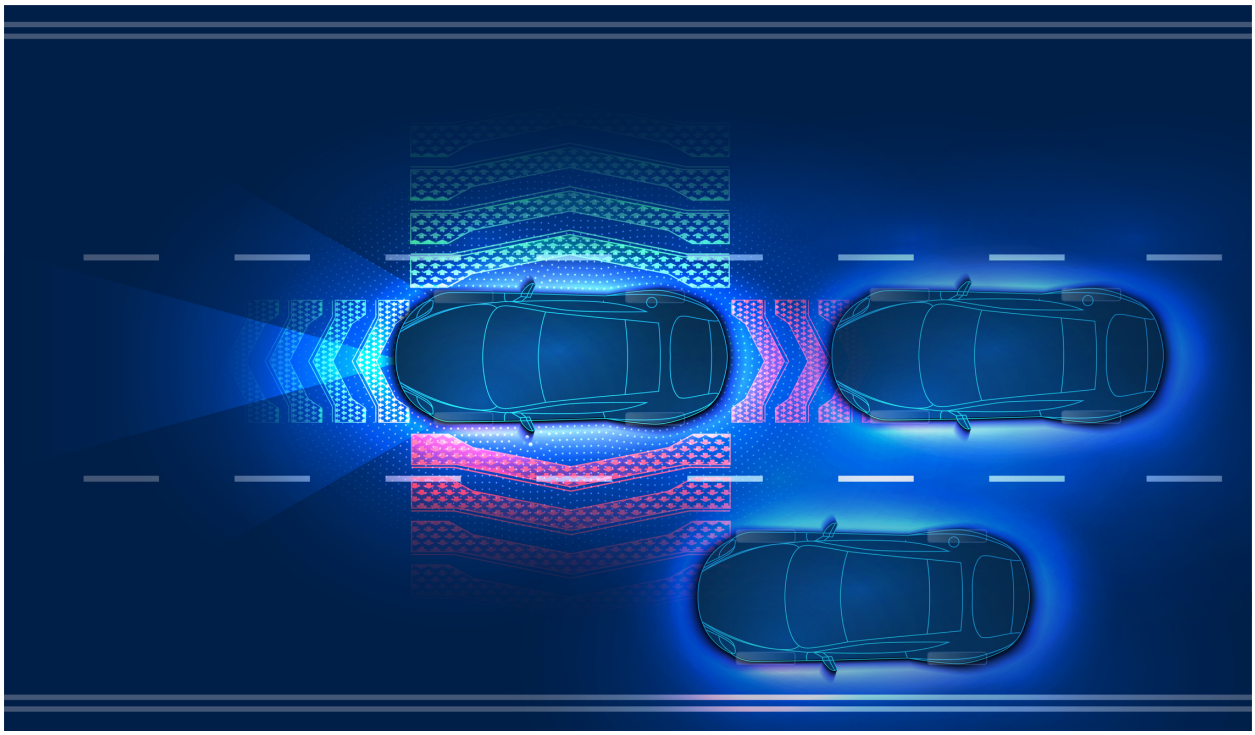
CHALMERS
UNIVERSITY OF TECHNOLOGY



# A step towards ground-truth estimation using offline radar tracking

Master's thesis in Automotive Masters Programme

AYAM JAIN

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

# A step towards ground-truth estimation using offline radar tracking

AYAM JAIN


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2022

A step towards ground-truth estimation using offline radar tracking
AYAM JAIN



Supervisor: Ludvig Hazard, Aptiv
Examiner: Prof. Henk Wymeersch, Department of Electical Engineering

Master's Thesis 2022
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

A step towards ground-truth estimation using offline radar tracking
AYAM JAIN
Department of Electical Engineering
Chalmers University of Technology

# Abstract

In this thesis, a non-causal radar tracking algorithm has been developed for better estimating the states of a vehicle. The focus has been on detecting and resolving the under-segmentation issue. The problem was studied in a simulated environment using synthetic radar detections and a GM-PHD tracker was used for tracking the objects in the scenario. Different clustering techniques have been used for data association and also as a method to detect under-segmentation. Further, a cost function with multiple components has been developed to evaluate the track quality and perform jump moves to minimize the cost associated with the track. Finally, RTS smoothing is applied to the data to better estimate the vehicle states. The developed tracking algorithm demonstrates that it is capable of handling certain scenarios, while others are more difficult. The findings indicate that further work is required to arrive at a more robust/complete solution.

Keywords: radar tracking, extended object tracking, GM-PHD tracker, RTS Smoother, non-causal, offline, energy optimization, vehicle safety

# Acknowledgements

Firstly, I want to thank APTIV for giving me the opportunity to do my thesis at such a great company and with the amazing F360 group at APTIV.

I would like to express my deepest appreciation to my supervisor - Ludvig Hazard at APTIV for his constant guidance and mentoring through out the duration of the thesis. This thesis wouldn't have been carried out the way that it did without his support.

Further, I want to thank my advisor and examiner - Yu Ge and Prof. Henk Wymeersch (respectively) at Chalmers for their persistent advice and assistance during the thesis.

Finally, I would also like to thank all my friends and family for their support and encouragement throughout the journey.

Ayam Jain, Gothenburg, June 2022

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

| | |
|---|---|
| ADAS | Advanced driver assistance systems |
| CA | Constant acceleration |
| CT | Constant turn |
| CV | Constant velocity |
| EKF | Extended Kalman filter |
| FOV | Field of view |
| GT | Ground truth |
| GM | Gaussian mixture |
| GM-PHD | Gaussian mixture - probability hypothesis density |
| LIDAR | Light detection and ranging |
| RADAR | Radio detection and ranging |
| RFS | Random finite set |
| RTSS | Rauch-Tung-Striebel smoother |
| UKF | Unscented Kalman filter |

# Contents

**List of Acronyms** **ix**

**List of Figures** **xiii**

**List of Tables** **xv**

**1 Introduction** **1**
1.1 Background . . . 1
1.2 Purpose . . . 2
1.3 Objective . . . 3
1.4 Scope/Limitations . . . 3
1.5 Thesis outline . . . 3

**2 Theory** **5**
2.1 RADAR . . . 5
2.1.1 Range estimation . . . 5
2.1.2 Range-Rate (Velocity) estimation . . . 5
2.1.3 Angle estimation . . . 6
2.1.4 Radar measurements . . . 6
2.2 GM-PHD tracker . . . 7
2.2.1 Parameters of GM-PHD tracker . . . 9
2.2.2 Motion model . . . 9
2.2.2.1 Constant velocity (CV) model . . . 9
2.2.2.2 Constant acceleration (CA) model . . . 10
2.2.2.3 Constant turn (CT) model . . . 10
2.3 Clustering . . . 10
2.3.1 K-means clustering . . . 11
2.3.2 DBSCAN . . . 11
2.4 Smoothing . . . 12
2.4.1 Rauch-Tung-Striebel (RTS) smoother . . . 13

**3 Methods** **15**
3.1 Simulation Environment . . . 15
3.2 Scenario construction/overview . . . 15
3.3 Online tracker . . . 16
3.4 Data generation and pre-processing . . . 18
3.5 Non-Causal Algorithm . . . 19

3.5.1 Clustering the input detections . . . 19
3.5.2 Cost function . . . 21
3.5.3 Detection proximity cost . . . 22
3.5.4 Track initiation cost . . . 23
3.5.5 Physical exclusivity cost . . . 24
3.5.6 Clustering the track into 2 . . . 24
3.5.7 Simulated tracking . . . 24
3.5.8 Overlapping area cost . . . 25
3.5.9 Split and merge - jump actions . . . 27
3.5.10 Compare the cost . . . 28
3.5.11 Smoothing . . . 28

**4 Results** **29**
4.1 Tuning track initiation cost parameters . . . 29
4.2 Tuning GM-PHD tracker for simulated tracking . . . 30
4.3 Scenario 1 . . . 30
4.4 Scenario 2 - a limitation . . . 35

**5 Discussion** **37**
5.1 Scenario . . . 38
5.2 Tracker . . . 38
5.3 Clustering . . . 39

**6 Conclusion** **41**

**7 Future Work** **43**

# List of Figures


2.1 A pulse wave hitting an object distance 'd' apart and bouncing back towards the radar antenna . . . . . 5
2.2 Angle estimation utilizing numerous receiver antennas and a transmitor antenna designated Rx and Tx respectively. . . . . 6
2.3 Visualization of azimuth vs range cross section of a radar. . . . . 7
2.4 Block diagram implementation of GM-PHD tracker . . . . . 8

3.1 The Driving/Road Scenario. Blue vehicle is the ego vehicle and the other vehicles are actors. . . . . 16
3.2 The Under-segmentation Scenario . . . . . 18
3.3 Block Diagram implementation of Non-Causal Algorithm. . . . . 20
3.4 A scenario where only K-means with n = 2 is applied directly on the detection data. . . . . 21
3.5 A scenario in which outliers are removed using the DBSCAN method, and subsequently clean detection data is processed using K-means with n = 2. . . . . 21
3.6 Example for the calculation of Detection Proximity cost. The red dots are detection points and the dotted rectangle is the bounding box representation of the object around the state estimate (aka centroid). 23
3.7 Clustering detections into 2 groups. On the left, it is group of detections from a single time instance. On the right, 2 clustered groups after running k-means clustering algorithm. . . . . 25
3.8 Perfect area-overlap. The detections are divided into two groups, but they are all associated with the same single track, thus the state estimates from both cluster groups completely overlap. . . . . 25
3.9 No area-overlap. The detections are divided into two groups, but because the detections are connected with separate tracks, the state estimates from both cluster groups do not overlap. . . . . 26
3.10 Moving from a region of no overlap to a region of significant overlap. 26

4.1 Output of exponential decay function for varying decay constant, C. . 30
4.2 Track Initiation cost function with C = 0.89. and radar FOV, rng = 10 m. . . . . 31
4.3 The Detections associated to different tracks for the original input scenario to the non-causal algorithm. The detection points that are linked to the same track are colored the same. . . . . 32

4.4 The Detections associated to different tracks after running the non-causal algorithm. . . . . 32
4.5 The smoothed state estimate of the tracks described in scenario 1. . . 33
4.6 The absolute distance error plot of the smoothed and filtered states from the ground truth for track 1 in red. . . . . 34
4.7 The absolute distance error plot of the smoothed and filtered states from the ground truth for track 2 in blue. . . . . 34
4.8 The absolute distance error plot of the smoothed and filtered states from the ground truth for track 3 in green. . . . . 35
4.9 The Detections associated to different tracks for the original input scenario to the non-causal algorithm. The radar sensor is located at origin (0,0) . . . . . 36
4.10 The Detections associated to different tracks after running the non-causal algorithm. . . . . 36

5.1 Situations where bounding box estimation is vulnerable. The detections are in red. The box in blue is the real vehicle and the bounding box is represented by purple dotted box. . . . . 38
5.2 Scenario where bounding box estimation is vulnerable. The detections are in red. The box in blue is the real vehicle and the bounding box is represented by purple dotted box. . . . . 39

# List of Tables

2.1 Notations for GM-PHD filter . . . . . 8

3.1 Structure of TrackData . . . . . 18
3.2 Notations . . . . . 22

4.1 Cost error values from running the non-causal algorithm for scenario 1. 31
4.2 Cost error values from running the non-causal algorithm for scenario 2. 35

# 1

# Introduction

## 1.1 Background

The automotive industry as a whole is undergoing a continuous transformation of technologies in many aspects, such as electric drives, autonomous driving, and infotainment systems. The desire for a vehicle to travel from point A to point B autonomously (without human assistance) is no secret, and everyone wishes for this to become a reality as soon as possible while preserving the highest level of safety. The safety of drivers, passengers, and vulnerable road users is critical, so the autonomous vehicle must perceive its surroundings and make appropriate decisions based on the available information. This is difficult because driving is a complex task [1].

To improve safety, a vehicle needs to develop a complete and trustworthy perception of the vehicle's immediate surroundings. Advanced driver assistance systems (ADAS), which are electronic systems designed to assist drivers in various traffic circumstances, frequently employ this expertise. An autonomous vehicle depends laboriously on several sensors including cameras, lidars, radars, and others to perceive the surrounding environment to produce a legitimate perception [2]. Cameras are typically good at categorizing objects, but they cannot precisely estimate object speed. Although a lidar, which uses a laser, can and is utilized for this purpose [3]. The laser's disadvantage is that it is extremely sensitive to weather conditions such as rain, snow, and dirt. The radar, unlike the camera and lidar, is weather-resistant and is referred to as an all-season sensor [4], but it does not provide good angular resolution or as good object classification as the lidar [5]. This is why, in an automotive application, a sensor combination is required.

A driving environment, for instance, has several vehicles (referred to as objects/targets) moving around dynamically and it is crucial to accurately track these objects. Several trackers have been developed over years to gather data from sensors and filter the data from the same object over time into a 'track' [6]. The tracker also determines the estimated properties of tracked objects such as velocities, accelerations, and future position. One of the most significant aspects of tracking numerous targets is associating data with the appropriate tracks which is known as data association, and is the most challenging aspect of object tracking. Over time, the trackers have become better at handling more arduous situations such as abrupt object motion, changing appearance patterns of both the object and the scene, non-

rigid object structures, object-to-object, and object-to-scene occlusions [7], and can also resolve the extent of the object [8]. There are several state-of-the-art trackers such as near-online multi-target tracking (NOMT), and GM-PHD tracker among others available for use and perform well [9].

