14}$$

$$\begin{aligned} \nu_k(\mathbf{x}) = & \left[1 - p_{D,k}(\mathbf{x}_k)\right] \nu_{k|k-1}(\mathbf{x}) \\ & + \sum_{\mathbf{z}_k \in Z_k} \frac{p_{D,k}(\mathbf{x}_k) \, g_k(\mathbf{z}_k \mid \mathbf{x}_k) \, \nu_{k|k-1}(\mathbf{x})}{\kappa_k(\mathbf{z}_k) + \int p_{D,k}(\mathbf{x}) g_k(\mathbf{z}_k \mid \boldsymbol{\zeta}) \, \nu_{k|k-1}(\mathbf{x}) d\boldsymbol{\zeta}} \end{aligned} \tag{4.15}$$

Table 4.1: Symbol legend for full PHD recursion

Term	Explanation	Term	Explanation
$\gamma_k(\cdot)$	Intensity of the birth RFS Γ_k at time k	Γ_k	RFS of spontaneous birth at time k
$\beta_{k\|k-1}(\cdot\|\boldsymbol{\zeta})$	Intensity of the RFS $B_{k\|k-1}(\zeta)$ spawned at time k by a target with previous state $\boldsymbol{\zeta}$	$B_{k\|k-1}(\zeta)$	RFS of targets spawned at time k from a target with previous state $\boldsymbol{\zeta}$
$p_{S,k}(\boldsymbol{\zeta})$	Probability that a target still exists at time k given that its previous state is $\boldsymbol{\zeta}$	$p_{D,k}(\mathbf{x_k})$	Probability of detection given a state $\mathbf{x}_k$ at time k
$\kappa_k(\cdot)$	Intensity of clutter RFS K_k at time k	K_k	RFS of clutter received by the sensor at time k
$f_{k\|k-1}(\mathbf{x}_k \mid \boldsymbol{\zeta})$	Single-target transition density: probability density that a target at state $\boldsymbol{\zeta}$ at time $k-1$ moves to state $\mathbf{x}_k$ at time k	$g_k(\mathbf{z}_k \mid \mathbf{x}_k)$	Single-target measurement likelihood: probability density of observing measurement $\mathbf{z}_k$ given the target is at state $\mathbf{x}_k$

However, since the computation of the integrals necessary to implement the filter is a complicated or even intractable task, to simplify the problem we will follow the proposed Gaussian Mixture approximation by Vo et al. [74] as well as the implementation approach outlined in [74] and [49], which comes with an example database in MATLAB `https://ba-tuong.vo-au.com/codes.html`.

PHD Recursion for Linear Gaussian Models

Table 4.2: Simplifying Assumptions

#	Description
1	Targets follow a linear Gaussian dynamical model, and the sensor has a linear Gaussian measurement model, i.e. $$f_{k\|k-1}(\mathbf{x}\|\boldsymbol{\zeta}) = \mathcal{N}(\mathbf{x}; \mathbf{F}_{k\|k-1}\boldsymbol{\zeta}, \mathbf{Q}_k), \quad g_k(\mathbf{z}\|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{H}_k\mathbf{x}, \mathbf{R}_k),$$ where $\mathcal{N}(\cdot; \mathbf{m}, \mathbf{P})$ denotes a Gaussian density with mean $\mathbf{m}$ and covariance $\mathbf{P}$, and $\mathbf{F}_{k\|k-1}$, $\mathbf{Q}_k$, $\mathbf{H}_k$, and $\mathbf{R}_k$ are respectively the state transition matrix, process noise covariance, observation matrix, and observation noise covariance.
2	The survival and detection probabilities are state-independent, i.e. $$p_{S,k\|k-1}(\mathbf{x}) = p_{S,k\|k-1}, \quad p_{D,k}(\mathbf{x}) = p_{D,k}.$$
3	The intensity of the birth RFS is a Gaussian mixture: $$\gamma_k(\mathbf{x}) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{\gamma,k}^{(i)}, \mathbf{P}_{\gamma,k}^{(i)}),$$ where $J_{\gamma,k}$, $w_{\gamma,k}^{(i)}$, $\mathbf{m}_{\gamma,k}^{(i)}$, and $\mathbf{P}_{\gamma,k}^{(i)}$ are model parameters determining the shape of the birth intensity.

4 No target spawning is considered i.e. the spawned target RFS intensity is zero:

$$\beta_{k|k-1}(\mathbf{x}|\boldsymbol{\zeta}) = 0 \quad \Rightarrow \quad B_{k|k-1} = \emptyset$$

Given the posterior PHD $\nu_{k-1}(\mathbf{x})$ which follows

$$v_{k-1}(\mathbf{x}) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)}) \tag{4.16}$$

and under the assumptions summarised in Table 4.2, the predicted PHD (with no spawning model) is given by:

$$v_{k|k-1}(\mathbf{x}) = \gamma_k(\mathbf{x}) + p_{S,k|k-1} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{S,k|k-1}^{(j)}, \mathbf{P}_{S,k|k-1}^{(j)}) \tag{4.17}$$

where

$$\begin{aligned} \mathbf{m}_{S,k|k-1}^{(j)} &= \mathbf{F}_k \mathbf{m}_{k-1}^{(j)} \\ \mathbf{P}_{S,k|k-1}^{(j)} &= \mathbf{Q}_k + \mathbf{F}_k \mathbf{P}_{k-1}^{(j)} \mathbf{F}_k^\top \end{aligned}$$

Now, suppose that the predicted PHD is rewritten as

$$v_{k|k-1}(\mathbf{x}) = \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}(\mathbf{x}; \mathbf{m}_{k|k-1}^{(i)}, \mathbf{P}_{k|k-1}^{(i)})$$

Then the update step of the PHD recursion, under the linear Gaussian assumptions, is given by the posterior PHD $\nu_k(\mathbf{x})$:

$$v_k(\mathbf{x}) = (1 - p_{D,k}) v_{k|k-1}(\mathbf{x}) + p_{D,k} \sum_{\mathbf{z} \in Z_k} \sum_{j=1}^{J_{k|k-1}} \frac{w_{k|k-1}^{(j)} q_k^{(j)}(\mathbf{z}) \mathcal{N}(\mathbf{x}; \mathbf{m}_{k|k}^{(j)}(\mathbf{z}), \mathbf{P}_{k|k}^{(j)}(\mathbf{z}))}{\kappa_k(\mathbf{z}) + p_{D,k} \sum_{\ell=1}^{J_{k|k-1}} w_{k|k-1}^{(\ell)} q_k^{(\ell)}(\mathbf{z})}, \tag{3.28}$$

where

$$q_k^{(j)}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\eta}_{k|k-1}^{(j)}, \mathbf{S}_{k|k-1}^{(j)}), \tag{4.18}$$

$$\boldsymbol{\eta}_{k|k-1}^{(j)} = \mathbf{H}_k \mathbf{m}_{k|k-1}^{(j)}, \tag{4.19}$$

$$\mathbf{S}_{k|k-1}^{(j)} = \mathbf{H}_k \mathbf{P}_{k|k-1}^{(j)} \mathbf{H}_k^\top + \mathbf{R}_k, \tag{4.20}$$

$$\mathbf{m}_{k|k}^{(j)}(\mathbf{z}) = \mathbf{m}_{k|k-1}^{(j)} + \mathbf{K}_k^{(j)} (\mathbf{z} - \boldsymbol{\eta}_{k|k-1}^{(j)}), \tag{4.21}$$

$$\mathbf{P}_{k|k}^{(j)} = \mathbf{P}_{k|k-1}^{(j)} - \mathbf{P}_{k|k-1}^{(j)} \mathbf{H}_k^\top [\mathbf{S}_{k|k-1}^{(j)}]^{-1} \mathbf{H}_k \mathbf{P}_{k|k-1}^{(j)}, \tag{4.22}$$

$$\mathbf{K}_k^{(j)} = \mathbf{P}_{k|k-1}^{(j)} \mathbf{H}_k^\top [\mathbf{S}_{k|k-1}^{(j)}]^{-1} \tag{4.23}$$

After the prediction and update steps, a simple pruning, merging, and capping scheme is implemented to manage the number of components in the mixture. Considering the posterior intensity:

$$v_k(\mathbf{x}) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(\mathbf{x}; \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}) \tag{4.24}$$

the pruning step consist of obtaining an approximation of $v_k(\mathbf{x})$ by truncating components with weights $w_k^{(i)}$ below a certain threshold. The number of considered components is further reduced by merging components which have similar means and covariances. This can be done using a clustering algorithm where the components are re-ordered from highest to lowest weight, and descending from the first to the last component are merged if they fall within a threshold distance from the covariance of the i-th component computed using

$$\frac{w_k^{(i)} w_k^{(j)}}{w_k^{(i)} + w_k^{(j)}} (\mathbf{m}_k^{(i)} - \mathbf{m}_k^{(j)})^\top (\mathbf{P}_k^{(i)})^{-1} (\mathbf{m}_k^{(i)} - \mathbf{m}_k^{(j)}) \tag{4.25}$$

The merging is then performed using:

$$\tilde{w}_k^{(\ell)} = \sum_{i \in L} w_k^{(i)} \tag{4.26}$$

$$\tilde{\mathbf{x}}_k^{(\ell)} = \frac{1}{\tilde{w}_k^{(\ell)}} \sum_{i \in L} w_k^{(i)} \mathbf{x}_k^{(i)} \tag{4.27}$$

$$\tilde{\mathbf{P}}_k^{(\ell)} = \frac{1}{\tilde{w}_k^{(\ell)}} \sum_{i \in L} w_k^{(i)} \left[\mathbf{P}_k^{(i)} + (\tilde{\mathbf{m}}_k^{(\ell)} - \mathbf{m}_k^{(i)})(\tilde{\mathbf{m}}_k^{(\ell)} - \mathbf{m}_k^{(i)})^T \right] \tag{4.28}$$

In case the number of components is still deemed too large, we can then impose a hard limit on the total number of components to consider, and only retain the J_{max} components with the highest weights, also referred to as capping. A very important consideration is that for the prune/cap processes, the weights need to be renormalised. Last, but not least, the multi-target state estimate can be extracted by selecting the means of the Gaussians that have weights greater than some threshold.

The GM-PHD filter can be extended to non-linear target models, where the state transition and measurement relationships are expressed as

$$\mathbf{x}_k = \boldsymbol{\varphi}_{k|k-1}(\mathbf{x}_{k-1}, \mathbf{w}_k), \quad \mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \epsilon_k), \tag{4.29}$$

with known non-linear functions $\boldsymbol{\varphi}_{k|k-1}$ and $\mathbf{h}_k$, zero-mean process noise $\mathbf{w}_k$ and measurement noise ϵ_k, and covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively.

In this non-linear formulation, the single-target transition density $f_{k|k-1}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ and measurement likelihood $g_k(\mathbf{z}_k \mid \mathbf{x}_k)$ are induced by the corresponding non-linear functions $\boldsymbol{\varphi}_{k|k-1}$ and $\mathbf{h}_k$ together with their respective noise distributions. That is, $f_{k|k-1}$ and g_k retain the same role as in the linear GM-PHD recursion, representing the probabilistic mappings from previous states to predicted states and from states to measurements.

The GM-PHD prediction and update steps are then applied using these densities, with Gaussian mixture approximations propagated through the non-linear transformations using either the extended Kalman filter (EKF) or unscented Kalman filter (UKF) approach as outlined in [28], Appendix B.1. The remaining pruning, merging, and capping procedures remain unchanged from the linear case.

4.2.2. Numerical Example

To illustrate the functionality of the filter, a numerical example[1] following the parameters set in ([49], Chapter 3.8, p.111) is presented, in which a maximum of 10 targets over a timespan of $100\,\text{s}$ ($\equiv$ 100 time-steps k with $\Delta t = 1\text{s}$) are simulated in a two dimensional scenario. The ground truth tracks can be seen in Figure 4.2 which shows objects moving in straight lines where the start/stop symbols

[1] All models used for the results compiled in the report have been fully describeed in subsection 4.5.4

i.e. circle/square represent the birth/death of targets at different times of the simulation, within the 2D region of interest $x \in (-1000, 1000)\,\mathrm{m}$, $y \in (-1000, 1000)\,\mathrm{m}$.

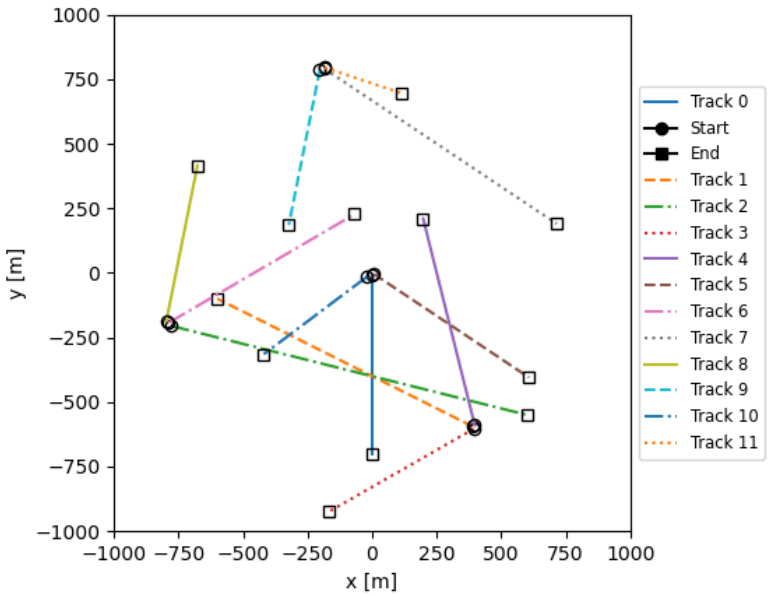


Figure 4.2: Ground truth trajectories in xy plane

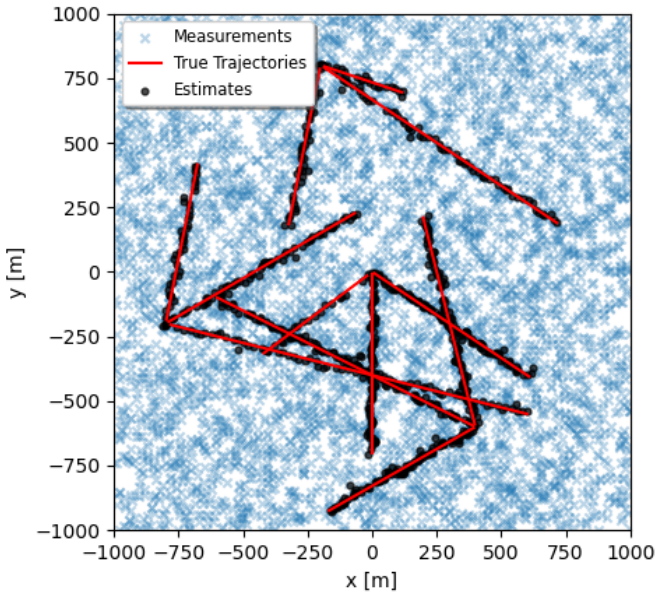
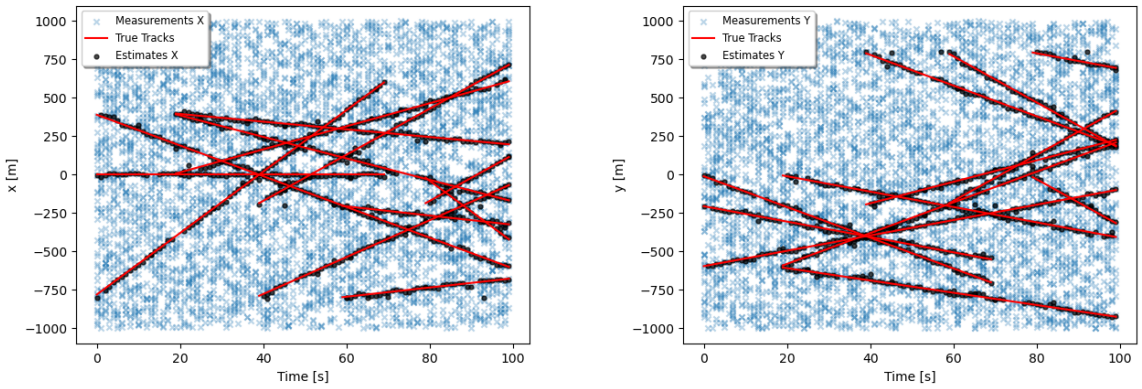


Figure 4.3: PHD filter estimates and true tracks in xy coordinates for single run simulation with random seed = 0.



(a) PHD filter x-coordinates estimates vs. time

(b) PHD filter y-coordinates estimates vs. time

Figure 4.4: Time series of the PHD filter estimates, ground truth tracks and measurements in cartesian coordinates for single run simulation with random seed = 0.

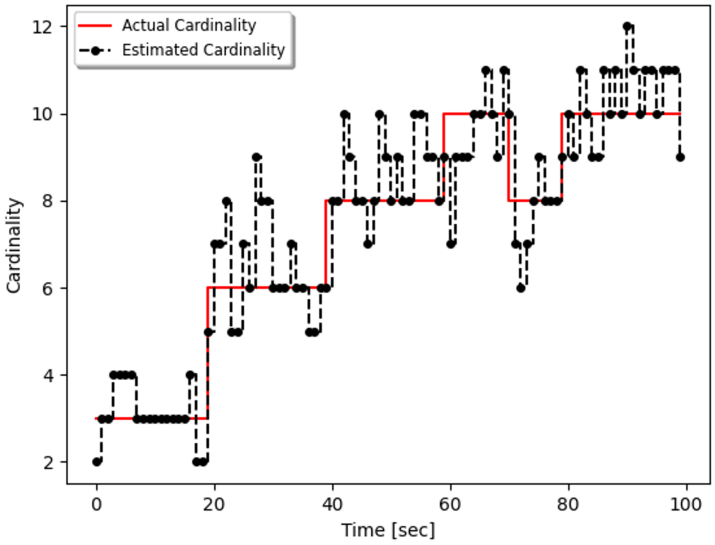


Figure 4.5: Estimated vs. true cardinality for the PHD Filter for single run simulation with random seed = 0.

The PHD filter estimates and performance for a single simulation run are presented in Figure 4.3 - Figure 4.6, where in Figure 4.3 the PHD estimates are plotted in Cartesian space against the true trajectories and the measurements. Figure 4.4 represents the time series plot of the same single run from which we can observe the modelled birth/deaths of the targets as well as the qualitative PHD accuracy performance. Furthermore, for this same run quantitative results as to the estimated cardinality performance of the PHD are presented in Figure 4.5 and the OSPA metrics are presented in Figure 4.6. Before proceeding with the analysis of the results and the introduction of Figure 4.7, representing the average cardinality error of a Monte-Carlo simultaion of 1000 runs on the same target trajectories but with independent measurements and clutter for each run, and Figure 4.8 representing the average OSPA metrics of the same Monte-Carlo simulation, we need to further elaborate on the concept of OSPA metrics.

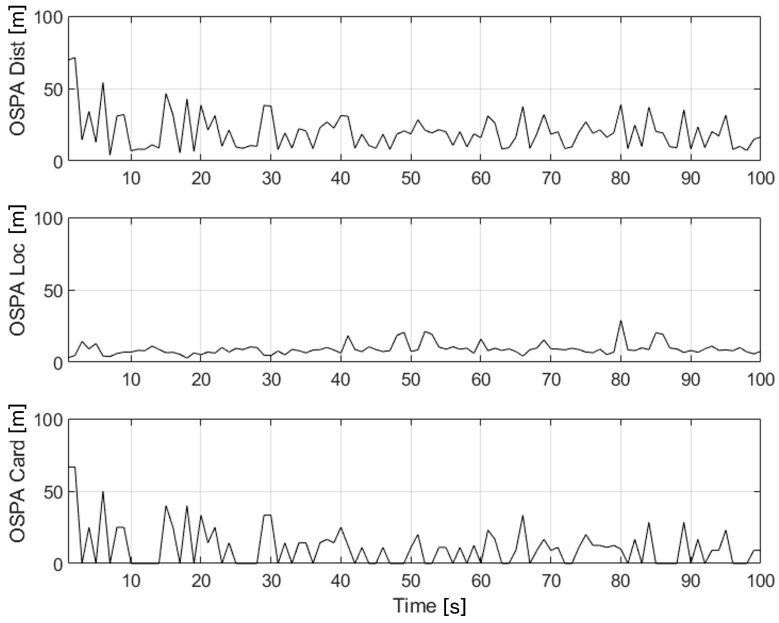


Figure 4.6: OSPA metrics with parameters $c = 100$, $p = 1$ for single run simulation with timestep (k) = 1s and random seed = 0.

OSPA Metric

The Optimal Sub-Pattern Assignment (OSPA) metric is a fundamental performance measure for multi-target tracking systems that provides a unified framework for evaluating both localization accuracy and cardinality estimation errors simultaneously. Background on the introduction and development of this metric is provided in subsection 2.2.2, the following discussion concerns the computation methodology for this metric.

The OSPA distance between two finite sets $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, with $|X| \geq |Y|$ is defined as:

$$d_p^{(c)}(X,Y) = \left(\frac{1}{n} \left[\min_{\pi} \sum_{i=1}^{m} d^{(c)}(x_i, y_{\pi(i)})^p + c^p(n-m) \right] \right)^{1/p} \tag{4.30}$$

Where:

- c is the cut-off parameter (in meters) - maximum penalty for a localization error.
- p is the order parameter (typically $p = 1$ or $p = 2$).
- $n = \max(|X|, |Y|)$ is the cardinality of the larger set, such that if the cardinality of Y is larger than that of X we compute $d_p^{(c)}(Y, X)$ instead ensuring that $m \leq n$.
- $d^{(c)}(x,y) = \min\left(c, d\left(\mathbf{x}_i, \mathbf{y}_{\pi(i)}\right)\right)$ is the cut-off distance, where $d\left(\mathbf{x}_i, \mathbf{y}_{\pi(i)}\right)$ represents a distance metric in the single target space, for instance the Euclidean distance between the points.
- π represents the optimal assignment minimizing the total cost, with $\pi \in \prod_n$ where $\prod_n$ represents the set of permutations on $\{1, \ldots, n\}$.

Equation 4.30 is referred to as the p-th order OSPA metric with cut-off c between X and Y, or simply OSPA distance $\equiv$ total OSPA, and decomposes into two fundamental components - localisation and cardinality.

On one hand, the localization error captures the accuracy of position estimates for correctly detected targets:

$$\text{Localization OSPA} = \left(\frac{1}{n} \min_{\pi} \sum_{i=1}^{m} d^{(c)}(x_i, y_{\pi(i)})^p \right)^{1/p} \tag{4.31}$$

This component measures how well the filter estimates the positions of targets that are successfully tracked. Lower values indicate better localization accuracy.

On the other hand, the cardinality error penalizes discrepancies in the number of estimated targets:

$$\text{Cardinality OSPA} = \left(\frac{1}{n} c^p (n - m) \right)^{1/p} \tag{4.32}$$

This component accounts for when the estimated cardinality is underestimated i.e. undercount targets or when the estimated cardinality is overestimated i.e. over-count targets. The penalty is proportional to the cardinality difference and is capped by the cut-off parameter c.

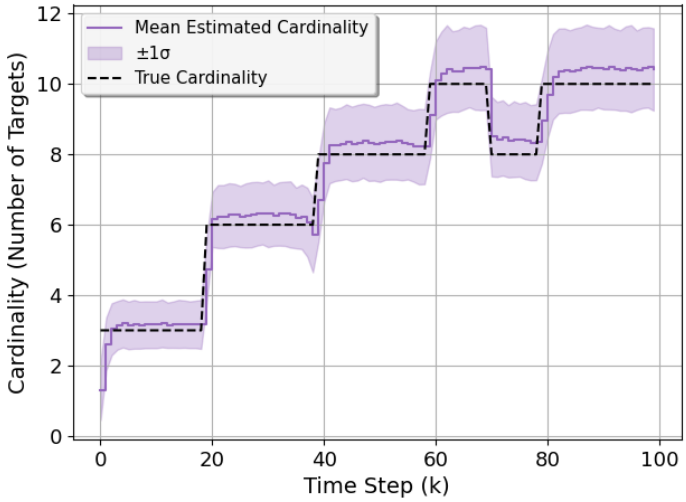


Figure 4.7: Average estimated cardinality vs time over 1000 MC runs

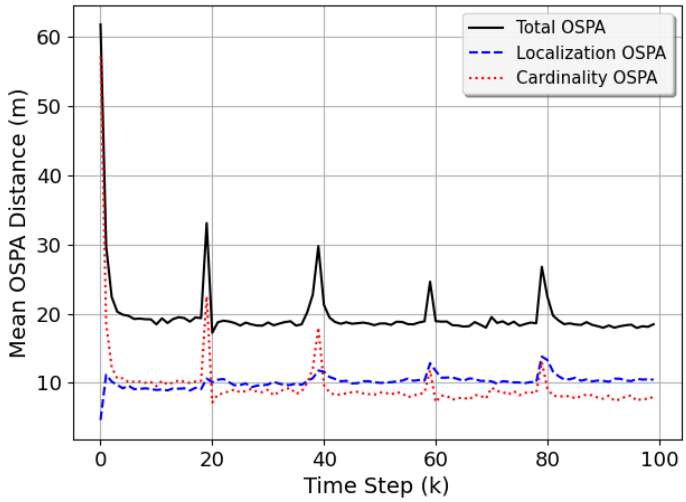


Figure 4.8: Average OSPA metrics with parameters $c = 100$, $p = 1$, over 1000 MC runs with time step (k) = 1s.

Discussion of the results

The first general observation is that the filter seems to manage to find all the tracks i.e. all the true trajectories are covered, which can be seen from both Figure 4.3 and Figure 4.4. However, this is disproven by Figure 4.5, which shows the cardinality distribution over one run. Indeed, as expected from theory the PHD filter suffers from inaccuracies in the cardinality estimations as it tends to over and under-estimate the cardinality at consistently over during the simulation. It is the latter that indicates that at some times k not all tracks are found by the filter, and although it might ”re-acquire” the ”lost” track later on this erratic behaviour is unwanted. On the other, hand from Figure 4.7 it can be seen that on average over many runs the mean estimated cardinality of the filter is very close to the true cardinality with one standard deviation representing the loss/gain of one existing/non-existing target.

From Figure 4.6 and Figure 4.8 we can analyse the OSPA metrics performance. The first observation is that at the start of the simulation the OSPA cardinality error is around 60 m for both the ”one run” and ”1000 runs” simulations and is also the peak error during the entire simulation. This transient behaviour is characteristic of multi-target filters during initialisation and as such will not be discussed further in any subsequent result graphs. From Figure 4.6 we can observe that the localisation error is in the range $[10, 20]\,\mathrm{m}$, while the OSPA cardinality error is behaving erratically and is the largest contributor to the total OSPA error (OSPA Dist). Interestingly enough when running the MC 1000 runs simulation, we can observe distinct spikes in OSPA card at $k \approx 20, 40, 60, 80$ which coincides with the appearance/death of new/existing targets, but in between appearances/deaths the cardinality error averages out. This has somewhat interesting implication, as this shows that by running parallel PHD filters and averaging the results out would lead to a somewhat marginally more accurate PHD filter result. Local OSPA cardinality peaks around the birth/death of targets can be expected from intuition and will be further demonstrated by the more advanced MTT filters implemented in this research as discussed in chapter 5.

4.3. Labelled Multi Bernoulli

4.3.1. Theory

The LMB filter was introduced by Reuter et al. [65] and can be interpreted as an efficient approximation of the δ-GLMB filter in [77]. However, before discussing the prediction and update components of the LMB filter, we need to introduce several base concepts. We begin with the concept of a Bernoulli RFS, an RFS with the property that its cardinality distribution $\rho(n)$ is Bernoulli [67]. As such a Bernoulli RFS can either be empty (with probability $1-r$) or have one element (with probability r) distributed over the state space $\mathbb{X}$ according to PDF $p(\mathbf{x})$. The FISST PDF of a Bernoulli RFS X is thus given by

$$\pi(X) = \begin{cases} 1-r & \text{if } X = \emptyset \\ r \cdot p(X) & \text{if } X = \{\mathbf{x}\} \end{cases} \tag{4.33}$$

By extension a multi-Bernoulli RFS Y is given by the union of the M independent Bernoulli RFSs $X^{(i)}$ i.e. $Y = \bigcup_{i=1}^{M} X^{(i)}$ and is completely defined by the parameter set $\{(r^{(i)}, p^{(i)})\}_{i=1}^{M}$. Simply put a multi-Bernoulli RFS is a set with each of its elements being a Bernoulli RFS. Furthermore, the probability density of a multi-Bernoulli RFS is given by ([47], p.368)

$$\pi(Y) \equiv \pi(\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}) = \prod_{j=1}^{M} \left(1 - r^{(j)}\right) \times \sum_{1 \le i_1 \ne \ldots, \ne i_n \le M} \left[\prod_{j=1}^{n} \left(\frac{r^{(i_j)} p^{(i_j)}(\mathbf{x_j})}{1 - r^{(i_j)}} \right) \right] \tag{4.34}$$

Remark on the difference from the previous example.

