FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	PHD filter-based monitoring of birds migration
Student:	Bc. Tomáš Holas
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 2024/2025

Instructions

Abstract: The rapid development of tiny devices capable of GNSS positioning and communication has led to their intensive use in biological sciences. There, the (more or less online) tracking of wild animals allows to intensively study the behavior of animal in their habitat and during their migration. The goal of the master thesis is to perform a (currently offline) multi-target-based positioning of the selected species using a probability hypothesis density filter in its basic or modified form, allowing for trajectory reconstruction. In particular, the proposed framework should be evaluated on the rodent-specialized birds of prey dataset [1], which contains GPS measurements of 35 individuals of a total of 5 species. This dataset originates from the Research Institute for Nature and Forest (INBO). The data should be limited to a reasonable time window.

Goals:
1) Shortly describe the studied biological problem.
2) Describe the algorithm used for target tracking. If possible, propose its modifications for the problem at hand.
3) Develop a working implementation, discuss its performance and qualities.

References:
[1] Spanoghe G, Janssens K, Klaassen R, Schaub T, Milotic T, Desmet P (2020) BOP_RODENT - Rodent specialized birds of prey (Circus, Asio, Buteo) in Flanders (Belgium).
[2] R. Mahler, Statistical Multisource-Multitarget Information Fusion. Artech house, 2007.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 31 January 2024 in Prague.

[3] A. F. Garcia-Fernandez and L. Svensson, "Trajectory PHD and CPHD Filters," IEEE Trans. Signal Process., vol. 67, no. 22, pp. 5702–5714, Nov. 2019, doi: 10.1109/TSP.2019.2943234.
[4] T. Kropfreiter, F. Meyer, and F. Hlawatsch, "An Efficient Labeled/Unlabeled Random Finite Set Algorithm for Multiobject Tracking," IEEE Trans. Aerosp. Electron. Syst., vol. 58, no. 6, pp. 5256–5275, Dec. 2022, doi: 10.1109/TAES.2022.3168252.

Master's thesis

PHD FILTER-BASED MONITORING OF BIRDS MIGRATION

Bc. Tomáš Holas

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: doc. Ing. Kamil Dedecius, Ph.D.
May 7, 2024

Citation of this thesis: Holas Tomáš. *PHD filter-based monitoring of birds migration.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

List of code listings

I am profoundly thankful for the unwavering support and guidance of my advisor, doc. Ing. Kamil Dedecius, Ph.D, throughout my master's program. Their expertise and patience have been invaluable, playing a pivotal role in the success of this thesis.

Declaration

In Prague on May 7, 2024

Abstract

The swift advancement of miniature devices with GNSS positioning and communication capabilities has spurred their widespread utilization in biological research. Specifically, the real-time or near-real-time tracking of wild animals enables a thorough examination of their behavior within their natural environment and throughout migration patterns. This study focuses on localizing birds cataloged in the Research Institute for Nature and Forest (INBO) database BOP_RODENT. This localization approach employs the probability hypothesis density (PHD) filter in its standard configuration and an adapted version tailored for individual bird tracking. The primary motivations for choosing this filter include its ability to localize multiple targets simultaneously, resilience to external noise and false positives, and independence from prior knowledge of target quantity. The original contributions of this research are twofold: proposing a PHD-based method for bird tracking and enhancing the basic PHD filter to facilitate track formation.

Keywords bird tracking, real-time animal behavior monitoring, multi-target tracking, Probability Hypothesis Density filter, Random Finite Set, trajectories

Abstrakt

Rychlý vývoj miniaturních zařízení s polohovacími a komunikačními schopnostmi GNSS podnítil jejich široké využití v biologickém výzkumu. Konkrétně sledování volně žijících živočichů v reálném čase nebo téměř v reálném čase umožňuje důkladně zkoumat jejich chování v jejich přirozeném prostředí a v průběhu migrace. Tato studie se zaměřuje na lokalizaci ptáků katalogizovaných v databázi BOP_RODENT Research Institute for Nature and Forest (INBO). Tento lokalizační přístup využívá probability hypothesis density (PHD) filtr ve standardní konfiguraci a jeho upravenou verzi přizpůsobenou pro sledování jednotlivých ptáků. Mezi hlavní důvody pro volbu tohoto filtru patří jeho schopnost lokalizovat více cílů současně, odolnost vůči vnějšímu šumu a falešně pozitivním výsledkům a nezávislost na předchozí znalosti množství cílů. Původní přínosy tohoto výzkumu jsou dvojí: návrh metody založené na PHD pro sledování ptáků a vylepšení základního filtru PHD pro usnadnění tvorby trajektorií.

Klíčová slova sledování ptáků, sledování chování zvířat v reálném čase, sledování více cílů, Probability Hypothesis Density filter, náhodná konečná množina, trajektorie

List of abbreviations

PHD	Probability Hypothesis Density
STT	Single Target Tracking
MTT	Multi Target Tracking
KF	Kalman Filter
PDF	Probability Density Function
RFS	Random Finite Set
CVM	Constant Velocity Model
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
GSM	Global System for Mobile Communications
GOSPA	Generalized optimal sub-pattern assignment metric

Introduction

In recent years, the rapid evolution of miniature devices equipped with Global Navigation Satellite System (GNSS) positioning and communication capabilities has revolutionized the field of biological research. These advancements have facilitated the widespread adoption of such technologies, particularly in wildlife tracking and behavioral analysis. By enabling real-time or near-real-time tracking of wild animals, researchers can gain unprecedented insights into their behaviors within their natural habitats and elucidate intricate migration patterns.

This study is centered around the localization of avian species documented in the extensive database maintained by the Research Institute for Nature and Forest (INBO), specifically focusing on birds cataloged within the BOP_RODENT dataset. Leveraging the capabilities of GNSS-enabled devices, we aim to track the movements of individual birds with precision and accuracy. To achieve this goal, we employ a localization approach based on the probability hypothesis density (PHD) filter in its standard configuration and through a tailored adaptation designed explicitly for avian tracking.

The selection of the PHD filter as our primary localization method is motivated by several key advantages it offers. Firstly, the PHD filter is adept at localizing multiple targets simultaneously, making it well-suited for scenarios involving the tracking of numerous individuals within a population of birds. Its inherent resilience to external noise and false positives also ensures robust performance in real-world environments where data integrity may be compromised. Furthermore, the PHD filter operates independently of prior knowledge regarding the number of targets present, providing flexibility and adaptability to dynamic tracking scenarios.

This research makes two significant contributions to the field of avian tracking. Firstly, we propose a novel PHD-based methodology tailored explicitly for the tracking of birds, leveraging the unique characteristics of avian movement patterns and behaviors. Secondly, we enhance the fundamental PHD filter framework to facilitate the formation of coherent bird tracks, thereby improving the accuracy and reliability of localization results.

Chapter 1

Single Target Tracking

This chapter will contain a short theoretical background to single target tracking. We will start with the state-space model, the Constant Velocity Model, the Bayes theorem theory, a description of the Kalman filter, and brief information about clutter. All the informations in this chaper are based on [1, 2, 3, 4, 5, 6, 7, 8].

1.1 State-space model

A state-space model is a mathematical framework used to describe the behavior of a dynamic system over time. It consists of two main components: the state equation and the observation equation. When only considering the discrete space and discrete time, the state-space model with latent variable x_t and observed output y_t can be represented as follows

$$x_t = f(x_{t-1}, w_t), \tag{1.1}$$
$$y_t = g(x_t, v_t), \tag{1.2}$$

where g and f represent known functions and v_t and w_t are independent zero-centered noise variables.

We assume that a state evolves in some way over time. We assume this. It evolves according to the Markov model, that is, its current state at time t depends only on the previous state at time $t-1$ and potentially some other external variable, which will be ignored in the sequel. The state cannot be directly observed. The target state use the hidden process model for their evolution described by Equation (1.1). However, it can be inferred from the observed measurements, which we denote by y_t. These measurements are generated at each time instant t by measurement model described by Equation 1.2. The state evolution is then shown in Figure 1.1.

State in state-space model can be represented as vector. Discrete time instants equally spaced apart are denoted by t = 1, 2, 3, ... An instance of time can be represented by a unit, such as one second. In our case, we will use days as the unit of time. (For more information refer to Chapter 3)

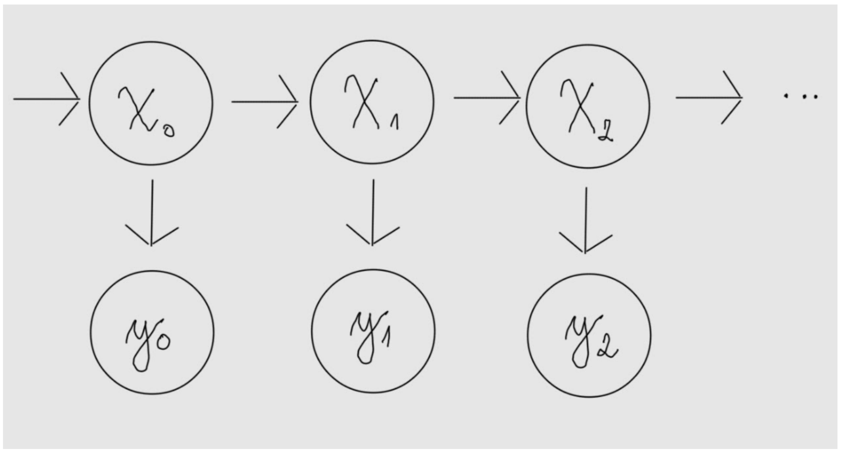


Figure 1.1 The Evolution of the target state and its role in generating the measurement.

As an example of a state-space model, consider the following linear state-space model

$$x_t = A_t x_{t-1} + w_t, \tag{1.3}$$
$$y_t = H_t x_t + v_t, \tag{1.4}$$

where the matrices A_t and H_t are matrices of the appropriate dimension. Furthermore, v_t is iid in time and obeys a Gaussian distribution. Similarly, w_t is iid and it is modeled by a Gaussian distribution.

To simplify our analysis in this thesis, we will assume that our models' matrices A and H are not time-varying. This means that these matrices remain constant at each moment in time. Therefore, we can rewrite our linear model accordingly

$$x_t = A x_{t-1} + w_t, \tag{1.5}$$
$$y_t = H x_t + v_t, \tag{1.6}$$

1.1.1 Constant Velocity Model

The constant velocity model describes the position of the moving target on a N-dimensional space. For our case and other applications, we restrict ourselves to 2-dimensional space. Each object has 2 components $x_{1,t}, x_{2,t}$ that correspond to a position on the 2D surface.For example $x_{1,t}$ corresponds to latitude and $x_{2,t}$ corresponds to longitude. These two components are modeled from the state at time $t-1$ and the corresponding velocity in a given direction $r_{x_1,t}$ and $r_{x_2,t}$. We denote the time difference between t and $t-1$ by Δt. In our case, the velocities will correspond to a random walk. AA random walk is a simple model that allows for the dynamic modeling of many time-varying phenomena without precise knowledge of the evolution model. Formulas for state evolutions are

$$x_{1,t} = x_{1,t-1} + r_{x_1,t}\Delta t + v_{x_1,t}, \tag{1.7}$$
$$x_{2,t} = x_{2,t-1} + r_{x_2,t}\Delta t + v_{x_2,t}, \tag{1.8}$$
$$r_{x_1,t} = r_{x_1,t-1} + v_{r_{x_1},t}, \tag{1.9}$$
$$r_{x_2,t} = r_{x_2,t-1} + v_{r_{x_2},t}. \tag{1.10}$$

Using the matrix notation

$$\underbrace{\begin{bmatrix} x_{1,t} \\ x_{2,t} \\ r_{x_1,t} \\ r_{x_2,t} \end{bmatrix}}_{x_t} = \underbrace{\begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A_t} \underbrace{\begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \\ r_{x_1,t-1} \\ r_{x_2,t-1} \end{bmatrix}}_{x_{t-1}} + \underbrace{\begin{bmatrix} v_{x_1,t} \\ v_{x_2,t} \\ v_{r_{x_1},t} \\ v_{r_{x_2},t} \end{bmatrix}}_{v_t},$$

where

- x_t is the state variable at time t.
- A_t is the state transition matrix.
- x_{t-1} is the state variable at time $t-1$.
- v_t is the process noise covariance matrix.

Assume that there are normaly distributed measurements in the same coordinate space. Let us denote them $y_{1,t}$ and $y_{2,t}$. We can model these measurements as follows

$$y_{1,t} = x_{1,t} + w_{x_1,t}, \tag{1.11}$$

$$y_{2,t} = x_{2,t} + w_{x_2,t}, \tag{1.12}$$

using the matrix notation

$$\underbrace{\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix}}_{y_t} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H_t} \underbrace{\begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix}}_{x_t} + \underbrace{\begin{bmatrix} w_{x_1,t} \\ w_{x_2,t} \end{bmatrix}}_{w_t},$$

where

- y_t is the vector representing measurements for longtitude and latitude t.
- H_t is the observation matrix at time t, which maps the state space to the measurement
- x_t is the state variable at time t.
- w_t is the measurement noise covariance matrix.

1.2 Gaussian distribution

The Gaussian distribution, also known as the normal distribution, is a probability distribution that is symmetric around its mean, forming a bell-shaped curve. It is characterized by two parameters: the mean (μ), which represents the center of the distribution, and the variance (σ^2) which determines the spread or dispersion of the distribution. We define Gaussian distribution of one real variable X is defined as follows

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad \mu \in \mathbb{R}, \sigma > 0.$$

Notation:

$$X \sim \mathcal{N}(\mu, \sigma^2).$$

1.2.1 Gaussian distribution in multiple dimensions

The joint probability density distribution of a vector X consisting of N random variables with mean vector μ_X and covariance matrix Σ is given by a multivariate Gaussian distribution described as follows

$$f(x) = \frac{1}{(2\pi)^{N/2} det(\Sigma)} e^{(-\frac{1}{2}(x-\mu_x)^T \Sigma^{-1}(x-\mu_x))}.$$

1.2.2 Example: Multi-dimensional Gaussian distribution

Assume the position on the globe. We only consider two-dimensional positions (e.g., longitude, latitude)and height is ignored. For example, the latitude and longitude of the CTU FIT building is $\mu = (50.105060, 14.389683)$. If we use a measurement covariance matrix

$$\Sigma = \begin{bmatrix} 0.0000001 & 0 \\ 0 & 0.0000001 \end{bmatrix}$$

we get a bivariate Gaussian distribution that describes how our instrumental measurements will be distributed. That is, most of the measurements will lie close to the mean, while the more distant measurements will be rather improbable. This is the principle behind the GPS-based localization. The Gaussian distribution is shown in Figure 1.2.

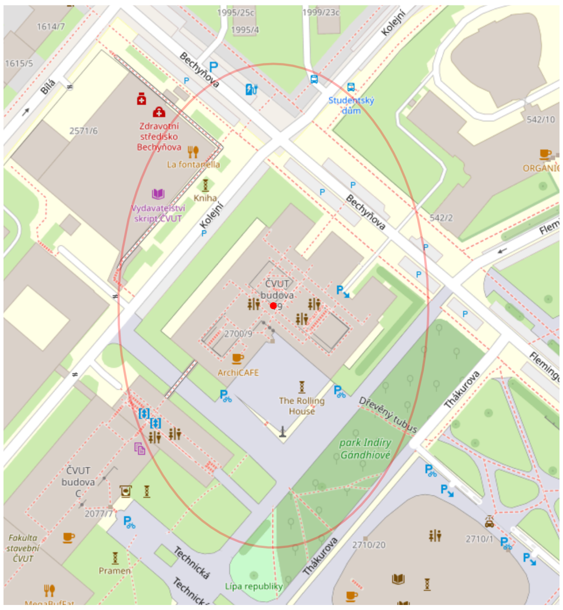


Figure 1.2 Top view of Gaussian distribution centered on CTU FIT

1.3 Bayesian updating

Bayesian updating is a fundamental concept in probability theory that describes how to update the probability of a hypothesis based on new evidence. It is named after

Thomas Bayes, an 18th-century British mathematician. This procedure involves combining the original probability of a hypothesis with new evidence to obtain an updated probability.

1.3.1 Bayes' theorem

Bayes' theorem is a key element in the fields of statistics, machine learning, and other fields where there is a need to quantify uncertainty and make estimates of probabilities based on available data. The exact wording is as follows

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \qquad P(B) > 0,$$

where

- $P(A|B)$ is the conditional probability of event A occurring given that B is true,
- $P(A)$ and $P(B)$ are the probabilities of events A and B.

For our purposes, we use Bayes' theorem for state estimation. The estimator is represented by a prior distribution $p(\theta)$ that is updated by new data obeying a model $f(x|\theta)$. The result is a posterior distribution $p(\theta|x)$.The following formulation is preferable.

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{f(x)}, \qquad f(x) > 0.$$

where

- $\pi(\theta|x)$ is the **posterior** conditional density of θ
- $\pi(\theta)$ is the **prior** density
- $f(x|\theta)$ is the model or likelihood of the data
- $f(x)$ is the marginal density of X also known as the evidence.

In this case, the denominator (marginal density of X) is a normalizing factor independent of the estimated θ. The denominator is obtained by integrating the numerator

$$f(x) = \int f(x|\theta)p(\theta)d\theta.$$

For simplicity is written by proportionality

$$\pi(\theta|x) \propto f(x|\theta)\pi(\theta).$$

This can be equivalently written using the constant c

$$\pi(\theta|x) = c \cdot f(x|\theta)\pi(\theta).$$

1.3.2 Bayesian estimation of state-space models

The Bayesian approach to estimating state space models based on Equation (1.1) and Equation (1.2) can be rewritten in terms of conditional probabilities using pdfs

$$x_t|x_{t-1} \sim \pi(x_t|x_{t-1}) \tag{1.13}$$

$$y_t|x_t \sim p(y_t|x_t) \tag{1.14}$$

Bayesian estimation runs in two steps. Prediction and update. Denoting all previous values $\{y_1, ..., y_t\}$ as Y_t, and the prior pdf of x_{t-1} as $\pi(x_t|Y_{t-1})$ we can define this steps as follows

(i) Prediction using the Chapman-Kolmogorov equation:

$$\pi(x_t|Y_{t-1}) = \int \pi(x_t|x_{t-1})\pi(x_{t-1}|Y_{t-1})dx_{t-1}. \tag{1.15}$$

(ii) Update - using the Bayes' theorem 1.3.1

$$\pi(x_t|Y_t) = \frac{\pi(x_t|Y_{t-1})p(y_t|x_t)}{\int \pi(x_t|Y_{t-1})p(y_t|x_t)dx_t}. \tag{1.16}$$

1.4 Kalman Filter

The Kalman filter is one of the most used algorithms for sequential estimation. Its main function is to make estimates of hidden variables based on imprecise measurements while predicting the system's future state using past estimates. It has become an important tool in many other areas, including navigation systems, target tracking, control systems, computer graphics, financial analysis, biomedicine, and more. The Kalman filter is a linear model. This is perfectly satisfactory since we will only work with the linear model in this thesis. From Section 1.1 we have the following model

$$x_t = A_t x_{t-1} + w_t, \qquad w_t \sim \mathcal{N}(0, Q), \tag{1.17}$$

$$y_t = H_t x_t + v_t, \qquad v_t \sim \mathcal{N}(0, R). \tag{1.18}$$

Using normality, we can rewrite the equations in this form

$$x_t \sim \mathcal{N}(Ax_{t-1}, Q), \tag{1.19}$$

$$y_t \sim \mathcal{N}(Hx_t, R). \tag{1.20}$$

From now on we mark all variables after the prediction step with a superscript "-" and variables after the correction (update) step with a superscript "+". Let us denote

$$\pi(x_t|y_{0:t-1}) = \mathcal{N}(x^+_{t-1}, P^+_{t-1}). \tag{1.21}$$

The Kalman filter works in two steps. The first step is to predict the state and the second is to update the predicted state with the incoming measurement. We use the first equation of the state model for prediction and the second equation of the state model for update.