Further, Developers need to determine how closely the environmental models generated by those sensors resemble the real world. Ground-truth is the term for this gold standard [10]. Ground-truth is crucial information that is used to test and benchmark the tracker performance. However, ground-truth might be considered a 'relative' phrase since it is information that must be regarded as the best possible knowledge about the objects in the environment at a specific time. Vehicles are equipped with a variety of inertial measurement devices, highly precise GPS technology, lidar, and cameras on their roofs to record ground truth. To deliver such precision, these sensors are typically very expensive, large, and inconvenient to use in common automobiles. As a result, it is clear that ground-truth information is critical yet difficult to get; hence, it is worthwhile to investigate alternative methods to obtain ground-truth estimates and, as a result, improve tracker performance.

## 1.2 Purpose

Most modern vehicles come with some form of ADAS. Vehicles are equipped with sensors and tracking software (referred to as online tracker) that usually runs on a chip with limited resources. The online tracker must output predictions within milliseconds i.e. an online tracker is designed to yield the best predictions given the computational resources and time constraints. It is therefore critical to validate the accuracy and precision of online trackers to assure correct functioning and the development of future autonomous drive systems. Verification of the system necessitates a significant investment in resources, and the most helpful data for verification is ground truth, which is not always readily available and can be costly. As a result, it is intriguing to implement some type of non-causal (offline) tracking using the knowledge available from the whole scenario. An offline tracker is intended to produce the best estimates without any constraints on computational performance and time. Working in an offline environment allows access to considerably more processing power, which can be utilized to evaluate much more data over a longer time frame and improve the precision of tracking predictions to a point such that they could be considered ground truth.

There are also several scenarios where the online tracker fails to detect multiple objects and tracks them incorrectly. For instance, when two objects are close to each other, the tracker could track them as one single object as the detections are close to each other. But with more information available in the offline environment, the offline tracker should be able to better perceive the scenario and therefore be better able to predict the states.

## 1.3 Objective

The objective of this thesis is to develop an algorithm that can run in an offline environment and output the best possible track data accounting for the past, present, and, future information without any constraints on runtime. This thesis focuses on exploring energy minimization methods and smoothing techniques for tracking in an offline environment.

The following questions were sought to be answered: Can the cost computed by the energy minimization method be used to associate the proximity of a track to ground truth? Does applying a smoothing algorithm improve the state estimates and therefore bring us closer to ground truth? Can the developed algorithm detect and track the trajectories for certain scenarios where the online tracker fails to do a great job in a better way?

## 1.4 Scope/Limitations

Object tracking is a vastly researched topic and has progressed significantly over the years [7], [8]. Therefore, it beyond the scope of a thesis to develop an algorithm that does offline tracking. As a result, the focus of this thesis is on resolving the 'under-segmentation' phenomena, which occurs when two objects are near together and the tracker tracks them as a single object.

Some limitations were further set to fit the scope of the thesis. To generate data for constructing the offline tracking system, a single radar sensor configuration positioned on top of the vehicle with a 360-degree field of view is used in this thesis. The tracking algorithms can only work in a 2D Cartesian coordinate system as the radars used can only detect objects in a plane. Further, The main focus has been on handling only 4-wheeled objects (cars, trucks, etc).

This thesis does not focus on developing an online tracker and hence a readily available tracker with some adaptation was used for this thesis. The algorithm has not been tested with data from real-world cases and is only capable of running in simulation environments.

## 1.5 Thesis outline

Chapter 1 provides background and introduction to the objective of this thesis. The underlying theory of the methods used in this work is discussed in chapter 2. Chapter 3 gives a brief overview of the simulation environment and the scenario overview. The proposed algorithm and its implementation are discussed in chapter 4. Chapter 5 presents the results obtained by implementing the proposed algorithm. Chapters 6, 7, and 8 talk about the various discussion points associated with the work, the conclusion, and the possible future work respectively.

# 2
# Theory

This section seeks to offer the nomenclature and information needed to understand the methodologies presented in later chapters.

## 2.1 RADAR

Radio Detection and Ranging (RADAR) is a device that uses transmitted and reflected electromagnetic radio waves for detecting objects in the surroundings [11]. Automotive radar is an important sensor that helps in detecting the pedestrians and vehicles in the surroundings and aids the ADAS systems.

This thesis does not delve deep into radar functioning but it is important to have an overview of the different properties of radar measurements. The following subsections aim to provide some insights into radar properties.

### 2.1.1 Range estimation

A radar constitutes transmitter and receiver antennas. An electromagnetic wave is transmitted by the transmitter antenna and bounces off the objects in the surrounding [12]. These reflected waves are received by the receiver antenna. The electromagnetic waves travel at the speed of light, and therefore, the time between transmission and receiving of the reflected wave, which is called the time of flight, can be used to determine the range of the objects around.

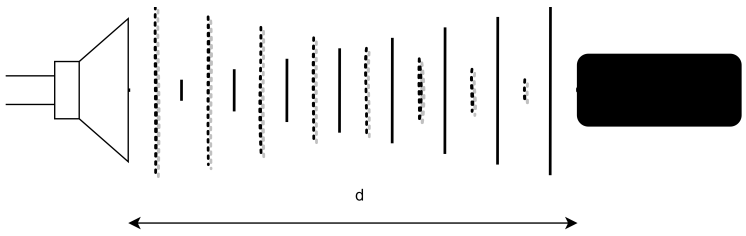



**Figure 2.1:** A pulse wave hitting an object distance 'd' apart and bouncing back towards the radar antenna

### 2.1.2 Range-Rate (Velocity) estimation

The radar transmits electromagnetic waves at a certain known frequency. Once, the transmitted wave bounces off a moving object, The returning wave will experience

a slight shift in the frequency due to the Doppler effect [13]. This difference in frequency is used to determine the relative velocity between the radar and the target object.

### 2.1.3 Angle estimation

A radar-equipped with a single receiving antenna is unable to differentiate between multiple targets that are positioned at the same distance from the sensor. However, if multiple receiver antennas are spaced appropriately, it will be possible to differentiate between multiple targets. Due to the spacing between the receiving antennas, the reflected signal will have to travel slightly longer distances before they are received by the antennas. This will lead to a phase shift between the signal received at the antennas. The phase shift is proportional to the extra distance traveled by the reflected signal and the signal's wavelength. The change in phase shift is analyzed [12] to determine where the reflected signal originates from and thereby determine the azimuth angle information.

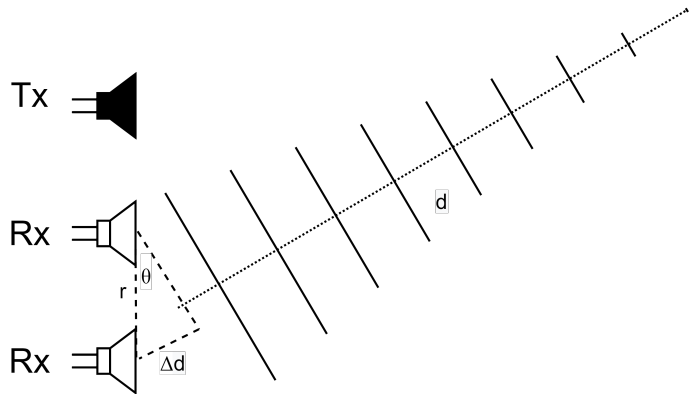



**Figure 2.2:** Angle estimation utilizing numerous receiver antennas and a transmitor antenna designated Rx and Tx respectively.

It can be interpreted from figure 2.2 that due to the distance r between the receiver antennas, the flight duration of the reflected wave is increased by $\Delta d$. As a result, the azimuth angle information can be determined by determining where the reflected signal originates from.

### 2.1.4 Radar measurements

Several different radar components have been discussed in the preceding sections and it is important to consider/understand a few properties associated with these components. The position information of the target is determined by its range and azimuth. The field of view (FOV) of radar is conical in shape with a curved grid as shown in figure 2.3.

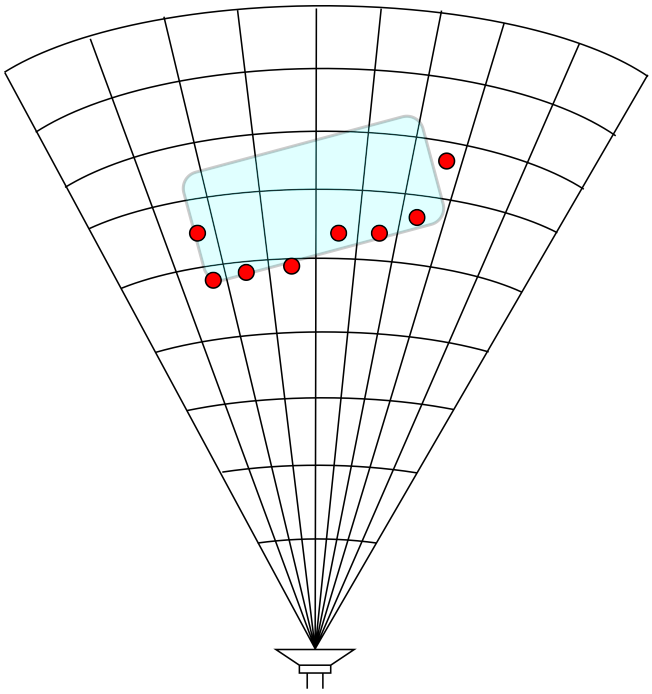


**Figure 2.3:** Visualization of azimuth vs range cross section of a radar.

The length and width of the conical curve grid are determined by the range and azimuth resolution of the radar. Each cell in the grid represents a potential detection if an object/target exists in the cell. It can be seen in figure 2.3 that the grid cell size increases with distance from the sensor. As a result, a distant target will create fewer detection points compared to a close target.

The range-rate information is the measurement of the target's relative velocity in the direction of the motion of the sensor i.e. radial velocity measurement. Therefore, the target velocity is only truly measured when the target is moving linearly towards or away from the sensor. The range-rate estimation becomes complicated and needs additional methods to resolve the true object velocity and direction when the object/target is rotating/moving in a turn or when the object is moving perpendicular to the sensor. In these cases, the velocity vectors have to be resolved to estimate the motion of the object.

## 2.2 GM-PHD tracker

A Gaussian mixture-probability hypothesis density (GM-PHD) tracker implements the probability hypothesis density (PHD) filter using a mixture of Gaussian components for tracking extended targets.

The PHD filter is a computationally cheaper multi-object target filter that iteratively estimates the number and state of a set of targets from a series of observations. PHD filter is based on random finite set (RFS) approach [14] and works by propagating the strength of the target RFS in time rather than the whole multi-target posterior density, allowing it to operate in situations with false alarms and miss detections [15].

Figure 2.4 shows the logical implementation of the GM-PHD tracker.

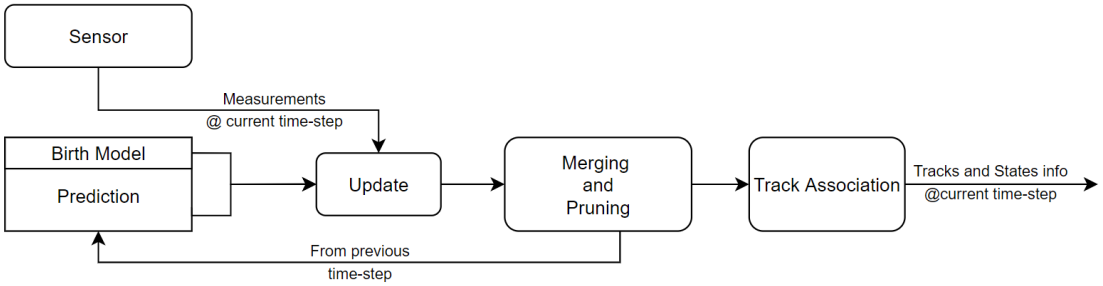



**Figure 2.4:** Block diagram implementation of GM-PHD tracker

| Symbol | Description |
|---|---|
| $J_{t\|t}$ | Weighted Gaussian component |
| $m_{t\|t}$ | Mean of the Gaussian component |
| $P_{t\|t}$ | Covariance of the Gaussian component |
| $w_{t\|t}$ | weight of the Gaussian component |
| $\gamma_t(x)$ | A Gaussian mixture for appearance of new targets |

**Table 2.1:** Notations for GM-PHD filter

From [16], The GM-PHD filter utilizes the following PHD representation

$$v_{t|t}(x) = \sum_{i=1}^{J_{t|t}} w_{t|t}^{i} N(x; m_{t|t}^{i}, P_{t|t}^{i}) \tag{2.1}$$

Every iteration begins with the birth model producing some GM. This corresponds to the birth rate which is a tuneable parameter. This birth density is then combined with the current density (which is calculated from the death rate and elapsed time from the previous prediction) to compute the current predictions.