In the earlier example in Figure 2.6, only the existence probabilities of the targets were considered, ignoring the continuous state distributions $p_i(\mathbf{x})$. In that case, the Multi-Bernoulli probabilities reduce to simple products of the Bernoulli existence probabilities:

$$
\begin{aligned}
P(\varnothing) &= (1-r_1)(1-r_2),\\
P(\{\text{target}_1\}) &= r_1(1-r_2),\\
P(\{\text{target}_2\}) &= r_2(1-r_1),\\
P(\{\text{target}_1, \text{target}_2\}) &= r_1 r_2.
\end{aligned}
$$

In contrast, the full Multi-Bernoulli PDF in Equation 4.34 incorporates both the existence probabilities and the continuous state distributions. Assuming a two-targets 2D example i.e. ($M = 2$) and $Y = \{\mathbf{x}_1, \mathbf{x}_2\}$, the sum in Equation 4.34 goes over all combinations of $i_1 \neq i_2$:

$$
(i_1, i_2) = (1, 2) \quad \text{and} \quad (i_1, i_2) = (2, 1).
$$

Plugging in $M = 2$, the Multi-Bernoulli PDF evaluates as:

$$
\pi(\{\mathbf{x}_1, \mathbf{x}_2\}) = (1-r_1)(1-r_2)\left[\frac{r_1 p_1(\mathbf{x}_1)}{1-r_1} \cdot \frac{r_2 p_2(\mathbf{x}_2)}{1-r_2} + \frac{r_2 p_2(\mathbf{x}_1)}{1-r_2} \cdot \frac{r_1 p_1(\mathbf{x}_2)}{1-r_1}\right].
$$

Simplifying the terms:

$$
\begin{aligned}
(1-r_1)(1-r_2) \cdot \frac{r_1 p_1(\mathbf{x}_1)}{1-r_1} \cdot \frac{r_2 p_2(\mathbf{x}_2)}{1-r_2} &= r_1 r_2 p_1(\mathbf{x}_1) p_2(\mathbf{x}_2),\\
(1-r_1)(1-r_2) \cdot \frac{r_2 p_2(\mathbf{x}_1)}{1-r_2} \cdot \frac{r_1 p_1(\mathbf{x}_2)}{1-r_1} &= r_1 r_2 p_1(\mathbf{x}_2) p_2(\mathbf{x}_1),
\end{aligned}
$$

so that the final Multi-Bernoulli PDF for this two-target set is:

$$
\pi(\{\mathbf{x}_1, \mathbf{x}_2\}) = r_1 r_2 \big(p_1(\mathbf{x}_1) p_2(\mathbf{x}_2) + p_1(\mathbf{x}_2) p_2(\mathbf{x}_1)\big),
$$

which matches the expected structure of Equation 4.34.

The Multi-Bernoulli probabilities for the chosen points are then:

$$
\begin{aligned}
\pi(\varnothing) &= (1-r_1)(1-r_2),\\
\pi(\{\mathbf{x}_1\}) &= r_1(1-r_2) p_1(\mathbf{x}_1),\\
\pi(\{\mathbf{x}_2\}) &= r_2(1-r_1) p_2(\mathbf{x}_2),\\
\pi(\{\mathbf{x}_1, \mathbf{x}_2\}) &= r_1 r_2 \big(p_1(\mathbf{x}_1) p_2(\mathbf{x}_2) + p_1(\mathbf{x}_2) p_2(\mathbf{x}_1)\big).
\end{aligned}
$$

Let the existence probabilities be $r_1 = 0.8$ and $r_2 = 0.4$. Substituting these values (but keeping the continuous PDFs symbolic) gives

$$
\begin{aligned}
\pi(\varnothing) &= 0.12,\\
\pi(\{\mathbf{x}_1\}) &= 0.48\, p_1(\mathbf{x}_1),\\
\pi(\{\mathbf{x}_2\}) &= 0.08\, p_2(\mathbf{x}_2),\\
\pi(\{\mathbf{x}_1, \mathbf{x}_2\}) &= 0.32 \big(p_1(\mathbf{x}_1) p_2(\mathbf{x}_2) + p_1(\mathbf{x}_2) p_2(\mathbf{x}_1)\big),
\end{aligned}
$$

making it clear that the continuous state densities remain functions of $\mathbf{x}$, while the existence probabilities scale their contribution to the overall multi-target density.

In Eq. (4.34), the sum in the second term is taken over all permutations of $n \leq M$ of all the independent Bernoulli RFSs $X^{(i)}$ of Y. Moreover, the numerator of the product within the second represents the probability density that the Bernoulli components with indices $i_1, \ldots, i_n$ generate the realisations $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

On the other hand, the first term product is a constant that cancels out with the denominators inside the sum to give the probability that the leftover Bernoulli components with indices $\{1, \ldots, M\} - \{i_1, \ldots, i_n\}$ generate null realisations i.e. $\emptyset$. In [47] it is shown that this leads to the expected result of

$$\int \pi(Y) dY = 1 \tag{4.35}$$

Furthermore, by neglecting the spatial distribution term in Equation 4.34, the cardinality of a multi-Bernoulli RFS is obtained as

$$\rho(n) = \prod_{j=1}^{M} \left(1 - r^{(j)}\right) \times \sum_{1 \leq i_1 \neq \ldots, \neq i_n \leq M} \left[\prod_{j=1}^{n} \left(\frac{r^{(i_j)}}{1 - r^{(i_j)}} \right) \right] \tag{4.36}$$

The next concept introduced is that of labelled random finite sets [65, 48].

Recall that the PHD filter maintains an unlabelled set of target estimates at each time step, meaning it does not output trajectories or uniquely identified tracks of individual objects. Likewise, the standard Multi-Bernoulli RFS is also unlabelled and therefore suffers the same limitation. For SSA applications, however, maintaining continuous tracks of individual resident space objects is essential for cataloguing, conjunction assessment, and custody maintenance. This shortcoming can be addressed by augmenting each target state with a unique label, resulting in a labelled RFS formulation that enables consistent target identification over time.

Hence, to be able to estimate the trajectory of a single tracked object with state $\mathbf{x} \in \mathbb{X}$ in a multi-target problem, we attach to it a label $\ell \in \mathbb{L}$, where $\mathbb{L} = \{\alpha_i : i \in \mathbb{N}\}$ is a discrete label space with distinct elements. Thus, we can define a labelled RFS with state space $\mathbb{X}$ and label space $\mathbb{L}$ as a RFS on $\mathbb{X} \times \mathbb{L}$ with distinct labels, i.e. a finite-set-valued random variable on $\mathbb{X} \times \mathbb{L}$ such that all realisations of the labels are unique. Additionally, labels for individual targets are ordered pairs of integers $\ell = (k, i)$, where k is the time of birth and $i \in \mathbb{N}$ is a unique index to distinguish between targets born at the same time. The label space $\mathbb{L}_k \equiv \{k\} \times \mathbb{N}$ denotes the label space of new targets born at t_k, each of which has state $\mathbf{x} \in \mathbb{X} \times \mathbb{L}_k$. Therefore the label space for all targets at t_k i.e. new births and survivals is $\mathbb{L}_{0:k}$ and is constructed recursively using $\mathbb{L}_{0:k} = \mathbb{L}_{k-1} \cup \mathbb{L}_k$.

Before proceeding further, we need to (re-)introduce some notations/abbreviations which will be used throughout this research.

Inner Product

$$\langle f, g \rangle \equiv \int f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}$$

Multi-Object Exponential Notation with real valued function h

$$h^X = \begin{cases} 1 & \text{if } X = \emptyset \\ \prod_{\mathbf{x} \in X} h(\mathbf{x}) & \text{if } X \neq \emptyset \end{cases}$$

Generalised Kronecker delta function supporting sets, vectors and integers

$$\delta_Y(X) \equiv \begin{cases} 1 & \text{if } X = Y \\ 0 & \text{otherwise} \end{cases}$$

Inclusion Function

$$1_Y(X) \equiv \begin{cases} 1 & \text{if } X \subseteq Y \\ 0 & \text{otherwise} \end{cases}$$

Additionally in case X is a singleton i.e. $X = \{\mathbf{x}\}$ the notation $1_Y(\mathbf{x})$ is used instead of $1_Y(\{\mathbf{x}\})$. Moreover, when no confusion may arise, the following abbreviation can be used for the label space for all targets $\mathbb{L}_{0:k} \equiv \mathbb{L}$, and the cardinality of a set is denoted by $|\cdot|$.

$\mathcal{L}(X) = \{\mathcal{L}(\mathbf{x}) : \mathbf{x} \in X\}$ gives the set of labels of a labelled RFS (LRFS) X, where $\mathcal{L} : \mathbb{X} \times \mathbb{L} \to \mathbb{L}$ is the transformation defined by $\mathcal{L}((\mathbf{x}, \ell)) = \ell$. To ensure that the labels of each element in the LRFS must be unique, we introduce the distinct label indicator $\Delta(\cdot)$

$$\Delta(X) = \delta_{|X|}(|\mathcal{L}(X)|) = 1 \tag{4.37}$$

On the other hand, $\mathcal{V}(X) = \{\mathcal{V}(\mathbf{x}) : \mathbf{x} \in X\} = Y$ is the unlabelled version of the LRFS X, where $\mathcal{V} : \mathbb{X} \times \mathbb{L} \to \mathbb{X}$ is the transformation defined by $\mathcal{V}((\mathbf{x}, \ell)) = \mathbf{x}$. One of the key characteristics of LRFS is that its cardinality distribution is equivalent to the cardinality distribution of its unlabelled version [65], i.e. $|X| = |Y|$.

It then follows that similarly to a MB-RFS the LMB-RFS can be fully described using a parameter set $\{(r^{(\zeta)}, p^{(\zeta)}) : \zeta \in \Psi\}$, with index set $\Psi \in \mathbb{N}$, and if there exists at least one component $(r^{(\zeta)}, p^{(\zeta)})$ which returns a non-empty set, we can attach a label using the transformation $\alpha(\zeta)$ defined as a general one-to-one mapping $\alpha : \Psi \to \mathbb{L}$. A procedure for sampling from a LRFS can be found in [77], and the PDF of a labelled multi-Bernoulli RFS on the space $\mathbb{X} \times \mathbb{L}$ is given by

$$\begin{aligned}\pi(\{(\mathbf{x}_1, \ell_1), \ldots, (\mathbf{x}_n, \ell_n)\}) = \delta_n(|\{\ell_1, \ldots, \ell_n\}|) \prod_{\zeta \in \Psi} (1 - r^{(\zeta)}) \\ \times \prod_{j=1}^{n} \frac{1_{\alpha(\Psi)}(\ell_j)\, r^{(\alpha^{-1}(\ell_j))}\, p^{(\alpha^{-1}(\ell_j))}(\mathbf{x}_j)}{1 - r^{(\alpha^{-1}(\ell_j))}}\end{aligned} \tag{4.38}$$

The notation in Eq. (4.38) can be further simplified by realising that even though α is a general mapping, we can assume it to be an identity mapping. Thus, the PDF of a LMB RFS X with set parameters $\{(r^{(\ell)}, p^{(\ell)})\}_{\ell \in \mathbb{L}}$ is given by

$$\pi(X) = \Delta(X) w(\mathcal{L}(X)) p^X \tag{4.39}$$

where

$$w(\mathcal{L}(X)) = \prod_{i \in \mathbb{L}} \left(1 - r^{(i)}\right) \prod_{\ell \in \mathbb{L}} \left(\frac{1_{\mathbb{L}}(\ell) r^{(\ell)}}{1 - r^{(\ell)}}\right)$$

$$p(\mathbf{x}, \ell) = p^{(\ell)}(\mathbf{x})$$

The LMB RFS described in Eq. (4.38) is actually a special case of the generalised labelled multi-Bernoulli RFS (GLMB-RFS) [77]. The latter RFS can be defined as a labelled RFS with state space $\mathbb{X}$ and (discrete) label space $\mathbb{L}$, whose distribution is given by

$$\pi(X) = \Delta(X) \sum_{c \in \mathbb{C}} w^{(c)}(\mathcal{L}(X)) \left[p^{(c)}\right]^X \tag{4.40}$$

where $\mathbb{C}$ is a discrete index set, and the weights and spatial distributions respectively satisfy the normalisation conditions

$$\sum_{L \subseteq \mathbb{L}} \sum_{c \in \mathbb{C}} w^{(c)}(L) = 1, \tag{4.41}$$

$$\int p^{(c)}(\mathbf{x}, \ell)\, d\mathbf{x} = 1. \tag{4.42}$$

The index $c \in \mathbb{C}$ can be interpreted as labelling different hypotheses about the target set, in a manner similar to Multiple Hypothesis Tracking (MHT). Each hypothesis c has an associated weight $w^{(c)}(L)$,

which represents the probability of that particular hypothesis given a set of labels L, and an associated spatial distribution $p^{(c)}(\mathbf{x}, \ell)$ describing the kinematic state of each labelled target

Equation 4.41 then expresses the multi-target density $\pi(X)$ as a weighted mixture over all hypotheses:

$$\pi(X) = \Delta(X) \sum_{c \in \mathbb{C}} w^{(c)}(\mathcal{L}(X)) \left[p^{(c)}\right]^X,$$

where $\Delta(X)$ enforces distinct labels for each target in X. The normalization condition

$$\sum_{L \subseteq \mathbb{L}} \sum_{c \in \mathbb{C}} w^{(c)}(L) = 1$$

ensures that the total weight across all hypotheses sums to one, i.e., that the set of hypotheses forms a valid probability distribution over all possible label sets. Similarly, the spatial distributions are individually normalized:

$$\int p^{(c)}(\mathbf{x}, \ell)\, d\mathbf{x} = 1,$$

so that each target's state distribution is a proper probability density. In this way, the labelled multi-target density can be seen as a probabilistic mixture over different hypotheses, with the weights and state densities satisfying standard normalization conditions.

The LMB RFS is thus a GLMB which satisfies

$$p^{(c)}(\mathbf{x}, \ell) = p^{(\ell)}(\mathbf{x}) \tag{4.43}$$

$$w^{(c)}(L) = \prod_{i \in \mathbb{L}} \left(1 - r^{(i)}\right) \prod_{\ell \in L} \frac{\mathbf{1}_L(\ell)\, r^{(\ell)}}{1 - r^{(\ell)}} \tag{4.44}$$

for which the subscript (c) can be omitted. Hence, the fundamental difference consists in the mixture against the single component representation, where the sum over $c \in \mathbb{C}$ facilitates the propagation of multiple hypotheses, involving different sets of track labels, as opposed to the LMB process, where it is only possible to propagate the uncertainty of a single set of track labels. As such, one may draw the following analogy: the GLMB explicitly maintains multiple label-set hypotheses (analogous to MHT), whereas the LMB provides a single-component summary that approximates those hypotheses (analogous to JPDA or GNN).

Another special case of a GLMB RFS is the δ-generalised labelled multi-Bernoulli (δ-GLMB) with state space $\mathbb{X}$, (discrete) label space $\mathbb{L}$ and

$$\begin{aligned} \mathbb{C} &= \mathbb{F}(\mathbb{L}) \times \Xi \\ w^{(c)}(L) &= w^{(I,\xi)} \delta_I(L) \\ p^{(c)} &= p^{(I,\xi)} = p^{(\xi)} \end{aligned}$$

where $\mathcal{F}(\cdot)$ denotes the collection of all finite subsets of the space $(\cdot)$, Ξ is a discrete space typically representing the history of track to measurements associations, ξ are the realisations of Ξ, and I denotes a set of track labels. Hence, the density of the δ-GLMB RFS is given by

$$\pi(X) = \Delta(X) \sum_{(I,\xi) \in \mathcal{F}(\mathbb{L}) \times \Xi} \left[w^{(I,\xi)} \delta_I(\mathcal{L}(X)) \left[p^{\xi}\right]^X \right] \tag{4.45}$$

and its cardinality distribution by

$$\rho(n) = \sum_{(I,\xi)\in\mathcal{F}(\mathbb{L})\times\Xi} w^{(I,\xi)} \tag{4.46}$$

Additionally, the PHD of the corresponding unlabelled RFS is given by

$$v(\mathbf{x}) = \sum_{\ell\in\mathbb{L}} \left[\sum_{(I,\xi)\in\mathcal{F}(\mathbb{L})\times\Xi} w^{(I,\xi)} 1_I(\ell) p^{(\xi)}(\mathbf{x}, l) \right] \tag{4.47}$$

where the inclusion function $1_I(\ell)$ makes the summand of the inner sum zero when $\ell \notin I$. Moreover, the existence probability of $r^{(\ell)}$ can be obtained using

$$r^{(\ell)} = \sum_{(I,\xi)\in\mathcal{F}(\mathbb{L})\times\Xi} w^{(I,\xi)} 1_I(\ell) \tag{4.48}$$

where the discrete index pairs $1_I(\ell)$ will be referred to as hypothesis.

In summary, the LMB, GLMB, and δ-GLMB filters share a common labelled RFS structure but differ primarily in how they represent and manage multiple hypotheses. The GLMB provides the most general and expressive formulation, maintaining a full mixture of label-set hypotheses. The δ-GLMB introduces an explicit parameterisation of these hypotheses that makes implementation more tractable but still incurs significant computational cost. The LMB, on the other hand, collapses the hypothesis mixture into a single labelled multi-Bernoulli component, preserving the ability to maintain object identity while achieving higher computational efficiency. A broader background on the development and introduction of these filters is summarised in section 2.2.

For SSA applications, characterised by large numbers of resident space objects, this trade-off is particularly important. While the GLMB and δ-GLMB filters offer the highest representational accuracy, their hypothesis growth can render them less suitable for sustained large-scale tracking. Conversely, the LMB retains essential track-labelling capability and consistent cardinality statistics while remaining scalable for catalogue maintenance and multi-sensor updates. For these reasons, the LMB is selected for implementation in this research.

4.3.2. Implementation Approach

Now that all the base concepts have been discussed, this subsection introduces the implementation approach for the LMB filter. The general overview schematic of the approach is depicted in Figure 4.9 courtesy of [65], and is separated into three main phases: the prediction, the grouping and the updating phases.

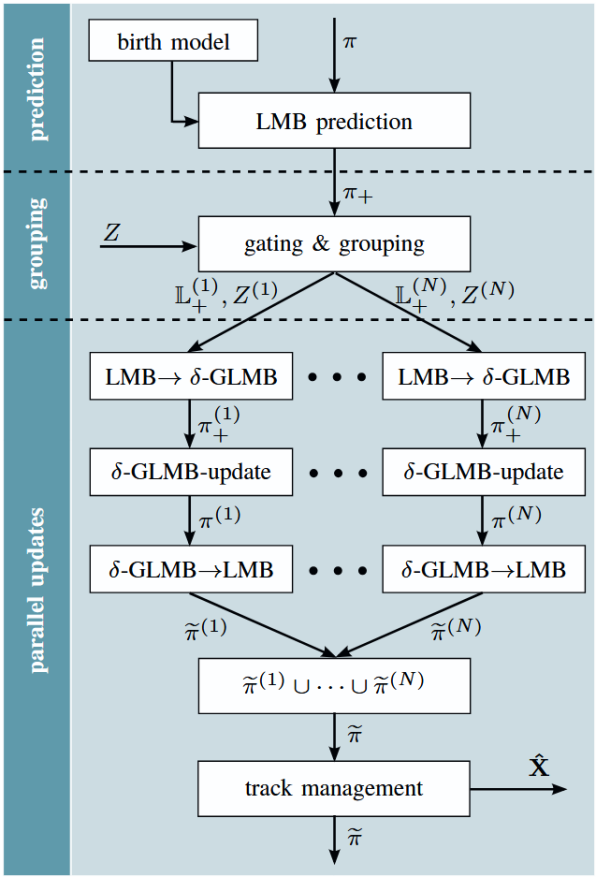


Figure 4.9: LMB filter schematic overview [65]

Prediction

Given the posterior LMB distribution π with state space $\mathbb{X}$ and label space $\mathbb{L}$,

$$\pi = \left\{\left(r^{(\ell)}, p^{(\ell)}\right)\right\}_{\ell \in \mathbb{L}}, \tag{4.49}$$

the prediction to the time of next measurement follows an LMB distribution π_+ with state space $\mathbb{X}$ and (finite) label space $\mathbb{L}_+ = \mathbb{L} \cup \mathbb{B}$ given by

$$\pi_+ = \left\{\left(r_{+,S}^{(\ell)}, p_{+,S}^{(\ell)}\right)\right\}_{\ell \in \mathbb{L}} \cup \left\{\left(r_B^{(\ell)}, p_B^{(\ell)}\right)\right\}_{\ell \in \mathbb{B}} = \left\{\left(r_+^{(\ell)}, p_+^{(\ell)}\right)\right\}_{\ell \in \mathbb{L}_+}, \tag{4.50}$$

where the first term is a LMB RFS representing the surviving labelled Bernoulli tracks of the previous time step, and the second term represents the a priori LMB birth components. In the survival RFS, the probability of survival and the probability density function are respectively given by

$$r_{+,S}^{(\ell)} = \eta_S(\ell)\, r^{(\ell)}, \tag{4.51}$$

$$p_{+,S}^{(\ell)} = \langle p_S(\cdot, \ell)\, f(x|\cdot, \ell),\, p(\cdot, \ell)\rangle / \eta_S(\ell), \tag{4.52}$$

where $\eta_S(\ell)$ is the survival probability of track ℓ and $f(\mathbf{x}|\cdot, \ell)$ is the single track transition density for track ℓ. The predicted label for the surviving tracks is the same as the previous label, while for the new birth tracks $\ell \in \mathbb{B}$ are new distinct labels such that $\mathbb{L} \cap \mathbb{B} = \emptyset$

Gating - Grouping

In order to perform the update, it is necessary to transform the LMB to δ-GLMB, i.e. performing a full δ-GLMB update and collapsing back to a matching LMB, which can be done using Equation 4.45 and Equation 4.47. However, considering all permutations of track to label association can be computationally expensive. To mitigate this, a gating and grouping strategy can be employed following the scheme

proposed in [65], where the gating restricts the number of measurements-to-track associations by only considering measurements within the validation gates of the tracks, while grouping partitions the LMB tracks and measurements into smaller, disjoint sets, which can be updated in parallel. Implementing this scheme suggested in [65] reduces the computational complexity, while only marginally affecting accuracy, particularly when groups are well separated spatially. It can be noted that the standard LMB $\rightarrow \delta-$GLMB update is analogous to the gating/grouping scheme discussed in this section, with a single group.

The first step of the procedure consists of generating a partition of the predicted LMB parameters $P_{\mathbb{L}} = \{\mathbb{L}_+^{(1)}, \ldots, \mathbb{L}_+^{(N)}\}$ and received measurement set $P_Z = \{Z^{(0)}, Z^{(1)}, \ldots, Z^{(N)}\}$, with the following properties,

$$\mathbb{L}_+ = \bigcup_{n=1}^{N} \mathbb{L}_+^{(n)}, \qquad \mathbb{L}_+^{(n)} \cap \mathbb{L}_+^{(m)} = \emptyset \quad \forall (n \neq m);$$

$$Z = \bigcup_{n=0}^{N} Z^{(n)}, \qquad Z^{(n)} \cap Z^{(m)} = \emptyset \quad \forall (n \neq m);$$

Therefore a grouping $\mathcal{G}^{(n)}$ is defined as a set of pairs

$$\mathcal{G}^{(n)} = \left(\mathbb{L}_+^{(n)}, Z^{(n)}\right) \in \left\{\left(\mathbb{L}_+^{(1)}, Z^{(1)}\right), \ldots, \left(\mathbb{L}_+^{(n)}, Z^{(n)}\right)\right\} \tag{4.53}$$

It should be noted that $Z^{(0)}$ is the set of measurements which are not assigned to any target labels. Using this definition of grouping for generating a particular grouping for the parallel group updates, we first generate a tentative group $\widetilde{\mathcal{G}}^{(\ell)}$, i.e. a group formed with temporary/uncertain membership,

$$\widetilde{\mathcal{G}}^{(\ell)} = \left(\{\ell\}, \{\mathbf{z} : d_{MHD}(\hat{\mathbf{z}}^{(\ell)}, \mathbf{z}) < \sqrt{\gamma}\}\right) \tag{4.54}$$

where $d_{MHD}(\hat{\mathbf{z}}^{(\ell)}$ is the Mahalanobis distance between the predicted measurement for track ℓ and the received measurement $\mathbf{z} \in Z$, and γ is the gating distance threshold computed using Chi-squared cumulative distribution corresponding to the desired σ-gate size for gating of measurements from tracks. The following step consists of merging tentative groups with common measurements, i.e.

$$\widetilde{\mathcal{G}}^{(i,j)} = \left(\mathbb{L}_+^{(i)} \cup \mathbb{L}_+^{(j)}, Z^{(i)} \cup Z^{(j)}\right), \qquad Z^{(i)} \cap Z^{(j)} \neq \emptyset$$

and this is repeated until for all tentative groups there are no common measurements, yielding a total of N groups $\mathcal{G}^{(1)} \ldots, \mathcal{G}^{(N)}$ of tracks and associated measurements. A visual representation of grouping is shown in Figure 4.10, where the black dashed rectangles represent the abstract boundary of the groups. Intuitively this additional partitioning/approximation into groups works best when the label subsets are well separated spatially, as determined by the gating of the measurements, such that the partitions have negligible influence on each other during the update. This occurs when the birth distribution has reasonably small covariances, which can be ensured by using a measurement driven birth model or assuming static birth locations.

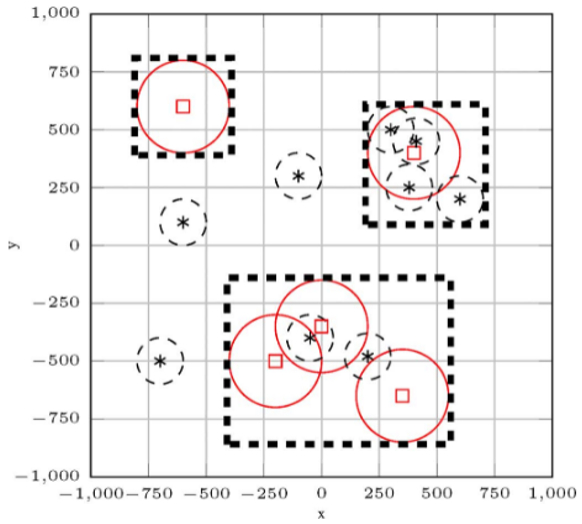


Figure 4.10: Example of partitioning into groups, with five tracks (red squares) and nine measurements (black stars), courtesy of [65]

Therefore, the predicted multi-target density can be rewritten as

$$\pi_+ = \bigcup_{i=1}^{N} \pi_+^{(i)} \qquad \text{with } \pi_+^{(i)} = \left\{ \left(r_+^{(\ell)}, p_+^{(\ell)} \right) \right\}_{\ell \in \mathbb{L}_+^{(i)}} \tag{4.55}$$

This concludes the grouping and gating needed for the parallel group updates, which consists of 3 sub-tasks:

1. Representation of the predicted LMB as δ-GLMB
2. Parallel δ-GLMB group updates
3. Approximation of the updated δ-GLMB as LMB

Hence, the predicted δ-GLMB for the i-th group of labels and measurements $\mathcal{G}^{(i)} = \left(\mathbb{L}_+^{(i)}, Z^{(i)} \right)$ is given by

$$\pi_+^{(i)} \left(\widetilde{X}_+^{(i)} \right) = \Delta \left(\widetilde{X}_+^{(i)} \right) \sum_{I_+ \in \mathcal{F}(\mathbb{L}_+^{(i)})} \left[w_{+,i}^{(I_+)} \, \delta_{I_+} \left(\mathcal{L} \left(\widetilde{\mathbf{X}}_+^{(i)} \right) \right) \, [p_+]^{\widetilde{\mathbf{X}}_+^{(i)}} \right] \tag{4.56}$$

with

$$w_{+,i}^{(I_+)} = \prod_{\ell \in \mathbb{L}_+^{(i)}} \left(1 - r_+^{(\ell)} \right) \prod_{\ell' \in I_+} \frac{\mathbf{1}_{\mathbb{L}_+^{(i)}}(\ell') \, r_+^{(\ell')}}{1 - r_+^{(\ell')}},$$

where $\widetilde{X}_+^{(i)}$ is the multi-target state for group (i). To compute this sum several methods exist, with the most computationally heavy being the generation of all possible combinations for a set of labels $\mathcal{L}_+^{(i)}$ and cardinalities $0, 1, \ldots, |\mathbb{L}_+^{(i)}|$. However, this results in the number of combinations for a set of track labels being $2^{|\mathbb{L}_+^{(i)}|}$ and the number of combinations for each cardinality given by the binomial coefficient $\mathcal{C}(|\mathbb{L}_+^{(i)}|, n) = |\mathbb{L}_+^{(i)}|! / \left(n! \left(|\mathbb{L}_+^{(i)}| - n \right)! \right)$. Thus, in case the number of targets is large, computing the sum becomes computationally unviable. In these cases it is possible to approximate the sum to its k most significant terms by the use of k-shortest paths algorithm [52].

