FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	Multiple target tracking with external information
Student:	Bc. Andrey Babushkin
Supervisor:	doc. Ing. Kamil Dedecius, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2023/2024

Instructions

Abstract: The state-of-the-art multiple-target tracking (MTT) algorithms can be used to monitor several observed targets simultaneously. They are mostly not limited to permanently existing targets but also admit their random spawning and disappearance, misdetections, and false alarms. However, some targets may be temporarily or permanently hidden from detection due to the sensing principles. For instance, the primary active radars may not see targets that move at low altitudes. If there are other sensors that can provide any (external) information about these targets, it could be beneficial to incorporate this information.

Goal: The aim of the thesis is to propose a framework fusing external information into the knowledge of the multiple-target tracking algorithm. The steps are as follows:
1) Study and describe a selected MTT algorithm, its working principles, advantages, and disadvantages.
2) Propose a method for fusing external information into the knowledge of the MTT algorithm.
3) Experimentally assess the feasibility of the resulting framework. That is, provide some simulation examples and evaluate the results.
4) Discuss the properties of the proposed framework, its performance, advantages and disadvantages. Suggest possible future research directions.

References:
[1] Á. F. García-Fernández and B. -N. Vo, "Derivation of the PHD and CPHD Filters Based on

---

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 20 January 2023 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Direct Kullback–Leibler Divergence Minimization," in IEEE Transactions on Signal Processing, vol. 63, no. 21, pp. 5812-5820, Nov.1, 2015, doi: 10.1109/TSP.2015.2468677.
[2] C. Fantacci, B. -N. Vo, B. -T. Vo, G. Battistelli and L. Chisci, "Robust Fusion for Multisensor Multiobject Tracking," in IEEE Signal Processing Letters, vol. 25, no. 5, pp. 640-644, May 2018, doi: 10.1109/LSP.2018.2811750.
[3] B. Ristic, B. -T. Vo, B. -N. Vo and A. Farina, "A Tutorial on Bernoulli Filters: Theory, Implementation and Applications," in IEEE Transactions on Signal Processing, vol. 61, no. 13, pp. 3406-3430, July1, 2013, doi: 10.1109/TSP.2013.2257765.
[4] D. E. Clark, K. Panta and B. -n. Vo, "The GM-PHD Filter Multiple Target Tracker," 2006 9th International Conference on Information Fusion, Florence, Italy, 2006, pp. 1-8, doi: 10.1109/ICIF.2006.301809.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 20 January 2023 in Prague.

Master's thesis

MULTIPLE TARGET TRACKING WITH EXTERNAL INFORMATION

Bc. Andrey Babushkin

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: doc. Ing. Kamil Dedecius, Ph.D.
May 3, 2023

Contents

List of Figures

List of Tables

First and foremost, I would like to express my heartfelt gratitude to doc. Ing. Kamil Dedecius, Ph.D. for his unwavering support and guidance throughout the course of this thesis. His invaluable advice, expertise, and continued positive outlook on the world have been a source of inspiration and motivation.
I would also like to extend my thanks to prof. Ing. Václav Hlaváč, CSc. and Mgr. Radoslav Škoviera, Ph.D. for giving me the opportunity to work as a researcher and for introducing me to the fascinating field of multi-target tracking.
Furthermore, I wish to express my deepest appreciation to my family and my girlfriend, who provided me with unwavering support during the challenging times and kept me motivated throughout the journey.
Last, but not least, I express my gratitude and love to my friends Ing. Mykyta Boiko and Ing. Vladyslav Zavirskyy, with whom I started my academic journey when we enrolled in the Bachelor's program together and with whom we have been together throughout the years until the end of our Master's degree.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 3, 2023 .....

Abstract

The thesis focuses on the multi-target tracking problem in cluttered environments with the uncertainty on the number of targets and using Bayesian inference. In this work, we mainly focus on the popular Gaussian mixture probability hypothesis density (GM-PHD) filter and introduce a technique to include additional information for cases when a sensor fails to detect targets due to environmental or physical limitations. We provide all required theoretical background from the basics of probability theory to the discussion of various multi-target tracking methods and the introduction of the Finite Set Statistics (FISST) framework. We also conclude with an extensive performance measurement and analysis of results, where we demonstrate that the proposed fusion technique significantly improves tracking results. Finally, we discuss the limitations of the filter and propose possible measures to overcome them.

Keywords Bayesian filtering, multi-target tracking, random finite sets, Gaussian Mixture Probability Density filter, external information fusion

Abstrakt

Tato práce se zaměřuje na problém sledování více cílů v prostředí s vysokým množstvím rušení a nejistotou ohledně počtu sledovaných objektů pomocí Bayesovské inference. V této práci se hlavně zaměřujeme na populární Gaussian mixture probability hypothesis density (GM-PHD) filtr a představujeme techniku, jak zahrnout dodatečné informace pro případy, kdy senzor nedokáže detekovat sledované objekty z důvodu fyzických nebo prostředkových omezení. Poskytujeme veškeré potřebné teoretické pozadí od základů teorie pravděpodobnosti až po diskusi o různých metodách sledování více cílů a představení rámce Finite Set Statistics (FISST). Dále zahrnujeme rozsáhlé měření výkonnosti a analýzu výsledků, kde ukazujeme, že navržená technika fúze výrazně zlepšuje sledovací výsledky. Nakonec diskutujeme omezení filtru a navrhujeme možné způsoby, jak je překonat.

Klíčová slova Bayesovské filtrování, sledování více cílů, náhodné množiny, gaussian mixture probability density filtr, fúze externích informací

List of Acronyms

CPEP	Circular Position Error Probability
CV	Constant Velocity
EKF	Extended Kalman filter
FISST	Finite Set Statistics
FoV	Field of view
GM	Gaussian mixture
GM-PHD	Gaussian Mixture Probability Density
HO-MHT	Hypothesis-oriented Multiple Hypothesis Tracker
JPDA	Joint Probabilistic Data Association
KF	Kalman filter
MAP	Maximum a posteriori
MHT	Multiple Hypothesis Tracker
MLE	Maximum Likelihood Estimator
MMSE	Minimum Mean Square Error
MTT	Multi-target tracking
PDA	Probabilistic Data Association
PHD	Probability hypothesis density
PPP	Poisson Point Process
RFS	Random Finite Set
TO-MHT	Track-oriented Multiple Hypothesis Tracker
UKF	Unscented Kalman Filter

Introduction

Object tracking is a critical problem in signal processing and computer vision. The primary goal of tracking algorithms is to estimate the states of moving targets based on a sequence of sensor measurements in environments with a high degree of uncertainty. Different domains of the problem require different approaches to target tracking. In this work, we provide an overview of the existing areas of target tracking and introduce the scope of this study.

Tracking algorithms can be categorized based on the number of targets they track, whether it is a fixed or known number of targets, or a variable and unknown number of targets. When the number of targets in the tracking area is greater than one, we refer to it as multi-target tracking (MTT). Object tracking is widely used in various fields, including robotics, autonomous vehicles, surveillance, and medical imaging. With the recent emergence of deep learning algorithms, there has been renewed interest in MTT, and many new deep learning-based approaches have been proposed. However, the learned model predictive power is questionable, as these approaches estimate the internal object dynamics using complex non-linear models, while the Bayesian approach assumes fixed known model that are specified manually according to some prior knowledge, for instance, physical laws. In this study, we focus on MTT using Bayesian inference, the statistical approach based on recursive updates of the target posterior distribution using a set of measurements.

Bayesian algorithms are classified based on the results they achieve [1, p. 11]. The first type of algorithms are the Bayesian smoothers, which remove the noise of states based on past and future measurements, i.e., they smooth the signal. The other type are the Bayesian predictors, which predict the future state of a target more than one step ahead based on past measurements. The last type are the Bayesian filters, which predict the current state of a target based on measurements up to the moment of the predicted state. The Kalman filter, the fundamental algorithm for many modern and sophisticated tracking methods, is an example of Bayesian filters. This study focuses on Bayesian filters, and we cover the Kalman filter in detail.

Algorithms can also be separated based on the type of sensor and measurements used. There are several sets of algorithms available to address specific problems depending on the sensors used. For instance, point object tracking is used when the sensor generates one measurement per object, such as a radar or a camera with an object recognition algorithm. When lidars are used, one target generates a set of measurements, and this is referred to as extended object tracking [2]. In surveillance, especially when tracking people, group object tracking algorithms are used to estimate groups of individuals rather than every individual separately [3]. Additionally, tracking with multipath propagation refers to scenarios where a sensor receives more than one detection caused by the multipath phenomenon or multiple reflections from surrounding objects of the same signal [4]. Finally, tracking with unresolved targets occurs when several targets produce only one measurement [5]. In this study, we focus on point object tracking, assuming that one target produces at most one measurement, and each measurement can be generated by at most

one target.

This study aims to enhance the performance of the Gaussian Mixture probability hypothesis density (GM-PHD) filter, a popular Bayesian filter for multi-target tracking (MTT), by proposing a novel approach to incorporate external information. Specifically, we provide a detailed description of the GM-PHD filter and its internals, and demonstrate through an example how the filter may fail under environmental limitations. We then introduce our proposed method for integrating external information and analyze its impact on the filter's performance. Finally, we evaluate the filter's strengths and weaknesses in various scenarios and discuss the significance of our contribution in improving the accuracy and robustness of the GM-PHD filter for MTT.

Structure

The thesis is structured as follows. Chapter 1 provides the theoretical background necessary for understanding the rest of the thesis, including the introduction of probability theory and probability distributions and the Bayes' rule. Next, we present the Bayesian inference framework, which is the basis for the general Bayes filter, a recursive algorithm for state estimation. Finally, we present the concepts of state-space models and the Kalman filter.

Chapter 2 is dedicated to multi-target tracking. It starts with an overview of target tracking methods, followed by an introduction to Random Finite Sets (RFS). The chapter then describes the PHD filter and the PHD function. Next, we give a PHD filter formal definition, and then observe the Gaussian Mixture PHD filter. The chapter concludes with a discussion of track maintenance, the GM-PHD recursion, and ends with the framework for external information fusion.

Chapter 3 describes the implementation and testing methodology of the GM-PHD filter. We then introduce several metrics that are used to measure the performance of multi-target tracking algorithms, then we describe four testing scenarios. The final section on parameters testing and methodology explains how the tests were conducted and how the results were analyzed.

Chapter 4 analyzes the results of the tests conducted in Chapter 3. This chapter includes sections on test results for each scenario and a discussion of the results, where a comprehensive analysis of the filter strengths and weaknesses is provided.

The final Chapter 5 concludes this work by summarizing the main contributions of the thesis and highlighting areas for future research.

Chapter 1

Theoretical background

The field of target tracking, both single and multiple, rely on mathematical concepts and methods based on probability theory. In this chapter, we provide an introduction to key concepts of probability and inference. We start by reviewing the basics of probability theory, including random variables, probability distributions with examples of the most important ones for multi-target tracking, and the Bayes' theorem. Then, we describe the concept of state-space models, a powerful framework that allows to describe dynamic system evolving in time. Then, we apply these concepts with a Bayes filter, a general method for recursive estimation the state of a dynamic system using measurements obtained from a sensor over time. Finally, we define and prove two important in tracking theorems, and use these theorems to derive the Kalman filter, and important and widely used framework that provides an optimal state estimate of a linear system in the presence of Gaussian noise. The theoretical foundations in this chapter serve as a basis for the next chapter, which covers more sophisticated algorithms for tracking multiple targets in noisy environments.

1.1 Probability theory foundations

When we are dealing with probability of an event, we assume that there is a possibility of that event occurring and we measure it using a number between 0 and 1 or a percentage between 0% and 100%. In other words, we use the probability theory framework to assign numerical values to arbitrary events. This section covers several fundamental concepts of probability theory, which serve as the basis for this work.

The *event*, which we formally denote as E, comes from some space of all possible events, the *outcome space* Ω. We also denote the *probability* of the event E as $\Pr\{E\}$. This probability is a real non-negative number, that is $\Pr\{E\} \in \mathbb{R}, \Pr\{E\} \geq 0$. The outcome space covers all possible events, that is $\Pr\{\Omega\} = 1$. It then follows that the probability of disjoint events from the outcome space Ω is the sum of probabilities of these events, that is for $E_1, \ldots, E_n \in \Omega, \Pr\{\bigcup_{i=1}^{n} E_i\} = \sum_{i=1}^{n} \Pr\{E_i\}$.

We have defined three main axioms of probability theory. In addition to these axioms, several crucial concepts illustrate the relationship between events. Given two events, E_1 and E_2, the *conditional probability* of E_1 given E_2 is defined as

$$\Pr\{E_1 \mid E_2\} = \frac{\Pr\{E_1 \cap E_2\}}{\Pr\{E_2\}}. \tag{1.1}$$

If the events are *independent* of each other, the probability of them occurring simultaneously is given by

$$\Pr\{E_1 \cap E_2\} = \Pr\{E_1\} \Pr\{E_2\}. \tag{1.2}$$

These relationships allow us to define the *law of total probability* [6, p. 31], which is a key component of Bayes' rule. Let A be an event, $A \int \Omega$, and $\{B_n : n = 1, 2, \ldots\}$ be a countable partition of the space Ω. Then, we have:

$$\Pr\{A\} = \sum_n \Pr\{A \mid B_n\} \Pr\{B_n\}. \tag{1.3}$$

Now, we can deduce the following rule:

▶ **Definition 1.1** (Bayes' rule)**.** *Let A and B be two events from the outcome space Ω. Then the following applies:*

$$\Pr\{A \mid B\} = \frac{\Pr\{B \mid A\}}{\Pr\{B\}} = \frac{\Pr\{B \mid A\}}{\sum_n \Pr\{B \mid A_n\} \Pr\{A_n\}}. \tag{1.4}$$

Bayes' rule plays a crucial role in Bayesian inference and serves as the basis for Bayesian filters, including the PHD filter.

1.1.1 Probability distributions

Events and their probabilities are not sufficient for our purposes; we need a framework that allows us to obtain a formal, general description of a set of events from some outcome space. Specifically, we need an abstraction around events, which is some function that maps the outcome space Ω to some other space, typically $\mathbb{R}$ (but not necessarily). This abstraction is called a *random variable* and one outcome of it is called a *realization*.

▶ **Definition 1.2** (Random variable and its realization)**.** *Let Ω be a set of possible events, $\omega \in \Omega$ is some event, and $\mathbb{O}$ be another space. A function $X : \Omega \to \mathbb{O}$ is called a random variable and $x = X(\omega)$ is called a realization of X.*

The above definition provides a general framework for understanding random variables. However, for the purposes of Bayesian statistics, we will simplify the discussion by focusing on scalar, continuous-valued random variables, represented by $\mathbb{O} = \mathbb{R}$. In practice, it is often impractical to calculate the probability of every possible event from the outcome space Ω, especially for continuous random variables with uncountably infinite outcomes. Instead, it may suffice to work with intervals on the outcome space and their probabilities. We call such a probability distribution a *cumulative distribution function (cdf)* of X, and the first-order derivative of the cdf is called a *probability density function (pdf)*.[1]

▶ **Definition 1.3** (Cumulative distribution function (cdf))**.** *Given a scalar real-valued random variable $X \in \mathbb{R}$ and x as a realization of X, we define the probability $\Pr\{X \in (-\infty, x)\} = \Pr\{X < x\}$ as the cumulative distribution function of random variable X and denote it as $P(x)$.*

▶ **Definition 1.4** (Probability density function (pdf))**.** *The probability density function $p(x)$ of a scalar real-valued random variable X is defined as:*

$$p(x) = \frac{\partial P(x)}{\partial x}. \tag{1.5}$$

[1] For discrete random variables, the corresponding distribution function is called a *probability mass function (pmf)*.

Probability distributions are fundamental in Bayesian statistics and provide a way to model and analyze the behavior of random variables. In the next section, we will discuss some common probability distributions used in Bayesian filters, including the Gaussian distribution and the Poisson distribution.

Probability distributions are fundamental in Bayesian statistics and provide a way to model and analyze the behavior of random variables. Cdfs and pdfs (as well as pmfs for discrete cases) are used to specify the probability distribution of a random variable, providing a formal description of the relationship between events and probabilities. In addition to this formal description, we also need to know several statistical properties of probability distributions to fully understand their behavior. Two main properties of probability distributions are the expected value, $E[X]$, and the variance, $\text{Var}[X]$:

$$E[X] = \int_{-\infty}^{\infty} xp(x)dx, \tag{1.6}$$

$$\text{Var}[X] = E\left[(X - E[X])^2\right] = \int_{-\infty}^{\infty} (x - E[X])^2 p(x)dx. \tag{1.7}$$

The expected value, or the first moment, represents the most probable value of a random variable. The variance, or the second central moment, is a measure of the variability of a random variable and is often denoted as $\text{Var}[X] = \sigma^2$. Furthermore, we often study the relationship between two random variables, say X and Y, and measure their joint variability, which we call a *covariance*:

$$\text{Cov}[X, Y] = E\left[(X - E[X])(Y - E[Y])\right] = \int_{-\infty}^{\infty} (x - E[X])(y - E[Y])p(x, y)dxdx. \tag{1.8}$$

For vector-valued random variables, that is $X, Y \in \mathbb{R}^k$ and $\mathbf{x}$ is the realization of X, the formulas are the following. It is worth noting that the covariance in the vector case becomes a matrix and is called a *covariance matrix*:

$$E[X] = \int_{-\infty}^{\infty} \mathbf{x}p(\mathbf{x})d\mathbf{x}, \tag{1.9}$$

$$\text{Var}(X) = E\left[(X - E[X])\,(X - E[X])^\intercal\right] \tag{1.10}$$

$$= \int_{-\infty}^{\infty} (\mathbf{x} - E[X])\,(\mathbf{x} - E[X])^\intercal\, p(\mathbf{x})d\mathbf{x}, \tag{1.11}$$

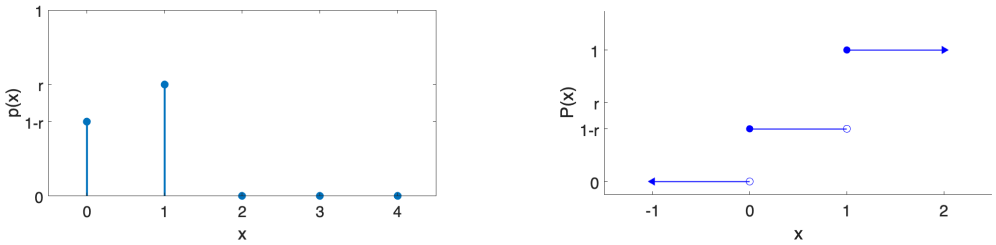
$$\text{Cov}[X, Y] = E\left[(X - E[X])\,(Y - E[Y])^\intercal\right] \tag{1.12}$$

$$= \int_{-\infty}^{\infty} (\mathbf{x} - E[X])\,(\mathbf{y} - E[Y])^\intercal\, p(\mathbf{x}, \mathbf{y})d\mathbf{x}d\mathbf{y}. \tag{1.13}$$

Probability distributions are fundamental in Bayesian statistics and provide a way to model and analyze the behavior of random variables. For the purpose of Bayesian filters, some common probability distributions are particularly useful. In this section, we will discuss some of these distributions and their main properties. We shall start from discrete cases and go on to the continuous variables.

1.1.1.1 Bernoulli distribution

The Bernoulli distribution is the simplest discrete probability distribution, and the Bernoulli random variable is binary, with only two realizations, 0 or 1. This distribution is parameterized with r, the probability of the positive outcome. Its probability mass function is given by



(a) PMF. **(b)** CDF.

Figure 1.1 Bernoulli distribution.

$$p(x) = \begin{cases} 1-r & \text{if } x = 0, \\ r & \text{if } x = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{1.14}$$

The pmf and the cdf of the Bernoulli distribution can be seen on Figure 1.1. We denote this distribution as $p(x) = \text{Bernoulli}(x; r)$. The expected value and the covariance of a Bernoulli random variable X are

$$E[X] = r, \qquad \text{Var}[X] = r(1-r). \tag{1.15}$$

In object tracking, we use Bernoulli random variables to model the existence of an object at some time step k, which we will discuss in detail in the following sections.

1.1.1.2 Binomial distribution

Binomial random variables are the generalization of Bernoulli random variables, representing the probability of x positive outcomes after n consecutive and independent trials. This distribution is parameterized by the probability of the positive outcome in one trial r and the number of trials n. Its pmf, as well as the expected value and the variance, are given by

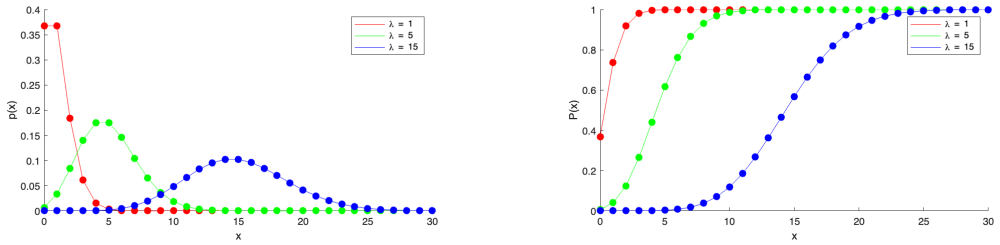
$$\begin{aligned} p(x) &= \binom{n}{x} r^x (1-r)^{(n-x)}, \\ E[X] &= nr, \\ \text{Var}[X] &= nr(1-r). \end{aligned} \tag{1.16}$$

1.1.1.3 Poisson distribution

The Poisson distribution is a discrete random distribution that can be used as an approximation of the Binomial distribution in cases where n tends towards infinity and r tends towards 0 such that their product stays about equal to a parameter λ, which is the parameter of the Poisson distribution and also its expected value and variance. The outcome space of a Poisson random variable is $\mathbb{N}_0$, that is natural numbers including 0, and its pmf, expected value and variance are given by

$$p(x) = \text{Poisson}(x; \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, \tag{1.17}$$

$$E[X] = \text{Var}[X] = \lambda. \tag{1.18}$$



(a) PMF. **(b)** CDF.

Figure 1.2 Poisson distribution.

The probability mass function and the corresponding cumulative density functions of the Poisson distribution can be seen in Figure 1.2. The Poisson distribution is a critical component of modern multi-object tracking systems, as it is used to model noise measurements, and some systems even use it to model objects that exist but are not visible in the field of view of a sensor [7]. The Poisson distribution is the foundation of the so-called Poisson Point Processes (PPP), which will be discussed later.

1.1.1.4 Uniform distribution

The uniform distribution is a continuous probability distribution and it describes such random variables which outcomes on some interval $[a, b]$ are possible with equal probability. This distribution is parameterized by a and b and its pdf, $E[X]$ and $\mathrm{Var}[X]$ are:

$$p(x) = \mathrm{Uniform}(x; [a, b]) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq x \leq b, \\ 0 & \text{otherwise,} \end{cases} \tag{1.19}$$

$$E[X] = \frac{a+b}{2}, \tag{1.20}$$

$$\mathrm{Var}[X] = \frac{(b-a)^2}{12}. \tag{1.21}$$

The uniform distribution is generally used to represent the uncertainty when all outcomes are equally possible. In this work, we use the uniform distribution to model positions of clutter measurements at some time step k.

1.1.1.5 Gaussian distribution

The Gaussian distribution, also known as the normal distribution, is a continuous probability distribution that is widely used in many branches of statistics, including Bayesian filtering. It is typically used to model a large number of independent and identically distributed (i.i.d.) random variables.

The Gaussian distribution is characterized by two parameters: the mean μ and the variance σ^2. Its probability density function (pdf), expected value, and variance are given by:

$$p(x) = \mathcal{N}\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{1.22}$$

$$E[X] = \mu, \tag{1.23}$$

$$\mathrm{Var}[X] = \sigma^2. \tag{1.24}$$

In the notation $\mathcal{N}\left(x;\mu,\sigma^2\right)$, the first parameter x means "evaluated at." Despite the cdf of the Gaussian distribution not having a closed-form representation, the distribution possesses several desirable properties that allow for closed-form solutions in many applications, including object tracking. Although it may be too simple to model certain scenarios, it represents the approximate average position of objects and their uncertainty quite well. This work utilizes mixtures of Gaussians, which will be discussed in more detail later.

1.2 Bayesian inference

In statistics, there are two ways to understand the uncertainty. The first one, called *frequentist*, assumes that the source of uncertainty lays in the nature of events. If we modeled a random process following this approach, we would calculate parameters of the model using their maximum likelihood estimation, which in fact is the probability density function if the parameter evaluated at the observed data (we will talk more about estimators in Section 1.2.2). In addition to the estimated parameter value, we could also calculate confidence intervals, which would give us a range where a possible true value of the parameter can lay considering some probability of possible error.

The *Bayesian* approach has a different philosophy. It main assumption is that the uncertainty origin is in the modeling itself and that this are we who have limited knowledge about the ground true model. This difference leads to a completely distinct path of estimating values of a model. At the beginning, we give parameters a *prior distribution*, our initial belief where true values of parameters may lay. Next, after every new data piece we update the distribution using the Bayes' rule. Using the *likelihood function*, which represents our updated beliefs about the parameter after observing the data, and the prior, we compute the *posterior distribution*, the updated belief about the value of the parameter. And, in an every subsequent step, the posterior becomes a new prior.

One of the main differences, however, in these two approaches is the output of such estimation. While in the frequentist statistics we get values of parameters, the Bayesian approach will give us a full posterior distribution of parameter values. From such a distribution, we can extract information for our needs, such as an estimation of a value or some uncertainty measure.

In object tracking, the Bayesian approach to estimation has several advantages. Firstly, we can estimate the posterior as time passes, one measurement at a time. This is good because rarely do we have all measurements in advance, and often we want our systems to work in an online manner. Secondly, full posteriors allow us to work with the uncertainty of estimation and implement techniques to reduce the number of new hypotheses using merging techniques.[2]

Last but not least, the Bayesian approach allows us to incorporate prior knowledge about the motion models of objects and their birth positions. All of the above helps to improve tracking performance and reduce the impact of noisy measurements.

1.2.1 Bayes' rule in terms of pdfs

In Bayesian object tracking, probability distributions are used instead of pure probabilities. Therefore, Bayes' rule must be defined using distributions. Fortunately, this is straightforward after we define the conditional probability in terms of pdfs.

▶ **Definition 1.5** (Conditional probability for pdfs)**.** *Let x and y be random variables with pdfs $p(x)$ and $p(y)$, respectively, and the joint distribution $p(x,y)$. Then the conditional probability of x given y is defined as:*

$$p(x|y) = \frac{p(x,y)}{p(y)}. \tag{1.25}$$

[2] As we will see later, the GM-PHD filter uses the Mahalanobis distance between Gaussians to decide what hypotheses should be merged into one. The computation of Mahalanobis distance includes the covariance matrix.

▶ **Definition 1.6** (Bayes' rule for pdfs)**.** *Let z and x be random variables with densities $p(z|x)$ and $p(x)$, respectively, and let $p(z, x)$ be their joint distribution. The Bayes' rule for $p(x|z)$ is defined as follows:*

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} = \frac{p(z|x)p(x)}{\int p(z|x)p(x)dx}. \tag{1.26}$$

The distribution $p(x|z)$ is the posterior distribution, $p(z|x)$ is the likelihood, $p(x)$ is the prior, and $p(z)$ is called *evidence*.

Evidence here is only the normalization constant for the distribution so that the integral of the posterior equals to one. It is therefore convenient to omit this denominator in the text and write the posterior only in terms of prior and likelihood. Since it is not already the equality, we use the proportionality symbol:

$$p(x|z) \propto p(z|x)p(x). \tag{1.27}$$

We can use the rule defined above to compute the posterior of some parameter based on all the data. However, in many real-world applications, measurements do not arrive all at once, but rather sequentially over time. In the context of object tracking, sensors generate measurements in a discrete manner, once per some predefined time interval, and the goal of the tracking algorithm to estimate targets' state at each time step. Fortunately, the construction of the Bayes' rule allows us to define the sequential data update in a very straightforward manner.

▶ **Theorem 1.7** (Sequential Bayes' rule)**.** *Let z_k be an observed random variable at discrete time steps $k = 1, 2, \ldots$ and let $z_{1:k-1}$ represent the realizations of z_k obtained from the time step $k = 1$ up to $(k-1)$, that is $z_{1:k-1} = \{z_1, z_2, \ldots, z_{k-1}\}$. Let $p(x_k|z_{1:k-1})$ denote the posterior distribution obtained at time $k-1$ using all the measurements up to $k-1$ and $p(z_k|x_k)$ be the likelihood of the new measurement z_k. The posterior distribution $p(x_k|z_{1:k})$ at time step k is:*

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|x_k)p(x_k|z_{1:k-1})dx_k} \propto p(z_k|x_k)p(x_k|z_{1:k-1}). \tag{1.28}$$

It should be mentioned that x_k and z_k must not always be scalars. As we will later see, they can be vectors, denoted as $\mathbf{x}_k$ and $\mathbf{z}_k$, or even sets, denoted X_k and Z_k. The Bayes' rule will have the same form in any case.

The application of the Bayes' rule has one big disadvantage. In practice, calculating the posterior can be challenging or even intractable. The evidence term in the denominator contains the integral of the product of the likelihood and the prior over the whole measurement space. This product may (and often will) produce functions whose integration cannot be expressed in a closed-form solution.

However, we can obtain a tractable solution by choosing a prior distribution from a set of *conjugate distributions* with respect to the likelihood. In this case, the posterior will be in the same form as the prior and we obtain an explicit closed-form solution. A conjugate prior for a given likelihood function is a prior distribution that, when used in combination with the likelihood, leads to a posterior distribution that is in the same family as the prior.