Then, in the update block, the current measurements (current readings from the sensor) along with the current computed predictions are used to update the GM. The updated GMs are merged if the peaks appear close to each other and/or pruned if the weights of the Gaussian terms are below a certain pre-defined threshold ( which is a tunable parameter).

The resulting GMs are utilized by the tracker algorithm to associate with current tracks and, as a result, continue or create a new track if necessary. The resulting

output of the GM-PHD tracker is the track information including the track ID, the track age, the state vector, the state covariance matrix, and more.

### 2.2.1 Parameters of GM-PHD tracker

Some of the tunable parameters of the GM-PHD tracker are discussed below -

**Birth rate** - It is a scalar quantity that defines the number of components that are added to the density per unit of time. The tracking algorithm uses a function to define where the components are born. This function can be modified to adapt to the requirements of the tracker use case. The Gaussian-mixture birth process (from [16]) can be represented as follows -

$$\gamma_t(x) = \sum_{i=1}^{J_{\gamma,t}} w_{\gamma,t}^i N(x; m_{\gamma,t}^i, P_{\gamma,t}^i) \tag{2.2}$$

**Death rate** - It is a scalar quantity that is used to model the death of components in density per unit of time. The death rate, therefore, corresponds to the probability of survival of the component in the density upon prediction. This survival probability can be represented as follows - $(1 - DeathRate)^{dT}$ where dT is the timestep (prediction interval)

**Assignment threshold** - This is a scalar value that controls the number of detection cells to be considered for birth in adaptive birth density.

**Extraction threshold** - This is a minimum scalar value above which a component in the density is labeled as a tentative track. This value is usually lower than the confirmation threshold.

**Confirmation threshold** - This is a minimum scalar value above which the tentative tracks (above extraction threshold) are marked as confirmed tracks. The confirmation threshold is higher than the extraction threshold.

**Merging threshold** - If the Kullback-Leibler difference [17] between 2 components is smaller than this scalar value, the 2 components are merged into one component.

### 2.2.2 Motion model

A motion model helps in determining the states at the next time step based on the current state of the object. Various distinct types of motion models can be used to describe a vehicle's motion, as follows:

#### 2.2.2.1 Constant velocity (CV) model

In this model, the object is considered to move in a straight line at a constant velocity. The CV model is described as follows-

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \tilde{q}(t) \tag{2.3}$$

where $x(t)$ is the state vector defined as $x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$, where p(t) and v(t) are position and velocity respectively and $\tilde{q}(t)$ is noise.

#### 2.2.2.2 Constant acceleration (CA) model

In this model, the object is considered to be moving with a constant acceleration. The CA model is described as follows-

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x(t) + \tilde{q}(t) \tag{2.4}$$

where $x(t)$ is the state vector defined as $x(t) = \begin{bmatrix} p(t) \\ v(t) \\ a(t) \end{bmatrix}$, where p(t), v(t) and a(t) are position, velocity and acceleration respectively and $\tilde{q}(t)$ is noise.

#### 2.2.2.3 Constant turn (CT) model

In this model, the object is considered to be moving at constant velocity along a curvilinear path ( circular segments).
The CT model is described as follows-

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & cos\phi & 0 & 0 \\ 0 & 0 & sin\phi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \tilde{q}(t) \tag{2.5}$$

where $x(t)$ is the state vector defined as $x(t) = \begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \phi(t) \\ \omega(t) \end{bmatrix}$, where, $x(t), y(t), v(t), \phi(t), \omega(t)$ are x-position, y-position, velocity, yaw angle, yaw rate respectively and $\tilde{q}(t)$ is noise.

## 2.3 Clustering

Clustering is the classification of data into groups such that all the data in one group are related/similar to each other than to the data in other groups [18].

Several different clustering techniques have been developed over years [19]. Different clustering algorithms can be implemented for different applications and each has its own set of advantages and disadvantages. The 2 different clustering techniques used within the thesis are as follows -

### 2.3.1 K-means clustering

This clustering algorithm groups the multi-dimensional data into a predetermined number of clusters. The groups/clusters are such that the euclidean distance of all the data to the centroid of the cluster is minimized (i.e minimize the sum of point-to-centroid distances).

For the desired number of clusters K, The algorithm initializes K number of random points and labels them as cluster centroids. Then, the nearby data points are associated with the closest centroid and form a cluster. This is followed by the recalculation of the centroids. These steps occur recursively until the solution converges to global minima. That is, the partition of any of the clusters would lead to a higher cost. The K-means Clustering algorithm is summarized below

**Algorithm 1** K-means Clustering

**Require:** Data points, Number of clusters desired - K
**Ensure:** Data points grouped into K clusters
  K Random points are initialized as centroids
  **repeat**
    Associate Nearby Data points to the cluster centroids
    Compute new centroids for the clustered group
  **until** convergence
  **return** Clustered Data

K-means does a great job at classifying the data into a set number of clusters as desired. However, the k-means algorithm is dependent on the initialized center point (seeds) i.e. running K-means with different initial centers could lead to different clusters. This leads to inconsistent clustering between different runs. k-means clustering also fails to detect any pattern/shape in the data as k-means clustering is purely based on the distance of the data points to the centroids.

### 2.3.2 DBSCAN

DBSCAN stands for *density-based spatial clustering of applications with noise* is a density-based clustering technique that does a great job in clustering arbitrary shapes and detecting outliers. DBSCAN algorithm groups all points that are close to each other based on distance and a minimum number of points.

The most important parameters in the DBSCAN algorithm are *eps* which is the maximum distance between the 2 points for them to be associated with the same cluster; *minPoints* which specifies the minimum number of neighboring points required for the point to be considered as a core point [20].

The DBSCAN clustering algorithm initializes by labeling a data point as a core point if there are more than *minPoints* data points within the *eps* distance. All the points which fall within these parameters are labeled as core points. The points that have at least 1 core point as a neighbor are called Border points. All the points

that are neither core nor border points are labeled as outliers. The algorithm goes through all the data points and labels them into the core, border, or outlier points. Therefore, after labeling all the points, an edge is created between all core points that are within *eps* of each other. The connected core points are converted into a cluster and the associated border points are assigned to the same cluster. This step is repeated until all the core points and border points are clustered into their respective groups.

The following is a summary of the algorithm:

**Algorithm 2** DBSCAN Clustering

**Require:** Data points, Eps, minPoints
**Ensure:** Clustered Data points and Outlier points
Label points as core, border, or noise points.
Put an edge between all core points that are within Eps of each other.
Group connected core points into clusters.
Assign the boundary point to the core point cluster it belongs to.
**return** Clustered Data

The DBSCAN algorithm does not require the number of clusters to be stated beforehand, which is fantastic but also an issue since it does not enable us to specify the number of clusters. The algorithm clusters the data based on $minPoints$ and $eps$. The DBSCAN algorithm also performs well when clustering clusters of arbitrary shapes [20].

## 2.4 Smoothing

Smoothing is a filtering technique where the state estimate for the next time step is computed using the past, present, and future data. i.e. all N measurements are accessible and this information is used to better estimate the possible state $\hat{x}_{k|N}$ at time step k.

Smoothers are usually a two-pass algorithm. The forward pass is a standard Kalman filter (EKF, UKF, or others) and the Backward recursion (the second pass) is used to reduce the inherent bias and improve the predictions from the forward pass. Some of the applications where smoothing techniques are used are as follows:

In communication systems, decoding the whole message after it is received

In sports, determining where the ball bounced in tennis or if the ball crossed the goal line in football.

In medicine, arterial blood pressure sequences can be used to estimate intracranial pressure.

There are different types of smoothing algorithms as follows-

**Fixed point Smoother** - Estimates the states at a fixed point of time in the past. i.e. compute $p(x_k|y_{1:K})$ at a fixed time k as K grows and more data is collected.

**Fixed lag smoother** - Estimates the state at a fixed delay in the past. i.e. compute $p(x_k|y_{1:k+n})$ for all k and a fixed n $> 0$. It resembles filtering, but a small-time delay on the estimates is accepted.

**Fixed-interval smoother** - Estimates the states on the interval [0, T] given measurements in the same interval. i.e. find $p(x_k|y_{1:K})$ for all k and a fixed K.

### 2.4.1 Rauch-Tung-Striebel (RTS) smoother

RTS Smoother is a two-pass fixed-interval smoother. A conventional Kalman filter method is used for the forward pass. The RTS smoother (backward pass) operates backward in time i.e. start at the most recent time step and works backward in time. The use of a forward pass estimator and a backward pass is thought to have fully exploited all available data [21]. As a result, the method can do better estimates than the forward pass. The RTS smoother's backward recursion is summarized in the equation below:

$$A_k = P_{k|k}\Phi_k^T P_{k+1|k}^{-1} \tag{2.6}$$

$$\hat{x}_{k|n} = \hat{x}_{k|k} + A_k(\hat{x}_{k+1|n} - \hat{x}_{k+1|k}), k = N-1, \ldots, 0 \tag{2.7}$$

$$P_{k|n} = P_{k|k} + A_k[P_{k+1|n} - P_{k+1|k}]A_k^T \tag{2.8}$$

where:
A is the smoother gain matrix
N is the final time step
$P_{k|n}$ is the corresponding state error covariance matrix
$\hat{x}_{k|n}$ is the smoothed states of $k^{th}$ time step

The summary of the algorithm is as follows:

**Algorithm 3** RTS Smoothing Algortihm

**Require:** Using the forward pass, store the output states and covariance, as well as the prediction state and covariance - $\hat{x}_{k|k}$, $\hat{x}_{k|k-1}$, $P_{k|k}$, $P_{k|k-1}$.
$K \leftarrow$ number of time steps
**Ensure:** Smoothed states $\hat{x}_{k|n}$ and covariances $P_{k|n}$
**for** $k \leftarrow K-1:1$ **do**
$A_k = P_{k|k}\Phi_k^T P_{k+1|k}^{-1}$
State vector, $\hat{x}_{k|n} = \hat{x}_{k|k} + A_k(\hat{x}_{k+1|n} - \hat{x}_{k+1|k})$
Covariance matrix, $P_{k|n} = P_{k|k} + A_k[P_{k+1|n} - P_{k+1|k}]A_k^T$
**end for**

# 3
# Methods

In this chapter, firstly, different simulation environment options are explored and discussed. This is followed by a brief introduction to the scenario used throughout the project work. Finally, the non-causal tracking algorithm developed is presented.

## 3.1 Simulation Environment

The ground truth data represents the real condition of the vehicle at any given time step and can be utilized for bench-marking, whilst the radar detections are used as input to the tracking algorithm. To create ground truth data for the vehicles as well as radar detections, a simulation environment is necessary. For this purpose, it was possible to work with 2 different approaches for the tracker and the input data.

Firstly, for the tracker, it is possible to use a real tracker (provided by APTIV) that is used in real cars or use a simulated tracker (a MATLAB-based tracker). The real tracker is APTIV's intellectual property. Therefore, it must be treated as a black box and this would limit the control over data that can be accessed from the tracker itself. Hence, a MATLAB-based tracker was selected as it offered more flexibility with modifications and adapting to the needs of this thesis.

Secondly, real-life data and/or synthetic data were considered for input data, but it was determined that synthetic data (simulated scenarios) would be more viable because it would allow us to manipulate the scenario as needed. The real-life scenarios can be easily replicated in the simulated environment and therefore also offer greater control over the scenario. MATLAB-based **Automated Driving Toolbox** was used since it is capable of generating, customizing, and simulating the required scenario. The toolkit includes functionality for producing synthetic radar readings and modeling vehicle movements.

## 3.2 Scenario construction/overview

For reasons discussed in the section 3.1 , the automated driving toolbox is used to generate synthetic scenarios. This allows easy replication of the real-life scenarios and also gives the freedom to develop unique driving scenarios. A MATLAB-based tracker is used for tracking the objects in the developed driving scenarios.

The scenario considered for the development of the non-causal algorithm throughout this thesis is presented in figure 3.1 below.

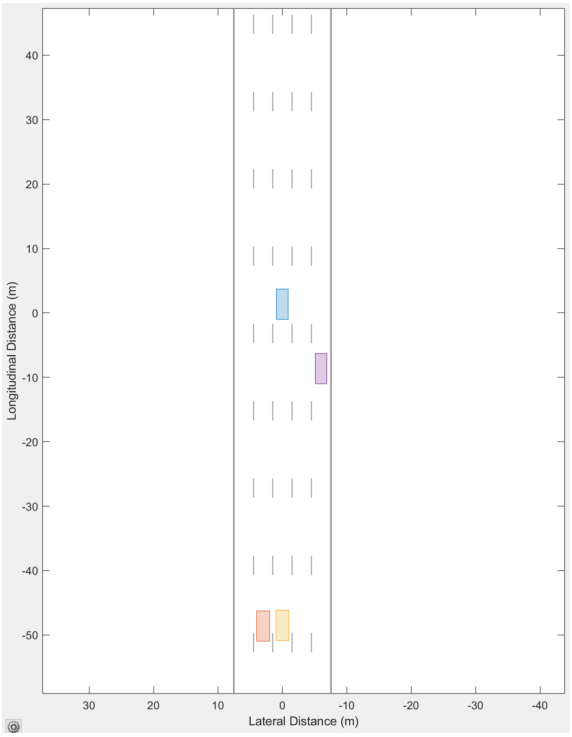



**Figure 3.1:** The Driving/Road Scenario. Blue vehicle is the ego vehicle and the other vehicles are actors.

Here, a highway scenario is considered where the blue vehicle is the ego vehicle. The ego vehicle is fitted with a 360° Field of View (FOV) RADAR on top of the vehicle. The ego vehicle is moving in the center lane at a constant speed. The surrounding vehicles (in purple, orange and yellow) are all part of the driving scenario as actors. All the vehicles (actors) are moving at a constant speed greater than the ego vehicle's speed but in the same direction as the ego vehicle i.e the actor vehicles will be performing an overtaking maneuver on the ego vehicle.