The next step is the parallel δ-GLMB update for each group (i) given by:

$$\pi^{(i)}\left(\widetilde{X}^{(i)} \mid Z^{(i)}\right) = \Delta\left(\widetilde{X}^{(i)}\right) \times \sum_{(I_+,\theta)\in\mathcal{F}(\mathbb{L}_+^{(i)})\times\Theta_{I_+}} w^{(I_+,\theta)}(Z^{(i)})\,\delta_{I_+}(\mathcal{L}(\widetilde{X}^{(i)}))\,\left[p^{(\theta)}(\cdot \mid Z^{(i)})\right]^{\widetilde{X}^{(i)}} \quad (4.57)$$

for which Θ_{I_+} is the space of mappings $\theta : I_+ \to \{0, 1, \ldots, |Z^{(i)}|\} \mid (\theta(\iota) = \theta(\iota') > 0) \implies \iota = \iota'$ and

$$w^{(I_+,\theta)}(Z^{(i)}) \propto w_{+,i}^{(I_+)}\left[\eta_{Z^{(i)}}^{(\theta)}\right]^{I_+}$$

$$p^{(\theta)}(\mathbf{x}, \ell \mid Z^{(i)}) = \frac{p_{+,i}(\mathbf{x}, \ell)\psi_{Z^{(i)}}(\mathbf{x}, \ell; \theta)}{\eta_{Z^{(i)}}^{(\theta)}(\ell)}$$

$$\eta_{Z^{(i)}}^{(\theta)}(\ell) = \langle p_{+,i}(\mathbf{x}, \ell), \psi_{Z^{(i)}}(\cdot, \ell; \theta)\rangle$$

$$\psi_{Z^{(i)}}(\mathbf{x}, \ell; \theta) = \begin{cases} \frac{p_D(\mathbf{x},\ell)\, p_G\, g(\mathbf{z}_{\theta(\ell)}|\mathbf{x},\ell)}{\kappa(\mathbf{z}_{\theta(\ell)})}, & \text{if } \theta(\ell) > 0 \\ q_{D,G}(\mathbf{x}, \ell), & \text{if } \theta(\ell) = 0 \end{cases}$$

Note the dependency on the gating probability p_G as smaller gates increase the probability of missed detections $q_{D,G}(\mathbf{x}, \ell) = 1 - p_D(\mathbf{x}, \ell) \cdot p_G$. Similarly to the gating and grouping in the update the number of components grow exponentially with the number of track labels $|\mathbb{L}_+^{(i)}|$. As such, one solution is to truncate the solution space by only evaluating the M most significant hypotheses using a ranked assignment algorithm [56].

The last step consists of approximating the updated δ-GLMB as LMB across all groups $\mathcal{G}^{(i)}$ with

$$\pi^{(i)}(\cdot \mid Z^{(i)}) \approx \widetilde{\pi}^{(i)}(\cdot \mid Z^{(i)}) = \left\{\left(r^{(\ell,i)}, p^{(\ell,i)}\right)\right\}_{\ell\in\mathbb{L}_+^{(i)}}$$

where $r^{(\ell,i)}, p^{(\ell,i)}$ are computed according to

$$r^{(\ell)} = \sum_{(I_+,\theta)\in\mathcal{F}(\mathbb{L}_+)\times\Theta_{I_+}} w^{(I_+,\theta)}(Z)\,1_{I_+}(\ell)$$

$$p^{(\ell)}(\mathbf{x}) = \frac{1}{r^{(\ell)}} \sum_{(I_+,\theta)\in\mathcal{F}(\mathbb{L}_+)\times\Theta_{I_+}} w^{(I_+,\theta)}(Z)\,1_{I_+}(\ell)\,p^{(\theta)}(\mathbf{x}, \ell)$$

Consequently, the LMB approximation to the multi-target posterior can be assembled analogously to Eq. (4.55),

$$\pi(\cdot \mid Z) \approx \widetilde{\pi}(\cdot \mid Z) = \bigcup_{i=1}^{N}\left\{\left(r^{(\ell,i)}, p^{(\ell,i)}\right)\right\}_{\ell\,\in\,\mathbb{L}_+^{(i)}}$$

and in order to further decrease the number of considered tracks, while not suppressing the output of previously confirmed tracks due to a missed detection, the following track management scheme can be used:

$$\hat{X} = \left\{(\hat{\mathbf{x}}, \ell) : r_{\mathsf{max}}^{(\ell)} > \vartheta_u \wedge r^{(\ell)} > \vartheta_l\right\}$$

where $\hat{\mathbf{x}} = \arg\max_x(p^{(\ell)}(\mathbf{x}))$, and the track ℓ is kept if its maximum existence probability $r_{\mathsf{max}}^{(\ell)}$ has ever exceeded an upper threshold ϑ_u and its current existence probability is higher than a lower threshold ϑ_l.

4.4. Poisson Labelled Multi Bernoulli

The PMBM filter provides a solution for MTT by representing the multi-target posterior as a combination of a Poisson Point Process (PPP) for undetected targets i.e. potential births and a mixture of Multi-Bernoulli components for detected targets i.e. surviving tracks, thereby maintaining multiple hypotheses over data associations, which can become computationally expensive in dense target scenarios, such as SSA [26, 48, 71].

The PLMB filter provides a computationally efficient approximation of the PMBM filter by collapsing the multiple MB hypotheses for detected targets into a single LMB component, while maintaining a PPP for undetected targets. In this sense, a parallel can be drawn between the PMBM→PLMB relationship and the δ-GLMB→LMB, MHT→JPDA relationships. We will first define what is a Poisson RFS, also commenly denoted PPP [71].

A RFS X on $\mathbb{X}$ is said to be Poisson with a given intensity function v (defined on $\mathbb{X}$) if its cardinality $|X|$ is Poison distributed with mean $\bar{N} = \int v(\mathbf{x})d\mathbf{x}$, and for any finite cardinality, the elements $x \in X$ are independently and identically distributed according the probability density $v(\cdot)/\bar{N}$ [71]. The Poisson RFS is fully characterised by its intensity function (PHD) and has probability density

$$\pi(X) = e^{-\bar{N}} v^X$$

where

$$\forall h \,:\, X \to \mathbb{R},\, h^X \equiv \prod_{\mathbf{x}\in X} h(\mathbf{x})$$

with $h^{\emptyset} = 1$ by convention. Poisson RFSs are used in the PMBM/PLMB filters to process the birth models, for the spontaneous appearance of new targets in the surveillance areas as well as for undetected targets

Two different PLMB approaches can be followed. First, based on the work of Cament et al. [13] leading to a particle filter implementation, while the second implementation approach is inspired from the work of Fernandez et al. [26] on the PMBM filter implementation, allowing the reuse of a large portion of the already implemented structures for the PHD and LMB filters, as this work assumes both the intensity of the Poisson component and the predicted intensity to be Gaussian Mixtures.

The PLMB filter implementation in Cament et al. [13] is capable of processing optical observations from multiple disparate sensors, with similar performance to the LMB filter, while outperforming the PHD/CPHD filters. Moreover, it has the advantage of using a Poisson birth intensity, allowing for the implementation of a Partially Uniform Birth (PUB) model [36], and the solution also employs the Probabilistic Admissible Region (PAR), which is used to determine the initial orbits of the RSOs. The PUB and PAR are the two components of the multi-RSO initialisation, with the first appearing during the prediction step and the second providing the single RSO spatial density parameters in the update step. However, neither PUB nor PAR are an integral feature of any PLMB filter, but rather specific approaches proposed by Cament et al. [13]. The PUB intensity generates a uniformly distributed birth density in the sensor FoV defined by

$$\pi_{\mathcal{B}}(\mathbf{x}) = \lambda_{\beta} \cdot \mathcal{U}(\theta(\mathbf{x}); \mathcal{B}) \cdot \sum_{i=1}^{J_B} \left(w_b^{(i)} \mathcal{N}\left(\phi(\mathbf{x}); \mathbf{m}_{\phi}^{(i)}, \mathbf{P}_{\phi}^{(i)}\right)\right) \tag{4.58}$$

where $\mathcal{B}$ represents the sensor FoV, $\mathbf{x}$ is a target state in the ECI frame, $\theta(\mathbf{x})$ maps the target state to the observable part of the state, $\phi(\mathbf{x})$ is the non-observable part of the state, λ_{β} is the expected number of targets of the Poisson intensity, $\mathcal{U}(\theta(\mathbf{x}); \mathcal{B})$ is the uniform density for the observable part of the target state with boundary $\mathcal{B}$, the sum term represents a Gaussian Mixture Model of the non-observable part of the target, with J_B components $\mathcal{N}\left(\phi(\mathbf{x}); \mathbf{m}_{\phi}^{(i)}, \mathbf{P}_{\phi}^{(i)}\right)$ representing a multivariate Gaussian PDF over $\phi(\mathbf{x})$ with mean vector $\mathbf{m}_{\phi}^{(i)}$, covariance matrix $\mathbf{P}_{\phi}^{(i)}$ and weights $w_b^{(i)}$. The PUB represents prior knowledge about where new targets might appear, particularly in the non-observable state dimensions, as such it is important to carefully choose the Gaussian parameters. For instance, the means $\mathbf{m}_{\phi}^{(i)}$ can be centred on the most likely values for the unknown states in the region of interest,

such as typical velocities or orbital elements, while large covariances and large number of components will affect the computational efficiency negatively, but respectively allow to cover more uncertainties and for more flexibility depending on the complexity of the non-observable state distribution. Furthermore, the weights $w_b^{(i)}$ determine the relative importance of each Gaussian component in the non-observable state space, and if no prior preference exists they can be set uniformly, or non-uniformly to reflect possible prior knowledge about the target states.

Once measurements at time t_k become available, the PUB is refined using the PAR approach. The PAR conditions the predictive PUB density on the measurements to produce a measurement-informed Gaussian mixture over potential new target states. Specifically, the PAR multiplies the PUB with the measurement likelihood, i.e. the new RSO distribution $\pi'(\mathbf{x}, \mathbf{z})$ is given by multiplying the birth intensity Eq. (4.58) by the measurement likelihood distribution $l_z(\mathbf{z}|\mathbf{x})$, approximated using

$$\pi'(\mathbf{x}, \mathbf{z}) = \pi_{\mathcal{B}}(\mathbf{x}) l_z(\mathbf{z}|\mathbf{x}) \approx \lambda_\beta \cdot \mathcal{N}(\theta(\mathbf{x}); \mathbf{z}, \mathbf{R}) \cdot \sum_{i=1}^{J_B} \left(w_b^{(i)} \mathcal{N}\left(\phi(\mathbf{x}); \mathbf{m}_\phi^{(i)}, \mathbf{P}_\phi^{(i)} \right) \right) \tag{4.59}$$

where, $\mathcal{N}(\theta(\mathbf{x}); \mathbf{z}, \mathbf{R})$ is a Gaussian distribution on $\theta(\mathbf{x})$ modelling the resulting observable density for a given measurement $\mathbf{z}$, with sensor noise matrix $\mathbf{R}$. The resulting PAR-conditioned components are incorporated as new LMB tracks and updated together with surviving tracks in the standard multi-target update. This approach ensures that measurements inform the placement of new tracks, while the standard update adjusts existence probabilities and state estimates for all tracks, avoiding double-counting.

The admissible region approach can be further improved by imposing two distinct constraints, which are obeyed by the RSOs. One in terms of eccentricity, within a range of orbits of interest and the second in terms energy with the orbital semi-major axis length. Both constraints can be derived from the celestial two-body energy equation

$$\varepsilon = \frac{\|\dot{\mathbf{r}}\|^2}{2} - \frac{\mu_E}{\|\mathbf{r}\|} \tag{4.60}$$

with energy ε, gravitational constant of the Earth μ_E and position vector $\mathbf{r}$ in the ECI frame. On one hand, the eccentricity constraint $e \leq e_{\text{max}}$ is given by the angular momentum $\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}}$, where

$$\varepsilon = -\frac{\mu^2(1 - e^2)}{2\|\mathbf{h}\|^2} \tag{4.61}$$

On the other hand, the semi-major axis length constraint $a \leq a_{\text{max}}$ is provided by

$$\varepsilon = -\frac{\mu_E}{2a} \tag{4.62}$$

As an illustrative example Figure 4.11 how these constraints can restrict the set of possible target states in range and range-rate space. In this figure, the left panel depicts the basic admissible region based on range and range-rate measurements. The middle panel shows how additional constraints, such as a maximum orbital eccentricity and a fixed semi-major axis, further limit feasible target states. Finally, the right panel presents the resulting constrained admissible region (CAR), which is used to condition the PUB and generate the PAR GMM.

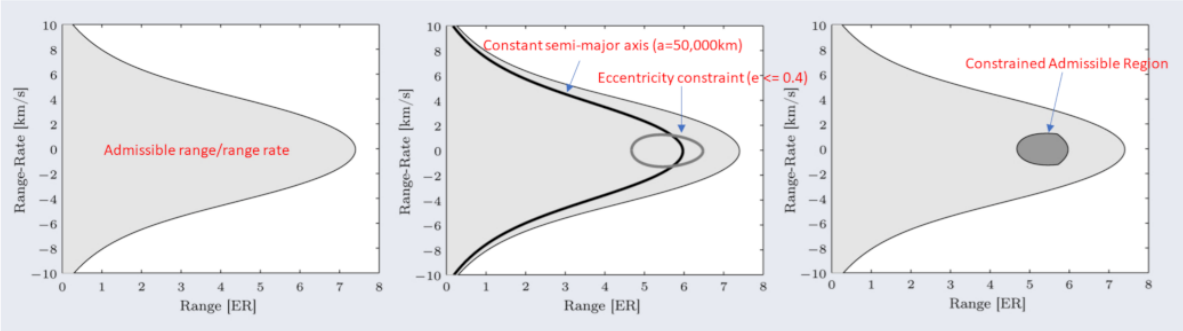


Figure 4.11: Illustrative example of the CAR/PAR approach for initialising target tacks, curtsey of Cament et al. [13].

For the PLMB filter, the multi-target tracking density is given by a LMB density characterised by its Probability Generating Functional (PGFL). Before defining the PGFL form of the LMB, it is useful to introduce what a PGFL represents in the context of Random Finite Sets. The PGFL is to random sets what the probability generating function or moment generating function is to random variables: it provides a compact functional representation of the entire multi-target distribution. Formally, for a multi-target RFS X with density $\pi(\cdot)$, the PGFL is defined as

$$G_X[h] = \sum_{n=0}^{\infty} \frac{1}{n!} \int \left(\prod_{i=1}^{n} h(\mathbf{x}_i) \right) \pi(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) d\mathbf{x}_1 \ldots d\mathbf{x}_n,$$

where $h : \mathbb{X} \to [0, 1]$ is a test function [48].

Intuitively, the PGFL encodes the distribution of both the number of targets and their states in a single functional. It is important to emphasize that the PGFL contains equivalent information to the probability density functions, but is more convenient to use when deriving RFS filters. Furthermore, it is possible to switch between PGFL and PDF formulation, which is more familiar for code implementation. Once in PGFL form, evaluating $G_X[h]$ for different choices of $h(\cdot)$ allows one to extract key probabilistic information: for example, setting $h(x) = 1$ recovers that the PGFL normalizes to one, while functional derivatives of $G_X[h]$ can be used to obtain cardinality distributions, intensity functions, and higher-order statistics. In other words, the PGFL completely characterises the RFS and serves as the main tool by which multi-target densities (such as Poisson, Bernoulli, or LMB) are concisely represented in closed form [48, 65].

For example:

- A Poisson RFS with intensity $v(\mathbf{x})$ has PGFL

$$G^P[h] = \exp\left(\int (h(\mathbf{x}) - 1) v(\mathbf{x}) \, d\mathbf{x} \right).$$

- A Bernoulli RFS with existence probability r and spatial density $f(\mathbf{x})$ has PGFL

$$G^B[h] = 1 - r + r\langle f, h \rangle.$$

These building blocks naturally extend to the LMB form used in the PLMB filter for which,

$$G_X^{\text{LMB}}[h] = \prod_{l \in \mathbb{L}} (1 - r_\ell + r_\ell \langle f_\ell(\mathbf{x}), h(\mathbf{x}) \rangle) \tag{4.63}$$

where for the RFS X, f_ℓ is the single target density of a target with label $l \in \mathbb{L}$, r_ℓ is the probability of existence and $h(\mathbf{x})$ is a function defined in the space of individual elements with $0 \leq h(\mathbf{x}) \leq 1$.

On the other hand the birth process is modelled by $\pi_{\mathcal{B}} = \lambda_{\mathbb{B}} f_B(\mathbf{x})$, where λ_B is the expected number of targets to be born with spatial distribution $f_B(\mathbf{x})$. Hence,

$$G_X^{\text{PLMB}}[h] = G_{\mathcal{B}}^P[h] G_Y^{LMB}[h], \tag{4.64}$$

represents the union of Poisson and LMB densities in PGFL form, where $G_Y^{LMB}[h]$ follows Eq. (4.63) with RFS Y and $G_{\mathcal{B}}^{P}[h]$ is a Poisson PGFL with

$$G_{\mathcal{B}}^{P}[h] = e^{\pi_{\mathcal{B}}[h-1]} \tag{4.65}$$

The filter consists of a multi-target prediction and multi-target update step. For a prior LMB defined by $\{(r_\ell, f_\ell)\}_{\ell \in \mathbb{L}}$ the prediction $\{(r'_\ell, f'_\ell)\}$ is given by

$$r'_\ell = r_\ell \langle P_S, f_\ell \rangle \quad \text{and} \quad f'_\ell(\mathbf{x}) = \frac{\langle P_S l_{\mathbf{x}}(\mathbf{x}|\cdot), f_\ell \rangle}{\langle P_S, f_\ell \rangle}, \tag{4.66}$$

with the probability of survival $P_S(\mathbf{x})$ and the kinematic model $l_X(\mathbf{x}_k|\mathbf{x}_{k-1})$. On the other hand, the update for a PMB-PGFL is a LMB Mixture PGFL, which can be approximated by a LMB-PGFL, consisting of new targets, detected targets and miss-detected targets,

$$G_{X|Z}^{+}[h] = \prod_{\ell \in \mathbb{L}} \left(1 - r_\ell^+ + r_\ell^+ \langle f_\ell^+, h \rangle\right) \tag{4.67}$$

where $\{(r_\ell^+, f_\ell^+)\}$ represent the probability of existence and the PDF of the updated target identified by label ℓ. In order to obtain r_ℓ^+ we solve the assignment problem on the cost matrix $\mathbf{W}_{n,m}$, which represents the weights of different combinations of targets $\{\ell_n : n \in \mathbb{N}\}$, for a total of N targets and M measurements $\{\mathbf{z}_m : m \in \mathbb{N}\}$. To solve this assignment problem, three different methods can be considered: the Loopy Belief Propagation algorithm [83], Murty's algorithm [56] or Gibbs sampling [64].

Alternatively, there is a more direct overall implementation path for the PLMB filter, based on the work performed so far with the LMB filter and inspired by Fernandez et al. [26] PMBM implementation. The overall implementation schematic is found in Figure 4.12, where the left hand-side is identical to the LMB implementation while the right hand side depicts the addition used to transform the filter into a PLMB one.

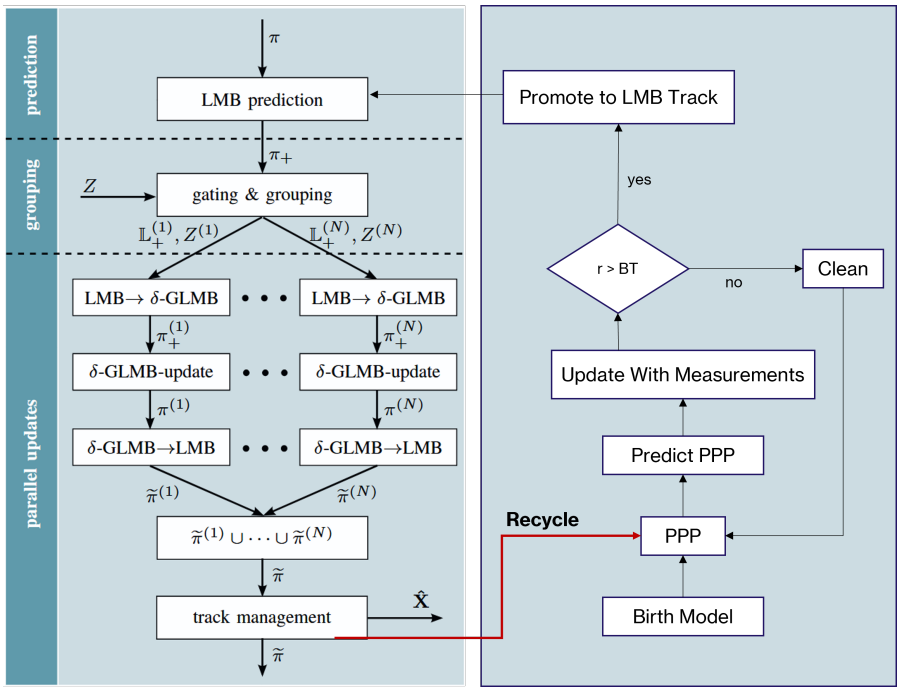


Figure 4.12: PLMB filter schematic overview

In the PLMB scheme the undetected targets i.e. ones not yet associated with any measurements are

represented by a PPP characterised by its intensity function[2] $\nu(\mathbf{x})$, which describes the expected density of points in state space. Although the PPP is not inherently Gaussian it is convenient to approximate it through a Gaussian Mixture, similarly to the process described for the PHD filter. As such after the prediction, gating, and update steps for the PPP are performed, a new element not yet discussed is introduced - the promotion of the PPP Gaussian Mixture to LMB track structure.

Concretely, when a measurement $\mathbf{z}$ is received, each Gaussian component of the PPP intensity is updated with respect to that measurement through either a linear Kalman Filter update or an Unscented Kalman filter update depending on the assumed motion model as described in subsection 4.5.2. This produces a set of candidate post-update means $\mathbf{m}^{(\mathbf{z})}$, covariances $P^{(\mathbf{z})}$, and likelihoods $q^{(\mathbf{z})}$. These are then normalised relative to the total PPP intensity and clutter intensity, yielding the existence probability for a potential new track [26]:

$$r^{(\mathbf{z})} = \frac{\sum_i p_D \cdot q_i^{(\mathbf{z})} w_i}{\kappa(\mathbf{z}) + \sum_i p_D \cdot q_i^{(\mathbf{z})} w_i}, \tag{4.68}$$

where p_D is the detection probability, subscript i indicates the number of Gaussian components w_i are the Gaussian weights of the PPP components, and $\kappa(\mathbf{z})$ is the clutter intensity at measurement z [26]. We then define a Birth Threshold (BT) such that if the BT is exceeded by $r^{(\mathbf{z})}$ the corresponding Gaussian Component is promoted to a LMB component $(\tilde{r}^{(\mathbf{z})}, p^{(\mathbf{z})}(\mathbf{x}))$. Two important remarks are to be made at this point:

Remark 1: How to choose the Birth Threshold? Initially, a fixed constant value BT = C was used, but this approach is suboptimal since it would require manual adjustment for every different scenario. Instead, an adaptive Birth Threshold is computed for each measurement to account for both the clutter intensity and the total PPP intensity. Concretely, without tuning, the adaptive threshold for a measurement $\mathbf{z}$ is computed as

$$BT_{\text{adaptive}}(\mathbf{z}) = \frac{\text{clutter weight}}{\text{total likelihood} + \text{clutter weight}}, \tag{4.69}$$

where the clutter weight is defined as the product of the clutter rate and the spatial density of clutter in the measurement space, and total likelihood is the sum of all PPP component likelihoods $q_i^{(\mathbf{z})}$. This ensures that measurements in regions of high clutter or low PPP intensity are less likely to trigger a new track, while measurements strongly supported by the PPP are more likely to be promoted.

To provide additional flexibility, a tuning parameter β_{birth} is introduced such that the final adaptive threshold becomes

$$BT^{\text{tuned}}_{\text{adaptive}}(\mathbf{z}) = \beta_{\text{birth}} \frac{\text{clutter weight}}{\text{total likelihood} + \text{clutter weight}}. \tag{4.70}$$

β_{birth} affects the promotion of Gaussian components as follows:

- $\beta_{\text{birth}} < 1$: lowers the BT potentially promoting more Gaussian components to new tracks. This increases the sensitivity to real targets but may also generate more spurious tracks from clutter.
- $\beta_{\text{birth}} = 1$: standard adaptive threshold based purely on clutter and PPP likelihood.
- $\beta_{\text{birth}} > 1$: raises the BT potentially promoting fewer components. This reduces spurious tracks but may miss real targets if the evidence is not strong enough.

Remark 2: Initially, in the implementation the Gaussian component was promoted to an LMB component with existence probability following Equation 4.68 i.e. $\tilde{r}^{(\mathbf{z})} = r^{(\mathbf{z})}$. However, through testing it was found that this led to large existence probabilities which would overestimate the cardinality, as will further be shown in chapter 5. This was happening despite the BT

[2]PPP and Poisson RFS can be used interchangeably in this context as PPP is the general point process definition, while a Poisson RFS is the FISST version described in [48]

improvement implemented from *Remark 1*. As such it was decided to take inspiration from some of the adaptive birth models proposed for LMB filters [30], and restrict the existence probability of the newly generated track using: $\tilde{r}^{(\mathbf{z})} \equiv \min(r^{(\mathbf{z})}, r_{max})$, where r_{max} is a constant chosen based on the studied scenario.

On the other hand, the Gaussian components which did not get promoted get their weights reduced by a factor of $(1 - p_D)$ to account for missed detections, ensuring consistency of the underlying PPP intensity representation. These components are then pruned, merged and capped as otherwise they may grow exponentially. It should be noted that the promotion scheme described in the two remarks is inspired by the implementations proposed in [26] and [13].

Moreover, the existence probability of the LMB surviving tracks is not impervious to changes. As such it is possible for a formal LMB track for which the existence probability dropped too low to be recycled back to a PPP Gaussian mixture, restarting the RHS process in Figure 4.12, using the following logic $w_i^{\text{PPP,new}} = \tilde{r}^{(\mathbf{z})} \cdot w_i^{\text{track}}$, while the means and covariances of the track just get copied over to the PPP as is.

4.5. Supporting Concepts and Methods

In this section we will provide some additional supporting theory about concepts which have been mentioned and used in the MTT filters, but have not been fully described.

4.5.1. Gaussian Mixture

A Gaussian Mixture (GM) is a probabilistic model that represents a distribution as a weighted sum of multiple Gaussian components. Formally, a Gaussian Mixture with J components in d-dimensional space is written as

$$p(\mathbf{x}) = \sum_{i=1}^{J} w_i \mathcal{N}(\mathbf{x}; \mathbf{m}_i, P_i), \tag{4.71}$$

where

- $w_i \geq 0$ are the component weights satisfying $\sum_{i=1}^{J} w_i = 1$ (or $\sum_{i=1}^{J} w_i = \lambda$ in the case of PPP intensity functions, where λ is the expected number of objects),
- $\mathcal{N}(\mathbf{x}; \mathbf{m}_i, P_i)$ denotes a Gaussian distribution with mean $\mathbf{m}_i \in \mathbb{R}^d$ and covariance $P_i \in \mathbb{R}^{d \times d}$,
- J is the number of Gaussian components.

Gaussian Mixtures are widely used in multi-target tracking because they can approximate arbitrary probability densities while retaining tractable analytical properties. In particular, both the PHD and LMB families of filters use GM representations to describe either the intensity function of a Poisson Point Process or the spatial probability density of a Bernoulli component.

Intuitively, one may view each Gaussian component as representing a hypothetical target located near $\mathbf{m}_i$ with uncertainty described by P_i, while the weight w_i expresses its relative contribution to the overall intensity.

Two small visual examples of what a GM represents have been provided in Figure 4.13 for a 1D case and Figure 4.14 for a 2D case, with numerical parameters summarised in Table 4.4. The 1D example in Figure 4.13 illustrates how multiple Gaussian bell curves combine into a multi-modal distribution, while the 2D example in Figure 4.14 shows how Gaussian components can be interpreted as elliptical density regions in the state space. The dashed ellipses in the 2D plot correspond to the 1σ and 2σ covariance contours of each component, highlighting the uncertainty structure encoded in their covariance matrices. Together, these figures demonstrate how Gaussian mixtures provide both flexibility and interpretability in representing complex multi-target state distributions.