For example, if the likelihood is expressed using the Bernoulli distribution and the prior is the Beta distribution, the posterior will also be a Beta distribution. The same applies for the Gaussian distribution, where if both the prior and the likelihood are Gaussian, the posterior is also Gaussian. This is particularly useful in practice, as many distributions have well-known conjugate priors. The latter case is used, for instance, in the GM-PHD filter.

1.2.2 Estimators

As mentioned before, the output of the Bayesian inference is always a distribution. However, we are often interested in obtaining an estimated value of the parameter we are estimating.

Estimators do exactly that. Strictly speaking, estimators take a set of observations and produce an estimate of the value of an unknown parameter of a distribution. We denote an estimate of an unknown parameter x as $\hat{x}$. There are several known estimators but we are interested in two of them: ML and MAP.

The *Maximum Likelihood (ML)* estimator finds the value of the parameter such that it maximizes the likelihood function. Formally:

$$\hat{x}_{\text{ML}} = \arg\max_x p(z|x). \tag{1.29}$$

The ML estimator is a point estimator that finds the parameter value that makes the observed data most probable. For instance, the frequentist approach uses this estimator to find the parameter value. The MLE does not take the prior distribution into account and is more sensitive to outliers [8]

The *Maximum A Posteriori (MAP)* estimator, on the other hand, finds the value of the parameter that maximizes the posterior distribution. In formal notation:

$$\hat{x}_{\text{MAP}} = \arg\max_x p(x|z) = \arg\max_x p(z|x)p(x). \tag{1.30}$$

The MAP estimator finds the most probable value of the parameter given the observed data. Comparing to the MLE, MAP uses prior information and more robust when the data is noisy or incomplete. That is the reason why MAP is often used in Bayesian inference.[3]

1.3 Gaussian-linear Bayesian filtering

We have now introduced the main principles of Bayesian inference. Now, we can turn to the Kalman filter, a popular and widely used algorithm for state estimation. The Kalman filter (KF, for short) is a recursive algorithm that uses Bayesian inference to estimate the state of a dynamic system based on a sequence of noisy measurements.

The key feature of the KF is the ability to predict future states of the system. This is essential for applications that require to foresee the behavior of tracked objects before the state of these objects is measured. Examples of such systems may include autonomous driving or air defence systems. The prediction ability helps also to overcome situations when a sensor fails to measure the position of an object (a so-called misdetection).

In the following subsections, we will introduce the state-space models, give a formal definition of a general Bayes filter, infer the Kalman filter formulas and discuss the Constant Velocity (CV) model, a common motion model used in the Kalman filter.

1.3.1 Multivariate Gaussian Distribution

Before we start learning the basics of Bayesian filtration, we should extend our knowledge of the Gaussian distribution to the multidimensional case. As mentioned in Section 1.1.1.5, it plays a very important role in object tracking and, in particular, in this work. Formulas and theorems defined in this section are essential for defining and proving the Kalman filter formulas. We will define several properties of multivariate Gaussians required later.

First, we define the multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$, covariance matrix Σ, and evaluated at $\mathbf{x}$ as:

$$\mathcal{N}(\mathbf{x}; \mu, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{1.31}$$

[3] In this work, we use Gaussian posterior distributions and MAP estimates are equal to posterior mean estimates, or MMSE-estimates. However, for general distributions, it will rarely be the case.

where n is the length of the vector $\mathbf{x}$, and $|\cdot|$ denotes the determinant of a matrix.

The multivariate Gaussian distribution is a generalization of the univariate Gaussian distribution, where instead of a single mean and variance, we have a mean vector and a covariance matrix that characterizes the correlation between the variables. Note that the exponent contains the expression known as the Mahalanobis distance, which we will encounter throughout this work. It is given by:

▶ **Definition 1.8** (Mahalanobis distance)**.** *For vectors* $\mathbf{x}$ *and* $\mathbf{y}$ *and the symmetrical positive-definite matrix* $\mathbf{S}$*, the Mahalanobis distance is defined as:*

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\intercal \mathbf{S}^{-1} (\mathbf{x} - \mathbf{y})}. \tag{1.32}$$

As we will later see, in object tracking, we heavily use conditional probabilities, and in particular, conditioning and marginalization of joint distributions of Gaussian random variables. Thus, we define the following two theorems.

▶ **Theorem 1.9** (Conditioning on a Gaussian joint distribution)**.** *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be Gaussian random variables with distributions* $\mathcal{N}(\mathbf{x}; \mu_x, \mathbf{\Sigma}_{xx})$ *and* $\mathcal{N}(\mathbf{y}; \mu_y, \mathbf{\Sigma}_{yy})$*, respectively. Let their joint probability be given by:*

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_{xx} & \mathbf{\Sigma}_{xy} \\ \mathbf{\Sigma}_{xy}^\intercal & \mathbf{\Sigma}_{yy} \end{bmatrix} \right). \tag{1.33}$$

Then the conditional distribution of $\mathbf{x}$ *given* $\mathbf{y}$ *is defined as:*

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_{x|y}, \mathbf{\Sigma}_{x|y}\right), \tag{1.34}$$

where

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \mathbf{\Sigma}_{xy} \mathbf{\Sigma}_{yy}^{-1} (\mathbf{y} - \boldsymbol{\mu}_y), \tag{1.35}$$

$$\mathbf{\Sigma}_{x|y} = \mathbf{\Sigma}_{xx} - \mathbf{\Sigma}_{xy} \mathbf{\Sigma}_{yy}^{-1} \mathbf{\Sigma}_{xy}^\intercal. \tag{1.36}$$

▶ **Theorem 1.10** (Marginalization of a Gaussian joint distribution)**.** *Consider the same* $\mathbf{x}$*,* $\mathbf{y}$*,* $\mathcal{N}(\mathbf{x}; \mu_x, \mathbf{\Sigma}_{xx})$*,* $\mathcal{N}(\mathbf{y}; \mu_y, \mathbf{\Sigma}_{xx})$ *and* $p(\mathbf{x}, \mathbf{y})$ *given in Theorem 1.9. The marginal distribution of* x *is defined as:*

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu_x, \mathbf{\Sigma}_{xx}). \tag{1.37}$$

The proofs for Theorem 1.9 and Theorem 1.10 can be found in classical statistical textbooks such as [9, pp. 161–163].

1.3.1.1 Gaussian mixtures

In multi-target tracking, the posterior density cannot be described in a simple Gaussian distribution. Generally, the posterior distribution can have any form with only assumption that the integral of it sums to one. However, when we are dealing with Gaussian-linear cases, the posterior density is often represented as a mixture of many Gaussian components. A Gaussian mixture is a linear combination of multiple Gaussian distributions and its pdf is expressed in the following way:

$$p(\mathbf{x}) = \sum_{i=1}^{N} w_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \Sigma_i), \tag{1.38}$$

where N is the number of Gaussian components in the mixture, w_i is the weight of the ith component, the weights $w_i \geq 0$ sum to one, i.e. $\sum_{i=1}^{N} w_i = 1$, and each component $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \Sigma_i)$

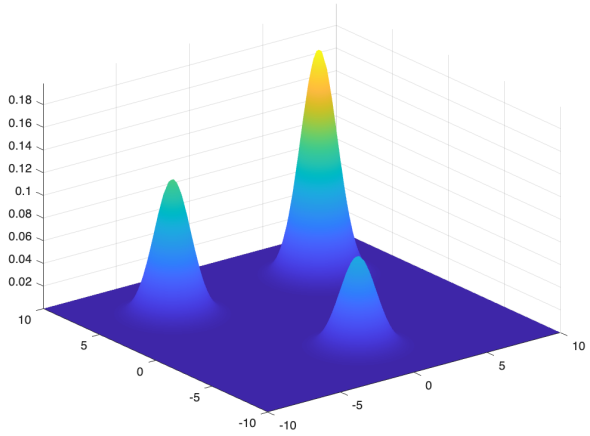


Figure 1.3 An example of a Gaussian mixture with three components. The first component is centered at $[0, -5]^\intercal$ with weight 0.2, the second component has mean in $[-5, 5]^\intercal$ with weight 0.3 and the last has mean $[5, 5]^\intercal$ and the weight 0.5.

is a Gaussian distribution with the mean in $\boldsymbol{\mu}_i$ and the covariance matrix Σ_i. An example of a Gaussian mixture pdf is illustrated in Figure 1.3.

Since the weights sum to one, the Gaussian mixture satisfies the normalization requirement of pdfs and is thus a valid probability density function itself. The weights in the mixture represent the relative importance of the corresponding component. Gaussian mixtures are used to model complex distributions and have several nice mathematical properties as the Gaussian distribution, as we will later see. That is why Gaussian mixtures are widely used to represent posterior distributions in many MTT filters. Moreover, one can easily approximate any distribution with a Gaussian mixture using algorithms such as the Expectation-Maximization algorithm [10].

A mixture $f(\mathbf{X})$ with N components will have the expected value $\hat{\boldsymbol{\mu}}$ and the covariance $\hat{\Sigma}$ according to the following equations:

$$\hat{\boldsymbol{\mu}} = \sum_{i=1}^{N} w_i \boldsymbol{\mu}_i, \tag{1.39}$$

$$\hat{\Sigma} = \sum_{i=1}^{N} w_i \Sigma_i + \tilde{\Sigma}, \tag{1.40}$$

where the term $\tilde{\Sigma}$ is called the spread-of-the-innovations and is defined as:

$$\tilde{\Sigma} = \sum_{i=1}^{N} w_i (\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{\mu}_i - \hat{\boldsymbol{\mu}})^\intercal. \tag{1.41}$$

The spread-of-the-innovations quantifies the magnitude of the difference between expectations of individual components.

1.3.2 State-space model

In the previous section, we established the relationships between prior and posterior distributions and learned about recursive Bayesian estimation of the posterior. Now, we will explore these concepts in the context of object tracking.

Object tracking involves estimating the internal states of objects that move in some space, which are often unknown to us. These states may include physical properties of objects such as position, velocity, acceleration, and orientation, depending on the application. Sensors such as cameras, infrared scanners, lidars or radars constantly generate measurements of the object states, which may be the distance of an object to the sensor, or its temperature. However, these measurements are often noisy due to environmental conditions or imperfections of the sensor. We use Bayesian filtering to estimate the real state of the objects and filter out the noise.

To handle the complexity and diversity of possible situations and properties, we need a systematic framework for modeling object motion that captures all sources of uncertainty and the nature of physical behavior of objects. This framework is called state-space modeling. In state-space models, the internal state we want to estimate is represented as a vector of variables that evolve over time according to a set of rules specified by the *motion model*, which is a known function. The measurements obtained at each time step are modeled as a different function of the state variables, known as the *measurement model*[4].

For example, consider a moving bicycle and a surveillance camera with a rectangular field of view (FoV). The bicycle enters the FoV and crosses it at a constant speed. The camera has an algorithm that estimates the position of objects in the space from the image. With each frame f, it returns a new position, say $(\hat{x}_f, \hat{y}_f)$. We can represent the state of the bicycle as a vector of the position and velocity, which are unknown variables. Based on the measurements, we estimate their values using the known physical relation between distance, velocity, and time: $d = v\Delta t$, where d is the distance that the object with velocity v traveled in time Δt.

Both the motion and measurement models should include some noise. In the example above, the movement of the bicycle cannot perfectly follow the formula, and there will always be some minor changes in speed. Moreover, the measurement model, which translates an estimated state into a measurement, will also contain some uncertainty since the camera does not have a perfect representation of the space.

We can now describe these models formally. Note that since the state is a vector, we will use $\mathbf{x}$ for vectors. The motion and measurement models can be generally expressed as:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = f_t(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k), \tag{1.42}$$
$$p(\mathbf{z}_k|\mathbf{x}_k) = h_t(\mathbf{x}_k, \mathbf{v}_k), \tag{1.43}$$

where f_t and h_t are functions, $\mathbf{x}_{k-1}$ is the estimated state from time step $k-1$, $\mathbf{u}_k$ is called the input (or control) vector, and $\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean noise random variables.

However, this definition is too general since the functions f_t and h_t can be anything. As mentioned earlier, to get a closed-form solution in the Bayesian inference framework, we need to choose conjugate distributions for the likelihood and the prior. The Kalman filter, which we will describe soon, works for the Gaussian-linear case, where the functions f_t and h_t are linear and noise variables are distributed as Gaussian, i.e., $\mathbf{w}_k, \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \Sigma)$. The Gaussian- linear state-space model has the following representation:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k, \qquad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{1.44}$$
$$p(\mathbf{z}_k|\mathbf{x}_k) = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{1.45}$$
$$p(\mathbf{x}_0) \sim \mathcal{N}(\hat{\mathbf{x}}_0), \mathbf{P}_0), \tag{1.46}$$

where:

[4]It should be mentioned that the names 'motion' and 'measurement' models are not the only terms used to describe them. The reader may also encounter terms such as kinematic, state-transition, dynamic, or system models for the motion model, and terms like observation, sensor, or likelihood models for the measurement model. All of these names are correct, and their use depends on the context in which they are applied. However, in this work, the author strives for consistency and will use the terms 'motion' and 'measurement' models throughout the entire text.

- $\mathbf{F}$ is the *transition matrix*,
- $\mathbf{B}$ is the *input matrix*,
- $\mathbf{H}$ is the *measurement matrix*,
- $\mathbf{Q}$ and $\mathbf{R}$ are symmetric positive definite matrices that describe the statistical properties of the motion noise $\mathbf{v}_k$ and the measurement noise $\mathbf{w}_k$, respectively,
- $\hat{\mathbf{x}}_0$ and $\mathbf{P}_0$ are mean and variance of the prior state.

The control variable $\mathbf{u}_k$, in general, represents some input signal from the environment, and $\mathbf{B}$ specifies how the input signal affects the dynamic system. This may include the effect of gravity on the vertical traveled distance or the voltage applied to some circuit. However, in object tracking, we rely on measurements obtained from sensors to update our estimate of the object state. We do not have control over the object motion. Thus, the input matrix is often redundant, and we omit it (or set it to a zero matrix) along with the input variable $\mathbf{u}_k$.

The notation above may be inconvenient in some cases. In this work, we will often use a shorter version that has the same meaning:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{F}\mathbf{x}_{k-1}, \mathbf{Q}), \tag{1.47}$$
$$p(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}), \tag{1.48}$$
$$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}_0; \hat{\mathbf{x}}_0, \mathbf{P}_0). \tag{1.49}$$

1.3.2.1 Constant Velocity Model

The Constant Velocity (CV) model is one of the most commonly used motion and measurement models in the context of object tracking. It describes the kinematics of objects in a 2D space that move with a constant velocity. The state vector comprises a position vector and a velocity vector, and while the position vector contains the coordinates of the object in the space, the velocity vector contains the object speed in the direction of each axis.

The CV model is linear and one of the simplest models. It assumes that the speed of objects remains unchanged during tracking, with only small deviations from the constant. This model can be reasonably utilized in scenarios where objects do not change their speed or direction, such as vehicles on highways or planes in the sky. Since the state vector contains only four variables, the use of this model does not increase the computational burden on tracking algorithms.

The motion equation for the CV model is given by 1.47, and the state vector $\mathbf{x}_k$, the state transition matrix $\mathbf{F}$, and the process noise matrix $\mathbf{Q}$ can be defined as:

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ v_{x_1,k} \\ v_{x_2,k} \end{bmatrix}; \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{Q} = q^2 \cdot \begin{bmatrix} \frac{dt^3}{3} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} \\ \frac{dt^2}{2} & 0 & dt & 0 \\ 0 & \frac{dt^2}{2} & 0 & dt \end{bmatrix}, \tag{1.50}$$

where dt is the change in time between the last estimation and the newly computed one, and q is the motion noise parameter, which represents the uncertainty in the state transition.

The measurement model transforms a state vector from the state space into the measurement space. Since the vanilla Kalman filter, which we will derive soon, works only with linear models, the measurement model for the CV model assumes sensors measurements in the same 2D space as in the state vectors. The measurement equation is given by 1.48, and the measurement matrix $\mathbf{H}$ and the measurement noise matrix $\mathbf{R}$ can be defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \quad R = r^2 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{1.51}$$

where r is the measurement noise parameter, which determines the variance of the measurement noise.

We need to describe the choice of parameters q and r in more detail. These parameters are crucial in obtaining good estimates from a filter. These parameters are not known a priori and their values should be chosen carefully by the means of trial and error, or using automated methods like, for instance, in [11]. However, selecting appropriate values often involves a trade-off between tracking accuracy and computational complexity, and the use of exact methods depends on the application and one's requirements.

The CV model can be extended with additional information, such as change of speed. In this way we will come to the constant acceleration model. However, it will not be used in this work and, therefore, we will not leave formal definition of this model here.

1.3.3 The Bayes filter

We have learned how the internal state of a system can be represented in terms of a state vector and motion and measurement models. Now, we are ready to present the formal definition of the Bayes filter, the general abstraction of any Bayesian filter, including the Kalman filter and the PHD filter.

First, we need to make a very important assumption on states. This assumption is called the *Markov model property*. This property states that, for the motion model, the present state of a system x_k is dependent only on the state on the previous time step x_{k-1} given all past states $x_{1:k-1}$ and measurements $z_{1:k-1}$. More formally:

$$p(x_k|x_1, \ldots, x_{k-2}, x_{k-1}, z_1, \ldots, z_{k-2}, z_{k-1}) = p(x_k|x_{k-1}). \tag{1.52}$$

A similar requirement must hold for measurement model, that is:

$$p(z_k|x_1, \ldots, x_{k-2}, x_{k-1}, x_k z_1, \ldots, z_{k-2}, z_{k-1}) = p(z_k|x_k). \tag{1.53}$$

The Markov property may seem restrictive but in reality it is not because state- space models allows us to capture dependencies in the system dynamics by simply introducing more variables into the state vector. For instance, if we have a system where the current state has a dependence on the previous two states, we can augment the state vector to include the last two states as variables, and the Markov property will still hold.

The Bayes filter uses the motion and measurement models and assumes that this property holds. The Bayes filter is a recursive framework that estimates an internal state of the system over time using measurements. Every iteration of the filter consists of two steps: prediction and update. On the prediction step, the filter estimates a internal state x_k based on the previous state x_{k-1} and the motion model $p(x_k|x_{k-1})$. The prediction step is also known as the Chapman-Kolmogorov equation.

▶ **Theorem 1.11** (The prediction step. The Chapman-Kolmogorov equation)**.** *Given the set of measurements $z_{1:k-1} = \{z_1, z_2, \ldots, z_{k-1}\}$ and the current state x_{k-1} and the motion model $p(x_k|x_{k-1})$, the prediction step of the Bayes filter is computed as follows:*

$$p\left(x_k|z_{1:k-1}\right) = \int p\left(x_k, x_{k-1}|z_{1:k-1}\right) \mathrm{d}x_{k-1} = \int p\left(x_k|x_{k-1}\right) p\left(x_{k-1}|z_{1:k-1}\right) \mathrm{d}x_{k-1}, \tag{1.54}$$

where the integral is taken over the entire state space of x_{k-1}, and $p\left(x_{k-1}|z_{1:k-1}\right)$ is the posterior density of the state at time $k-1$, given all measurements up to time $k-1$.

Next, on the update step, the filter corrects the predicted state with the measurement z_k using the measurement model $p(z_k|x_k)$. This step is computed using the standard Bayes' rule.

▶ **Theorem 1.12** (The update step)**.** *Given the output of the prediction step of the Bayes filter* $p(x_k|z_{1:k-1})$*, the observed measurement* z_k *and the measurement model* $p(z_k|x_k)$*, the update step of the Bayes filter is computed as follows:*

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)\,p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \propto p(z_k|x_k)\,p(x_k|z_{1:k-1})\,. \tag{1.55}$$

These two steps create a loop, and to use the filter, we first predict the next state using the Chapman-Kolmogorov equation, and then we update our guess with a measurement. Note, that we will often use the simplified notation to explicitly state the time step of the value. The notation $\bullet_{k|k-1}$ represents the predicted value at time step k, and the notation $\bullet_{k|k}$ represents the updated value at time step k after incorporating a measurement.

Now, having introduced the general Bayes filter, we will continue with exploring the Kalman filter in detail, the popular and widely used tool for state estimation.

1.3.4 The Kalman filter

The Kalman filter is one of the most well-known and widely used algorithms in signal processing and control theory. It is a recursive algorithm that allows the estimation of internal states of entities in dynamic systems from a set of measurements that may be noisy or missing at some time steps. The filter was proposed by Rudolf Kalman in 1960 [12] and has since been pervasively used to control a vast array of consumer, health, commercial, and defense products [13].

The development of the filter was motivated by the need to improve aerospace technology in the United States during the Cold War between the Soviet Block and the North American Treaty Organization. Because the Soviet Union managed to launch its artificial satellites and successfully send a human to space, the federal government of the United States supported research into new technologies in the aerospace area.

The Kalman filter is an example of a general Bayes filter that was introduced earlier. This means that the filter estimates the posterior distribution of the internal state at discrete time steps using prior information about the observed object state and a set of noisy measurements. As briefly mentioned in Section 1.3.2, the Kalman filter works on state-space linear models with Gaussian noise and a Gaussian prior of the state. The predict- update loop in the Kalman filter is the same as in the Bayes filter, and the same Chapman-Kolmogorov equation defined in 1.11 and the update equation defined in 1.12 are incorporated.

The main advantage of the Kalman filter is that it allows for the estimation of the state of a system in real-time. It handles noisy measurements well and is fast; however, it is sensitive to initial parameter settings, such as the noise covariance matrices [14]. Nonetheless, there are new methods being developed that propose mechanisms to overcome this drawback, as in [15] and [16].

One of the main drawbacks of the Kalman filter is its Gaussian-linear assumption. In real-world applications, many systems exhibit non-linearity, and the Kalman filter may be ineffective. Nonetheless, several extensions of the filter have been proposed that address this. Two well-known algorithms are the Unscented Kalman Filter (UKF) and the Extended Kalman filter (EKF). The UKF is an algorithm that uses a set of carefully chosen sigma points to capture the true mean and covariance of the predicted and updated distributions without the need for linearization [17]. The EKF, on the other hand, linearizes the nonlinear motion and measurement models using a first-order Taylor expansion [18]. These methods have been proven to be effective in many applications, including the PHD filter. However, they are beyond the scope of this thesis and will not be covered in detail.

1.3.4.1 The Gaussian identity

Before we introduce the actual formulas of the Kalman filter, we should introduce a new fundamental theorem in object tracking and then deduce a corollary from it. These will also be used later when we infer formulas for the the GM-PHD filter. We define this theorem using a general notation without any meaning for Bayesian filters to avoid the confusion of variables when we use this theorem in different parts of different filters.

▶ **Theorem 1.13** (The Gaussian product identity)**.** *Given matrices and vectors A, U, m, d, and V of appropriate dimensions, and that U and V are positive definite, the following identity applies:*

$$\mathcal{N}(x; Ay + d, U)\mathcal{N}(y; m, V) = \mathcal{N}(x; Am + d, U + AVA^\intercal)\mathcal{N}(y; \hat{m}, \hat{V}), \tag{1.56}$$

where

$$\hat{m} = m + K(y - Am - d), \tag{1.57}$$
$$\hat{V} = (I - KA)V, \tag{1.58}$$
$$K = VA^\intercal(U + AVA^\intercal)^{-1}. \tag{1.59}$$

Proof. The proof of Theorem 1.13 presented here follows similar proofs in [19] (Appendix D) and [20] (Section 3.8). However, the main idea of using algebraic manipulation of matrices was taken from [21].

The proof is based on the idea that both sides of 1.56 represent the joint Gaussian distribution of two random variables, $p(x, y)$. From 1.5, we know that we can express this distribution as:

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x). \tag{1.60}$$

We claim that the left-hand side (LHS) of 1.56 represents this equality, i.e.,

$$p(x, y) = p(x|y)p(y) = \mathcal{N}(x; Ay + d, U)\mathcal{N}(y; m, V). \tag{1.61}$$

Next, we can express the product of two Gaussians in their quadratic forms, the same form as defined in 1.31, i.e.

$$\mathcal{N}(x; Ay + d, U)\mathcal{N}(y; m, V) = \frac{1}{(2\pi)^n\sqrt{|U||V|}} \exp\left(-\frac{1}{2}\Big[(x - Ay - d)^\intercal U^{-1}(x - Ay - d) + (y - m)^\intercal V^{-1}(y - m)\Big]\right). \tag{1.62}$$

If we introduce a small algebraic trick for matrices:

$$\begin{bmatrix} I & -A \\ 0 & I \end{bmatrix}\begin{bmatrix} x - Am - d \\ y - m \end{bmatrix} = \begin{bmatrix} x - Am - d - Ay + Am \\ y - m \end{bmatrix} = \begin{bmatrix} x - Ay - d \\ y - m \end{bmatrix}, \tag{1.63}$$

we can manipulate the exponent to obtain another quadratic form:

$$
\begin{aligned}
&(x - Ay - d)^\intercal U^{-1}(x - Ay - d) + (y - m)^\intercal V^{-1}(y - m) \\
&= \begin{bmatrix} x - Ay - d \\ y - m \end{bmatrix}^\intercal \begin{bmatrix} U^{-1} & 0 \\ 0 & V^{-1} \end{bmatrix} \begin{bmatrix} x - Ay - d \\ y - m \end{bmatrix} \\
&= \begin{bmatrix} x - Am - d \\ y - m \end{bmatrix}^\intercal \begin{bmatrix} I & 0 \\ -A^\intercal & I \end{bmatrix} \begin{bmatrix} U^{-1} & 0 \\ 0 & V^{-1} \end{bmatrix} \begin{bmatrix} I & -A \\ 0 & I \end{bmatrix} \begin{bmatrix} x - Am - d \\ y - m \end{bmatrix} \\
&= \begin{bmatrix} x - Am - d \\ y - m \end{bmatrix}^\intercal \left(\begin{bmatrix} I & A \\ 0 & I \end{bmatrix} \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} I & 0 \\ A^\intercal & I \end{bmatrix} \right)^{-1} \begin{bmatrix} x - Am - d \\ y - m \end{bmatrix} \\
&= \begin{bmatrix} x - Am - d \\ y - m \end{bmatrix}^\intercal \begin{bmatrix} U + AVA^\intercal & AV \\ VA^\intercal & V \end{bmatrix}^{-1} \begin{bmatrix} x - Am - d \\ y - m \end{bmatrix}. \qquad (1.64)
\end{aligned}
$$

This expression is also a joint Gaussian distribution $p(x, y)$, but it cannot be split into two independent Gaussians due to the fact that the matrix is not block-diagonal. Therefore, to conclude the proof, we need to infer two other distributions, $p(y|x)$ and $p(x)$, from the derived distribution $p(x, y)$.

Notice that this is also a quadratic form, and it expresses a dependency on both x and y. Therefore, his expression is also a joint Gaussian distribution $p(x, y)$, but it cannot be split into two independent Gaussians due to the fact that the covariance matrix in 1.64 is not block-diagonal. Therefore, to conclude the proof, we need to infer two other distributions, $p(y|x)$ and $p(x)$, from the derived distribution $p(x, y)$.

We are going to obtain the $p(y|x)$ distribution by conditioning on the joint distribution, i.e. using the formula that was presented in Theorem 1.10. For completeness and simplicity, we will write the joint distribution the same way that it is presented in the theorem:

$$
p(x, y) = \mathcal{N}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} Am + d \\ m \end{bmatrix}, \begin{bmatrix} U + AVA^\intercal & AV \\ VA^\intercal & V \end{bmatrix} \right). \qquad (1.65)
$$

Now, using equations 1.34 and 1.36, we will infer:

$$
\begin{aligned}
\mu_{y|x} &= \mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x - \mu_x) \\
&= m + VA^\intercal(U + AVA^\intercal)^{-1}(x - Am - d), \qquad (1.66) \\
\Sigma_{y|x} &= \Sigma_{yy} - \Sigma yx\Sigma_{xx}^{-1}\Sigma_{yx} \\
&= V - VA^\intercal(U + AVA^\intercal)^{-1}AV \\
&= (I - VA^\intercal(U + AVA^\intercal)^{-1}A)V, \qquad (1.67) \\
p(y|x) &= \mathcal{N}\left(y; \mu_{y|x}, \Sigma_{y|x}\right). \qquad (1.68)
\end{aligned}
$$

Using Theorem 1.10, we obtain the expression for $p(x)$:

$$
p(x) = \mathcal{N}(x; Am + d, U + AVA^\intercal). \qquad (1.69)
$$

If we introduce a new notation, $K = VA^\intercal(U+AVA^\intercal)^{-1}$, we obtain the exact same expressions for the Gaussians on the right-hand side (RHS) of the initial theorem. That confirms that both LHS and RHS are equal due to the equivalence of their quadratic forms. And that concludes the proof. ◀

From Theorem 1.13 we can also derive a corollary, that we will also need.

▶ **Corollary 1.14.** *Given matrices and vectors A, U, m, d, and V of appropriate dimensions, and that U and V are positive definite, the following identity applies:*

$$
\int \mathcal{N}(x; Ay + d, U)\mathcal{N}(y; m, V)\mathrm{d}y = \mathcal{N}(x; Am + d, U + AVA^\intercal). \qquad (1.70)
$$

Proof. The proof of Corollary 1.14 is straightforward:

$$\int \mathcal{N}(x; Ay + d, U)\mathcal{N}(y; m, V)\mathrm{d}y = \int \mathcal{N}(x; Am + d, U + AVA^\intercal)\mathcal{N}(y; \hat{m}, \hat{V})\mathrm{d}y \tag{1.71}$$

$$= \mathcal{N}(x; Am + d, U + AVA^\intercal) \underbrace{\int \mathcal{N}(y; \hat{m}, \hat{V})\mathrm{d}y}_{=1} \tag{1.72}$$

$$= \mathcal{N}(x; Am + d, U + AVA^\intercal). \tag{1.73}$$

In 1.71, we applied Equation 1.56 from Theorem 1.13, then, in 1.72 we notice that one of the integrands does not depend on the integration variable y and we take it out from the integral. The integrand that is left under the integral is a Gaussian distribution probability density function, and it integrates to 1. Equation 1.73 is the right-hand side of Equation 1.70. It concludes the proof of Corollary 1.14. ◀

1.3.4.2 The Kalman filter algorithm

We have derived several important formulas and relations, which are essential to infer the formulas and algorithm of the Kalman filter. We should recall that the filter has Gaussian-linear motion and measurement models and is essentially a Bayesian filter. The formal description of the linear models was given in 1.47 and 1.48, the prediction step of the Bayesian filter in Theorem 1.11, and the update step in Theorem 1.12. With all the necessary pieces in place, we can now derive the prediction and update formulas for the Kalman filter.