1.4.1 Prediction step

The whole point of the prediction step is to estimate the current state by estimating the previous state using a state evolution model. That is, we solve for the evolution of the state $x_{t-1} \to x_t$. We use the estimate from the last posterior, which is now our prior for the next time step. We apply the state evolution model to this estimate.

$$\pi(x_t|y_{0:t-1}) = \int p(x_t|x_{t-1})\,\pi(x_{t-1}|y_{0:t-1})\,\mathrm{d}x_{t-1}. \tag{1.22}$$

Since we multiply the two normal distributions and marginalize, we get the normal distribution again $\mathcal{N}(x_t^-, P_t^-)$ with hyperparameters

$$x_t^- = Ax_{t-1}^+, \tag{1.23}$$
$$P_t^- = AP_{t-1}^+A^\intercal + Q. \tag{1.24}$$

1.4.2 Update step

Update step uses the Bayes' theorem to correct the predicted state with new data y_t,

$$\pi(x_t|y_{0:t}) \propto (y_t|x_t)\pi(x_t|y_{0:t-1}). \tag{1.25}$$

We now rewrite the model in the form of an exponential distribution. The Bayesian update then corresponds to the sum of the hyperparameters and the sufficient statistic. The model after the above modifications looks as follows

$$\begin{aligned} f(y_t|x_t) &\propto \exp\left\{-\frac{1}{2}(y_t - Hx_t)^\intercal R^{-1}(y_t - Hx_t)\right\} \\ &= \exp\left\{\mathrm{Tr}\left(\underbrace{-\frac{1}{2}\begin{bmatrix}-1\\ x_t\end{bmatrix}\begin{bmatrix}-1\\ x_t\end{bmatrix}^\intercal}_{\eta}\underbrace{\begin{bmatrix}y_t^\intercal\\ H^\intercal\end{bmatrix}R^{-1}\begin{bmatrix}y_t^\intercal\\ H^\intercal\end{bmatrix}^\intercal}_{T(y_t)}\right)\right\} \end{aligned}$$

and prior distribution looks as follows

$$\begin{aligned} \pi(x_t|y_{0:t-1}) &\propto \exp\left\{-\frac{1}{2}(x_t - x_t^-)^\intercal(P_t^-)^{-1}(x_t - x_t^-)\right\} \\ &= \exp\left\{\mathrm{Tr}\left(\underbrace{-\frac{1}{2}\begin{bmatrix}-1\\ x_t\end{bmatrix}\begin{bmatrix}-1\\ x_t\end{bmatrix}^\intercal}_{\eta}\underbrace{\begin{bmatrix}(x_t^-)^\intercal\\ I\end{bmatrix}(P_t^-)^{-1}\begin{bmatrix}(x_t^-)^\intercal\\ I\end{bmatrix}^\intercal}_{\xi_t}\right)\right\}, \end{aligned}$$

where I is identity matrices of the appropriate dimension. The Bayesian update corresponds to the sum of the hyperparameters and the sufficient statistic.

$$\begin{aligned} \xi_t &= \xi_{t-1} + T(y_t) \\ &= \begin{bmatrix}(x_t^-)^\intercal(P_t^-)^{-1}x_t^- + y_t^\intercal R^{-1}y_t, & (x_t^-)^\intercal(P_t^-)^{-1} + y_t^\intercal R^{-1}H \\ (P_t^-)^{-1}(x_t^-)^\intercal + H^\intercal R^{-1}y_t, & (P_t^-)^{-1} + H^\intercal R^{-1}H\end{bmatrix}. \end{aligned}$$

The posterior distribution hyperparameters are defined as follows

$$\begin{aligned} P_t^+ &= (\xi_{t;[2,2]})^{-1} \\ &= \left[(P_t^-)^{-1} + H^\intercal R^{-1} H\right]^{-1} \\ &= (I - K_t H)P_t^- \\ x_t^+ &= (\xi_{t;[2,2]})^{-1}\xi_{t;[2,1]} \\ &= P_t^+ \left[(P_t^-)^{-1}(x_t^-)^\intercal + H^\intercal R^{-1} y_t\right] \\ &= x_t^- + P_t^+ H^\intercal R^{-1}(y_t - Hx_t^-) \end{aligned}$$

where

$$K_t = P_t^- H^\intercal (R + HP_t^- H^\intercal) \tag{1.26}$$

is a Kalman gain. Increasing the value of the Kalman gain amplifies the impact of a measurement during the update process. Greater Kalman gain makes the filter more sensitive, but it becomes less capable of filtering out noise. For easy understanding we will expand the Kalman filter equations as follows

$$\hat{y}_t^- = Hx_t^-, \quad \text{prediction of measurement} \tag{1.27}$$

$$\nu_t = y_t - \hat{y}_t^-, \quad \text{innovation,prediction error } y_t \tag{1.28}$$

$$S_t = HP_t^- H^\intercal + R, \quad \text{covariance innovation } \nu_t \tag{1.29}$$

$$K_t = P_t^- H^\intercal S_t^{-1}, \quad \text{Kalman gain} \tag{1.30}$$

$$x_t^+ = \hat{x}_t^- + K_t \nu_t, \quad \text{posterior state estimation } x_t \tag{1.31}$$

$$P_t^+ = (I - K_t H)P_t^-. \quad \text{posterior covariance} \tag{1.32}$$

1.5 The data association problem in target tracking

The Kalman filter works with the assumption that it has just one measurement to update at any given time. In the real world, this may not be the case. We may get no measurements or, conversely, several measurements, some of which are false. So, we get a situation where one target has generated multiple measurements. In the context of this paper, we restrict ourselves to the case where each target could only generate one measurement, and all other measurements are either from another target or fake measurements.

1.5.1 Clutter

Clutter are false, spurious measurements. These false measurements can distort the measurement from the actual target. That is why clutter must be filtered out. Examples of clutter can be storm clouds, signal reflections, animals, the sea, or some other object we are not tracking.

1.5.2 Poisson distribution

The Poisson distribution is a useful tool for determining the probability of an event happening a certain number of times. The Poisson distribution in the target tracking domain serves as a model for the nonnegative numbers of targets, false alerts (clutter), or similar phenomena. Formula for Poisson distribution

$$f(x) = \frac{\lambda^x}{x!}e^{-\lambda}, \qquad \lambda > 0, \lambda = E(X) = Var(X) \tag{1.33}$$

where

- X is the discrete random variable
- x is the number of occurrences
- λ is equal to the expected value of x when that is also equal to its variance.

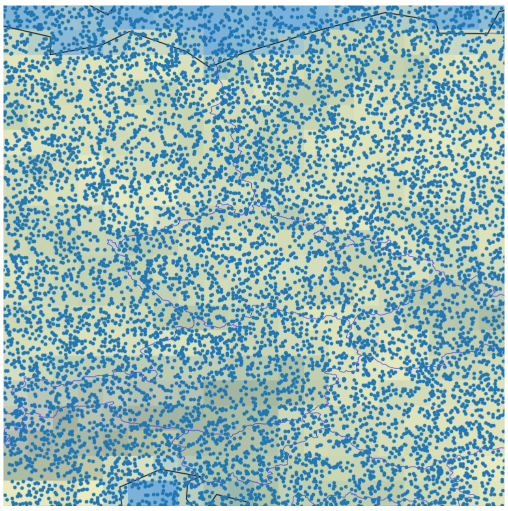
Notation:

$$X \sim Po(\lambda).$$

1.5.3 Amount of clutter

We denote the clutter intensity as λ and the Poisson rate as $\Lambda = \lambda \cdot V$, where V stands for volume of the area. The amount of clutter will be $\sim Po(\Lambda) = Po(\lambda \cdot V)$. To estimate the amount of clutter, we must know the area over which our target moves.

For example we take an area of 1000×1000 Its volume will be 1 000 000. In Figure 1.3 we can see results for $\lambda = 0.01$ and $\lambda = 0.0001$.


(a) $\lambda = 0.01$

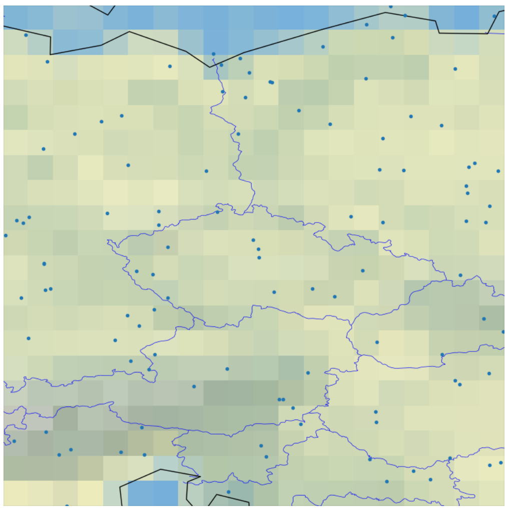

(b) $\lambda = 0.0001$

Figure 1.3 Clutter amount based on λ

1.5.4 Mahalanobis distance

Consider a scenario where a sample vector $X \in \mathbb{R}^m$ originates from a distribution $\mathbb{P}$ characterized by a mean vector μ and a covariance matrix Σ. It is worth noting that the vector μ, represented as $\mu = (\mu_1, \dots, \mu_m)$, encapsulates the mean values of the m variables of $\mathbb{P}$. Subsequently, we denote the jth sample vector as $X_j = (x_{j1}, \dots, x_{jm})$, where each x_{ji} represents one of the m elements. The multivariate Mahalanobis distance (d_M) of the sample vector concerning μ is formulated as

$$d_M(X_j, \mu) = \sqrt{(X_j - \mu)^\intercal \Sigma^{-1} (X_j - \mu)}. \tag{1.34}$$

When dealing with two sample vectors originating from the same distribution $\mathbb{P}$, the multivariate Mahalanobis distance is a dissimilarity metric for assessing interpoint separation. Consequently, between X_1 and X_2, it is calculated as:

$$d_M(X_1, X_2) = \sqrt{(X_1, X_2)^\intercal \Sigma^{-1} (X_1, X_2)}. \tag{1.35}$$

This distance metric operates by initially projecting the data onto a principal component space where the axes are independent, then rescaling to unit variance. Consequently, the Mahalanobis distance becomes analogous to a standard Euclidean distance within the transformed space [9].

1.5.5 Gating

In target tracking, gating refers to a technique used to associate detections or measurements with existing targets. It involves spatially constraining potential detections to a region where it is plausible for the target to be located based on its predicted position and uncertainty. This helps mitigate the risk of associating spurious measurements with targets, leading to track divergence or association errors. In our application, the validation region is the elliptical region

$$\nu(t, \gamma) = \{y : [(y - \hat{y}_{t|t-1}))^\intercal S(t)^{-1} (y - \hat{y}_{t|t-1}))] \leq \gamma\}. \tag{1.36}$$

Where γ is the gate threshold, $S(t)$ is the covariance of the innovation corresponding to the true measurement given by Equation (1.29).

1.5.6 Misdetections

Another problem we face is misdetection. This means that the sensor has not detected the tracked object and thus we shave no measurements to update our model. We assume that detection and misdetection are independent at each time step and target state. Then the model for this event is Bernoulli model that reads.

$$P(\delta) = \begin{cases} p_D & \text{if } \delta = 1, \\ 1 - p_D & \text{if } \delta = 0, \end{cases} \tag{1.37}$$

where p_D is the probability of detection, $\delta = 1$ indicates target detected and $\delta = 0$ target non-detected.For further use in our work, we will use $p_D = 0.95$, which corresponds to the probability of aircraft detection.

Target	Measurement
$\hat{y}$	$\emptyset$
$\hat{y}$	y_1^*
$\hat{y}$	y_2^*
$\hat{y}$	y_3^*

Table 1.1 Table showing possible data association for single target

1.5.7 Data association - single target

The problem with single target tracking is encountered when multiple measurements occur in the validation region (the area where we expect to measure). Among all these measurements, there is one real one from our target (if there was no misdetection), and the others are clutter. The situation where we have multiple measurements in a single validation region is shown in Figure 1.4. The validation region is in the form of an ellipse centered at the measurement's prediction $\hat{y}$. All measurements in the validation region (y_1^*, y_2^*, y_3^*) can come from the target. All possible assignments for each measurement can be seen in Table 1.1.

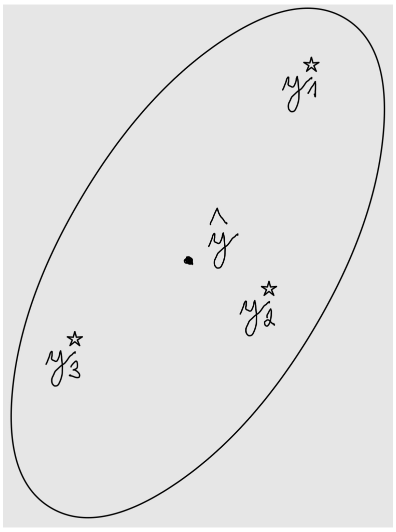


Figure 1.4 Multiple measurements in validation raegion for predicted $\hat{y}$

1.5.8 Data association - multiple targets

For multi-target tracking, data association is a much more complex problem. If multiple targets are in the near distance and their validation regions intersect, the situation depicted in Figure 1.5 is that one measurement can be assigned to multiple targets. At the same time, each target has multiple measurements only in its own validation regions. In this situation, we do not know whether to associate the measurement y_1^* to the target $\hat{y}^1$ or the target $\hat{y}^2$. Here, we have many options for assigning measurements to a target. All possible assignments for each measurement can be seen in Table 1.2.

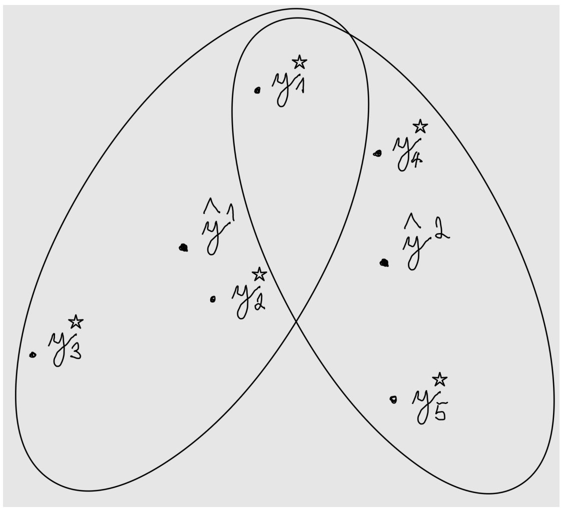


Figure 1.5 Two targets $\hat{y}^1$ and $\hat{y}^2$ with measurements in their validation regions with measurement y_1^* in the intersection of validation regions

Target $\hat{y}^1$ measurement	Target $\hat{y}^2$ measurement
$\emptyset$	$\emptyset$
$\emptyset$	y_1^*
$\emptyset$	y_4^*
$\emptyset$	y_5^*
y_1^*	$\emptyset$
y_1^*	y_4^*
y_1^*	y_5^*
y_2^*	$\emptyset$
y_2^*	y_1^*
y_2^*	y_4^*
y_2^*	y_5^*
y_3^*	$\emptyset$
y_3^*	y_1^*
y_3^*	y_4^*
y_3^*	y_5^*

Table 1.2 Table showing possible data association for multiple targets

Chapter 2

Multi Target Tracking

So far, we have worked with a single target only. Now, we are moving into a situation where we have more than one target. In many cases, the real number of targets is not known a priori from a sensor's measurements. There can be one or multiple targets. There may also be a situation where there is no target. If we know that there are no targets in our observation area and we still get an echo from the sensor, we call these measurements clutter 1.5.1.

The biggest problem with multi-target tracking is the data association Section 1.5.8. This problem is compounded by the fact that we do not know the number of targets we have in the observation area. Therefore, assigning the correct measurement to the proper target is difficult task.

Another thing to watch out for is knowing where the target might come from. Essentially, it is assumed that the targets may occur either on the edge of the sensor's field of view (inbound targets) or they may arise in specific locations within the field of view. On the other hand, we have to take into account the possibility of the target disappearing in our area (the target reaches its final destination).

The main content of this chapter will be the introduction of the Probability Hypothesis Density filter, which will be subsequently modified and used for experiments.

All the information in this chapter are based on [10, 11, 12, 13, 14, 15, 16].

2.1 Random Finite set

This section will explain what RFS is to help us better understand the rest of the text. A RFS is a finite, set with random cardinality of random unique random variables. We can denote it as $x = \{x^1, ...x^n\}$, where n may be different at each time step.

In RFS, we do not care about the order of elements. Each element can occur in the set once. We denote the empty RF by $\emptyset$. The number of elements in RFS x will be denoted by $|x|$. For instance $x = \{a, b, c\} \Rightarrow |x| = 3$.

Let us denote by $M(t)$ the number of targets at time step t and assume that at time step $t-1$ the states of the targets were as follows $x_{t-1,1}, x_{t-1,2}, ..., x_{t-1,M(t-1)} \in \mathcal{X}$, where $\mathcal{X}$ is the state space. In the next time step, some targets may disappear, some new ones may appear, and the current targets evolve to their new states. This gives $M(t)$ a new set of targets $x_{t,1}, x_{t,2}, ..., x_{t,M(t)}$. For these targets, at time t, we receive $N(t)$ measurements $y_{t,1}, y_{t,2}, ..., y_{t,N(t)} \in \mathcal{Y}$ from the radar. Without individual identification, it is challenging to attribute specific measurements to particular targets. It is also not clear whether the measurements belong to any of the targets at all and are the measurements are false measurements (Clutter). The goal of multiple-target tracking is to estimate the number of targets and assign the correct measurements to them. In an ideal scenario where only true measurements from actual targets are received, without any interference from clutter or false detections. It may still be difficult or impossible to determine which measurement corresponds to which specific target. See Section 1.5.8. Since we do not care about the order, we can represent the states of the targets and measurements at time t as follows

$$X_t = \{x_{t,1}, x_{t,2}, ..., x_{t,M(t)}\} \in \mathcal{F}(\mathcal{X}), \tag{2.1}$$

$$Y_t = \{y_{t,1}, y_{t,2}, ..., y_{t,N(t)}\} \in \mathcal{F}(\mathcal{Y}), \tag{2.2}$$

where $\mathcal{F}(\mathcal{X})$ and $\mathcal{F}(\mathcal{Y})$ are the respective collections of all finite subsets of $\mathcal{X}$ and $\mathcal{Y}$.

In the random finite set formulation, the approach considers the target set X_t and the measurement set Y_t as representations of the multiple-target state and multiple-target observation, respectively. This perspective allows for the multiple-target tracking issue to be framed as a filtering problem, where the state space $(\mathcal{F}(\mathcal{X}))$ and observation space $(\mathcal{F}(\mathcal{Y}))$ both represent multiple targets.

Targets can die, survive, and new targets can be born from time step $t-1$ to time step t. The set of all targets that have survived to time step t merged with the set of all newly born targets forms the set X_t.

The set of surviving targets is obtained by computing for all $x_{t-1} \in X_{t-1}$ their RFS $S_{t|t-1}(x_{t-1})$ such that it either contains $\{x_t\}$ with probability $p_{S,t}$ if the target survives or it contains $\{\emptyset\}$ with probability $1 - p_{S,t}(x_{t-1})$ if the target dies. The set of newly born targets Γ_t is created by initializing new targets at locations where the target may be born (e.g., airports or the boundaries of the area we are monitoring). As opposed to the single target Kalman filter, which creates one target in the first time step and ignores other potential births. Thus, we write the target X_t as

$$X_t = \left[\bigcup_{\zeta \in X_{t-1}} S_{t|t-1}(\zeta) \right] \cup \Gamma_t. \tag{2.3}$$

The RFS model for the measurement, which accounts for detection uncertainty and clutter, is described as follows. Each target $x_t \in X_t$ is either detected with probability $p_{D,t}(x_t)$ or not detected with probability $1 - p_{D,t}(x_t)$. At each time step t, each target $x_t \in X_t$ generates an RFS

$$\Theta_t(x_t),$$

which takes the values $\{y_t\}$ if the target was detected or $\{\emptyset\}$ if the target was undetected. In addition to the original measurements from the targets, the sensor may receive a set of false 0.measurements or clutter K_t. Thus, the set of all measurements obtained by the sensor can be referred to as the union of the true measurements and the clutter, i.e.,

$$Y_t = K_t \cup \left[\bigcup_{x \in X_t} \Theta_t(x) \right]. \tag{2.4}$$

2.2 Bayes filter for multiple-targets

In earlier chapters, we have already explained the principles of using Bayes' theorem for single-target tracking. We will show how Bayes' theorem can help us in multiple-target tracking. Let $\pi_t(\cdot|Y_{1,t})$ denote the multiple-target posterior density for RFS variables. The optimal multiple-target Bayesian filter propagates the multiple-target posterior density conditional on all measurements up to time step t. We then obtain the following recursion

$$\pi_{t|t-1}(X_t|Y_{1:t-1}) = \int f_{t|t-1}(X_t|X)\pi_{t-1}(X|Y_{1:t-1})\mu_s(dX), \tag{2.5}$$

$$\pi_t(X_t|Y_{1:t}) = \frac{g_t(Y_t|X_t)\pi_{t|t-1}(X_t|Y_{1:t-1})}{\int g_t(Y_t|X)\pi_{t|t-1}(X|Y_{1:t-1})\mu_s(dX)}, \tag{2.6}$$

where μ_s is an appropriate reference measure on $\mathcal{F}(\mathcal{X})$. The function $g_t(Y_t|X_t)$ is global density of the set of measurements Y given by set of target states X. This refers to the total probability density governing the association between the measurements in Y and the parameters in X.

2.3 Intensity

Intensity is also known in the literature about tracking as probability hypothesis density. For an RFS with probability distribution P, its first-order moment is a nonnegative function υ on $\mathcal{X}$, referred to as the intensity. This intensity function satisfies the condition for each region $S \in \mathcal{X}$,

$$\int |X \cap S| P(dX) = \int_s \upsilon(x)dx. \tag{2.7}$$

After integrating *upsilon* over any region S, we get the expected number of elements of X in the region S. The expected total number of elements is given by the total mass $\hat{N} = \int \upsilon(x)dx$. To estimate the elements from X, we can use the local maxima of the υ intensity since these are the points from $\mathcal{X}$ with the highest local concentration of the expected number of elements. The most straightforward approach is to round and choose the resulting number of highest peaks from the intensity.

2.4 The Probability Hypothesis Density filter

Because the Bayes filter for multiple targets demands significant computational resources, the PHD filter was devised to mitigate this challenge. In contrast to the Bayes filter, which evolves the posterior density of multiple targets, the PHD filter advances the posterior intensity, representing the first-order statistical moment of the posterior density. An analogy can be drawn to a constant-gain Kalman filter, which advances the mean values of the single-target state. Before we go any further, let's introduce the three assumptions we have for our PHD filter.

▶ Assumption 1. Each target evolves and generates observations independently of one another.

▶ Assumption 2. Clutter is Poisson and independent of target-originated measurements.

▶ Assumption 3. The predicted multiple-target RFS is Poisson.

Next, let us define some variables to work with.

- $\gamma_t(\cdot)$ intensity of the birth RFS at time t;
- $p_{S,t}(\zeta)$ probability that a target still exists at time t given that its previous state is ζ;
- $p_{D,t}(x)$probability of detection given a state x at time t;
- $\kappa_t(\cdot)$ intensity of clutter RFS K_t at time t;

Under the assumptions defined above, we can show that the posterior intensity can be propagated in time using PHD recursion as follows:

$$\upsilon_{t|t-1}(x) = \int p_{S,t}(\zeta) f_{t|t-1}(x|\zeta)\upsilon_{t-1}(\zeta)d\zeta, \tag{2.8}$$

$$\upsilon_t(x) = [1 - p_{D,t}(x)]\upsilon_{t|t-1}(x) + \sum_{y\in Y_t} \frac{p_{D,t}(x)g_t(y|x)\upsilon_{t|t-1}(x)}{\kappa_t(y) + \int p_{D,t}(\xi)g_t(y|\xi)\upsilon_{t|t-1}(\xi)}. \tag{2.9}$$

Since the posterior intensity is a function on the single-target state space $\mathcal{X}$, the PHD filter completely avoids combinatorial computations from the unknown assignment of measurements to their respective targets, as can be seen in Equation (2.8) and Equation (2.9) For this reason, the PHD filter requires much less computational power than the computation in Equation (2.5) and Equation (2.6), which works over $\mathcal{F}(\mathcal{X})$.

2.5 Linear Gaussian multiple-target model

The linear Gaussian filter requires three additional assumptions to those already stated in Section 2.4.