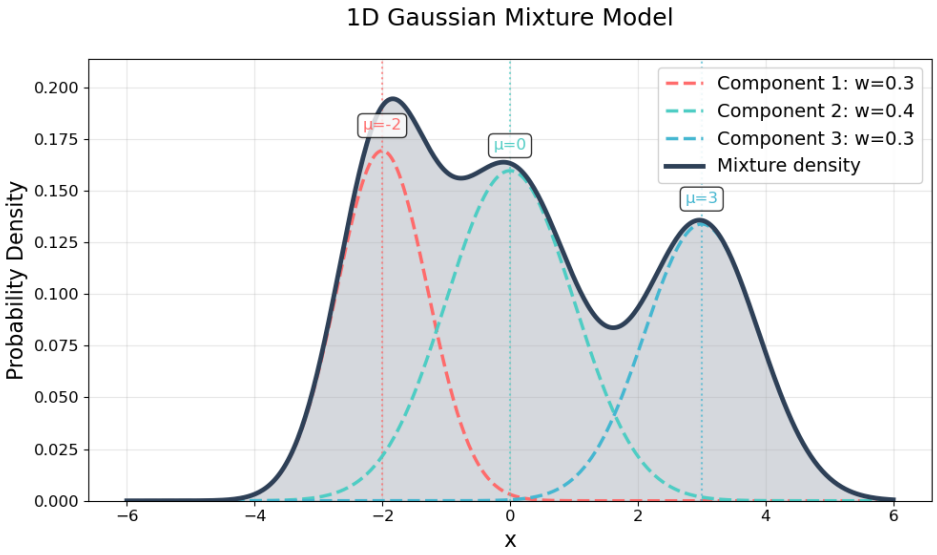


Figure 4.13: One-dimensional Gaussian mixture example showing three weighted Gaussian components (dashed lines) and their resulting mixture density (solid line).

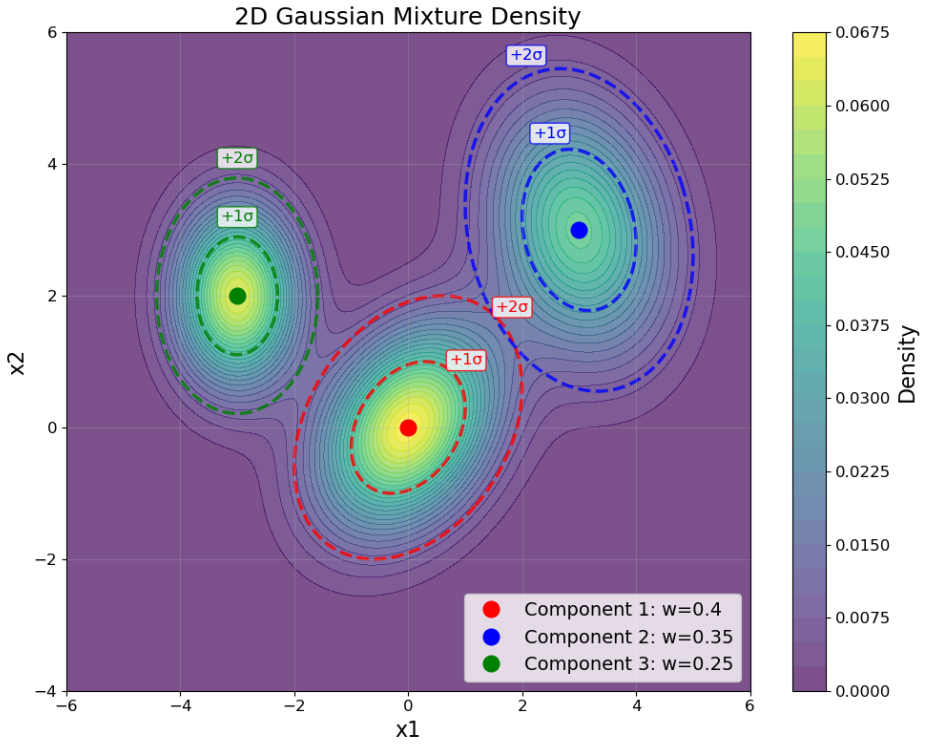


Figure 4.14: Two-dimensional Gaussian mixture example showing three components with elliptical covariance contours at 1σ and 2σ levels. Each component is represented by its mean (coloured dot) and covariance structure (dashed ellipses).

Table 4.4: Parameters of the Gaussian Mixture used for the illustrative examples.

Dimension	Component	Weight w_i	Mean $\mathbf{m}_i$	Covariance P_i
1D	1	0.30	-2.0	0.5
	2	0.40	0.0	1.0
	3	0.30	3.0	0.8
2D	1	0.40	$[0.0,\ 0.0]^\top$	$\begin{bmatrix} 1.0 & 0.3 \\ 0.3 & 1.0 \end{bmatrix}$
	2	0.35	$[3.0,\ 3.0]^\top$	$\begin{bmatrix} 1.0 & -0.2 \\ -0.2 & 1.5 \end{bmatrix}$
	3	0.25	$[-3.0,\ 2.0]^\top$	$\begin{bmatrix} 0.5 & 0.0 \\ 0.0 & 0.8 \end{bmatrix}$

4.5.2. Kalman and Unscented Kalman Filter: Prediction & Update

The filtering process consists of two steps: *prediction* and *update* (also called *correction*). In the following sub-subsections we define these steps mathematically and the reader is invited to take a look at their pseudo-code implementation in algorithm 9 and algorithm 10 respectively. The prediction step propagates the state and uncertainty forward in time using the system dynamics, while the update step incorporates new measurements to refine the state estimate. We distinguish between the linear Kalman Filter (KF) [29] and the Unscented Kalman Filter (UKF) [29], which generalizes the update to non-linear models.

Linear Kalman Filter

For a linear dynamical system, the model is defined as:

$$x_k = Fx_{k-1} + w_k, \qquad w_k \sim \mathcal{N}(0, Q),$$
$$z_k = Hx_k + v_k, \qquad v_k \sim \mathcal{N}(0, R),$$

where:

- $x_k \in \mathbb{R}^n$ is the hidden state at time step k,
- $z_k \in \mathbb{R}^m$ is the measurement,
- $F \in \mathbb{R}^{n\times n}$ is the state transition matrix,
- $H \in \mathbb{R}^{m\times n}$ is the observation matrix,
- $Q \in \mathbb{R}^{n\times n}$ is the process noise covariance,
- $R \in \mathbb{R}^{m\times m}$ is the measurement noise covariance.

Prediction step:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1},$$
$$P_{k|k-1} = FP_{k-1|k-1}F^\top + Q,$$

where $\hat{x}_{k|k-1}$ and $P_{k|k-1}$ are the predicted mean and covariance.

Update step:

$$y_k = z_k - H\hat{x}_{k|k-1} \quad \text{(innovation)},$$
$$S_k = HP_{k|k-1}H^\top + R \quad \text{(innovation covariance)},$$
$$K_k = P_{k|k-1}H^\top S_k^{-1} \quad \text{(Kalman gain)},$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k,$$
$$P_{k|k} = (I - K_k H)P_{k|k-1}.$$

The likelihood of the measurement z_k given the state is:

$$p(z_k|\hat{x}_{k|k-1}) \propto \exp\left(-\tfrac{1}{2} y_k^\top S_k^{-1} y_k\right).$$

Unscented Kalman Filter

The UKF [29] consists of an unscented transform (UT), used by a prediction and update step, all of which are defined in the subsequent section. The reader is invited to go through the pseudo-code implementation of these concepts found respectively in algorithm 11 for the UT, algorithm 12 for a single measurement UKF prediction step and algorithm 13 for the prediction of a set of measurements, and equivalently for the update step to read algorithm 14 and algorithm 15.

For non-linear systems, the dynamics and measurement models are:

$$x_k = f(x_{k-1}, w_k), \qquad w_k \sim \mathcal{N}(0, Q),$$

$$z_k = h(x_k, v_k), \qquad v_k \sim \mathcal{N}(0, R),$$

where $f(\cdot)$ is the (possibly non-linear) state transition function, and $h(\cdot)$ is the (possibly non-linear) measurement function.

Unscented Transform. To approximate the mean and covariance of a non-linear transformation, the UKF generates a set of *sigma points*. For a Gaussian distribution with mean $\mu \in \mathbb{R}^n$ and covariance $P \in \mathbb{R}^{n\times n}$, the sigma points are:

$$\chi_0 = \mu, \qquad \chi_i = \mu + \sqrt{(n+\lambda)P}_i, \quad \chi_{i+n} = \mu - \sqrt{(n+\lambda)P}_i, \quad i = 1, \dots, n,$$

where $\sqrt{(n+\lambda)P}_i$ denotes the i-th column of the matrix square root (usually the Cholesky factor).

Each sigma point has an associated weight for computing expectations:

$$w_0^{(m)} = \frac{\lambda}{n+\lambda}, \qquad w_0^{(c)} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta),$$

$$w_i^{(m)} = w_i^{(c)} = \frac{1}{2(n+\lambda)}, \quad i = 1, \dots, 2n.$$

Legend of parameters.

- n: state dimension.
- α: spread parameter controlling the spread of sigma points around the mean. Typical choice: $10^{-3} \leq \alpha \leq 1$.
- κ: secondary scaling parameter. Often set to 0 or $3 - n$.
- $\lambda = \alpha^2(n + \kappa) - n$: scaling factor used to determine the spread of sigma points.
- β: parameter used to incorporate prior knowledge of the distribution (for Gaussian distributions, $\beta = 2$ is optimal).
- $w_i^{(m)}$: weight for the mean computation.
- $w_i^{(c)}$: weight for the covariance computation.

Prediction step. Each sigma point is propagated through the non-linear transition:

$$\chi_i^{(x)} = f(\chi_i, 0),$$

and the predicted mean and covariance are:

$$\hat{x}_{k|k-1} = \sum_{i=0}^{2n} w_i^{(m)} \chi_i^{(x)},$$

$$P_{k|k-1} = \sum_{i=0}^{2n} w_i^{(c)} \left(\chi_i^{(x)} - \hat{x}_{k|k-1}\right)\left(\chi_i^{(x)} - \hat{x}_{k|k-1}\right)^\top.$$

Update step. Sigma points are propagated through the measurement function:

$$\chi_i^{(z)} = h(\chi_i^{(x)}, 0).$$

The predicted measurement and covariance are:

$$\hat{z}_k = \sum_{i=0}^{2n} w_i^{(m)} \chi_i^{(z)},$$

$$S_k = \sum_{i=0}^{2n} w_i^{(c)} (\chi_i^{(z)} - \hat{z}_k)(\chi_i^{(z)} - \hat{z}_k)^\top + R,$$

$$P_{xz} = \sum_{i=0}^{2n} w_i^{(c)} (\chi_i^{(x)} - \hat{x}_{k|k-1})(\chi_i^{(z)} - \hat{z}_k)^\top.$$

Then:

$$K_k = P_{xz} S_k^{-1},$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_k),$$
$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^\top.$$

As in the linear case, the likelihood can be evaluated as:

$$p(z_k|\hat{x}_{k|k-1}) \propto \exp\left(-\tfrac{1}{2}(z_k - \hat{z}_k)^\top S_k^{-1}(z_k - \hat{z}_k)\right).$$

In summary, the Kalman Filter provides exact inference for linear Gaussian models, while the Unscented Kalman Filter extends the same predict–update framework to nonlinear systems by approximating distributions via sigma points and the unscented transform. There are two additional remarks that must be made with regards to choosing the UKF for the non-linear models :

Remark 1: It is also possible to perform a linearised propagation for the non-linear models using the Extended Kalman Filter (EKF), which has lower computational complexity, but also lower accuracy compared to the UKF. Figure 4.15 showcase the higher accuracy of the UKF, while the computational complexity associated with the UKF originates from its requirement to propagate $2n + 1$ sigma points where n is the dimension of the estimated state. Another advantage of the UKF is that there is no need for the derivation and calculation of the Jacobian matrices needed for the EKF.

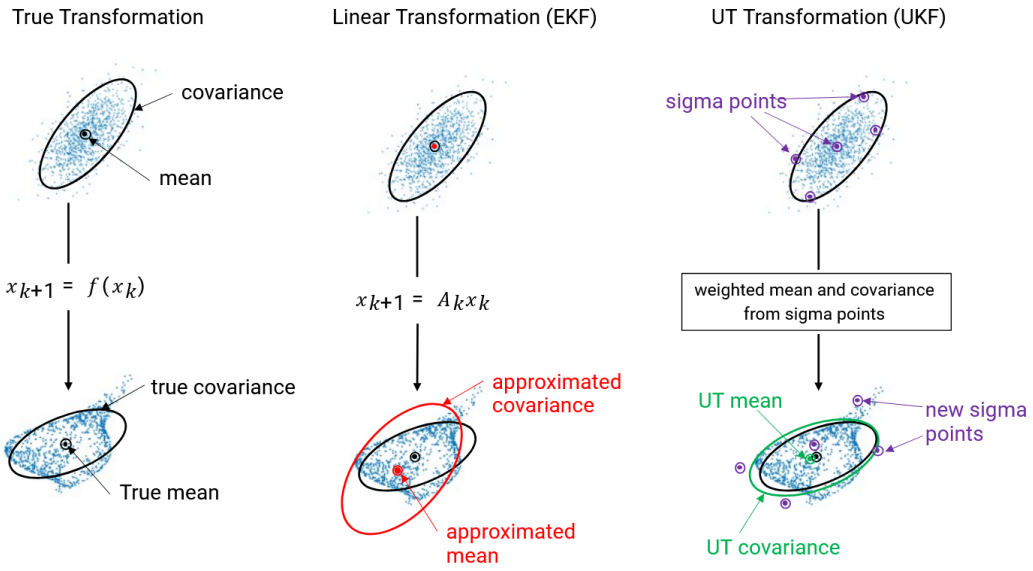


Figure 4.15: Visual representation of the accuracy difference between the EKF and the UKF, figure courtesy of MATHWORKS[3]

Remark 2: In case of coordinated frames or boundaries which are periodic and thus cropped it is important to be mindful of the location of the sigma points especially when the mean is close to the edge of the boundary, as can be seen visually in Figure 4.16.

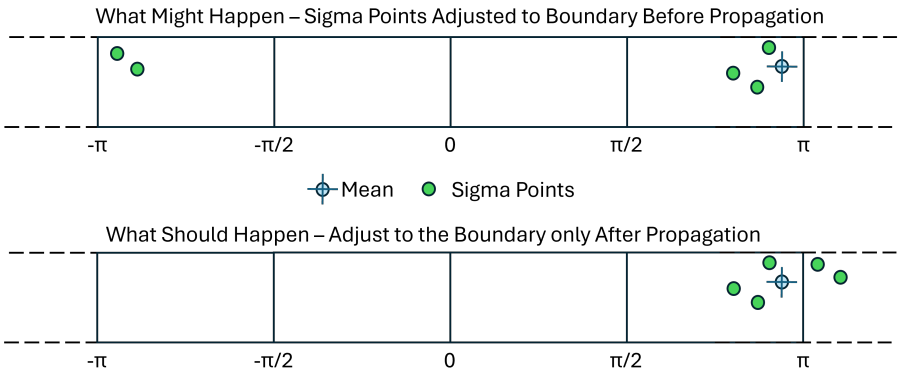


Figure 4.16: Boundary effects on sigma point representation

4.5.3. K-Shortest Paths and Murty's Algorithm for Hypothesis Management

A key difficulty in multi-target tracking is the exponential growth of possible association hypotheses: each measurement may originate from a true target, a false alarm, or clutter, and each target may or may not generate a detection. Exhaustively enumerating all possibilities is computationally infeasible. Instead, efficient algorithms are used to generate and rank only the most likely hypotheses. In this work, two complementary approaches are adopted: Yen's k-shortest paths algorithm (algorithm 3) for hypothesis generation in the prediction stage, and Murty's algorithm (algorithm 6, algorithm 8) for assignment ranking in the update stage.

Yen's K-Shortest Paths Algorithm.

Yen's algorithm[84] is a classical method for computing the k loopless shortest paths between two nodes in a weighted graph. In the LMB filter, the algorithm is applied when converting predicted tracks into GLMB hypotheses (`lmb2glmb`). The graph is built from track existence probabilities, with edge weights corresponding to negative log-likelihood ratios. Yen's algorithm is then used to extract the most probable subsets of tracks bounded by H_{req}, ensuring that only the most likely global hypotheses are retained.

DAG vs. General Graph Formulation.

The implementation of algorithm 3 supports two modes, depending on the structure of the hypothesis graph:

- **Directed Acyclic Graph (DAG) Case.** When the hypothesis space is acyclic, shortest paths can be computed efficiently using a topological sweep (algorithm 5) with linear complexity $O(V + E)$. In the filters considered here, the constructed graph is indeed a DAG. This is because hypotheses are formed by sequentially including or excluding tracks in index order, and once a decision is made, it is never revisited. Edges therefore always connect forward to higher-indexed nodes, which prevents the formation of cycles.

  The same property also holds in Space Situational Awareness (SSA) tracking problems, where tracks evolve forward in discrete time steps. Birth processes are indexed to the current scan, and assignment decisions do not feed back into past states. As a result, the hypothesis graph is topologically ordered by construction, ensuring that the DAG shortest path (DAGSP) solver (algorithm 5) is both valid and computationally advantageous.
- **General Graph Case.** If the graph may contain cycles or arbitrary edge directions, the Bellman–Ford–Moore (BFM) algorithm (algorithm 4)[6, 21, 53] is used instead. BFM computes single-

[3] `https://nl.mathworks.com/help/fusion/ug/introduction-to-estimation-filters.html` - Visited (10/09/25)

source shortest paths in $O(VE)$, handles negative edge weights, and detects negative cycles. Although slower than DAGSP, it ensures correctness in more general cases.

Thus, in the results it is expected that using DAG should increase the speed efficiency of the implementation at no cost to the accuracy.

Murty's Algorithm for M-Best Assignments.
While Yen's method generates candidate global hypotheses during prediction, the update stage requires solving the assignment problem between tracks and measurements. Murty's algorithm (algorithm 8, algorithm 7)[56] addresses this by extending the Hungarian algorithm (algorithm 6)[39, 40] to compute not just the optimal assignment, but the m best assignments in order of increasing cost.

Recall the simplified cost matrix example from chapter 2 for which the 5 best assignments are provided in Table 2.1,

$$C = \begin{array}{c} \begin{matrix} O_1 & O_2 & O_3 & O_4 \end{matrix} \\ \begin{bmatrix} \mathbf{7.1} & \times & \times & 4.3 \\ \times & \mathbf{5.2} & 8.4 & 5.1 \\ \times & \times & 6.6 & \mathbf{3.2} \end{bmatrix} \end{array} \begin{matrix} T_1 \\ T_2 \\ T_3 \end{matrix}$$

Using the Hungarian algorithm on the above cost matrix, we first proceed with a preparation step by padding the matrix with dummy values to make it square and setting $M \equiv \times \rightarrow +\infty$

$$C^{(0)} = \begin{array}{c} \begin{matrix} O_1 & O_2 & O_3 & O_4 \end{matrix} \\ \begin{bmatrix} 7.1 & M & M & 4.3 \\ M & 5.2 & 8.4 & 5.1 \\ M & M & 6.6 & 3.2 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} \end{array} \begin{matrix} T_1 \\ T_2 \\ T_3 \\ D_1 \end{matrix}$$

Then the cost allocation problem can be summarised in 4 steps:

1. Row Reduction $\rightarrow C^{(1)}$: Subtract the minimum value of each row from every entry of that row
2. Column Reduction $\rightarrow C^{(2)}$: Subtract the minimum value of each column from every entry of that column.
3. Cover zeroes with least number of lines $\rightarrow C^{(3)}$: find the minimum number of lines (row/column) required to cover every zero in the matrix.
   - $\#lines = \#allocations \equiv$: number of lines is the same as the number of allocations (i.e. the size of the matrix) we can draw the graph i.e. finding the independent zeros, building the assignment and mapping back to the original cost matrix (i.e. remove dummy assignments) $\rightarrow C^{(4)}$.
   - $\#lines < \#allocations$:
     (a) Find the smallest uncovered value $c_{ij}^{(2)}$ i.e. smallest value not part of a row/column covered by a line.
     (b) Subtract this value ($c_{ij}^{(2)}$) from every uncovered element.
     (c) Add this value to every element that is covered twice i.e. intersection of row and column covered lines.

     The above is equivalent to subtracting the value $c_{ij}^{(2)}$ from all uncovered rows and adding the value $c_{ij}^{(2)}$ to all columns that are covered. However doing the latter is more computationally efficient as the number of operations is generally lower.

4. Repeat Step 3.

As such, step 1 yields:

$$C^{(0)} = \begin{array}{c} \begin{matrix} O_1 & O_2 & O_3 & O_4 \end{matrix} \\ \begin{bmatrix} 7.1 & M & M & \mathbf{4.3} \\ M & 5.2 & 8.4 & \mathbf{5.1} \\ M & M & 6.6 & \mathbf{3.2} \\ \mathbf{0.0} & 0.0 & 0.0 & 0.0 \end{bmatrix} \end{array} \begin{array}{cc} & \min row \\ T_1 & 4.3 \\ T_2 & 5.1 \\ T_3 & 3.2 \\ D_1 & 0.0 \end{array} \rightarrow C^{(1)} = \begin{bmatrix} 2.8 & M & M & 0.0 \\ M & 0.1 & 3.3 & 0.0 \\ M & M & 3.4 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Step 2 results in:

$$C^{(1)} = \begin{array}{c} \begin{matrix} O_1 & O_2 & O_3 & O_4 \end{matrix} \\ \begin{bmatrix} 2.8 & M & M & \mathbf{0.0} \\ M & 0.1 & 3.3 & 0.0 \\ M & M & 3.4 & 0.0 \\ \mathbf{0.0} & \mathbf{0.0} & \mathbf{0.0} & 0.0 \end{bmatrix} \\ \min col \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \end{array} \begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \\ D_1 \\ \\ \end{array} \rightarrow C^{(2)} = \begin{bmatrix} 2.8 & M & M & 0.0 \\ M & 0.1 & 3.3 & 0.0 \\ M & M & 3.4 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

For step 3 we can observe that for a minimal covering column 4 (i.e. $col\ O_4$) and row 4 (i.e. $row\ D_1$) can be covered. As such the $\#lines = 2 < \#allocations = 4$ and we perform steps 3(a,b,c) with the smallest uncovered value $c_{ij}^{(2)} = c_{22}^{(3)} = 0.1$ as follow:

$$C^{(2)} = \begin{bmatrix} 2.8 & M & M & 0 \\ M & \mathbf{0.1} & 3.3 & 0 \\ M & M & 3.4 & 0 \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} \end{bmatrix} \rightarrow C^{(3)} = \begin{bmatrix} 2.7 & M & M & 0.0 \\ M & 0.0 & 3.2 & 0.0 \\ M & M & 3.3 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$

We now set the new $C^{(2)} = C^{(3)}$ and repeat step 3 until $\#lines = 4$:

$$C^{(2)} = \begin{bmatrix} \mathbf{2.7} & M & M & 0 \\ \cancel{M} & \cancel{0} & \cancel{3.2} & \cancel{0} \\ M & M & 3.3 & 0 \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0}1 \end{bmatrix} \rightarrow C^{(3)} = \begin{bmatrix} 0.0 & M & M & 0.0 \\ M & 0.0 & 3.2 & 2.7 \\ M & M & 0.6 & 0.0 \\ 0.0 & 0.0 & 0.0 & 2.8 \end{bmatrix} = C^{(2)}$$

$$C^{(2)} = \begin{bmatrix} \cancel{0} & \cancel{M} & \cancel{M} & \cancel{0} \\ \cancel{M} & \cancel{0} & \cancel{3.2} & \cancel{2.7} \\ M & M & 0.6 & 0 \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{2.8} \end{bmatrix} \rightarrow \#lines = 4 \rightarrow C^{(3)} = C^{(2)}$$

Thus we can now find the independent zeros i.e. exactly one zero in each column and each row and map back to the original matrix indices i.e ignore dummy to obtain the optimal assignment A_1:

$$C^{(3)} = \begin{bmatrix} \mathbf{0} & M & M & 0 \\ M & \mathbf{0} & 3.2 & 2.7 \\ M & M & 0.6 & \mathbf{0} \\ 0 & 0 & \mathbf{0} & 2.8 \end{bmatrix} \quad \longrightarrow A' = \{(T_1, O_1),\ (T_2, O_2),\ (T_3, O_4),\ (\cancel{D_1}, O_3)\},$$

$$\longrightarrow A_1 = \{(T_1, O_1),\ (T_2, O_2),\ (T_3, O_4)\}.$$

Using the optimal assignment as root, Murty's algorithm iteratively computes suboptimal solutions by branching from the root solution as can be seen in Figure 4.17. Each branch corresponds to forbidding the assignment at the given prefix position and forcing earlier prefix assignments. Furthermore, we can continue building the tree in the same manner by branching from each new suboptimal solution in turn. To generalise, given an optimal solution $S^{(1)} = (a_1, a_2, \ldots, a_n)$ where a_i is the column assigned to row i in some fixed row order. For each position $r = 1, 2, \ldots, n$ we create a sub-problem where we force $a_1, \ldots, a_{r-1}$ and forbid a_r (but do not forbidding other choices) and solve that sub-problem optimally. Each sub-problem produces a candidate solution; we insert all candidates into a min-heap by their costs and then extract the smallest to get the next-best solution. It is then possible to continue the process by branching from these solutions again and re-ranking the best assignments.

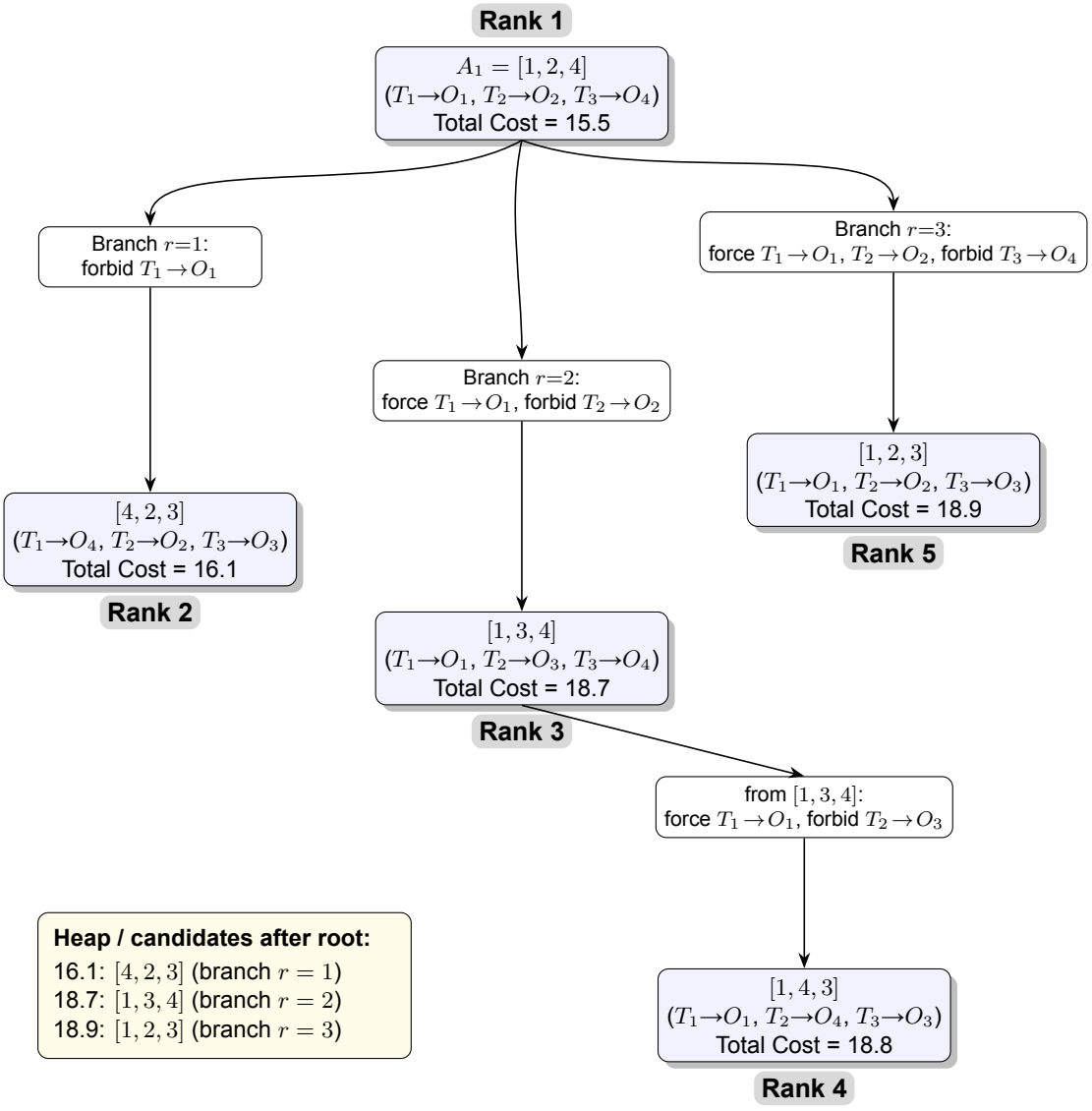


Figure 4.17: Murty branching tree for simplified cost matrix example.

In the LMB filter's update step (`updateGLMB`), a cost matrix is formed from the likelihood of assigning each measurement to each predicted track. Murty's algorithm is applied through a dedicated wrapper (algorithm 8), which pads the matrix with dummy assignments to allow for unassigned measurements, adjusts costs to ensure non-negativity, and converts the results into the required 1-indexed format. This produces multiple ranked assignment hypotheses. Each solution is converted into an updated GLMB

hypothesis with its own probability weight, allowing the filter to maintain multiple plausible association explanations.