▶ **Theorem 1.15** (The Kalman filter prediction)**.** *Assume that the motion and measurement models are given by 1.47 and 1.48, respectively, and that the Markov assumption of the Bayes filter holds, so that the Chapman-Kolmogorov equation is applicable. Let the posterior density at time step $k-1$ be denoted as $\mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1})$. Then, the predicted density is given by:*

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \mathcal{N}\left(\mathbf{x}_k; \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\intercal + \mathbf{Q}\right). \tag{1.74}$$

Proof.

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}\right) \mathrm{d}\mathbf{x}_{k-1} \tag{1.75}$$

$$= \int \mathcal{N}\left(\mathbf{x}_k; \mathbf{F}\mathbf{x}_{k-1}, \mathbf{Q}\right) \mathcal{N}\left(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}\right) \mathrm{d}\mathbf{x}_{k-1} \tag{1.76}$$

$$= \int \mathcal{N}\left(\mathbf{x}_k; \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{Q} + \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\intercal\right) \mathcal{N}\left(\mathbf{x}_{k-1}; \bullet, \bullet\right) \mathrm{d}\mathbf{x}_{k-1} \tag{1.77}$$

$$= \mathcal{N}\left(\mathbf{x}_k; \mathbf{F}\hat{\mathbf{x}}_{k-1}, \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\intercal + \mathbf{Q}\right). \tag{1.78}$$

We started with the application of the Chapman-Kolmogorov equation from 1.11 and the substitution of its terms with the suitable distributions. Next, in the equation 1.78, we used the result of Theorem 1.13, and finally we applied Corollary 1.14 to get 1.78. ◀

▶ **Theorem 1.16** (The Kalman filter update)**.** *Assume that the predicted density from 1.15 is $p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \mathcal{N}\left(\mathbf{x}_k; \mathbf{F}\hat{\mathbf{x}}_{k|k-1}, \mathbf{F}\mathbf{P}_{k|k-1}\mathbf{F}^\intercal + \mathbf{Q}\right)$, $p(\mathbf{z}_k|\mathbf{x}_k)$ is the measurement model given in 1.48, and $\mathbf{z}_k$ is the measurement at time k. Then the posterior density after the update step is given by the following relation:*

$$p(\mathbf{x}_k|\mathbf{z}_k) = \mathcal{N}\left(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}\right), \tag{1.79}$$

where the mean and covariance are given by:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}), \tag{1.80}$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}, \tag{1.81}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\intercal(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R})^{-1}. \tag{1.82}$$

Proof.

$$p(\mathbf{x}_k|\mathbf{z}_k) = p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k) \tag{1.83}$$
$$= p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \tag{1.84}$$
$$= \frac{\mathcal{N}\left(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, R\right)\mathcal{N}\left(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}\right)}{\int \mathcal{N}\left(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, R\right)\mathcal{N}\left(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}\right)\mathrm{d}\mathbf{x}_k} \tag{1.85}$$
$$= \frac{\mathcal{N}\left(\mathbf{z}_k; \mathbf{H}\hat{\mathbf{x}}_{k|k-1}, \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R}\right)\mathcal{N}\left(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}\right)}{\mathcal{N}\left(\mathbf{z}_k; \mathbf{H}\hat{\mathbf{x}}_{k|k-1}, \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R}\right)} \tag{1.86}$$
$$= \mathcal{N}\left(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}\right), \tag{1.87}$$

where, according to Theorem 1.13, the values of $\hat{\mathbf{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ are:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}), \tag{1.88}$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}, \tag{1.89}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\intercal(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R})^{-1}. \tag{1.90}$$

This proof utilizes the same Gaussian product identity theorem with its corollary utilizing in addition the application of the Bayes' rule. ◀

Before we conclude this section with the full algorithm of the Kalman filter, it should be noted that straightforward computation of the covariance matrix after the update step is sensitive to round-off numerical errors, and this can be avoided using a different form of the same expression for $\mathbf{P}_{k|k}$ [8]:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H})^\intercal + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^\intercal. \tag{1.91}$$

This form is called the *Joseph form* and is less sensitive to numerical errors. In the algorithm, we will use this form to compute the covariance matrix after the update step. The whole algorithm of the Kalman filter is defined as follows:

The Kalman filter is the best possible linear estimator in the MMSE (minimum mean square error) sense [22]. That means that the Kalman filter achieves the minimum expected squared error between the true and estimated values, among all possible linear estimation methods. In other words, assuming the motion model and the measurement model are linear, and the noise is Gaussian and uncorrelated, the filter provides the optimal estimate. However, as it was mentioned before, there are plenty of modifications that allow to use non-linear models or non-Gaussian correlated noise. In this work, we will use the vanilla Kalman filter.

Algorithm 1 Kalman filter algorithm

procedure KF($\hat{\mathbf{x}}_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$, $\mathbf{z}_k$)
  $\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1} \leftarrow$ KFPredict($\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}$)
  $\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k} \leftarrow$ KFUpdate($\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}$)
  return $\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}$
end procedure

procedure KFPredict($\hat{\mathbf{x}}_{k-1|k-1}$, $\mathbf{P}_{k-1|k-1}$)
  $\hat{\mathbf{x}}_{k|k-1} \leftarrow \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1}$ ▷ Predicted state estimate
  $\mathbf{P}_{k|k-1} \leftarrow \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\intercal + \mathbf{Q}$ ▷ Predicted covariance
  return $\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}$
end procedure

procedure KFUpdate($\hat{\mathbf{x}}_{k|k-1}$, $\mathbf{P}_{k|k-1}$)
  $\mathbf{K}_k \leftarrow \mathbf{P}_{k|k-1}\mathbf{H}^\intercal(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\intercal + \mathbf{R})^{-1}$ ▷ The Kalman gain
  $\hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1})$ ▷ Posterior state estimate
  $\mathbf{P}_{k|k} \leftarrow (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H})^\intercal + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^\intercal$ ▷ Posterior covariance in Joseph form
  return $\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}$
end procedure

Chapter 2

Multi-target tracking

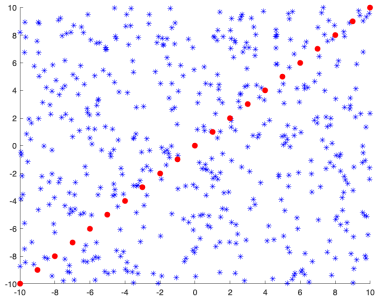
In this section, we provide the theoretical background of the multi-target tracking problem. We have already established the foundations of Bayesian filtering and explained in detail how the Kalman filter works. Now, we will discuss how it differs from other filters that are used for tracking objects. Before we start building the theoretical foundations for object tracking filters, we need to clarify the difference between Bayesian filtering and Bayesian object tracking.

When we track objects, we rely on measurements from sensors, such as cameras, lidars, or radars. These sensors have specific technical specifications and limitations, and there is no sensor that can be 100% reliable. The reliability of a sensor may be affected by noise, which occurs when a sensor detects an object that is not present. This behavior may be caused by weather conditions in the sensor operating area, the limited resolution of the sensor, or dirt or dust covering the sensor surface. We do not address the exact reasons why this happens; we only need to find ways to eliminate noisy measurements and separate them from measurements generated by existing objects.

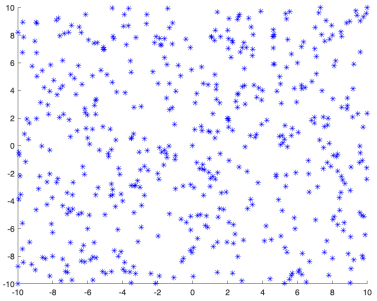
The second problem is closely related to the first. It occurs when a sensor fails to generate measurements for objects that are present in its field of view. The reasons for this may be the same as for noisy measurements, and we do not address these reasons directly. However, we must be able to mathematically model these situations to ensure that they can be properly handled by the filter we want to use to track objects.

These two problems have their names that we will use in this work. Noisy measurements are called *clutter*, and missing measurements are called *misdetections*. We have already seen that the latter problem can be handled by the Kalman filter by skipping the update step. Clutter, on the other hand, creates a much more challenging problem. Figure 2.1 shows what clutter looks like to the filter. In the left image, we see a track of an object and measurements generated by the sensor, shown in red. Clutter measurements are shown as blue asterisks. We can distinguish between measurements and clutter. However, in the right image, we see how the filter sees the same data points, with all points in the same color. To the filter, all these points look the same, and there is no easy way to distinguish between clutter and real measurements.

The uncertainty in distinguishing between real measurements and clutter has led to the development of many methods for addressing this problem. The main idea behind these methods is that, since we have no information about which measurements come from targets and which are clutter, we should consider all measurements at each time step as coming from a target. This involves creating all possible assignments between measurements and targets and then evaluating the probability of each such assignment. In other words, we evaluate the possibility that a given measurement comes from a specific target according to its motion model. If the measurement is too far from the target, the probability of such an assignment will be lower than that of an assignment with a measurement that is close to the predicted state. These assignments are



(a) Real measurements from some object are shown as red dots. Clutter measurements are displayed as blue asterisks.



(b) Both real measurements and clutter measurements are shown as blue asterisks.

Figure 2.1 Measurements in clutter. In this Figure, we indicate the clutter problem. When a sensor generates measurements, some of them may be clutter measurements, and there is no easy way to distinguish which data points come from a real object and which are noise.

referred to as *association hypotheses*. In the following subsection, we will provide a brief overview of various target tracking methods and approaches. But before doing so, we will outline the main assumptions of multi-target tracking (MTT).

2.1 Overview of target tracking methods

There are multiple filters that use association hypotheses. For single-target tracking, when the maximum number of targets is fixed to one, in Gaussian-linear scenarios the Probabilistic Data Association (PDA) filter can be utilized [23]. At each time step, this filter creates new hypotheses for all possible data associations and creates a joint posterior distribution after at the update step. Each hypothesis is assigned with a weight that reflects the likelihood that it actually originates from the target. Next, the resulting Gaussian mixture is reduced to only one Gaussian that represents the posterior state of the target. Both Gaussian mixtures and the reduction techniques will will be discussed later in this work.

This PDA filter is conceptually very simple and straightforward, but it may be too simple for many real-world scenarios, particularly when there are multiple targets present that are too close to each other to be tracked by multiple instance of the PDA filter. This is addressed in the extension and the resulting filter is named the Joint Probabilistic Data Association (JPDA) filter [24]. It can handle multiple targets all at once but only when the number of targets is known in advance. The main idea behind this filter is that at each time step it creates all possible measurement-to-target association hypotheses, and, for each target, it creates a joint probability from all partial probabilities computed for each measurement. The series of such probabilities create tracks, and the track with the highest probability is considered the real track of the object. Because of the way how hypotheses are calculated, the JPDA filter is computationally much more expensive than the PDA filter. Moreover, if tracks get too close to each other, this filter shows the problem called the track coalescence—when tracks of two targets merge into one track.

Both PDA and JPDA filters are single-scan methods, which means they process only one set of measurements at a time. Compared to single-scan methods, there exist multi-scan methods that compute probabilities of tracks based on the history of all measurements, in a hierarchical manner. This way of calculating posterior probabilities leads to significant improvements in tracking accuracy, since the filter takes into consideration the whole history of the object movement. The classical example of a multi-scan filter is Reid's Multiple Hypothesis Tracker (MHT), also known as the hypothesis-oriented MHT (HO-MHT) [25]. This filter is similar to the way it

computes probabilities; however, the calculation of the probability of each association hypothesis incorporates the probability of the parent hypothesis from the previous time step, thus creating a hypothesis tree at each filter cycle. The number of hypotheses is therefore multiplied by the number of new measurements at each step, and the total number of association hypotheses grows exponentially. Efficient implementations of the recursive HO-MHT include advanced techniques on how to prune the number of less probable hypotheses at each time step [26].

The computational complexity of Reid's MHT has led to modifications in the way hypotheses are created. Instead of generating new hypotheses for all parent hypotheses at each time step, we can create several sequences of the best associations for several time steps in the past and compute new hypothesis probabilities for those hypotheses only. This reduction in the computational complexity avoids the need to compute all possible branches in the hypothesis tree. Moreover, it allows for efficient implementation techniques like look-up tables, and there is no recursion in the computation of new hypotheses. This algorithm is known as the track-oriented MHT (TO-MHT) [27]. As a modification of Reid's MHT, TO-MHT is also a multi-scan method, but instead of recursively evaluating all possible hypotheses, it processes all hypotheses at once.

The way both variants of MHT handle track hypothesis initialization and the propagation of association hypotheses over time allows "the MHT approach inherently handle initiation and termination of tracks, and hence accommodate an unknown and time-varying number of targets" [28]. However, because the number of hypotheses grows exponentially, hypothesis reduction techniques should be used, and the pruning of hypotheses with low probabilities should be relatively aggressive. This makes MHT a strong algorithm but with higher computational requirements.

The JPDA filter and the MHT are two approaches for handling multiple targets in a scene. However, there is one alternative approach that is conceptually very different from both methods, which utilizes an abstract mathematical concept called Finite-Set Statistics (FISST) [19]. We will dedicate several next sections to explaining what FISST is, introducing the main theoretical assumptions of the multiple target tracking problem, and then discussing the main building blocks of the PHD filter.

2.2 Random Finite Sets

We have discussed one possible approach for evaluating the uncertainty between targets and measurements in multi-target tracking. Filters such as JPDA or MHT, in both variants, use a hypothesis-based approach where at each time step, a set of hypotheses is created to map all possible associations between new measurements and existing targets. However, these approaches do not model the uncertainty of the number of targets themselves. At any given moment, new targets may appear or disappear from the field of view, and the filters should be able to estimate tracks for every target.

To illustrate this point, let us consider the scenario of a common surveillance camera tracking people in a large mall. At every moment, people enter the area covered by the camera, cross it, and then leave the area. The number of people can vary greatly at different times of the day or on different days of the week, and there is no simple solution for estimating the number of mall visitors at every moment.

In the JPDA filter, the number of hypotheses is assumed to be known in advance, which is a rare case in real-world applications. In the MHT approach, object appearance events can be modeled by assuming that if two or more measurements have a high probability of being assigned to one target, this is probably because the number of objects in the vicinity of the target is greater than one. However, these approaches are not systematic nor optimal, and the exact estimation of the number of targets is rather a side effect [29].

This problem led to the development of the Random Finite Sets (RFSs) theory, a completely new approach to multi-target tracking. At each time step, a collection of objects that are present in the field of view of a sensor is modeled as a random finite set. Generally, a set is a collection of distinct objects without a specific order of the elements. In an RFS, the number of objects

is only known to be finite, but the exact cardinality, or the number of elements in the set, is modeled using some probability distribution. Moreover, all elements in the set are probability distributions. Therefore, RFSs are random variables that model the uncertainty of every state and the number of states as a single entity. An outcome, or a realization, of such a random variable is a fixed set with an exact state, and measurements are considered to be outcomes of the unknown internal state.

The RFS concept led to the development of a new branch in statistics called Finite Sets Statistics (FISST). In FISST, sets are random variables that have their probability distributions and probability density functions. Furthermore, FISST allows the use of RFSs for Bayesian inference, which leads to the need for defining basic algebraic operations on sets, such as integrals and derivatives. As we will soon see, the mathematics behind it becomes complex, and one way to simplify the notation is the introduction of a new way of expressing relations. This concept, called probability generating functionals (p.g.fls), significantly simplifies the notation in FISST. Unfortunately, the underlying level of mathematical abstraction becomes much more difficult to understand.

In this section, we will give a formal definition for a random finite set and also define the main formulas that are needed to understand the logic behind FISST and use it to infer the PHD filter. In this work, we will use standard notation that does not use p.g.fls. However, the reader may refer to the original work in [30] to learn about these concepts.

2.2.1 RFS formal definition

As we mentioned earlier, random finite sets allow us to model object states and the cardinality of the set as a single random variable. At each time step k, we have a set of object states with cardinality n_k. We also assume that states are vectors from some space, and without loss of generality, we assume that this space is $\mathbb{R}^n$. Formally, for every time step k, we have vectors $\mathbf{x}_k^1, \ldots, \mathbf{x}_k^{n_k}$, where $\mathbf{x}_k^i \in \mathbb{R}^d$ for $\forall i$ and $\forall k$. We define the random finite set as follows:

▶ **Definition 2.1** (Random finite set)**.** *Let k be a time step and $\mathbf{x}_k^1, \ldots, \mathbf{x}_k^{n_k}$ be vectors from $\mathbb{R}^d$, where n_k is a random number with a known distribution. Then, $\Xi_k \subseteq \mathbb{R}^d$ is a random finite set with cardinality $|\Xi_k| = n_k$, and $X_k = \{\mathbf{x}_k^1, \ldots, \mathbf{x}_k^{n_k}\}$ is called a realization of the RFS.*

It should be noted that the cardinality equal to zero is also valid, and in that case, the realization of an RFS is an empty set, i.e., $X_k = \emptyset$.

As we already know, RFSs are random variables, and we need an analogous mechanism as for classical random variables defined on vectors that allows us to describe the behavior of the RFS. For vector-based random variables, we have cumulative distribution functions and their first-order derivatives probability density function. The analogy to a cdf for a random finite set is called the *belief mass measure* and is defined as follows:

▶ **Definition 2.2** (Belief mass measure)**.** *Let $\Xi \subseteq \mathbb{R}^d$ be a random finite set, and $\mathcal{S} \subseteq \mathbb{R}^d$ be some region of the set space. The belief mass measure β_Ξ of Ξ is defined as:*

$$\beta_\Xi(\mathcal{S}) = \Pr\{\Xi \subseteq \mathcal{S}\} = \int_{\mathcal{S}} p_\Xi(X)\delta X, \tag{2.1}$$

where p_Ξ is a FISST density function, also called the multi-target pdf, and $\int_{\mathcal{S}} p_\Xi(X)\delta X$ is a set integral over all sets $X \subseteq \mathcal{S}$.

In this work, we do not include the formal proof that p_Ξ is indeed a pdf, i.e., $p_\Xi(X)$ for all X and $\int_{\mathbb{R}^d} p_\Xi(X)\delta X = 1$. However, the reader may refer to the standard reference on FISST [19] to see the proofs. Here, we only emphasize that this function is a pdf for RFSs and it captures both the cardinality of a set and the distribution over elements in the set.

2.2.2 Set integral and the convolution formula

In the definition above, we have used an integral on sets. For a set-valued function f, it is defined as:

$$\int_{\mathcal{S}} f(X)\delta X = \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\mathcal{S}\times\ldots\times\mathcal{S}} f\left(\{\mathbf{x}^1,\ldots,\mathbf{x}^n\}\right) \mathrm{d}\mathbf{x}^1 \ldots \mathrm{d}\mathbf{x}^n. \tag{2.2}$$

We can build the intuition for the set integral by understanding what integrated values really are. When we compute an integral over real values, we, in simple words, compute the value of the integrand for all possible values. The set integral is conceptually the same, however, the integrated value is a random set and there are two variables incorporated–the cardinality of the set, and the values. That means, that we sum up the function values for all possible cardinalities, from the empty set until infinity, and all possible elements in the set. The term $\frac{1}{n!}$ refers to the fact that for any two values a, b, sets $\{a, b\}$ and $\{b, a\}$ are equal, because sets are invariant to order. That means that any permutation of elements in the set leads to the same set, and the number of possible permutations is equal to $n!$.

The set integral is important for computing the posterior distribution in Bayesian inference. The Chapman-Kolmogorov equation is defined the same way for RFSs, as we will see later in this section, and it uses the total probability theorem where we compute the integral over all states from the previous time step. In case when the state is represented by a random set, the set integral is used.

In addition to the set integral, we are often interested in basic operations on sets, like union or intersection. For instance, if we have two random finite sets with two different distributions, what is the distribution of their union? To answer this question and to build the intuition, we shall return to a simpler case, discrete-valued random variables.

If we have two integer-valued random variables X, Y with pmfs $p_X(x)$ and $p_Y(y)$, respectively, and their sum $Z = X + Y$, the probability that a realization of Z is equal to the exact value z is described using the so-called convolution formula:

$$\Pr\{Z = z\} = \sum_{v=-\infty}^{+\infty} p_X(v)p_Y(z - v). \tag{2.3}$$

The main idea behind the convolution is simple: we have an infinite number of possibilities how we can sum up two values and get the result equal to z. If we take a value v, then the second value is $z - v$, for all v. And the result is obtained by summing the probabilities of this pair of numbers.

The convolution over sets has the same idea, where we sum over all possible pairs of values that add up to a particular value. If Ξ and Γ are two random finite sets with multi-object probability density functions $p_\Xi(X)$ and $p_\Xi(Y)$, and $\Sigma = \Xi \cup \Gamma$, the probability that a realization of Σ equals to the exact set X is:

$$p_\Sigma(Y) = \sum_{X \subseteq Y} p_\Xi(X)p_\Gamma(Y \setminus X), \tag{2.4}$$

Simply speaking, we take some values (or none at all) from X and create a set Y containing these values, and the rest are assigned to the other set, $Y \setminus X$. Then, the probability of this assignment is evaluated and we take the sum of probabilities of all such assignments. Having understood the main concept, we are ready to define the set convolution for general cases.

▶ **Definition 2.3** (Set convolution)**.** *Let Ξ be a random finite set and $\Xi_1 \cup \ldots \cup \Xi_n = \Xi$ be statistically independent subsets with multi-target probability density functions $p_{\Xi_1}(X), \ldots, p_{\Xi_n}(X)$, respectively. The multi-target pdf for Ξ is then defined as:*

$$p_\Xi(Y) = \sum_{X_1 \uplus \ldots \uplus X_n = Y} \prod_{i=1}^{n} p_{\Xi_i}(X_i), \tag{2.5}$$

where the expression $X_1 \uplus \ldots \uplus X_n = Y$ represents the union of mutually disjoint (and possibly empty) subsets of Y such that $X_1 \cup \ldots \cup X_n = Y$ [19, pp. 385–386].

The convolution formula is a fundamental concept in FISST and multi-target tracking. In many filters, like, for instance, the Poisson Multi-Bernoulli Mixture filter [7], we model existing targets as a Bernoulli RFS, and the clutter as the Poisson RFS. Measurements at every time step contain a random permutation of true measurements and clutter, and in order to calculate the posterior density we model them as a union of two RFSs. The PHD filter, the filter covered in detail later in this work, also uses unions of several RFSs.

Last but not least, we should define the expected value of a RFS pdf and the cardinality distribution. The expected value is defined in almost the same way like for vector- or scalar-valued pdfs, however, there is one important difference, as the sum of RFSs is not defined, the average is not defined too, and we cannot calculate the expected value directly. However, if we define a function that maps a RFS into a space where the addition is defined, we can compute the expected value using this formula:

▶ **Definition 2.4** (Expected value of a multi-target pdf)**.** *Let Ξ be a RFS and $p_\Xi(X)$ be its multi-target pdf. Let $f : \mathcal{F}(D) \to \mathbb{R}$ be some function that maps a RFS into a real number*[1]*. Then, the expected value is defined as follows:*

$$E[f(\Xi)] = \int f(X) p_\Xi(X) \delta X = \sum_{n=0}^{\infty} \frac{1}{n!} \int f(\{\mathbf{x}^1, \ldots, \mathbf{x}^n\} p_\Xi(\{\mathbf{x}^1, \ldots, \mathbf{x}^n\} \mathrm{d}\mathbf{x}^1 \ldots \mathrm{d}\mathbf{x}^n. \tag{2.6}$$

The expected value of a multi-target pdf is used when calculating the Chapman-Kolmogorov equation for RFSs describe in Section 2.2.4.

The cardinality distribution of a random finite set is a discrete distribution that has the probability mass function describing the probability of the set having the exact number of elements.

▶ **Definition 2.5** (Cardinality distribution of a RFS)**.** *Let Ξ be a RFS and $p_\Xi(X)$ be its multi-target pdf and let the Kronecker delta function be denoted δ_y defined by:*

$$\delta_y = \begin{cases} 1 & \text{if } y = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

Then, the cardinality pmf is defined as:

$$\Pr\{|\Xi| = k\} = E[\delta_{k-|\Xi|}] \tag{2.8}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \int \delta_{k-n} p_\Xi(\{\mathbf{x}^1, \ldots, \mathbf{x}^n\} \mathrm{d}\mathbf{x}^1 \ldots \mathrm{d}\mathbf{x}^n \tag{2.9}$$

$$= \frac{1}{k!} \int p_\Xi(\{\mathbf{x}^1, \ldots, \mathbf{x}^k\} \mathrm{d}\mathbf{x}^1 \ldots \mathrm{d}\mathbf{x}^k. \tag{2.10}$$

Before we proceed to the use of random finite sets in Bayesian inference and multi-target tracking, we should describe several known RFSs and their pdfs. In the next sections, we will describe set distributions that are already known to us: the Bernoulli RFS and the Poisson RFS. The latter has a direct relation to the important mathematical concept used in target-tracking, the point processes.

[1] $\mathcal{F}(D)$ is a special function that creates a so-called power set, or all possible subsets of the set D.

2.2.3 Bernoulli and Poisson RFS

The Bernoulli random finite sets is one of the simplest and straightforward RFS pdfs. Recall that the Bernoulli distribution is a discrete distribution that models the probability of the positive outcome. In multi-target tracking, Bernoulli RFSs are used to model the existence of one object. That is, the cardinality distribution takes values either one or zero (either an object exists or not) with some probability r. The state distribution of the object can be arbitrary. Formally, the Bernoulli RFS is defined as follows:

▶ **Definition 2.6** (Bernoulli RFS)**.** *Let* $\mathbf{x}$ *be a random variable with the pdf* $p(\mathbf{x})$*, and let* $0 \leq r \leq 1$ *be some number. Then, the Bernoulli RFS* Ξ *has the following pdf:*

$$p_{\Xi}(X) = \begin{cases} 1 - r & \text{if } X = \emptyset, \\ rp(\mathbf{x}) & \text{if } X = \{\mathbf{x}\}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

The cardinality of the Bernoulli RFS has the Bernoulli distribution, that is:

$$\Pr\{|\Xi| = k\} = \begin{cases} 1 - r & \text{if } k = 0, \\ r & \text{if } k = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

The Poisson RFS is also a widely used multi-target pdf. It is commonly used to model clutter, since it has a parameter λ that represents the intensity, or the expected number of noise measurements.

▶ **Definition 2.7** (Poisson RFS)**.** *Let* $\mathbf{x}$ *be a random variable with the pdf* $p(\mathbf{x})$*, and let* $\lambda(x)$ *be the intensity function*[2]*. Then, the multi-target pdf of the Poisson RFS* Ξ *is:*

$$p_{\Xi}(X) = \exp\left(-\int \lambda(\mathbf{x})\mathrm{d}\mathbf{x}\right) \prod_{\mathbf{x}\in\Xi} \lambda(\mathbf{x}). \tag{2.13}$$

The cardinality pmf of the Poisson RFS is Poisson-distributed with the rate parameter $\hat{\lambda} = \int \lambda(\mathbf{x})\mathrm{d}\mathbf{x}$ *is the following:*

$$\Pr\{|\Xi| = k\} = \text{Poisson}(k; \hat{\lambda}). \tag{2.14}$$

Intuitively, the Poisson RFS pdf represents the probability of observing any finite set of objects anywhere in the area, given that objects appear over it independently. In contrast to the Bernoulli distribution, the Poisson RFS cannot be used to model the target existence, since there is no way to limit the cardinality. However, when the location of the points is random through the whole surveillance area, and the number of these points is Poisson-distributed, the Poisson RFS is a good choice.

Both the Bernoulli RFS and the Poisson RFS are examples of so-called point processes, also called the Bernoulli process and the Poisson Point Process (PPP), respectively. Generally, a point process is a mathematical model that describes the random spatial distribution of points. In target tracking, point processes model various events, such as object appearance, disappearance, or movement. In particular, the PDF filter uses the PPP to model clutter, where clutter measurements are independent and appear uniformly over the whole surveillance area. For more information about point processes, the reader is referred to the classical textbook on the topic [31].

[2]Note that the intensity function can be an arbitrary function depending on the domain of the Poisson RFS.

2.2.4 Bayes filter in terms of RFSs

In Section 1.3.3, we have discussed the prediction step, also known as the Chapman-Kolmogorov equation, and the update step in term of state vectors and motion and measurement models. In this section, we will derive similar formulas for Bayesian inference in the context of random finite sets.

Since RFS pdfs are indeed densities, as we discussed in Section 2.2.1, the recursive relations for RFS have the same form, however, densities are multi-target densities that catch the motion of all objects at once, and arguments are sets.