As seen from figure 3.1, the orange and yellow actor vehicles in the adjacent lanes are really close to each other. As the yellow actor vehicle approaches the ego vehicle (in blue), it changes lane to the right to perform the overtaking maneuver.

## 3.3 Online tracker

Object tracking is performed on detection data to estimate or forecast the position of the object in subsequent time step. Conventionally, some form of Kalman filter technique is used to perform object tracking. However, A GM-PHD tracker is used in this thesis to track the input detections.
The object trajectories, states, and detection information are utilized by the non-causal algorithm and therefore it is important to have a tracker that can be used

for the same. The GM-PHD filter was readily available and with minimal modifications/tuning, the filter could be used for tracking objects and also extracting the required data (which includes object trajectories, state vectors, and the detection information as mentioned before). Hence, the GM-PHD tracker is used for this thesis.

Since, a highway scenario (as mentioned in section 3.2) is adopted in the development of the algorithm in this thesis, a constant turn-rate model is more efficient. This is because, on a highway, most cars maintain a steady speed and do not make any aggressive or random turns, with the exception of lane changes, which represent a very minimal change in direction. As a result, a MATLAB-based 'Constant turn-rate rectangular target motion model' (CTRECT model) is used. The CTRECT model differs slightly from the constant turn-rate model discussed in section 2.2.2. The states of CTRECT motion model are described as follows -

$$X_k = [x, y, s, \theta, \omega, L, W]^T \tag{3.1}$$

where, $(x, y)$ are the position of the rectangle center in $x$ and $y$ direction respectively (in $m$); $s$ is the speed in the heading direction (in $m/s$); $\theta$ is the orientation angle of the rectangle (w.r.t $x$ direction) (in $degrees$); $\omega$ is the yaw rate (in $degrees/s$); $(L, W)$ are the length and width of the rectangle respectively (in $m$).

The type of scenario considered in this thesis is **under-segmentation** scenarios. i.e. two vehicles close to each other are sometimes tracked as a single object (by the tracker). This can be seen in the figure 3.2.
It can be interpreted from figure 3.2 that the detections are close to each other and the tracker fails to classify them as different objects.
There are two possible reasoning for this -

1. The detections are so close to each other that the tracker assumes that they are coming from 1 single object and therefore only output one single track for these objects.
2. The merging threshold (as discussed in section 2.2) in the GM-PHD tracker is so high that it merges the 2 peaks that come from these detections/tracked target and therefore only show one object as a result of this. It is possible that the threshold is tuned such that the tracker outputs this as two different tracks, but then the performance of the tracker in other scenarios might be affected. For example, a lower threshold will result in several other tracks that do not exist in real life. Therefore, it is a trade-off for the overall better performance of the tracker.

Furthermore, when the object is adjacent to the ego vehicle (known as being in the 'cone-of-silence'), the state estimates are not good/accurate because all detections reflect the side of the item, leaving the tracker with no information about the object's extent. As a result, the tracker performs poorly in estimating the object's state. The track wears off in different directions, and the existing track must be pruned when a new track appears after the measurement update (in the GM-PHD filter). When

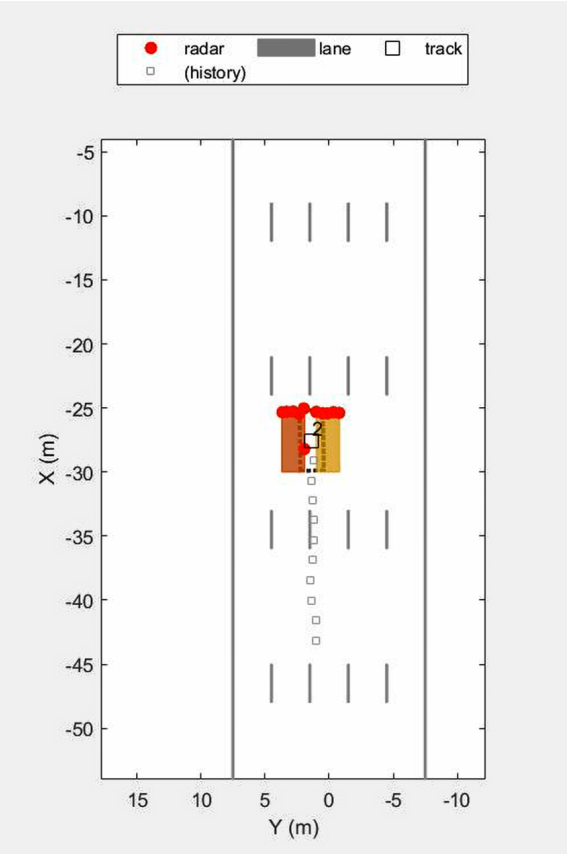



**Figure 3.2:** The Under-segmentation Scenario

the object is in the cone-of-silence zone, this almost always results in the formation of short-lived new tracks, which is another scenario that the offline tracker algorithm can simply capture and correct. This thesis, however, does not provide a solution for this problem because it is beyod the defined scope.

## 3.4 Data generation and pre-processing

The built-in GM-PHD tracker is run on the desired scenario and the tracker generates the track data as the output. The output data is structured as follows -

| Field Name | Description |
|---|---|
| TrackID | Unique integer to identify different active track |
| UpdateTime | The time the track was updated |
| Age | Number of timesteps the track has survived |
| State | value of state vector at the update time |
| StateCovariance | Uncertainty Covariance matrix for the state estimate |
| IsConfirmed | logical value: True if the track is assumed to be of a real target |

**Table 3.1:** Structure of TrackData

Several other data fields are collected in addition to track data for use in the non-

causal algorithm. This includes all detection data from each time instance, as well as ground-truth data, predicted states, and more. Further, The Ego Vehicle states, Sensor configuration and Sensor transform Parameters among other variables are saved from the online tracker.

The track data is then processed to extract and organize relevant information for the non-causal algorithm. Firstly, a list of all the tracks (with unique track id) is created. Next, this list is populated with more information such as the time-step when the track first appears, the age of the track, etc. The GM-PHD tacker sometimes generates ghost tracks that last for a very short duration (like a few time-steps). These short-lived tracks are deleted from the track data to produce clean data. An absolute threshold of 10 is set; i.e. all tracks below this threshold are deleted. This threshold value was chosen based on knowledge gathered by carefully examining tracker performance in different scenarios. The resulting track data information is important and is utilized later in the non-causal algorithm.

## 3.5 Non-Causal Algorithm

The following sections provide a detailed overview of the non-causal algorithm developed in this thesis work. In brief, the algorithm processes the track information obtained from the online tracker algorithm. The algorithm, first, computes a cost associated with a track and then splits the track into 2 by clustering the detections associated with that into two different groups. The tracker is then used again in a simulated environment (fed with this clustered data) to generate tracks. If the generated tracks overlap, a cost is added to the overall cost of the track. Next, the tracks are split and merged based on the cost to select a tack with the least cost. The algorithm runs sequentially through all the tracks until the cost cannot be reduced further. A simplified overview of the algorithm is presented in figure 3.3.

### 3.5.1 Clustering the input detections

The built-in GM-PHD tracker does not provide information about detection association to the tracks. This is a major drawback of the GM-PHD tracker and therefore it becomes important to cluster the detections based on the track information (processed in the previous step) in a discrete step.

The radar sometimes generates false detections which do not originate from real objects and hence can be classified as outliers. Including these outliers in data will significantly affect the clustering of actual target detections.
It can be observed from figure 3.4 that the clustered detections do not fit the objects (in grey) properly. Therefore, it can be concluded that outliers that are represented with black dots can negatively impact clustering. Therefore it is important to get rid of these outliers.

So, firstly, DBSCAN clustering technique is used to remove the outliers from the

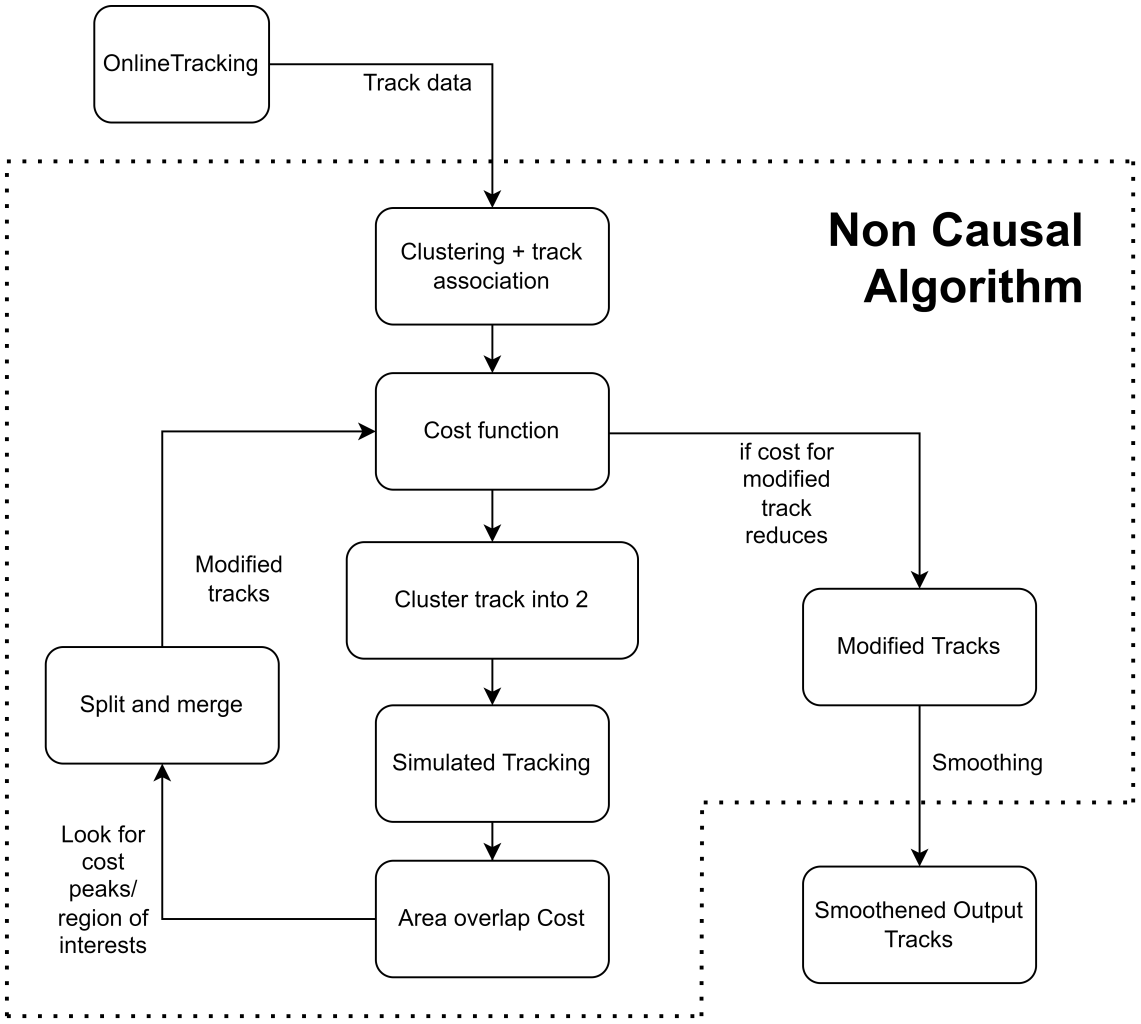



**Figure 3.3:** Block Diagram implementation of Non-Causal Algorithm.

input set of detections. Once outliers are eliminated from the detection data, k-means clustering is run on the clean data.
Figure 3.5 shows how the generated clusters fit the objects well after using both the DBSCAN and K-means algorithms.