▶ Assumption 4. Each target adheres to a linear Gaussian dynamical model, while the sensor operates under a linear Gaussian measurement model, implying that

$$f_{t|t-1}(x|\zeta) = \mathcal{N}(x; F_{t-1}\zeta, Q_{t-1}), \tag{2.10}$$

$$g_t(y|x) = \mathcal{N}(y; H_t x, R_t). \tag{2.11}$$

Here, $\mathcal{N}(\cdot; m, P)$ denotes a Gaussian density with mean m and covariance P. F_{t-1} represents the state transition matrix, Q_{t-1} signifies the process noise covariance, H_t stands for the observation matrix, and R_t denotes the observation noise covariance.

▶ Assumption 5. The survival and detection probabilities are independent of the state, meaning that

$$p_{S,t}(x) = p_{S,t}, \tag{2.12}$$

$$p_{D,t}(x) = p_{D,t}. \tag{2.13}$$

▶ Assumption 6. The intensities of the birth RFSs are represented as Gaussian mixtures in the following manner

$$\gamma_t(x) = \sum_{i=1}^{J_{\gamma,t}} w_{\gamma,t}^{(i)} \mathcal{N}(x; m_{\gamma,t}^{(i)}, P_{\gamma,t}^{(i)}), \tag{2.14}$$

where $J_{\gamma,t}$, $w_{\gamma,t}^{(i)}$, $m_{\gamma,t}^{(i)}$, $P_{\gamma,t}^{(i)}$, $i = 1, ..., J_{\gamma,t}$, represent the given model parameters defining the shape of the birth intensity.

▶ Remark 2.1. In Assumption 6, $m_{\gamma,t}^{(i)}$, $i = 1, ..., J_{\gamma,t}$ denote the peaks of the spontaneous birth intensity in Equation (2.14). These locations represent areas with the highest local concentrations of expected spontaneous births. For instance, in general target tracking applications, these are the air bases, airports, boundaries of the sensor's field of view etc. In our application, we consider bird nesting sites to be the following locations. The covariance matrix $P_{\gamma,t}^{(i)}$ determines the spread of the birth intensity around each peak $m_{\gamma,t}^{(i)}$. The weight $w_{\gamma,t}^{(i)}$ signifies the expected number of new targets originating from $m_{\gamma,t}^{(i)}$.

Now, with all the necessary components at hand, we can proceed to derive the recursion for the linear Gaussian PHD model.

2.6 The Gaussian mixture PHD recursion

Similarly to the Kalman filter derived in Section 1.4, the Gaussian mixture PHD filter operates in two subsequent steps. First, the previous posterior information is predicted for the current time step. Then, the prediction is updated by new observations.

2.6.1 Predict step

Suppose that at time $t-1$, the posterior intensity $\upsilon_{t-1}(x)$ is a Gaussian mixture expressed as

$$\upsilon_{t-1}(x) = \sum_{i=1}^{J_{t-1}} w_{t-1}^{(i)} \mathcal{N}(x; m_{t-1}^{(i)}, P_{t-1}^{(i)}). \tag{2.15}$$

Then, the predicted intensity for time t is also a Gaussian mixture and is provided by

$$\upsilon_{t|t-1}(x) = \upsilon_{S,t|t-1}(x) + \gamma_t(x), \tag{2.16}$$

where $\gamma_t(x)$ is given by Equation (2.14). In particular,

$$\upsilon_{S,t|t-1}(x) = p_{S,t}(x) \sum_{j=1}^{J_{t-1}} w_{t-1}^{(j)} \mathcal{N}(x; m_{S,t|t-1}^{(j)}, p_{S,t|t-1}^{(j)}), \tag{2.17}$$

$$m_{S,t|t-1}^{(j)} = F_{t-1} m_{t-1}^{(j)}, \tag{2.18}$$

$$p_{S,t|t-1}^{(j)} = Q_{t-1} + F_{t-1} P_{t-1}^{(j)} F_{t-1}^{\intercal}. \tag{2.19}$$

The prediction step of the Gaussian mixture PHD filter described in this section is depicted in Algorithm 1.

Algorithm 1: Pseudocode for the Gaussian mixture PHD filter - Predict step

```
   given  : $\{w_{t-1}^{(i)}, m_{t-1}^{(i)}, P_{t-1}^{(i)}\}_{i=1}^{J_{t-1}}$
   step 1.(prediction for birth targets)
1  $i = 0$
2  for $j = 1, ..., J_{\gamma,t}$ do
3      $i := i + 1$
4      $w_{t|t-1}^{(i)} = w_{t-1}^{(j)}$
5      $m_{t|t-1}^{(i)} = m_{\gamma,t}^{(j)}$
6      $P_{t|t-1}^{(i)} = P_{\gamma,t}^{(j)}$
   step 2.(prediction for existing targets)
7  for $j = 1, ..., J_{t-1}$ do
8      $i := i + 1$
9      $w_{t|t-1}^{(i)} = p_{S,t} w_{t-1}^{(j)}$
10     $m_{t|t-1}^{(i)} = F_{t-1} m_{t-1}^{(j)}$
11     $P_{t|t-1}^{(i)} = Q_{t-1} + F_{t-1} P_{t-1}^{(j)} F_{t-1}^{\intercal}$
12 $J_{t|t-1} = i$
   output: $\{w_{t|t-1}^{(i)}, m_{t|t-1}^{(i)}, P_{t|t-1}^{(i)}\}_{i=1}^{J_{t|t-1}}$
```

2.6.2 Update step

Predicted insensity $v_{t|t-1}(x)$ in time t is Gaussian mixture of the form

$$v_{t|t-1}(x) = \sum_{i=1}^{J_{t-1}} w_{t-1}^{(i)} \mathcal{N}(x; m_{t-1}^{(i)}, P_{t-1}^{(i)}). \tag{2.20}$$

The resulting intensity at time t consists of two parts. The intensity from the prediction step $v_{t|t-1}(x)$, which must be multiplied by the probability that the target was not detected $(1 - p_{D,t})$ and the sum of the intensities $v_{D,t}(x, y)$ over all measurements. Intensity $v_{D,t}(x, y)$ for one measurement $y \in Y_t$ is calculated as follows

$$v_{D,t}(x, y) = \sum_{i=1}^{J_{t|t-1}} w_t^{(i)}(y) \mathcal{N}(x; m_{t|t}^{(i)}(y), P_{t|t}^{(i)}). \tag{2.21}$$

If we break down all the components we get

$$w_t^{(i)}(y) = \frac{p_{D,t} w_{t|t-1}^{(i)} q_t^{(i)}(y)}{\kappa_t(y) + p_{D,t} + \sum_{j=1}^{J_{t|t-1}} w_{t|t-1}^{(i)} q_t^{(i)}}, \tag{2.22}$$

$$m_{t|t}^{(i)}(y) = m_{t|t-1}^{(i)} + K_t^{(i)}(y - H_t m_{t|t-1}^{(i)}), \tag{2.23}$$

$$P_{t|t}^{(i)} = \left[I - K_t^{(i)} H_t\right] P_{t|t-1}^{(i)}, \tag{2.24}$$

$$K_t^{(i)} = P_{t|t-1}^{(i)} H_t^{\intercal} (H_t P_{t|t-1}^{(i)} H_t^{\intercal} + R_t)^{-1}. \tag{2.25}$$

The equation to obtain the resulting intensity will therefore be written in the form

$$\upsilon_t(x) = (1 - p_{D,t})\upsilon_{t|t-1}(x) + \sum_{y \in Y} \upsilon_{D,t}(x, y). \tag{2.26}$$

The entire Update step is shown in Algorithm 2.

Algorithm 2: Pseudocode for the Gaussian mixture PHD filter - Update step

```
   given  : $\{w^{(i)}_{t|t-1}, m^{(i)}_{t|t-1}, P^{(i)}_{t|t-1}\}_{i=1}^{J_{t|t-1}}$, and the measurement set $Y_t$
   step 1.(construction of PHD components)
1  for $j = 1, ..., J_{t|t-1}$ do
2      $\eta^{(j)}_{t|t-1} = H_t m^{(j)}_{t|t-1}$
3      $S^{(j)}_t = R_t + H_t P^{(j)}_{t|t-1} H^\intercal_t$
4      $K^{(j)}_t = P^{(j)}_{t|t-1} H^\intercal_t [S^{(j)}_t]^{-1}$
5      $P^{(j)}_{t|t} = [I - K^{(j)}_t H_t] P^{(j)}_{t|t-1}$
   step 2.(update)
6  for $j = 1, ..., J_{t|t-1}$ do
7      $w^{(j)}_t = (1 - p_{D,t}) w^{(j)}_{t|t-1}$
8      $m^{(j)}_t = m^{(j)}_{t|t-1}$
9      $P^{(j)}_t = P^{(j)}_{t|t-1}$
10 $l := 0$
11 foreach $y \in Y$ do
12     $l := l + 1$
13     for $j = 1, ..., J_{t|t-1}$ do
14         $w^{(lJ_{t|t-1}+j)}_t = p_{D,t} w^{(j)}_{t|t-1} \mathcal{N}(y; \eta^{(j)}_{t|t-1} S^{(j)}_t)$
15         $m^{(lJ_{t|t-1}+j)}_t = m^{(j)}_{t|t-1} + K^{(j)}_t (y - \eta^{(j)}_{t|t-1})$
16         $P^{(lJ_{t|t-1}+j)}_t = P^{(j)}_{t|t}$
17     for $j = 1, ..., J_{t|t-1}$ do
18         $w^{(lJ_{t|t-1}+j)}_t := \frac{w^{(lJ_{t|t-1}+j)}_t}{\kappa_t(y) + \sum_{i=1}^{J_{t|t-1}}} w^{(lJ_{t|t-1}+j)}_t$
19 $J_t = lJ_{t|t-1} + J_{t|t-1}$
   output: $\{w^{(i)}_t, m^{(i)}_t, P^{(i)}_t\}_{i=1}^{J_t}$.
```

2.6.3 Merge

The Gaussian mixture PHD filter faces computational challenges due to the escalating number of Gaussian components over time. Specifically, at time t, the filter necessitates

$$(J_{t-1} + J_{\gamma,t})(1 + |Z_t|) = \mathcal{O}(K_{t-1}|Z_t|) \tag{2.27}$$

Gaussian components to represent v_t, where J_{t-1} denotes the number of components of v_{t-1}. Consequently, the number of components in the posterior intensities grows indefinitely.

To address this issue, a straightforward pruning procedure can be employed to reduce the number of Gaussian components propagated to the next time step. This involves obtaining a good approximation to the Gaussian mixture posterior intensity

$$v_t(x) = \sum_{i=1}^{J_t} w_t^{(i)} \mathcal{N}(x; m_t^{(i)}, P_t^{(i)}) \tag{2.28}$$

by truncating components with weak weights $w_t^{(i)}$. This pruning can be achieved by discarding components with weights below a preset threshold or retaining only certain components with the strongest weights. Additionally, some Gaussian components may be sufficiently close to each other that a single Gaussian can accurately approximate them. Consequently, these components can be merged into one. These concepts form the basis of the simple heuristic pruning algorithm depicted in Algorithm 3.

Algorithm 3: Pseudocode for the Gaussian mixture PHD filter - Prunning and Merging

```
given  : $\{w_t^{(i)}, m_t^{(i)}, P_t^{(i)}\}_{i=1}^{J_t}$, a truncation threshold $T$, a merging threshold $U$,
         and a maximum allowable number of Gaussians terms $J_{max}$.
         Set $\ell = 0$, and $I = \{i = 1, ..., J_t | w_t^{(i)} > T\}$.
1  repeat
2      $\ell := \ell + 1$
3      $j := \underset{i \in I}{\text{argmax}}(w_t^{(i)})$
4      $L := \left\{ i \in I \,\middle|\, (m_t^{(i)} - m_t^{(j)})^\intercal (P_t^{(i)})^{-1} (m_t^{(i)} - m_t^{(j)}) \leq U \right\}$
5      $\tilde{w_t}^{(\ell)} = \sum_{i \in L} w_t^{(i)}$
6      $\tilde{m_t}^{(\ell)} = \frac{1}{\tilde{w_t}^{(\ell)}} \sum_{i \in L} w_t^{(i)} x_t^{(i)}$
7      $\tilde{P_t}^{(\ell)} = \frac{1}{\tilde{w_t}^{(\ell)}} \sum_{i \in L} w_t^{(i)} (P_t^{(i)} + (m_t^{(\ell)} - m_t^{(i)})(m_t^{(\ell)} - m_t^{(i)})^\intercal)$
8      $I := I \setminus L$
9  until $I = \emptyset$;
10 if $\ell > J_{max}$ then
11     replace $\{\tilde{w_t}^{(i)}, \tilde{m_t}^{(i)}, \tilde{P_t}^{(i)}\}_{i=1}^{\ell}$ by those of the $J_{max}$ Gaussians with largests
         weights.
output: $\{\tilde{w_t}^{(i)}, \tilde{m_t}^{(i)}, \tilde{P_t}^{(i)}\}_{i=1}^{\ell}$ as pruned Gaussian components.
```

2.6.4 State extraction

In the Gaussian mixture representation of the posterior intensity v_t, extraction of multiple-target state estimates becomes straightforward as the means of the constituent Gaussian components serve as local maxima of v_t, provided they are reasonably well separated. It is important to note that after prunning Algorithm 3, closely spaced Gaussian components have been merged. Since the height of each peak is influenced by both weight and covariance, selecting the $\hat{N}_t$ highest peaks of might yield state estimates corresponding to Gaussians with weak weights. This outcome is undesirable as it implies a small expected number of targets associated with these peaks, despite their large magnitudes. A preferable approach is to choose the means of Gaussians with weights exceeding a certain threshold T_plot. This state estimation method for the Gaussian mixture PHD filter is summarized in Algorithm 4.

Algorithm 4: Pseudocode for the Gaussian mixture PHD filter - State extraction

```
given  : $\{w_t^{(i)}, m_t^{(i)}, P_t^{(i)}\}_{i=1}^{J_t}$, and $T\_plot$ a selecting threshold
         Set $\hat{X}_t = \emptyset$.
1 for $i = 1, ..., J_t$ do
2     if $w_t^{(i)} > T\_plot$ then
3         for $j = 1, ..., round(w_t^{(i)})$ do
4             update $\hat{X}_t := [\hat{X}_t, m_t^{(i)}]$
output: $\hat{X}_t$ as the multi-target state estimate.
```

2.7 Problem of trajectories

The PHD filter is integral to the assumed density filtering framework, propagating a specific type of multitarget density across prediction and update steps. The PHD filter operates on a Poisson multitarget density, where the set's cardinality follows a Poisson distribution, and its elements, for each cardinality, are independent and identically distributed (iid).

One of the main advantages of the PHD filter is its low computational burden, as it circumvents the measurement-to-target association problem. However, they exhibit relatively lower performance in certain scenarios and do not generate tracks, representing sequences of target states belonging to the same target.

Attempts to address track building for PHD filter were made by introducing labels to the target states. Nevertheless, resulting labeled Poisson and labeled iid cluster densities encounter complete confusion in the label-to-target association, rendering them ineffective for track formation.

This method led to problems when merging the individual components into one. A measurement history was introduced to tackle this challenge, allowing two targets with different labels to be merged. More detailed information about the solution will be described in the next section. Another method is to use the trajectory PHD filter, which is described in [17].

Chapter 3

Real life data

In this chapter, we will discuss the data used for testing my algorithm described in Section 4. The algorithm's performance is often evaluated on artificially generated data, which can show how well it estimates the state variables. Nevertheless, the theoretical properties of the PHD filter are well known. In this thesis, we limit ourselves to real-world data describing The movement of birds, whether its speed, direction, position in latitude and longitude, identification of the individual, and information regarding the measuring equipment [18].

3.1 Introduction to the dataset

For this thesis, the dataset BOP_RODENT was chosen. This dataset focuses on rodent-specialized birds of prey (Circus, Asio, Buteo) in Flanders (Belgium). This dataset is a collection of animal tracking data published by the Research Institute for Nature and Forest (INBO). Animal tracking data were collected by LifeWatch GPS tracking network for large birds [19]. The tracking devices used are developed by Ornitela [20]. One of these tracking devices can be seen in Figure 3.1, which is attached to the back of a herring gull.

This dataset was chosen for this thesis precisely because it contains the complete trajectories of individuals, which is exactly what we will need for the experiments.

3.2 Data collection

GPS trackers equipped with tri-axial accelerometers have revolutionized bird behavior and migration pattern monitoring since 2013. The UvA-BiTS GPS trackers, employed on over 240 birds, transmit data to ground stations, which then relay the information to the UvA-BiTS data platform developed by LifeWatch NL. Ground stations are strategically positioned near breeding colonies or based on sightings of tagged birds, such as birds of prey that do not breed in colonies. However, migration data retrieval is contingent upon the birds' return, limiting the accessibility of such data.

In recent years, the advent of 3G GSM transmitters has offered a more advanced solution, especially for medium-sized bird species. These transmitters, exemplified by Or-

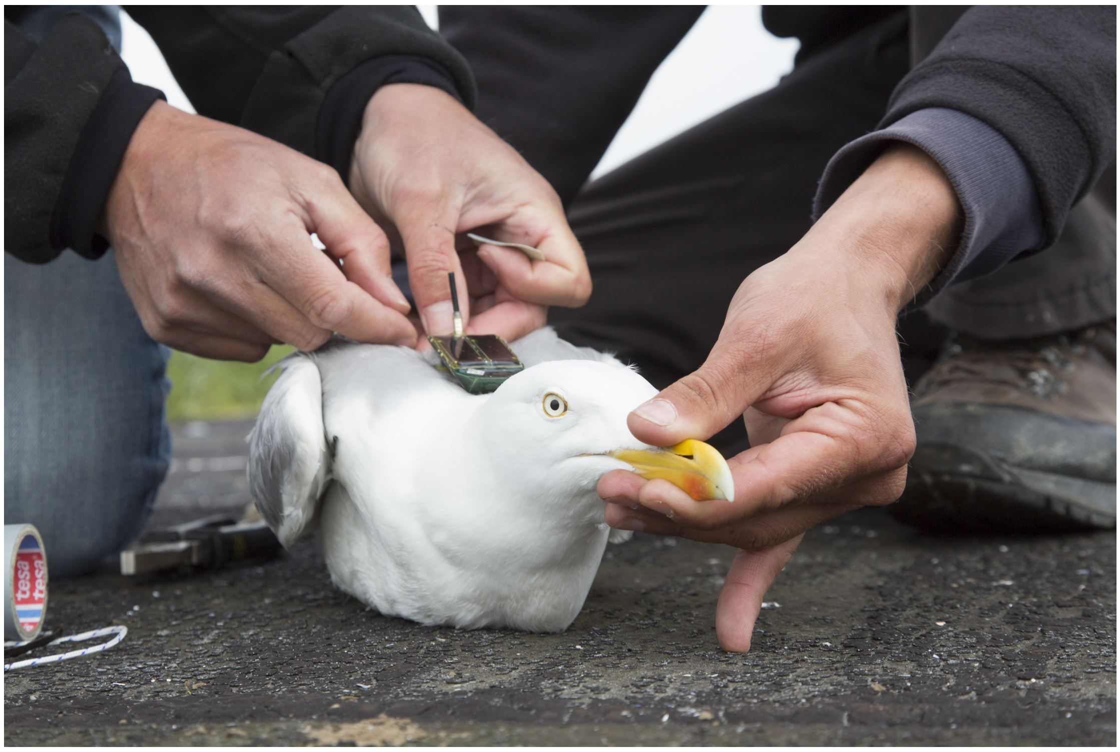


Figure 3.1 The UvA-BiTS GPS tracker attached on the herring gull. (Source: [21])

nitela's OrniTrack transmitters since 2019, obviate the need for ground stations by directly uploading data to a central server via the telecom network. Real-time data transmission capabilities are particularly advantageous during migration or when birds do not revisit their capture site.

Data collected by the trackers include the bird's position, altitude, air temperature, and activity level measured through accelerometers. UvA-BiTS trackers store data locally until the transmission is possible, especially during migration periods. Researchers can remotely configure measurement settings for each tracker using ground station software, with the data eventually uploaded to a central database at IBED. Ornitela trackers, on the other hand, transmit data directly to Ornitela's central database, accessible via the OrniTrack Control panel for registered users.

3.3 Dataset description

The study has been operational since 2020. The dataset contains records of 35 individuals of 5 bird species. This dataset was collected to observe bird habits and behavior during migration. The dataset contains a total of 2316561 records. The bird individual that has the fewest records has only 4217 records. On the other hand, the individual with the most records has 248190 records. A large number of records may not be an advantage in our application. Because birds were also observed outside their migration, a bird stays in its place of occurrence for most of the observation time. The dataset contains 30 columns containing information about the position, speed, direction of movement, altitude, sensor battery, unique individual identifiers, and more. All columns and their data type can be seen in Figure 3.2.3.2. For the purpose of this thesis, we re-

```
 #   Column                           Dtype
---  ------                           -----
 0   event-id                         int64
 1   visible                          bool
 2   timestamp                        object
 3   location-long                    float64
 4   location-lat                     float64
 5   acceleration-raw-x               float64
 6   acceleration-raw-y               float64
 7   acceleration-raw-z               float64
 8   bar:barometric-height            float64
 9   battery-charge-percent           int64
 10  battery-charging-current         float64
 11  external-temperature             float64
 12  gps:hdop                         float64
 13  gps:satellite-count              int64
 14  gps-time-to-fix                  float64
 15  ground-speed                     float64
 16  heading                          float64
 17  height-above-msl                 float64
 18  import-marked-outlier            bool
 19  gls:light-level                  float64
 20  mag:magnetic-field-raw-x         float64
 21  mag:magnetic-field-raw-y         float64
 22  mag:magnetic-field-raw-z         float64
 23  orn:transmission-protocol        object
 24  tag-voltage                      float64
 25  sensor-type                      object
 26  individual-taxon-canonical-name  object
 27  tag-local-identifier             int64
 28  individual-local-identifier      object
 29  study-name                       object
```

Figure 3.2 All the dataset columns.

strict ourselves to the columns timestamp, location-long, location-lat, individual-taxon-canonical-name, and tag-local-identifier. The timestamp column contains information about the time at which the individual's measurement was recorded. The format of this column is YYYY-MM-DD HH:MM:SS. Location-long and location-lat contain location measurements of the individual. The individual-taxon-canonical-name column contains the taxon canonical name of the bird. In this dataset, we have only 5 species of birds, namely Circus pygargus, Circus aeruginosus, Asio flammeus, Circus cyaneus, and Buteo buteo. Photos of individual birds can be seen in Figure 3.3. The last column contains a unique identifier for each individual, allowing us to distinguish them.