▶ **Theorem 2.8** (The prediction step for RFS)**.** *Given sets of measurements* $Z_{1:k-1} = \{Z_1, Z_2, \ldots, Z_{k-1}\}$*, the current state RFS* X_{k-1}*, the multi-target transition density* $f_{k|k-1}(X_k|X_{k-1})$*, and the multi-target posterior density from the previous time step* $p_{k-1}(X_{k-1}|Z_{1:k-1})$*, the prediction step of the Bayes filter for RFSs is computed as follows:*

$$p_{k|k-1}(X_k|Z_{1:k-1}) = \int f_{k|k-1}(X_k|X_{k-1})\, p_{k-1}(X_{k-1}|Z_{1:k-1})\, \delta X_{k-1}. \tag{2.15}$$

▶ **Theorem 2.9** (The update step for RFS)**.** *Given the predicted density* $p_{k|k-1}(X_k|Z_{1:k-1})$*, the observed set of measurements* Z_k *and the multi-target likelihood* $g_k(Z_k|X_k)$*, the update step of the Bayes filter for RFS is computed as follows:*

$$p_k(X_k|Z_{1:k}) = \frac{g_k(Z_k|X_k)\, p_{k|k-1}(X_k|Z_{1:k-1})}{\int g_k(Z_k|X)\, p_{k|k-1}(X|Z_{1:k-1})\, \delta X} \tag{2.16}$$

$$\propto g_k(Z_k|X_k)\, p_{k|k-1}(X_k|Z_{1:k-1}). \tag{2.17}$$

Note that the multi-target transition density is not the same motion model that we used before. The same applies for the multi-target likelihood that plays the role of the measurement model but is a completely different relation. The transition density captures the motion of all objects in a set all at once, including the birth (appearance) of new objects, and the death (disappearance). The likelihood, on the other hand, models the transition from a set of object states to probable measurements. Arguments of both are now sets, and explicit forms can be derived from motion and measurement models using FISST. These derivations along with formal proofs of the Bayesian recursion defined above can be found in [30].

Even though the relations look similarly on the first glance, the underlying mathematics differs drastically. In comparison to the single-object tracking, states now represent the distribution of all objects simultaneously. Moreover, the multi-target transition density now captures much more information: not only it describes the transition of objects that are already in the current state set, but also the behavior of the change of the cardinality of the set. In the next section, we will explore how these relations can be applied to address the multi-target tracking problem. We will also examine the assumptions that must be made in order to develop Bayes filters based on FISST.

2.2.5 Multi-target tracking standard model

We have now reached the final fundamental theoretical section where we will give a formal definition to the multi-target tracking problem. We have already mentioned that the multi-target transition density includes more than a standard motion model; more specifically, it describes not only the dynamics of all underlying objects in the current state set but also the birth and death of objects. In this section, we will cover in detail how new objects are born, the survival probability, and how we can model clutter. These concepts are fundamental for building the PHD filter.

Let $\mathcal{X}$ be the state space and $\mathcal{Z}$ be the measurement space. Elements of $\mathcal{X}$ are state vectors of one object, and $\mathcal{Z}$ contains elements that are measurement vectors. For instance, for the CV model covered in Section 1.3.2.1, the state space is $\mathbb{R}^4$, and the measurement space is $\mathbb{R}^2$. Assume that at time k, the number of targets is $M(k)$. This number varies at every time step since new targets may appear or disappear. The *multi-target state set* at time k is therefore $X_k = \{\mathbf{x}_k^1, \ldots, \mathbf{x}_k^{M(k)}\} \in \mathcal{F}(\mathcal{X})$. Also, at every time step, we get a measurement set. Let us denote the number of measurements at time k as $N(k)$. Thus, the *multi-target observation set* at time k is $Z_k = \{\mathbf{z}_k^1, \ldots, \mathbf{z}_k^{M(k)}\} \in \mathcal{F}(\mathcal{Z})$. The order of elements in both sets is not significant. The measurement set contains not only those measurements that come from real objects but also clutter, and those noise vectors are indistinguishable from those coming from targets. The goal of the multi-target tracker is to filter out the clutter and to find the best mapping between the objects and measurements.

Existing targets either move according to their underlying dynamics model to the next state or die (disappear). The transition from the state $\mathbf{x}_{k-1} \in X_{k-1}$ to $\mathbf{x}_k \in X_k$ happens according to the specified motion model of the target, $f_{k|k-1}(\mathbf{x}_k|\mathbf{x}_{k-1})$. The death of one target happens with the probability $1 - P_{S,k}(\mathbf{x}_{k-1})$, and the target survives with the probability $P_{S,k}(\mathbf{x}_{k-1})$. This probability is referred to as the *survival probability*. In terms of RFSs, the survival of a target with the state $\mathbf{x}_{k-1}$ is modeled using the survival function on an RFS $S_{k|k-1}$:

$$S_{k|k-1}(\mathbf{x}_{k-1}) = \begin{cases} \{\mathbf{x}_k\} & \text{if the target survives,} \\ \emptyset & \text{otherwise.} \end{cases} \tag{2.18}$$

At every time step, new targets may appear (be born). The birth process is considered spontaneous, and the exact distribution of newborn targets is domain-dependent. In general, the target birth distribution includes the location where new targets appear and their intensity. For instance, the birth process may be modeled by the Poisson Point Process, as in [7], or the Gaussian mixture for Gaussian-linear cases, as we will later see. In terms of FISST, the birth of new targets is a random set $B_{k|k-1}(\cdot)$ that contains newly born targets when the transition from the multi-target state X_{k-1} to the state X_k occurs.

Given that we have the relations for both the transition from the current multi-target state and the birth set, and assuming that the multi-target state at the previous time step $k-1$ is X_{k-1}, we can derive the following expression for the multi-target state X_k, assuming that the survival and birth processes are independent of each other[3]:

If we apply the convolution formula from Definition 2.3, and introduce the simplified notation $S_{k|k-1}^i = S_{k|k-1}(\mathbf{x}_{k-1}^i)$, the complete multi-target motion model $f_{k|k-1}(X_k|X_{k-1})$ has the following form:

$$f_{k|k-1}(X_k|X_{k-1}) = \sum_{\Gamma_k \uplus S_{k|k-1}^1 \ldots \uplus S_{k|k-1}^{M(k-1)} = X_{k-1}} \gamma_k(\Gamma_k) \prod_{i=1}^{M(k-1)} s_{k|k-1}(S_{k|k-1}^i|\mathbf{x}_{k-1}^i), \tag{2.19}$$

where $\gamma_k(\cdot)$ is the intensity function of the birth process, which can be a Poisson RFS pdf defined in Equation 2.13, and $s_{k|k-1}(S_{k|k-1}^i|\mathbf{x}_{k-1}^i)$ is the multi-target transition density defined as:

$$s_{k|k-1}(S_{k|k-1}^i|\mathbf{x}_{k-1}^i) = \begin{cases} P_{S,k}(\mathbf{x}_{k-1}^i) f_{k|k-1}(\mathbf{x}_k|\mathbf{x}_{k-1}^i) & \text{if } S_{k|k-1}^i = \{\mathbf{x}_k\}, \\ 1 - P_{S,k}(\mathbf{x}_{k-1}^i) & \text{if } S_{k|k-1}^i = \emptyset. \end{cases} \tag{2.20}$$

The multi-target measurement model, or the multi-target likelihood, also consists of several parts. At time k, a target may generate a new measurement with probability $P_{D,k}(\mathbf{x}_k)$, and there

[3]Note that, in addition to the survival set and the birth process, [32] assumes the existence of the spawning process $\bigcup_{\zeta \in X_{k-1}} B_{k|k-1}(\zeta)$, i.e., that the existing targets may randomly spawn new targets in the same location. While this is important in the defense area, such as when a fighter takes off from an aircraft carrier, we do not include this process in this work. Fundamentally, the underlying mathematics does not change.

will be a misdetection with probability $1 - P_{D,k}(\mathbf{x}_k)$. A measurement is generated according to the measurement model, $g_k(\mathbf{z}_k|x_k)$. Thus, the measurement RFS at time k is:

$$\Theta_k(\mathbf{x}_k) = \begin{cases} \{\mathbf{z}_k\} & \text{if target is detected,} \\ \emptyset & \text{otherwise.} \end{cases} \tag{2.21}$$

Additionally, at every time step, false measurements are generated by a sensor. The RFS with clutter is denoted as K_k. Both $\Theta_k(\mathbf{x}_k)$ and K_k create the multi-target measurement set Z_k:

$$Z_k = K_k \cup \left[\bigcup_{\mathbf{x}_k \in X_k} \Theta_k(\mathbf{x}_k) \right]. \tag{2.22}$$

As we did for the multi-target transition density, we denote $\Theta_k^i = \Theta_k(\mathbf{x}_k^i)$ and apply the set convolution formula to get the multi-target measurement likelihood:

$$p_k(Z_k|X_k) = \sum_{K_k \uplus \Theta_k^1 \ldots \uplus \Theta_k^{M(k)} = Z_k} \kappa_k(K_k) \prod_{i=1}^{M(k)} \theta_k(\Theta_k^i|\mathbf{x}_k^i), \tag{2.23}$$

where $\kappa_k(K_k)$ is the clutter intensity function, which is often modeled as a Poisson RFS, and $\theta_k(\Theta_k^i|\mathbf{x}_k^i)$ is the multi-target measurement density defined as:

$$\theta_k(\Theta_k^i|\mathbf{x}_k^i) = \begin{cases} P_{D,k}(\mathbf{x}_k^i) g_k(\mathbf{z}_k|\mathbf{x}_k^i) & \text{if } \Theta_k^i = \{\mathbf{z}_k\}, \\ 1 - P_{D,k}(\mathbf{x}_k^i) & \text{if } \Theta_k^i = \emptyset. \end{cases} \tag{2.24}$$

Let us examine Equation 2.23 closely to gain an understanding of its implications. The convolution formula suggests that we divide the measurement set Z_k into several subsets, where each subset can either be empty or contain measurements. Next, we compute the value of the multi-target measurement density function for every combination of state vectors and values in the set. In other words, we determine the likelihood of the association between the state vector and the measurement. Consequently, we can interpret the computation of the multi-target measurement likelihood as a joint formulation for determining the probability of all conceivable association hypotheses for all targets. While this may appear similar to the JPDA filter, in this case, the relationship is more general and supports an unknown number of targets.

In general, multi-target trackers are built on top of several general assumptions about the problem. These assumptions form the Standard Model of Multi-target tracking [19, pp. 311–313]. We will enumerate these assumptions, which summarize this section, in the following list:

1. The clutter process is Poisson-distributed in time and uniformly distributed in space.
2. The clutter process, target motions, and observations are statistically independent.
3. The transition density and the measurement likelihood are Markovian, that is once $\mathbf{x}_{k-1}$ is known, then $\mathbf{x}_k$ is independent of $\mathbf{x}_{k-2}$, $\mathbf{x}_{k-3}$, etc.
4. Existing targets survive from time $k-1$ to k with the probability $P_{S,k}(\cdot)$ and move to the next state with the motion model $f_k(\mathbf{x}_k|\mathbf{x}_{k-1})$.
5. A target generates a measurement with probability $P_{D,k}(\cdot)$ with the measurement model $g(\mathbf{z}_k|\mathbf{x}_k)$.
6. A target generates no more than one measurement at a time.
7. The birth process is Poisson-distributed in time.

The PHD filter also relies on these main assumptions. In the following sections, we present the PHD filter, which is based on the propagation of the first-order multi-target moment, or the PHD function, in time instead of the full posterior distribution. Next, we will derive relations for the Gaussian-linear case to get equations of the GM-PHD filter. Finally, we present the way how we can input additional information into the filter to get better tracking results.

2.3 The PHD filter

The Probability Hypothesis Density (PHD) filter is a popular approach to multi-target tracking, especially when the number of targets is unknown and varies over time. The general formulation of the filter was first introduced by Mahler [30] as a direct application of the probability hypothesis density function. The filter is based on finite set statistics (FISST), covered briefly in Section 2.2, and works by propagating the first-order multi-target moment, or the PHD function, in time instead of the full posterior distribution. This greatly reduces the computational burden of the filter compared to traditional hypotheses-based methods such as the MHT filter.

In 2003, Vo proposed the sequential Monte Carlo implementation of the PHD filter [33], known as the SMC-PHD filter. However, this implementation had several drawbacks, such as a high computational cost due to the large number of particles required for distribution approximation and the reliance on clustering techniques for state estimation. Later, in 2006, Vo and Ma introduced a closed-form solution for the Gaussian-linear case [32], called the GM-PHD filter. For non-linear models, the authors suggested implementations that utilize non-linear Kalman filters, such as the Unscented Kalman Filter and the Extended Kalman Filter. The versions of the PHD filter with these linearization techniques were called UK-PHD and EK-PHD, respectively. The GM-PHD filter was a significant improvement over the SMC-PHD filter in terms of computational efficiency while still providing accurate estimates of the number of targets and their states. The GM-PHD filter is the main topic of this work.

2.3.1 The PHD function

The probability hypothesis density (PHD) function is the first-order moment of a random finite set, also known as the *intensity function*. Its value is the expected number of targets in a certain area of the target space. From Section 1.1.1, we know that the first-order moment is the expectation value, but what is the expectation value of a set? To answer this question, let us define the problem formally. The computations that follow are based on the original book by Mahler [19, pp. 576–578].

Let a random finite set be denoted by Ξ with the multi-target probability distribution $p_\Xi(X)$, and the target space $\mathcal{X}$. The expected number of targets is, conceptually, the integral over all targets multiplied by the value of the probability density function for each state value. However, since summing over targets will not have any meaning in terms of the expected value, we introduce the delta function of a set:

$$\delta_\Xi(\mathbf{x}) = \begin{cases} 0 & \text{if } X = \emptyset, \\ \sum_{\mathbf{w}\in X} \delta_{\mathbf{w}}(\mathbf{x}) & \text{otherwise,} \end{cases} \tag{2.25}$$

where $\delta_{\mathbf{w}}$ is the Dirac delta density centered at $\mathbf{w}$, the continuous-space analog of the Kronecker delta function mentioned in Definition 2.5.

▶ **Definition 2.10** (PHD function)**.** *Given a random finite set Ξ, where elements of Ξ are from $\mathcal{X}$, with the multi-target density $p_\Xi(X)$, the expected value of the set Ξ is defined as:*

$$v_\Xi(\mathbf{x}) = E[\delta_\Xi(\mathbf{x})] = \int \delta_\Xi(\mathbf{x}) \cdot p_\Xi(X)\delta X. \tag{2.26}$$

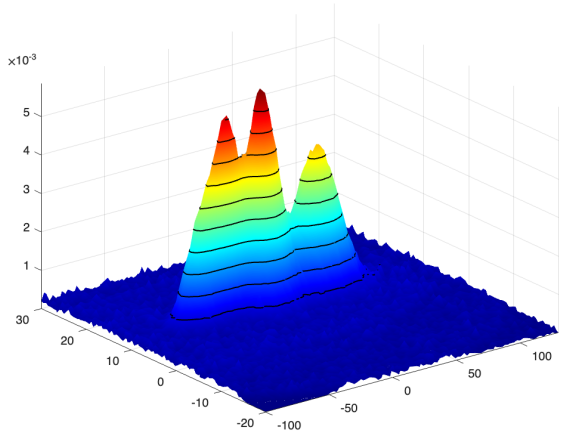


Figure 2.2 An example of the PHD function. Three targets move in the 2D Euclidean space according to a CV model. Note that this function is not normalized, which means that the integral over the entire space is not equal to one. In this example, the expected number of targets in the space is approximately 3.2245, which can be inferred from the location of the corresponding peaks.

The function $v_{\Xi}(\mathbf{x})$ is also called the Probability hypothesis density (PHD) function.

The PHD function is not a probability density function, that is, it is not normalized to integrate to one. The integral of the PHD function over some subspace S yields the expected number of targets in this subspace:

$$E[|\Xi \cap S|] = \int_S v_{\Xi}(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{2.27}$$

This formulation prompts that the PDF function is an unnormalized multi-target probability density function with the normalization constant equal to the expected number of existing targets $\hat{N}$ over the whole space $\mathcal{X}$, i.e., $\hat{N} = \int_{\mathcal{X}} v_{\Xi}(\mathbf{x})\mathrm{d}\mathbf{x}$. Therefore, the peaks of this distribution are the most probable locations of the tracked targets. The density evolves over time as targets move in space. An example of the PHD function is illustrated in Figure 2.2.

The fact that the PHD function does not include normalization allows an efficient computation of this function in terms of the computational complexity. This fundamental property of the PHD function led to the development of the PHD filter. In the next section, we will define several assumptions of the PHD filter and define the predict-update cycle.

2.3.2 PHD filter formal definition

We have defined the PHD function and mentioned that instead of propagating the full posterior density, the PHD filter propagates the first-order moment of the multi-target posterior state. In this section, we will describe the main assumptions of the PHD filter and derive the predict and update step equations in terms of the PHD function. It should be noted that while the PHD filter is a general concept, its implementation is specific to the problem domain. Later sections will provide the exact closed-form solution for Gaussian-linear problems.

The PHD filter relies on several general assumptions about the modeled problem. In addition to the standard assumptions, the PHD filter assumes:

1. Each target has a single-target motion model $f_k(\mathbf{x}_k|\mathbf{x}_{k-1})$.

2. Existing targets move to the next state with probability $P_{S,k}(\mathbf{x}_{k-1})$.

3. New targets are born with states from X according to the multi-target birth likelihood $\gamma_k(X)$. The corresponding PHD of this likelihood is:

$$\gamma_k(\mathbf{x}) = \int \gamma_k(\mathbf{x} \cup X)\delta X. \tag{2.28}$$

4. Existing targets generate measurements according to the single-target Markovian measurement model $g(\mathbf{z}_k|\mathbf{x}_k)$ with probability $P_{D,k}(\mathbf{x}_k)$.

5. Clutter measurements are Poisson-distributed in time with the spatial distribution $\kappa_k(\mathbf{z}_k)$.

6. The predicted multi-target distribution $p_{k|k-1}(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ is a Poisson RFS.

The last assumption is a simplification that allows to derive a closed form solution. A Poisson process assume that all targets are statistically independent, and such approximation is not that restrictive is the interactions between moving targets in negligible. In this work we do not include the spawning process, therefore, it can be shown that this requirement is satisfied [32].

Both the prediction and the update follow the same rules we have derived in Section 2.2.4. It can be shown that the Chapman-Kolmogorov equation and the measurement correction step for the PHD filter, excluding target spawning, has the following form:

▶ **Theorem 2.11** (The PHD recursion)**.** *Assuming that all requirements defined above hold, the PHD recursion has the following form:*

$$v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1}) = \gamma_k(\mathbf{x}_k) + \int P_{S,k}(\mathbf{x}_{k-1}) f_{k|k-1}(\mathbf{x}_k|\mathbf{x}_{k-1}) v_{k-1}(\mathbf{x}_{k-1}|Z_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1}, \tag{2.29}$$

$$\begin{aligned} v_k(\mathbf{x}_k|Z_{1:k}) = {} & [1 - P_{D,k}(\mathbf{x}_k)]\, v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1}) \\ & + \sum_{\mathbf{z}_k \in Z_k} \frac{P_{D,k}(\mathbf{x}_k) g_k(\mathbf{z}_k|\mathbf{x}_k) v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1})}{\kappa_k(\mathbf{z}_k) + \int P_{D,k}(\boldsymbol{\xi}) g_k(\mathbf{z}_k|\boldsymbol{\xi}) v_{k|k-1}(\boldsymbol{\xi}|Z_{1:k-1}) \mathrm{d}\boldsymbol{\xi}}, \end{aligned} \tag{2.30}$$

where $v_{k-1}(\mathbf{x}_{k-1}|Z_{1:k-1})$ is the posterior PHD function from the previous time step [32] [19, pp. 588–591].

Note that the computation of the posterior probability hypothesis density after the update does not include the set integral, which means that we do not need to compute all association hypotheses between measurements and states, decreasing the computational burden. It can be shown that the computational complexity of the PHD filter at every time step is $\mathcal{O}(mn)$, where m is the number of measurements and n is the number of current targets [19, p. 592].

In the next section, we will provide the formulation of the PHD filter in terms of Gaussian-linear filtering and Gaussian mixtures. We will see that for such cases a closed-form solution exists, and there are no approximations required to compute the posterior PHD density.

2.3.3 The Gaussian Mixture PHD filter

The Gaussian Mixture Probability Hypothesis Density (GM-PHD) filter was proposed in 2006 by Vo and Ma in [32] as a closed-form solution for the Gaussian-linear case of the general PHD filter and quickly became popular. As mentioned in the previous section, the PHD filter does not propagate the full posterior distribution, which eliminates the need to compute hypothesis likelihoods between measurements and existing targets. Therefore, the resulting filter is not computationally expensive as hypotheses-oriented filters like the MHT filter or filters that are

built on top of the FISST theory and utilize random finite sets and the convolution formula to compute the full posterior distribution at each time step.

In the GM-PHD filter, the posterior intensity function is described by a Gaussian mixture. The same applies to the initial prior distribution and birth components. The clutter is modeled using a Poisson Point Process. Both the prediction and update steps utilize the equations that were inferred in the previous section, and the target dynamics and the single-target measurement likelihood are modeled using the standard Kalman filter equations. In this section, we will give a formal definition of the GM-PHD filter along with the exact algorithm. All relations described in this section follow those provided in [32] with the addition of labels (or tags) for Gaussian components from [34] to enable basic track management. It should be noted that the notation in this work is slightly modified to stay consistent with the notation given in previous sections. Moreover, as noted before, we assume that existing targets cannot spawn additional targets, so the independence of targets' dynamics is preserved.

2.3.3.1 Track maintenance

In this section, we will briefly cover one of challenges in multi-target tracking, which is track maintenance and then we introduce the way how this challenge may be addressed in the GM-PHD filter. When an algorithm creates estimates of targets at each time step, these estimates cannot always be implicitly connected to those from previous time steps. If we require not only point estimates but also the trajectories of targets, we should use techniques to maintain identities of targets and to create associations between new measurements and these identities. Thus, the trajectory here refers to the collection of state estimates of a target in time. Generally, the track maintenance problem includes three disjoint sub-problems: track initialization, track confirmation and track deletion [35]. Track initialization refers to the creation of new tracks. A new track is initialized when a new measurement is received that cannot be assigned to any existing track. Since the measurement is not always the true measurement, the tentative track should be confirmed by receiving subsequent measurements that meet certain criteria. Different algorithm specify various different criteria depending on their internal logic. For instance, one way to confirm tracks is called the M/N rule, which states that the track becomes confirmed if at least M measurements out of N subsequent updates should be assigned to the tentative track [35, p. 871]. Confirmed tracks create trajectories of objects and we need a way to terminate tracks that do not receive measurements anymore, for example, when a target leaves the field of view. Yet again, there are multiple different ways how to approach this problem, including restricting the maximum age of a track, or when the likelihood of target existence falls under some threshold [29, p. 243].

Estimating tracks has several challenges that need to be addressed to implement an effective tracking algorithm. For example, the track loss occurs when a track is temporarily or permanently lost due to a missed detection or incorrect association between a measurement and some existing track. Wrong associations may also cause track swaps, when an algorithm swaps measurements from two or more targets moving closely to each other, and measurements generated by one target are assigned to a track of another, and vice versa. Closely moving targets may also create a so-called track coalescence, the situation when two tracks of different targets are merged into one track in the middle of trajectories of these two targets. For instance, the JPDA filter is known to suffer from track coalescence [36].

Despite there are many advanced techniques how to maintain track, we choose a straightforward approach in this work for the implementation of the GM-PHD filter. Every Gaussian component receives a unique tag, or label, that identifies this component. When the algorithm generates state estimates, these tags are used to connect single-target estimates from different time steps and having the same label into one trajectory. The track is terminated when the Gaussian component with the tag assigned to this track is removed from the mixture. While this approach is intuitive and simple, in Chapter 4 we show that is shows good performance results.

2.3.3.2 GM-PHD recursion

Before we start to deduce all formulas of the GM-PHD filter, we should make several additional assumptions for the Gaussian-linear case in addition to the general ones given in Section 2.3.2:

1. All targets have a Gaussian-linear single-target motion and measurement models, that is:

$$f_{k|k-1}(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}\left(\mathbf{x}_k; \mathbf{F}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}\right), \tag{2.31}$$
$$g_k(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}\left(\mathbf{z}_k; \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k\right), \tag{2.32}$$

   where the detailed explanation of the meaning of $\mathbf{F}_{k-1}, \mathbf{Q}_{k-1}, \mathbf{H}_k, \mathbf{R}_k$ are given in Section 1.3.2.

2. Both the survival and detection probabilities are constant in time[4], that is, they do not depend on states:

$$P_{S,k}(\mathbf{x}_{k-1}) = P_{S,k}, \tag{2.33}$$
$$P_{D,k}(\mathbf{x}_k) = P_{D,k}. \tag{2.34}$$

3. The birth intensity is modeled using a Gaussian mixture of the form:

$$\gamma_k(\mathbf{x}_{k-1}) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}\left(\mathbf{x}_{k-1}; \mathbf{m}_{\gamma,k}^{(i)}, \mathbf{P}_{\gamma,k}^{(i)}\right), \tag{2.35}$$

   where $J_{\gamma,k}$ denotes the number of birth components, and $\mathbf{m}_{\gamma,k}^{(i)}$ and $\mathbf{P}_{\gamma,k}^{(i)}$ are the mean vector and the covariance matrix of each component. These are given as parameters and specified in advance according to the problem domain. Since this mixture is not a probability density by an intensity function, weights do not need to sum to one, and the weights $w_{\gamma,k}^{(i)}$ here represent the expected number of new targets that are expected to appear in a location centered at $\mathbf{m}_{\gamma,k}^{(i)}$.

These assumptions form the basis for the Gaussian-linear PHD filter recursion formulas. Assuming these assumptions hold for a given problem, and by following the PHD filter recursion equations from Theorem 2.11, we propose the following theorem:

▶ **Theorem 2.12** (The GM-PHD filter prediction)**.** *Given that the posterior intensity of the previous time step $v_{k-1}(\mathbf{x}_{k-1}|Z_{1:k-1})$ is a Gaussian mixture of the form:*

$$v_{k-1}(\mathbf{x}_{k-1}|Z_{1:k-1}) = \sum_{i=1}^{J_{k-1}} \mathcal{N}\left(\mathbf{x}_{k-1}; \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)}\right), \tag{2.36}$$

where every Gaussian component has a unique tag or label assigned to it, which forms the set of tags:

$$T_{k-1} = \{T_{k-1}^{(1)}, \ldots, T_{k-1}^{(J_{k-1})}\}. \tag{2.37}$$

Then the predicted intensity at time k is also a Gaussian mixture, defined as:

$$v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1}) = \gamma_k(\mathbf{x}_k) + v_{S,k|k-1}(\mathbf{x}_k), \tag{2.38}$$

[4] A closed-form solution can be derived without this assumption, see Section III-E in [32].

where $\gamma_k(\mathbf{x}_k)$ is given in Equation 2.35 and $v_{S,k|k-1}(\mathbf{x}_k)$ is the PHD function of survived targets given by:

$$v_{S,k|k-1}(\mathbf{x}_k) = P_{S,k} \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}\left(\mathbf{x}_k; \mathbf{m}_{S,k|k-1}^{(i)}, \mathbf{P}_{S,k|k-1}^{(i)}\right), \tag{2.39}$$

$$\mathbf{m}_{S,k|k-1}^{(i)} = \mathbf{F}_{k-1}\mathbf{m}_{k-1}^{(i)}, \tag{2.40}$$

$$\mathbf{P}_{S,k|k-1}^{(i)} = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^{(i)}\mathbf{F}_{k-1}^{\intercal} + \mathbf{Q}_{k-1}. \tag{2.41}$$

The set of tags after the prediction step forms the union of the set of tags from the previous time step with the set of unique tags of components spontaneously born at time k:

$$T_{k|k-1} = T_{k-1} \cup \{T_{\gamma_k}^{(1)}, \ldots, T_{\gamma_k}^{(J_{\gamma_k})}\}. \tag{2.42}$$

Let us look at these equations closer. The dynamics of every component (or a possible target) follows the standard Gaussian-linear dynamics and their mean vectors and covariance matrices are calculated as a standard Kalman predict. Weights represent the expected number of targets in the area centered at mean vectors, or the uncertainty of target existence. We then apply Theorem 1.14 and get rid of the integral in the prediction step from Theorem 2.11. Finally, the Gaussian mixture is then multiplied by the survival probability.