The number of clusters that will be generated is determined by the number of tracks in the track data at a given time step. The K-means clustering method can only divide radar detections into n unlabeled partitions on its own. There is no association in clustering from one-time step to the next i.e. if there are 2 tracks in time-step 1 and 2; there is a possibility that the clustered group is associated with track 1 in the $1^{st}$ time step but with track 2 in the $2^{nd}$ time-step. Therefore, there must be a uniformity in labeling that is maintained between the time steps to rightly associate the clustered detection group to the right track. This is performed by comparing the Euclidean distance between the centroids of the cluster group from 1 time-step to another. The clustered detection group (in time-step 2) which is close to another clustered detection group (from time-step 1) is associated with the same track. This allows for maintaining the right association of the detection to the track itself. The

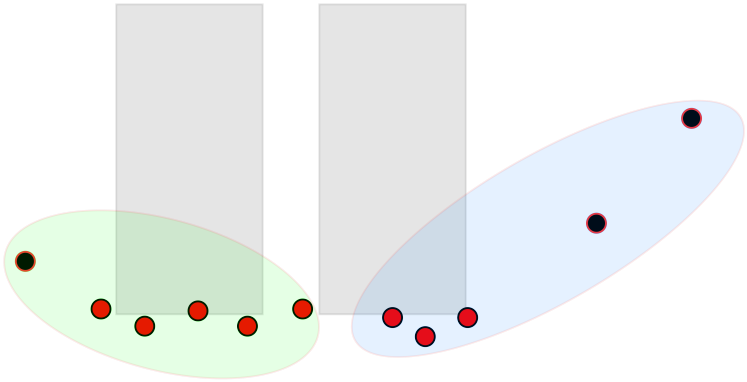


**Figure 3.4:** A scenario where only K-means with n = 2 is applied directly on the detection data.

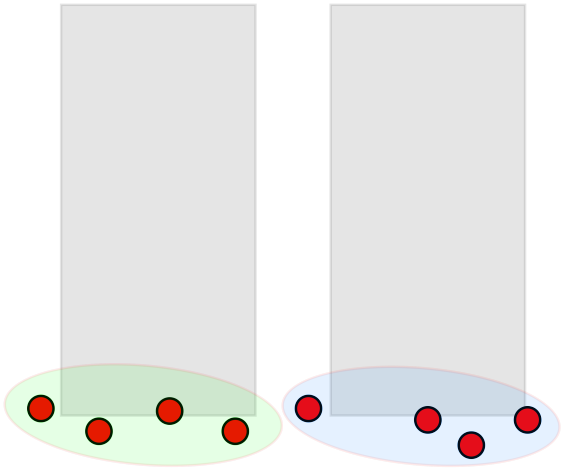


**Figure 3.5:** A scenario in which outliers are removed using the DBSCAN method, and subsequently clean detection data is processed using K-means with n = 2.

algorithm has been summarized below

### 3.5.2 Cost function

Energy minimization techniques have become relatively widespread within the domain of multi-object tracking [22]. The general objective is to build a function that assigns a cost to each conceivable alternative before determining the state with the lowest cost. There are numerous approaches to defining an energy function for a specific application. In this thesis work, an algorithm is modeled such that a cost is computed for each track, and then after some split and merge moves, the cost is recomputed. A decision to split/merge is made based on the new cost.
The Cost function used in this thesis is a linear combination of 3 components.

$$E = E_{Det} + E_{Prox} + E_{Area} \tag{3.2}$$

$E_{Det}$ is the cost associated with the proximity of the detection to the state estimate; $E_{Prox}$ is the cost associated with the proximity of track origin to the ego vehicle,

**Algorithm 4** Clustering Detections

**Require:** Detection points, Number of clusters desired - n, and track information
  $x \leftarrow$ Detections
  $eps \leftarrow$ min Distance between 2 points
  $mpints \leftarrow$ min number of points
**Ensure:** Detection data associated to right track
  **for** $i \leftarrow$ Starting index : End index **do**
    $CleanDet_i \leftarrow$ DBSCAN $(x_i, eps, mpints)$
    n $\leftarrow$ Number of tracks from track information
    $[Centroid_i, idx_i] \leftarrow$ Kmeans$(CleanDet_i, n)$
    **if** $i > 3$ **then**
      $ecd_{dis} \leftarrow$ EuclideanDistance $(Centroid_{i-1}, Centroid_i)$
      **if** miss-match in $ecd_{dis}$ **then**
        Switch $idx_i$ values
      **end if**
    **end if**
    **for** $n \leftarrow 1 : NumberofTracks_i$ **do**
      $\text{Det}_{clustered} \leftarrow$ Associate($Det_{clean,i}$, $Ecd_{dis,i}$, $idx_i$) Detections belonging to the same track are grouped together
    **end for**
  **end for**
  **return** Clustered Data

and $E_{Area}$ is the penalty cost associated to physical exclusivity of the state estimate. The remainder of this section delves deeper into each cost term and its function.

| Symbol | Description |
|---|---|
| $X^t$ | (x,y) world coordinate position of Estimated State at time t |
| $D_i^t$ | (x,y) world coordinate position of Detection i at time t |
| $E^t$ | (x,y) world coordinate position of Ego vehicle at time t |
| $N^t$ | Number of detections at time t |
| C | Decay constant |
| rng | The range of the radar |

**Table 3.2:** Notations

### 3.5.3 Detection proximity cost

In this thesis, the Detection proximity term's primary goal is to keep trajectories as close to the detection as possible. To put it another way, The goal of this cost component is to assign a high cost to scenarios in which the detected item does not/poorly matches the detections and a low cost to scenarios in which the tracked object 'fits' the detections. The cost is modeled as the normalized Euclidean distance between the detections and the state estimate for a particular time step as follows:

$$E_{Det} = (\sum_{i=1}^{N^t}(D_i^t - X^t))/N^t \tag{3.3}$$

A simple euclidean distance between the detection and the state estimate was chosen over other more complex distance metrics such as Minkowski Distance or Manhattan Distance. For example, the detections' proximity to the bounding box was considered. However, it is more difficult since the vehicle's bounding box boundary must be computed first, followed by the detection point's proximity to the bounding box border. Further, the simpler concept of normalized Euclidean distance of the detections (normalized by the number of detections in every time instance) to the state estimate gave satisfactory results and hence was opted as a method for this thesis.

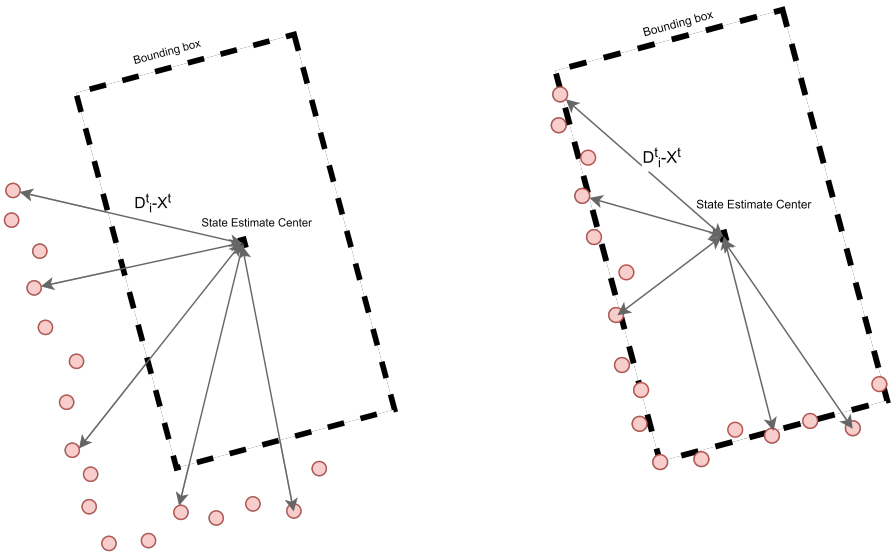



**Figure 3.6:** Example for the calculation of Detection Proximity cost. The red dots are detection points and the dotted rectangle is the bounding box representation of the object around the state estimate (aka centroid).

From figure 3.6, the Detection proximity cost is high for the scenario on the left as the detections are further away from the state estimate. Whereas, the Detection proximity cost is low for the scenario on the right. It can therefore be interpreted that normalizing the error in distance will yield good results in most cases.

### 3.5.4 Track initiation cost

The track initiation cost is the expense associated with a track's start being close to the ego vehicle. In practice, a track cannot be initiated at random near the ego car. The tracker should be able to track an object from distance i.e. when the target object enters the FOV of the radar. For example, on a highway, The targets do not randomly pop up next to the ego vehicle. As a result, the cost of starting a track is proportional to its proximity to the ego vehicle. Tracks that start closer to

the edge of the FOV (of the radar) are penalized negligibly. While the tracks that originate close to the ego vehicle are severely penalized. The cost is modeled as an exponential decay function as follows -

$$E_{Prox} = (rng * 10) * (C)^{((\sum_{i=1}^{N(1)}(D_i^1)/N(1)) - E^1)} \tag{3.4}$$

The cost function is tuned to punish the tracks that originate very close to the ego vehicle. The exponential factor is the distance between the average position of the detection in the first time instance of a track to the position of the ego vehicle. Due to the exponential decay nature of the cost function, the cost decreases exponentially as the distance increases.

### 3.5.5 Physical exclusivity cost

The physical exclusivity cost is the cost associated with the intersection of the object area. The cost variant in our case is the area overlap between the 2 tracked objects. A higher overlapping area leads to a higher cost. In a physical sense, 2 or more objects cannot occupy the same space and this detail can be used to penalize the tracks when the two tracks are overlapping each other. This cost can be very important when developing the offline tracker as a whole but within the scope of this thesis, physical exclusivity cost has been primarily used to detect the under-segmentation. This has been further discussed in Section 3.5.8.

### 3.5.6 Clustering the track into 2

Once the initial cost of the track is computed, the detections associated with the track are split into 2 along the direction of travel of the track. This is done to check if the detections have been associated with the right number of tracks i.e. if there exists a single track and only 1 track in ground truth, the detections when split will still be associated with that single track. However, if there exists a single track but 2 tracks in ground truth, the detections will be split and be associated with two tracks.

So, the K-means clustering algorithm is used to divide the Cartesian radar detections into two clusters i.e.

$$clusters = kmeans([D_x, D_y], 2) \tag{3.5}$$

As a result, the clustered detections will be partitioned as indicated in Figure 3.7.

### 3.5.7 Simulated tracking

Once the detections are split into 2 groups, each group of detections are passed into the tracker algorithm, consecutively, to generate the track data from these detections.

Since the track data generated for each group of detections will be compared to one another, it is important to force the tracker to track all the input detection data as a single object. As a result, key tracking algorithm parameters have been tweaked to output single tracks more consistently.

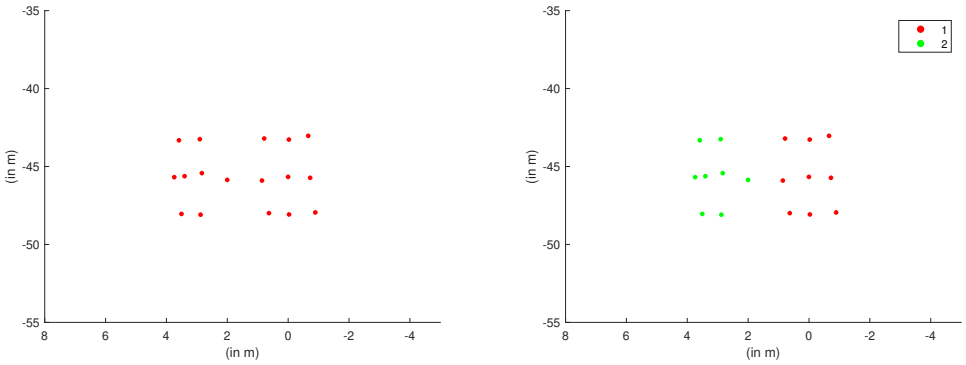



**Figure 3.7:** Clustering detections into 2 groups. On the left, it is group of detections from a single time instance. On the right, 2 clustered groups after running k-means clustering algorithm.

### 3.5.8 Overlapping area cost

Overlapping area cost is physical exclusivity cost (section 3.5.5) which has been adapted to detect the under-segmentation issue. The states from each time-step for the 2 tracks (after running the tracker on split detections (associated to a track) into 2 ) are overlaid on top of each other. There are 2 possibilities -
Ideally, since the detections are associated with one single track, the states will perfectly overlap each other. Therefore, it can conclude that all the detections are associated with the same track and only 1 track exists.

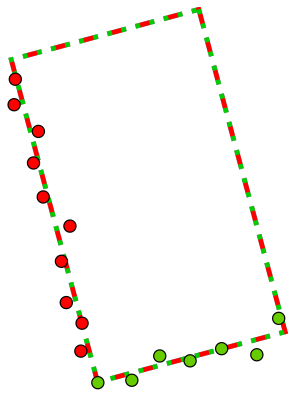


**Figure 3.8:** Perfect area-overlap. The detections are divided into two groups, but they are all associated with the same single track, thus the state estimates from both cluster groups completely overlap.

On the other hand, if there are 2 different tracks originally, the states will not overlap each other. Therefore, it can be assumed that there are 2 different tracks and it is a good idea to perform jump moves to change the configuration of the track i.e create new tracks or merge different tracks.