Integration into the (P)LMB Filter.

Together, these two algorithms divide the computational burden of hypothesis management:

- **K-shortest paths** (algorithm 3) is used in the prediction step to enumerate the most likely subsets of existing and birth tracks, relying on DAGSP (algorithm 5) in our case.
- **Murty's algorithm** (algorithm 7, algorithm 8) is used in the update step to compute the best-ranked data association hypotheses for each predicted subset.

This separation ensures that the filter remains both computationally feasible and statistically robust: the K-shortest paths algorithm reduces the global hypothesis search space, while Murty's algorithm explores multiple feasible measurement assignments within each global hypothesis.

4.5.4. Model Description

When it comes to the physical modelling, we use on the one hand a continuous time model for the dynamics of the RSOs based on the differential equation

$$\ddot{\mathbf{r}} = -\frac{\mu}{\|\mathbf{r}\|}\mathbf{r} + \mathbf{a}_p \tag{4.72}$$

with position vector $\mathbf{r}$, gravitational parameter μ, and perturbing accelerations $\mathbf{a}_p$, which can include (but not limited to) solar radiation pressure, drag, third-body, etc, depending on the considered orbital regime. This model is used to demonstrate the performance of the filters. Moreover, the state vector $\mathbf{x}$ consists of the position and velocity in the Earth-Centred Inertial (ECI) Frame.

$$\mathbf{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{4.73}$$

Furthermore, to prevent the collapse of the filter covariance, causing measurements to be ignored, a standard process noise model is implemented, in which fixed values are multiplied by Δt of propagation before getting added along the diagonal terms of the covariance matrix. Last but not least the sensor model which will be used for the simplified one sensor model used follows the example provided by Gehly [28], i.e. the measurement is a topocentric right ascension and declination taken from a singular ground station located at $[x_s,\, y_s,\, z_s] = [-5465.210,\, -2403.610\, 2242.120]\,\mathrm{km}$ in ECEF. Furthermore, the measurement equations are given by

$$\alpha = \tan^{-1}\left(\frac{y - y_{si}}{x - x_{si}}\right) \qquad \delta = \sin^{-1}\left(\frac{z - z_{si}}{\rho}\right) \tag{4.74}$$

where $\rho = \sqrt{(x - x_{si})^2 + (y - y_{si})^2 + (z - z_{si})^2}$ is the range and $_{si}$ indicate the ground station coordinates in ECI.

Linear Motion Model (Constant Velocity)

The linear motion model assumes targets move with constant velocity in 2D Cartesian coordinates and the ground truth state for this model can be observed in Figure 4.2.

State Vector:

$$\mathbf{x}(k) = [x, y, \dot{x}, \dot{y}]^T \tag{4.75}$$

State Transition:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{B}\mathbf{w}(k) \tag{4.76}$$

with

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.77}$$

Process Noise:

$$\mathbf{Q} = \sigma_w^2 \begin{bmatrix} T_s^4/4 & 0 & T_s^3/2 & 0 \\ 0 & T_s^4/4 & 0 & T_s^3/2 \\ T_s^3/2 & 0 & T_s^2 & 0 \\ 0 & T_s^3/2 & 0 & T_s^2 \end{bmatrix} = \begin{bmatrix} 6.25 & 0 & 12.5 & 0 \\ 0 & 6.25 & 0 & 12.5 \\ 12.5 & 0 & 25.0 & 0 \\ 0 & 12.5 & 0 & 25.0 \end{bmatrix} \tag{4.78}$$

where $\sigma_w = 5.0$ and $T_s = 1\text{s}$.

Non-Linear Motion Model (Coordinated Turn)

The coordinated turn model includes a turn rate ω, and a visual representation of the ground truth may be observed in Figure 4.18.

State Vector:

$$\mathbf{x}(k) = [x, \dot{x}, y, \dot{y}, \omega]^T \tag{4.79}$$

State Transition:

$$\mathbf{x}(k+1) = f(\mathbf{x}(k)) + \mathbf{B}_2 \mathbf{v}(k) \tag{4.80}$$

with transition equations for $\omega \neq 0$:

$$\begin{aligned} x_{k+1} &= x_k + \frac{\sin(\omega T_s)}{\omega}\dot{x}_k - \frac{1-\cos(\omega T_s)}{\omega}\dot{y}_k \\ \dot{x}_{k+1} &= \cos(\omega T_s)\dot{x}_k - \sin(\omega T_s)\dot{y}_k \\ y_{k+1} &= y_k + \frac{1-\cos(\omega T_s)}{\omega}\dot{x}_k + \frac{\sin(\omega T_s)}{\omega}\dot{y}_k \\ \dot{y}_{k+1} &= \sin(\omega T_s)\dot{x}_k + \cos(\omega T_s)\dot{y}_k \\ \omega_{k+1} &= \omega_k \end{aligned} \tag{4.81}$$

For $\omega \approx 0$, the model reduces to constant velocity motion.

Process Noise:

$$\mathbf{B}_2 = \begin{bmatrix} T_s^2/2 \cdot \sigma_{\text{vel}} & 0 & 0 \\ T_s \cdot \sigma_{\text{vel}} & 0 & 0 \\ 0 & T_s^2/2 \cdot \sigma_{\text{vel}} & 0 \\ 0 & T_s \cdot \sigma_{\text{vel}} & 0 \\ 0 & 0 & T_s \cdot \sigma_\omega \end{bmatrix} \tag{4.82}$$

with $\sigma_{\text{vel}} = 5.0$ m/s, $\sigma_\omega = \pi/180$ rad/s.

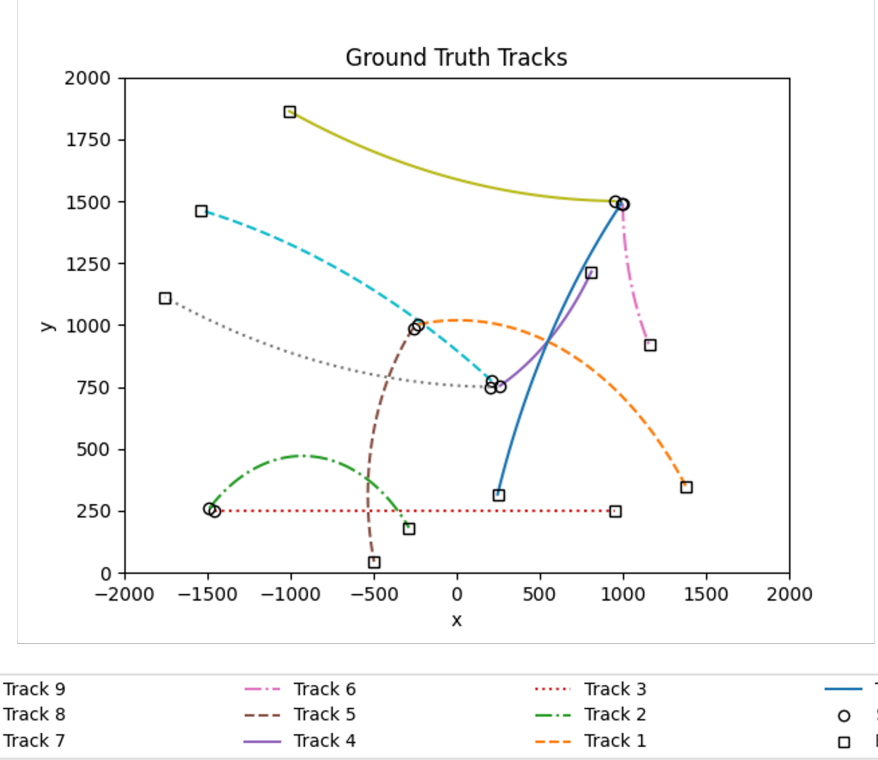


Figure 4.18: Ground Truth Tracks for the Coordinated-Turn (CT) model

Linear Measurement Model (Cartesian)

Measurement Equation:

$$\mathbf{z}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{n}(k), \tag{4.83}$$

where it should be noted that the linear model uses positions (x,y) for the measurements. Measurement Matrix:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \text{ (4D state)}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \text{ (5D state)} \tag{4.84}$$

Measurement Noise:

$$\mathbf{R} = \sigma_v^2 \mathbf{I}_2 = 100\mathbf{I}_2, \quad \sigma_v = 10.0 \tag{4.85}$$

Non-Linear Measurement Model (Radar)

Measurement Function:

$$\mathbf{z}(k) = h(\mathbf{x}(k)) + \mathbf{n}(k) \tag{4.86}$$

with

$$\text{bearing} = \arctan\frac{y}{x}, \quad \text{range} = \sqrt{x^2 + y^2} \tag{4.87}$$

Measurement Noise:

$$\mathbf{R} = \begin{bmatrix} \sigma_\alpha^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} = \begin{bmatrix} (2\pi/180)^2 & 0 \\ 0 & 10^2 \end{bmatrix} \approx \begin{bmatrix} 1.22 \times 10^{-3} & 0 \\ 0 & 100 \end{bmatrix} \tag{4.88}$$

with $\sigma_\alpha = 2°$, $\sigma_r = 10$ m.

4.5.5. Parameter Tables

This section regroups all the parameters needed to reproduce the simulations. These parameters have been chosen identical to the problem set up presented in the Vo. repository [4] which was used as a base of comparison for the verification of the filters' implementation.

Table 4.5: Motion Model Parameters (Linear / Non-Linear)

Parameter	Symbol	Value / Formula	Model
Sampling time	T_s	1.0 s	Linear / Non-Linear
Process noise std	σ_w	5.0 m/s²	Linear
Process noise std (velocity)	σ_{vel}	5.0 m/s	Non-Linear
Process noise std (turn rate)	σ_ω	$\pi/180$ rad/s	Non-Linear
Process noise covariance	$\mathbf{Q}$	$\begin{bmatrix} 6.25 & 0 & 12.5 & 0 \\ 0 & 6.25 & 0 & 12.5 \\ 12.5 & 0 & 25 & 0 \\ 0 & 12.5 & 0 & 25 \end{bmatrix}$	Linear
Process noise matrix	$\mathbf{B}_2$	$\begin{bmatrix} T_s^2/2 \cdot \sigma_{\text{vel}} & 0 & 0 \\ T_s \cdot \sigma_{\text{vel}} & 0 & 0 \\ 0 & T_s^2/2 \cdot \sigma_{\text{vel}} & 0 \\ 0 & T_s \cdot \sigma_{\text{vel}} & 0 \\ 0 & 0 & T_s \cdot \sigma_\omega \end{bmatrix}$	Non-Linear
Measurement noise std	σ_v	10.0 m	Linear
Measurement noise std (bearing)	σ_α	$2^\circ = 2\pi/180$ rad	Non-Linear
Measurement noise std (range)	σ_r	10 m	Non-Linear
Measurement noise covariance	$\mathbf{R}$	$\begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$	Linear
		$\begin{bmatrix} 1.22 \times 10^{-3} & 0 \\ 0 & 100 \end{bmatrix}$	Non-Linear
Clutter intensity	pdf_c	2.5×10^{-7}	Linear
		$1/(\pi \cdot 2000)$	Non-Linear

[4] Repository Link: `https://ba-tuong.vo-au.com/codes.html#`

Table 4.6: Target Birth Parameters

Parameter	Symbol	Value / Formula	Model
Birth probability	r_{birth}	$[0.03, 0.03, 0.03, 0.03]$	Linear
		$[0.02, 0.02, 0.03, 0.03]$	Non-Linear
Birth weight	w_{birth}	$[1, 1, 1, 1]$	Linear / Non-Linear
Birth state mean	m_{birth}	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 400 & -600 & 0 & 0 \\ -800 & -200 & 0 & 0 \\ -200 & 800 & 0 & 0 \end{bmatrix}$	Linear
		$\begin{bmatrix} -1500 & 0 & 250 & 0 & 0 \\ -250 & 0 & 1000 & 0 & 0 \\ 250 & 0 & 750 & 0 & 0 \\ 1000 & 0 & 1500 & 0 & 0 \end{bmatrix}$	Non-Linear
Birth state covariance	P_{birth}	$\text{diag}([100, 100, 100, 100])$	Linear
		$\text{diag}([50^2, 50^2, 50^2, 50^2, (6\pi/180)^2])$	Non-Linear

Table 4.7: Algorithmic Parameters (Merging, Pruning, Capping, Detection, etc.)

Parameter	Symbol	Value / Formula	Notes
Pruning threshold	T_p	10^{-5}	Linear / Non-Linear
Capping limit	N_{max}	100	Linear / Non-Linear
Merging threshold	U	4	Linear / Non-Linear
Detection probability	P_D	0.98	Linear / Non-Linear
Survival probability	P_S	0.99	Linear / Non-Linear
Maximum number of components	J_{max}	100	Linear / Non-Linear
Clutter intensity (already included in motion)	pdf_c	see motion table	Linear / Non-Linear

Table 4.8: Unscented Kalman Filter (UKF) Parameters for Non-Linear Models

Parameter	Symbol	Value	Notes
Primary scaling factor	α	1.0	Determines the spread of sigma points around the mean. Higher values increase spread.
Secondary scaling factor	κ	2.0	Secondary scaling factor that can improve higher-order moment matching; often set relative to state dimension.
Prior knowledge	β	2.0	Incorporates prior knowledge of the distribution; optimal value for Gaussian is 2.

5

Results & Discussion

In this chapter we will discuss the results obtained from our implementation of the filters. In total three single sensor multi-target filters were implemented: the PHD, LMB and PLMB filters, and have been tested on the linear scenario and non-linear coordinated turn scenario described in subsection 4.5.4. Furthermore, extrapolating from the behaviours observed in these models, some thoughts about their performance for SSA will be provided. The filters were compared between each other but also validated and verified against a repository[1] containing an LMB filter implementation in MATLAB provided by Ba Tuong Vo. First we will compare the linear/non-linear model results obtained from the LMB implementation against the PHD implementation, and follow by a comparison between the PLMB and LMB filter, as well as a time performance of the filters. Finally, extrapolating from what has been observed a discussion is provided on the expected performance when applied to an SSA problem.

5.1. PHD/LMB comparison

5.1.1. Linear Model

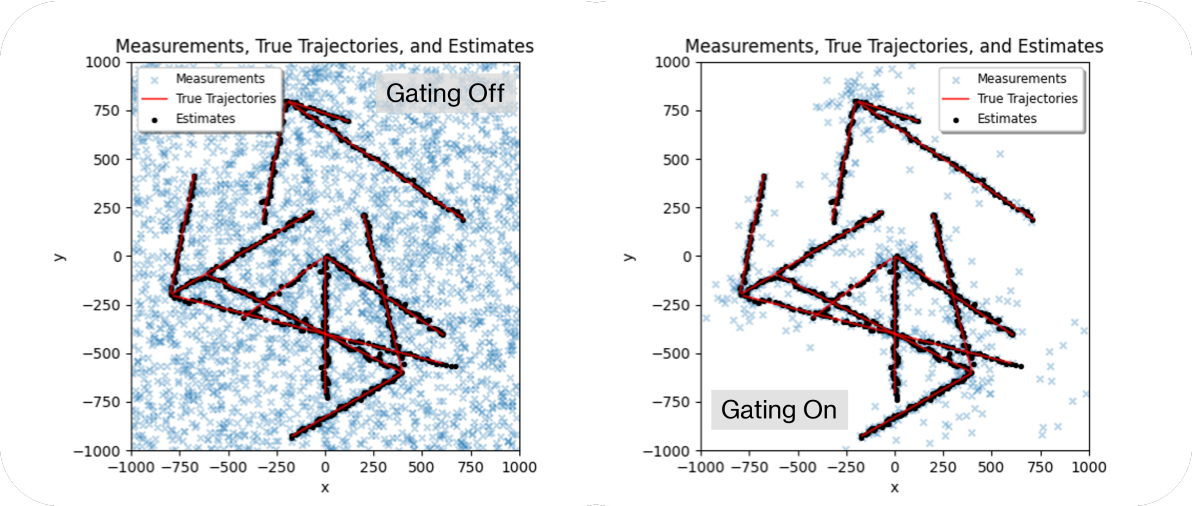


Figure 5.1: Ground truth, LMB estimates and measurements for the linear model without gating (left) and with gating (right)

In Figure 5.1 we can observe the effect of gating on the considered measurements for the data association problem. The largely reduced number of measurements considered for the association explains the large speed up present when the gating is on as seen in Table 5.1. Furthermore, as expected from theory in Figure 5.2 we can observe that the total OSPA error is more stable and reduced for the LMB

[1] Repository Link: `https://ba-tuong.vo-au.com/codes.html#`

estimates compared to the PHD estimates, and it is largely due to a much better cardinality estimation compared to the PHD filter as can be seen in Figure 5.4. Note that the spikes in the LMB OSPA cardinality error are centred around the time of target birth/death. Furthermore, in Figure 5.3 we can observe the labelling and tracking of the tracks that the LMB filter performs, whereas the PHD can not. It should be noted however that the PHD filter runs much faster i.e. $\mathcal{O}(10^{-1}) - O(10^{0})$ [s] compared to LMB $\mathcal{O}(10^{1})$ [s]. All the observations above are also true for the non-linear case for which it was decided to showcase instead the effect of gating on the errors.

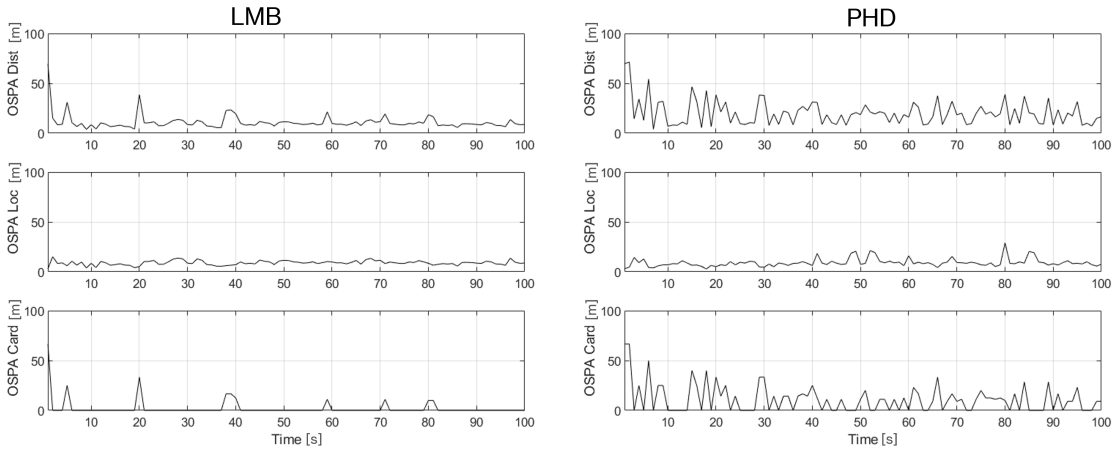


Figure 5.2: OSPA metric Comparison between LMB estimates (left) and PHD estimates (right)

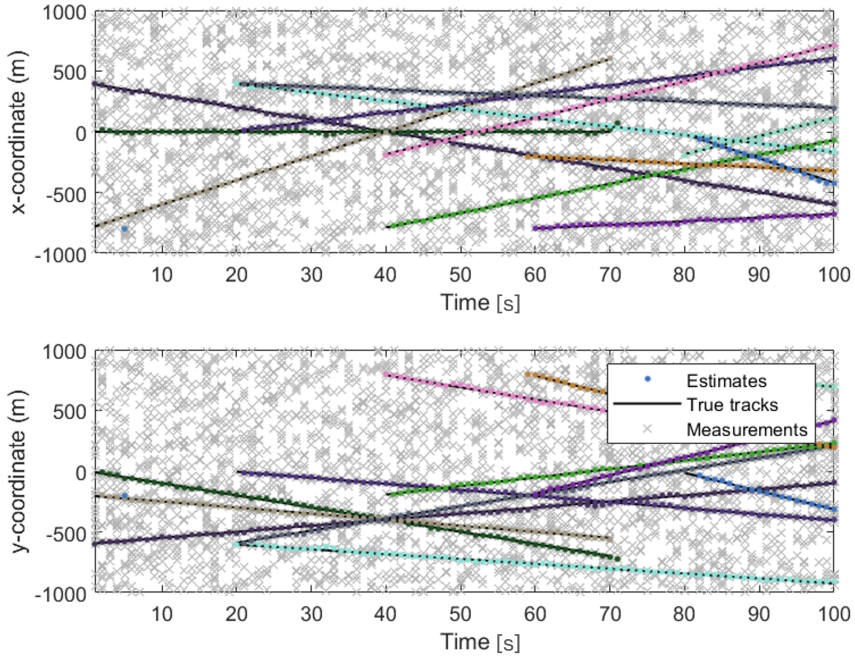


Figure 5.3: Visual representation of the LMB tracks over time in Cartesian coordinates for LIN-model

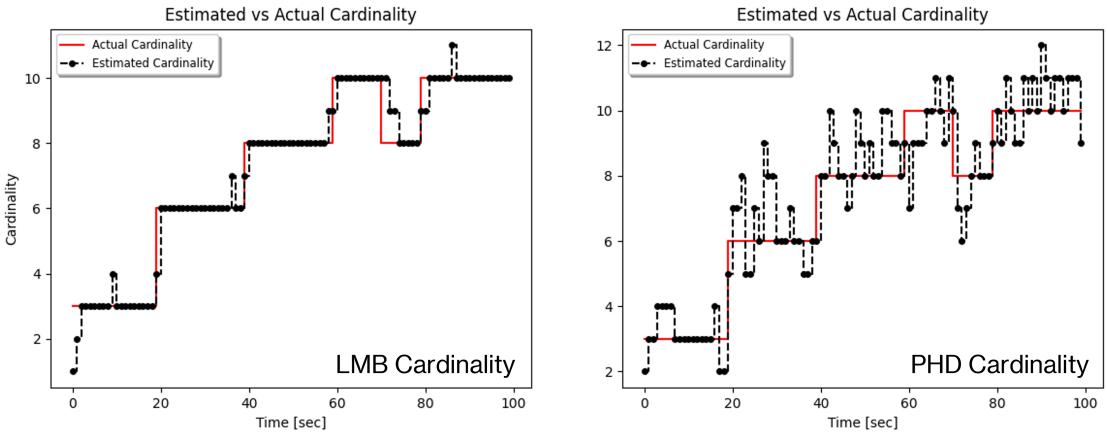


Figure 5.4: LMB (left) cardinalities compared to PHD (right) cardinalities estimates

5.1.2. Non-Linear Model

From theory we expect that gating the measurements will vastly increase the speed of the implementation but at the cost of some accuracy. This is well demonstrated in Figure 5.6 where it can be observed that the cardinality OSPA metric performs better when no gating is applied as the error peaks are reduced. However, on average this does not affect the total OSPA error as the values remain very similar between the gate on and off scenarios, and the estimated cardinality plots in Figure 5.5 are virtually indistinguishable. As such it can be said that the accuracy reduction caused by gating is insignificant when compared to the 25% speed up observed in Table 5.1 for the LMB filters implemented with gating. The track evolutions over time for the CT-model can be observed in Figure 5.7, where it can be seen that all tracks get identified and tracked correctly. Moreover, unlike the linear model, for the non-linear one the total OSPA error peaks of the LMB filter are not visibly/primarily distributed around the location of new target births, but are much more affected by the propagation of the non-linear model through the UKF, as can be seen from the larger OSPA localisation error, which behaviour is much less stable than for the linear case as it is expected.

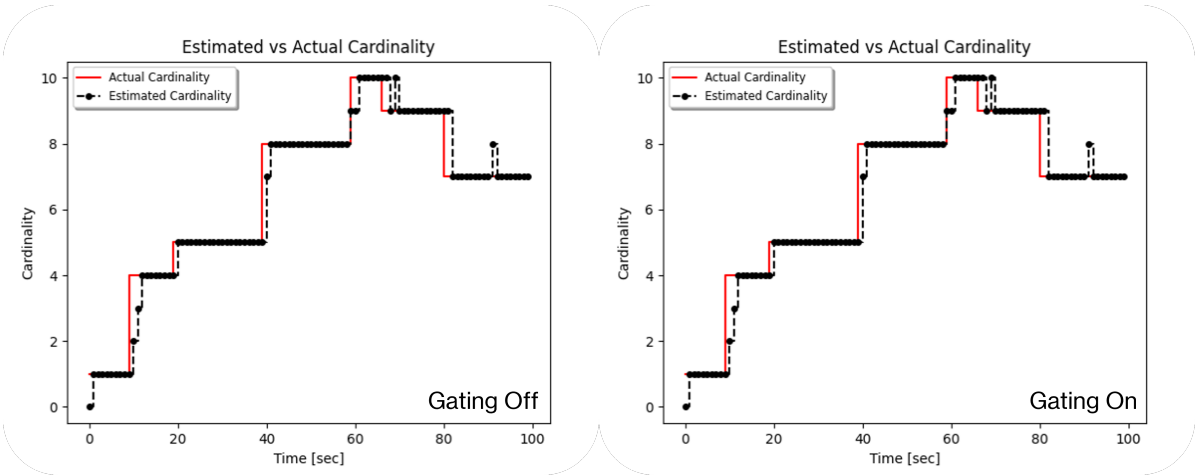


Figure 5.5: LMB filter estimated cardinalities vs. true cardinalities showcasing the effect of gating on the results on the non-linear model

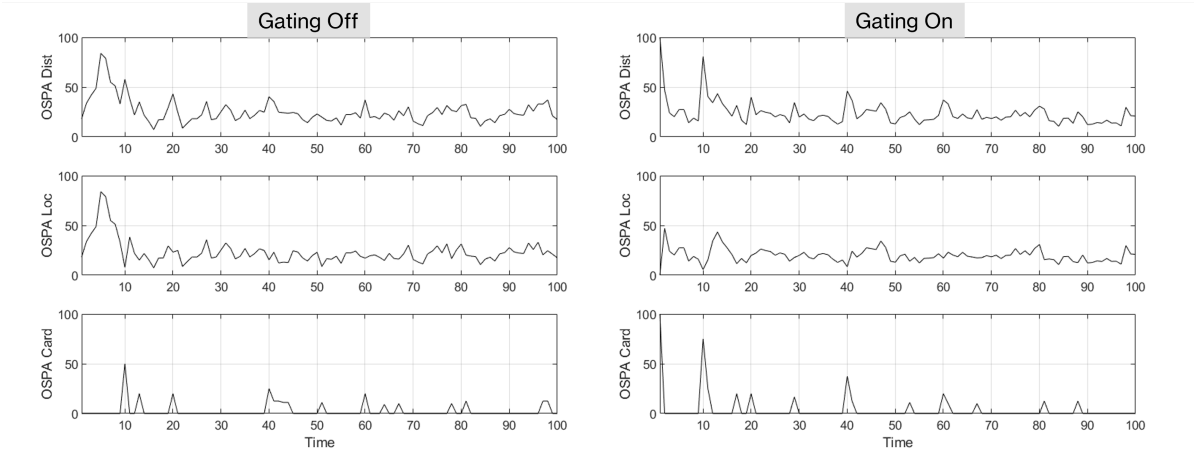


Figure 5.6: LMB filter estimated OSPA analysis showcasing the effect of gating on the results

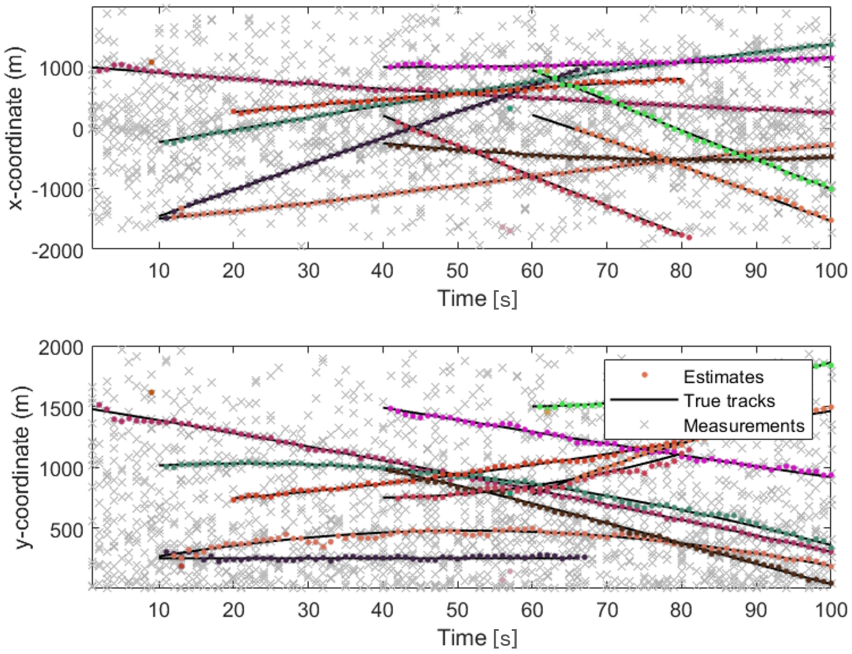


Figure 5.7: Visual representation of the LMB tracks over time in Cartesian coordinates for CT-model with gating on, for the CT-model

5.2. LMB/PLMB comparison

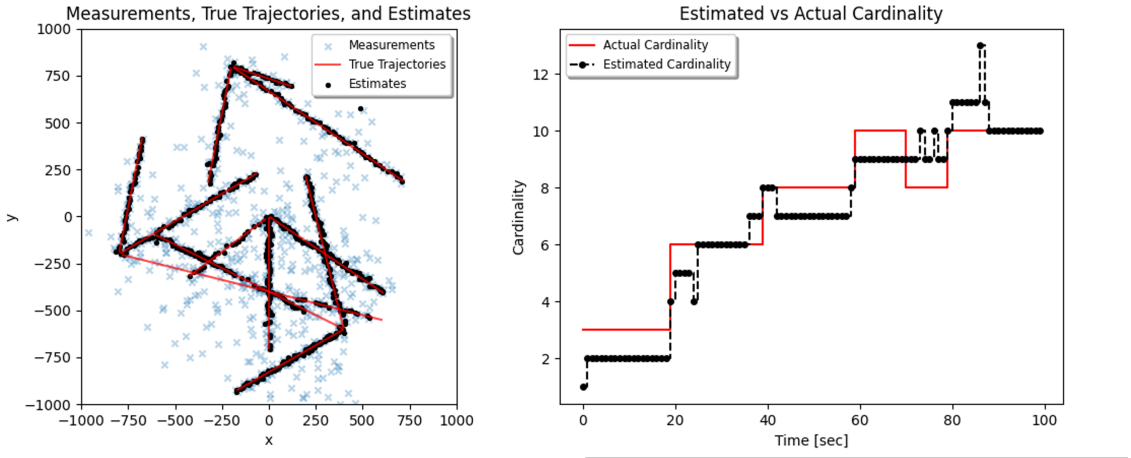


Figure 5.8: PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with constant birth threshold and active measurement gating

Initially, the PLMB filter was run with a constant birth threshold, unchecked probability of existence value of the undetected targets promoted to full LMB tracks, and active gating. The result are shown in Figure 5.8 and indicate poor performance in both track identification/localisation and cardinality estimate, with some tracks identified explaining the largely underestimated cardinality estimates. The test case is repeated with gating off turned off. As shown in Figure 5.9, this did not improve the performance, as once again a track is not recognised and the cardinality is underestimated.