The initial prior intensity $v_0(\mathbf{x})$ with the corresponding set of unique tags T_0 is also a Gaussian mixture and those are set in advance. If the initial state is not known or empty, this mixture may have no components, i.e. $J_0 = 0$. Formally, the initial prior is defined by:

$$v_0(\mathbf{x}) = \sum_{i=1}^{J_0} w_0^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_0^{(i)}, \mathbf{P}_0^{(i)}\right), \tag{2.43}$$

$$T_0 = \{T_0^{(1)}, \ldots, T_0^{(J_0)}\}. \tag{2.44}$$

At time step k, we receive a measurement set Z_k with true and noise measurements (clutter). The update step of the GM-PHD filter is described in the following theorem:

▶ **Theorem 2.13** (The GM-PHD filter update)**.** *Given that the GM-PHD assumptions hold and the predicted intensity $v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1})$ and the corresponding set of tags $T_{k|k-1}$ have the form:*

$$v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1}) = \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}\left(\mathbf{x}_k; \mathbf{m}_{k|k-1}^{(i)}, \mathbf{P}_{k|k-1}^{(i)}\right), \tag{2.45}$$

$$T_{k|k-1} = \{T_{k|k-1}^{(1)}, \ldots, T_{k|k-1}^{(J_{k|k-1})}\}. \tag{2.46}$$

Then, the posterior intensity function $v_k(\mathbf{x}_k|Z_{1:k})$ is a Gaussian mixture of the form:

$$v_k(\mathbf{x}_k|Z_{1:k-1}) = (1 - P_{D,k})v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1}) + \sum_{\mathbf{z}_k \in Z_k} v_{D,k}(\mathbf{x}_k, \mathbf{z}_k), \tag{2.47}$$

where

$$v_{D,k}(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{J_{k|k-1}} w_k^{(i)}(\mathbf{z})\mathcal{N}\left(\mathbf{x}_k;\mathbf{m}_{k|k}^{(i)},\mathbf{P}_{k|k}^{(i)}\right), \tag{2.48}$$

$$w_k^{(i)}(\mathbf{z}) = \frac{P_{D,k}w_{k|k-1}^{(i)}q_k^{(i)}(\mathbf{z})}{\kappa_k(\mathbf{z}) + P_{D,k}\sum_{j=1}^{J_{k|k-1}} w_{k|k-1}^{(j)}q_k^{(j)}(\mathbf{z})}, \tag{2.49}$$

$$q_k^{(i)}(\mathbf{z}) = \mathcal{N}\left(\mathbf{z};\mathbf{H}_k\mathbf{m}_{k|k-1}^{(i)},\mathbf{H}_k\mathbf{P}_{k|k-1}^{(i)}\mathbf{H}_k^\intercal + \mathbf{R}_k\right), \tag{2.50}$$

$$\mathbf{m}_{k|k}^{(i)}(\mathbf{z}) = \mathbf{m}_{k|k-1}^{(i)} + \mathbf{K}_k^{(i)}(\mathbf{z} - \mathbf{H}_k\mathbf{m}_{k|k-1}^{(i)}), \tag{2.51}$$

$$\mathbf{P}_{k|k}^{(i)} = \left[\mathbf{I} - \mathbf{K}_k^{(i)}\mathbf{H}_k\right]\mathbf{P}_{k|k-1}^{(i)}, \tag{2.52}$$

$$\mathbf{K}_k^{(i)} = \mathbf{P}_{k|k-1}^{(i)}\mathbf{H}_k^\intercal(\mathbf{H}_k\mathbf{P}_{k|k-1}^{(i)}\mathbf{H}_k^\intercal + \mathbf{R}_k)^{-1}. \tag{2.53}$$

For each existing component, there are $(1+|Z_k|)$ *new components. Assign each this component the same tag as for the parent component to form a new set for each measurement* $T_{k-1}^{(\mathbf{z}_k^i)}$ *for* $i = 1,\ldots,|Z_k|$*. The final set of non-unique tags after the update step is thus:*

$$T_k = T_{k|k-1} \cup T_{k|k-1}^{(\mathbf{z}_k^1)} \cup \ldots \cup T_{k|k-1}^{(\mathbf{z}_k^{|Z_k|})}. \tag{2.54}$$

It can be proven that the equations above are the result of the direct application of the general PHD update formulas defined in Theorem 2.11 with the Gaussian-linear measurement model from Equation 1.48 and the Kalman update from Theorem 1.16. Again, the closed-form solution is obtained by utilizing the result from the fundamental Gaussian identity defined in Theorem 1.13. The tag set T_k after the update step contains $(1+|Z_k|)$ copies of the set $T_{k|k-1}$. The uniqueness of these tags is ensured in the next stages.

Let us examine what happens during the update step. The first part of Equation 2.47 computes for all existing targets the hypothesis of misdetection. Recall that the predicted intensity is a Gaussian mixture with $J_{k|k-1}$ components. That is, the number of misdetection hypotheses is also $J_{k|k-1}$. The second term creates $J_{k|k-1}$ new components for each measurement. These components represent the likelihood that this exact measurement comes from one of the existing targets represented by one of the predicted intensity components, and the weight of this exact assignment is computed in Equation 2.49. Equation 2.50 represents a Gaussian distribution centered at the location where we expect to see a new measurement from a target according to the measurement model. This Gaussian is evaluated at vector $\mathbf{z}$; $q(\mathbf{z})$ is a scalar value representing the likelihood that the measurement comes from this exact target. Equations 2.51 and 2.52 are the standard Kalman update equations, analogous to those defined in Theorem 1.16. The last equation, 2.53, represents the Kalman gain in its standard form. For better computational stability, this form can be reformulated as the Joseph form, defined in Equation 1.91. The summarized version of the measurement update algorithm of the GM-PHD filter is described in Algorithm 2.

It should be noted that new components generated in the update step are stored in three arrays with $|Z_k| \times (1+J_{k|k-1})$ elements for all new hypotheses. One of these arrays contains the weights, the other has mean vectors as elements, and the third is for covariance matrices. We place the corresponding element of one association between a predicted state and a measurement into the corresponding position, calculated as $J_{k|k-1}\cdot j + i$, where j is the index of a measurement in Z_k and i is the index of a predicted component. The first $J_{k|k-1}$ elements are for misdetection hypotheses.

As we now see, at every time moment, the update step creates a new set of components, and their number is equal to $J_k = J_{k|k-1}(1+|Z_k|)$. That means that the number of components will grow exponentially if we do not incorporate pruning techniques. The GM-PHD filter, therefore, defines two additional steps of the algorithm: pruning and merging.

Algorithm 2 GM-PHD filter update step

Require: Measurements set Z_k
Require: The predicted Gaussian components $v_{k|k-1}$

procedure UPDATE($Z_k, \{w_{k|k-1}^{(i)}, \mathbf{m}_{k|k-1}^{(i)}, \mathbf{P}_{k|k-1}^{(i)}, T_{k|k-1}^{(i)}\}_{i=1}^{J_{k|k-1}}$)
  for $i \leftarrow 1, \ldots, J_{k|k-1}$ **do** ▷ Precompute common PHD update components
    $\boldsymbol{\eta}_{k|k-1}^{(i)} \leftarrow \mathbf{H}_k \mathbf{m}_{k|k-1}^{(i)}$
    $\mathbf{S}_k^{(i)} \leftarrow \mathbf{H}_k \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^\intercal + \mathbf{R}_k$
    $\mathbf{K}_k^{(i)} \leftarrow \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^\intercal \left[\mathbf{S}_k^{(i)}\right]^{-1}$
    $\mathbf{P}_{k|k}^{(i)} \leftarrow \left[\mathbf{I} - \mathbf{K}_k^{(i)} \mathbf{H}_k\right] \mathbf{P}_{k|k-1}^{(i)}$
  end for
  for $i \leftarrow 1, \ldots, J_{k|k-1}$ **do** ▷ Create misdetection hypothesis components
    $w_k^{(i)} \leftarrow (1 - P_{D,k}) w_{k|k-1}^{(i)}$
    $\mathbf{m}_k^{(i)} \leftarrow \mathbf{m}_{k|k-1}^{(i)}$
    $\mathbf{P}_k^{(i)} \leftarrow \mathbf{P}_{k|k}^{(i)}$
    $T_k^{(i)} \leftarrow T_{k|k-1}^{(i)}$
  end for
  for $j \leftarrow 1, \ldots, |Z_k|$ **do** ▷ Measurements update
    for $i \leftarrow 1, \ldots, J_{k|k-1}$ **do**
      $w_k^{(J_{k|k-1} \cdot j + i)} \leftarrow P_{D,k} w_{k|k-1}^{(i)} \mathcal{N}\left(\mathbf{z}_k^{(j)}; \boldsymbol{\eta}_{k|k-1}^{(i)}, \mathbf{S}_k^{(i)}\right)$
      $\mathbf{m}_k^{(J_{k|k-1} \cdot j + i)} \leftarrow \mathbf{m}_{k|k-1}^{(i)} + \mathbf{K}_k^{(i)} (\mathbf{z}_k^{(j)} - \boldsymbol{\eta}_{k|k-1}^{(i)})$
      $\mathbf{P}_k^{(J_{k|k-1} \cdot j + i)} \leftarrow \mathbf{P}_{k|k}^{(i)}$
      $T_k^{(J_{k|k-1} \cdot j + i)} \leftarrow T_{k|k-1}^{(i)}$
    end for
    for $i \leftarrow 1, \ldots, J_{k|k-1}$ **do** ▷ Update weights
      $w_k^{(J_{k|k-1} \cdot j + i)} \leftarrow \frac{w_k^{(J_{k|k-1} \cdot j + i)}}{\kappa_k(\mathbf{z}_k^{(j)}) + \sum_{l=1}^{J_{k|k-1}} w_k^{(J_{k|k-1} \cdot j + i)}}$
    end for
  end for
  $J_k = (|Z_k| + 1) \cdot J_{k|k-1}$
  return $\{w_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}, T_k^{(i)}\}_{i=1}^{J_k}$
end procedure

Pruning here refers to the technique of reducing the number of overall components by removing those Gaussians whose weight is less than a specified threshold. In other words, we get rid of those components that represent an assignment hypothesis with a low likelihood. The truncation threshold τ is a parameter of the filter and is set in advance. It can be set to any small number, such as 10^{-5}. Formally, the pruning algorithm is summarized in Algorithm 3.

Algorithm 3 GM-PHD filter pruning

Require: Pruning threshold τ
Require: Gaussian components from the update step v_k

procedure PRUNE(truncation threshold τ, $\{w_k^{(i)}, \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}, T_k^{(i)}\}_{i=1}^{J_k}$)
  $I \leftarrow \{i = 1, \ldots, J_k \mid w_k^{(i)} > \tau\}$
  $l \leftarrow 0$
  for $i \in I$ **do**
    $l \leftarrow l + 1$
    $\bar{w}_k^{(l)} \leftarrow \frac{w_k^{(i)}}{\sum_{j \in I} w_k^{(j)}}$
    $\bar{\mathbf{m}}_k^{(l)} \leftarrow \mathbf{m}_k^{(i)}$
    $\bar{\mathbf{P}}_k^{(l)} \leftarrow \mathbf{P}_k^{(i)}$
    $\bar{T}_k^{(l)} \leftarrow T_k^{(i)}$
  end for
  $\bar{J}_k \leftarrow l$
  return $\{\bar{w}_k^{(i)}, \bar{\mathbf{m}}_k^{(i)}, \bar{\mathbf{P}}_k^{(i)}, \bar{T}_k^{(i)}\}_{i=1}^{\bar{J}_k}$
end procedure

Merging in the context of the GM-PHD filter reduces the number of number of Gaussian components by combining hypotheses that are close to each other. The "closeness" of components is calculated using the Mahalanobis distance given in Definition 1.8 and the threshold U is given in advance. In this step, we also ensure that every component in the resulting mixture contains a unique label.

Moreover, to limit the number of components to a reasonable level, the GM-PHD filter incorporates the maximum number of allowed components $J_{\max}$ as another parameter. If the number of remaining components exceeds this threshold after the merging step, we select $J_{\max}$ components with the largest weights and eliminate the rest. In formal language, merging is summarized in Algorithm 4.

The final step of the GM-PHD filter is state extraction. At every time step, we want to extract the locations of existing targets so that we can visualize the filter results or compute performance metrics. The multi-target state estimate will then contain information from components with weights larger than a certain threshold. [32] suggests a threshold equal to 0.5. Single-target state estimates are obtained by taking the mean vectors of the Gaussian components with weights larger than the τ, and those components that have a smaller weight but whose weight was larger than τ at any previous time step. The reason for the second condition is that it enables track continuity for those targets that did not receive measurements at the current time step but still exist. If the target generates a new measurement later and its state estimate appears again in the multi-target state estimate, this target will have a continuous track without interruptions. The state extraction process is summarized in formal language in Algorithm 5.

We provide the final algorithm for the GM-PHD filter in Algorithm 6, including the modified prediction step with fusion of external information covered in the next subsection. For a better overview, we summarize the entire GM-PHD cycle in Figure 2.3.

Algorithm 4 GM-PHD filter merge

Require: Merge threshold U
Require: Maximum number of components $J_{\max}$
Require: Gaussian components after pruning $\bar{v}_k$

procedure MERGE(U, $J_{\max}$, $\{\bar{w}_k^{(i)}, \bar{\mathbf{m}}_k^{(i)}, \bar{\mathbf{P}}_k^{(i)}, \bar{T}_k^{(i)}\}_{i=1}^{\bar{J}_k}$)
  $I \leftarrow \{i = 1, \ldots, \bar{J}_k\}$
  $l \leftarrow 0$
  while $I \neq \emptyset$ **do**
    $l \leftarrow l + 1$
    $j \leftarrow \arg\max_{i \in I} \bar{w}_k^{(i)}$
    $L \leftarrow \left\{ i \in I \mid (\bar{\mathbf{m}}_k^{(i)} - \bar{\mathbf{m}}_k^{(j)})^\intercal [\bar{\mathbf{P}}_k^{(i)}]^{-1} (\bar{\mathbf{m}}_k^{(i)} - \bar{\mathbf{m}}_k^{(j)}) \leq U \right\}$
    $\tilde{w}_k^{(l)} \leftarrow \sum_{i \in L} \bar{w}_k^{(i)}$
    $\tilde{\mathbf{m}}_k^{(l)} \leftarrow \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} \bar{w}_k^{(i)} \bar{\mathbf{m}}_k^{(i)}$
    $\tilde{\mathbf{P}}_k^{(l)} \leftarrow \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} \bar{w}_k^{(i)} \left(\bar{\mathbf{P}}_k^{(i)} + (\tilde{\mathbf{m}}_k^{(l)} - \bar{\mathbf{m}}_k^{(i)})(\tilde{\mathbf{m}}_k^{(l)} - \bar{\mathbf{m}}_k^{(i)})^\intercal \right)$
    $\tilde{T}_k^{(l)} \leftarrow \bar{T}_k^{(j)}$
    $I \leftarrow I \setminus L$
  end while
  if $l > J_{\max}$ **then**
    Take $J_{\max}$ components from $\{\tilde{w}_k^{(i)}, \tilde{\mathbf{m}}_k^{(i)}, \tilde{\mathbf{P}}_k^{(i)}, \tilde{T}_k^{(i)}\}_{i=1}^{l}$ with largest weights
    $l \leftarrow J_{\max}$
  end if
  $\tilde{J}_k \leftarrow l$
  $I \leftarrow \{1, \ldots, \tilde{J}_k\}$
  while $I \neq \emptyset$ **do** ▷ Uniquify labels
    $j \leftarrow \arg\max_{i \in I} \tilde{w}_k^{(i)}$
    $L \leftarrow \left\{ i \in I \setminus \{j\} \mid \tilde{T}_k^{(i)} = \tilde{T}_k^{(j)} \right\}$
    for $i \in L$ **do**
      $\tilde{T}_k^{(i)} \leftarrow$ A new unique unused tag
    end for
    $I \leftarrow I \setminus L$
  end while
  return $\{\tilde{w}_k^{(i)}, \tilde{\mathbf{m}}_k^{(i)}, \tilde{\mathbf{P}}_k^{(i)}, \tilde{T}_k^{(i)}\}_{i=1}^{\tilde{J}_k}$
end procedure

Algorithm 5 GM-PHD filter state extraction

Require: Gaussian components after merge $\tilde{v}_k$

procedure EXTRACTSTATES($\{\tilde{w}_k^{(i)}, \tilde{\mathbf{m}}_k^{(i)}, \tilde{\mathbf{P}}_k^{(i)}, \tilde{T}_k^{(i)}\}_{i=1}^{\tilde{J}_k}$)
  $\hat{T}_k \leftarrow \{\tilde{T}_k^{(i)} \mid \tilde{w}_k^{(i)} > 0.5, i = 1, \ldots, \tilde{J}_k\}$
  $\hat{X}_k \leftarrow \{\tilde{\mathbf{m}}_k^{(i)} \mid \tilde{T}_k^{(i)} \in \hat{T}_j, j = 1, \ldots, k\}$
  return $\hat{X}_k$
end procedure

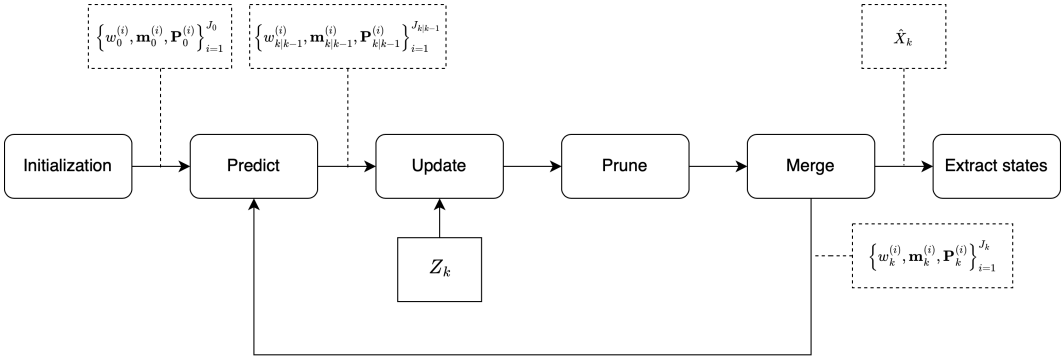


Figure 2.3 The visualization of the GM-PHD cycle.

2.4 External information fusion

In many real-world scenarios, one sensor and one source of information may not suffice. Sensors in general have specific technical specifications and limitations. For example, imagine a surveillance system tracking enemy troops. A raw image from a camera working in the visible spectrum may not capture information about people using camouflage, and therefore, tracking such targets with this camera alone becomes intractable. However, if we use additional information from an infrared camera, targets become clearly visible and combined measurements will have a higher signal-to-noise ratio. Techniques involving combining information from different sources are called information fusion. In this section, we will briefly describe the types of information fusion that exist and describe how this information can be used to get more accurate tracking results for the GM-PHD filter.

Information fusion, or information integration, is generally a broad term, mainly because sources of information differ a lot. For instance, *data fusion* is the process of merging data from different data sources and in different formats to create a comprehensive picture of the environment [37]. Data sources may refer to measurements from sensors, data from a database, or human input, while data types may include text, images, audio inputs, or binary data. On the other hand, *decision fusion* refers to collecting decisions from different algorithms that may look at the problem from different perspectives or work with different data inputs, and combining these decisions to create one final decision. Ensembles of different machine learning algorithms, for example, for pattern prediction, are an example of decision fusion [38].

In this work, we will refer to another fusion technique called *sensor fusion*. In principle, sensor fusion is a subset of data fusion, which involves collecting data from various sensors and merging it to get a more precise and complete estimate of the environment. The example from the introduction to this section is also an example of sensor fusion. There may be many different types of sensors, including cameras, radars, lidars, sonars, or infrared cameras. Every sensor measures different aspects of the tracked environment, and the combined information describes the surrounding world more precisely.

Regarding Bayesian inference and multi-target tracking, we create estimates of moving objects, and there are many variables that can input uncertainty: the detection rate of the sensor, the quality of the tracking algorithm and its implementation, the environment conditions where the sensor operates, etc. And while the algorithm may be replaced with the one providing better estimates for a specific problem, physical limitations of sensors and the environment cannot be addressed easily. For example, let us take the AN/MPQ-53 radar, which is a part of the Patriot system, a surface-to-air missile system used for air defense deployed in many countries. The exact technical specifications of the radar are far beyond the scope of this work. However, we will give an example of what may cause the radar to fail to detect targets.

The AN/MPQ-53 radar has a 120-degree field of view with a 170 km range [39]. However, there may be physical limitations of the environment that create blind and clutter zones. For instance, imagine the situation where the radar is deployed on a surface and is targeted to track air objects above the sea. The water may cause spontaneous reflections of the radar rays and cause clutter measurements. Moreover, the curvature of the terrain may create a blind zone so that the radar will not be able to track targets flying at low altitudes. This example is shown graphically in Figure 2.4.

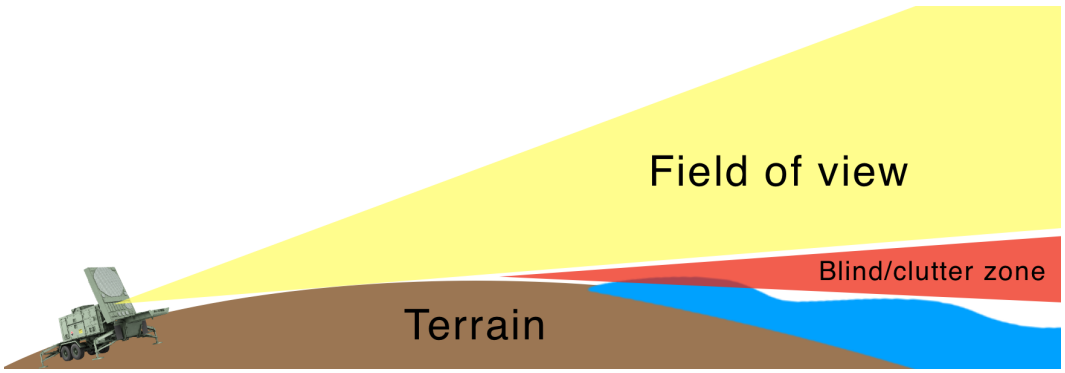


Figure 2.4 An example of the AN/MPQ-53 radar deployed on the surface with curvature and targeted at the sea. The curvature causes the blind zone, and reflections from the water surface may cause clutter measurements.

This problem may be addressed in several ways, one of which, that is covered in this work, is to use information fusion techniques. For example, we can incorporate additional information from other sensors placed after the curvature that supplements additional data about the blind zone and therefore covers the area that stays invisible to the radar. This source of additional information may be another radar or a different type of sensors, or even a soldier with binoculars. We can then input this additional information to the GM-PHD filter to improve the tracking performance. The example of such mitigation is illustrated in Figure 2.5.

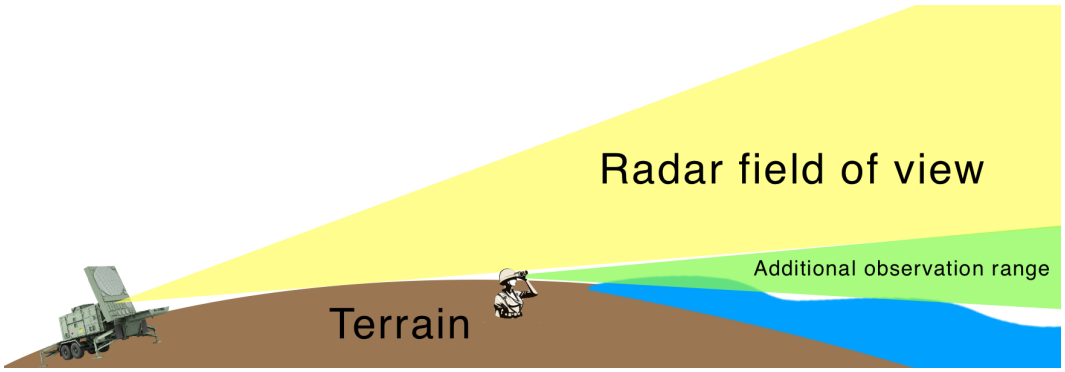


Figure 2.5 An example of the AN/MPQ-53 radar deployed on the surface with curvature and targeted at the sea. After the curvature, there is a soldier with binoculars with the additional observation range that covers the blind zone of the deployed radar.

We now define the imputation of additional data formally. Given that an instance of the GM-PHD filter with the CV model is running and at every time step k it gets measurements from a sensor mixed with clutter in a set Z_k. Let there be an additional sensor that operates in the blind zone of the first radar. The second sensor measures position and velocity of targets $\mathbf{m}_\psi$ appearing in the blind zone with the uncertainty given by the covariance matrix $\mathbf{P}_\psi$ and the

weight denoting the expected number of targets at this location, denoted $w_{\psi,k}$. Also, every such component is assigned with a unique tag $T_{\psi,k}$. All measurements of the second sensor create a set of additional states at time k, denoted $\Psi_{F,k} = \{w_{\psi,k}^{(i)}, \mathbf{m}_{\psi}^{(i)}, \mathbf{P}_{\psi}(i), T_{\psi,k}^{(i)}\}_{i=1}^{J_{\psi,k}}$, where $J_{\psi,k}$ denotes the number of additional state estimates. Given that the dynamics of targets is Gaussian-linear, we propose that the intensity function of these additional measurements as $\psi_k(\mathbf{x})$, which is a Gaussian mixture of the following form:

$$\psi_k(\mathbf{x}) = \sum_{i=1}^{J_{\psi,k}} w_{\psi,k}^{(i)} \mathcal{N}\left(\mathbf{x}; \mathbf{m}_{\psi,k}^{(i)}, \mathbf{P}_{\psi,k}^{(i)}\right). \tag{2.55}$$

The GM-PHD filter is then modified to include additional information. We include this intensity in the prediction step of the GM-PHD filter, defined in Theorem 2.12, that will thus have the following form:

▶ **Theorem 2.14** (The GM-PHD filter with external information prediction)**.** *Given that assumptions from Section 2.3.3 apply and the posterior intensity of the previous time step $v_{k-1}(\mathbf{x}_{k-1}|Z_{1:k-1})$ is a Gaussian mixture of the form:*

$$v_{k-1}(\mathbf{x}_{k-1}|Z_{1:k-1}) = \sum_{i=1}^{J_{k-1}} \mathcal{N}\left(\mathbf{x}_{k-1}; \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)}\right), \tag{2.56}$$

where every Gaussian component has a unique tag or label assigned to it, which forms the set of tags:

$$T_{k-1} = \{T_{k-1}^{(1)}, \ldots, T_{k-1}^{(J_{k-1})}\}. \tag{2.57}$$

Also, let additional state estimates from other sensors have the form $\Psi_{F,k} = \{w_{\psi,k}^{(i)}, \mathbf{m}_{\psi}^{(i)}, \mathbf{P}_{\psi}^{(i)}, T_{\psi,k}^{(i)}\}_{i=1}^{J_{\psi,k}}$ and the fusion intensity is given by Equation 2.55. Then the predicted intensity at time k is also a Gaussian mixture, defined as:

$$v_{k|k-1}(\mathbf{x}_k|Z_{1:k-1}) = \gamma_k(\mathbf{x}_k) + \psi_k(\mathbf{x}_k) + v_{S,k|k-1}(\mathbf{x}_k), \tag{2.58}$$

where $\gamma_k(\mathbf{x}_k)$ is given in Equation 2.35 and $v_{S,k|k-1}(\mathbf{x}_k)$ is the PHD function of survived targets given in Theorem 2.12. The set of tags after the prediction step forms the union of the set of tags from the previous time step with the set of unique tags of components spontaneously born at time k and tags from Gaussian terms from another sensor:

$$T_{k|k-1} = T_{k-1} \cup \{T_{\gamma_k}^{(1)}, \ldots, T_{\gamma_k}^{(J_{\gamma_k})}\} \cup \{T_{\psi}^{(1)}, \ldots, T_{\psi}^{(J_{\psi})}\}. \tag{2.59}$$

All other steps of the GM-PHD filter remain the same. The final algorithm of the GM-PHD filter with external information fusion is given in Algorithm 6.

In this chapter, we introduced the fundamentals of multi-object tracking and discussed ways how we can track multiple targets using measurements from a sensor in clutter. Subsequently, we described the PHD filter, which utilizes the Final Set Statistics to propagate the posterior intensity function of the multi-target probability distribution. Furthermore, we introduced the GM-PHD filter, a closed-form solution of the PHD filter for Gaussian-linear dynamics of moving objects. Lastly, we extended the GM-PHD filter with fusion of external information, which enhanced the tracking performance for those cases where one sensor may fail to detect targets due to the effects of physical limitations of the sensor or environment. In the next chapter, we will describe the implementation of the GM-PHD filter and its performance on different testing cases and metrics.

Algorithm 6 GM-PHD filter with additional information fusion

Require: Truncation threshold τ
Require: Merge threshold U
Require: Maximum number of components $J_{\max}$
Require: Initial Gaussian terms $v_0 = \{w_0^{(i)}, \mathbf{m}_0^{(i)}, \mathbf{P}_0^{(i)}, T_0^{(i)}\}_{i=1}^{J_0}$

procedure GM-PHD-Fusion($Z_k, \Psi_k, v_{k-1} = \{w_{k-1}^{(i)}, \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)}, T_{k-1}^{(i)}\}_{i=1}^{J_{k-1}}$)
  $v_{k|k-1} \leftarrow$ Predict-Fuse(Ψ_k, v_{k-1})
  $v_k \leftarrow$ Update($Z_k, v_{k|k-1}$) ▷ Defined in Algorithm 2
  $\bar{v}_k \leftarrow$ Prune(τ, v_k) ▷ Defined in Algorithm 3
  $\tilde{v}_k \leftarrow$ Merge($U, \bar{v}_k$) ▷ Defined in Algorithm 4
  $\hat{X} \leftarrow$ StateExtract($\tilde{v}_k$) ▷ Defined in Algorithm 5
  return $\hat{X}$
end procedure

procedure Predict-Fuse($\Psi_k, \{w_{k-1}^{(i)}, \mathbf{m}_{k-1}^{(i)}, \mathbf{P}_{k-1}^{(i)}, T_{k-1}^{(i)}\}_{i=1}^{J_{k-1}}$)
  $j \leftarrow 0$
  for $i \leftarrow 1, \ldots, J_{\gamma,k}$ **do** ▷ Create spontaneous birth components
    $i \leftarrow i + 1$
    $w_{k|k-1}^{(j)} \leftarrow w_{\gamma,k}^{(i)}$
    $\mathbf{m}_{k|k-1}^{(j)} \leftarrow \mathbf{m}_{\gamma,k}^{(i)}$
    $\mathbf{P}_{k|k-1}^{(j)} \leftarrow \mathbf{P}_{\gamma,k}^{(i)}$
    $T_{k|k-1}^{(j)} \leftarrow T_{\gamma,k}^{(i)}$
  end for
  for $i \leftarrow 1, \ldots, J_{\psi,k}$ **do** ▷ Fuse external information
    $i \leftarrow i + 1$
    $w_{k|k-1}^{(j)} \leftarrow w_{\psi,k}^{(i)}$
    $\mathbf{m}_{k|k-1}^{(j)} \leftarrow \mathbf{m}_{\psi,k}^{(i)}$
    $\mathbf{P}_{k|k-1}^{(j)} \leftarrow \mathbf{P}_{\psi,k}^{(i)}$
    $T_{k|k-1}^{(j)} \leftarrow T_{\psi,k}^{(i)}$
  end for
  for $i \leftarrow 1, \ldots, J_{k-1}$ **do** ▷ Prediction for existing targets
    $i \leftarrow i + 1$
    $w_{k|k-1}^{(j)} \leftarrow P_{S,k} w_{k-1}^{(i)}$
    $T_{k|k-1}^{(j)} \leftarrow T_{k-1}^{(i)}$
    $\mathbf{m}_{S,k|k-1}^{(i)} \leftarrow \mathbf{F}_{k-1} \mathbf{m}_{k-1}^{(i)}$
    $\mathbf{P}_{S,k|k-1}^{(i)} \leftarrow \mathbf{F}_{k-1} \mathbf{P}_{k-1}^{(i)} \mathbf{F}_{k-1}^{\intercal} + \mathbf{Q}_{k-1}$
  end for
  $J_{k|k-1} \leftarrow j$
  return $\{w_{k|k-1}^{(i)}, \mathbf{m}_{k|k-1}^{(i)}, \mathbf{P}_{k|k-1}^{(i)}, T_{k|k-1}^{(i)}\}_{i=1}^{J_{k|k-1}}$
end procedure

Chapter 3

Implementation and tests

This chapter presents the methodology we used to test the performance of the GM-PHD filter. Firstly, we introduce two metrics that are appropriate for use in measuring the performance of a multi-target tracking algorithm. Secondly, we describe in detail several test scenarios on which the testing was conducted. Finally, we explain the ranges of different initial settings that were tested and describe the implementation of the GM-PHD filter.

3.1 Metrics

For evaluating the performance of the GM-PHD filter, we used two metrics proposed in [32]. The first metric is called the Circular Position Error Probability (CPEP). It measures the probability that the estimated position of a target falls in a circular region with a given radius around the true position. This metric has its roots in the military area, where the weapons system precision is measured with the probability of hitting targets in a certain ellipse with a given mean [40]. While linear metrics like a simple Euclidean distance would measure the accuracy, the CPEP metrics measures precision of the tracking algorithm. In formal language, this metric is defined as:

$$\mathrm{CPEP}_k(r) = \frac{1}{|X_k|} \sum_{\mathbf{x} \in X_k} \rho_k(\mathbf{x}, r), \tag{3.1}$$

where r is the radius of the circluar area, and $\rho_k(\mathbf{x}, r)$ is defined as:

$$\rho_k(\mathbf{x}, r) = \Pr\{\|\mathbf{H}_k\hat{\mathbf{x}} - \mathbf{H}_k\mathbf{x}\|_2 > r, \forall \hat{\mathbf{x}} \in \hat{X}_k\}. \tag{3.2}$$

Here, $\hat{X}_k$ refers to the set of estimates generated by the tracking algorithm at time step k, $\mathbf{H}_k$ is the single-target measurement model, and $\|\cdot\|_2$ is the Euclidean norm of a vector.

The second measure of tracking error is the expected absolute error on the number of targets. In simple words, this metric gives the average difference between the predicted and true number of targets. Mathematically, it is defined as:

$$E\left[\left||\hat{X}_k| - |X_k|\right|\right]. \tag{3.3}$$

Generally, these two metrics measure the total uncertainty of a multi-target tracking filter given be a random finite set – both the number of targets in the set and the precision of the estimates. We evaluate these metrics for all tracking scenarios to show the influence of a parameter change or the presence of fusion techniques on the filter performance.

3.2 Test scenarios

We evaluate the performance of the algorithm in four different test scenarios using multiple parameter values. To facilitate reference to these scenarios later in the report, we denote them as **(C1)**–**(C4)**. The following subsections provide a detailed description of each test case. Here, we describe the complete setup common to all scenarios.

All test cases use the Gaussian-linear Constant Velocity motion and measurement models described in Section 1.3.2.1. Targets are assumed to move in 2D Euclidean space, where distance is measured in meters and time in seconds. Additionally, we assume that the sampling period is constant and set to $1s$. The process noise q is set to $5m/s^2$ for every target and every test case. Consequently, the single-target state vector $\mathbf{x}_k$, the single-target state transition matrix $\mathbf{F}$, and the process noise matrix $\mathbf{Q}$ have the following form:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix}; \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{Q} = \begin{bmatrix} \frac{25}{3} & 0 & \frac{25}{2} & 0 \\ 0 & \frac{25}{3} & 0 & \frac{25}{2} \\ \frac{25}{2} & 0 & 25 & 0 \\ 0 & \frac{25}{2} & 0 & 25 \end{bmatrix}. \tag{3.4}$$

Moreover, we set the standard deviation of the measurement noise to a constant value for all test cases, $r = 10m$. The measurement matrix $\mathbf{H}$ and the measurement noise matrix $\mathbf{R}$ are defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \quad R = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}. \tag{3.5}$$

The sensor field of view is a 2D square with an area of $V = 4 \times 10^6 m^2$, with boundaries delimited by $[-1000, 1000] \times [-1000, 1000]$ meters. This rectangular region corresponds to the surveillance area where the sensor is deployed. The sensor is assumed to be perfect with infinite resolution.

The initial detection and survival probabilities are set to $P_{D,k} = 0.98$ and $P_{S,k} = 0.99$, respectively. We will see the metrics for different values of these parameters in the next chapter.

The birth process is initialized as a Gaussian mixture with two components, where each component represents the location where new targets are expected to be born and its weight denotes the likelihood of a real target being present in the area. The specific relationships for each test case differ and will be presented in the following subsections.

The noise process is modeled using a Poisson random finite set, and the clutter measurements are Poisson distributed in time and uniformly distributed in space. The clutter intensity is denoted by a normalized intensity over the surveillance region, the clutter spatial intensity λ_c, and is set to a constant $\lambda_c = 12.5 \times 10^{-6} m^{-2}$. With the surveillance area equal to $V = 4 \times 10^6 m^2$, this Poisson process generates on average 50 clutter measurements at each time step. In later sections, we will evaluate the impact of changing the spatial clutter intensity, represented by the parameter λ_c, on the performance of the GM-PHD filter by testing it for different values of λ_c. This clutter process is described by the following equation:

$$\kappa_k(\mathbf{z}) = \lambda_c V u(\mathbf{z}), \tag{3.6}$$

where $u(\mathbf{z})$ represents the bivariate uniform distribution over the surveillance area $[x_0, x_1] \times [y_0, y_1]$, given by:

$$u(\mathbf{z}) = \begin{cases} \frac{1}{(x_1 - x_0)(y_1 - y_0)} & \text{if } \mathbf{z} \in [x_0, x_1] \times [y_0, y_1], \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

The truncation threshold for the pruning step of the GM-PHD filter is set to $\tau = 10^{-5}$, and the merging threshold is set to $U = 4$. The impact of changing these parameters on the filter

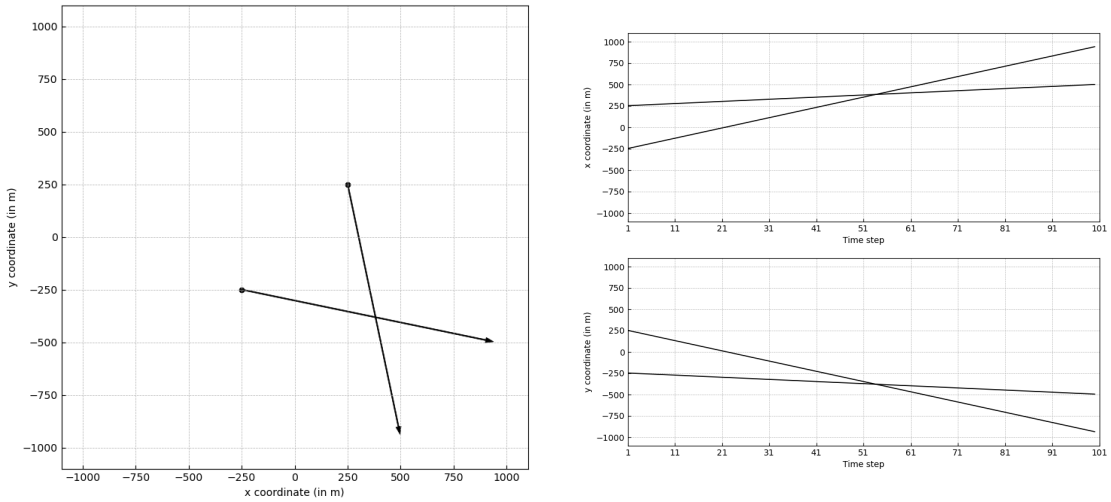


Figure 3.1 The (C1) scenario. In the left figure, we see the tracks of two objects in the 2D space, locations of their births (black circles) and deaths (arrow heads). In the right figure, we see how each coordinate of the tracks of both objects changes over time. We can see that the tracks of two objects intersect at time $k = 53$.

performance will be discussed. The maximum number of components is fixed at $J_{\max} = 1000$, intentionally set higher than the value proposed in [32] to evaluate different parameter settings with more precision, particularly in scenarios with a higher number of targets.

In the following subsections, we will provide a detailed description of the dynamics of targets for each test case. We will then evaluate the GM-PHD filter performance for each scenario and discuss the results.

3.2.1 Two objects with crossing paths (C1)

This test case is fully identical to the one given in [32] and demonstrates the accuracy of the implementation by showing that the results obtained by the authors of the original paper are reproducible.

In this scenario, two objects are born at the same time at $k = 1$ and follow independent linear paths. The initial state vectors of these objects are the following:

$$\mathbf{x}_1^{(1)} = \begin{bmatrix} 250.0 \\ 250.0 \\ 2.5 \\ -12.0 \end{bmatrix}, \qquad \mathbf{x}_1^{(2)} = \begin{bmatrix} -250.0 \\ -250.0 \\ 12 \\ -2.5 \end{bmatrix}. \tag{3.8}$$

At a certain moment, their tracks cross. Both objects disappear at time $k = 100$. Therefore, the number of objects remains constant throughout the simulation, with two objects present at all times. The paths of both objects are shown in Figure 3.1. The arrows show the direction of movement, and the position of the arrowhead denotes the position of the object death. The circles on the other side of the line illustrate the location of the target birth.

Finally, the initial Gaussian mixture of the birth intensity in this scenario is given by:

$$\gamma_k(\mathbf{x}) = 0.1\mathcal{N}\left(\mathbf{x}; \mathbf{m}_\gamma^{(1)}, \mathbf{P}_\gamma\right) + 0.1\mathcal{N}\left(\mathbf{x}; \mathbf{m}_\gamma^{(2)}, \mathbf{P}_\gamma\right), \tag{3.9}$$

where:

$$\mathbf{m}_{\gamma}^{(1)} = \begin{bmatrix} 250 \\ 250 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{m}_{\gamma}^{(2)} = \begin{bmatrix} -250 \\ -250 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{P}_{\gamma} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}. \tag{3.10}$$

3.2.2 Objects are born and die at different times (C2)

This test case represents a scenario in which different objects are born at different times and move in various directions. It demonstrates how well the filter performs on tracking multiple objects when the number of true targets X_k changes over time.

The birth intensity consists of two Gaussian terms, as in the (C1) case, each with a weight of 0.1, and is expressed as follows:

$$\gamma_k(\mathbf{x}) = 0.1\mathcal{N}\left(\mathbf{x}; \mathbf{m}_{\gamma}^{(1)}, \mathbf{P}_{\gamma}\right) + 0.1\mathcal{N}\left(\mathbf{x}; \mathbf{m}_{\gamma}^{(2)}, \mathbf{P}_{\gamma}\right), \tag{3.11}$$

where:

$$\mathbf{m}_{\gamma}^{(1)} = \begin{bmatrix} -1000 \\ 750 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{m}_{\gamma}^{(2)} = \begin{bmatrix} 1000 \\ -750 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{P}_{\gamma} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 30 \end{bmatrix}. \tag{3.12}$$

In total, there are six different tracks, with a maximum of five targets present at a time and a minimum of one target at a time step. Objects are born every 10 time steps, and each has a lifespan of 50 time steps. The paths of the six targets are depicted in Figure 3.2. The state vectors of the six targets are as follows:

$$\mathbf{x}_{0}^{(1)} = \begin{bmatrix} -1000.0 \\ 750.0 \\ 25.0 \\ 0.0 \end{bmatrix}, \quad \mathbf{x}_{10}^{(2)} = \begin{bmatrix} 1000.0 \\ -750.0 \\ -25.0 \\ 0.0 \end{bmatrix}, \quad \mathbf{x}_{20}^{(3)} = \begin{bmatrix} -1000.0 \\ 750.0 \\ 17.68 \\ -17.68 \end{bmatrix},$$
$$\mathbf{x}_{30}^{(4)} = \begin{bmatrix} 1000.0 \\ -750.0 \\ -17.68 \\ 17.68 \end{bmatrix}, \quad \mathbf{x}_{40}^{(5)} = \begin{bmatrix} -1000.0 \\ 750.0 \\ 0.0 \\ -25.0 \end{bmatrix}, \quad \mathbf{x}_{50}^{(6)} = \begin{bmatrix} 1000.0 \\ -750.0 \\ 0.0 \\ 25.0 \end{bmatrix}. \tag{3.13}$$

3.2.3 Unexpected objects with no fusion (C3)

The third scenario is based on the second one, (C2), and it tests the appearance of two additional components on two unexpected locations. The birth intensity is defined by the same relation as in Equations 3.11 and 3.12. The initial states of objects, including the time of birth is also the same and given by Equation 3.13. However, there are also two additional objects entering the scene at time $k = 20$ and $k = 60$. Their initial state vector are given by:

$$\mathbf{x}_{20}^{(7)} = \begin{bmatrix} -1000.0 \\ -1000.0 \\ 20.0 \\ 20.0 \end{bmatrix}, \qquad \mathbf{x}_{60}^{(8)} = \begin{bmatrix} 1000.0 \\ 1000.0 \\ -20.0 \\ -20.0 \end{bmatrix}. \tag{3.14}$$

The lifespan of these objects is 40 time steps of each. The whole picture of tracks in the scene is depicted in Figure 3.3.

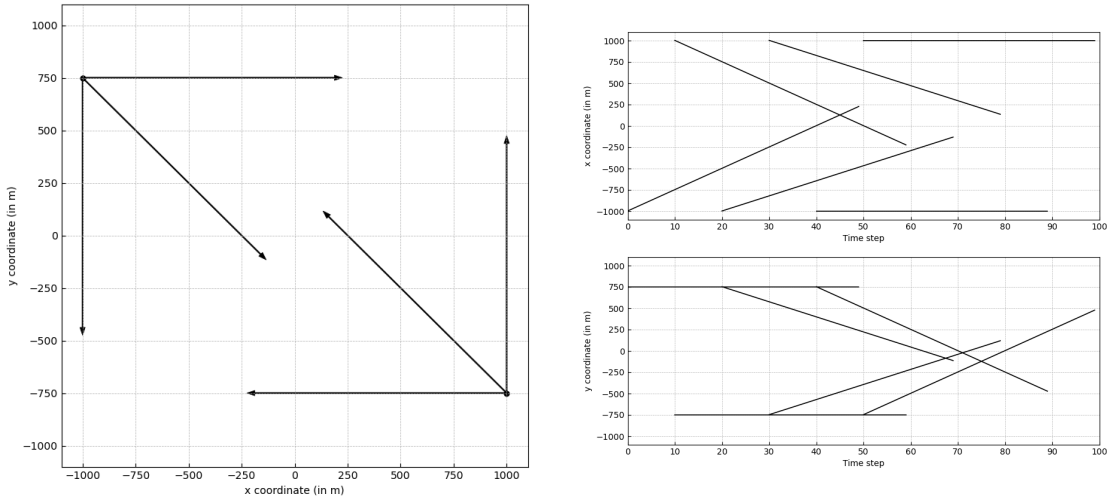


Figure 3.2 The (C2) scenario. In the left figure, we can see the tracks of multiple objects in the 2D space, their locations of birth (black circles) and death (arrow heads). The right figure shows how the coordinates of each object track change over time and the variation in the number of objects present in the scene.

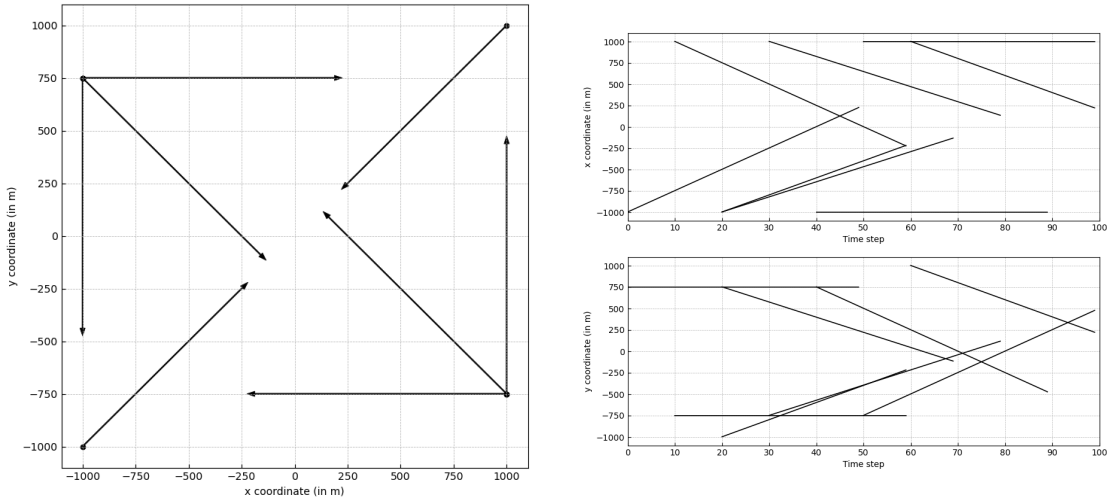


Figure 3.3 The (C3) scenario. In the left figure, we can see the tracks of multiple objects in the 2D space, their locations of birth (black circles) and death (arrow heads). In comparison to the (C2) case, two additional objects appear on two additional positions. The right figure shows how the coordinates of each object track change over time.

For instance, this scenario may occur in cases when a sensor has a blind zone, and the objects appear in the field of view too late. We will later show that without imputation of external information, the GM-PHD filter is unable to capture the dynamics of these objects.

3.2.4 Unexpected objects with external information fusion (C4)

The last test scenario is identical to the previous one, (C3), with the addition of two fused components. The example provided in Section 2.4 suggests that the additional information may be provided by another sensor or, for instance, by a soldier. The filter is then able to capture objects that remain invisible due to its physical limitations or the environmental conditions.

The birth intensity is given by Equations 3.11 and 3.12. Initial states of moving targets are defined by Equations 3.13 and 3.14, and tracks of moving targets are visually demonstrated in Figure 3.3. However, in this test case, we additionally create fusion sets at times $k = 20$ and $k = 60$ with one element. At every other time step, the fusion set remains empty. Recall that the fusion intensity ψ_k is given by Equation 2.55. The set $\Psi_{F,k}$ in our scenario is defined as follows:

$$\Psi_{F,k} = \begin{cases} \left\{(w_{20}^{(1)}, \mathbf{m}_{\psi,20}^{(1)}, \mathbf{P}_{\psi,20}^{(1)}, T_{\psi,20})\right\} & \text{if } k = 20, \\ \left\{(w_{60}^{(2)}, \mathbf{m}_{\psi,60}^{(2)}, \mathbf{P}_{\psi,60}^{(2)}, T_{\psi,60})\right\} & \text{if } k = 60, \\ \emptyset & \text{otherwise,} \end{cases} \tag{3.15}$$

where:

$$w_{20}^{(1)} = 0.6, \qquad \mathbf{m}_{\psi,20}^{(1)} = \begin{bmatrix} -1050 \\ -1050 \\ 25 \\ 25 \end{bmatrix}, \qquad \mathbf{P}_{\psi,20}^{(1)} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}, \tag{3.16}$$

$$w_{60}^{(2)} = 0.7, \qquad \mathbf{m}_{\psi,60}^{(2)} = \begin{bmatrix} 1020 \\ 1020 \\ -22 \\ -22 \end{bmatrix}, \qquad \mathbf{P}_{\psi,60}^{(3)} = \begin{bmatrix} 40 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}. \tag{3.17}$$

and $T_{\psi,20}$ and $T_{\psi,20}$ are some unique tags for new Gaussian components.

We assume that there are two sensors, each measuring at two different locations. The first sensor generates estimates with larger uncertainty; therefore, the covariance matrix has higher values, and the weight, the measure of uncertainty of the sensor, is lower. The second sensor, which inputs information at time $k = 60$, has better precision, and the estimate of the position of the second object and its speed is more precise, thus having a higher weight value.

3.3 Parameters testing and methodology

Each test scenario **(C1)**–**(C4)** has default settings as defined in Section 3.2. However, to obtain a complete picture of how the GM-PHD filter behaves in different situations, we also evaluate each test case with various combinations of filter settings. This testing is performed by fixing the default values of every parameter and iterating through different values of one parameter at a time. The list of tested settings includes:

- Clutter spatial density $\lambda_c \in [0, 1.25, 2.5, 3.75, 5, 6.25, 7.5, 8.75, 10, 11.25, 12.5] \times 10^{-6}$
- Detection probability $P_{D,k} \in [0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0]$

- Survival probability $P_{S,k} \in [0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0]$
- Truncation threshold $\tau \in [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}]$
- Merge threshold $U \in [4, 10, 20, 50]$

The GM-PHD filter with additional information fusion was implemented using Python 3.10 and the following libraries:

- NumPy v1.23.5 for mathematical operations and efficient matrix multiplication,
- SciPy v1.10.0 for a convenient and reliable sampling from multivariate distributions,
- Matplotlib v3.7.0 for plot generation and visualization.

Additionally, for numerical stability of calculation of covariance matrices, so that they remain symmetrical and positive definite, a small trick is applied:

$$\mathbf{P} = 0.5 \times (\mathbf{P} + \mathbf{P}^\intercal). \tag{3.18}$$

For each testing scenario and every combination of default settings and changed parameters, a testing case is created. Each testing case is evaluated using 100 Monte Carlo input data samples, which are randomly generated measurements, clutter, and misdetections. The filter is run on each sample separately, resulting in 100 filter runs per testing case. In total, we conducted $13,200$ runs of the GM-PHD filter. The testing was performed using a MacBook Pro model A2141 with a 2.6 GHz 6-Core Intel Core i7 CPU and 16 GB RAM. The total running time, including steps from data generation to evaluation and plotting, was approximately 17 hours.

Chapter 4

Results analysis

In this chapter, we present the results of a thorough testing of different parameters of the GM-PHD filter on different test cases presented in the previous chapter. For convenience, we have split this chapter into multiple sections, referring to each test scenario and included multiple plots that depict the change of both the CPEP metric and the expected absolute error on the number of targets. Finally, we discuss the results, outline the strengths and weaknesses of the GM-PHD filter, and compare the results of the (C3) and (C4) scenarios, with and without external information fusion.

4.1 Test results (C1)

We begin by examining the simplest scenario where the number of targets is constant in time and there are only two targets present in the scene. As mentioned before, we generated 100 Monte Carlo measurement and clutter samples for every test scenario and parameter setting, given that the true trajectories of these objects do not change in time. Figure 4.1 shows one of these samples for the default filter settings described in Section 3.2. The left image displays how the true tracks of two objects are generated. The transition from black to yellow color illustrates the change in time for a target. The red stars represent the measurements that the filter receives at a time step, with the intensity of the red color representing the time step. Gray crosses indicate clutter measurements, with the intensity of the gray color having the same meaning as for the red stars. On the right side of the figure, we see the change of both coordinates in time with noise measurements and the estimates of the filter represented by black circles at each time step.

We can observe that the filter provides very accurate estimates even in cluttered environments. For instance, in this case, the filter receives an average of 50 noise measurements for every measurement set. However, we also notice that estimates are not perfect. For instance, one of the objects did not receive estimates at time $k = 21$ and it lasted for three time steps, and at $k = 24$, the object appeared again.

Let us examine how changes in settings affect the filter performance in terms of metrics. To illustrate all runs for every parameter, we use a box plot. For every value of a parameter, we draw a box whose boundaries represent the first and third quartiles. The "whiskers" depict the minimum and maximum values of the distribution of metric values from all runs, and black dots represent those measurements that are considered outliers. The minimum and maximum values are calculated as one and a half times the interquartile range from the first and third quartiles, respectively, where the interquartile range is the difference between the two quartiles. Outliers are defined as those values that fall outside the range between the minimum and maximum. The black line inside boxes represents the median value, and the red dot is for the mean value. Means

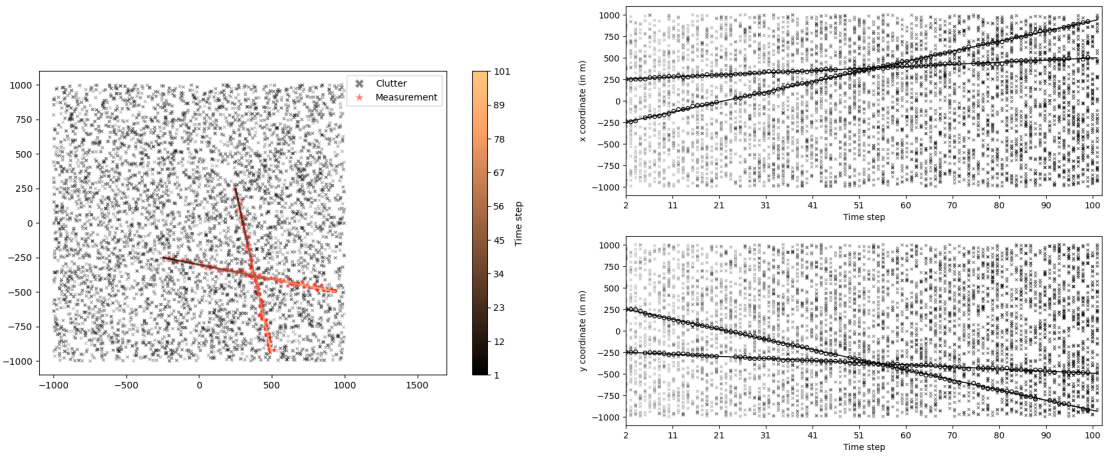


Figure 4.1 One sample of data and estimates for the (C1) scenario. Left: True tracks of two objects (black to yellow) with clutter measurements (gray crosses) and received measurements (red stars) for a single Monte Carlo sample. Right: Change of both coordinates in time with noise measurements (red circles) and filter estimates (black circles) for the same Monte Carlo sample. The filter shows accurate estimates even in very cluttered environments.

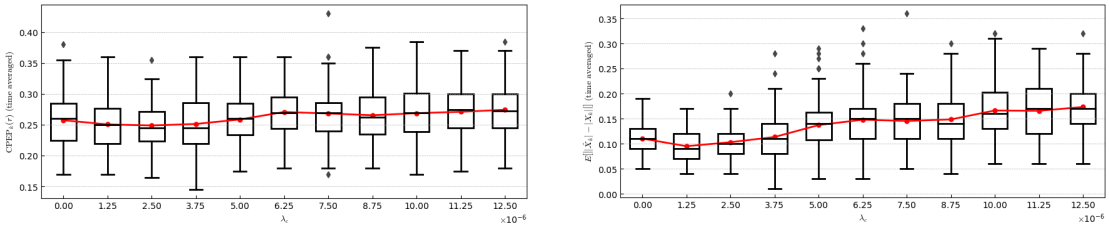


Figure 4.2 Here, we examine the change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C1) scenario for different values of the clutter spatial rate λ_c. Every box is the representation of the distribution of 100 independent samples. As the clutter spatial rate increases, the estimation error also increases, which may be observed on the growing trend of both metrics. All other setting are set to default values, i.e. $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

are connected to better see the overall trend.

First, we examine the change of performance for different clutter rates. In Figure 4.2, we see that the increase of clutter measurements worsens the performance in both the precision of target estimates, and the estimate of the number of targets. It should be noted, that the increase in the expected absolute error is more pronounced than for the CPEP. It is clear that higher clutter rates causes more spurious estimates that the filter may consider to be real targets. Even in cases where these false tracks are not confirmed later, the number of estimated targets can be higher than the real number at a given time moment. However, in general, we see that the increase is not drastic, and the performance of the filter is good for even high values of the clutter rate.

Next, let us look how the filter behaves for different values of the detection probability. The detection probability is a parameter that may vary for different environments and sensors, and its setting may affect the overall tracking performance. Figure 4.3 shows how the estimation error changes for multiple settings of the detection probability. It can be seen that low detection probabilities influence the performance of the filter drastically in both the estimation of the number of targets and the precision of state estimates. For instance, the comparison of two values of $P_{D,k} = 0.7$ and $P_{D,k} = 1.0$ gives more than seven times better estimation results in

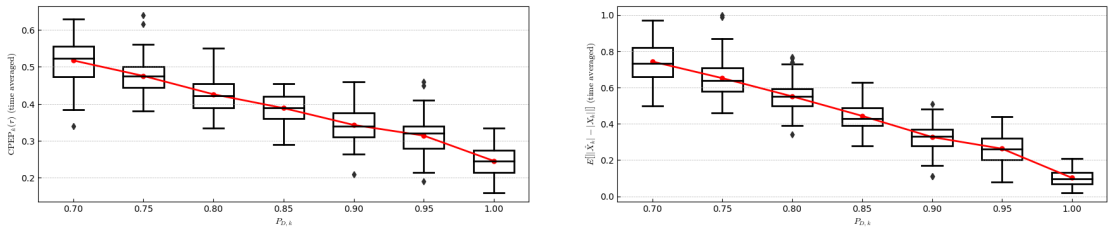


Figure 4.3 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C1) scenario for different values of the detection probability $P_{D,k}$. Every box is the representation of the distribution of 100 independent samples. As the detection probability increases, the estimation error decreases, which may be observed on the decreasing trend of both metrics. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

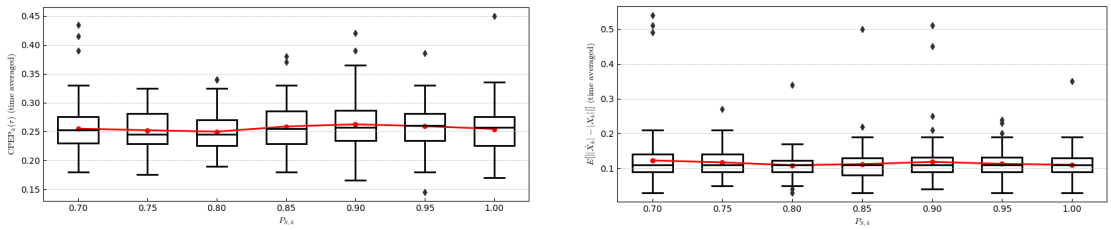


Figure 4.4 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C1) scenario for different values of the survival probability $P_{S,k}$. Every box is the representation of the distribution of 100 independent samples. Neither of metrics shows a change when the survival probability changes. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $\tau = 10^{-5}$ and $U = 4$.

terms of the number of targets and twice better results for the state estimation. This suggests that the GM-PHD filter may show poor results for higher clutter rates and lower detection probabilities.