For each individual, the measurements come at different intervals. One individual has measurements every seven minutes, and another individual every hour. Due to this fact, it was necessary to preprocess the data.

3.4 Data preprocessing

During the data preprocessing, several tasks had to be performed to make the dataset more straightforward and usable for our purposes. The first step during preprocessing

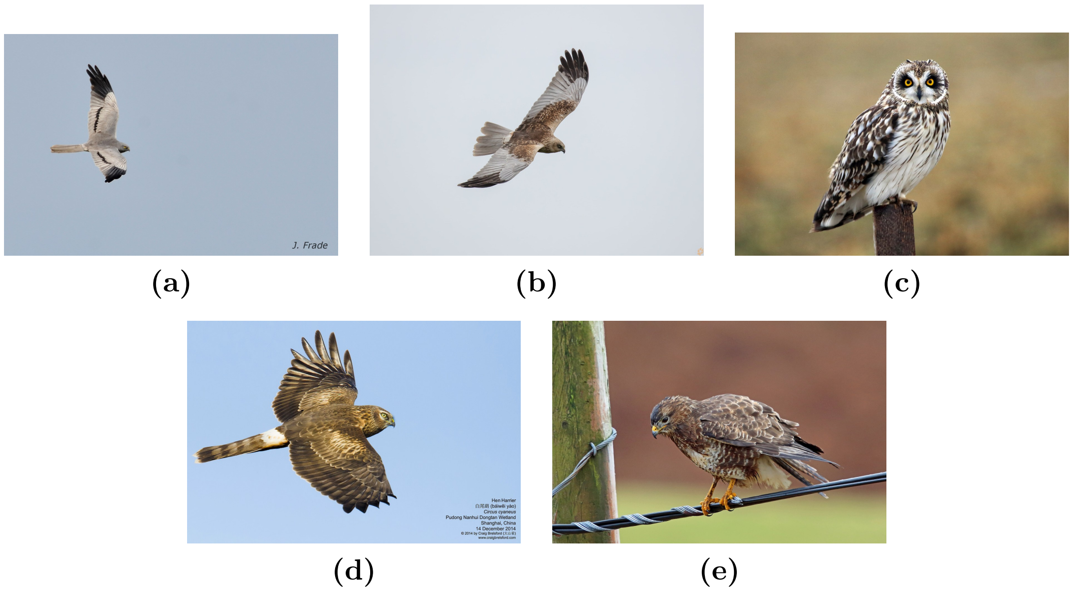


Figure 3.3 Photos of all birds listed in the dataset. **(a)** Circus pygargus. (Source: [22]) **(b)** Circus aeruginosus. (Source: [23]) **(c)** Asio flammeus. (Source: [24]) **(d)** Circus cyaneus. (Source: [25]) **(e)** Buteo buteo. (Source: [26])

was to remove redundant columns, thus reducing the total amount of data to be stored. Further preprocessing is more challenging, so we will discuss it in separate sections.

3.4.1 tag-local-identifier column

This column contains a unique six-digit number that represents the individual. To make these numbers more straightforward to navigate, a new artificial column bird_names was created, where each individual was assigned a name. The assignment of each name to each identifier can be seen in Table 3.1.

3.4.2 timestamp column

As mentioned in the previous text, we receive measurements from each individual at different intervals. The individual with the most significant time spacing has a few days between measurements. Therefore, all records in the dataset are grouped by one day. If an individual received any number of measurements during a day, we only use the last one closest to the next day. This change will not affect the results because the algorithm will work with the fact that every two steps are separated by one day apart. By day, we mean 24 hours. ¡ For this modification, a new column *time_step* was created that contains the time index when the measurement will be used for the algorithm. As shown in Figure 3.4, at time step $t = 1080$, the algorithm gets seven measurements from seven individuals. We can also see that all data come from an interval of one day.

tag-local-identifier	**bird_names**	**tag-local-identifier**	**bird_names**
193185	Charlie	210205	Greta
193683	Baby	210206	Ava
193949	Coco	210207	Jenny
193980	Clive	210208	Pikachu
200085	Max	211048	Opal
200570	Kiwi	211049	Sunny
202039	Jasper	211050	Bella
202040	Mango	211051	Ozzy
202041	Buddy	211052	Teal
202042	Daisy	212925	Penny
202056	Pepper	212928	Paloma
203242	Tweety	213911	Susan
210196	Skittles	213913	Chloe
210198	Phoenix	224079	Tori
210200	Aja	224080	Wyatt
210202	Cher	224081	Yara
210204	Ted	224082	Noah
230618	Barry		

Table 3.1 mapping of tag-local-identifier to bird_names

3.4.3 Removal of immobile birds

As part of the data preprocessing, for our purposes, we decided to remove eight individuals who had not moved a significant distance over the entire time period. This distance was set to 50 km to suppress unnecessary creation of new targets for a bird that isn't going anywhere anyway. For example, Daisy and Pepper were removed. Their movement was not large enough, as shown in Figure 3.5.

Finally, the entire dataset was sorted by the *time_step* column to get the order in which we want our algorithm's measurements to come in. After all previous modifications, the resulting dataset is shown in Figure 3.6.

3.5 Trajectories of birds

As already mentioned, each individual has the path that they took during migration. Some have a longer one, and some have a shorter one. In Figure 3.7, we show all the birds' trajectories for the preprocessing dataset. This gives us an idea of the scale at which we are moving.

	timestamp	location-long	location-lat	individual-taxon-canonical-name	tag-local-identifier	bird_names	time_step
265743	2023-03-03 06:43:05	-11.646406	16.244186	Circus aeruginosus	193980	Clive	1080
711855	2023-03-03 01:23:21	-9.350202	15.061477	Circus pygargus	202040	Mango	1080
1714151	2023-03-03 06:50:31	3.666967	51.246891	Circus cyaneus	212928	Paloma	1080
1169205	2023-03-03 06:59:46	-15.907057	16.029322	Circus aeruginosus	210206	Ava	1080
1169206	2023-03-03 06:59:46	-15.907057	16.029322	Circus aeruginosus	210206	Ava	1080
1354231	2023-03-03 06:48:21	5.901513	52.515900	Circus cyaneus	213913	Chloe	1080
716518	2023-03-02 12:56:04	5.882195	50.961643	Circus cyaneus	200085	Max	1080
2305733	2023-03-03 06:56:47	-4.222232	13.067744	Circus pygargus	224082	Noah	1080
1563716	2023-03-03 06:56:49	-15.078161	16.864468	Circus aeruginosus	210207	Jenny	1080
1781754	2023-03-03 06:56:24	-5.217740	14.500261	Circus pygargus	213911	Susan	1080
1317268	2023-03-03 06:33:48	-15.648169	18.335373	Circus aeruginosus	210204	Ted	1080
2146714	2023-03-03 06:56:29	-3.875686	15.423932	Circus pygargus	224080	Wyatt	1080
1436901	2023-03-03 03:40:35	-16.819962	14.350475	Circus aeruginosus	210205	Greta	1080
2093251	2023-03-03 06:55:44	-10.011239	14.711814	Circus pygargus	224079	Tori	1080
569218	2023-03-03 06:26:06	-7.463387	15.159710	Circus aeruginosus	200570	Kiwi	1080
2209642	2023-03-03 06:59:51	-4.712117	15.206722	Circus pygargus	224081	Yara	1080

Figure 3.4 All records at time step $t = 1080$.

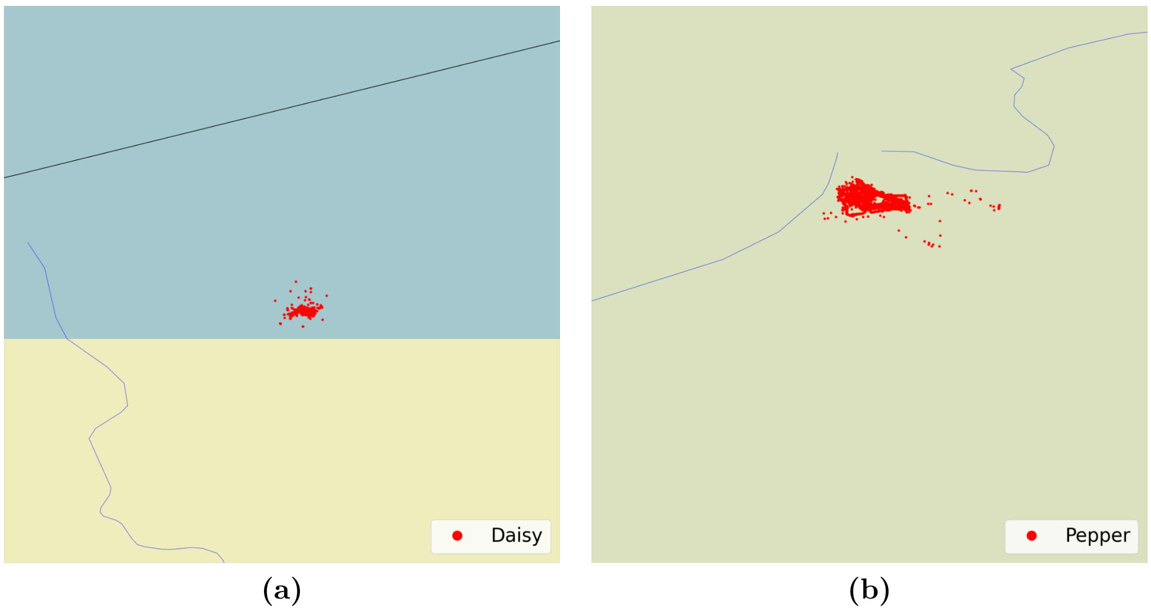
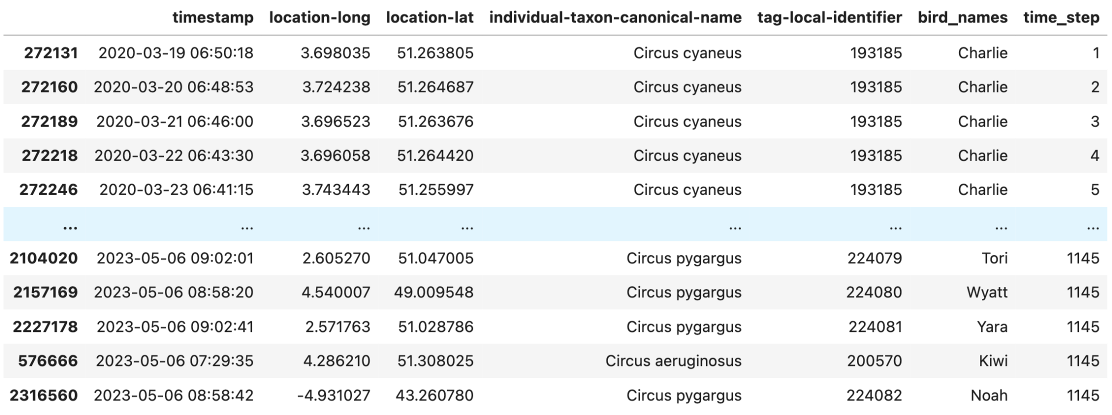


Figure 3.5 Data for two of eight removed individuals. The area depicted is 46 × 46 km in size. **(a)** Daisy, **(b)** Pepper



	timestamp	location-long	location-lat	individual-taxon-canonical-name	tag-local-identifier	bird_names	time_step
272131	2020-03-19 06:50:18	3.698035	51.263805	Circus cyaneus	193185	Charlie	1
272160	2020-03-20 06:48:53	3.724238	51.264687	Circus cyaneus	193185	Charlie	2
272189	2020-03-21 06:46:00	3.696523	51.263676	Circus cyaneus	193185	Charlie	3
272218	2020-03-22 06:43:30	3.696058	51.264420	Circus cyaneus	193185	Charlie	4
272246	2020-03-23 06:41:15	3.743443	51.255997	Circus cyaneus	193185	Charlie	5
...	...	...	...	...	...	...	...
2104020	2023-05-06 09:02:01	2.605270	51.047005	Circus pygargus	224079	Tori	1145
2157169	2023-05-06 08:58:20	4.540007	49.009548	Circus pygargus	224080	Wyatt	1145
2227178	2023-05-06 09:02:41	2.571763	51.028786	Circus pygargus	224081	Yara	1145
576666	2023-05-06 07:29:35	4.286210	51.308025	Circus aeruginosus	200570	Kiwi	1145
2316560	2023-05-06 08:58:42	-4.931027	43.260780	Circus pygargus	224082	Noah	1145

Figure 3.6 Final dataset after all the data preprocesing.

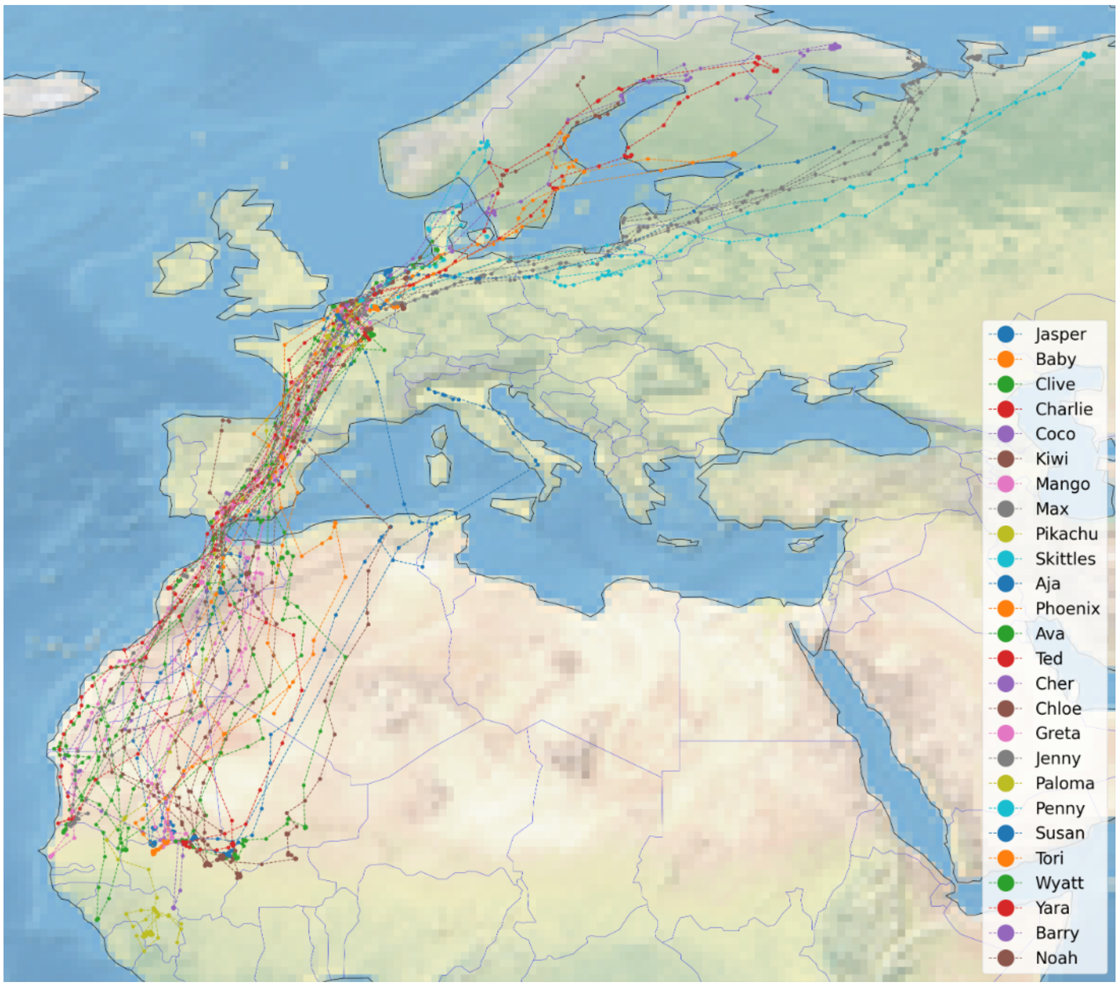


Figure 3.7 Trajectories for all individuals after data processing.

Chapter 4

Implementation

In this chapter, we will examine the actual implementation of the PHD filter. First, we explain the Gaussian class that represents a single potential target. Next, we describe the entire run of the PHD filter. In the last section, we will look at the actual implementation of the PHD filter and all of its methods needed for proper functionality.

4.1 Gaussian class

The Gaussian class represents a potential target. To represent only the Gaussian distribution, we would only need two parameters. The parameter m is the mean of the Gaussian (in our case, the vector [longitude, latitude, longitude velocity, latitude velocity]) and the matrix P (in our case, with size 4x4). For use in the PHD filter, we will add additional variables to make it easier to work with the filter itself later.

4.2 PHD class

This whole class represents the algorithm described in Section 2.6. Now, let us look at the individual parameters. The parameter $targets$ is a list containing the individual instance of the Gaussian class, i.e., it represents a Gaussian mixture. The parameter $sources$ contains a list of all the places where the target can be born. In our case, these will be the nesting sites. The parameters A, H, R, and Q are matrices from the Kalman filter presented in Section 1.4.Ps and Pd represent survival and detection probability, respectively. The default value for the survival probability is 0.99, and the detection probability is 0.98. The parameter lam is for the clutter intensity, and T is the pruning threshold. U is the merging threshold. The T_plot is the threshold for visualizing a given target (any target with a weight of at least T_plot will be shown in the visualization). The $label\_counter$ parameter numbers the target with a unique value, allowing us to track its trajectory. The $gamma$ parameter determines the maximum Mahalanobis distance from the measurement to the target to pass the gate of the filter. $measurements$ are all measurements that have passed through the Gate. The w_th denotes the minimum required weight of the target to create a new trajectory or to add its position to an existing trajectory created from that target. The $trajectories$ contains all tra-

```
class Gaussian():
    def __init__(self, w=0, m=[], P=[], label=0, z_history=[], z_history_len=0):
        self.w = w # Weight of~Gaussian
        self.m = m # Mean of~Gaussian
        self.P = P # Covariant matrix
        self.Eta = None
        self.S = None
        self.K = None
        self.P_new = None
        self.w_old = None
        self.label = label
        self.z_history = z_history
        self.z_history_len = z_history_len
```

Code listing 4.1 Gaussian class

jectories that the filter has created. It is represented by a dictionary, where the key is the label of the target, and the value is the list of target positions with the given label. The last parameter is *trajectory_prune*. This parameter tells how many records a given trajectory must have to be considered valid.

This class represents the entire model for the PHD filter. If we want to work with it, we must first initialize it. We must specify the matrix F, Q, H, R, and the clutter intensity *lam*. We may or may not need to initialize the others because the default values are sufficient for our many cases.

After initialization, we are ready to start using the filter. The standard filter, as mentioned earlier, runs in two steps. Predict and Update. We have modified PHD filter, and so we have four steps. Predict, Gate, Update, MergeLabels. We will show each of them separately. The overall run of the algorithm looks like this.

4.2.1 Predict

The first step of the algorithm is the prediction step. This step consists of two parts. One produces a prediction for any previous target. This part is described by Equations (2.17), (2.18), (2.19). In the second part, the algorithm initializes new instances of the Gaussian class. In other words, at each location where we expect a new target to be created, we initialize one target with a low weight, a reasonable covariance matrix, and a new label.

4.2.2 Gate

Right after the prediction function is the *Gating* function. This function takes as a parameter a list of all measurements for a given time step from the sensor, regardless of where it is located. Then, the function iterates over all the targets and finds all measurements with a Mahalanobis distance shorter than the gamma parameter. This measurements are then used in the Update step.

```
class PHD():
    def __init__(self, targets=[], sources=[], A=[], Q=[], H=[], R=[],
                 Ps=0.99, Pd=0.98, lam=0, U=4, T=1.e-6, T_plot=0.5,
                 gamma=9.2, w_th=0.5, trajectory_prune=10):
        self.targets = targets
        self.sources = sources
        self.A = A
        self.H = H
        self.R = R
        self.Q = Q
        self.Ps = Ps
        self.Pd = Pd
        self.lam = lam
        self.T = T
        self.U = math.sqrt(U)
        self.T_plot = T_plot
        self.label_counter = 1
        self.gamma = gamma
        self.measurements = set()
        self.w_th = w_th
        self.trajectories = defaultdict(list)
        self.trajectory_prune = trajectory_prune
```

Code listing 4.2 PHD class

```
PHDFilter = PHD(sources=sources, F=F, Q=Q, H=H, R=R, lam=lam)
```

Code listing 4.3 Initialization of PHD filter.

```
PHDFilter.predict(cov=cov)
PHDFilter.gate(measurements=measurements)
PHDFilter.update()
PHDFilter.mergeLabels()
```

Code listing 4.4 One run of PHD filter at time t.

```python
def predict(self, cov):
        # part one
        for target in self.targets:
            target.w = self.Ps * target.w
            target.m = self.A @ target.m
            target.P = self.Q
                     + self.A @ target.P @ self.A.T
        # part two
        for source_spot in self.sources:
            self.targets.append(
                                Gaussian(w=0.2,
                                         m=deepcopy(source_spot),
                                         P=cov,
                                         label=self.label_counter))
            self.label_counter += 1
```

Code listing 4.5 Prediction function

```python
def gate(self, measurements):
        self.measurements.clear()
        for target in self.targets:
            for z in measurements:
                d = mahalanobis(z, self.H @ target.m,
                        np.linalg.inv(self.H @ target.P @ self.H.T + self.R))
                if d <= self.gamma:
                    self.measurements.add(z)
```

Code listing 4.6 Gating function

4.2.3 Update

This is the most important function of the whole algorithm. This part consists of three parts. In the first part, we precompute all the auxiliary matrices. The details of these matrices are described by Formulas (1.27), (1.29), (1.30), (1.32). The second part is an update of all current targets in case of a misdetection. This is the case where all measurements are clutter. The third part is the actual updating of the individual components by measurement. It is important to note that each measurement updates all components of the prediction step. This increases the number of components exponentially.