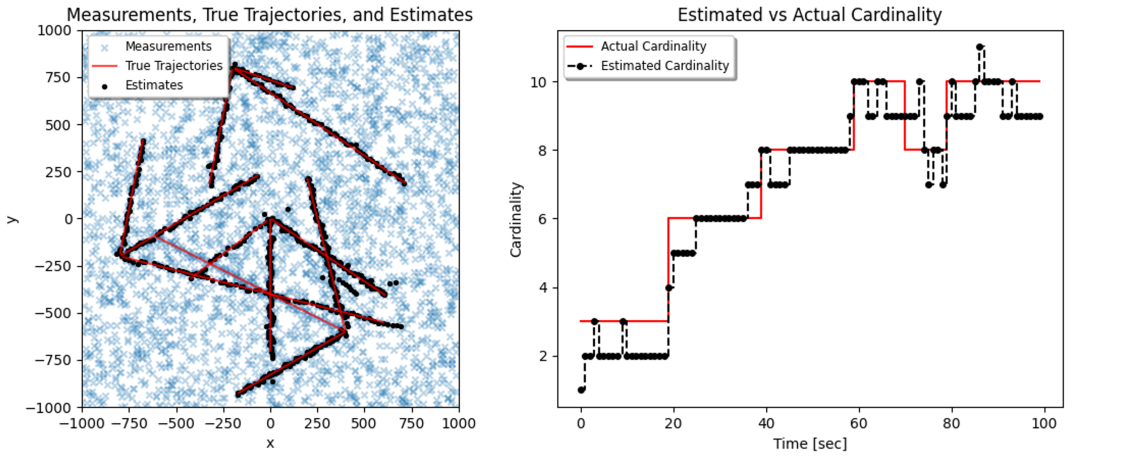


Figure 5.9: PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with constant birth threshold and no measurement gating

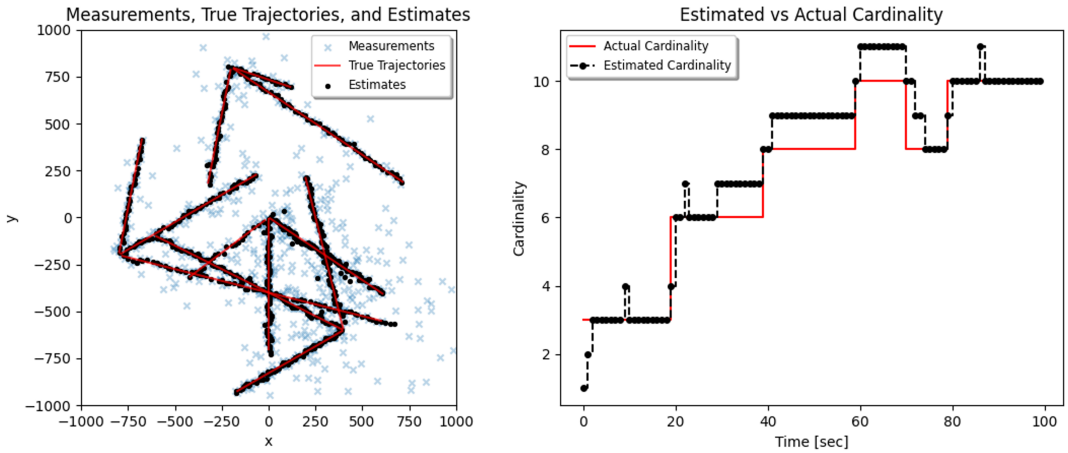


Figure 5.10: PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and no track existence probability bounding

This is where the idea of implementing an adaptive birth threshold based on the expected number of target, the clutter weights, and the total likelihood of a measurement discussed in section 4.4 originated, and the results of which implementation can be seen in Figure 5.10. Indeed, this time all tracks get recognised, but this time around the cardinality gets overestimated. The reason for which was found in the existence probability r of the promoted track which on occasions was very high e.g. around the r = 1.0 mark. This was also observed for the non-linear model where only implementing an adaptive birth threshold allowed for all tracks to be found but the cardinality overestimated as well as seen in Figure 5.12. The solution to which was to set a maximum existence probability for any track that gets promoted as discussed in section 4.4. The PLMB filters accuracy, although smaller than the LMB filter accuracy, was deemed sufficient to consider it a successful implementation as can be seen from Figure 5.11 and Figure 5.13. We especially want to highlight the result from Figure 5.13, which uses the non-linear model, and as such is most applicable to SSA. The PLMB track promotion methodology proposed in chapter 4 with the adaptive birth threshold, when tuned correctly, shows promising results as can be seen from Figure 5.13.

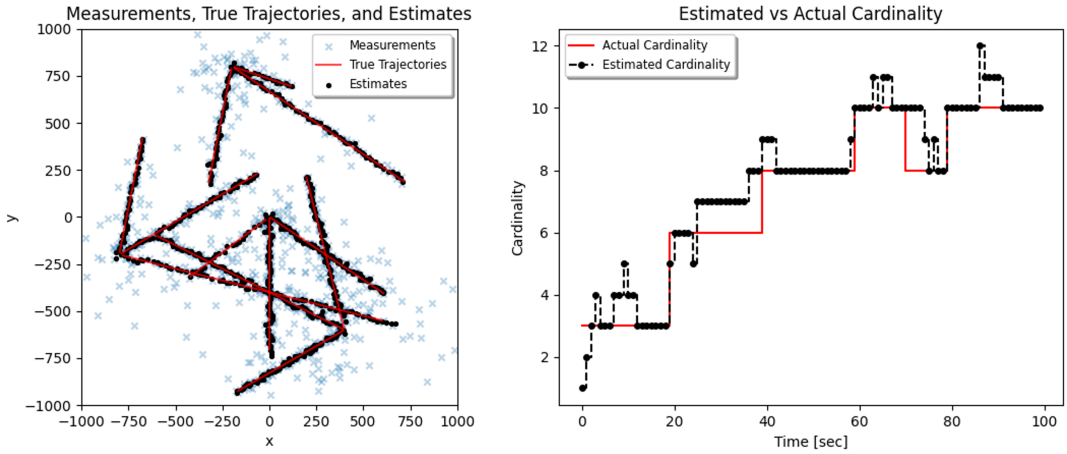


Figure 5.11: PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the LIN-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and track existence probability bounding

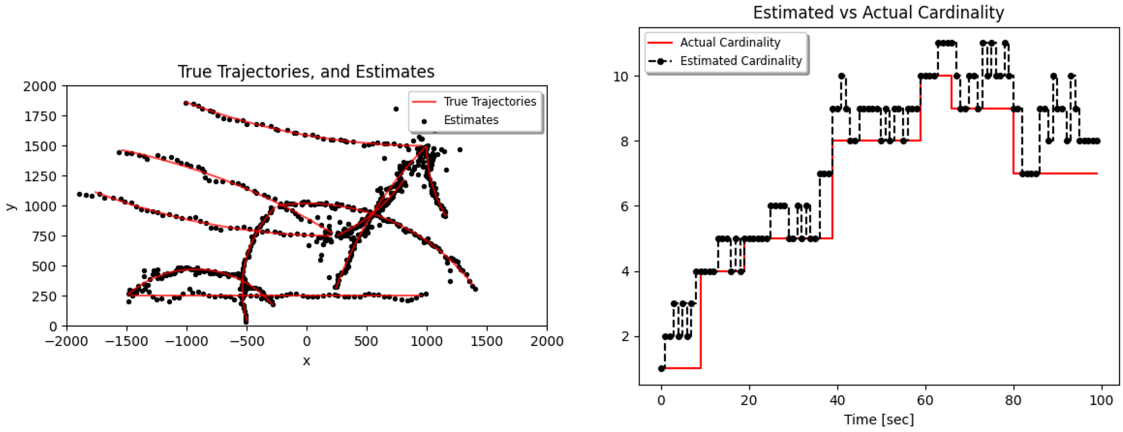


Figure 5.12: PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the CT-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and no track existence probability bounding for the CT-model

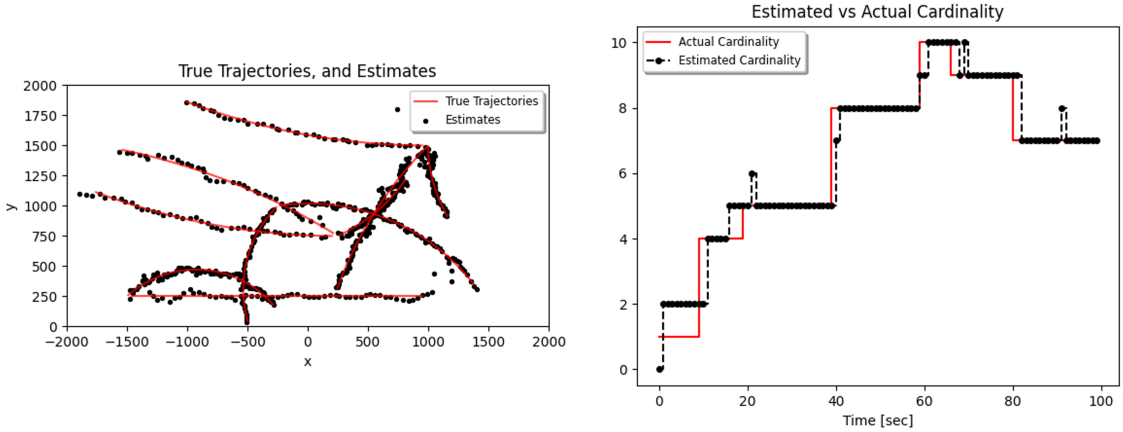


Figure 5.13: PLMB estimates, ground truth tracks, and gated measurements in Cartesian coordinates for the CT-model (left) and estimated vs. actual cardinality plot (right) with adaptive birth threshold and track existence probability bounding for the CT-model

5.3. Time Performance

The time performance of the filters is measured against a baseline example found in the MATLAB repository[2] of Ba Tuong Vo which considers the same non-linear CT-model as the one implemented in this research, with gating on, a hypotheses creation process which uses Yen's algorithm with a BFM implementation and also uses the Murty assignment algorithm for the data association problem. The CPU time is computed based on the average time it took over 10 runs of the filters and rounded up. The results are compiled in Table 5.1, from which we can see that the LMB implementation in the C++ repo performs as well as the baseline for the same parameters. As already discussed deactivating gating increases the CPU time drastically compared to similar runs with gating on. Moreover, the DAG implementation for the hypotheses creation as expected from theory leads to an improvement in CPU time which is significant, $\approx 30\%$ without any significant accuracy loss observable, as discussed in subsection 4.5.3. Finally, it can be observed that over the 10 runs the LMB slightly outperforms the PLMB in the time efficiency factor as well which is unexpected from theory, but can be due to implementation inefficiencies. One main implementation inefficiency is suspected as the main contributor to this result. The track promotion method promote a PPP to an LMB track, while performing a measurement update

[2]Repository Link: `https://ba-tuong.vo-au.com/codes.html#`

step. However, the current implementation actually promotes it to a tentative track for which the main acquired tracks update i.e. δ-GLMB update, also performs a measurement update on the tentative track effectively performing a relatively costly operation twice.

Table 5.1: Average CPU time performance of the LMB/PLMB filters for the non-linear coordinated turn model over 10 runs

Language	Filter	Gating	Hypotheses Creation	Time Performance [ms]	Improvement [%]
Matlab	**LMB**	**On**	**`BFM implementation`**	**52445**	**0**
C++	LMB	Off	BFM implementation	61075	-16.5
C++	LMB	Off	DAG implementation	49333	5.9
C++	LMB	On	BFM implementation	46734	10.9
C++	LMB	On	DAG implementation	36274	30.8
C++	PLMB	Off	BFM implementation	58076	-10.7
C++	PLMB	Off	DAG implementation	61075	-16.5
C++	PLMB	On	BFM implementation	47375	9.7
C++	PLMB	On	DAG implementation	37358	28.8

5.4. Implication for SSA

The results obtained in the linear and non-linear tracking scenarios provide insight into the expected behaviour of the studied filters when applied to SSA problems, such as tracking RSOs or resolving dense conjunction events in Earth orbit. While no SSA-specific measurement models were implemented in this work, several implications can be drawn.

First, the relative strengths of the filters observed in simulation remain relevant in the SSA context. The LMB filter's ability to provide accurate cardinality estimates and maintain track labels is especially important for SSA, where object custody across multiple sensor passes is a critical requirement. By contrast, the PHD filter, despite its computational efficiency, suffers from weaker cardinality estimation and the absence of persistent labelling, which would limit its utility in catalogue maintenance or collision avoidance tasks where track continuity is important.

Second, the impact of non-linear dynamics observed in the CT-model reflects challenges in SSA, where objects in eccentric or perturbed orbits often follow highly non-linear trajectories. The increased instability in OSPA location error for the non-linear case suggests that the choice of motion and measurement models will strongly influence filter performance in realistic orbital environments. A different non-linear filtering scheme, potentially tailored to orbital dynamics, could be considered to achieve higher accuracy, such using adaptive Gaussian Mixtures [85, 48] or a particle filter implementation [13].

Third, the PLMB filter demonstrates the value of adaptive mechanisms in balancing detection sensitivity and false track suppression. In SSA, where both missed detections (e.g. faint debris) and false alarms (e.g. sensor artefacts, clutter) are frequent, the ability to tune birth thresholds and bound existence probabilities can be crucial for scalable catalogue management. Although the PLMB accuracy was somewhat lower than the LMB, its flexibility indicates potential for further adaptation to SSA-specific conditions such as variable sensor coverage and cluttered measurement environments.

Finally, computational performance is a decisive factor in SSA applications. The significant speed-ups observed with gating and efficient hypothesis management imply that the studied algorithms, when carefully engineered, could be suitable for operational pipelines. However, further optimisation and parallelisation would likely be required for large-scale catalogue maintenance involving tens of thousands of RSOs. This parallelization can be achieved by properly implementing and testing the grouping mechanism described in subsection 4.3.1 which should speed up the update step of the filters as each sub-partition can be attributed to a separate thread, and the implemented C++ source code repo has been written with future parallelization in mind.

In summary, while the tested filters were not explicitly tuned to SSA measurement models or orbital dynamics, their relative behaviours in the linear and non-linear scenarios provide a meaningful indica-

tion of their prospective roles in SSA. The LMB appears most suited for high-fidelity catalogue tracking, the PHD for lightweight filtering when approximate situational estimates suffice, and the PLMB as a promising hybrid approach with potential for adaptive extensions tailored to the unique challenges of SSA.

6

Conclusion

This thesis has addressed the challenges of multi-target tracking in Space Situational Awareness, with a focus on the optimisation and scalability of filtering techniques within the Random Finite Set framework. By evaluating the performance and computational efficiency of both established and emerging multi-target filters, the research aimed to contribute toward the development of frameworks capable of supporting large-scale SSA applications, where accurate and efficient tracking of resident space objects is essential. The work combined theoretical analysis with practical implementation, emphasizing adaptability to distributed, multi-sensor architectures in C++.

The primary objective was to identify and assess which RFS-based methods can be optimised, scaled, and combined to address the computational demands of distributed multi-sensor, multi-target tracking in SSA. After the literature review phase, it was decided to implement three filters—the PHD, LMB, and PLMB—in C++ and evaluate their respective performances on linear, non-linear, and SSA models, for both single-sensor and multi-sensor scenarios. This research successfully implemented the three multi-target filters from scratch in C++ for both linear and non-linear models. The implementation demonstrated comparable, if not better, time efficiency than the MATLAB baseline, with very similar accuracy readings. Due to time constraints, extending these filters to a multi-sensor scenario or a full SSA model was not possible. Nonetheless, insight was gained regarding which methods are likely to perform best for SSA models, based on extrapolation from the achieved results. Consequently, the main research question was de-scoped to:

> Which methods within the Random Finite Set framework can be optimised, scaled up efficiently, and combined to handle the increased computational demands of the *single-sensor multi-target problem* for SSA?

The implementation of the filters for the linear and non-linear models demonstrated distinct trade-offs. The PHD filter achieved superior runtime performance, offering strong scalability potential; however, its lack of target labelling and unstable cardinality estimation make it less suitable for dense clutter environments or long-term custody of RSOs. In contrast, the LMB filter, while an order of magnitude slower, consistently provided better OSPA metrics and reliable target labelling, making it more appropriate for catalogue maintenance and scenarios requiring track continuity. The PLMB filter was expected to bridge these trade-offs by combining the advantages of the PHD and LMB filters. Results indicated that the PLMB achieved accuracy comparable to the LMB but did not yet outperform it in runtime. Nevertheless, the PLMB remains the most promising candidate for SSA applications, as its Poisson Point Process component retains information about undetected objects—a key capability in environments where new RSOs may unpredictably appear, and where catalogues must evolve dynamically. The proposed implementation of the PLMB promotion process was found to be successful for the non-linear model after the introduction of the tuned adaptive birth threshold, thereby establishing the PLMB as a promising approach for SSA problems.

In summary, the results from linear and non-linear scenarios provide a strong basis for assessing the filters' potential in SSA applications. While SSA-specific dynamics were not directly implemented, the

observed trade-offs indicate clear roles: the LMB for high-fidelity catalogue tracking, the PHD for computationally efficient approximate situational awareness, and the PLMB as a flexible hybrid with scope for adaptive enhancements. These findings highlight both the practical feasibility and the scalability considerations of RFS-based filters for large-scale SSA, guiding future extensions toward multi-sensor and SSA-specific implementations.

Future work should focus on incorporating an SSA-specific model for testing, extending the implementation to multi-sensor scenarios, and refining the promotion of Poisson Point Process components to LMB tracks through improved adaptive birth models and existence probability computations. Additionally, the grouping-based code structures developed in this work merit further investigation, as they have the potential to significantly reduce update-time complexity in large-scale SSA tracking systems. Moreover, implementation of advanced filtering schemes such as particle filter or adaptive Gaussian Mixture may improve the OSPA metric for non-linear scenarios. Finally, many of the filter parameters used, such as the merging threshold, were selected based on the baseline model used for code verification. It was observed that across the literature, many studies adopted similar parameter values with analogous reasoning. While intuitive effects of parameter changes—for example, increasing or decreasing the merging threshold and thus the number of Gaussian components—are understood, a deeper analysis of the impact of parameter choices is lacking. Therefore, performing a hyper-parameter analysis to identify which combinations of parameters most strongly influence performance would be an interesting and valuable extension of this work.

References

[1] Martin Adams et al. “SLAM Gets a PHD: New Concepts in Map Estimation”. In: *IEEE Robotics & Automation Magazine* 21 (2 June 2014), pp. 26–37. ISSN: 1070-9932. DOI: 10.1109/MRA.2014.2304111. URL: http://ieeexplore.ieee.org/document/6814323/.

[2] U. Balbin. *Course - AE4010 Research Methodologies*. Technical University Delft. Dec. 2024.

[3] Michael Beard, Ba Vo, and Ba-Ngu Vo. “OSPA (2) : Using the OSPA metric to evaluate multi-target tracking performance”. In: *2017 International Conference Control, Automation Information Science*. Oct. 2017, pp. 86–91. DOI: 10.1109/ICCAIS.2017.8217598.

[4] Michael Beard, Ba Tuong Vo, and Ba Ngu Vo. “A Solution for Large-Scale Multi-Object Tracking”. In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 2754–2769. ISSN: 19410476. DOI: 10.1109/TSP.2020.2986136.

[5] Michael Beard, Ba Tuong Vo, and Ba-Ngu Vo. “Performance Evaluation for Large-Scale Multi-Target Tracking Algorithms”. In: *2018 21st International Conference on Information Fusion (FUSION)*. 2018, pp. 1–5. DOI: 10.23919/ICIF.2018.8455700.

[6] Richard Bellman. “On a routing problem”. In: *Quarterly of Applied Mathematics* 16.1 (1958), pp. 87–90.

[7] José Bento and Jia Jie Zhu. *A metric for sets of trajectories that is practical and mathematically consistent*. 2020. arXiv: 1601.03094 [cs.CV]. URL: https://arxiv.org/abs/1601.03094.

[8] Keni Bernardin and Rainer Stiefelhagen. “Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics”. In: *EURASIP Journal on Image and Video Processing* 2008.1 (May 2008), p. 246309. ISSN: 1687-5281. DOI: 10.1155/2008/246309. URL: https://doi.org/10.1155/2008/246309.

[9] Margrit Betke et al. “Tracking Large Variable Numbers of Objects in Clutter”. In: *Computer Society Conference on Computer Vision and Pattern Recognition*. July 2007, pp. 1–8. ISBN: 1-4244-1180-7. DOI: 10.1109/CVPR.2007.382994.

[10] S Blackman and R Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, 1999.

[11] S.S. Blackman. “Multiple hypothesis tracking for multiple target tracking”. In: *IEEE Aerospace and Electronic Systems Magazine* 19.1 (2004), pp. 5–18. DOI: 10.1109/MAES.2004.1263228.

[12] Samuel S. Blackman. *Multiple-target tracking with radar applications*. Artech House, 1986.

[13] Leonardo Cament, Martin Adams, and Pablo Barrios. “Space debris tracking with the poisson labeled multi-bernoulli filter”. In: *Sensors* 21 (11 June 2021). ISSN: 14248220. DOI: 10.3390/s21113684.

[14] Leonardo Cament et al. “The histogram Poisson, labeled multi-Bernoulli multi-target tracking filter”. In: *Signal Processing* 176 (2020), p. 107714. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2020.107714. URL: https://www.sciencedirect.com/science/article/pii/S0165168420302577.

[15] Nicolas Chenouard, Isabelle Bloch, and Jean-Christophe Olivo-Marin. “Multiple Hypothesis Tracking for Cluttered Biological Image Sequences”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.11 (2013), pp. 2736–3750. DOI: 10.1109/TPAMI.2013.97.

[16] Coordinator-BME-3mE@TUDelft.nl. *Literature Research MSc BME YR2-Guidance document 2B*. Aug. 2023.

[17] I Cox and S Hingorani. “An efficient implementation of Reid’s multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (2 1996), pp. 138–150.

[18] Kai Da et al. “Recent advances in multisensor multitarget tracking using random finite set”. In: *Frontiers of Information Technology & Electronic Engineering* 22.1 (Jan. 2021), pp. 5–24. ISSN: 2095-9230. DOI: 10.1631/FITEE.2000266. URL: https://doi.org/10.1631/FITEE.2000266.

[19] E. Delande et al. “A new multi-target tracking algorithm for a large number of orbiting objects”. In: *Advances in Space Research* 64 (3 Aug. 2019), pp. 645–667. ISSN: 18791948. DOI: 10.1016/j.asr.2019.04.012.

[20] Claudio Fantacci et al. “The Marginalized δ-GLMB Filter”. In: *IEEE Signal Processing Letters* (2016). DOI: 10.1109/lsp.2016.2557078.

[21] Lester R Ford and Delbert R Fulkerson. “Network flow theory”. In: *Rand Corporation*. 1956.

[22] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. “Sonar tracking of multiple targets using joint probabilistic data association”. In: *IEEE Journal of Oceanic Engineering* 8.3 (1983), pp. 173–184. DOI: 10.1109/JOE.1983.1145560.

[23] D. Franken, M. Schmidt, and M. Ulmke. “”Spooky Action at a Distance” in the Cardinalized Probability Hypothesis Density Filter”. In: *IEEE Transactions on Aerospace and Electronic Systems* 45.4 (2009), pp. 1657–1664. DOI: 10.1109/TAES.2009.5310327.

[24] K. Gaast, M. Keestra, and L. Koender. *Chapters on interdisciplinary research and research skills*. Amsterdam University PR, 2020. ISBN: 9048553970. URL: https://www.jstor.org/stable/10.2307/j.ctv1fx4hbw.

[25] Lin Gao, Giorgio Battistelli, and Luigi Chisci. “Fusion of Labeled RFS Densities with Minimum Information Loss”. In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 5855–5868. ISSN: 1053587X. DOI: 10.1109/TSP.2020.3028496. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85094907620&doi=10.1109%2fTSP.2020.3028496&partnerID=40&md5=caf795f08ceab815681b3489bb11d391.

[26] Ángel F. García-Fernández et al. “Poisson Multi-Bernoulli Mixture Filter: Direct Derivation and Implementation”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.4 (2018), pp. 1883–1901. DOI: 10.1109/TAES.2018.2805153.

[27] Steve Gehly. “Distributed Fusion Sensor Networks for Space Situational Awareness”. In: *Proceedings of the 68th International Astronautical Congress (IAC)*. Paper code: IAC-17,A6,7,5,x40189. Adelaide, Australia, Sept. 2017.

[28] Steven Gehly. “Estimation of Geosynchronous Space Objects Using Finite Set Statistics Filtering Methods”. Available at https://www.colorado.edu/ccar/sites/default/files/attached-files/estimation_of_geosynchronous_s.pdf. PhD thesis. Boulder, CO: University of Colorado, 2016.

[29] Mohinder S. Grewal and Angus P. Andrews. *Kalman filtering: Theory and practice using MATLAB*. Wiley, 2015.

[30] Patrick Hoher et al. “A Detection Driven Adaptive Birth Density for the Labeled Multi-Bernoulli Filter”. In: *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. 2020, pp. 1–8. DOI: 10.23919/FUSION45008.2020.9190532.

[31] R Hoseinnezhad et al. “Visual tracking of numerous targets via multi-Bernoulli filtering of image data”. In: *Pattern Recognition* 45 (10 2012), pp. 3625–3635.

[32] K. Hussain et al. “Resident Space Objects Tracking Using Estimation-Based Data Fusion”. In: *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2024. ISBN: 9798350349610. DOI: 10.1109/DASC62030.2024.10748796.

[33] Khaja Faisal Hussain et al. “Space-Based Debris Trajectory Estimation Using Vision Sensors and Track-Based Data Fusion Techniques”. In: *Acta Astronautica* (Jan. 2025). ISSN: 00945765. DOI: 10.1016/j.actaastro.2025.01.038. URL: https://linkinghub.elsevier.com/retrieve/pii/S0094576525000396.

[34] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: https://www.mathworks.com.

[35] Brandon A Jones et al. “Challenges of multi-target tracking for space situational awareness”. In: *2015 18th International Conference on Information Fusion, Fusion 2015*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1278–1285. ISBN: 978-098244386-6. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84960540103&partnerID=40&md5=76de1b0bd3c3f729fbeb888705d1d11c.

[36] Brandon A. Jones. “CPHD Filter Birth Modeling Using the Probabilistic Admissible Region”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.3 (2018), pp. 1456–1469. DOI: 10.1109/TAES.2018.2793378.

[37] Donald Kessler et al. “The Kessler Syndrome: Implications to Future Space operations”. In: *Advances in the Astronautical Sciences* 137 (Jan. 2010).

[38] S Krishnaswamy and M Kumar. “Data association via tensor compression with application to GEO multi-target tracking”. In: *AIAA Scitech 2019 Forum*. 2019. DOI: 10.2514/6.2019-0376. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083943899&doi=10.2514%2f6.2019-0376&partnerID=40&md5=a06ae7d88a1e57db941bd8b17cb1814f.