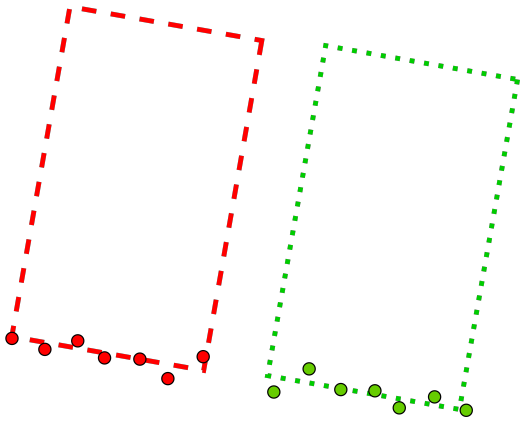


**Figure 3.9:** No area-overlap. The detections are divided into two groups, but because the detections are connected with separate tracks, the state estimates from both cluster groups do not overlap.

The physical exclusivity cost can be used to determine where along the length of the track to perform the jump actions. Let's consider the under-segmentation scenario. If we look closely at the track which suffers under-segmentation, it can be seen that tracks move from a region of no overlap to a region of significant overlap.

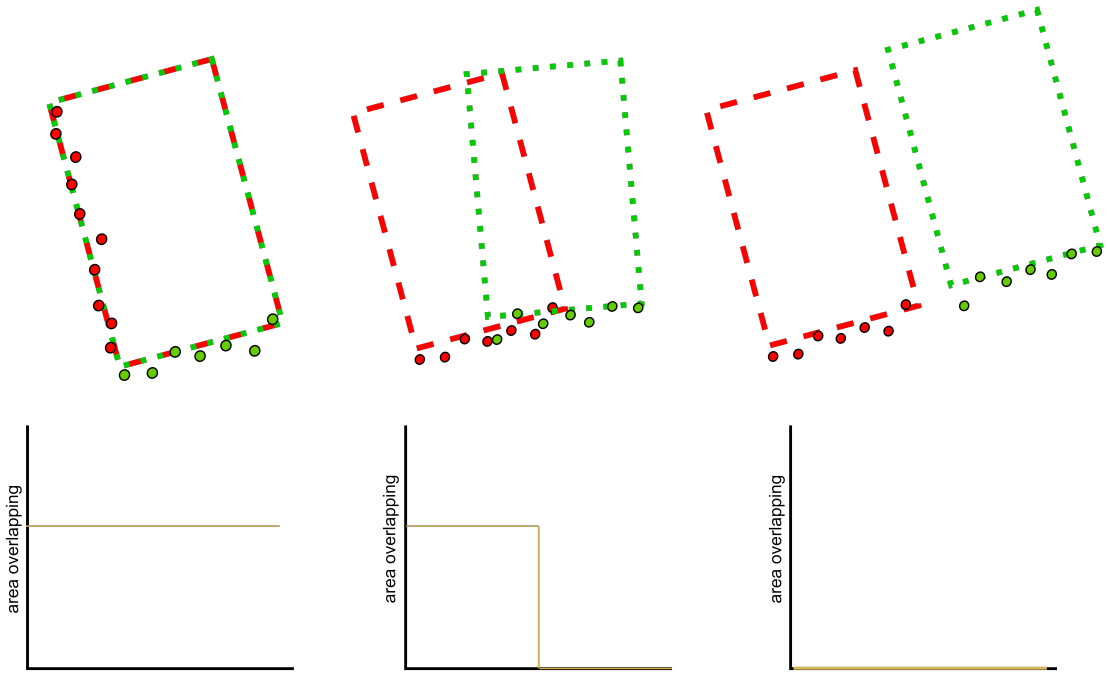



**Figure 3.10:** Moving from a region of no overlap to a region of significant overlap.

It can be interpreted from figure 3.10 that there is a rapid change in the area overlap during the transition time (figure in the middle). This change in area overlap is captured by subtracting consequent area-overlap measurements. In the region where the overlap area is constant ( both - no overlap and complete overlap), the resulting difference will always be zero. However, when the tracks transition from a region of no overlap to a region of complete overlap, there is a spike in the difference in consequent area overlap measurements. The time step corresponding to this spike is the region of interest, and here is where the jump actions are performed.

### 3.5.9 Split and merge - jump actions

Once a region of interest is identified from the previous step, some jump moves are performed. The jump moves include splitting the track and merging it with some other track.

**Algorithm 5** Jump Moves

**Require:** Track information, Region of interest points, Detection information
  $s \leftarrow$ Index where there is a spike in difference cost (aka Region of Interest) from the previous step
  $Track_{split,1} \leftarrow$ Split track 1 from the main track
  $Track_{split,2} \leftarrow$ Split track 2 from the main track
  **for** $k \leftarrow 1:2$ A track is usually split into 2 and therefore the loop runs twice **do**
    **for** $i \leftarrow 1 : NumberOfTracks$ **do**
      Compute *Endpoint* Avg. position of the Detections from index $s-1$ belonging to the $track_k$ we are interested in
      Compute *Startpoint* Avg. position of the Detections from 1st index belonging to $track_i$
      $Dis_{k,i} \leftarrow$ Start point - End point
    **end for**
  **end for**
  $[row, col] \leftarrow$ get index for $Dis$ less than 2 metres.
  **if** $row, col$ are not empty **then**
    $Track_{new,1} \leftarrow$ merge $Track_{split,row}$ before index $s$ and $track_{col}$ after index $s$
    $Track_{new,2} \leftarrow$ merge $Track_{split,oppositeofrow}$ before index $s$ and $track_{row}$ after index $s$
  **end if**
  **return** $Track_{new}$

### 3.5.10 Compare the cost

After the jump moves are performed, the cost is re-computed for the new tracks. As discussed in Section 3.5.2, the Detection Proximity Cost and Track Initiation Cost are used for continuous comparison of the track. If the total cost for the new track reduces, the new track is confirmed to be more optimized. If the cost does not reduce and/or remains the same, there will be no jump moves performed on the track and as a result, the track remains the same.

At this stage, The algorithm runs recursively for the same track until the solutions converge, i.e. the cost does not reduce for 2 successive recursions. Once the algorithm is out of the recursive loop, the algorithm is applied to the next track in the scenario.

### 3.5.11 Smoothing

The updated detection data are fed into the GM-PHD tracker to generate newly updated track data once the algorithm exits the recursive loop. As we only pass detection data for a single track, only one output track is expected, hence a slightly modified GM-PHD tracker is utilized to provide the relevant track data. To reduce the number of output tracks, certain GM-PHD tracker parameters - 'Extraction-Threshold' and 'ConfirmationThreshold' have been raised to 0.9 and 0.97, respectively, and the 'Merging Threshold' has been raised to 150. This allows the tracker to output only a single track more consistently. The predicted state from the tracker is also saved to be used later while smoothing.

RTS smoother algorithm (as discussed in Section 2.4.1) is next run on the newly generated filtered state estimates and predicted states (from the GM-PHD tracker). The resulting smoothed states are saved and plotted against filtered state estimates and Ground-truth data.

# 4

# Results

This section seeks to evaluate the performance of the thesis's non-causal tracking algorithm. Evaluating performance is a lengthy and difficult task due to a large number of conceivable scenarios. Furthermore, the tracking algorithm has parameters that must be tuned, and the ideal set of parameters differs depending on the scenario. This thesis makes use of a simulation environment enabling simulating the same scenario multiple times with varied parameters. Each scenario was handcrafted in an attempt to recreate a real-world phenomenon. Instead, a small number of examples are chosen to illustrate some of the algorithm's advantages and disadvantages by utilizing credible algorithm parameters.

This section is organized so that the different scenarios are provided, followed by their findings.

## 4.1 Tuning track initiation cost parameters

The track initiation cost discussed in section 3.5.4 is an exponential decay function. Hence, the output is expected to decay exponential as the input increase. It can be inferred from figure 4.1 that by varying the decay constant, the rate of decay changes.

The purpose of track initiation cost is to punish tracks that originate very close to the ego vehicle, it was found after some tuning that C = .89 yielded the best compromise for the rate of decay of the cost.

The resulting equation looks as follows -

$$E_{Prox} = (rng * 10) * (C)^{((\sum_{i=1}^{N(1)}(D_i^1)/N(1))-E^1)} \tag{4.1}$$

from figure 4.2, if the track initiates 2 m away from the ego vehicle, then the penalty is $\approx 80$ (high) and if the track initiates 20 m away from the ego vehicle, the penalty is $\approx 10$ (low). The radar range used for in figure4.2 is an example. In practice, the radar range is much greater and the cost function is much more representative.

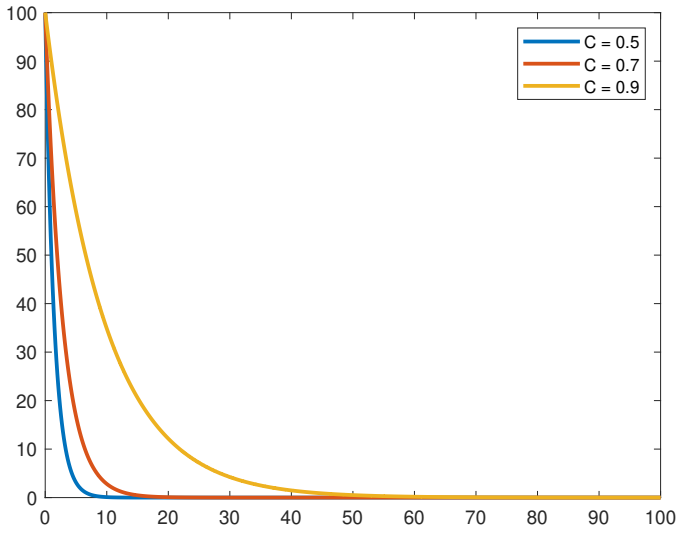



**Figure 4.1:** Output of exponential decay function for varying decay constant, C.

## 4.2 Tuning GM-PHD tracker for simulated tracking

For the purpose of simulated tracking, it is important to only output one track from the tracker algorithm. Hence, some parameters (discussed in section 2.2.1) were modified as follows to obtain the desired results. Extraction threshold and confirmation threshold were raised to .9 and .97 respectively such that only the highest peaks (of the target probability) were confirmed and selected. Further, Merging Threshold was also raised to a higher limit of 150 as this allows for nearby peaks to be merged into 1 single peak. Hence, with this tuning, the number of output tracks was mostly limited to 1 track as desired.

## 4.3 Scenario 1

This thesis was primarily concerned with the scenario of under-segmentation (described in detail in 3.3). One of the most important outcome from this thesis has been the ability to split an under-segmented track into 2 different tracks. The scenario as described in section 3.2 is represented in the figure 4.3.
In figure 4.3, the radar sensor is located at origin (0,0). Each point is a detection that originates from an object. These detections are then grouped together and are associated to an object. As the object moves in space, a track is formed (seen as trail of detection points) and this can be seen clearly in figure 4.3. There are multiple tracks in the scenario and , all the detections in red form a track which starts when the object is approximately 5 $m$ behind the ego vehicle on the right and travels in a straight line for almost 45 $m$ in front relative to the ego vehicle position.
It can also be seen in the image that a lot of detections are grouped together in

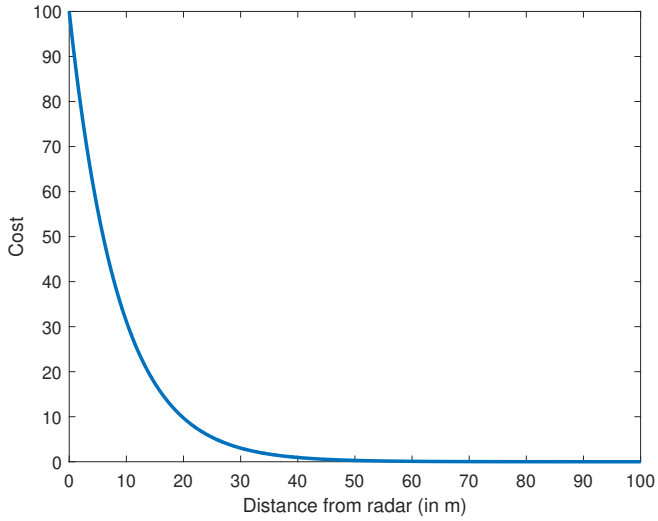



**Figure 4.2:** Track Initiation cost function with C = 0.89. and radar FOV, rng = 10 m.

green. In reality, there are 2 vehicles very close to each other and therefore the detections coming of those vehicles are very close to each other. As a result, the tracker associates the detections to a single vehicle and outputs a single tracked object, which is the cause of the under-segmentation problem. As the distance between the detections increases (approximately 20m before the origin), the tracker initiates a new track (seen in blue).

The non-causal algorithm was run multiple times and yielded the same result for this scenario. However, a randomly selected simulation run was selected to demonstrate the algorithm's behavior in action rather than through aggregated data. The cost error values for each track from the randomly selected simulation run is summarized in table 4.1.

| **Iteration** | **Track 1 (in Red)** | **Track 2 (in Green)** | **Track 3 (in Blue)** |
|---|---|---|---|
| 1 | 577.29 | 91.54 | 219.72 |
| 2 | 577.29 | 87.25 | 102.38 |
| 3 | 577.29 | 87.25 | 102.38 |

**Table 4.1:** Cost error values from running the non-causal algorithm for scenario 1.

Once, the non-causal algorithm is run on this scenario, the output tracks are as seen in figure 4.4.