For this reason, the $Merge$ function will be followed, which merges components with too much weight and prunes components with low weight.Maintaining each component's current measurement history is essential to this implementation. We will need this history in the mergeLabels function. With this history, we can decide if two different components with different labels can come from the same target. It is essential to set the length of the measurement history correctly. If the history is too long, many components with different labels will be in the same position. This will increase our computational complexity considerably. If the history is too short, trajectory information may be lost. This implementation uses a history of 2 measurements because it is the ideal compromise between computational power and results.

4.2.4 Merge

This function in the classical PHD filter merges all components within a certain distance from each other and thus can very likely come from the same target. In our modified PHD filter, this function is made more difficult because we use labels for individual components. This assumes that each component with a different label comes from a different target. Therefore, we can only target components close enough to each other and have the same label. Thus, we only target components that can come from the same target.

4.2.5 Merge Labels

As described above, the Merge function only merges components with the same label. This could lead to an exponential increase in the number of components, which needs to be avoided. That is why the Merge labels function was created. This function merges all components that are close enough to each other and have the same history of assigned measurements. We look at the last two measurements that led to the current state of this component. Then, all targets need to be assigned a unique label. Two components could have had the same label after the Update step but were not close enough to be matched together. After all these, a component will contribute to the trajectories if it has a specific weight given by the w_th parameter.

```
def update(self):
        # part one
        for target in self.targets:
            target.Eta = deepcopy(self.H @ target.m)
            target.S = deepcopy(self.R + self.H @ target.P @ self.H.T)
            target.K = deepcopy(target.P @ self.H.T @ np.linalg.inv(target.S))
            target.P_new = deepcopy((np.eye(target.K.shape[0]) - target.K @ self.H)
                                    @ target.P)

        # part two
        for target in self.targets:
            target.w_old = target.w
            target.w = (1 - self.Pd) * target.w

        # part three
        targets_copy = deepcopy(self.targets)
        for z in self.measurements:
            new_targets = []
            sum_weight = 0
            z = np.array(z)
            for target in targets_copy:
                new_target_w = self.Pd * target.w_old *
                               mvn.pdf(z, target.Eta.flatten(), target.S)
                new_target_m = deepcopy(target.m + target.K @ (z - target.Eta))
                new_targets.append(
                                    Gaussian(w=new_target_w,
                                             m=new_target_m,
                                             P=target.P_new,
                                             label=target.label,
                                             z_history=deepcopy(target.z_history),
                                             z_history_len=target.z_history_len))
                sum_weight += new_target_w

            for new_target in new_targets:
                new_target.w = new_target.w / (self.lam + sum_weight)
                if new_target.z_history_len == 2:
                    new_target.z_history.pop(0)
                    new_target.z_history_len -= 1
                new_target.z_history.append(z)
                new_target.z_history_len += 1
            self.targets.extend(deepcopy(new_targets))
```

Code listing 4.7 Update function

```
I = deepcopy(self.targets)
I[:] = [x for x in I if x.w > self.T]
self.targets.clear()
while(I != []):
    j = max(enumerate(I), key=lambda x: x[1].w)[0]
    L = []
    for i in range(len(I)):
        if (I[i].label == I[j].label)
            and
            (mahalanobis(I[i].m, I[j].m, np.linalg.inv(I[i].P)) < self.U):
            L.append(I[i])
    w_L = sum([l.w for l in L])
    m_L = deepcopy((1 / w_L) * sum([l.w * l.m for l in L]))
    P_L = deepcopy((1 / w_L) * sum([l.w *
                               (l.P + np.outer((m_L - l.m), (m_L - l.m)).T)
                                for l in L]))
    self.targets.append(
                    Gaussian(w=w_L,
                             m=m_L,
                             P=P_L,
                             label=I[j].label,
                             z_history=deepcopy(I[j].z_history),
                             z_history_len=I[j].z_history_len))
    for element in L:
        if element in I:
            I.remove(element)
```

Code listing 4.8 Merge function

```python
def mergeLabels(self):
        I = deepcopy(self.targets)
        I[:] = [x for x in I if x.w > self.T]
        self.targets.clear()
        while(I != []):
            j = max(enumerate(I), key=lambda x: x[1].w)[0]
            L = []
            labels = []
            for i in range(len(I)):
                if all([
                np.allclose(x, y) for x, y in zip(I[i].z_history, I[j].z_history)
                ])
                and (mahalanobis(I[i].m, I[j].m, np.linalg.inv(I[i].P)) < self.U):
                    L.append(I[i])
                    labels.append(I[i].label)
            w_L = sum([l.w for l in L])
            m_L = deepcopy((1 / w_L) * sum([l.w * l.m for l in L]))
            P_L = deepcopy((1 / w_L) * sum([l.w *
                                     (l.P + np.outer((m_L - l.m), (m_L - l.m)).T)
                                      for l in L]))
            labels.sort()
            self.targets.append(
                                Gaussian(w=w_L,
                                         m=m_L,
                                         P=P_L,
                                         label=labels[0],
                                         z_history=deepcopy(I[j].z_history),
                                         z_history_len= I[j].z_history_len))
            for element in L:
                if element in I:
                    I.remove(element)

        self.targets.sort(key=operator.attrgetter('w'), reverse=True)
        unique_labels = []
        for target in self.targets:
            if target.label not in unique_labels:
                unique_labels.append(target.label)
            elif target.z_history_len == 2:
                target.label = self.label_counter
                self.label_counter += 1

        for target in self.targets:
            if target.w > self.w_th:
                self.trajectories[target.label].append(target)
```

Code listing 4.9 mergelabels function

Chapter 5
Experiments

In this chapter, we will track birds using the algorithm described above. First, we try tracking only one individual in a clutter-free environment; then, we add clutter. Next, we try to remove measurements to simulate data loss. Further tests will be conducted over multiple targets. We will compare the observed trajectory with the actual trajectory. In a multi-target comparison, we will also test for a clean environment and cluttered environment.

5.0.1 Generalized optimal sub-pattern assignment metric

Before we get into the actual tests, we need to define the metrics that will be used to compare the algorithm's performance.

▶ **Definiton 5.1.** *Let $c > 0, 0 < \alpha \leq 2$ and $1 < p < \infty$. Let $d(x, y)$ denote a metric for any $x, y \in \mathbb{R}^N$ and let $d^{(c)}(x, y) = min(d(x, y), c)$ be its cut-off metric. Let Π_n be the set of all permutations of $\{1, \dots, n\}$ for any $n \in \mathbb{N}$ and any element $\pi \in \Pi_n$ be a sequence $(\pi(1), \dots, \pi(n))$. Let $X = \{x_1, \dots, x_{|X|}\}$ and $Y = \{y_1, \dots, y_{|Y|}\}$ be finite subsets of $\mathbb{R}^N$. For $|X| \leq |Y|$, the GOSPA metric is defined as*

$$d_p^{(c,\alpha)}(X,\ Y) \tag{5.1}$$

$$\triangleq \left(\min_{\pi \in \Pi_{|Y|}} \sum_{i=1}^{|X|} d^{(c)}(x_i,\ y_{\pi(i)})^p + \frac{c^p}{\alpha}(|Y| - |X|) \right)^{\frac{1}{p}}. \tag{5.2}$$

$$\text{If } |X| > |Y|, d_p^{(c,\alpha)}(X,\ Y) \triangleq d_p^{(c,\alpha)}(Y,\ X) \tag{5.3}$$

The definition is taken from [27].

5.0.2 Parameters for experiments

All experiments are based on the CVM model. We will observe birds on a 2D surface. The measurements have a period of 24 hours. The state-space model is characterized as follows: The matrices A and Q have the following form

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Q = q^2 \begin{bmatrix} \Delta t & 0 & \frac{\Delta t}{2} & 0 \\ 0 & \Delta t & 0 & \frac{\Delta t}{2} \\ \frac{\Delta t}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t}{2} & 0 & \Delta t \end{bmatrix},$$

where Δt is the time step. In this case the time step is 24 hours. Parameter q is a parameter for noise matrix Q. We consider that $q = 0.2$.

The measurement matrix H and the measurement noise matrix R are in the following form

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

where r is a parameter for measurement noise matrix R, we consider that $r = 0.4$.

Five main nesting sites were extracted from the data. We will consider these nesting sites as birthplaces of new targets. These nesting sites are at the following longitudes and latitudes $[1.9068316, 50.1637650]^\intercal$, $[2.7221153, 50.9518077]^\intercal$, $[3.5380737, 51.2485191]^\intercal$, $[4.4330171, 51.1495518]^\intercal$, and $[5.0285015, 50.7659454]^\intercal$. New targets are born at these positions with a weight w and scatter matrix P. In addition, we describe everything using the following equations

$$P = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}, \qquad w = 0.1 \tag{5.4}$$

$$m_1 = \begin{bmatrix} 1.9068316 \\ 50.1637650 \\ 0 \\ 0 \end{bmatrix}, m_2 = \begin{bmatrix} 2.7221153 \\ 50.9518077 \\ 0 \\ 0 \end{bmatrix}, \qquad m_3 = \begin{bmatrix} 3.5380737 \\ 51.2485191 \\ 0 \\ 0 \end{bmatrix}, \tag{5.5}$$

$$m_4 = \begin{bmatrix} 4.4330171 \\ 51.1495518 \\ 0 \\ 0 \end{bmatrix}, m_5 = \begin{bmatrix} 5.0285015 \\ 50.7659454 \\ 0 \\ 0 \end{bmatrix}. \tag{5.6}$$

These parameters are used to create new components in the PHD filter.

The parameters for the $Merge$ and $MergeLabels$ functions are as follows: parameter U specifies the maximum distance between two components that can still be merged into one. The parameter T is a threshold, and once the weight of the component w falls below this threshold, the component is removed. The values are the following:

$$U = 9.2, \quad T = 0.00001. \tag{5.7}$$

The gating parameter $gamma$ specifies the distance of the gating area. Parameters w_th and $trajectory\_prune$ are used to set the minimum weight of a component to create a trajectory and the minimum number of entries in the trajectory to consider it valid. Their values are the following:

$$gamma = 9.2, \quad w\_th = 0.5, \quad trajectory\_prune = 50. \tag{5.8}$$

For the last two parameters, we need to set our survival probability p_S and detection probability p_D; their values are following:

$$p_S = 0.99, \quad p_D = 0.98. \tag{5.9}$$

The main performance indicator of our algorithm will be *location_error*, which we get using GOSPA. To use GOSPA, we need to set three parameters c, p, and $alpha$. For our experiments, we choose the parameters as follows

$$c = 0.5, \quad alpha = 2, \quad p = 2. \tag{5.10}$$

5.1 Single target

To experiment with one target, we select two individuals on which to run our algorithm. Namely, Charlie and Ava. Both individuals will make one migration route during their flight. They will fly to their new nesting site and then back. Charlie's trajectory length is 265 steps, and Ava's trajectory length is 267 steps.

The full trajectory for Ava can be seen in Figure 5.1, and the full trajectory for Charlie can be seen in Figure 5.2.

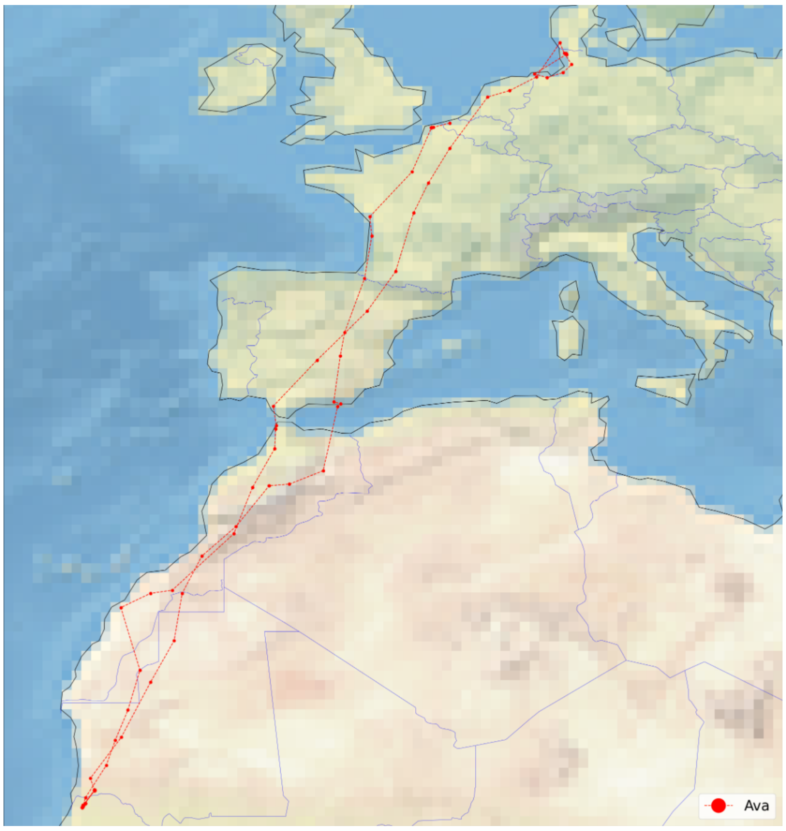


Figure 5.1 True trajectory of a bird named Ava.

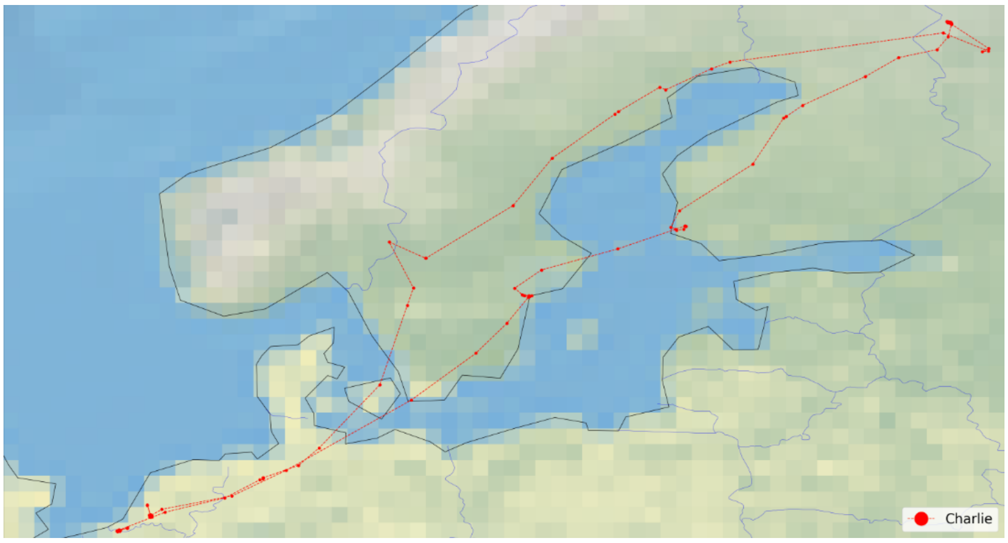


Figure 5.2 True trajectory of a bird named Charlie.

5.1.1 STT - clear environment

We perform the first experiment in a clear environment. This means our filter will only see the true measurements, not the false measurements. The filter will only get one measurement at a time. This experiment has no random variable; therefore, it will be evaluated on a single algorithm run. The entire trajectory of the algorithm for the individual Charlie is shown in Figure 5.3, and the whole trajectory of the algorithm for Ava is shown in Figure 5.4. A comparison of each *location_error* from GOSPA is shown in Figure 5.5.

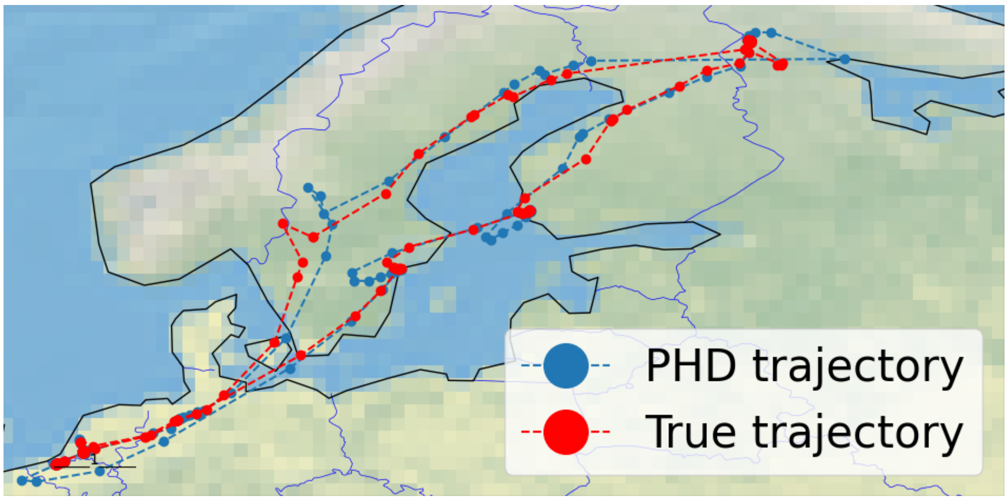


Figure 5.3 Comparison of true and PHD trajectory of Charlie in clear enviroment.

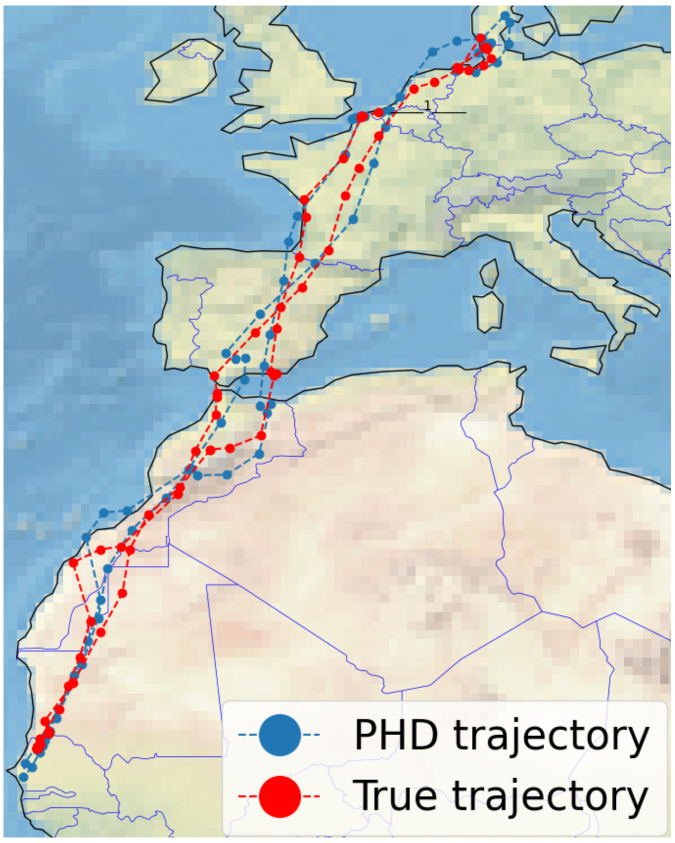


Figure 5.4 Comparison of true and PHD trajectory of Ava in clear enviroment.

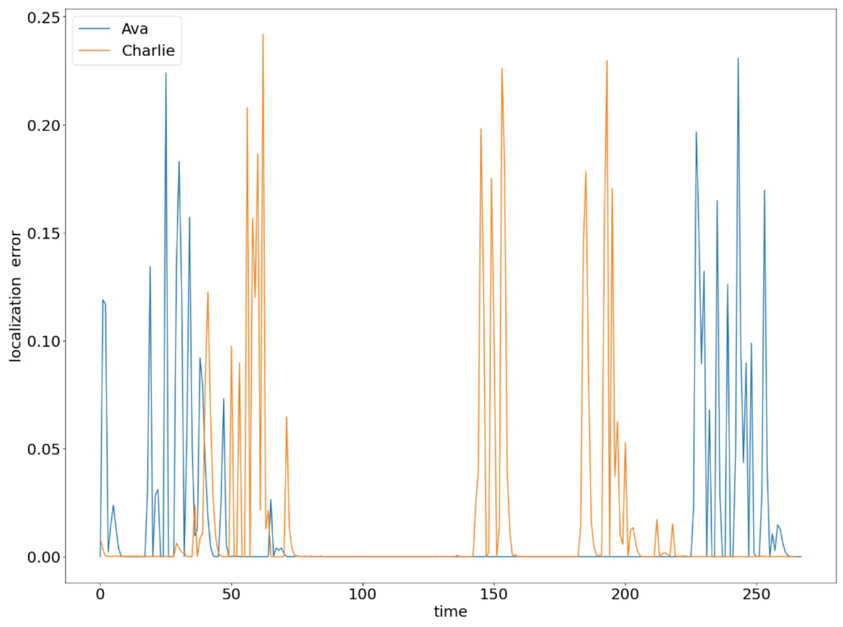


Figure 5.5 *location_error* of Ava and Charlie in clear enviroment.

5.1.2 STT - cluttered environment

Since we have a dataset containing no false measurements, we must add them artificially. Clutter will always be generated around the true measurement. In our implementation, it makes no sense for clutter to exist all over the Earth. The intensity has been set to $Po(6.25)$. This means that we expect 6.25 clutter points at each point in time. For this experiment, we perform ten runs on each individual. We do this because the number and position of clutter are random so that each run may have a different result. Measurements that the filter receives (true + clutter) are shown in Figure 5.6. True measurements are red; clutter is grey.

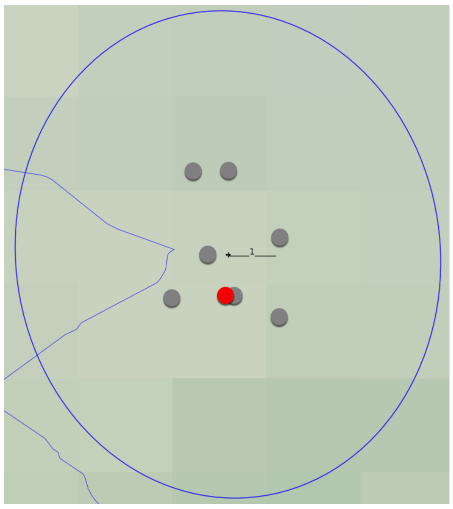


Figure 5.6 Display all measurements received by the the filter at one time step.

Out of ten runs, both individuals had one run with two trajectories instead of one. The depicted runs for each individual can be seen in Figure 5.7 for Charlie and in Figure 5.8 for Ava.