[39] Harold W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97.

[40] Harold W. Kuhn. “Variants of the Hungarian method for assignment problems”. In: *Naval Research Logistics Quarterly* 3.4 (1956), pp. 253–258.

[41] B Li. “Novel Rao–Blackwellized jump Markov CBMeMBer filter for multi-target tracking”. In: *International Journal of Systems Science* 49 (15 2018), pp. 3007–3022. DOI: 10.1080/00207721.2018.1531320. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85054719842&doi=10.1080%2f00207721.2018.1531320&partnerID=40&md5=70c6435a501a627f001948c2256d877f.

[42] Suqi Li et al. “Multi-Sensor Multi-Object Tracking with Different Fields-of-View Using the LMB Filter”. In: *2018 21st International Conference on Information Fusion (FUSION)*. 2018, pp. 1201–1208. DOI: 10.23919/ICIF.2018.8455250.

[43] Tiancheng Li, Juan M. Corchado, and Shudong Sun. “Partial Consensus and Conservative Fusion of Gaussian Mixtures for Distributed PHD Fusion”. In: *IEEE Transactions on Aerospace and Electronic Systems* 55.5 (2019), pp. 2150–2163. DOI: 10.1109/TAES.2018.2882960.

[44] Tiancheng Li et al. *On Arithmetic Average Fusion and Its Application for Distributed Multi-Bernoulli Multitarget Tracking*. Jan. 2020. DOI: 10.36227/techrxiv.11599902.v1.

[45] H Liu et al. “Optimizing Distributed Multi-Sensor Multi-Target Tracking Algorithm Based On Labeled Multi-Bernoulli Filter”. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. ISBN: 2379-190X. DOI: 10.1109/ICASSP49357.2023.10095971.

[46] L Liu et al. “Multi-Target Tracking by Associating and Fusing the Multi-Bernoulli Parameter Sets”. In: *IEEE Access* 8 (2020), pp. 82709–82731. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2991365.

[47] Ronald Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, 2007.

[48] Ronald P. S.. Mahler. *Advances in statistical multisource-multitarget information fusion*. Artech House, 2014. ISBN: 9781608077984.

[49] Mahendra Mallick, Vikram Krishnamurthy, and Ba-Ngu Vo. “Bayesian Multiple Target Filtering Using Random Finite Sets”. In: *Integrated Tracking, Classification, and Sensor Management: Theory and Applications*. 2012, pp. 75–126. DOI: 10.1002/9781118450550.ch3.

[50] Gary Martin. *NewSpace: The Emerging Commercial Space Industry*. Accessed: 28-09-2025. 2015. URL: https://www.earthdata.nasa.gov/s3fs-public/2023-11/newspace_nasa.pdf.

[51] J. McKenzie et al. “The PRISMA 2020 statement: An updated guideline for reporting systematic reviews”. In: *International Journal of Surgery* 88 (Apr. 2021). ISSN: 17439159. DOI: 10.1016/j.ijsu.2021.105906.

[52] Sh. Meral. *K-Shortest Path- Yen's algorithm*. https://www.mathworks.com/matlabcentral/fileexchange/32513-k-shortest-path-yen-s-algorithm. MATLAB Central File Exchange. Retrieved May 29, 2025. 2025.

[53] Edward F. Moore. "The shortest path through a maze". In: *Proceedings of the International Symposium on the Theory of Switching* (1957), pp. 285–292.

[54] Michael Münz et al. "Generic Centralized Multi Sensor Data Fusion Based on Probabilistic Sensor and Environment Models for Driver Assistance Systems". In: *IEEE Intelligent Transportation Systems Magazine* (2010). DOI: 10.1109/mits.2010.937293.

[55] David D. Murakami et al. "Space Traffic Management with a NASA UAS Traffic Management (UTM) inspired architecture". In: *AIAA Scitech 2019 Forum* (Jan. 2019). DOI: 10.2514/6.2019-2004.

[56] Katta G. Murty. "An Algorithm for Ranking all the Assignments in Order of Increasing Cost". In: *Operations Research* 16.3 (1968), pp. 682–687. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/168595 (visited on 05/29/2025).

[57] Kusha Panta, Daniel Clark, and Ba-Ngu Vo. "Data Association and Track Management for the Gaussian Mixture Probability Hypothesis Density Filter". In: *Aerospace and Electronic Systems, IEEE Transactions on* 45 (Aug. 2009), pp. 1003–1016. DOI: 10.1109/TAES.2009.5259179.

[58] W J Park and C G Park. "Distributed GM-CPHD Filter Based on Generalized Inverse Covariance Intersection". In: *IEEE Access* 9 (2021), pp. 94078–94086. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3093719.

[59] Graham Pulford. "Taxonomy of multiple target tracking methods". In: *Radar, Sonar and Navigation, IEE Proceedings* - 152 (Nov. 2005), pp. 291–304. DOI: 10.1049/ip-rsn:20045064.

[60] S Quan, D Ding, and N Zhaodong. "Space Debris Tracking Via Generalized Labeled Multi-Bernoulli Random Finite Sets". In: *2019 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. 2019, pp. 1–4. DOI: 10.1109/ICSPCC46631.2019.8960764.

[61] Abu Sajana Rahmathullah, Ángel García-Fernández, and Lennart Svensson. "A Metric on the Space of Finite Sets of Trajectories for Evaluation of Multi-Target Tracking Algorithms". In: *IEEE Transactions on Signal Processing* PP (May 2016). DOI: 10.1109/TSP.2020.3005309.

[62] Abu Sajana Rahmathullah, Ángel F. García-Fernández, and Lennart Svensson. "Generalized optimal sub-pattern assignment metric". In: *2017 20th International Conference on Information Fusion (Fusion)*. 2017, pp. 1–8. DOI: 10.23919/ICIF.2017.8009645.

[63] D. Reid. "An algorithm for tracking multiple targets". In: *IEEE Transactions on Automatic Control* 24.6 (1979), pp. 843–854. DOI: 10.1109/TAC.1979.1102177.

[64] Stephan Reuter et al. "A fast implementation of the Labeled Multi-Bernoulli filter using gibbs sampling". In: *null* (2017). DOI: 10.1109/ivs.2017.7995809.

[65] Stephan Reuter et al. "The labeled multi-Bernoulli filter". In: *IEEE Transactions on Signal Processing* 62 (12 June 2014), pp. 3246–3260. ISSN: 1053587X. DOI: 10.1109/TSP.2014.2323064.

[66] Branko Ristic et al. "A Metric for Performance Evaluation of Multi-Target Tracking Algorithms". In: *IEEE Transactions on Signal Processing* 59 (July 2011), pp. 3452–3457. DOI: 10.1109/TSP.2011.2140111.

[67] Branko Ristic et al. "A Tutorial on Bernoulli Filters: Theory, Implementation and Applications". In: *IEEE Transactions on Signal Processing* 61.13 (2013), pp. 3406–3430. DOI: 10.1109/TSP.2013.2257765.

[68] Dominic Schuhmacher et al. "A Consistent Metric for Performance Evaluation of Multi-Object Filters". In: *IEEE Transactions on Signal Processing* (2008). DOI: 10.1109/tsp.2008.920469.

[69] Kai Shen et al. "Consensus-Based Labeled Multi-Bernoulli Filter for Multitarget Tracking in Distributed Sensor Network". In: *IEEE Transactions on Cybernetics* 52 (12 Dec. 2022), pp. 12722–12733. ISSN: 2168-2267. DOI: 10.1109/TCYB.2021.3087521. URL: https://ieeexplore.ieee.org/document/9478329/.

[70] J Smith et al. “Systematic Analysis of the PMBM, PHD, JPDA and GNN Multi-Target Tracking Filters”. In: *2019 22th International Conference on Information Fusion (FUSION)*. 2019, pp. 1–8. DOI: 10.23919/FUSION43075.2019.9011349.

[71] . IEEE Staff. *2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, 2011. ISBN: 9781457706745.

[72] Zhenzhen Su et al. “A Partitioned Poisson Multi-Bernoulli Filter”. In: *IEEE Sensors Journal* 23 (14 2023), pp. 16002–16012. ISSN: 1530437X. DOI: 10.1109/JSEN.2023.3283441. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85162707547&doi=10.1109%2fJSEN.2023.3283441&partnerID=40&md5=a3b6b623c7d900c7c35446945305f7bb.

[73] Jeffrey K. Uhlmann. “Algorithms for Multiple-Target Tracking”. In: *American Scientist* 80.2 (1992), pp. 128–141. ISSN: 00030996. URL: http://www.jstor.org/stable/29774599 (visited on 03/02/2025).

[74] B.-N. Vo and W.-K. Ma. “The Gaussian Mixture Probability Hypothesis Density Filter”. In: *IEEE Transactions on Signal Processing* 54 (11 Nov. 2006), pp. 4091–4104. ISSN: 1053-587X. DOI: 10.1109/TSP.2006.881190. URL: https://ieeexplore.ieee.org/document/1710358/.

[75] B.-N. Vo et al. “Multitarget Tracking”. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. Wiley, Sept. 2015, pp. 1–15. DOI: 10.1002/047134608X.W8275. URL: https://onlinelibrary.wiley.com/doi/10.1002/047134608X.W8275.

[76] Ba-Ngu Vo et al. “Sequential Monte Carlo methods for multitarget filtering with random finite sets”. In: *IEEE Transactions on Aerospace and Electronic Systems* (2005). DOI: 10.1109/taes.2005.1561884.

[77] Ba-Tuong Vo and Ba-Ngu Vo. “Labeled Random Finite Sets and Multi-Object Conjugate Priors”. In: *IEEE Transactions on Signal Processing* 61.13 (2013), pp. 3460–3475. DOI: 10.1109/TSP.2013.2259822.

[78] Tuyet Vu and Rob Evans. “A new performance metric for multiple target tracking based on optimal subpattern assignment”. In: *17th International Conference on Information Fusion (FUSION)*. July 2014, pp. 1–8.

[79] Gan Wang. “A pipeline algorithm for detection and tracking of pixel-sized target trajectories”. In: *Signal and Data Processing of Small Targets 1990*. Ed. by Oliver E. Drummond. Vol. 1305. International Society for Optics and Photonics. SPIE, 1990, p. 167. DOI: 10.1117/12.2321758. URL: https://doi.org/10.1117/12.2321758.

[80] K Wang, Q Zhang, and X Hu. “Label GM-PHD filter based on threshold separation clustering”. In: *Sensors* 22 (1 2022). DOI: 10.3390/s22010070. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85122300254&doi=10.3390%2fs22010070&partnerID=40&md5=713fd149caf46fbd1bed8f51aa7adca7.

[81] L Wang et al. “Simulation of CBMeMber Multi-target Tracking Algorithm Based on Gauss Mixture”. In: *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. 2019, pp. 1524–1528. ISBN: 2576-7828. DOI: 10.1109/ICCT46805.2019.8947076.

[82] B Wei and B D Nener. “Multi-Sensor Space Debris Tracking for Space Situational Awareness With Labeled Random Finite Sets”. In: *IEEE Access* 7 (2019), pp. 36991–37003. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2904545.

[83] Jason L. Williams. “Marginal multi-bernoulli filters: RFS derivation of MHT, JIPDA, and association-based member”. In: *IEEE Transactions on Aerospace and Electronic Systems* 51.3 (2015), pp. 1664–1687. DOI: 10.1109/TAES.2015.130550.

[84] Jin Y. Yen. “Finding the k Shortest Loopless Paths in a Network”. In: *Management Science* 17.11 (1971), pp. 712–716. DOI: 10.1287/mnsc.17.11.712.

[85] Huanqing Zhang, Hongwei Ge, and Jinlong Yang. “Adaptive Gaussian mixture probability hypothesis density for tracking multiple targets”. In: *Optik* 127.8 (2016), pp. 3918–3924. ISSN: 0030-4026. DOI: https://doi.org/10.1016/j.ijleo.2016.01.098. URL: https://www.sciencedirect.com/science/article/pii/S0030402616001479.

[86] Shangyu Zhao et al. “PMBM-based multi-target tracking under measurement merging”. In: *Signal Processing* 225 (2024). ISSN: 01651684. DOI: 10.1016/j.sigpro.2024.109610. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85198736104&doi=10.1016%2fj.sigpro.2024.109610&partnerID=40&md5=7d91fe0ad30ad10d74d070f70ad9d6cd.

[87] R Zhu et al. “An Improved Multi-Target Tracking Method for Space-Based Optoelectronic Systems”. In: *Remote Sensing* 16 (15 2024). DOI: 10.3390/rs16152847. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85200878897&doi=10.3390%2frs16152847&partnerID=40&md5=5ce85089e9bdfabee304117697317ab1.

A

Literature Selection Methodology

The selection of sources for this thesis was performed using the guidelines presented by Gaast [24] and Balbin [2], giving priority to (recent) peer-reviewed articles, books and conference proceedings. Moreover, the procedure employed to select and exclude sources systematically was inspired by the bio-medical field's use of the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) method[1] [16, 51]. The selection process is depicted in Figure A.1 and Figure A.2, where the distribution of the selected sources per year and the selection flowchart are shown, respectively.

The search was conducted using three different databases: Scopus, IEEE, and Research Rabbit. On one hand, for Scopus and IEEE the search query presented in Listing A.1 was used.

Listing A.1: Search Query Scopus/IEEE

```
TITLE-ABS-KEY[( "Multi-target tracking" OR "Multi target tracking" ) AND ( "Aerospace
    " OR "Space" OR "Space situational awareness" ) AND ( "Random finite set" OR "
    Bernoulli" OR "Joint Probabilistic Data Association" OR "Multi Hypotheses
    Tracking" )]
```

On the other hand, for Research Rabbit, the procedure consisted of feeding the supervisor recommended sources and papers obtained by using the search query presented in Listing A.2 with Scopus, and then using the 'Similar Work' functionality within Research Rabbit to extend the database.

Listing A.2: Search Query Scopus/IEEE

```
TITLE-ABS-KEY[( "Multi-target tracking" OR "Multi target tracking" ) AND ( "Aerospace
    " OR "Space" OR "Space situational awareness" ) AND ( "Random finite set" OR "
    Bernoulli" )]
```

This yielded a non-unique database of 412 sources, from which the duplicates were removed. Furthermore, a first filtering process was performed where the title and abstract of each source were read and the sources were given a score from $1^\star$ to $5^\star$ depending on relevance and publication year criteria. Sources scoring $\leq 2^\star$ are directly removed, while sources scoring $\geq 4^\star$ are included for the next phase. Sources that are hard to evaluate based on title and abstract are scored $3^\star$ and sent to the next phase with a special mention. Subsequently, in Phase 2, all papers are skimmed through, focusing on the research objective, results, and conclusion sections. The papers of interest are then included for Phase 3, which is the thorough source analysis. Finally, the sources are either discarded or included for the thesis and/or research proposal. The latter distinction is made in case a source is of relevance/interest for the thesis. Still, enough other sources are representative/similar and of good quality to obtain the overview needed in the research proposal.

[1] This method mainly applies to articles, but the same idea was adapted and employed for other source types.

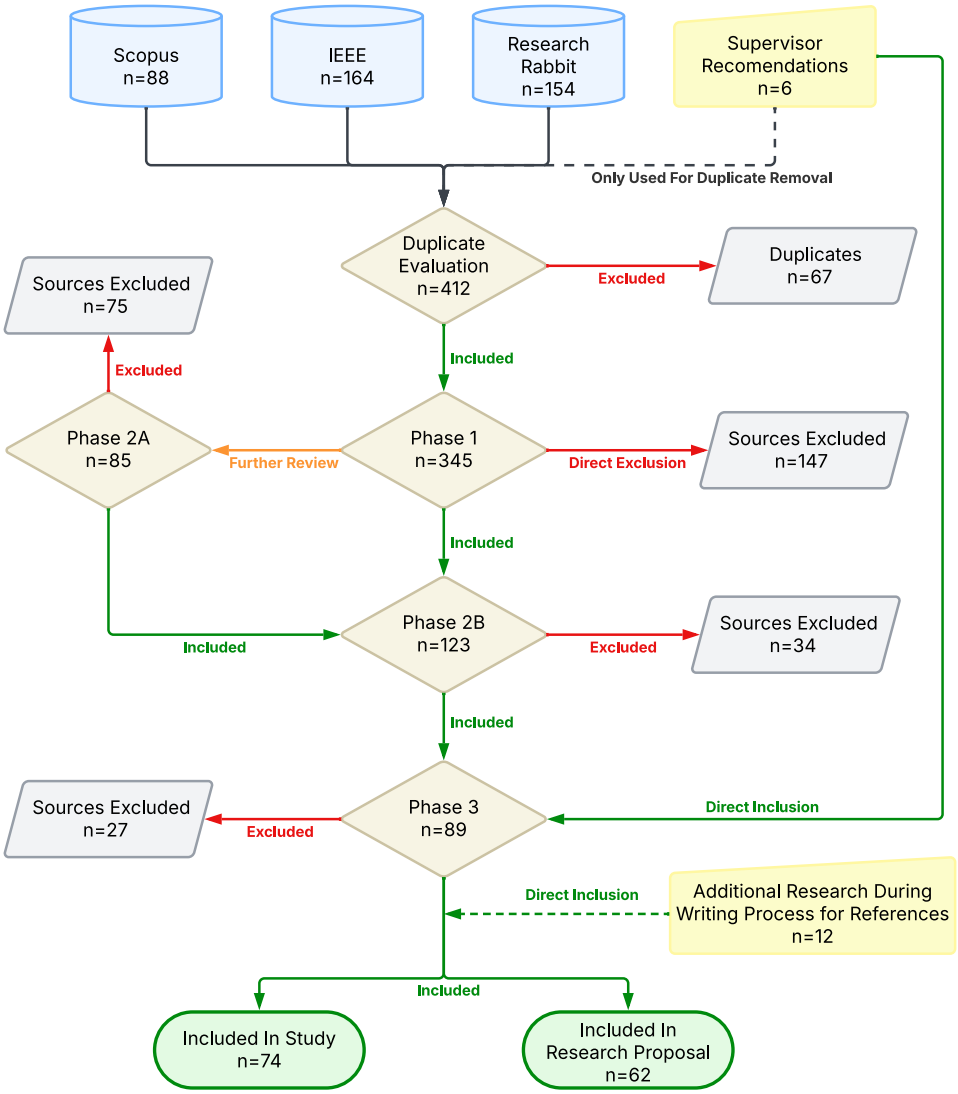


Figure A.1: PRISMA selection flowchart

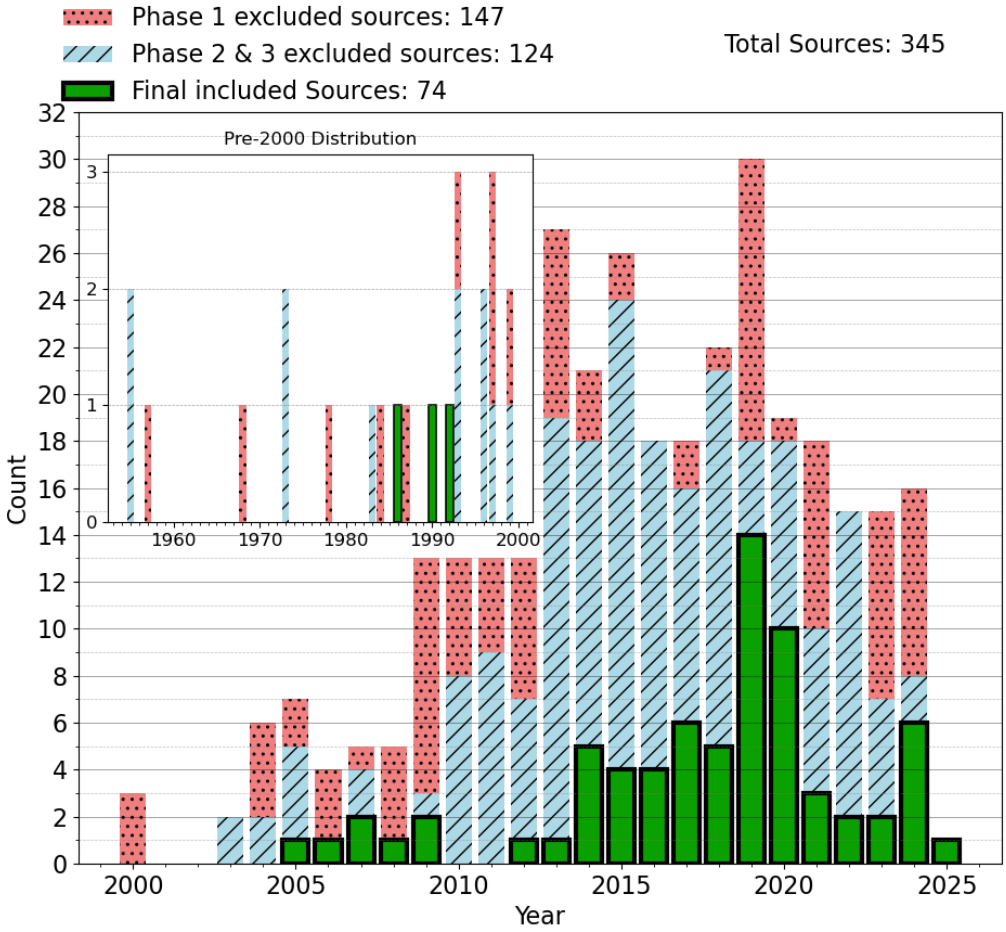


Figure A.2: Distribution of the selected sources for the thesis by publication year

B

Code Structures and Pseudo-Code

Algorithm 1: Labelled Multi-Bernoulli (LMB) Filter

Input: Measurement sequence $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$
Output: Estimated states $\hat{X}$
Data: Filter parameters: p_s, p_D, q_D, λ_c, p_c, thresholds (γ, track, elim., merge), caps ($T_{\max}, L_{\max}$)

Initialize empty estimates E and track set T_{LMB}
Compute gating threshold γ from $\mathcal{Z}$ (chi-square inverse)
foreach *time* $k = 1$ **to** K **do**
    `/* --- Prediction --- */`
    Create birth tracks (r, w, m, P) with labels (k, i)
    For each track $t \in T_{LMB}$: $r_t \leftarrow p_s \cdot r_t$; propagate (m, P) via Kalman/UKF
    `/* --- Conversion to GLMB --- */`
    Convert $T_{\text{birth}}, T_{\text{survive}}$ into GLMB
    Compute costs $c_i = r_i/(1 - r_i)$, apply K-shortest paths
    Normalize hypothesis weights (log-sum-exp)
    Compute cardinality distribution
    `/* --- Update --- */`
    Gate Z_k with γ (Linear/UKF)
    For each $z \in Z_k, t \in T$: update (m, P) and normalize w
    Build cost matrix, apply Murty's algorithm
    Update log-weights using λ_c, q_D, p_D, p_c
    Normalize weights, compute $\rho(n)$
    `/* --- GLMB → LMB --- */`
    Merge tracks with identical labels
    Redistribute weights by hypothesis probability
    Normalize mixture weights
    `/* --- Track cleaning --- */`
    Remove tracks with $r <$ thr, cap $T_{\max}$
    For each track: prune ($w <$ elim), merge close comps, keep $L_{\max}$
    `/* --- Estimate extraction --- */`
    Compute existence probs r_i
    Compute ESF, $\rho(n)$, mode N
    Pick top-N tracks by r_i, take max-weight Gaussian
    Store $\hat{X}_k$
    `/* --- Diagnostics --- */`
    Compute $\mathbb{E}[n]$, $\mathrm{Var}[n]$ from $\rho(n)$
    Print diagnostics
return $\hat{X}$

Algorithm 2: Poisson LMB (PLMB) Filter

Input: Measurement sequence $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$
Output: Estimated states $\hat{X}$
Data: Filter parameters: p_s, p_D, q_D, birth tuning, λ_c, p_c, thresholds (recycle, elim., merge, γ), caps $(T_{\max}, L_{\max}, J_{\max})$

Initialize empty estimates E, PPP intensity PPP, and track set T
Compute gating threshold γ from $\mathcal{Z}$ (chi-square inverse)
foreach *time* $k = 1$ **to** K **do**
`/* --- PPP prediction --- */`
Propagate each PPP Gaussian with Kalman/UKF
Multiply weights by p_s
Add birth Gaussian mixture
`/* --- Form new tracks from PPP --- */`
Gate Z_k with γ
For each $z \in Z_k$: compute likelihood $q(z)$ from PPP components
compute updated intensity $w' = \frac{p_D q(z) w}{\lambda_c + p_D Q q(z)}$
compute adaptive threshold $\theta =$ birth_tuning $\cdot \frac{\lambda_c}{\sum q(z) + \lambda_c}$
if $w' > \max(\theta,$ min Birth Threshold$)$: create new track with $r = \min($max $r_{\text{cont}}, w')$, label (k, i), Gaussian (m', P')
Reduce PPP weights by $(1 - p_D)$
`/* --- PPP cleaning --- */`
Remove PPP components with $w <$ elim
Merge close Gaussians (Mahalanobis distance $<$ merge)
Keep top $J_{\max}$ components
`/* --- Predict active LMB tracks --- */`
For each track $t \in T$: $r_t \leftarrow p_s \cdot r_t$; propagate (m, P) via Kalman/UKF
`/* --- Conversion to GLMB and update --- */`
Convert $T_{\text{new}}, T_{\text{survive}}$ into GLMB hypotheses
Gate Z_k, update (m, P) and normalize w
Build cost matrix, apply Murty's algorithm
Update log-weights using λ_c, q_D, p_D, p_c
Normalize with log-sum-exp, compute $\rho(n)$
`/* --- GLMB → LMB --- */`
Merge tracks with identical labels
Redistribute Gaussian components by hypothesis weights
Normalize mixtures
`/* --- Track cleaning --- */`
Remove tracks with $r <$ thr, cap $T_{\max}$
For each track: prune ($w <$ elim), merge close comps, keep $L_{\max}$
`/* --- Recycling --- */`
For each track with $r <$ recycle thr: add its Gaussians (scaled by $r \cdot w$) to PPP
remove track from T
`/* --- Estimate extraction --- */`
Compute existence probs r_i
Compute ESF, $\rho(n)$, mode N
Pick top-N tracks by r_i, take max-weight Gaussian
Store $\hat{X}_k$
`/* --- Diagnostics --- */`
Compute $\mathbb{E}[n]$, $\mathrm{Var}[n]$ from $\rho(n)$
Print diagnostics: tracks and PPP size
return $\hat{X}$

Algorithm 3: K-Shortest Paths (Yen's Algorithm)

Input: Cost matrix G, source s, destination d, number of paths k, DAG flag $useDAG$
Output: List of k shortest paths with costs
Data: Hash sets $seenHashes$, $inHeapHashes$; Priority queue B

Initialize result list $A \leftarrow []$, hash sets, priority queue
if $useDAG$ **then**
    $firstPath \leftarrow$ DAGSP(G, s, d)
else
    $firstPath \leftarrow$ BFMSP(G, s, d)
if $firstPath$ *is empty* **then**
    return A
Add $firstPath$ to A and its hash to $seenHashes$
if $k = 1$ **then**
    return A
for $m = 1$ **to** $k - 1$ **do**
    $prevPath \leftarrow A[m-1]$
    for $i = 0$ **to** $|prevPath| - 2$ **do**
        $spurNode \leftarrow prevPath[i]$
        $rootPath \leftarrow prevPath[0 : i+1]$
        Create modified graph G' by removing conflicting edges
        `/* Remove edges to force path deviation */`
        for $j = 0$ **to** $i - 1$ **do**
            Remove all edges incident to $prevPath[j]$ in G'
        Remove edges that would recreate previous paths with same root
        if $useDAG$ **then**
            $spurPath \leftarrow$ DAGSP$(G', spurNode, d)$
        else
            $spurPath \leftarrow$ BFMSP$(G', spurNode, d)$
        if $spurPath$ *is not empty* **then**
            $totalPath \leftarrow rootPath + spurPath[1:]$
            $pathHash \leftarrow$ computeHash$(totalPath)$
            if $pathHash \notin seenHashes$ ***and*** $pathHash \notin inHeapHashes$ **then**
                Compute total cost and add to priority queue B
                Add $pathHash$ to $inHeapHashes$
    Remove paths from B that are already in $seenHashes$
    if B *is empty* **then**
        Break;
    $nextBest \leftarrow B$.pop()
    Add $nextBest$ to A and its hash to $seenHashes$
return A

Algorithm 4: Bellman-Ford-Moore Shortest Path (BFMSP)

Input: Cost matrix G, source node s, destination node d
Output: Shortest path result $(path, cost)$

Initialize distance array $dist[n] \leftarrow \infty$, predecessor array $pred[n] \leftarrow -1$
Build edge list $edges$ from cost matrix G
$dist[s] \leftarrow 0$
`/* Relax edges up to` $n-1$ `times */`
for $iter = 0$ **to** $n-2$ **do**
  $updated \leftarrow$ false
  foreach *edge* $(u, v, weight) \in edges$ **do**
    if $dist[u] \neq \infty$ ***and*** $dist[v] > dist[u] + weight + \epsilon$ **then**
      $dist[v] \leftarrow dist[u] + weight$
      $pred[v] \leftarrow u$
      $updated \leftarrow$ true
  if ***not*** $updated$ **then**
    Break;
`/* Check for negative cycles */`
foreach *edge* $(u, v, weight) \in edges$ **do**
  if $dist[u] \neq \infty$ ***and*** $dist[v] > dist[u] + weight + \epsilon$ **then**
    return *empty path (negative cycle detected)*
if $dist[d] = \infty$ **then**
  return *empty path*
Reconstruct path from $pred$ array
return $(path, dist[d])$

Algorithm 5: DAG Shortest Path (DAGSP)

Input: Cost matrix G, source node s, destination node d
Output: Shortest path result $(path, cost)$

Initialize distance array $dist[n] \leftarrow \infty$, predecessor array $pred[n] \leftarrow -1$
$dist[s] \leftarrow 0$
`/* Process nodes in topological order */`
for $u = s$ **to** $n-1$ **do**
  if $dist[u] = \infty$ **then**
    continue
  for $v = u+1$ **to** $n-1$ **do**
    $w \leftarrow G[u][v]$
    if w *is valid edge* **then**
      $newDist \leftarrow dist[u] + w$
      if $dist[v] > newDist - \epsilon$ **then**
        $dist[v] \leftarrow newDist$
        $pred[v] \leftarrow u$
if $dist[d] = \infty$ **then**
  return Empty Path;
Reconstruct path from $pred$ array
return $(path, dist[d])$

Algorithm 6: Hungarian Algorithm for Optimal Assignment

Input: Cost matrix $C \in \mathbb{R}^{m \times n}$
Output: Optimal assignment vector $assignment[m]$, total cost
Data: Star matrix $starMatrix$, prime matrix $primeMatrix$, covered rows/columns

Initialize assignment vector, cost $\leftarrow 0$
Generate working copy of cost matrix, handle infinite values
Allocate boolean matrices for stars, primes, and coverage
`/* Preliminary steps: reduce matrix */`
if $m \leq n$ **then**
    for *each row* r **do**
        Find minimum value $minVal$ in row r
        Subtract $minVal$ from all elements in row r
else
    for *each column* c **do**
        Find minimum value $minVal$ in column c
        Subtract $minVal$ from all elements in column c
`/* Step 1 & 2a: Initial assignment */`
Create initial starred zeros without conflicts
`/* Main algorithm loop */`
repeat
    `/* Step 2b: Check if done */`
    Count covered columns
    if *covered columns* $= \min(m, n)$ **then**
        Build assignment vector from star matrix
        Break;
    else
        `/* Step 3: Cover zeros */`
        Find uncovered zeros and create primes
        Handle conflicts between starred and primed zeros
        if *augmenting path found* **then**
            `/* Step 4: Augment matching */`
            Update star matrix along augmenting path
            Clear all primes and row coverings
        else
            `/* Step 5: Modify costs */`
            Find minimum uncovered value h
            Add h to covered rows, subtract h from uncovered columns
until *assignment complete*
Compute total assignment cost
return $(assignment, cost)$

Algorithm 7: Murty's Algorithm for M-Best Assignments with Prefix Forcing

Input: Cost matrix C, number of desired solutions m
Output: List of m best assignments and corresponding costs
Data: Priority queue $queue$ (min-heap), assignment list $assignments$

Initialize empty lists $assignments \leftarrow [\,]$, $costs \leftarrow [\,]$
if $m \leq 0$ ***or*** C *is empty* **then**
    return $(assignments, costs)$
$(initialAssignment, initialCost) \leftarrow$ Hungarian(C)
if *no valid solution found* **then**
    return $(assignments, costs)$
Create subproblem $P_0 \leftarrow (C, initialAssignment, initialCost)$
Add P_0 to $queue$
while $queue$ *not empty* ***and*** $|assignments| < m$ **do**
    $current \leftarrow queue$.pop() (lowest-cost subproblem)
    Append $current.assignment$ to $assignments$
    Append $current.cost$ to $costs$
    `/* Generate child subproblems by enforcing prefixes and forbidding current pair */`
    for *each row* r *in* $current.assignment$ **do**
        $col \leftarrow current.assignment[r]$
        if $col < 0$ **then**
            continue
        $C_{child} \leftarrow current.costMatrix$
        `/* 1. Enforce prefix assignments for rows 0 to` $r-1$ `*/`
        for $p \leftarrow 0$ ***to*** $r-1$ **do**
            $fixedCol \leftarrow current.assignment[p]$
            for *each column* j *in* C_{child} **do**
                if $j \neq fixedCol$ **then**
                    $C_{child}[p][j] \leftarrow \infty$
        `/* 2. Forbid current assignment` (r, col) `*/`
        $C_{child}[r][col] \leftarrow \infty$
        `/* 3. Solve the modified problem */`
        $(childAssignment, childCost) \leftarrow$ Hungarian(C_{child})
        if *valid complete assignment found* **then**
            Create subproblem $P_{child} \leftarrow (C_{child}, childAssignment, childCost)$
            Add P_{child} to $queue$
return $(assignments, costs)$

Algorithm 8: Murty Wrapper for (P)LMB Filter

Input: Original cost matrix P_0, number of solutions m
Output: 1-indexed assignments, adjusted costs

if $m \leq 0$ ***or*** *matrix is empty* **then**
  return *empty results*
$n_1 \leftarrow P_0$.rows(), $n_2 \leftarrow P_0$.cols()
`/* Create padded matrix with dummy assignments */`
Create padded matrix $[n_1 \times (n_2 + n_1)]$
$padded$.leftCols$(n_2) \leftarrow P_0$
$padded$.rightCols$(n_1) \leftarrow 0.0$ (dummy assignments)
`/* Make matrix non-negative */`
$minVal \leftarrow padded$.minCoeff()
$padded \leftarrow padded - minVal$
`/* Solve m-best assignment problem */`
$(assignmentsPadded, costsPadded) \leftarrow$ Murty$(padded, m)$
`/* Convert results: strip dummies, adjust to 1-indexing */`
foreach $(assignment_i, cost_i) \in (assignmentsPadded, costsPadded)$ **do**
  Create stripped assignment vector
  foreach *assignment value* a **do**
    if $a < n_2$ **then**
      Add $(a + 1)$ to stripped vector (1-indexed)
    else
      Add 0 to stripped vector (unassigned)
  $adjustedCost \leftarrow cost_i + minVal \times n_1$
  Add stripped assignment and adjusted cost to results
return $(assignmentsStripped, costsAdjusted)$

Algorithm 9: Multiple Kalman Prediction

Input: State means $\{m_i\}$, covariances $\{P_i\}$, transition matrix F, process noise Q
Output: Predicted means $\{m_i^-\}$, predicted covariances $\{P_i^-\}$

Initialize result vectors $mPredicts$, $covsPredicts$
foreach *component* i *in mixture* **do**
  $m_i^- \leftarrow F \cdot m_i$
  $P_i^- \leftarrow Q + F \cdot P_i \cdot F^T$
  Add (m_i^-, P_i^-) to result vectors
return $(mPredicts, covsPredicts)$

Algorithm 10: Multiple Kalman Update

Input: Measurement z, prior means $\{m_i\}$, covariances $\{P_i\}$, measurement matrix H, noise R
Output: Likelihoods $\{q_i\}$, updated means $\{m_i^+\}$, updated covariances $\{P_i^+\}$

$H^T \leftarrow H$.transpose()
Initialize result vectors $qzUpdate$, $mUpdate$, $covsUpdate$
foreach *component* i *in mixture* **do**
  $\mu_i \leftarrow H \cdot m_i$ (predicted measurement)
  $S_i \leftarrow R + H \cdot P_i \cdot H^T$ (innovation covariance)
  `/* Efficient inversion using Cholesky decomposition */`
  $L_i \leftarrow$ cholesky(S_i).matrixL()
  $\det(S_i) \leftarrow (\prod L_i$.diagonal$)^2$
  $S_i^{-1} \leftarrow (L_i^{-1})^T \cdot L_i^{-1}$
  `/* Kalman gain and update */`
  $K_i \leftarrow P_i \cdot H^T \cdot S_i^{-1}$
  $\nu_i \leftarrow z - \mu_i$ (innovation)
  `/* Likelihood computation */`
  $e_1 \leftarrow -\frac{1}{2}\nu_i^T S_i^{-1} \nu_i$
  $e_2 \leftarrow -\frac{1}{2}|z| \log(2\pi)$
  $e_3 \leftarrow -\frac{1}{2}\log(\det(S_i))$
  $q_i \leftarrow \exp(e_1 + e_2 + e_3)$
  `/* State update */`
  $m_i^+ \leftarrow m_i + K_i \cdot \nu_i$
  $P_i^+ \leftarrow (I - K_i H) P_i$
  Add (q_i, m_i^+, P_i^+) to result vectors
return $(qzUpdate, mUpdate, covsUpdate)$

Algorithm 11: Unscented Transform

Input: Mean μ, covariance Σ, nonlinear function $f(\cdot)$, UKF parameters
Output: Sigma points matrix $\mathcal{X}$, weights vector w

$n \leftarrow |\mu|$, $\alpha \leftarrow$ UKF alpha parameter, $\kappa \leftarrow$ UKF kappa parameter
$\lambda \leftarrow \alpha^2(n + \kappa) - n$
$\sqrt{P} \leftarrow$ cholesky$((n + \lambda)\Sigma)$.matrixL()
`/* Generate sigma points */`
$\mathcal{X}[:, 0] \leftarrow \mu$ (center point)
for $i = 0$ **to** $n - 1$ **do**
  $\mathcal{X}[:, i + 1] \leftarrow \mu - \sqrt{P}[:, i]$
  $\mathcal{X}[:, i + 1 + n] \leftarrow \mu + \sqrt{P}[:, i]$
`/* Compute weights */`
$w[0] \leftarrow \lambda/(n + \lambda)$
for $i = 1$ **to** $2n$ **do**
  $w[i] \leftarrow 1/(2(n + \lambda))$
return $(\mathcal{X}, w)$

Algorithm 12: UKF Single Prediction

Input: Prior mean m, covariance P, state transition function $f(\cdot)$, filter parameters
Output: Predicted mean m^-, predicted covariance P^-

Get process noise covariance Q from filter parameters
`/* Create augmented state with process noise */`
$\mu_{aug} \leftarrow [m^T, 0^T]^T \in \mathbb{R}^{n+q}$
$\Sigma_{aug} \leftarrow \begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}$
`/* Generate sigma points for augmented state */`
$(\mathcal{X}_{aug}, w) \leftarrow \text{unscentedTransform}(\mu_{aug}, \Sigma_{aug}, f, \text{params})$
`/* Propagate sigma points through state transition */`
for $i = 0$ **to** $2(n+q)$ **do**
  $x_i \leftarrow \mathcal{X}_{aug}[0:n, i]$ (state part)
  $v_i \leftarrow \mathcal{X}_{aug}[n:n+q, i]$ (noise part)
  $\mathcal{Y}[:, i] \leftarrow f(x_i, v_i, \text{params})$
`/* Compute predicted statistics */`
$m^- \leftarrow \mathcal{Y} \cdot w$
$\mathcal{Y}_{centered} \leftarrow \mathcal{Y} - m^- \mathbf{1}^T$
`/* Adjust weight for covariance (incorporate beta parameter) */`
$\alpha, \beta \leftarrow$ get UKF parameters
$w[0] \leftarrow w[0] + (1 - \alpha^2 + \beta)$
$P^- \leftarrow \mathcal{Y}_{centered} \cdot \text{diag}(w) \cdot \mathcal{Y}_{centered}^T$
return (m^-, P^-)

Algorithm 13: UKF Multiple Prediction

Input: Prior means $\{m_i\}$, covariances $\{P_i\}$, filter parameters
Output: Predicted means $\{m_i^-\}$, predicted covariances $\{P_i^-\}$

Get state transition function from filter parameters
Initialize result vectors $mPredicts$, $covsPredicts$
foreach *component* i *in mixture* **do**
  $(m_i^-, P_i^-) \leftarrow \text{ukfPredictSingle}(m_i, P_i, f, \text{params})$
  Add (m_i^-, P_i^-) to result vectors
return $(mPredicts, covsPredicts)$

Algorithm 14: UKF Single Update

Input: Measurement z, measurement function $h(\cdot)$, prior mean m, covariance P, filter parameters
Output: Likelihood ℓ, updated mean m^+, updated covariance P^+

Get measurement noise covariance R from filter parameters
`/* Create augmented state with measurement noise */`
$\mu_{aug} \leftarrow [m^T, 0^T]^T$
$\Sigma_{aug} \leftarrow \begin{bmatrix} P & 0 \\ 0 & R \end{bmatrix}$
`/* Generate sigma points and propagate through measurement function */`
$(\mathcal{X}_{aug}, w) \leftarrow \text{unscentedTransform}(\mu_{aug}, \Sigma_{aug}, h, \text{params})$
for $i = 0$ **to** $2(n + r)$ **do**
  $x_i \leftarrow \mathcal{X}_{aug}[0 : n, i]$ (state part)
  $n_i \leftarrow \mathcal{X}_{aug}[n : n + r, i]$ (noise part)
  $\mathcal{Z}[:, i] \leftarrow h(x_i, n_i, \text{params})$
`/* Compute predicted measurement statistics */`
$\hat{z} \leftarrow \mathcal{Z} \cdot w$
$\mathcal{Z}_{centered} \leftarrow \mathcal{Z} - \hat{z}\mathbf{1}^T$
$\alpha, \beta \leftarrow$ get UKF parameters
$w[0] \leftarrow w[0] + (1 - \alpha^2 + \beta)$
$S \leftarrow \mathcal{Z}_{centered} \cdot \text{diag}(w) \cdot \mathcal{Z}_{centered}^T$
`/* Compute cross-covariance using efficient method */`
$U \leftarrow \text{cholesky}(S).\text{matrixU}()$
$S^{-1} \leftarrow (U^{-1})^T \cdot U^{-1}$, $\det(S) \leftarrow (\prod U.\text{diagonal})^2$
$\mathcal{X}_{centered} \leftarrow \mathcal{X}_{aug}[0 : n, :] - m\mathbf{1}^T$
$G \leftarrow \mathcal{X}_{centered} \cdot \text{diag}(w) \cdot \mathcal{Z}_{centered}^T$
$K \leftarrow G \cdot S^{-1}$ (Kalman gain)
`/* State update and likelihood */`
$\nu \leftarrow z - \hat{z}$ (innovation)
$m^+ \leftarrow m + K \cdot \nu$
$P^+ \leftarrow P - G \cdot S^{-1} \cdot G^T$
$\log \ell \leftarrow -\frac{1}{2}(\nu^T S^{-1} \nu + |z| \log(2\pi) + \log(\det(S)))$
$\ell \leftarrow \exp(\log \ell)$
return (ℓ, m^+, P^+)

Algorithm 15: UKF Multiple Update

Input: Measurement z, prior means $\{m_i\}$, covariances $\{P_i\}$, filter parameters
Output: Likelihoods $\{\ell_i\}$, updated means $\{m_i^+\}$, updated covariances $\{P_i^+\}$

Get measurement function from filter parameters
Initialize result vectors $likelihoods$, $updatedMeans$, $updatedCovs$
foreach *component* i *in mixture* **do**
  $(\ell_i, m_i^+, P_i^+) \leftarrow \text{ukfUpdateSingle}(z, h, m_i, P_i, \text{params})$
  Add (ℓ_i, m_i^+, P_i^+) to result vectors
return $(likelihoods, updatedMeans, updatedCovs)$

C

Original Research Proposal Plan

Having a well-established plan for research is essential for the success of any project, as such in this chapter an overview of the nominal timeline as well as contingency measures are provided. The planning for the project is based on the recommended structure for the master thesis as described on the official TU Delft website[1] but tailored specifically to this project. The thesis is estimated to be 35 weeks long and a general overview of the planned weekly hours over the project span can be found in Figure C.1. The weekly hours are split between four categories:

1. **Literature:** Hours dedicated to literature review, including (but not limited to) finding/analysing/understanding source/papers.
2. **Code:** Hours dedicated to coding/implementation/debugging of the codebase, but also including derivations and theoretical set-up for the problem.
3. **Writing:** Hours dedicated to writing the reports, deliverables, presentations, and weekly meetings (including meeting notes and prepared questions).
4. **Admin:** Hours dedicated to ensuring that all deliverables and administrative procedures for the various key milestones are performed correctly.

The project is further subdivided into four phases:

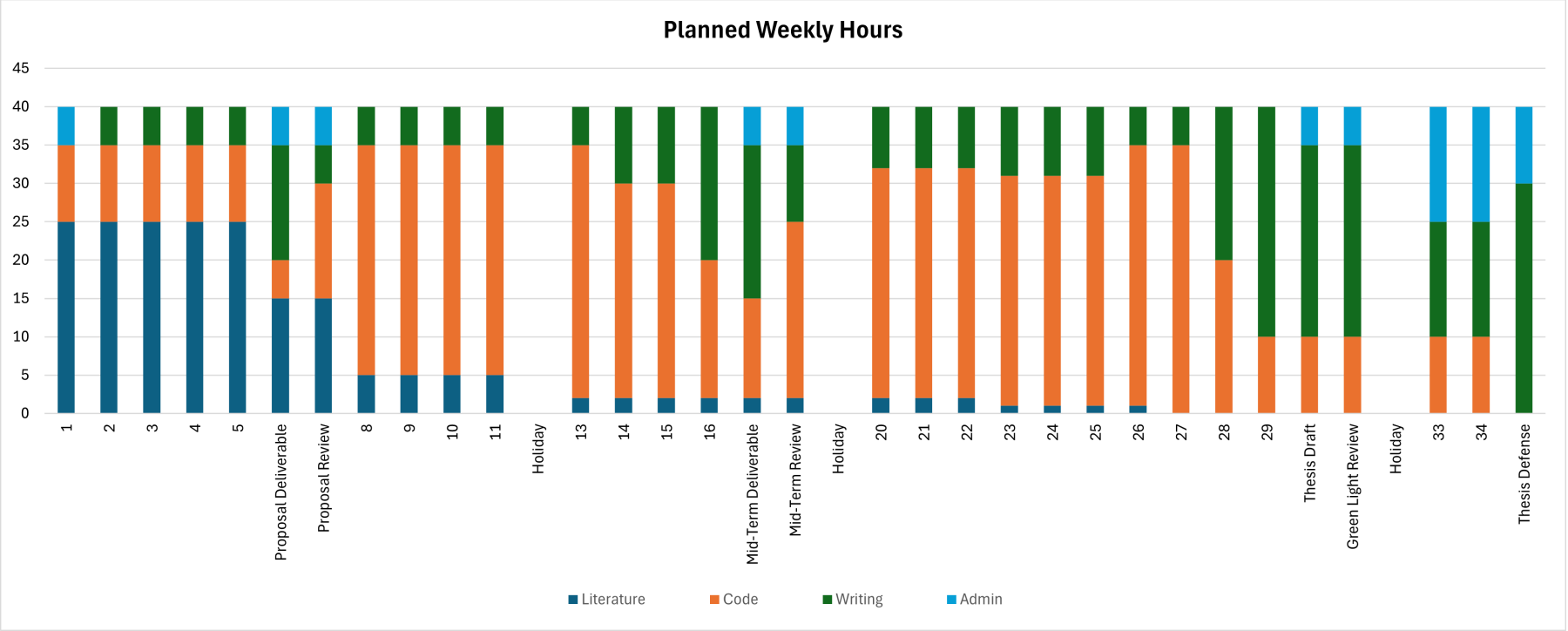
P1. **Literature Review and Research Proposal - Weeks $1 \rightarrow 7$**
During this initial phase the focus will be on gathering the relevant literature, compiling the research proposal, setting up all of the necessary research materials/resources/tools/environments, and starting to implement some basic filters such as the PHD. Nonetheless, the main focus of P1 remains the literature as seen by the planned hour distribution in Figure C.1.

P2. **Research Phase 1 - Weeks $7 \rightarrow 19$**
The focus will shift from literature to coding, implementation, and mid-term review compilation (consisting of a report and oral presentation). During this phase, more advanced (labelled) filters will be implemented, and the first and second research guiding questions will be tackled. Furthermore, the research can/will be further developed/modified for the midterm report.

P3. **Research Phase 2 - Weeks $19 \rightarrow 30$**
By this stage, the research is very well defined and the amount of literature hours is further replaced by coding/implementation/derivation hours. The end of this stage is marked by the thesis draft for which all the estimators/filters of interest have been implemented/compared/tested, and data/codebase/results have been verified/validated/analysed/discussed.

P4. **Research Dissemination - Weeks $30 \rightarrow 35$**
During this phase, the green light review is held and minor improvements and changes can be made to the thesis. Preparations for the thesis defence are made and the weekly hours focus changes from coding to admin and writing hours.

[1] `https://www.tudelft.nl/studenten/lr-studentenportal/onderwijs/master/thesis` Accessed 05/02/2025



Milestone	Projected Date	Milestone	Projected Date
Kick-off Meeting	20/01/2025	Thesis-Draft	18/08/2025
Research Proposal	05/03/2025	Green Light Review	Week 31
Mid-Term Paper	19/05/2025	Thesis Defence	Week 35

Figure C.1: Thesis draft timeline for weekly hours distribution

In addition to the presented planned weekly hours in Figure C.1, to keep the project on track the use of a Notion webpage[2] is made to track progress via a daily logbook and three timelines (one for coding, research and writing respectively). The timelines represent three Gant-Charts which are updated weekly with new tasks and subtasks aiming to achieve the general objective of each of the four project phases as described in Table C.2. This method along with Figure C.1 is used to keep flexible during the project without losing sight of the important objectives/work-packages. Last but not least, it should be noted that documenting and writing tasks are being performed along the project each week, so as not to fall into a situation where extensive writing is needed before a deliverable.

Table C.2: Main objectives description per phase and category - Literature (L), Code(C), Writing(W), Admin(A) - non-critical tasks marked with a star ★

Phase ID	Objective ID	Description
P1	A-1.1	Kick-Off meeting and submission of the Kick-off Form
	A-1.2	Submit Research Proposal and Proposal Review
	L-1.1	Select relevant literature using the PRISMA method
	L-1.2	Analyse selected literature
	C-1.1	Set-up coding environments and resources (Python, C++, Tudat, DelftBlue, CLion)
	★ C-1.2	Implementation of basic filters and scenarios to get a feeling (PHD)
	W-1.1	Compile the research proposal
P2	A-2.1	Apply for DelftBlue access
	A-2.2	Mid-Term Deliverable $\rightarrow$ submit report, give oral presentation
	L-2.1	Identify additional required materials for case-specific problem encounters
	★ L-2.2	Check current publications and journals for new relevant materials weekly
	C-2.1	Implement advanced (labelled) filters/estimators (at least PHD, LMB, PMBM)
	★ C-2.2	Implement unit/system tests
	C-2.3	Implement comparison/benchmarking structures
	★ W-2.1	Write codebase documentation
	W-2.2	Write midterm report
	W-2.3	Compile midterm presentation
P3	A-3.1	Submit Thesis Draft
	A-3.2	Apply for green-light review
	L-3.1	Identify additional required materials for case-specific problem encounters
	★ L-3.2	Check current publications and journals for new relevant materials weekly
	C-3.1	Implement advanced (labelled) filters/estimators (PMBM, LMB variations)
	C-3.2	Implement unit/system tests
	C-3.3	Data/results validation and verification
	C-3.4	Data/results analysis/discussion

[2]Link to request access: `https://www.notion.so/invite/f880f1fcdd4ad8d4bb27bb731e1a19f5c085981c`

	C-3.5	Implement mixed estimator labelled/unlabelled RSO dependent
	★ C-3.6	Compile for Delft Blue
	W-3.1	Thesis Draft
	★ W-3.2	Codebase Documentation
P4	A-4.1	Request examination and committee
	A-4.2	Research Submission
	A-4.3	Thesis Defence
	C-4.1	Results analysis/discussion
	W-4.1	General report improvements
	W-4.2	Thesis defence preparation

Contingency Plan

Having contingency plans prepared in advance is important when planning the research and even though it is hard to predict what the future holds, knowing which tasks are critical and which may be changed, delayed or deleted is essential. The weekly hours planning presented in the nominal plan allows for flexibility when it comes to task distribution, and having the objectives split into critical and non-critical allows for additional flexibility. Since the hours per package/category are allocated in advance and the specific task hours are allocated each week in notion for the next week, with the logbook data it is possible to track the progress and apply one of the following contingency scenarios:

- **Unfulfilled Hours** due to external factors (e.g. administrative/health appointments, delays, material malfunctions) - the hours will be re-allocated during the week, either during weekdays or weekends, depending on scenarios, and can also be re-allocated to a further week.
- **Task Delayed** - if a tasks gets delayed and it is marked as non-critical in Table C.2 it can be rescheduled, delayed or deleted. However, if it is a critical task, additional hours will be allocated at the end of the day/weekends until the task is performed if it is believed to be a time issue. Finally, the holidays were planned at strategic points such that in a worst-case scenario, they can also be used for additional hours.
- **Task Infeasibility** - By the midterm review, an idea of the implementation of all filter categories would have been acquired. As such, in case a filter is too problematic or a research question needs to be modified, this can be done before the submission of the midterm. It should still be noted that at least one member of each RFS branch presented in Figure 2.2 would ideally be implemented uniquely, such that the best combinations are implemented for the combined labelled/unlabelled estimator.

C.1. Experimental Set-Up Ideal

For this research, the most important resource is the computational power accessible which specifications are included in Table C.3. This consists of the personal laptop of the student and access to one DelftBlue[3] node of either compute type-a or compute type-b. It should be noted that the access request for DelftBlue will be submitted after the research proposal phase in P2, where the exact compute specifications will be made clear. Furthermore, there exists additional GPU-specific nodes and high memory nodes which are not believed to be needed at this stage. For the computing power, when upscaling the problem to a (very) large population of objects it is expected for the personal laptop to be a limiting factor,

Table C.3: Available computing resources during the project

Personal Laptop - Windows 11 / WSL2 (Ubuntu 20/22)	
Model	Asus ROG Zephyrus M16

[3] `https://www.tudelft.nl/dhpc/system` (Accessed 19/02/2025)

CPU	Intel i9-12th Gen
GPU	NVIDIA RTX 3070Ti - CUDA architecture 86
SSD	1 TB
RAM	32 GB

	DelftBlue - Linux
	Compute type-a
Cores	48
CPU	2x Intel XEON E5-6248R 24C 3.0 GHz
Memory	192 GB
SSD	480 GB

	Compute type-b
Cores	64
CPU	2x Intel XEON E5-6448Y 32C 2.1 GHz
Memory	256 GB
SSD	480 GB

In addition to the computational power, several open-source source software/codebases and programming languages will be employed during the project, as well as paid software/licences accessible through TU Delft's student account. As such a short description of the most important resources is listed below:

Programming Language

Even though the main programming language for the project is C++17 (note that some of the software which will be used compiles code in C++14), visualising data can also be performed with Python-3.10 (or above). Additionally compiling the C++ code will be performed using CMake 3.22.1 (or above), and g++ 11.4.0 (or above). Moreover, even though some of the open-source repositories accompanying the selected literature (notably the Vo&Vo research) have been written in MATLAB, whose licence is provided by TU Delft, the project codebase will not include MATLAB code. Finally, all the reports will be written using LaTeX.

Integrated Development Environments (IDE)

The main IDEs which will be used are Visual Studio Code[4] (free) for any Python code involved and CLion[5] (JetBrains Licence required - provided by TU Delft) for the C++ code as it allows linking the environment with WSL2 and is the recommended option for compiling Tudat software on Windows based OS. As an alternative, it is possible to use the Visual Studio 2022 Community version[6] (free) if needed but would require more steps to set up the environment.

Open Source Repositories

The most important open-source repositories which will be used for the project are *Tudat*[7] for all of the astrodynamics propagations and simulations, *metis*[8] by S. Gehly and *RFS tracking toolbox*[9] by B. Vo for verification, benchmarking and inspiration. In addition, it is worth mentioning *Rerun.io*[10] as visualization software that can potentially be used for both plotting and debugging.

[4] `https://code.visualstudio.com/` (Accessed 19/02/2025)
[5] `https://www.jetbrains.com/clion/` (Accessed 19/02/2025)
[6] `https://visualstudio.microsoft.com/vs/` (Accessed 19/02/2025)
[7] `https://github.com/tudat-team/tudat-bundle` (Accessed 19/02/2025)
[8] `https://github.com/gehly/metis` (Accessed 19/02/2025)
[9] `https://ba-tuong.vo-au.com/codes.html#` (Accessed 19/05/2025)
[10] `https://rerun.io/` (Accessed 19/05/2025) - version 0.19.0 or above

Data Management & Miscellaneous

For data management, the codebase will be saved both locally and remotely using *Github*, the reports will be written using *Overleaf*, all presentations will be created using *PowerPoint*, and all flowcharts will be created using *Lucidchart*. Moreover, *Mendeley* and *Research Rabbit* are used for the reference management. Furthermore, *Grammarly* will be used for grammar and spelling. Last but not least, it is important to mention that *doxygen* will be used to generate the codebase documentation.