The third parameter we will examine is the probability of survival, P_S. In Figure 4.4, we see that the performance is not affected by different values of the survival probability. The result suggests that even though the intensity of existing targets will rapidly decrease at every time step with a lower value of $P_{S,k}$, the tracks will still remain confirmed when the target generates enough measurements.

The next variable parameter is the truncation threshold τ. Figure 4.5 suggests that pruning too many components due to higher values of τ reduces the performance of the GM-PHD filter. For instance, $\tau = 10^{-3}$ caused a small decrease in both the precision and the estimate of the number of targets. The explanation of this effect is fairly straightforward. When the truncation threshold is too large, the number of hypotheses is drastically reduced at every time step and the loss of information in the posterior intensity is therefore not negligible.

The final parameter that was tested is the merge threshold U, or the maximum Mahalanobis distance between components on which these components will be merged into one. The results for different values of U are shown in Figure 4.6. The increase in the distance between components causes the same effect as the increase of the truncation threshold τ, that is, the loss of information from the posterior intensity. However, for this specific case and tested values of the parameter, the performance in both metrics worsens only slightly. The exact choice of the merge threshold U is, however, domain-dependent.

Finally, let us visually examine how trajectories are estimated and the final state of the posterior distribution. In Figure 4.7, the left image illustrates the error in trajectory estimation. The track with the tag 667 was terminated when two targets' paths cross each other, and the

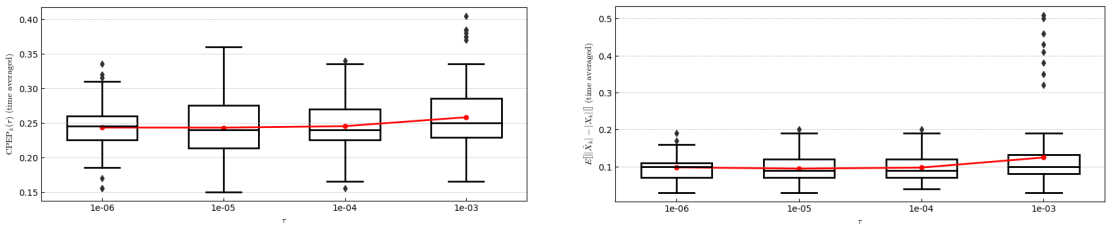


Figure 4.5 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C1) scenario for different values of the truncation threshold τ. Every box is the representation of the distribution of 100 independent samples. The X axis is on the logarithmic scale. We see, that, for higher values of τ, the performance becomes worse. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$ and $U = 4$.

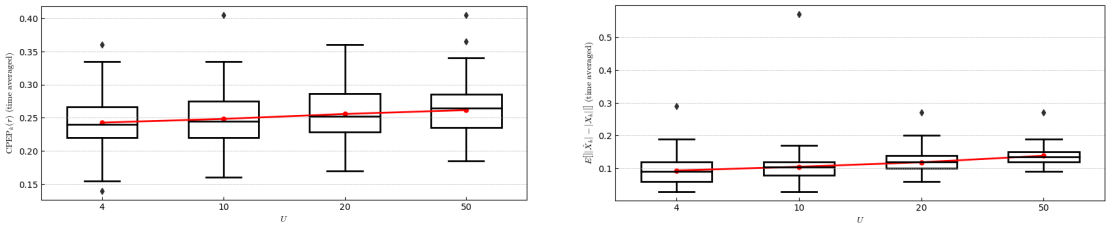


Figure 4.6 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C1) scenario for different values of the merge threshold U. Every box is the representation of the distribution of 100 independent samples. The X axis does not have any scale. We see that, for higher values of U, the performance slightly worsens. All other settings are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, and $\tau = 10^{-5}$.

other track is created right after the intersection with a new tag 854. We will later see how the effect of this problem worsens for more complex scenarios. However, we observe great results in terms of state estimation for both targets. The right image shows the internal state of the GM-PHD filter at time step $k = 100$, more precisely the posterior intensity. It can be seen that two peaks are have expected value in the locations of two objects.

4.2 Test results (C2)

The second test scenario contains six independent objects that are born and die at different times. We have already discussed the influence of changing values over various parameters and will not include all the plots here, since every metric shows the exact same trend for all parameter values except for higher magnitudes of errors due to the increased number of objects. The reader can, however, find all figures in Appendix A. In this section, we will compare two cases of trajectory estimation to point out the track continuity problem of the chosen tagging strategy.

First of all, let us examine the results of target state estimation of one sample, a worst-case scenario. In Figure 4.8, we see how tracks, measurements, and clutter are generated along with state estimates depicted for two axes on the right side of the figure. We clearly see that for this exact sample there are many ”holes” where the estimates were not suggested by the filter. One of the reasons for this behavior is false positive measurements that appear in close proximity to the real targets, which causes the filter to underestimate the likelihood of hypotheses of real target-to-measurement assignments.

The described behavior causes the track continuity problem already mentioned in the previous section. Figure 4.9 compares two different runs on two different data samples. On the left picture, we see that there are six trajectory estimates for six targets, thus the estimate is correct. On the

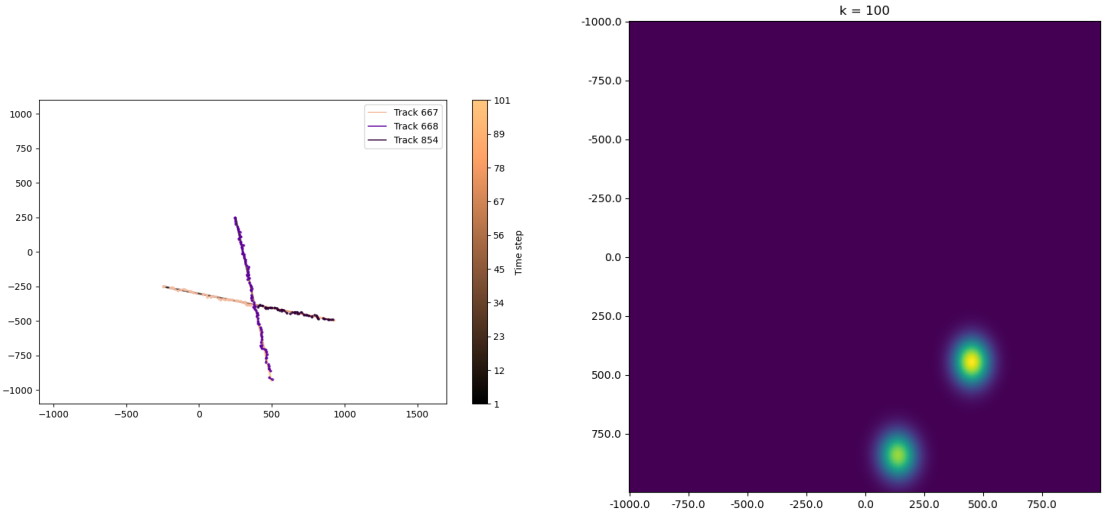


Figure 4.7 Left: The visualization of trajectories estimates and the posterior distribution at time $k = 100$. The straight gradient lines represent true tracks, the points represent state estimates generated by the GM-PHD filter, and different colors of the points illustrate unique tags of the estimated trajectories. Note, that when two tracks intersect, one of estimated tracks was falsely terminated and a new one created. Right: The posterior distribution at time step $k = 100$. For visualization purposes, the covariance matrices of all Gaussian components were multiplied by the factor of 81. Two Gaussian terms are visible on locations where two targets were at this time step. The bottom target has a lower weight, thus its color is dim. The filter was run with default settings, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$, and $U = 4$.

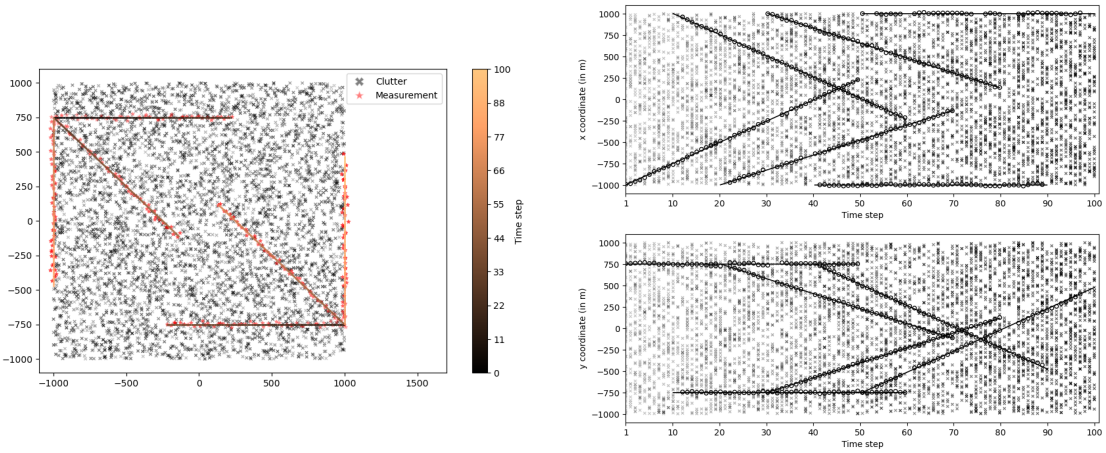


Figure 4.8 One sample of data and estimates for the (C2) scenario. Left: True tracks of two objects (black to yellow) with clutter measurements (gray crosses) and received measurements (red stars) for a single Monte Carlo sample. Right: Change of both coordinates in time with noise measurements (red circles) and filter estimates (black circles) for the same Monte Carlo sample. In comparison to the (C1) scenario, the number of missed estimates is increased.

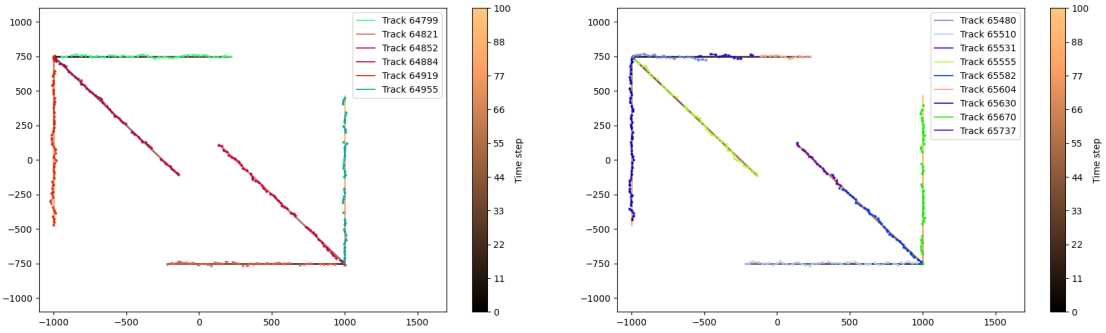


Figure 4.9 Left: The visualization of trajectory estimates at time $k = 100$ for the ideal case. For six targets, there are six different trajectories with good state estimates. Right: The same test scenario with wrong track estimates. We see that there are nine estimated trajectories, and the track continuity is not maintained. The filter was run with default settings, i.e., $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$, and $U = 4$.

right image, however, the number of trajectories is higher, because tracks become terminated and new tracks are initialized. This is caused by the logic of tagging Gaussian components during merging. During this step of the GM-PHD filter, the component with the highest likelihood is chosen as the parent component, and other Gaussian terms are merged into the chosen one. If some new hypothesis created for a clutter measurement happens to have a higher weight, the Gaussian that had the initial tag is merged into this wrong component, and the correct tag disappears from the posterior intensity. The uncertainty in the posterior intensity is also shown in Figure 4.10. The best-case scenario is illustrated on the left. At time $k = 100$, there is one target present in the scene, and the posterior intensity contains only one significant component. However, the bad scenario has a higher uncertainty in estimating the posterior PHD function, and we see several blobs in the image.

4.3 Test results (C3)

The third case depicts the same scenario as (C2), however, with two additional targets that appear at unexpected locations. Figure 4.11 shows that the GM-PHD filter fails to estimate states of these two additional targets. On the right side of this figure, we see that there are no black circles for two lines on both the X axis, and the Y axis. The figures illustrating the change in metrics over different values of filter parameters is located in Appendix A.

If we look at Figure 4.12, we will clearly see, that two targets starting from the bottom-left corner and the top-right corner were not detected by the filter. The track continuity problem persists. It should be noted, that for the demonstration purposes, we intentionally choose the sample where the track continuity problem is present and visible. Generally, the filter performs well, and the expected absolute error on the number of targets should be compared using the average value over all samples.

4.4 Test results (C4)

Finally, we evaluate tracking results for the similar case as (C3) but with external information fusion. The tracking scenario with estimates generated by the GM-PHD filter is illustrated in Figure 4.13. We can see that the filter now detects new targets and the estimates are precise even though the initial mean vectors and covariance matrices are not precise.

In Figure 4.14, we see the same track continuity problem, where the filter created nine

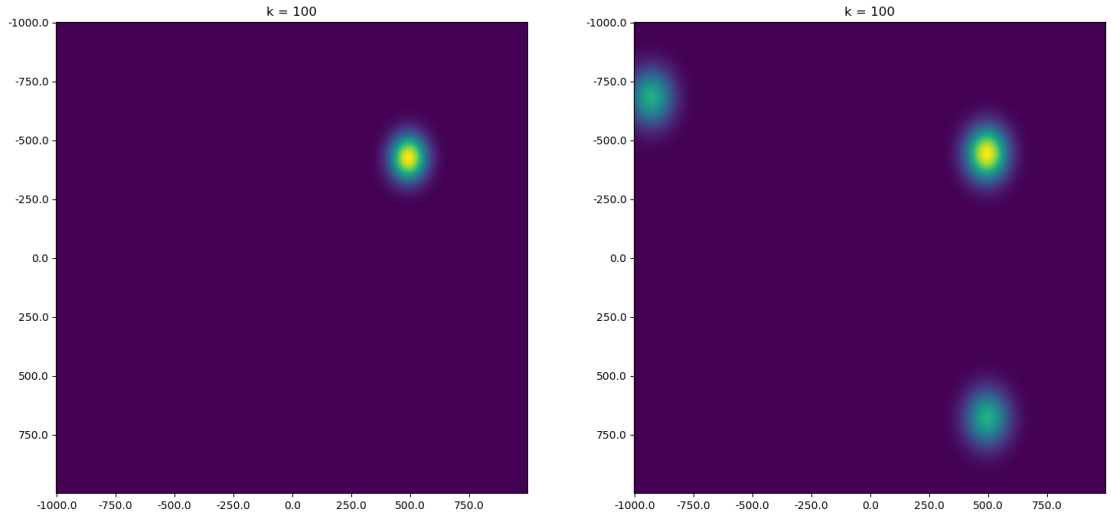


Figure 4.10 Left: The visualization of posterior intensity at time $k = 100$. The intensity has only one Gaussian component with a high weight, which corresponds to the location of the only target present in the scene. Right: The posterior distribution at time step $k = 100$ for a different sample of data. The uncertainty is higher, and there are three visible blobs for one existing target. Note that, for visualization purposes, the covariance matrices of all Gaussian components were multiplied by a factor of 81. The filter was run with default settings, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$, and $U = 4$.

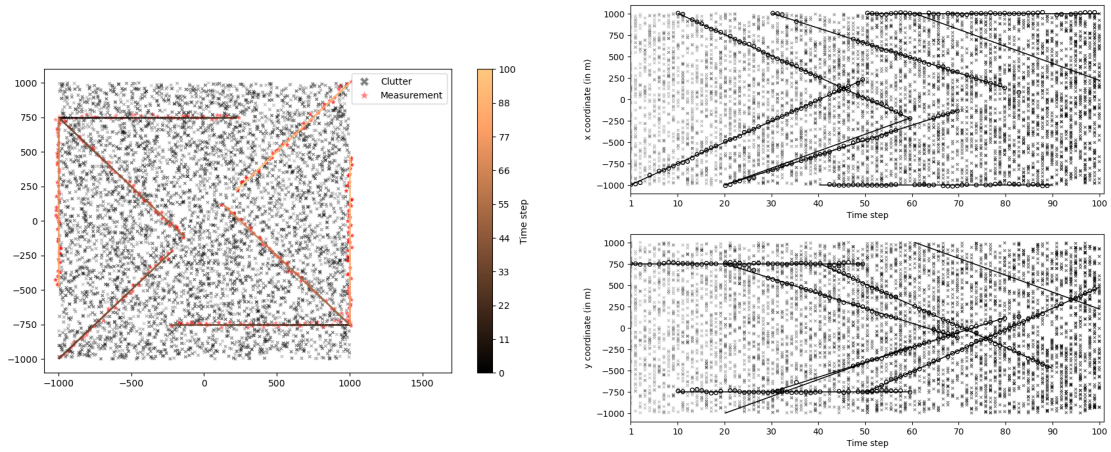


Figure 4.11 One sample of data and estimates for the (C3) scenario. Left: True tracks of two objects (black to yellow) with clutter measurements (gray crosses) and received measurements (red stars) for a single Monte Carlo sample. Right: Change of both coordinates in time with noise measurements (red circles) and filter estimates (black circles) for the same Monte Carlo sample. Two targets were not detected by the filter, and no estimates were generated for them.

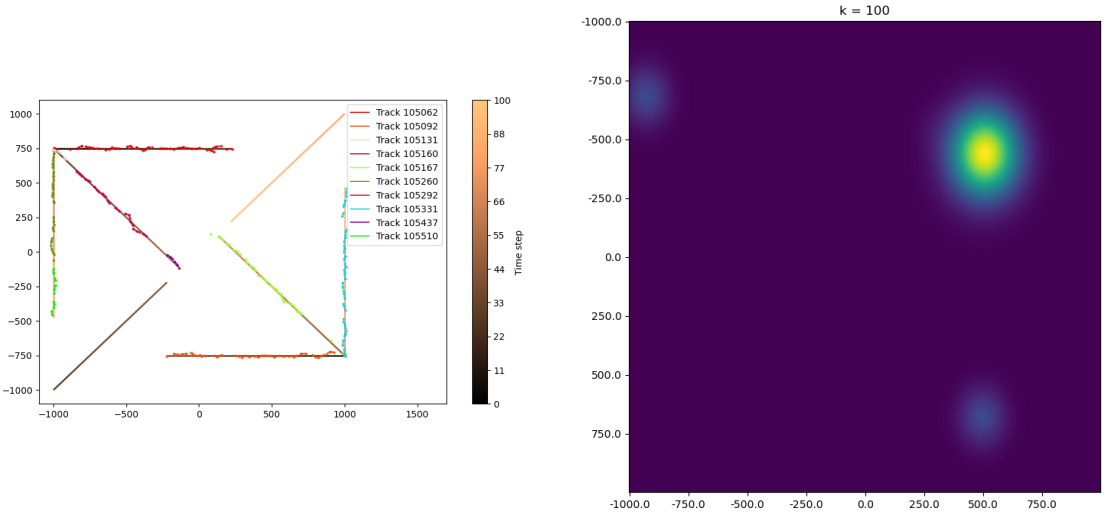


Figure 4.12 Left: The visualization of trajectories estimates and the posterior distribution at time $k = 100$. Two targets were not detected. Right: The posterior distribution at time step $k = 100$. For visualization purposes, the covariance matrices of all Gaussian components were multiplied by the factor of 81. The filter was run with default settings, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$, and $U = 4$.

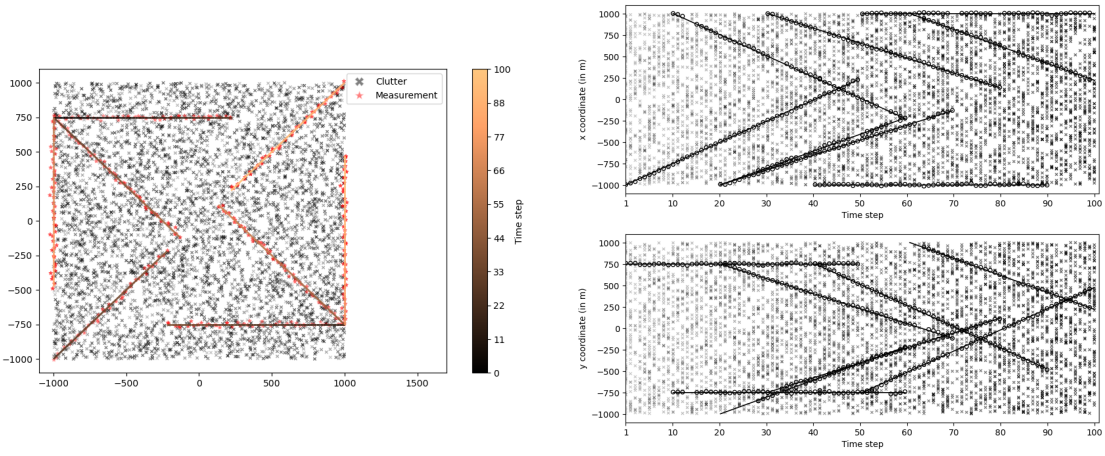


Figure 4.13 One sample of data and estimates for the (C4) scenario. Left: True tracks of two objects (black to yellow) with clutter measurements (gray crosses) and received measurements (red stars) for a single Monte Carlo sample. Right: Change of both coordinates in time with noise measurements (red circles) and filter estimates (black circles) for the same Monte Carlo sample. Two targets were not detected by the filter, and no estimates were generated for them.

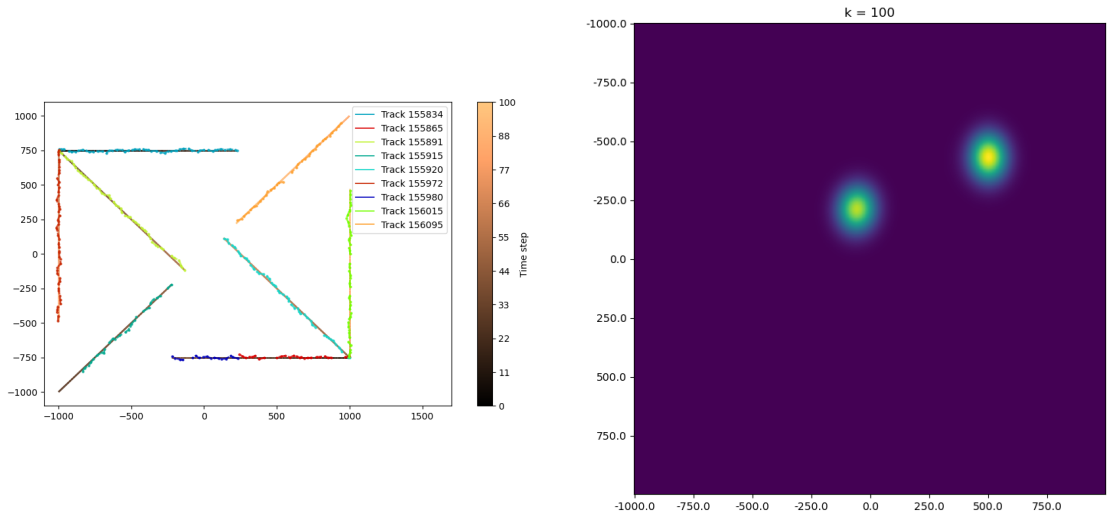


Figure 4.14 Left: The visualization of trajectories estimates and the posterior distribution at time $k = 100$. In comparison to the (C3) scenario, two targets are now detected correctly. Right: The posterior distribution at time step $k = 100$. We see two Gaussian components that refer to two existing objects. For visualization purposes, the covariance matrices of all Gaussian components were multiplied by the factor of 81. The filter was run with default settings, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$, and $U = 4$.

	CPEP	EAE
(C3)	0.87876	0.85900
(C4)	**0.85900**	**0.35380**

Table 4.1 The comparison of CPEP and the expected absolute error on the number of targets (EAE) for the scenarios (C3) and (C4). The numbers represent the averaged values of metrics over 100 Monte Carlo samples. The filter was run with default settings, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$, and $U = 4$.

trajectories, while there are only eight targets. However, in comparison to the (C3) scenario, the objects are now detected, and their tracks are estimated correctly. Despite trajectory estimates being present, note that the filter required some time to start detecting the the bottom-left target. It is due to its large uncertainty and the small weight, their exact values are given in Section 3.2.4. Thus, the GM-PHD algorithm required several additional measurements to confirm the track.

The visualization of all metrics for different filter settings is included in Appendix A. Here, we compare the evaluation of these metrics for default parameters in scenarios (C3) and (C4), as shown in Table 4.1.

4.5 Discussion

As we have presented the results of performance measurement, let us discuss them and draw several conclusions. In general, the GM-PHD filter is a robust algorithm. However, like any other algorithm, it has its drawbacks.

First of all, it is obvious that the filter is sensitive to the quality of data coming from a sensor. For instance, if the sensor has a low detection probability and high clutter rates, track-

ing performance may suffer from the uncertainty on both the state estimate precision and the estimated number of targets. Moreover, the filter becomes sensitive to the setting of the survival probability. For these cases, it is recommended to either use additional sources of information or use another tracking algorithm.

Secondly, let us discuss track maintenance. Generally, the GM-PHD filter has an in-built implicit track maintenance. When a Gaussian has a sufficient weight, in our case greater than 0.5, the track is considered initialized and confirmed. However, there is a problem. If several clutter measurements happen to be in the vicinity of birth components, the likelihood of these components getting enough weights is quite high. This will lead to existence of tracks with only a few estimates, often only one. This problem may be addressed by increasing the minimum weight required for generating an estimate or by introducing some additional logic, such as the minimum number of estimates, something like the M/N logic discussed in Section 2.3.3.1.

The next problem is the consequence of the simple strategy we have chosen for tagging Gaussian components to connect estimates into trajectories. We have seen that it suffers from the track continuity problem, which is discussed in detail along with the reason in Section 4.2. We may try to address this issue by introducing more sophisticated techniques of track maintenance. For instance, we could track the history of track estimates for each tag, and if the correct tag would have been discarded in favor of a tag with an empty history, we would detect it and assign the correct tag to preserve the track continuity. The other option would be to use clustering techniques like in the JPDA filter.

Next, we have observed the change of performance when there are no fusion components. They are fundamentally similar to birth components, but we have shown how they could be used for inputting additional external information into the filter. For blind zones, there is no way to append information using the standard techniques. The filter is unable to capture targets that are invisible to a sensor, and the way how the posterior intensity is represented in the GM-PHD filter makes it impossible to confirm tracks since there are no Gaussian terms to which new measurements can be assigned. We could have defined the birth components that cover the whole field of view, however, it would lead to the problem discussed in the previous paragraph, i.e., too many false estimates because of confirming birth components with clutter measurements.

Last but not least, it is obvious that the complexity $\mathcal{O}(mn)$ discussed in Section 2.3 assumes that the computational burden of the GM-PHD filter is highly dependent on the clutter rate. If there are too many false positive measurements at each time step, the performance in terms of processing time degrades. This is another reason why it is required to reduce the clutter rate as much as possible.

Despite all the drawbacks we have discussed, the GM-PHD filter remains one of the most popular and robust solutions to multi-target tracking. The filter is fast and reliable; it does not need to estimate the full posterior distribution, which is often complex. For Gaussian-linear cases, the GM-PHD filter has a closed form solution, and non-linear models can be estimated using the UK-PHD or the EK-PHD filter.

Chapter 5
Conclusion

This work explored one of the most popular Bayesian filters for multi-target tracking in Gaussian-linear cases, the Gaussian Mixture Probability Density (GM-PHD) filter. We investigated its performance and robustness in several tracking scenarios, including challenging situations where a sensor failed to collect measurements from targets and external information fusion was required.

We have demonstrated that by incorporating external estimates from other sensors, we can improve the performance of the GM-PHD filter, as shown by evaluating several metrics that indicated significant improvement in tracking results.

However, we have also identified a problem with track continuity when using the simple tagging of Gaussian components. More sophisticated techniques may improve track maintenance of the GM-PHD filter and make it more robust to complex scenarios with high uncertainty due to clutter measurements and a high number of targets.

Additionally, we measured and discussed the filter performance depending on the setup of internal parameters. We observed that the GM-PHD filter was sensitive to the choice of settings, particularly the clutter rate and probability of detection, highlighting the need to consider environmental conditions and the quality of the sensor when using the filter.

In conclusion, the GM-PHD filter remains a powerful and robust algorithm for the state estimation of multiple targets in complex scenarios. Its computational complexity is outstanding compared to traditional multi-target tracking algorithms. The propagation of only the posterior first-order moment may be sufficient to achieve good tracking results. The techniques introduced in this work and the comprehensive performance measurements may aid the multi-tracking research community in further improving the GM-PHD filter.

Appendix A

Appendix 1. Results comparison across different filter parameters

In this appendix, we present all testing results for the scenarios described in the main text. The results for each testing case are similar to those presented in Section 4.1; however, the reader may find it helpful to review the measurements for other test cases. Figures A.1, A.2, and A.3 show the change of metrics depending on the change of the clutter spatial rate λ_c for scenarios (C2), (C3), and (C4), respectively. Figures A.4, A.5, and A.6 depict the change of metrics for different values of the detection probability $P_{D,k}$ for each individual case in the same order. Figures A.7, A.8, and A.9 show the change for the survival probability $P_{S,k}$, Figures A.10, A.11, and A.12 for the truncation threshold τ, and finally, Figures A.13, A.14, and A.15 for the merge threshold U for scenarios (C2), (C3), and (C4), respectively. All figures are ordered in a way in which metrics for one parameter are grouped together so that the reader can compare results for different test cases. Every figure is provided with a detailed description of what it depicts.

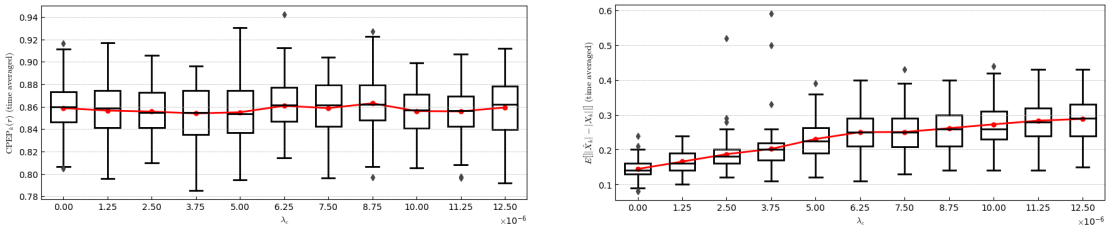


Figure A.1 Here, we examine the change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C2) scenario for different values of the clutter spatial rate λ_c. Every box is the representation of the distribution of 100 independent samples. As the clutter spatial rate increases, the estimation error also increases, which may be observed on the growing trend of both metrics. All other setting are set to default values, i.e. $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

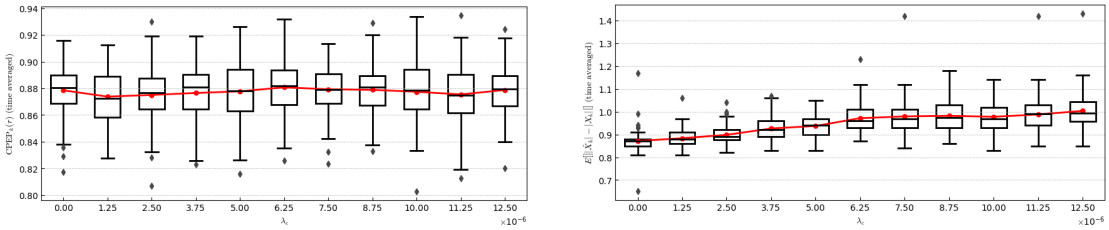


Figure A.2 Here, we examine the change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C3) scenario for different values of the clutter spatial rate λ_c. Every box is the representation of the distribution of 100 independent samples. As the clutter spatial rate increases, the estimation error also increases, which may be observed on the growing trend of both metrics. All other setting are set to default values, i.e. $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

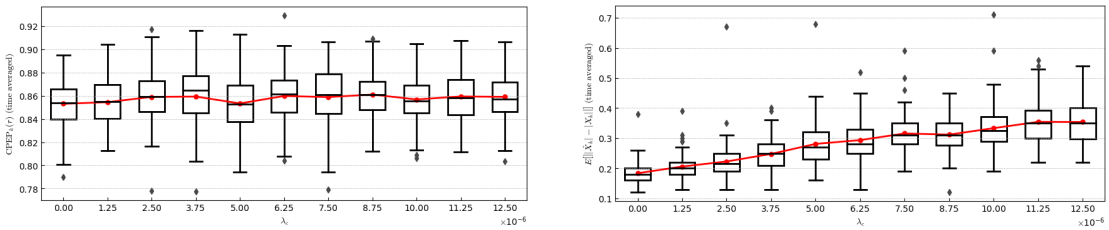


Figure A.3 Here, we examine the change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C4) scenario for different values of the clutter spatial rate λ_c. Every box is the representation of the distribution of 100 independent samples. As the clutter spatial rate increases, the estimation error also increases, which may be observed on the growing trend of both metrics. All other setting are set to default values, i.e. $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

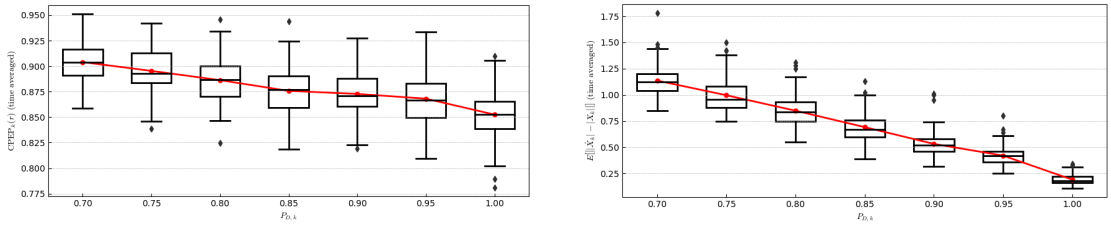


Figure A.4 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C2) scenario for different values of the detection probability $P_{D,k}$. Every box is the representation of the distribution of 100 independent samples. As the detection probability increases, the estimation error decreases, which may be observed on the decreasing trend of both metrics. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

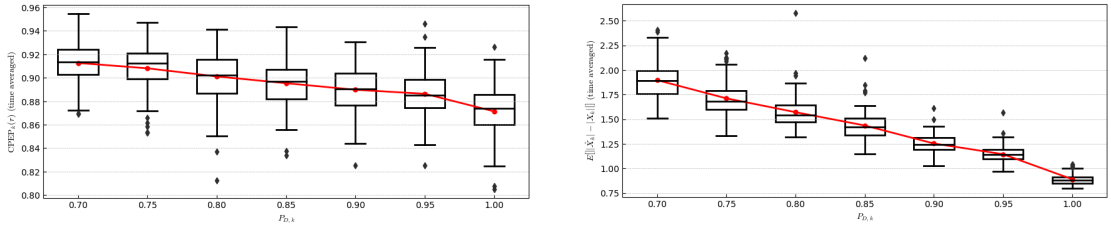


Figure A.5 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C3) scenario for different values of the detection probability $P_{D,k}$. Every box is the representation of the distribution of 100 independent samples. As the detection probability increases, the estimation error decreases, which may be observed on the decreasing trend of both metrics. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

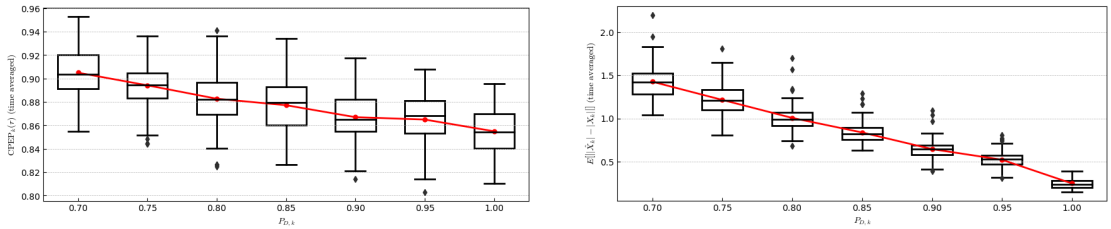


Figure A.6 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C4) scenario for different values of the detection probability $P_{D,k}$. Every box is the representation of the distribution of 100 independent samples. As the detection probability increases, the estimation error decreases, which may be observed on the decreasing trend of both metrics. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{S,k} = 0.99$, $\tau = 10^{-5}$ and $U = 4$.

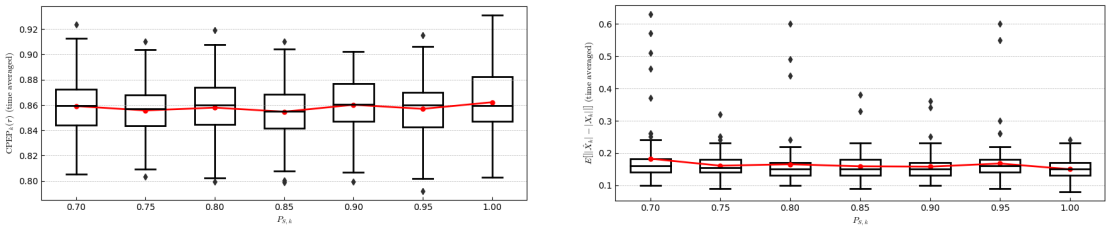


Figure A.7 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C2) scenario for different values of the survival probability $P_{S,k}$. Every box is the representation of the distribution of 100 independent samples. Neither of metrics shows a change when the survival probability changes. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $\tau = 10^{-5}$ and $U = 4$.

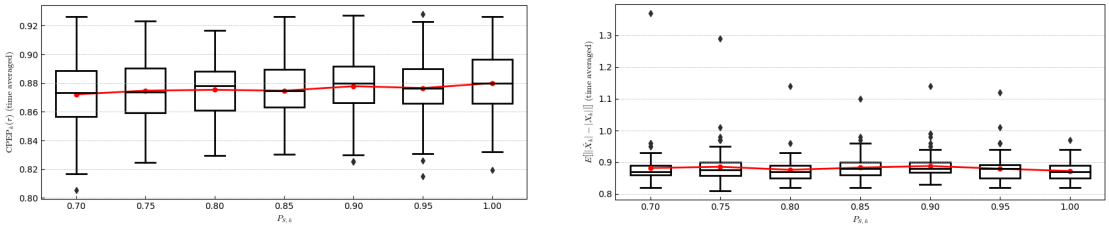


Figure A.8 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C3) scenario for different values of the survival probability $P_{S,k}$. Every box is the representation of the distribution of 100 independent samples. Neither of metrics shows a change when the survival probability changes. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $\tau = 10^{-5}$ and $U = 4$.

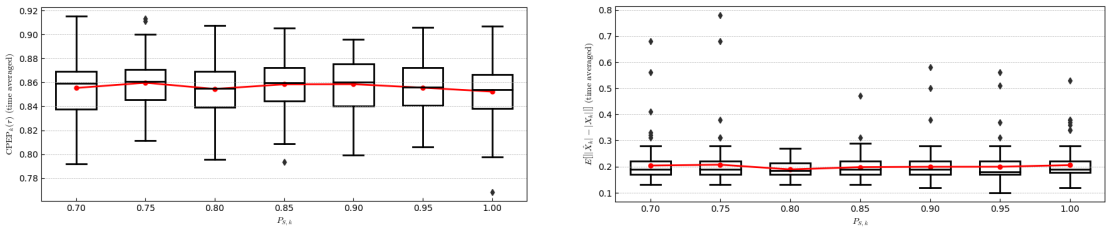


Figure A.9 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C4) scenario for different values of the survival probability $P_{S,k}$. Every box is the representation of the distribution of 100 independent samples. Neither of metrics shows a change when the survival probability changes. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $\tau = 10^{-5}$ and $U = 4$.

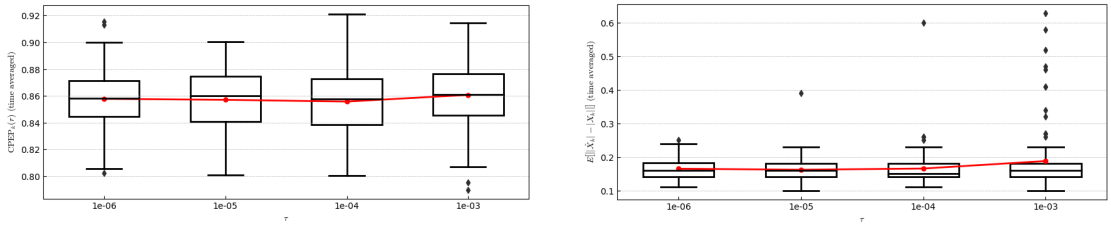


Figure A.10 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C2) scenario for different values of the truncation threshold τ. Every box is the representation of the distribution of 100 independent samples. The X axis is on the logarithmic scale. We see, that, for higher values of τ, the performance becomes worse. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$ and $U = 4$.

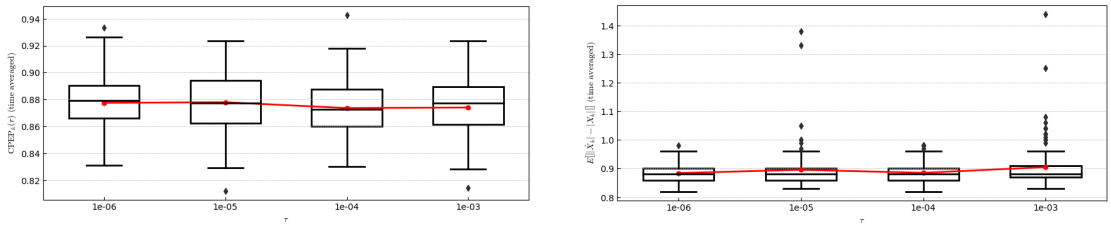


Figure A.11 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C3) scenario for different values of the truncation threshold τ. Every box is the representation of the distribution of 100 independent samples. The X axis is on the logarithmic scale. We see, that, for higher values of τ, the performance becomes worse. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$ and $U = 4$.

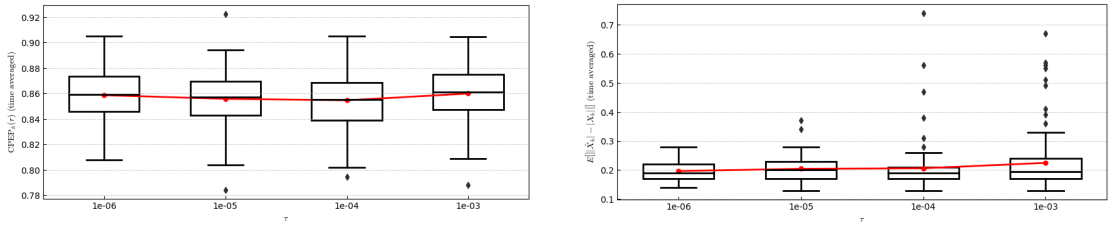


Figure A.12 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C4) scenario for different values of the truncation threshold τ. Every box is the representation of the distribution of 100 independent samples. The X axis is on the logarithmic scale. We see, that, for higher values of τ, the performance becomes worse. All other setting are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$ and $U = 4$.

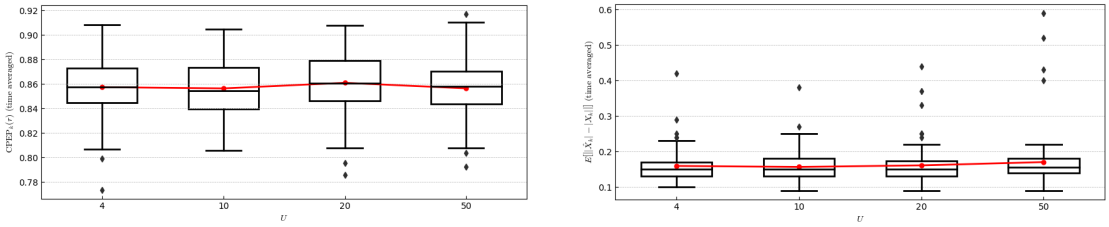


Figure A.13 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C2) scenario for different values of the merge threshold U. Every box is the representation of the distribution of 100 independent samples. The X axis does not have any scale. We see that, for higher values of U, the performance slightly worsens. All other settings are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, and $\tau = 10^{-5}$.

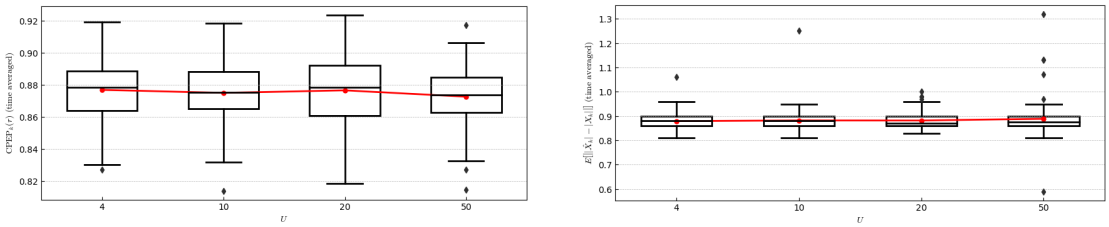


Figure A.14 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C3) scenario for different values of the merge threshold U. Every box is the representation of the distribution of 100 independent samples. The X axis does not have any scale. We see that, for higher values of U, the performance slightly worsens. All other settings are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, and $\tau = 10^{-5}$.

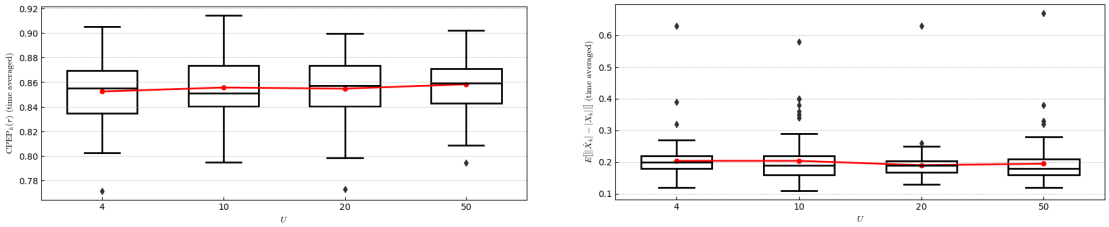


Figure A.15 The change of CPEP (left) and the expected absolute error on the number of targets (right) for the (C4) scenario for different values of the merge threshold U. Every box is the representation of the distribution of 100 independent samples. The X axis does not have any scale. We see that, for higher values of U, the performance slightly worsens. All other settings are set to default values, i.e. $\lambda_c = 12.5 \times 10^{-6}$, $P_{D,k} = 0.98$, $P_{S,k} = 0.99$, and $\tau = 10^{-5}$.

Bibliography

1. SÄRKKÄ, Simo. *Bayesian Filtering and Smoothing* [online]. 1st ed. Cambridge University Press, 2013 [visited on 2023-04-29]. ISBN 978-1-107-03065-7. Available from DOI: 10.1017/CBO9781139344203.
2. GRANSTROM, Karl; BAUM, Marcus; REUTER, Stephan. *Extended Object Tracking: Introduction, Overview and Applications* [online]. 2017-02-21. [visited on 2023-04-29]. Available from arXiv: 1604.00970 [cs, eess].
3. SALMOND, D.J.; GORDON, N.J. Group and Extended Object Tracking. In: *IEE Colloquium on Target Tracking: Algorithms and Applications (Ref. No. 1999/090, 1999/215)*. 1999, pp. 16/1–16/4. Available from DOI: 10.1049/ic:19990517.
4. BAR-SHALOM, Y.; KUMAR, A.; BLAIR, W.D.; GROVES, G.W. Tracking Low Elevation Targets in the Presence of Multipath Propagation. *IEEE Transactions on Aerospace and Electronic Systems*. 1994, vol. 30, no. 3, pp. 973–979. ISSN 1557-9603. Available from DOI: 10.1109/7.303775.
5. ANGLE, R. Blair; STREIT, Roy L.; EFE, Murat. Multiple Target Tracking With Unresolved Measurements. *IEEE Signal Processing Letters*. 2021, vol. 28, pp. 319–323. ISSN 1558-2361. Available from DOI: 10.1109/LSP.2021.3051858.
6. ZWILLINGER, Daniel; KOKOSKA, Stephen. *CRC Standard Probability and Statistics Tables and Formulae*. Boca Raton: Chapman & Hall/CRC, 2000. ISBN 978-1-58488-059-2.
7. GARCÍA-FERNÁNDEZ, Ángel F.; WILLIAMS, Jason L.; GRANSTRÖM, Karl; SVENSSON, Lennart. Poisson Multi-Bernoulli Mixture Filter: Direct Derivation and Implementation. *IEEE Transactions on Aerospace and Electronic Systems*. 2018, vol. 54, no. 4, pp. 1883–1901. ISSN 1557-9603. Available from DOI: 10.1109/TAES.2018.2805153.
8. BAR-SHALOM, Yaakov; LI, X.-Rong; KIRUBARAJAN, Thiagalingam. *Estimation with Applications to Tracking and Navigation* [online]. New York, USA: John Wiley & Sons, Inc., 2001 [visited on 2023-04-07]. ISBN 978-0-471-41655-5. Available from DOI: 10.1002/0471221279.
9. JOHNSON, Richard A.; WICHERN, Dean W. *Applied Multivariate Statistical Analysis*. 6th ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2007. ISBN 978-0-13-187715-3.
10. HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, J. H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction: With 200 Full-Color Illustrations*. New York: Springer, 2001. Springer Series in Statistics. ISBN 978-0-387-95284-0.

11. BULUT, Yalcin; VINES-CAVANAUGH, D.; BERNAL, Dionisio. Process and Measurement Noise Estimation for Kalman Filtering. In: PROULX, Tom (ed.). *Structural Dynamics, Volume 3*. New York, NY: Springer, 2011, pp. 375–386. Conference Proceedings of the Society for Experimental Mechanics Series. ISBN 978-1-4419-9834-7. Available from DOI: 10.1007/978-1-4419-9834-7_36.
12. KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* [online]. 1960, vol. 82, no. 1, pp. 35–45 [visited on 2023-04-11]. ISSN 0021-9223. Available from DOI: 10.1115/1.3662552.
13. GREWAL, Mohinder; ANDREWS, Angus. Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]. *Control Systems, IEEE*. 2010, vol. 30, pp. 69–78. Available from DOI: 10.1109/MCS.2010.936465.
14. GE, Quanbo; SHAO, Teng; DUAN, Zhansheng; WEN, Chenglin. Performance Analysis of the Kalman Filter With Mismatched Noise Covariances. *IEEE Transactions on Automatic Control*. 2016, vol. 61, pp. 1–1. Available from DOI: 10.1109/TAC.2016.2535158.
15. MATISKO, Peter; HAVLENA, Vladimír. Noise Covariances Estimation for Kalman Filter Tuning. *IFAC Proceedings Volumes* [online]. 2010, vol. 43, no. 10, pp. 31–36 [visited on 2023-04-11]. ISSN 1474-6670. Available from DOI: 10.3182/20100826-3-TR-4015.00009.
16. YUEN, Ka-Veng; LIANG, Peng-Fei; KUOK, Sin-Chi. Online Estimation of Noise Parameters for Kalman Filter. *Structural Engineering and Mechanics*. 2013, vol. 47. Available from DOI: 10.12989/sem.2013.47.3.361.
17. WAN, E.A.; VAN DER MERWE, R. The Unscented Kalman Filter for Nonlinear Estimation. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)* [online]. Lake Louise, Alta., Canada: IEEE, 2000, pp. 153–158 [visited on 2023-04-11]. ISBN 978-0-7803-5800-3. Available from DOI: 10.1109/ASSPCC.2000.882463.
18. SMITH, Gerald L.; SCHMIDT, Stanley F.; MCGEE, Leonard A. *Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity on Board a Circumlunar Vehicle* [online]. 1962-01-01. [visited on 2023-04-11]. NASA-TR-R-135. Available from: https://ntrs.nasa.gov/citations/20190002215.
19. MAHLER, Ronald P. S. *Statistical Multisource-Multitarget Information Fusion*. Boston: Artech House, 2007. Artech House Information Warfare Library. ISBN 978-1-59693-092-6.
20. RISTIC, Branko; ARULAMPALAM, Sanjeev; GORDON, Neil. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Boston, MA: Artech House, 2004. Artech House Radar Library. ISBN 978-1-58053-631-8.
21. TOKLE, Lars-Christian Ness. *Multi Target Tracking - Using Random Finite Sets with a Hybrid State Space and Approximations* [online]. 2018. [visited on 2023-04-12]. Available from: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2575375. MA thesis. NTNU.
22. HUMPHERYS, Jeffrey; REDD, Preston; WEST, Jeremy. A Fresh Look at the Kalman Filter. *SIAM Review* [online]. 2012, vol. 54, no. 4, pp. 801–823 [visited on 2023-04-13]. ISSN 0036-1445, ISSN 1095-7200. Available from DOI: 10.1137/100799666.
23. BAR-SHALOM, Yaakov; DAUM, Fred; HUANG, Jim. The Probabilistic Data Association Filter. *IEEE Control Systems Magazine*. 2009, vol. 29, no. 6, pp. 82–100. ISSN 1941-000X. Available from DOI: 10.1109/MCS.2009.934469.
24. BAR-SHALOM, Yaakov; LI, Xiao-Rong. *Multitarget-Multisensor Tracking: Principles and Techniques*. 3rd printing. Storrs, Conn: YBS, 1995. ISBN 978-0-9648312-0-9.
25. REID, D. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*. 1979, vol. 24, no. 6, pp. 843–854. ISSN 1558-2523. Available from DOI: 10.1109/TAC.1979.1102177.

26. COX, I.J.; HINGORANI, S.L. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1996, vol. 18, no. 2, pp. 138–150. ISSN 1939-3539. Available from DOI: 10.1109/34.481539.

27. WERTHMANN, John R. Step-by-Step Description of a Computationally Efficient Version of Multiple Hypothesis Tracking. In: DRUMMOND, Oliver E. (ed.) [online]. Orlando, FL, 1992, pp. 288–300 [visited on 2023-04-17]. Available from DOI: 10.1117/12.139379.

28. VO, Ba-Ngu; MALLICK, Mahendra; BAR-SHALOM, Yaakov; CORALUPPI, Stefano; OSBORNE, Richard; MAHLER, Ronald; VO, Ba-Tuong. Multitarget Tracking. In: WEBSTER, John G. *Wiley Encyclopedia of Electrical and Electronics Engineering* [online]. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2015, pp. 1–15 [visited on 2023-04-17]. ISBN 978-0-471-34608-1. Available from DOI: 10.1002/047134608X.W8275.

29. CHALLA, Subhash; MORELANDE, Mark R.; MUŠICKI, Darko; EVANS, Robin J. *Fundamentals of Object Tracking* [online]. 1st ed. Cambridge University Press, 2011 [visited on 2023-04-18]. ISBN 978-0-521-87628-5. Available from DOI: 10.1017/CBO9780511975837.

30. MAHLER, R.P.S. Multitarget Bayes Filtering via First-Order Multitarget Moments. *IEEE Transactions on Aerospace and Electronic Systems*. 2003, vol. 39, no. 4, pp. 1152–1178. ISSN 1557-9603. Available from DOI: 10.1109/TAES.2003.1261119.

31. STREIT, Roy L. *Poisson Point Processes: Imaging, Tracking and Sensing*. New York: Springer, 2010. ISBN 978-1-4419-6922-4. Available also from: https://link.springer.com/book/10.1007/978-1-4419-6923-1.

32. VO, B.-N.; MA, W.-K. The Gaussian Mixture Probability Hypothesis Density Filter. *IEEE Transactions on Signal Processing*. 2006, vol. 54, no. 11, pp. 4091–4104. ISSN 1941-0476. Available from DOI: 10.1109/TSP.2006.881190.

33. VO, Ba-Ngu; SINGH, S.; DOUCET, A. Sequential Monte Carlo Implementation of the PHD Filter for Multi-Target Tracking. In: *Sixth International Conference of Information Fusion, 2003. Proceedings of The*. 2003, vol. 2, pp. 792–799. Available from DOI: 10.1109/ICIF.2003.177320.

34. CLARK, Daniel E.; PANTA, Kusha; VO, Ba-Ngu. The GM-PHD Filter Multiple Target Tracker. In: *2006 9th International Conference on Information Fusion*. 2006, pp. 1–8. Available from DOI: 10.1109/ICIF.2006.301809.

35. BLACKMAN, Samuel S.; POPOLI, Robert. *Design and Analysis of Modern Tracking Systems*. Boston: Artech House, 1999. Artech House Radar Library. ISBN 978-1-58053-006-4.

36. KROPFREITER, Thomas; MEYER, Florian; CORALUPPI, Stefano; CARTHEL, Craig; MENDRZIK, Rico; WILLETT, Peter. Track Coalescence and Repulsion: MHT, JPDA, and BP. In: *2021 IEEE 24th International Conference on Information Fusion (FUSION)* [online]. Sun City, South Africa: IEEE, 2021, pp. 1–8 [visited on 2023-04-25]. ISBN 978-1-73774-971-4. Available from DOI: 10.23919/FUSION49465.2021.9626958.

37. KLEIN, Lawrence A. *Sensor and Data Fusion: A Tool for Information Assessment and Decision Making*. Bellingham, Wash: SPIE Press, 2004. ISBN 978-0-8194-5435-5.

38. MANGAI, Utthara Gosa; SAMANTA, Suranjana; DAS, Sukhendu; CHOWDHURY, Pinaki Roy. A Survey of Decision Fusion and Feature Fusion Strategies for Pattern Classification. *IETE Technical Review* [online]. 2010, vol. 27, no. 4, p. 293 [visited on 2023-04-24]. ISSN 0256-4602. Available from DOI: 10.4103/0256-4602.64604.

39. WOLFF, Dipl-Ing (FH) Christian. *AN/MPQ-53 - Radartutorial* [online]. Dipl.-Ing. (FH) Christian Wolff [visited on 2023-04-24]. Available from: https://www.radartutorial.eu/19.kartei/06.missile/karte003.en.html.

40. NELSON, William. *Use of Circular Error Probability in Target Detection* [online]. [visited on 2023-04-26]. Available from: `https://apps.dtic.mil/sti/citations/ADA199190`.

Contents of the attached medium

- `readme.txt` ..... a brief description of the contents of the medium
- `src`
  - `impl` ..... the implementation source code
  - `thesis` ..... the source code of the thesis in LaTeX language
- `text` ..... the text of the thesis
  - `thesis.pdf` ..... the text of the thesis in the PDF format