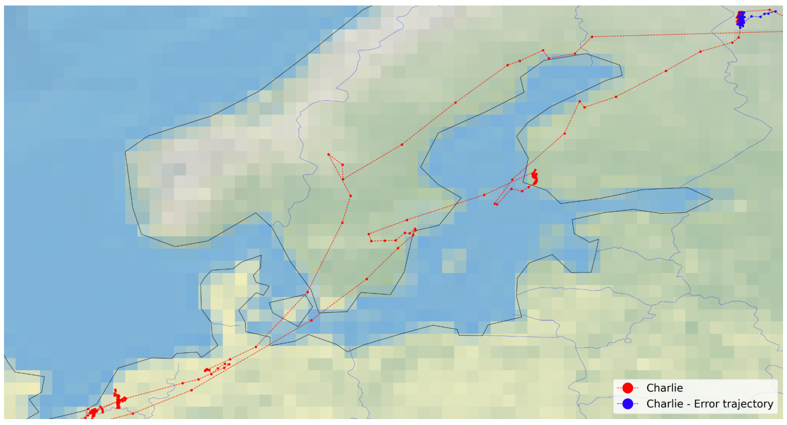


Figure 5.7 Two trajectories created by the PHD filter for the individual Charlie in cluttered enviroment.

In Charlie's case, the second trajectory occurred after migration. In a place where the bird had been for a long time, most of the true and false measurements were within a short distance of each other. Thus, another Gaussian component may have formed that was not merged with the other component and thus formed another trajectory. This trajectory could be removed if we increased the *trajectory_prune* parameter.

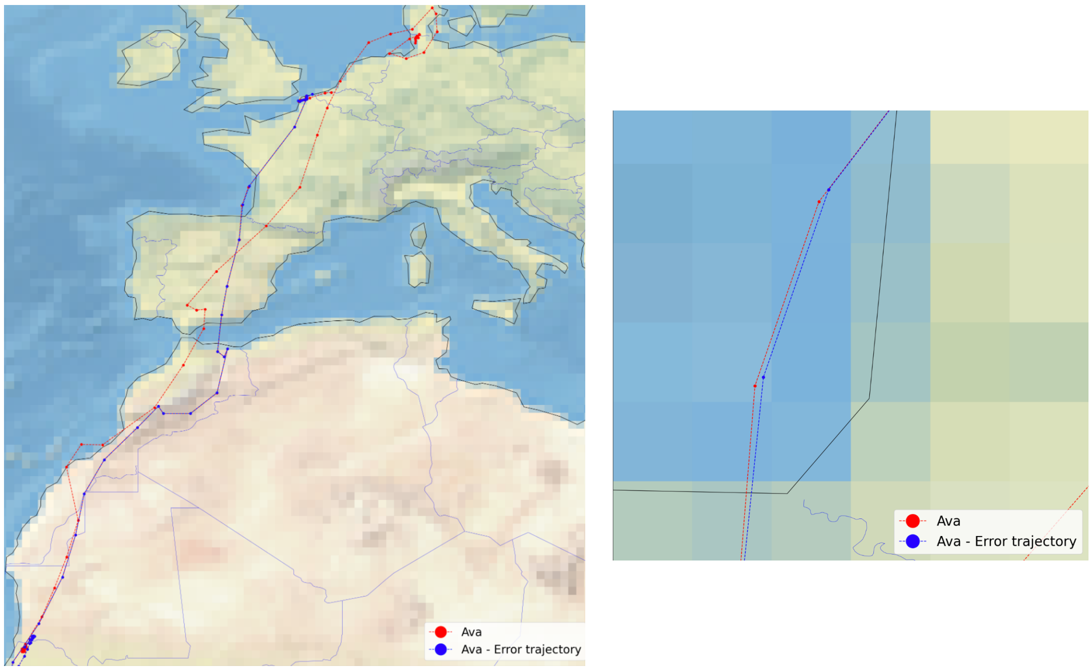


Figure 5.8 Two trajectories created by the PHD filter for the individual Ava in cluttered enviroment and detailed view of trajectory.

Ava's trajectories are both from nesting site to migration destination site. In Figure 5.8, both trajectories are shown. In the first picture, they appear identical. They are not. In the adjacent image, which shows more detail, it can be seen that the trajectories are slightly different.

This is due to the clutter. When each component goes for a different measurement, because this component follows the clutter, it disappears at the point of migration because it follows the false measurement.

Since there were ten runs of the algorithm, the resulting *location_error* from GOSPA shown in Figure 5.9 is the average GOSPA over all runs.

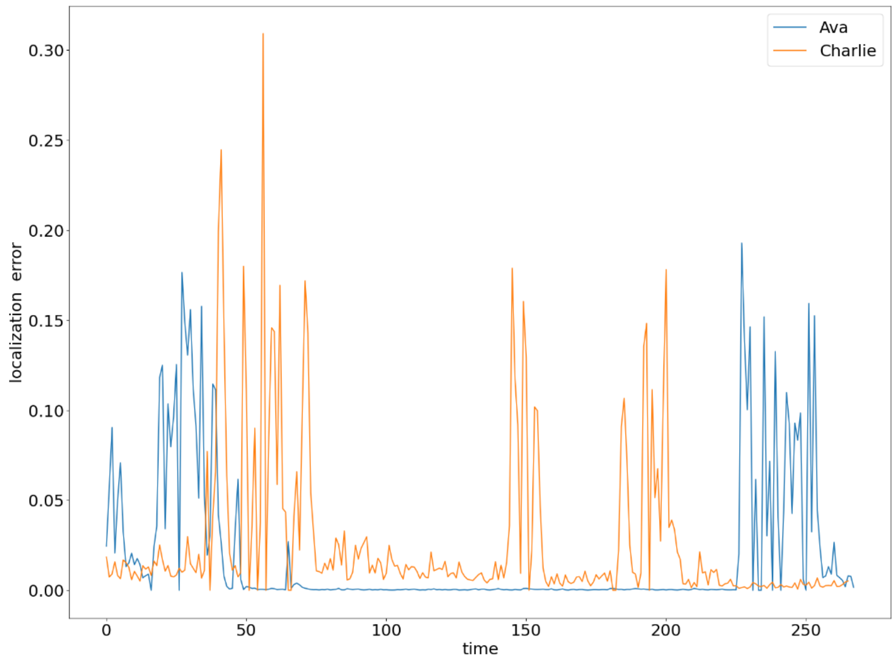


Figure 5.9 *location_error* of Ava and Charlie in a cluttered environment.

5.1.3 STT - missing data

This experiment aims to show how the algorithm copes with missing data. We will perform this experiment only on the individual Ava. The experiment will be conducted in both cluttered and clean environments. For this experiment, the T_plot parameter was set to $T\_plot = 0$ to see what happens in the background when the filter has no data to update.

In the resulting trajectory shown in Figure 5.10, you can see precisely where the filter did not have a measurement, so the trajectory is straight up to the point where it got a measurement and the Gaussian component had sufficient weight to produce a trajectory.

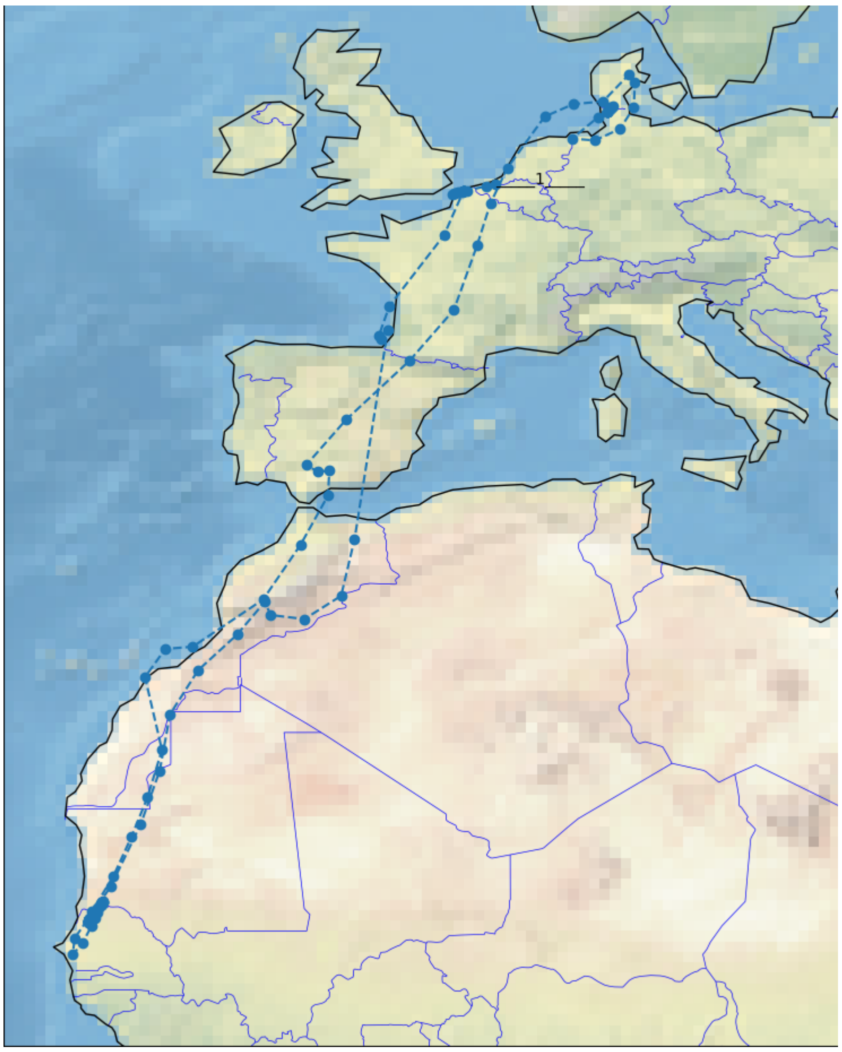


Figure 5.10 Ava's trajectory with missing measurements in clear enviroment.

At times $t = 19$ and $t = 20$, we do not give any measurements to the filter. The evolution of the PHD filter is shown in Figure 5.11. The figure shows the measurements even when we do not use them to update the PHD filter, and this is just for reference to see how the actual trajectory goes. The figure shows precisely how, at steps $t = 17$ and $t = 18$, when the PHD filter has measurements to update, the target is more or less at the exact position. When the data is lost, the target stops moving, and its covariance matrix grows. After the covariance matrix grows so much that the subsequent measurement falls into the validation region, a new target is created at time $t = 23$. This target takes over the label from its parent. Subsequently, the targets are relabeled to avoid having two targets with the same label. This is described in Section 4.2.5. At time $t = 24$, the uncertainty of the original target has increased so much that it would also catch the measurement. Both Gaussian components follow the same target, so they will be merged into one in the following steps.

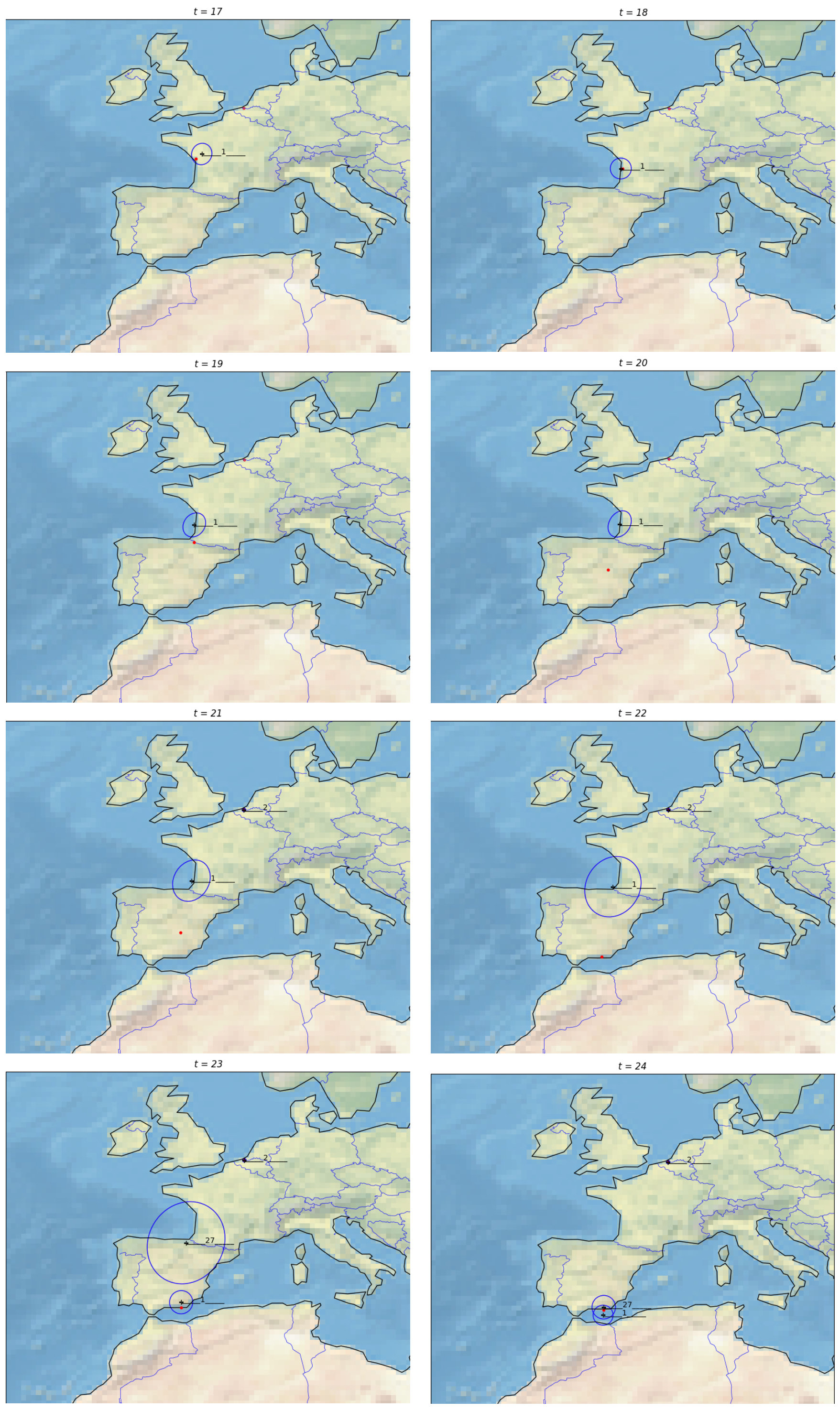


Figure 5.11 Display of the PHD filter run at times $t = 17$ to $t = 24$, with missing measurements in steps between $t = 19$ and $t = 20$.

Now, let's add clutter. After adding clutter, the trajectory evolution does not change. The same situation is repeated, where the covariance matrix increases. And after receiving the measurements, new targets are created. Only here are additional components with different labels created due to clutter, which are then conjugated together. This is shown in Figure 5.13. After step $t = 24$, two components continue. A component with label 1 and a component with label 19. These two components are later merged, and only the component with the label 1 remains. This can be seen in Figure 5.14, where we show a time step $t = 25$ and a time step $t = 72$. The resulting trajectory is shown in Figure 5.12. It is very similar to Figure 5.10. It differs only in minor differences where the components were affected by clutter.

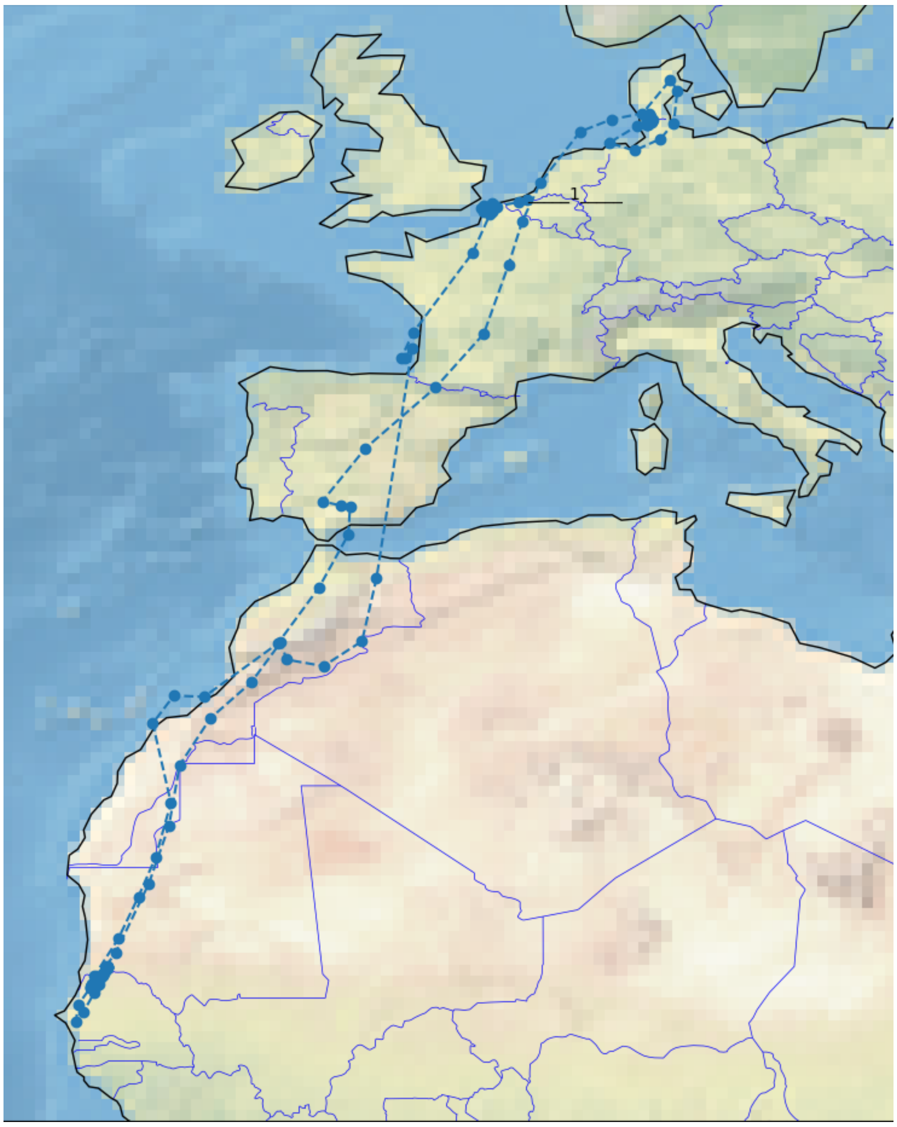


Figure 5.12 Ava's trajectory with missing measurements in cluttered enviroment.

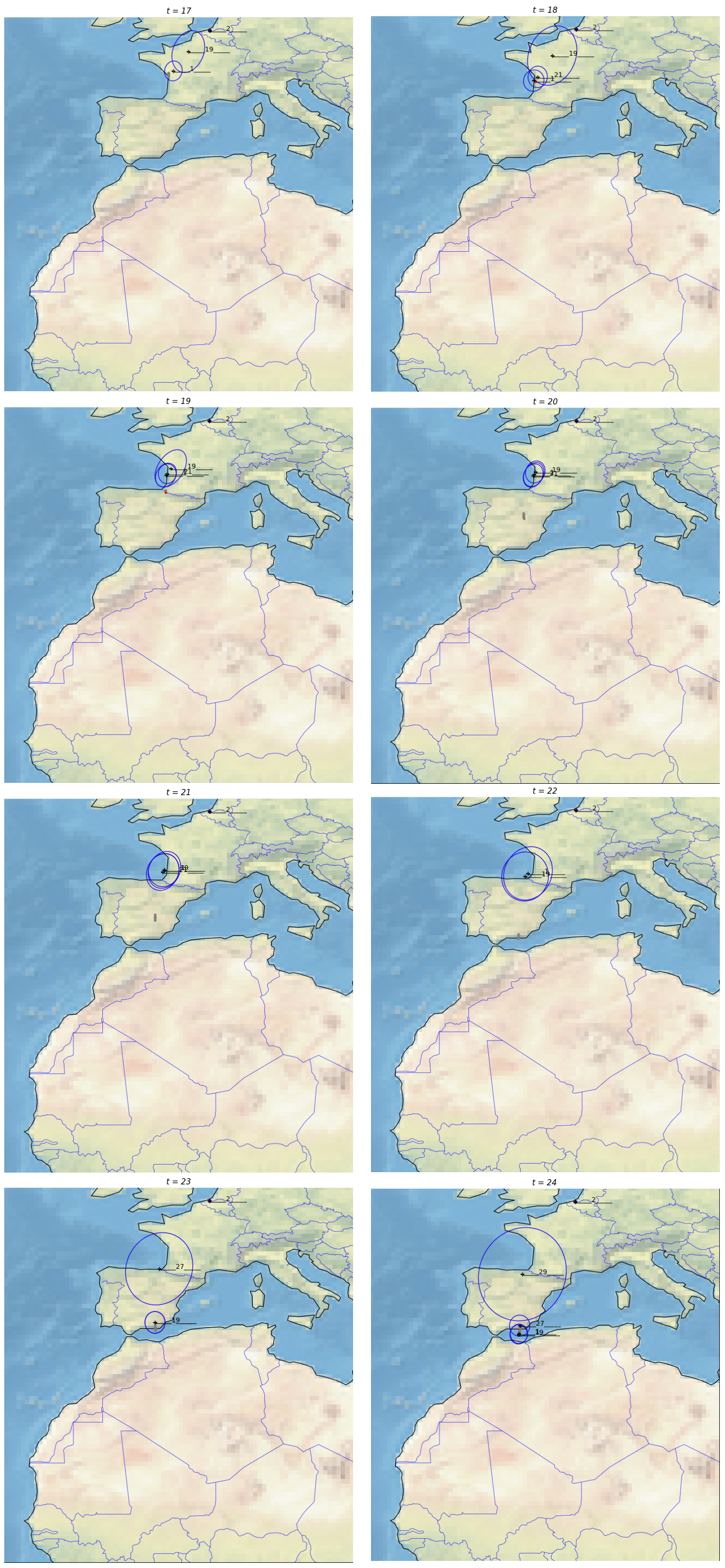


Figure 5.13 Display of the PHD filter run at times $t = 17$ to $t = 24$, with missing measurements in steps between $t = 19$ and $t = 20$ in cluttered enviroment.

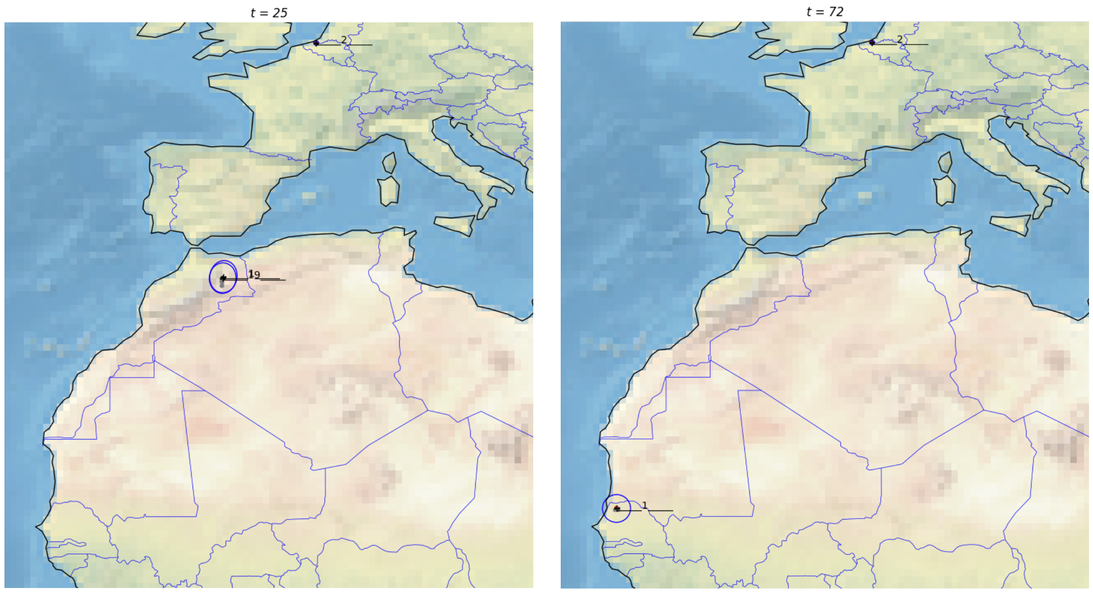


Figure 5.14 Display of the PHD filter run at times $t = 25$ and $t = 72$.

At the end of the single target tracking section, we compare *location_error* across all experiments performed in Figure 5.1. The yellow-shaded part is the time interval when the data was lost.

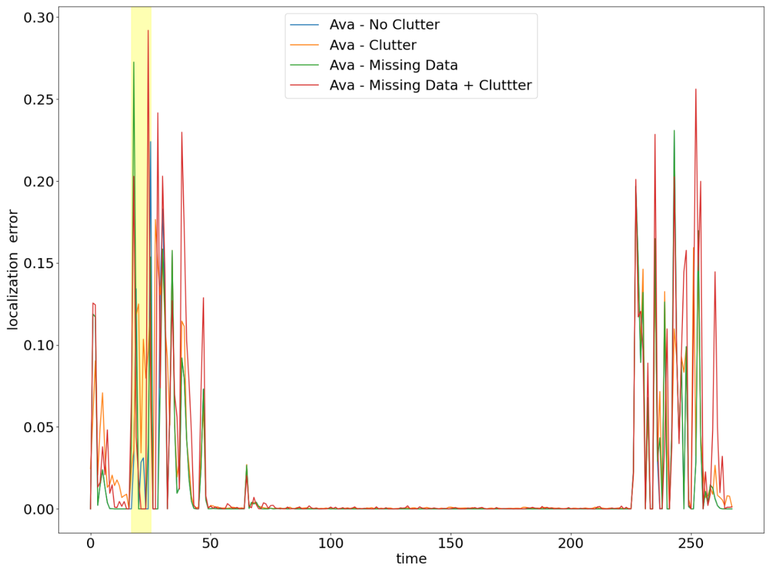


Figure 5.15 *location_error* of all experiments in the STT section.

5.2 Multiple-targets

We have now shown that the PHD filter can track a single target. However, many other algorithms, which are much more straightforward, can do this. Therefore, we will move on to the Multi-Target Tracking section, in which is the power of the PHD filter.

For all experiments in the MTT section, we will track 6 targets. Namely: Baby, Skittles, Penny, Charlie, Ava, and Greta. With up to 4 targets at a single time step. Figure 5.16 depicts all the actual trajectories in a single image.

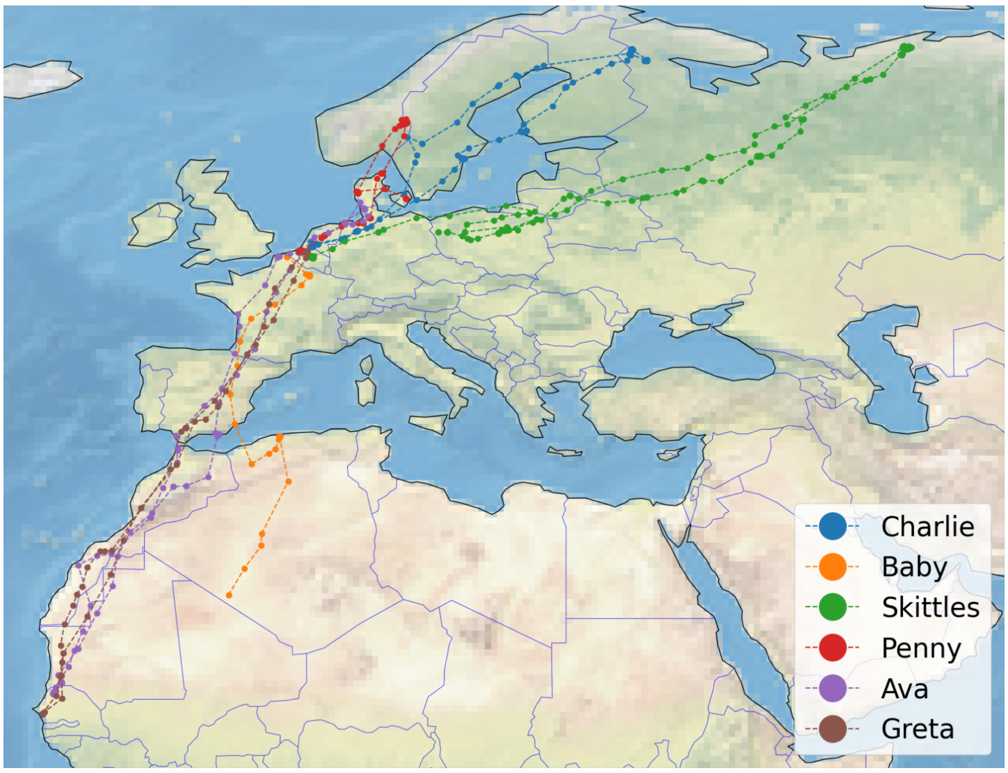


Figure 5.16 True trajectories of six individuals.

5.2.1 MTT - clear environment

The first experiment is to run the PHD filter in a clean environment. This means the filter will only receive true measurements and no clutter. Since there is no clutter and thus all randomness is eliminated, it will be sufficient to perform this experiment only once. Figure 5.17 shows the result of the experiment. That is, the resulting trajectories of all six individuals.

First, we map which trajectory comes from which individual. This mapping can be seen in Table 5.1.

bird name	PHD label
Charlie	1
Baby	36
Skittles	132 + 333
Penny	320
Ava	683
Greta	800

Table 5.1 Mapping of bird names to PHD filter labels.

As shown in Figure 5.17, the PHD filter evaluated seven trajectories instead of six. The trajectory with label 132 should not have been separate but part of the trajectory with label 333. The rapid change in bird movement and subsequent stopping caused this splitting of trajectories. This caused a new component to be created at the point where the bird stopped, while the previous component continued to move as if the bird had not stopped.

We can also see that part of Baby's trajectory is missing. A rapid change in direction and speed caused this missing part. We would have lost track of him if Baby had continued to fly. A detailed breakdown of each step can be seen in 5.18.

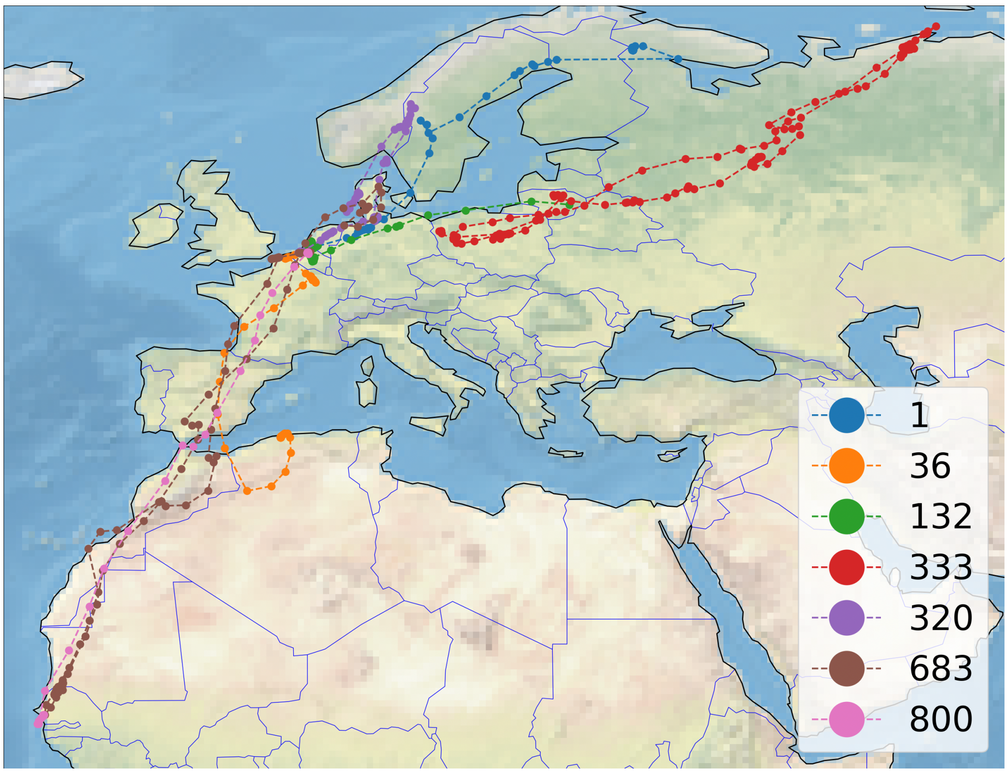


Figure 5.17 Ressulting trajectories of PHD filter.

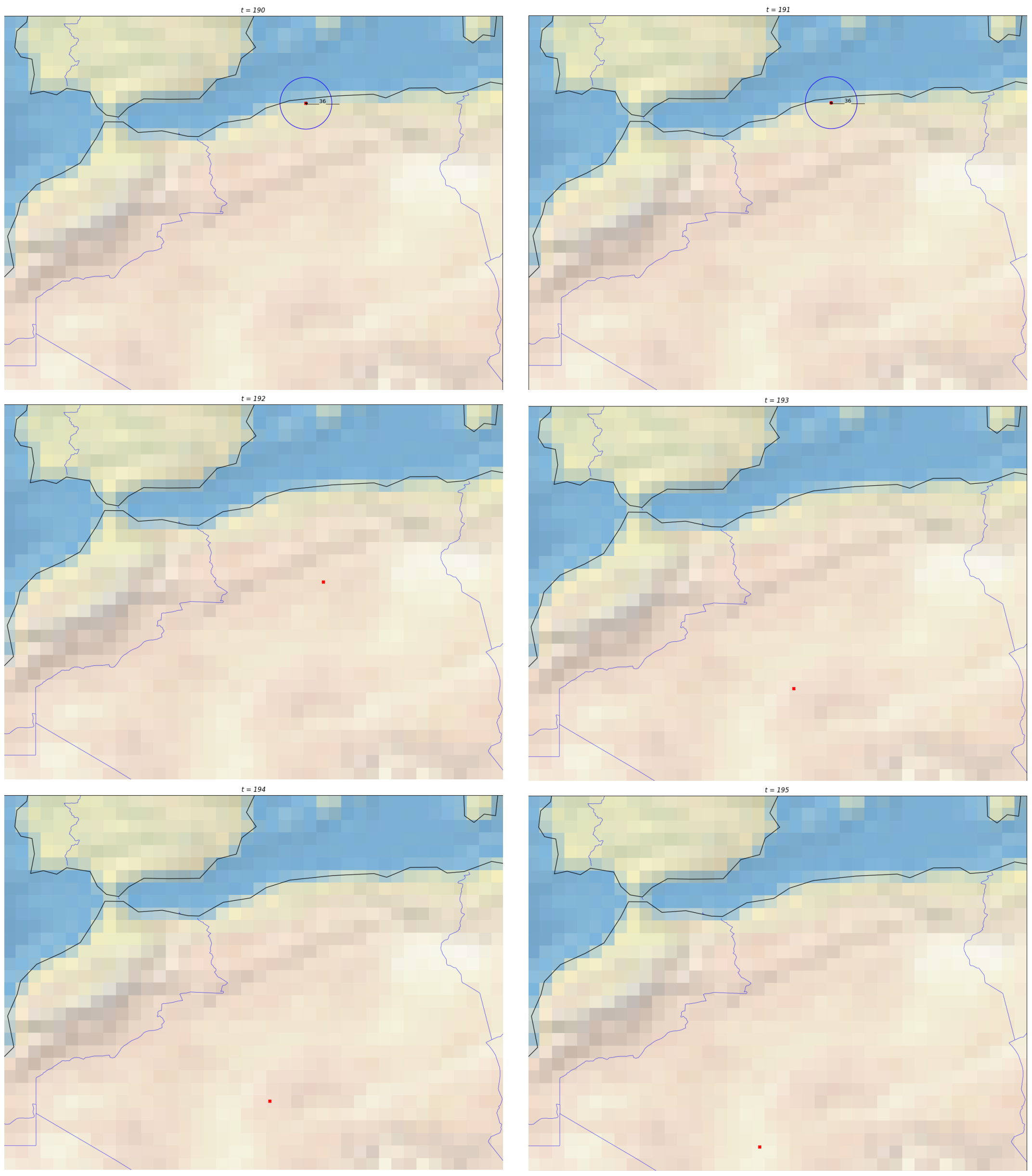


Figure 5.18 A detailed view of the trajectory evolution for the individual Baby when the trajectory was lost.

The final observation is that Charlie does not have a complete trajectory. This is due to missing measurements at that point in time. At time $t = 107$, the data transfer is interrupted. This interrupt takes a total of 14 time steps. The data transfer is resumed at time instant $t = 121$. That is too long for the target to stay alive. This is shown in more detail in Figure 5.19. The resulting graph of the evolution of $localization\_error$ for the six targets is shown in Figure 5.20.

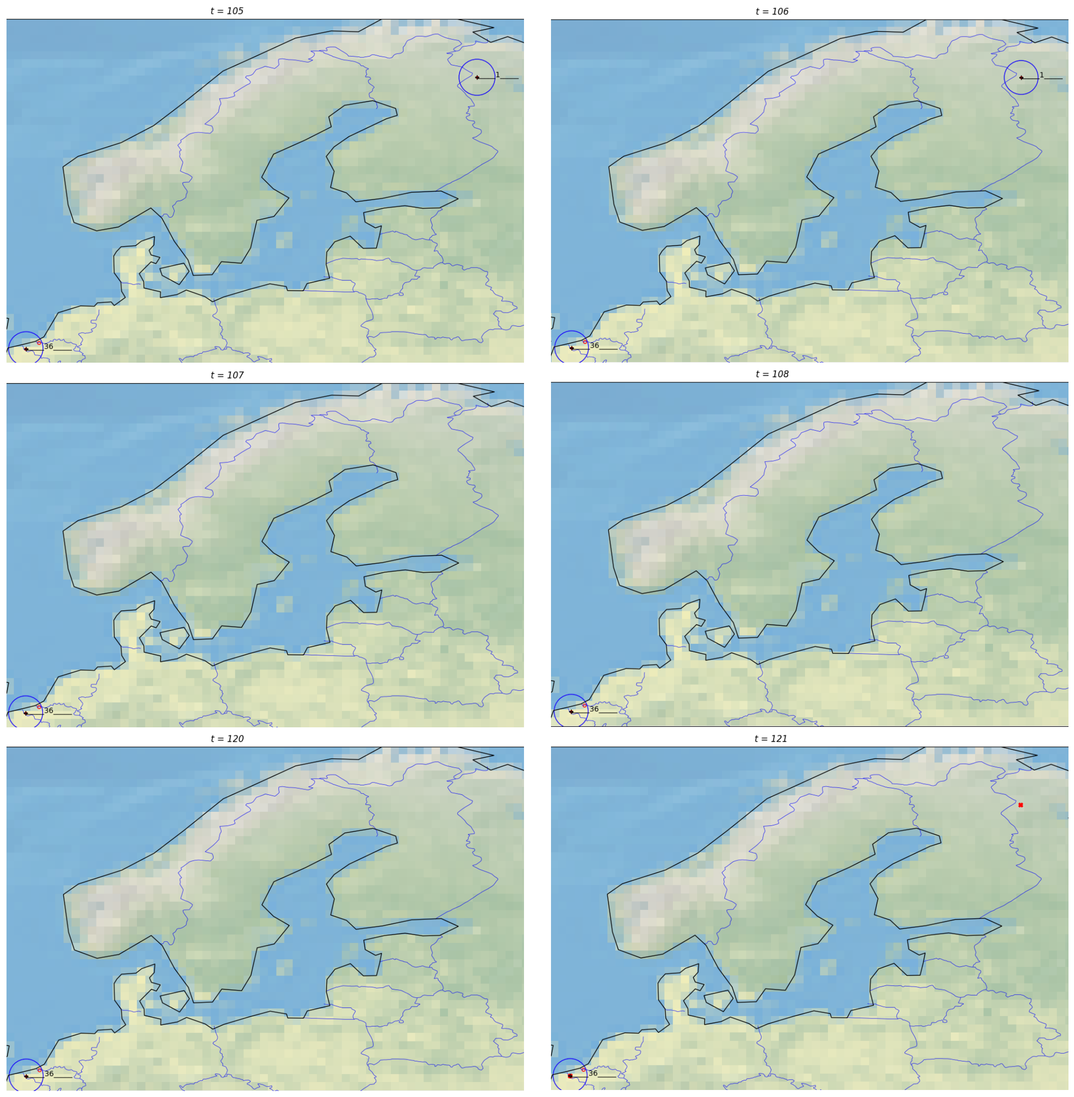


Figure 5.19 A detailed view of the trajectory evolution for the individual Charlie when the data was lost.

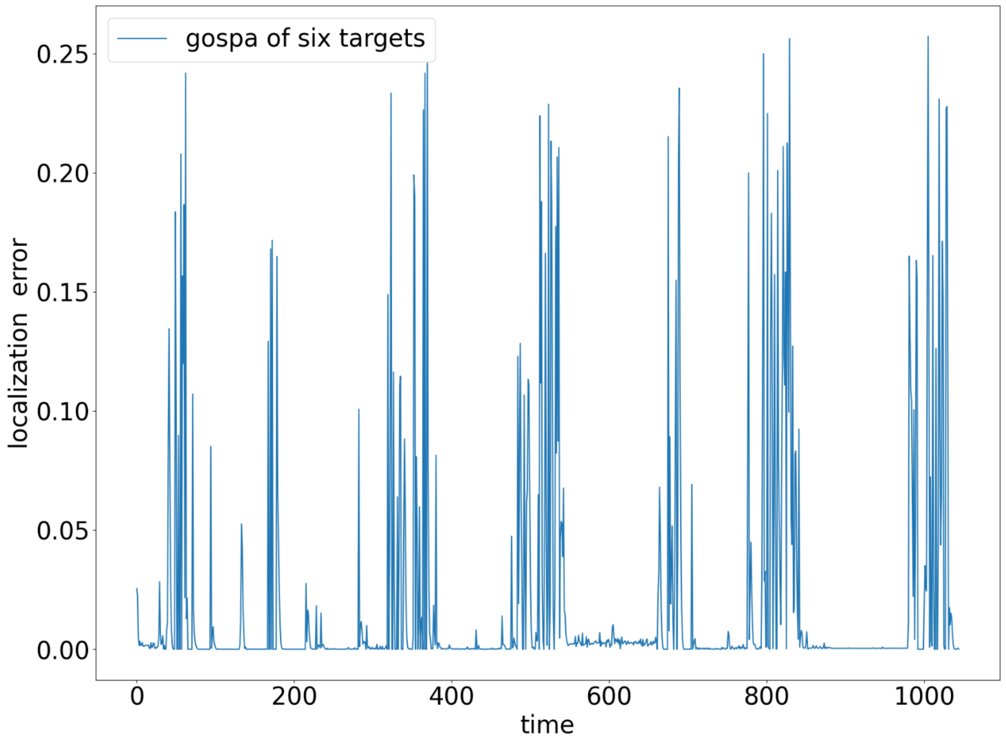


Figure 5.20 *localization_error* for six targets in clear enviroment.

5.2.2 MTT - cluttered environment

The following experiment will track the same six targets as the previous experiment, except that false measurements will be added to the true measurements. Again, we let the algorithm run ten times because of the randomness of the clutter.

The algorithm always found seven trajectories out of the ten runs, just like in the previous experiment. This means that for the individual Charlie, the algorithm always produces two trajectories for us. But in three runs, the PHD filter found nine trajectories. Let's look at them separately.

5.2.2.1 First run with nine trajectories

In this experiment, a trajectory split occurred. The trajectory of an individual named Ava was split into two trajectories. This is shown in Figure 5.21. One trajectory leads to the migration site (green trajectory), and the other is from the migration site to the breeding site (blue trajectory). Here, the clutter caused the creation of other components with different labels. While the component with the original label lost measurements, the new component with the new label continued.

At the same time, we can see that clutter, on the contrary, helps us track the individual named Baby (red trajectory). Because of the clutter here, the target was in a slightly different position, so the subsequent true measurement fell in the validation region.