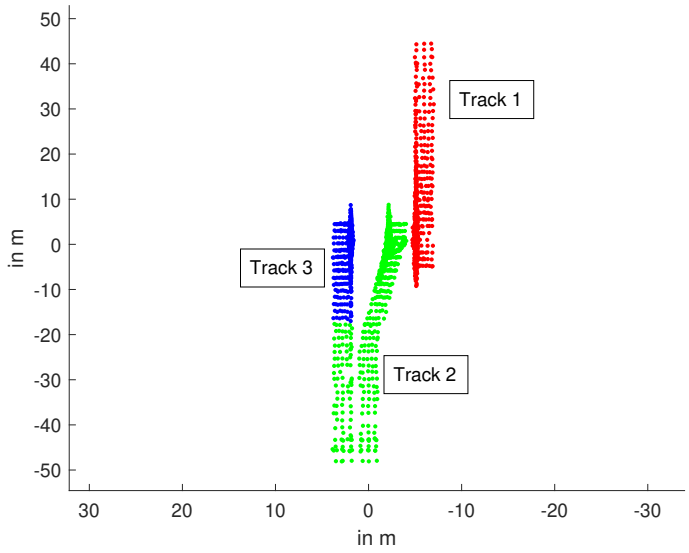



**Figure 4.3:** The Detections associated to different tracks for the original input scenario to the non-causal algorithm. The detection points that are linked to the same track are colored the same.

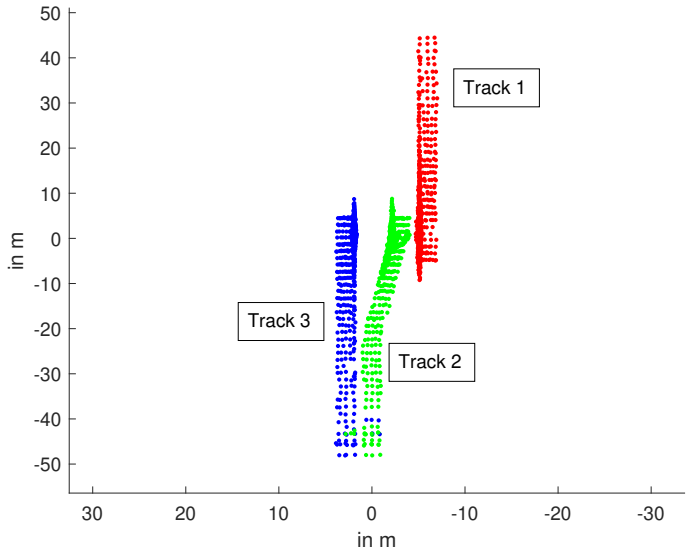



**Figure 4.4:** The Detections associated to different tracks after running the non-causal algorithm.

It is clear from 4.4 that the non-causal algorithm does a great job in detecting and resolving under-segmentation.
The image above shows that the tracks that were previously recognized as a single track have now been divided into two separate tracks, as they should be. It can also be interpreted from table 4.1 that -

1. **Red track -** After executing the non-causal algorithm on the detections linked with red track multiple times, the cost stays the same. This is a promising

outcome from the non-causal technique because all the detections are coming from a single object. The method does not divide or merge the track with others since it is an independent track throughout, and hence the cost associated with this track remains constant throughout numerous iterations.

2. **Green track -** The cost has been reduced for this track from the first run to the next as the non-causal algorithm splits the under-segmented track to form a single track associated with the object. Initially, the tracker used detections from two tracks to create a single track, but after the non-causal technique, the tracker only tracks the item with detections linked with that object. This results in a cost reduction. Once the track has been split, the cost does not drop further as the track is associated with the right number of objects.
3. **Blue track -** It is not possible for an object to randomly appear so close to the ego vehicle on a highway. As a result, the cost is initially fairly high because the track originates quite close to the ego vehicle. Further, as the green track is split and merged with the original blue to form the true blue track, the cost of the overall track reduces. There is a significant drop in cost and this is primarily because of dependence on the proximity of the track to the ego vehicle.

**Smoothing results -**

Once the non-causal algorithm outputs the new split detections, the tracker algorithm runs on these detections and the new tracks are generated. These new tracks are smoothed to improve the state estimates at a given time. The smoothing algorithm uses the future state information to improve the state estimates in the current time. The smoothed state estimates for the tracks are shown in the figure 4.5-

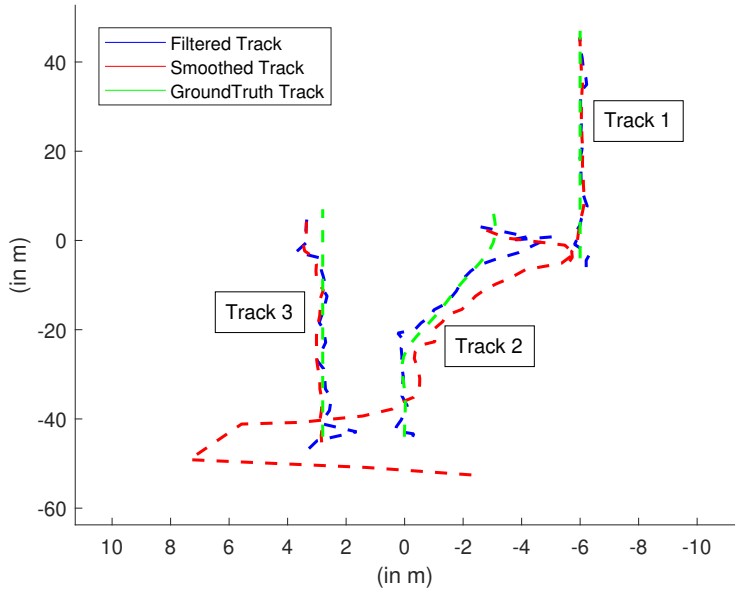



**Figure 4.5:** The smoothed state estimate of the tracks described in scenario 1.

The graphic above shows that the smoothing algorithm does a decent job of smoothing the states for tracks 1 and 3, but not so much for track 2. The smoothed states

are derived from the filtered and predicted states (from the online pass) and therefore can only improve the states based on this information. Hence, the smoothing algorithm cannot drive the states to be closer to the ground truth. However, the smoothing algorithm does smoothen the states and removes any aggressive/jerky motion in the filtered states. The absolute error charts for each track shown below demonstrate the same -

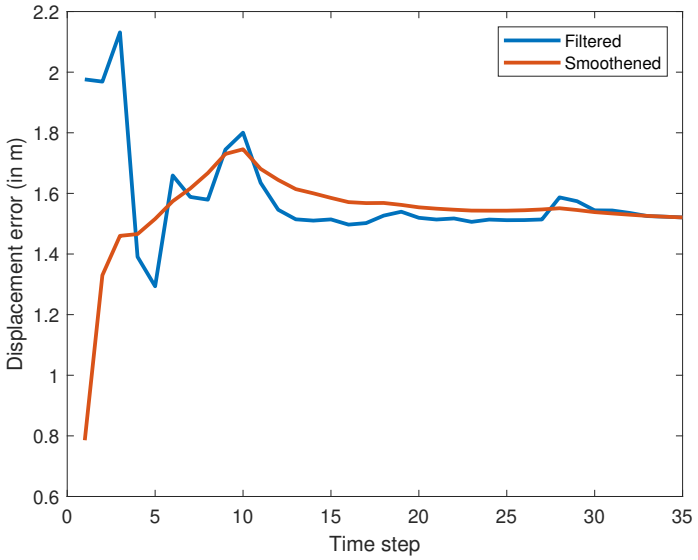



**Figure 4.6:** The absolute distance error plot of the smoothed and filtered states from the ground truth for track 1 in red.

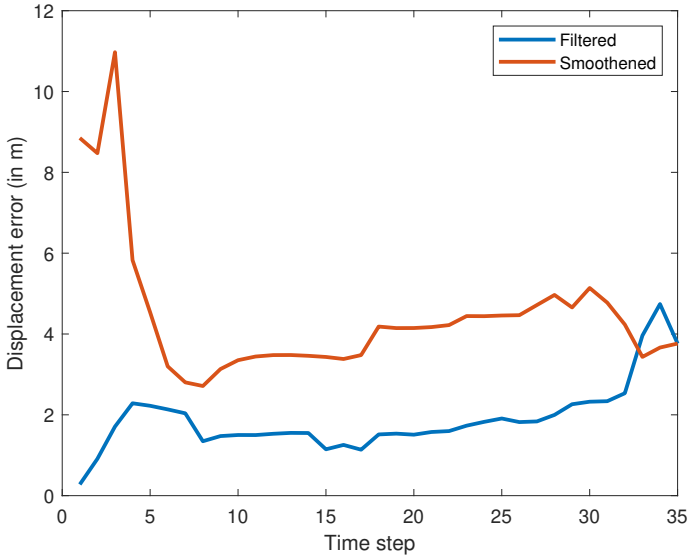



**Figure 4.7:** The absolute distance error plot of the smoothed and filtered states from the ground truth for track 2 in blue.

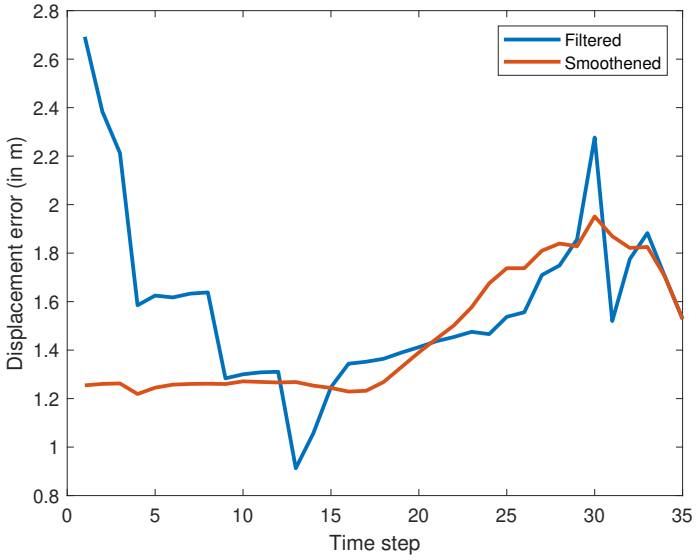



**Figure 4.8:** The absolute distance error plot of the smoothed and filtered states from the ground truth for track 3 in green.

The error for the smoothed states for tracks 1 and 3 (track in red and green respectively) are relatively lower than the filtered states. The smoothing algorithm calculates the smoothed states using the filtered and predicted states, as well as covariances. As a result, if the covariances are large, the smoothed states will be considerably off, as illustrated for track 2 in figure 4.5.

## 4.4 Scenario 2 - a limitation

The non-causal algorithm developed as part of this thesis has its own set of limitation. In the following scenario, the object takes a prolong time to switch lanes and there is a significant delay before the second track is initiated. The non-causal algorithm does detect the second track but fails to resolve the under-segmentation. The scenario is represented in figure 4.9.
The cost error values for each track from the randomly selected simulation run is summarized in table 4.2.

| **Iteration** | **Track 1 (in Red)** | **Track 2 (in Green)** |
|---|---|---|
| 1 | 245.80 | 72.54 |
| 2 | 394.99 | 219.78 |
| 3 | 394.99 | 219,78 |

**Table 4.2:** Cost error values from running the non-causal algorithm for scenario 2.

Once, the non-causal algorithm is run on this scenario, the output tracks are as seen in figure 4.10.