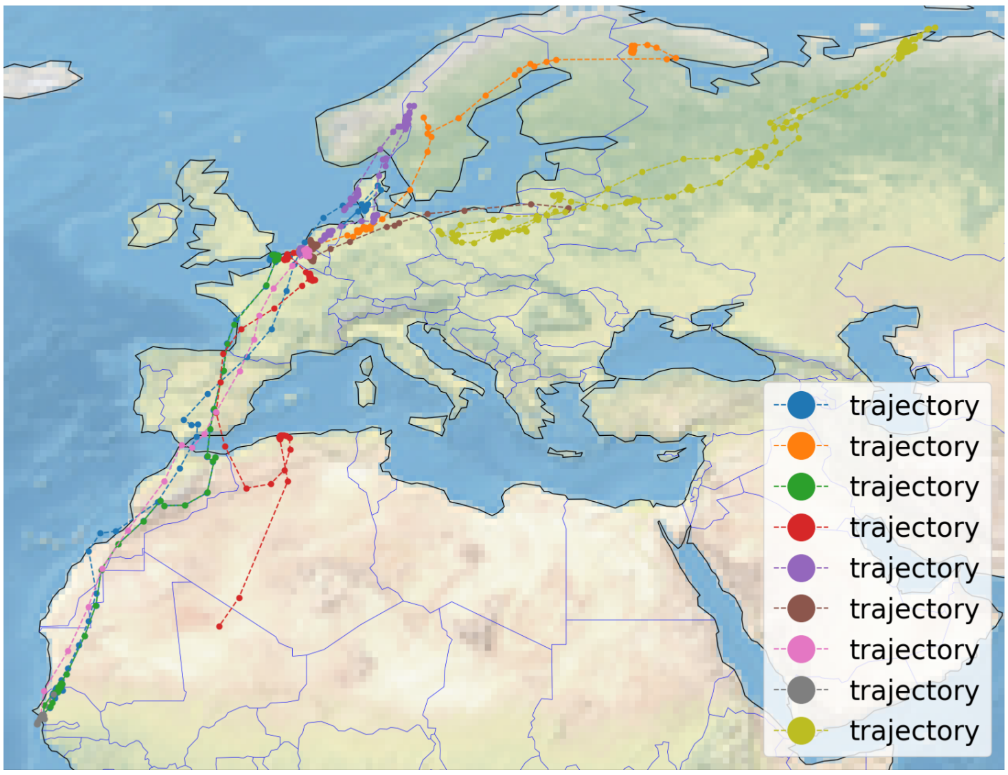


Figure 5.21 A representation of all 9 trajectories, containing the trajectory split.

5.2.2.2 Second run with nine trajectories

Here, compared to the previous experiment, there was no trajectory splitting. Figure 5.21 shows that the PHD filter created grey and brown false trajectories. These trajectories are long enough to satisfy the condition that they must be longer than *trajectory_prune*. However, here, the trajectories are more than 50 steps long but have not moved anywhere. Thus, new trajectories were created at the locations where the birds stayed during migration. But once the birds flew back towards its breeding site, these trajectories did not continue. It could be removed by increasing the *trajectory_prune* parameter or by some heuristic that removes trajectories shorter than a certain distance. These false trajectories can be seen marked in red in Figure 5.23.

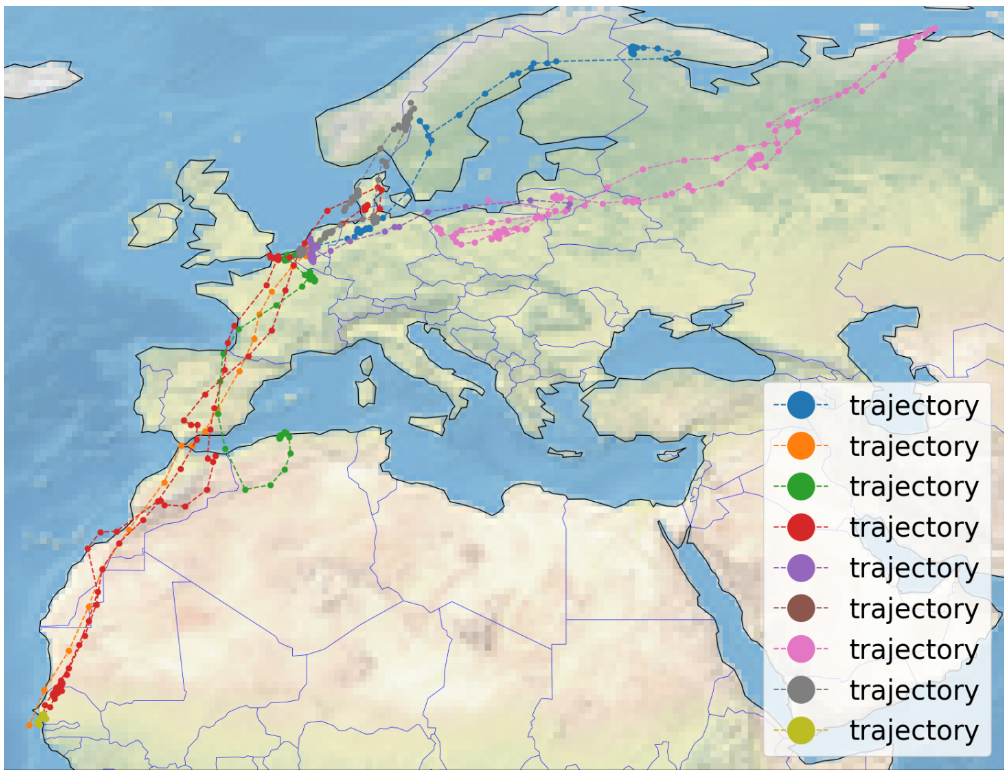


Figure 5.22 A representation of all 9 trajectories, containing the false trajectories.

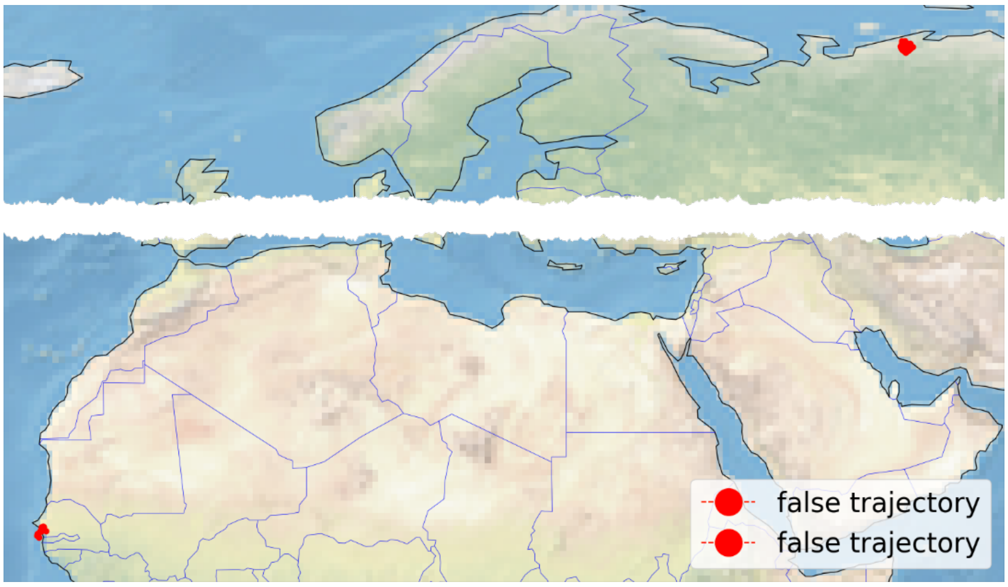


Figure 5.23 Two false trajectories from PHD filter in cluttered enviroment.

5.2.2.3 Third run with nine trajectories

Except for trim details, this experiment was precisely the same as in Section 5.2.2.2. Apart from the resulting trajectories being slightly different due to clutter, there was nothing new. Only the trajectory of the individual named Baby was one measurement longer. Anyway, it disappeared again after that.

In Figure 5.24 we compare *location_error* for six targets between the clear enviroment and the cluttered environment. The average *location_error* over all ten runs is taken for the cluttered environment.

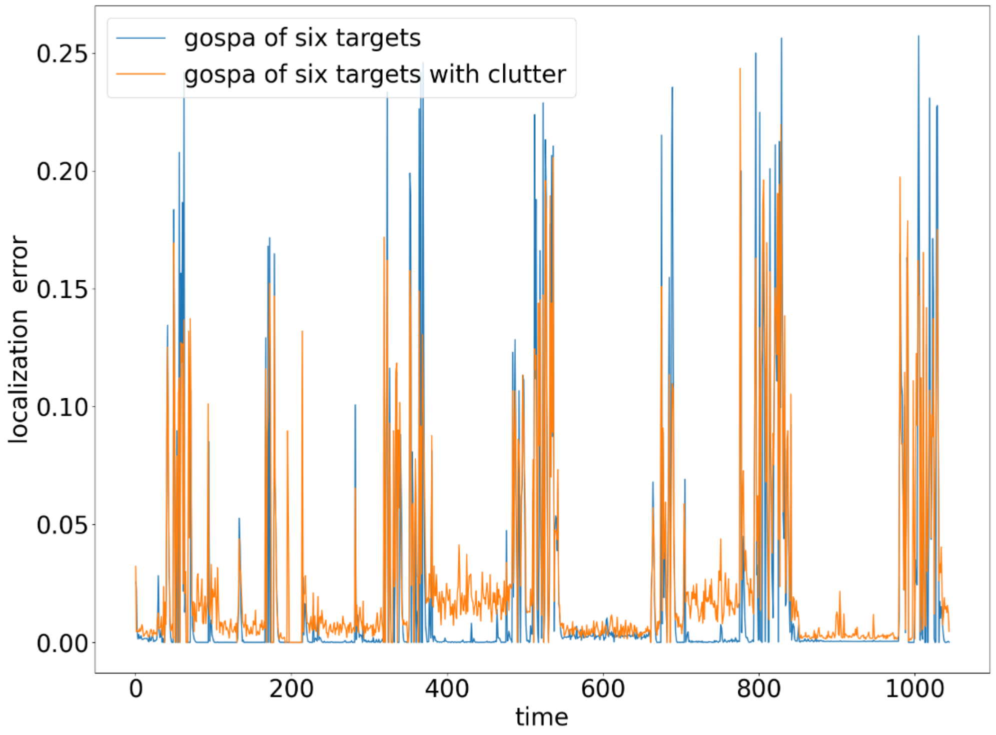


Figure 5.24 Comparison of *location_error* for clear environment and cluttered environment.

5.2.3 MTT- all the birds

In this last experiment, we run our algorithm over the entire dataset. This means there will be 26 targets in total. The most targets in one time step will be 16. As can be seen in Figure 5.25, the filter managed to load 30 trajectories. This means that some of the trajectories were definitely split or were created only at the breeding or migration site. Compared to Figure 3.7, which contains all the true trajectories, we see that some trajectories are completely missing. On the other hand, a large number of trajectories correspond to reality. In Figure 5.26 we can see the final *location_error* of PHD filter running over targets.

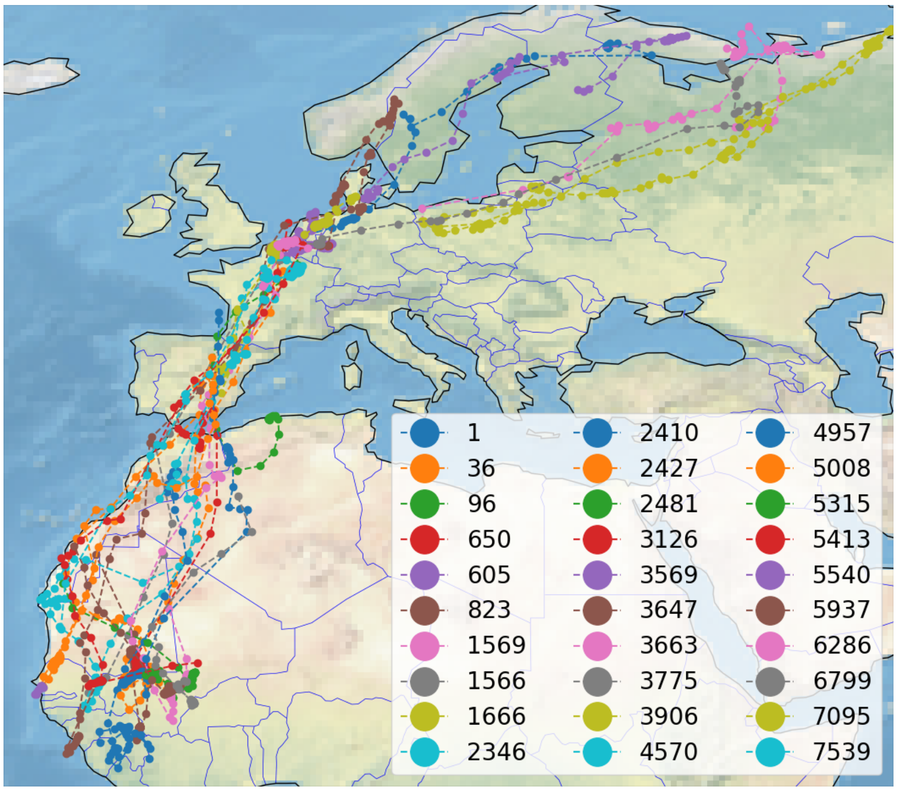


Figure 5.25 30 trajectories created by the PHD filter.

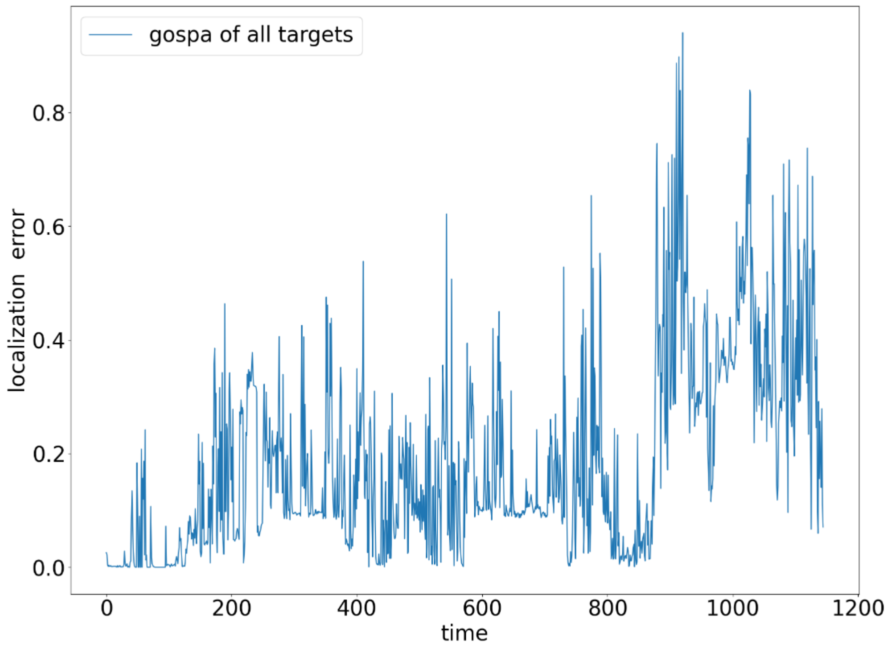


Figure 5.26 *localization_error* for all targets in clear enviroment.

Conclusion

This final chapter summarizes the work that has been done and suggests possible improvements.

Summary of thesis

The objectives of this work were as follows:

1. Shortly describe the studied biological problem.
2. Describe the algorithm used for target tracking. If possible, propose its modifications for the problem at hand.
3. Develop a working implementation, discuss its performance and qualities.

In Chapter 3, we explained the origin of the problem and its associated data. This accomplished the first goal. Chapter 1 covered all the theoretical background related to single target tracking. We have shown the best single-target tracking algorithm - the Kalman filter. In Chapter 2, we explained the basics and problems with multi-target tracking. We explained what RFS means and derived the PHD filter, which was used just for the experiments. We modified the PHD filter itself in Chapter 4. With these three chapters, we met the second goal of our work. With the last chapter, Chapter 5, we have also fulfilled the last objective of this thesis by performing experiments for our modified PHD filter under different conditions.

Future work

As seen in some experiments, modeling bird movement with the CVM model is not the best. Birds often change their velocity a lot because they stop for a while to rest or change their direction abruptly. In future work, the PHD filter could be modified not internally to respect the CVM model but to some other model.

Bibliography

1. DEDECIUS, Kamil. Diffusion estimation of state-space models: Bayesian formulation. In: *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. 2014, pp. 1–6. Available from DOI: `10.1109/MLSP.2014.6958920`.
2. WELCH, Greg; BISHOP, Gary. An Introduction to the Kalman Filter. *Proc. Siggraph Course*. 2006, vol. 8.
3. SIMON, Dan. Optimal state estimation: Kalman, H [infinity], and nonlinear approaches. *Choice (Middletown)*. 2007, vol. 44, no. 06, pp. 44–3334–44–3334.
4. GOLBON-HAGHIGHI, Mohammad-Hossein; ZHANG, Guifu. Detection of ground clutter for dual-polarization weather radar using a novel 3D discriminant function. *J. Atmos. Ocean. Technol.* 2019, vol. 36, no. 7, pp. 1285–1296.
5. DEDECIUS, Kamil; DJURIĆ, Petar M. Sequential Estimation and Diffusion of Information Over Networks: A Bayesian Approach With Exponential Family of Distributions. *IEEE Transactions on Signal Processing*. 2017, vol. 65, no. 7, pp. 1795–1809. Available from DOI: `10.1109/TSP.2016.2641380`.
6. MILLER, S L; CHILDERS, D G. *Probability and random processes : with applications to signal processing and communications*. London: Academic, 2012.
7. GURAJALA, Ramakrishna; CHOPPALA, Praveen B.; MEKA, James Stephen; TEAL, Paul D. Derivation of the Kalman filter in a Bayesian filtering perspective. In: *2021 2nd International Conference on Range Technology (ICORT)*. 2021, pp. 1–5. Available from DOI: `10.1109/ICORT52730.2021.9581918`.
8. BLACKMAN, S S. *Multiple-target Tracking with Radar Applications. Artech House Radar Library*. Ha, 1986.
9. YANG, Junjie; DELPHA, Claude. A Local Mahalanobis Distance Analysis Based Methodology for Incipient Fault Diagnosis. In: *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*. 2021, pp. 1–8. Available from DOI: `10.1109/ICPHM51084.2021.9486625`.
10. XIA, Yuxuant. *Lecture 4: Random finite sets [[online]]* [`https://chalmersuniversity.app.box.com/s/kbkmglktznkb2tjlr9pqefz3ezbiyw8p/file/1139153828080.`]. Chalmers University of Technology, 2021.

11. 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. Available from DOI: `10.1109/TSP.2006.881190`.

12. GARCÌA-FERNÀNDEZ, Àngel F.; XIA, Yuxuan; SVENSSON, Lennart. A comparison between PMBM Bayesian track initiation and labelled RFS adaptive birth. In: *2022 25th International Conference on Information Fusion (FUSION)*. 2022, pp. 1–8.

13. LAM, Quang; CRASSIDIS, John. Probability hypothesis density filter based design concept: A survey for space traffic modeling and control. In: *Infotech@Aerospace 2012*. Garden Grove, California: American Institute of Aeronautics and Astronautics, 2012.

14. 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. Available from DOI: `10.1109/TAES.2003.1261119`.

15. KROPFREITER, Thomas; MEYER, Florian; HLAWATSCH, Franz. An Efficient Labeled/Unlabeled Random Finite Set Algorithm for Multiobject Tracking. *IEEE Transactions on Aerospace and Electronic Systems*. 2022, vol. 58, no. 6, pp. 5256–5275. Available from DOI: `10.1109/TAES.2022.3168252`.

16. MAHLER, Ronald P. S. *Statistical Multisource-Multitarget Information Fusion*. USA: Artech House, Inc., 2007. ISBN 1596930926.

17. GARCÍA-FERNÁNDEZ, Ángel F.; SVENSSON, Lennart. Trajectory PHD and CPHD Filters. *IEEE Transactions on Signal Processing*. 2019, vol. 67, no. 22, pp. 5702–5714. Available from DOI: `10.1109/TSP.2019.2943234`.

18. SPANOGHE, Geert; JANSSENS, Kjell; KLAASSEN, Raymond; SCHAUB, Tonio; MILOTIC, Tanja; DESMET, Peter. *BOP_RODENT - Rodent specialized birds of prey (Circus, Asio, Buteo) in Flanders (Belgium)*. Research Institute for Nature and Forest (INBO), 2023.

19. *GPS tracking network for large birds* [`https://lifewatch.be/en/gps-tracking-network-large-birds`]. [N.d.]. Accessed: 2024-3-31.

20. *ornitela*. Ornithology and telemetry applications [`https://www.ornitela.com`]. [N.d.]. Accessed: 2024-4-19.

21. MISJEL DECLEER, VLIZ photo gallery. *Tagged herring gull* [`https://www.lifewatch.be/en/mediagallery?album=4092&pic=73912`]. [N.d.]. Accessed: 2024-4-23.

22. FRADE, José. *Montagu's Harrier Circus pygargus* [`https://macaulaylibrary.org/asset/44390971`]. [N.d.]. Accessed: 2024-4-18.

23. WANG, Jeffrey. *Western Marsh Harrier Circus aeruginosus* [`https://macaulaylibrary.org/asset/190082101`]. [N.d.]. Accessed: 2024-4-18.

24. STEVENS, Guy. *Short-eared Owl (Northern) Asio flammeus flammeus* [`https://macaulaylibrary.org/asset/407977671`]. [N.d.]. Accessed: 2024-4-18.

25. BRELSFORD, Craig. *Hen Harrier Circus cyaneus* [`https://macaulaylibrary.org/asset/57601871`]. [N.d.]. Accessed: 2024-4-18.

26. VOADEN, Nigel. *Common Buzzard (Western) Buteo buteo buteo* [`https://macaulaylibrary.org/asset/87039801`]. [N.d.]. Accessed: 2024-4-18.

27. RAHMATHULLAH, Abu Sajana; GARCÍA-FERNÁNDEZ, Ángel F.; SVENSSON, Lennart. Generalized optimal sub-pattern assignment metric. In: *2017 20th International Conference on Information Fusion (Fusion)*. 2017, pp. 1–8. Available from DOI: `10.23919/ICIF.2017.8009645`.

Enclosed medium contents

```
readme.txt ............................ The file with media contents description
src
  code ......................... The directory of source codes of implementation
  data ............................................. The directory of datasets
text
  master-thesis-holastom.pdf ........................... Text in PDF format
  master-thesis-holastom ..... The directory of LaTeX source codes of the thesis
```