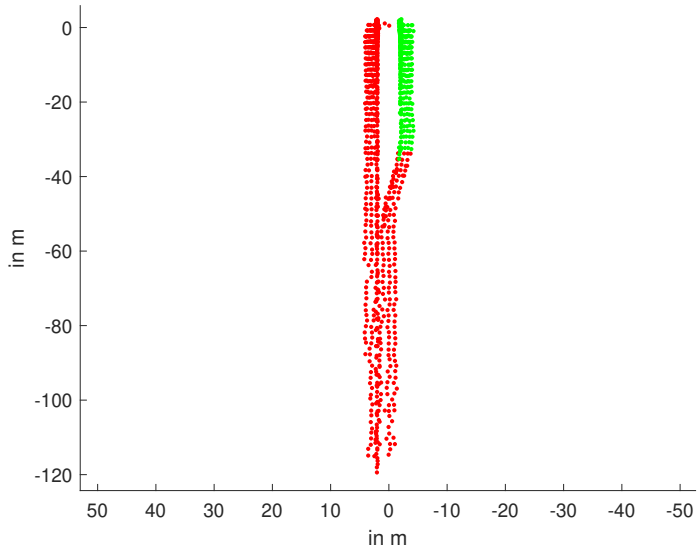



**Figure 4.9:** The Detections associated to different tracks for the original input scenario to the non-causal algorithm. The radar sensor is located at origin (0,0)

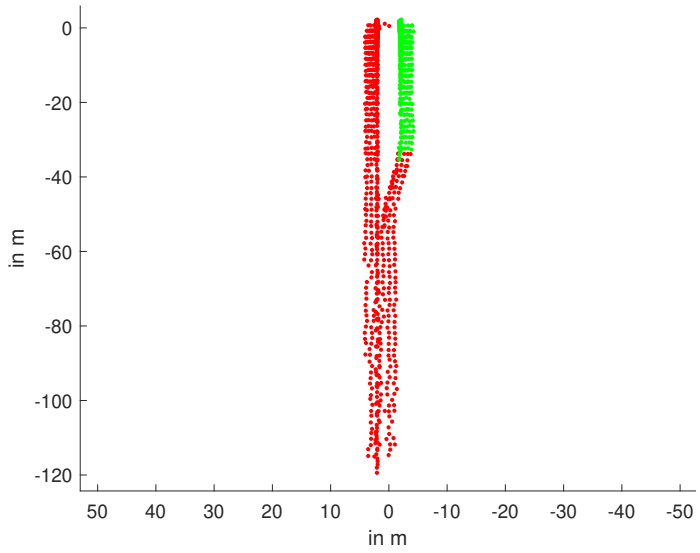



**Figure 4.10:** The Detections associated to different tracks after running the non-causal algorithm.

It is clear from figure 4.10 that the non-causal algorithm does a great job in detecting but fails in resolving the under-segmentation. As discussed earlier, due to the late initiation of the $2^{nd}$ track, when the track is split at the region of interest ( as described in section 3.5.8 and 3.5.9 ), there is no nearby track to merge the split track with and therefore the non-causal algorithm fails to resolve the under-segmentation in such scenarios.

# 5

# Discussion

This chapter focus on reflecting on the challenges, choices, and decisions made while developing the non-causal algorithm. Some of the challenges faced in object tracking have also been brought up as they are closely connected.

Offline tracking is a broad topic with a lot of potential. This thesis has concentrated on a single aspect/issue (under-segmentation) amid a plethora of other difficulties that can be remedied via Offline tracking. A lot of research is available on the topic of offline tracking for image tracking applications but not so much for radar tracking applications. Therefore, there's a lot of untapped potential to improve online tracking by extending the capabilities of offline radar tracking and hence improve vehicle/occupant safety.

The majority of the work in this thesis has gone into establishing the structure for the offline-tracking environment as a whole but the focus has been on resolving the 'under-segmentation' phenomenon. Resolving the under-segmentation issues can be considered as a single module among the many that will make up the overall offline-tracking environment. Additionally, several features/tools have been created to detect and remedy the issue of under-segmentation. The area-overlap calculation, for example, is a unique cost that aids in the detection of under-segmentation (and this can also be extended to detect over-segmentation). As a result, many pieces can be added to this basis to improve and add more features.

Several concepts are utilized in the thesis that can easily be improved although left in their present conditions due to lack of time. Furthermore, because this algorithm has been developed from the ground up and with access to limited prior research, some of the foundations are based on concepts borrowed from other sources, leaving numerous areas for improvement. One such key concept is the energy minimization technique which is quite widely used for image tracking applications. In image tracking, the objects are first labeled/classified and can then be directly used for state estimation and tracking. But in the case of radar tracking, the detections have to be first clustered and associated with the right object before the objects can be used for state estimation and tracking. Hence, handling detection data adds an additional layer of unavoidable complexity (to radar tracking) when implementing concepts borrowed from image tracking applications. The cost function components in this thesis were inspired and developed based on the energy minimization concept used by Anton Milan in [22].

## 5.1 Scenario

When two objects are close to each other (traveling in the same direction and with same velocity) for an extended period of time, such as on a highway, under-segmentation is more likely to occur. The algorithm for detecting and resolving under-segmentation was developed primarily for a highway scenario. As a consequence, there are a few scenarios that were targeted by the work done in this thesis. Further, it is assumed that the tracked object moves away from the second object/performs a lane change, allowing a new track to be initiated.

## 5.2 Tracker

Tracking objects is a very complex task and depends heavily on input detection data. Any change in detection data will affect the estimation of the bounding box and thereby affect the output track estimates. Further, the yaw of the vehicle and the positional measurement relies on the quality of the bounding box estimation. Figure 5.1 presents some situations where the bounding box estimation can be vulnerable. Hence, there are occurrences when sufficient information is not available about detection data and then the resulting tracking is not very accurate. Due to poor bounding box estimation, the covariance is high and this then negatively affects the tracking and other aspect such as smoothing operation.

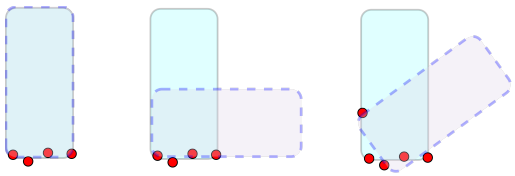


**Figure 5.1:** Situations where bounding box estimation is vulnerable. The detections are in red. The box in blue is the real vehicle and the bounding box is represented by purple dotted box.

Further, when dealing with under segmentation issue, since the detection points are cluttered in a small region, the tracker could associate the detections to a single large object such as a truck or bus. It can be observed from figure 5.2 that the tracker could estimate the same set of detections in different ways and yield very contrasting bounding box estimates and tracks. This once again produces very inaccurate object states and covariances.

This thesis work has used the GM-PHD tracker and is not an exception to the above-discussed inconsistency. Advanced tracking algorithms could be used to improve the tracking accuracy and thereby improve the performance of the offline tracking algorithm. However, if a very powerful and accurate tracker is utilized, there is no need for an offline tracking technique as the tracker's output is already quite accurate.

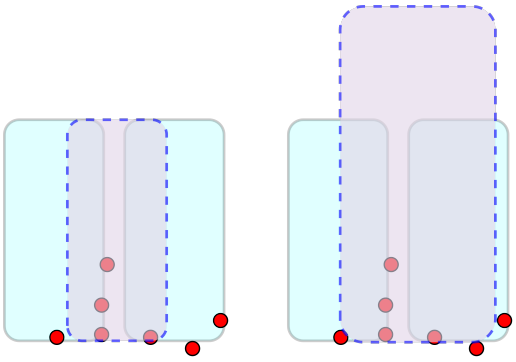


**Figure 5.2:** Scenario where bounding box estimation is vulnerable. The detections are in red. The box in blue is the real vehicle and the bounding box is represented by purple dotted box.

## 5.3 Clustering

The GM-PHD tracker did not output the information about the detection's association with the track. Hence, it was necessary to have a step to associate the detections to the right track as this information was often required in the development of the algorithm. As a result, this problem is quite similar to the data association problem, which involves linking a radar detection to the right object. Advanced/well-known data association techniques were not investigated. Instead, the effort was focused on developing a specific solution that would use the track information (from input track data) to cluster the radar detections into required groups, which was more in line with the thesis's premise.

As discussed in section 2.3, the DBSCAN clustering technique was used to eliminate noisy radar detections, and then the K-means clustering technique was used to cluster the remaining detections into the expected number of tracks.

Although the method yields promising findings, it is vital to keep in mind that the results could be an artifact of the simulation environment. Several aspects that are likely to be unfavorable to this technique in the actual world are not accounted for in the simulations. To name a few circumstances that would invalidate the approach of known cluster formation, the target vehicle could be hidden by other objects, or radar detections might not reflect as predicted from all areas of the target vehicle.

Further, It's also worth noting that the clustering was performed just on the positioning information of the detections; however, it may be of great interest to use the velocity information as well, since this could potentially provide superior results.

# 6
# Conclusion

This thesis has delved into the concept of offline radar tracking and an algorithm was proposed to detect and resolve the under-segmentation phenomenon specifically. Inspired by energy minimization approach, a cost function was developed to calculate the cost of the tracks before and after a jump move was executed. Splitting and combining tracks to produce new tracks based on cost attributes is known as a jump move. Furthermore, the state estimates for the new tracks (resolved from under-segmentation) were smoothed using a smoothing function.

The algorithm was tested on a few synthetically produced scenarios. A highway scenario with vehicles performing an overtaking maneuver on the ego vehicle was examined and the algorithm did a good job in detect and resolve the under-segmentation here. However, the algorithm fails to perform as expected when the scenario differs.

The groundwork was laid for creation of an offline radar tracking by utilizing principles that exists but required adaptions. It is a difficult and enormous task to develop an universal offline tracker and requires future work to further develop the current algorithm into a complete offline radar tracker.

# 7
# Future Work

Only particular scenarios with vehicles driving in the same direction as the ego vehicle were examined in this study. It would be a natural step forward to investigate instances with more difficult surroundings, such as junctions. Although the algorithm has not been evaluated in other contexts, it is likely that issues may develop. Because repairing under-segmentation is a module of Offline Tracker, there are a lot of different circumstances that can be caught in the offline environment with additional information. When an object travels over a cone of silence, for example, new tracks are sometimes created; this can simply be corrected with an offline tracker.

Furthermore, using several sensors has its own set of problems, such as no tracking when the object travels from one sensor's FOV to another's. Sensor fusion techniques [23] assist in tracking the object, but as previously noted, they might result in the generation of new tracks, therefore an offline tracker will be invaluable in such situations.

Creative approaches, such as B-Spline Chained Ellipses Model [24], Learned Structural Measurement Model [25] may be used to determine the object's extent, which will only improve the offline tracker.

# Bibliography


[1] Vincent Tabora. Why making cars self-driving is so difficult, Feb 2022.

[2] Yassine Ruichek, Fadi Dornaika, and Maan El Badaoui El Najjar. Sensors technologies and methods for perception systems in intelligent vehicles. *Journal of Sensors*, 2016:1–1, 07 2016.

[3] R. Rasshofer and Gresser K. Automotive radar and lidar systems for next generation driver assistance functions. *Advances in Radio Science - Kleinheubacher Berichte*, 3, 05 2005.

[4] Robin Heinzler, Philipp Schindler, Jürgen Seekircher, Werner Ritter, and Wilhelm Stork. Weather influence and classification with automotive lidar sensors. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1527–1534, 2019.

[5] Ralph H Rasshofer and Klaus Gresser. Automotive radar and lidar systems for next generation driver assistance functions. *Advances in Radio Science*, 3(B. 4):205–209, 2005.

[6] Subhash Challa, Mark R Morelande, Darko Mušicki, and Robin J Evans. *Fundamentals of object tracking*. Cambridge University Press, 2011.

[7] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es, 2006.

[8] Karl Granstrom, Marcus Baum, and Stephan Reuter. Extended object tracking: Introduction, overview and applications. *arXiv preprint arXiv:1604.00970*, 2016.

[9] Laura Leal-Taixé, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth. Tracking the trackers: an analysis of the state of the art in multiple object tracking. *arXiv preprint arXiv:1704.02781*, 2017.

[10] Mobility Insider. What is ground truth?, Sep 2021.

[11] Merrill Ivan Skolnik. Introduction to radar systems. *New York*, 1980.

[12] Michael Parker. Chapter 18 - radar basics. In Michael Parker, editor, *Digital Signal Processing 101 (Second Edition)*, pages 231–240. Newnes, second edition edition, 2017.

[13] C Neipp, A Hern ndez, J J Rodes, A M rquez, T Bel ndez, and A Bel ndez. An analysis of the classical doppler effect. *European Journal of Physics*, 24:497–505, 9 2003.

[14] Ba Tuong Vo. *Random finite sets in multi-object filtering*. Citeseer, 2008.

[15] Michele Pace. Multi-target tracking with phd filters.

[16] Gustaf Hendeby and Rickard Karlsson. Gaussian mixture phd filtering with variable probability of detection. In *17th International Conference on Information Fusion (FUSION)*, pages 1–7, 2014.

[17] James M. Joyce. *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[18] Enrique H Ruspini. A new approach to clustering. *Information and control*, 15(1):22–32, 1969.

[19] Richard Dubes and Anil K. Jain. Clustering techniques: The user's dilemma. *Pattern Recognition*, 8(4):247–260, 1976.

[20] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.

[21] Arthur Gelb et al. *Applied optimal estimation*. MIT press, 1974.

[22] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72, 2013.

[23] R Omar Chavez-Garcia. *Multiple sensor fusion for detection, classification and tracking of moving objects in driving environments*. PhD thesis, Université de Grenoble, 2014.

[24] Gang ; Yao, Perry ; Wang, Karl ; Berntorp, Hassan ; Mansour, Petros T ; Boufounos, Philip V Orlik, G Yao, P Wang, K Berntorp, H Mansour, P Boufounos, and P V Orlik. Extended object tracking with automotive radar using b-spline chained ellipses model, 2021.

[25] Yuxuan Xia, Pu Wang, Karl Berntorp, Petros Boufounos, Philip Orlik, Lennart Svensson, and Karl Granstrom. Extended object tracking with automotive radar using learned structural measurement model. volume 2020-September. Institute of Electrical and Electronics Engineers Inc., 9 2020.

**DEPARTMENT OF ELECTRICAL ENGINEERING**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY