Data association algorithms for multi target tracking in a probabilistic framework

Sharma, Prabhat K.

Award date:
2021

Department of Mechanical Engineering
Dynamics and Control Research Group

Data association algorithms for multi target tracking in a probabilistic framework

Prabhat Kumar Sharma | 1431390
M.Sc. Automotive Technology

DC 2021.076

TU/e Supervisor(s):
dr. ir. T.P.J.v.d.Sande
prof. dr. Henk Nijmeijer

TNO Supervisor(s):
Robin Smit M.Sc.
Frank Evers M.Sc.
Alexis Siagkris M.Sc.

Eindhoven, Monday 6th September, 2021

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

10/08/2021

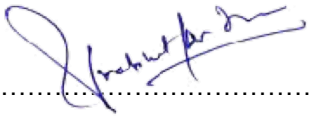
Name

Prabhat Kumar Sharma

ID-number

1431390

Signature



Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Abstract

In order to take maneuvering decisions for its safe and efficient navigation, an autonomous vehicle is required to keep track of various objects in its vicinity so that it can predict their (relative) motion trajectory. To that end, a multi-target tracking (MTT) algorithm is used. Various sensors are employed to detect the relevant objects or targets. The task of the MTT algorithm is then to use these measurements to recursively update the estimated states of the targets like position, velocity, heading angle, etc., and maintain a coherent track. One of the main challenges in MTT is assigning appropriate measurements to the corresponding tracks. This step is referred to as data association. TNO currently uses global nearest neighbor (GNN) data association in its multi-target tracker. However, this association method faces some difficulties in performing the measurement-to-track assignment when a large number of measurements are present and the targets are located relatively close to each other.

The measurement-to-track association task is far from trivial as there are many uncertainties involved like unknown and time-varying number of targets, noisy measurements with spurious instances, the possibility of missing the targets due to occlusion, uncertain motion behavior of the targets, etc. This study aims to investigate other data association methodologies in a probabilistic framework that can handle these uncertainties and performs better than GNN in real-time. Based on the literature study, multi hypothesis tracking (MHT) is chosen for further investigation as it shows great potential to account for the underlined uncertainties involved in the data association step. Both GNN and MHT algorithms are then implemented in a multi-sensor simulation environment. Various test scenarios were developed in simulation to capture different ambiguities which a tracker needs to resolve in a real-world application. The performance of both the algorithms is then evaluated in these test scenarios using different performance metrics. The relevant tuning parameters for each algorithm are identified and their effect on tracking performance is also studied. Simulation results show that MHT does outperform GNN in challenging dynamic scenarios. Furthermore, the performance of GNN was found at par with MHT in simple scenarios with less ambiguity.

Preface

This report describes the work carried out during my graduation thesis as a part of my automotive technology masters program at TU/e, in close cooperation with the integrated vehicle safety department of TNO. There are few people whom I would like to thank for their kind support and guidance. First of all, my sincere gratitude goes to prof. dr. Henk Nijmeijer for his guidance and suggestions throughout this project. I appreciate the time he provided for this project even after his retirement. Big thanks to my TU/e supervisor dr. ir. T.P.J.v.d.Sande for his guidance in planning the project, encouraging me to dig deep into the concepts, and help in providing structure to my work. Apart from the technical discussions, I appreciate his help in understanding myself as a person. I would also like to thank my primary TNO supervisor, Robin Smit, for insightful discussion over practical aspects, and for reminding me to always see the bigger picture and ask relevant questions to ourselves. Thanks to my secondary supervisors at TNO, Alexis Siagkris and Frank Evers, for their short but effective suggestions. Special mention to my friends, Siddhanth, Shreyas, Diya, and Ashwin, for the laughs, food, and fun we shared during these tough times.

Finally, a special thanks and regards to my parents, Shri Jyoti Bhushan Sharma, Smt. Renu Devi and my brother Prakash Sharma for their unconditional love, support, and blessings, without which nothing would have been possible.

Contents

Nomenclature

ARSI	Ambiguity Resolution Success Indicator
AV	Autonomous Vehicle
CMA	Cumulative Moving Average
CNN	Convolutional Neural Network
CPHD	Cardinalized Probability Hypothesis Density
EKF	Extended Kalman Filter
FAF	False Alarms per Frame
FM	Fragmentation
FOV	Field Of View
GNN	Global Nearest Neighbour
GOSPA	Generalised Optimal Sub-Pattern Assignment
IDS	ID Switch
JPDA	Joint Probabilistic Data Association
KF	Kalman Filter
LLR	Log Likelihood Ratio
MCMC	Markov Chain Monte Carlo
MHT	Multiple Hypothesis Tracking
ML	Mostly Lost
MODA	Multiple Object Detection Accuracy
MODP	Multiple Object Detection Precision
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
MT	Mostly Tracked
MTT	multi-target Tracking
ODD	Operational Design Domain
OSPA	Optimal Sub-Pattern Assignment

PDF	Probability Density Function
PF	Particle Filter
PHD	Probability Hypothesis Density
PMBM	Poisson Multi-Bernoulli Mixture
PPP	Poisson Point Process
PT	Partly Tracked
RFS	Random Finite Set
RL	Recover from Long-term occlusion
RMS	Root Mean Square
RNN	Recurrent Neural Network
RS	Recover from Short occlusion
SPRT	Sequential Probability Ratio Test
TCI	Track Continuity Indicator
TCSR	Track Continuity Success Ratio
TDE	Tracking Distance Error
TO-MHT	Track-Oriented Multiple Hypothesis Tracking
X2V	Everything To Vehicle

Chapter 1

Introduction

1.1 Background

Autonomous vehicles (AVs) have drawn the attention of both, the industry and the research communities in recent years [30]. In order to take maneuvering decisions, an AV needs to have a proper understanding of its environment. The process of modeling this environment in AVs is generally termed as world modeling and it is considered as one of the key aspects in the realization of a fully self-driving vehicle [7]. One of the vital components of world modeling is multi-target tracking (MTT). It is responsible for keeping the track of different objects such as pedestrians, cars, bicycles, etc., emerging and moving at different speeds, in the field of view (FOV) of the ego vehicle. Thus, just like a human driver, an AV keeps track of relevant objects in its FOV using MTT to predict their motion and thereby taking appropriate maneuvering decisions.

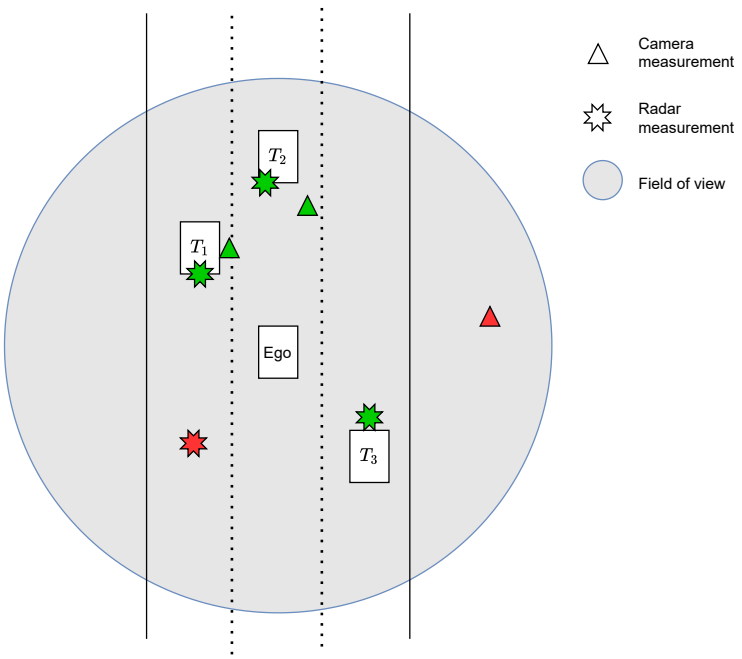


Figure 1.1: Schematic representation of a typical autonomous driving situation showing uncertainties related to measurements generated by sensors. The green detections represents the measurements generated from a true target whereas the red detections are false positives. Also note that target T_3 is missed by the camera sensor

A typical autonomous driving situation is shown in Figure 1.1. Multiple sensors attached to the ego vehicle generate measurements from the objects in their FOV. An object represents any physical entity present in the environment, for example, trees, electric poles, cars, etc. However, not all objects are required to be tracked. The relevant objects that needed to be tracked like cars, pedestrians, cyclists, etc., are called targets. The sequence of poses of a target with respect to time is called its track [40]. The goal of the MTT algorithm is to use the measurements to keep track of these targets by estimating their states recursively. The states represent the relevant information about the target that is of our interest while tracking, for example, kinematic properties like position, velocity, acceleration, etc.

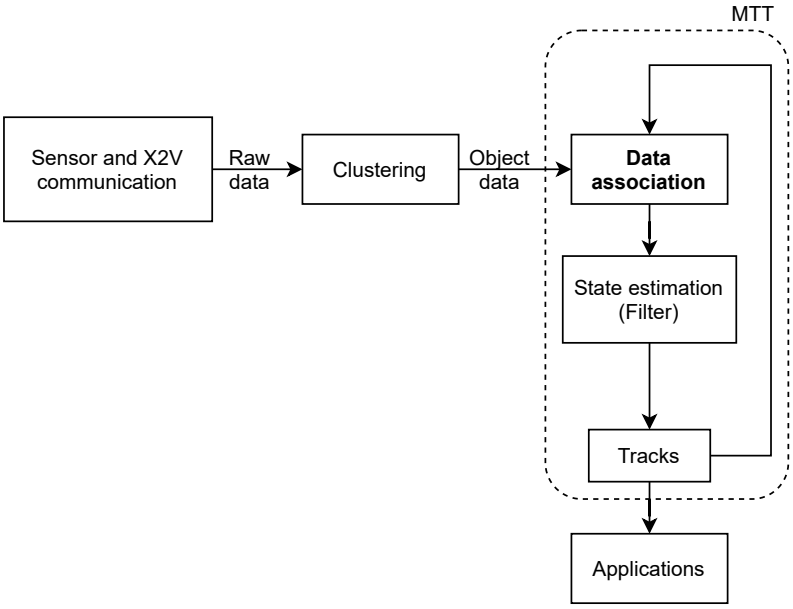


Figure 1.2: High level block scheme for MTT

A high-level block scheme of an MTT algorithm is shown in Figure 1.2. The raw data coming from different sensors and/or X2V communication is first clustered to identify relevant objects and generate object level detections. Many of the modern automotive sensors are capable of filtering their raw data and present the object level detections. The detections include information regarding the measurements of various states of the object like position, velocity, bearing, etc. These object-level detections, henceforth referred to as measurements, are then used to update the estimated states of the targets. To do so, first, the measurements need to be assigned or associated with the corresponding track of the target. This step is called the data association. The associated measurements are then used to estimate the states of the respective targets using state estimation filters like Kalman filter(KF) , extended Kalman filter(EKF) , or particle filter(PF) [22]. However, this data association step is not trivial due to various uncertainties involved with the measurements:

1. The measurements are typically noisy and have spurious instances or clutter, i.e., some measurements are just false positive detections.

2. Targets may get miss-detected for one or several time steps, either due to occlusions or weak signal to noise ratio for the sensor.

3. The number of targets in the field of view varies with time. Objects enter and leave the FOV.

4. Poor resolution of sensors may result in only one detection for many objects placed closely.

Thus, in many MTT literature, data association is referred to as arguably the most difficult and crucial step [48]. This is therefore the main objective of this thesis work.

1.2 TNO's current implementation

The Netherlands Organization for Applied Scientific Research (TNO) is an independent research organization that is recognized by the industry as a valuable knowledge partner with unique expertise, tools, and facilities. The Integrated Vehicle Safety (IVS) department at TNO in Helmond is currently developing a multi-target tracker for cooperative and autonomous driving applications. An overview of the current tracker algorithm is shown in Figure 1.3. It takes the object level measurements from the sensors and V2V communication as input and provides the estimated position of the tracked objects in the vehicle's coordinate frame as output. The tracker employs an EKF for the recursive prediction and update of the target state estimates.

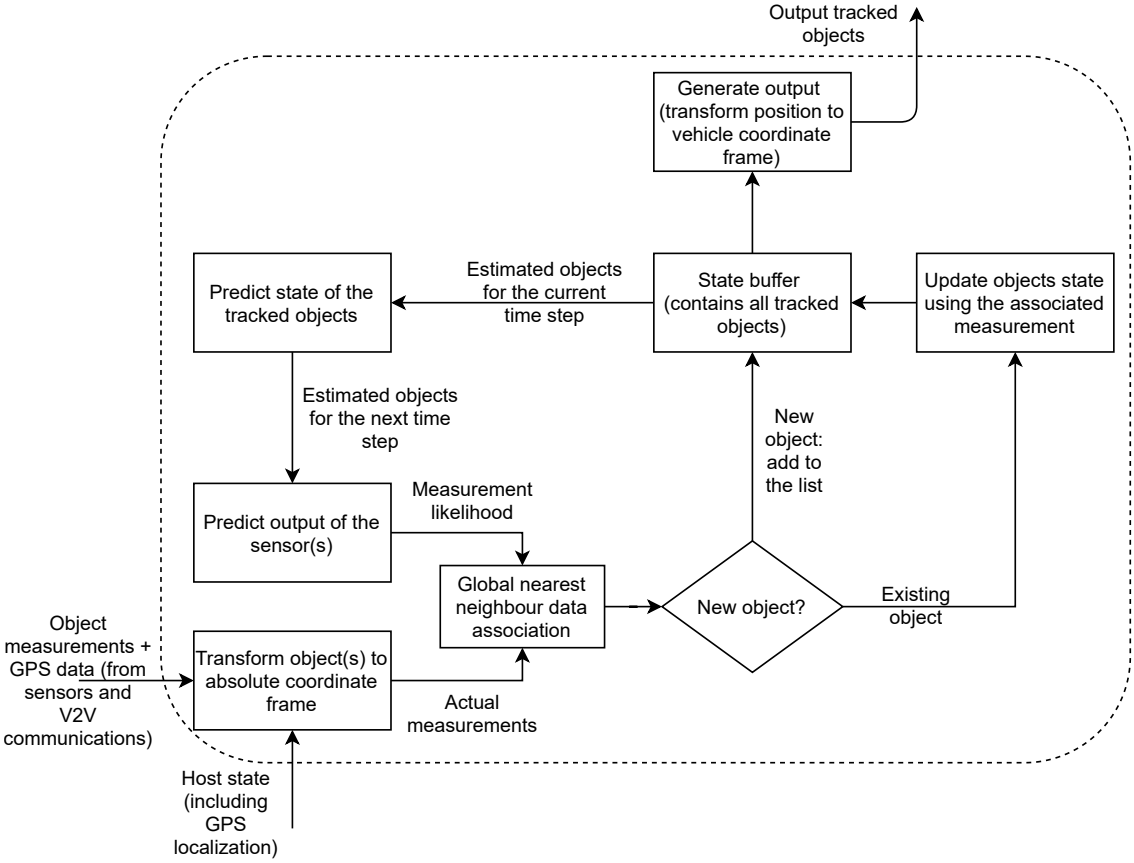


Figure 1.3: TNO target tracker

Utilizing the estimated states of the targets in the current time step, their states are predicted in the next time step using a motion model. Then, given this predicted state, a prediction is made regarding the expected sensor measurements, called the measurement likelihood, using the respective sensor models. Based on this measurement likelihood, the incoming measurements are associated with the corresponding tracks. This implementation uses global nearest neighbour (GNN) [23] algorithm as the data association method. The associated measurements are then used to update the corresponding target's states using the Kalman filter update step. The unassigned measurements are kept in a buffer and based on a pre-defined condition, may spawn a new track or simply be ignored as false detections. Another criterion, called the probability of existence (not

shown in Figure 1.3), is used to discard a current target if no measurement is assigned to it. The updated states of the targets are then used to predict the states in the next time step and the whole process is repeated to estimate the target states recursively.

The GNN data association works well in simple scenarios with few targets distributed sparsely in the FOV and having low clutter with few false-positive detections [23] [4]. It is also simple to implement and computationally cheap. However, there are some shortcomings of this method identified by TNO:

1. An irreversible decision has to be made at each time step regarding the assignment of an incoming measurement to either an existing track or tentatively new track. Thus, there is always a finite chance of wrong association which cannot be corrected once more measurements are available in the future.

2. It cannot handle long-term occlusion as the target age is short in GNN. The occluded or missed target may appear as a new target after a few time steps.

Due to these shortcomings, it performs poorly in challenging scenarios when there are a lot of ambiguities regarding the assignment, for instance, when objects are close to each other or probability of the detection is low. The aim of this study is to look into other data association methods and corresponding MTT algorithms to tackle the above mentioned shortcomings and compare their performance with respect to the baseline GNN method.

1.3 Problem definition and scope

In this section, the multi-target tracking problem and the scope of the project are defined. Firstly the requirements from the target tracker are presented. Based on these requirements, research objectives are formulated. Using these research objectives, the problem is defined and research questions are derived which will be answered after the execution of this project.

1.3.1 Requirement analysis

The multi-target tracking for automated driving application is a complex task. One can imagine the task of the MTT algorithm as that of the cognitive ability of humans that allows them to estimate motion states of different objects in their FOV. There are some general requirements that an MTT algorithm needs to fulfill in order to mimic, if not outperform, that cognitive ability:

- Accurate tracking: It is the primary task of an MTT algorithm. The tracker should be able to track the objects and estimate their position, velocity, bearing, etc. in order to facilitate decision making of the ego vehicle.

- Occlusion handling: Short-term occlusions need to be handled. The algorithm should retain objects in its memory and anticipate their motion even if they are not detected for few time steps.

- Ambiguity solving: A tracker should be able to handle spurious detections, missed detections, and interaction of different objects.

- Real-time performance: Driving involves a highly dynamic environment, thus, real-time tracking is necessary to take appropriate decisions timely.

Moreover, there are some requirements that need to be fulfilled prior to the implementation of such an algorithm in an actual vehicle.

- Robustness and general applicability: The tracker should perform in variety of operational design domains (ODDs) without much tuning. This inherently calls for less parameterization of the algorithm, making it suitable for general applicability.

- Testing and evaluation in the simulation environment: The algorithm should be thoroughly tested in the simulation environment and evaluated based on some quantitative performance metric.

1.3.2 Research objective

Broadly, the task of multi-target tracking consists of two aspects as shown in Figure 1.2. The first aspect is related to the appropriate association of measurements to the tracks such that requirements mentioned in the previous section such as ambiguity solving, occlusion handling, etc. are fulfilled. The second aspect includes estimating the states of the tracks using the associated measurements, which is a state estimation filtering problem. These two aspects are closely dependent on each other as better estimates lead to better measurement likelihood calculation required for data association and better association leads to better estimations. Thus, the overall performance of a target tracker is dependent on both of these aspects. The primary objective of this project is to carry out a comparative study to find a suitable data association method that fulfills all the requirements mentioned in the previous section. To that end, objective performance metrics are used to compare different MTT algorithms. To study the effect of the data association component only on the tracking performance, the Kalman filter will be fixed for all the MTT algorithms. Furthermore, filter tuning might be required to achieve better performance. This becomes the secondary objective of this project.

1.3.3 Scope

In practice, the raw sensor data is first used to extract the object-level data. For instance, the point cloud received from the LiDAR scan is clustered to figure out if the detected point cloud belongs to a car or pedestrian, or some other object. In this thesis, it is assumed that the object-level data is readily available to the MTT algorithm. Furthermore, the targets are assumed to be point objects, i.e., their shape and size are not considered in this study. The kinematic states of the objects like position, velocity, and accelerations are treated as the relevant states for tracking and estimation. All the possible uncertainties regarding the data association step are considered in this study:

1. Unknown and varying number of objects
2. False-positive measurements
3. Miss-detection of objects for one or several time steps

It is also ensured that the implemented algorithms are flexible to accommodate multiple sensors.

1.3.4 Research questions

Based on the established requirements and problem definition in the above sections, relevant research questions that will be answered during this project are:

1. Which data association algorithm works best in a dynamic environment and what is the performance gain achieved with the newly developed algorithm with respect to the baseline GNN method?
2. What are the crucial tuning parameters that can improve the overall performance of the target tracker?

1.4 Report outline

The literature study regarding the various multi-target tracking algorithms is presented in chapter 2. A comparison study of different methods is also presented in this chapter to select a suitable method for implementation. In chapter 3, the complete multi-target tracking problem is defined in a mathematical framework. Apart from the data association, other aspects of the MTT problem like the Kalman filter and measurement setup are defined in this chapter. After fixing all other aspects, the two data association methodologies, namely global nearest neighbour (GNN) and multi hypothesis tracking (MHT) are discussed in chapters 4. The evaluation of these algorithms in different simulation scenarios is then presented in chapter 5. Based on the results obtained from the simulation study, conclusions and recommendations for future studies are presented in chapter 6.

Chapter 2

Literature study

In its most general form, the multi-target tracking problem can be seen as jointly estimating the number of objects and their states from noisy sensor measurements [58]. Initially driven by the aerospace applications in the 1960s, it has been a sought after problem in a plethora of applications such as radar technologies for defense and air traffic control [46], surveillance [54], computer vision [27], robotics [11], automated driving [64], bio information [62], etc. Further texts on various applications can be found in [21] [4].

Broadly, there are two main approaches found for solving the MTT problem in literature [53]. The first approach is the classical *tracking-by-detection*, wherein multiple tracks are updated and maintained by associating the measurements from sensors and/or detectors to the corresponding tracks. Thus, data association is the heart of this approach. The second approach is *deep learning* based, wherein the algorithm is 'trained' to identify objects and estimate their movements using, for example, a convolutional neural network (CNN) or a recurrent neural network (RNN) framework [64]. This approach is relatively new and has drawn the attention of researchers mostly in the last decade. The two approaches are discussed in detail in the following section, enumerating their strengths and weaknesses. Based on this discussion, algorithms will be selected for further evaluation and implementation. A brief literature study comprising various performance metrics used for the evaluation of different MTT algorithms is also presented.

2.1 Tracking by detection

The general framework of the tracking-by-detection approach consists of two steps [36] [41]. The first step is detecting the objects from the raw sensor data. This step is typically performed by feeding raw sensors data to a *detector*, which is usually a trained CNN, capable of producing object-level detection [1]. However, modern automotive sensors already have this capability. The second step is feeding the object-level detections to an MTT algorithm, which typically utilizes a statistical approach for associating the detections to the tracks and thereby tracking the targets. The most prominent data association methods and related MTT algorithms found in tracking-by-detection literature are provided in this section.

One of the simplest approaches for data association is the global nearest neighbour (GNN) [4]. It associates a measurement that is nearest to a track. The nearest is usually referred in *mahalanobis distance* sense [8]. Thus, GNN tries to find the best possible association for a track at each time step. GNN data association does not handle the track initiation and track deletion by itself. Generally, the M/N rule is applied, in which a track is confirmed only if it receives the detection at least M times in the last N time steps [4]. In addition, a threshold (N_D) for consecutive misses is used for the deletion of the track. For example, if a target is not detected for $N_D = 5$ consecutive time steps, it will be deleted [4]. Furthermore, *gating* [4] is used in many practical implementations to avoid nonsense data associations. Based on the predicted state of the target, a statistical region is formed in the measurement space where the measurement is expected.

This region is called the *gate* and is used to filter out the measurements from data association considerations. In simple words, the measurements lying outside the gating region of the track are not considered for the association. Conflicting scenarios, as shown in Figure 2.1, when multiple observations lie within the gate of one track or when one observation is shared by gates of multiple objects, can pose a problem to GNN. Generally, such a scenario is handled in GNN by forming and solving an *assignment* problem. For the given scenario in Figure. 2.1, association problem can be seen as an assignment problem, wherein we intend to assign the three observations (O_i for $i \in \{1, 2, 3\}$) to the three tracks (P_j for $j \in \{1, 2, 3\}$) such that overall *cost* of assignment is minimum. Let cost of assigning each observation-track pair is $L_{i,j}$. This cost can be seen as mahalanobis distance between the observation-track pair. Now, the problem we are left with is to find the best observation-track pairs such that the overall cost of the assignment is minimum, given constraints that (1) only one observation can be associated to one track, and (2) no two tracks can share one observation [4]. Algorithms like Munkers or Hungarian can be used to solve this assignment problem [18]. Out of the two algorithms, Munkers guarantees the global optimal solution, however, it is computationally expensive, whereas Hungarian is computationally cheap but doesn't guarantee a global optimal solution [37]. GNN works well in simple scenarios where targets are sparsely located in a clutter-less environment. However, its performance degrades as clutter and ambiguity increases [23]. This is because GNN makes irreversible decisions regarding the data association at each time step rendering the possibility of a wrong association, especially in cluttered and ambiguous situations [23]. Thus, its application in real-world scenarios is limited.

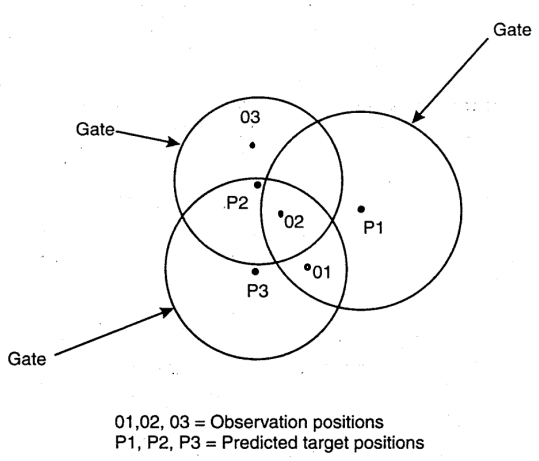


Figure 2.1: An instance of conflicting scenario for GNN [4]

There are a few established tracking-by-detection MTT algorithms that are used in many challenging scenarios across industry. An overview of these state-of-the-art MTT algorithms is provided in [8] [29] [58]. The most popular among these algorithms are joint probabilistic data association (JPDA) filter, multiple hypothesis tracking (MHT), and random finite set (RFS) based multi-target filters.

The basic idea of the JPDA algorithm is that instead of finding the best match for each track, a weighted sum of all measurements is obtained for each track. The weights represent the probability of the measurement originated from a given track. In this method, since the probability of association is computed jointly for all measurements and all tracks, it is called *joint probabilistic data association*. JPDA works better than GNN in challenging scenarios as it avoids the hard assignment of a wrong measurement to a track which is possible in GNN, by considering a weighted sum of all valid measurements lying within the *gate* [5]. However, the computation complexity increases exponentially in JPDA with the number of targets and the number of measurements as it involves the calculation of joint association probabilities for each valid target-measurement pair. Several variations of JPDA algorithms have been developed as described in [4] [59] [26] to find approximations and make it tractable in actual implementation. [48] shows an implementation of

a tractable JPDA algorithm for visual pedestrian tracking application using the m-best solutions of an integer linear programming, wherein JPDA assignment scores are calculated by formulating a series of *assignment* problems with linear objective functions and integer constraints. Moreover, the original JPDA algorithm did not have an explicit way of handling the varying number of objects [8] [4] and is applied only when the number of objects is known in prior. [38] provides a variation of JPDA, called joint integrated probabilistic data association (JIPDA), which includes an elegant way of handling the automatic track initiation and termination by incorporating the existence probabilities of individual tracks.

Both GNN and JPDA make an irreversible decision regarding the data association at each time step. In essence, these methods need to solve all ambiguities regarding the data association at every time step. This leads to a poor performance from these methods when there is a large number of targets moving close to each other and a lot of ambiguities exist around the association of measurement to the track, especially in a densely cluttered environment [4]. In order to tackle such scenarios, MHT was proposed [46].

MHT is a 'deferred' decision algorithm [4], wherein different hypotheses are considered for associating a measurement to a track in conflicting scenarios such as shown in Figure 2.1. These hypotheses are then propagated such that the subsequent measurement data can be used to resolve all the ambiguities present in the current time step. In this way, it avoids making irreversible assignment decisions at every time step, reducing the possibility of wrong association and hence enabling a more robust association method [58]. However, the major drawback of MHT is that the number of hypotheses grows exponentially as it considers all the possible association combinations. Thus, it is impossible to implement such an algorithm in real-time applications. Only an approximation of this algorithm can be implemented. [9] [8] [46] describe such implementations in detail. [9] showed an efficient real-time implementation of the original MHT algorithm by generating only k-best hypothesis using Murty's algorithm. An alternative method for implementing an MHT algorithm efficiently is the track-oriented multiple hypothesis tree (TO-MHT) [4]. In this method, hypotheses are not maintained from between the measurement updates as done in [46], rather tracks formed at each measurement update step are reformed into hypotheses, and the track that survives from pruning result into a prediction for the next iteration. To make the MHT algorithm tractable and ensure manageable computation load, various hypotheses and track management techniques like *pruning*, *capping* and *clustering* are utilized [4]. [8] suggests an 'N-scan-back' pruning method on the assumption that any ambiguity at time step k will be resolved by time $k + N$, thus, at any given time, the depth of a hypothesis tree [5] is limited to N and all other branches are pruned.

[2] proposed a sampling-based data association algorithm, called Markov Chain Monte Carlo (MCMC), for finding sub-optimal approximation of Bayesian filter. The basic idea in this approach is that instead of considering all hypotheses at all times, hypotheses space is sampled based on the location of the track in the previous measurement update step. In this way, it avoids the exponential combinatorial problem encountered in MHT, resulting in a lesser computational load. This method is also capable of handling an unknown and varying number of targets. An extension of MCMC based MTT algorithm with particle filter is provided in [19]. The benefit of using PF is that it can handle both linear and non-linear motion models. However, the results showed in [19] suggest that this method sometimes loses the correct track, when samples are generated in wrong spaces, especially with interacting targets.

In the last decade, the random finite set (RFS) based MTT approach has received significant interest [58]. RFS is a set of variables, in which the cardinality (number of elements in the set) is also treated as a random variable. In this approach, tracks are represented as a random set, with its cardinality representing the number of targets while the set elements represent the target's state. In this way, it provides a mathematical framework for capturing the uncertainty involving the number of targets and their states [32]. The multi-target states and measurements are represented as random sets and the tracking problem is solved by calculating the first moment of the joint distribution, called probability hypothesis density (PHD), recursively [63]. Probability hypothesis density (PHD) filter first developed in [33], is a recursive method based on *assumed Gaussian density* in which the posterior distribution is approximated as Poisson point process

(PPP) [25]. At every time step, the intensity function of the PPP is set to the PHD of the distribution [37]. However, PPP is not suitable to capture the cardinality as the uncertainty increases with an increase in the Poisson point intensity function. This means that as the number of targets increases, the uncertainty regarding their estimated numbers also increases. To counter this problem, a generalized form of PHD, called cardinalized PHD (CPHD) was developed which jointly propagates the intensity function and the entire distribution of the number of targets [63]. The analytical implementation detail for PHD and CPHD is presented in [56]. These filters are computationally efficient but there is no closed-form solution exists and their tracking performance is poor in densely cluttered environment [58]. Recent developments in RFS based MTT algorithms are provided in [60] [14] [13]. Such algorithms inherently contain all the facets of a general MTT problem, for example, assumed probability densities, object birth model, object survival model, etc. in their formulation. Poisson multi-Bernoulli mixture (PMBM) filter is one such algorithm, wherein the Poisson point process is used to describe the birth of an object and target's states are represented as a multi-Bernoulli(s) [37]. The data association step in this filter is although equivalent to MHT [37]. The real-time application of the PMBM filter is not well-established yet. For example, the simulation study presented in [61] shows that the average run-time for the PMBM filter is as high as 4.5 seconds.

2.2 Deep learning based MTT

In the past few years, the advancement of neural networks and deep learning in the artificial intelligence domain has revolutionized the image-based target tracking paradigm [64]. These networks are capable of extracting relevant features from the raw sensor data. CNN and RNN are the two neural networks that are popular in object detection and tracking applications. In [64] a convolutional neural network (CNN) is used along with correlation filter [6] to create an affinity model that can track objects between the consecutive frames. The CNNs are great in extracting features from the raw sensor data but they cannot handle *temporal information* or sequence of data. Therefore, they require some filters to create an affinity model to associate features from one frame to another. On the other hand, RNNs are capable of handling sequential data, making them the obvious choice for end-to-end deep learning-based object tracking MTT algorithms. Moreover, in combination with long short-term memory mechanism [16], RNN based tracking algorithms are capable of estimating the future trajectories of the targets. This feature makes them desirable in the MTT application. Such implementation is provided in [34]. However, the performance of their algorithm was not at par with the conventional trackers. One major drawback of using deep learning-based trackers is that they require extensive training to learn various aspects of tracking such as appearance models, motion models etc. [41]. This limits their general applicability as these algorithms fail in unknown situations for which they are not trained.

2.3 Comparison of various MTT algorithms

Based on the literature study, an overview of popular MTT algorithms available in the literature is shown in Figure 2.2. Note that the name of the data association method and the estimation filter is used in naming the full MTT algorithm. As the focus of this thesis is on the data association part, different data association methodologies are compared with each other based on some criterion mentioned in literature [4] [58]. Figure 2.3 shows the comparison table of the data association methods. Deep learning based algorithms are not considered in this comparison as training such algorithms is beyond the scope of this project. It should be noted that there is no direct comparison provided in the literature for all these algorithms to the best knowledge of the author. Thus, the comparison table provided in this section is based on the author's judgement after reading the literature for different methods.

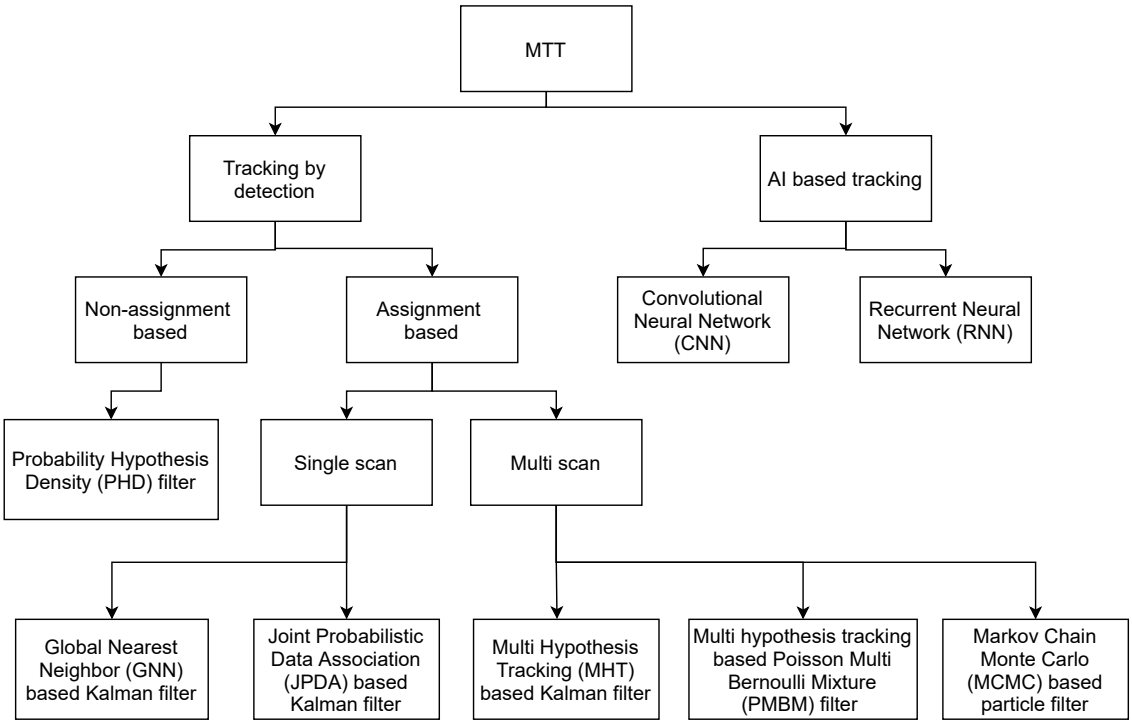


Figure 2.2: Classification of the MTT algorithms available in literature

Data association method / Criterion	GNN	JPDA	MHT	MCMC
Computation Cost	↑↑↑	↓	↓↓	↓↓
Closeness to exact solution	↓↓	↓	↑↑	↓
Ambiguity Handling	↓↓	↑	↑↑	↓
Occlusion Handling	↓↓	↓↓	↓	↓↓
Parameterization	↑↑	↑	↓	↑

Figure 2.3: Comparison of different established data association methods. ↑ / ↓ represents the relative strength/weakness of the algorithms

Key take-away points from the literature study:

- GNN is computationally cheapest among all algorithms, however, its performance degrades in challenging scenarios.
- JPDA outperforms GNN in ambiguous situations but overall performance is not significantly better as it also fails in a densely cluttered environment with interacting targets.
- Both JPDA and GNN take an irreversible decision at each time step regarding the association of measurement to a track, which always leaves a finite chance of the wrong assignment.
- MHT is the most complete method of capturing all the possibilities of data association. However, due to combinatorial explosion, it is computationally expensive. Thus, in actual implementation, several approximation techniques are required to keep the algorithm computationally tractable in real-time.
- PHD is an RFS based sub-optimal MTT algorithm. Its performance, however, degrades when there is a high number of targets present.
- MHT based PMBM shows a good promise as it contains all the facets of the MTT problem in its formulation. However, there are many parameters in the problem formulations related to the target birth model and survival model that are scenario dependent. Its real-time applicability is also not yet proven.
- Sampling based MCMC technique is also an approximation of the optimal Bayesian filter, however, it has the possibility of converging the tracks to wrong measurement spaces and thus has limited applicability in automated driving applications.

Based on these key takeaways, MHT is selected for further investigation and implementation. Performance of the GNN algorithm (currently used in TNO) will be considered as a baseline to find if MHT performs better. To have a fair comparison between the two data association methodologies, the same Kalman filter will be used for both the data association methods.

2.4 Performance metrics for evaluation of MTT algorithms

In [29], the performance of various MTT algorithms has been evaluated on public data set using several metrics. The description of these metrics is shown in Figure 2.4. It should be noted that the favorable trend is shown for each metric by arrow signs (↑ or ↓). For example, a higher value(↑) of the MOTA metric suggests better tracking performance while a lower value(↓) of the IDS metric suggests better performance. Even though there are several metrics available, in most of the literature study, only Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) are considered [3] [34]. Many public data set based tracking challenges like PETS, CLEAR, etc. [44] use MOTA and MOTP metrics for evaluation of tracking performance. These metrics are intuitive and widely accepted. MOTA accounts for all the errors made by the tracker like false positives, miss-detections, and mismatches over all frames without taking errors between true and estimated positions of the objects into account. On the other hand, MOTP accounts for the errors between the true and estimated states of the objects, without considering the track consistency, mismatching, etc. Thus, MOTP and MOTA both are used together to evaluate the overall performance of a tracker. It should be noted that MOTP and MOTA are calculated as an average number over all frames. They do not provide a real-time metric that can evaluate the performance of the tracker frame by frame.

Real-time tracking performance can be evaluated with OSPA [52] and GOSPA [45] metrics. These metrics are based on *miss-distance* error between two sets, the first set being the true object's state and the second being the estimated object's state. OSPA has been used traditionally for MTT algorithm evaluation. It computes "per-target error" at each time step, thus applicable in those situations also when the number of objects in the two sets is not equal [43]. However,

this feature restricts it to include the cardinality error into the overall performance metric. To include the cardinality error into account, a more generalized version of OSPA, called GOSPA, was presented in [45]. Thus, GOSPA can be seen as a complete metric to evaluate the real-time performance of an MTT algorithm.

Metric	Description	Note
Recall	Ratio of correctly matched detections to ground-truth detections	↑
Precision	Ratio of correctly matched detections to total result detections	↑
FAF/FPPI	Number of false alarms per frame averaged over a sequence	↓
MODA	Combines missed detections and FAF	↑
MODP	Average overlap between true positives and ground truth	↑
MOTA	Combines false negatives, false positives and mismatch rate	↑
IDS	Number of times that a tracked trajectory changes its matched ground-truth identity (or vice versa)	↓
MOTP	Overlap between the estimated positions and the ground truth averaged over the matches	↑
TDE	Distance between the ground-truth annotation and the tracking result	↓
OSPA	Cardinality error and spatial distance between ground truth and the tracking results	↓
MT	Percentage of ground-truth trajectories which are covered by the tracker output for more than 80% of their length	↑
ML	Percentage of ground-truth trajectories which are covered by the tracker output for less than 20% of their length	↓
PT	1.0 - MT - ML	-
FM	Number of times that a ground-truth trajectory is interrupted in the tracking result	↓
RS	Ratio of tracks which are correctly recovered from short occlusion	↑
RL	Ratio of tracks which are correctly recovered from long occlusion	↑

Figure 2.4: Overview of different MTT performance evaluation metrics [29]

Chapter 3

Multi target tracking problem setup

The goal of this chapter is to introduce the multi-target tracking (MTT) problem in a mathematical framework and build the base for the simulation setup used in this thesis. A general workflow of a statistical MTT algorithm is explained in section 3.1. A baseline simulation setup used to define the MTT problem is provided in section 3.2. The measurement setup used to generate measurement signals is also discussed in this section. Finally, the Kalman filter which will be coupled with different data association methods as state estimation filter is presented in section 3.3. The basic probability theory that forms the basis of the statistical framework is provided in appendix A for reference.

3.1 Statistical framework of a general MTT algorithm

As established in section 1.1, there are several uncertainties involved with the measurements. To accommodate those uncertainties, the MTT problem is formulated in a statistical framework. A general workflow of a statistical MTT algorithm is presented in Figure 3.1. The target's state at the current time step is used to predict its state in the next time step using a *motion model*. Given the target's predicted state, *measurement likelihood* is calculated to predict the output from the sensor using a *sensor model*. In practical implementations, the measurement likelihood is used to define a threshold region in FOV, called *gate*, to eliminate the unlikely association by considering the measurements only within the gate for data association [4]. Then, a data association algorithm is used to associate the measurement either with a previous track or a new track or simply treated as a false alarm. The sequence of measurement-to-track association is generally termed as *association hypothesis*. Based on these association hypotheses, tracks are evaluated under a probabilistic framework. This evaluation may result in the initiation of a new track, or continuation of a previously seen track, or deletion of a track. The task of initiation, confirmation, and deletion of the track is referred to as *track maintenance* [5]. Using the respectively associated measurement, the target's state is updated and the process is repeated recursively.

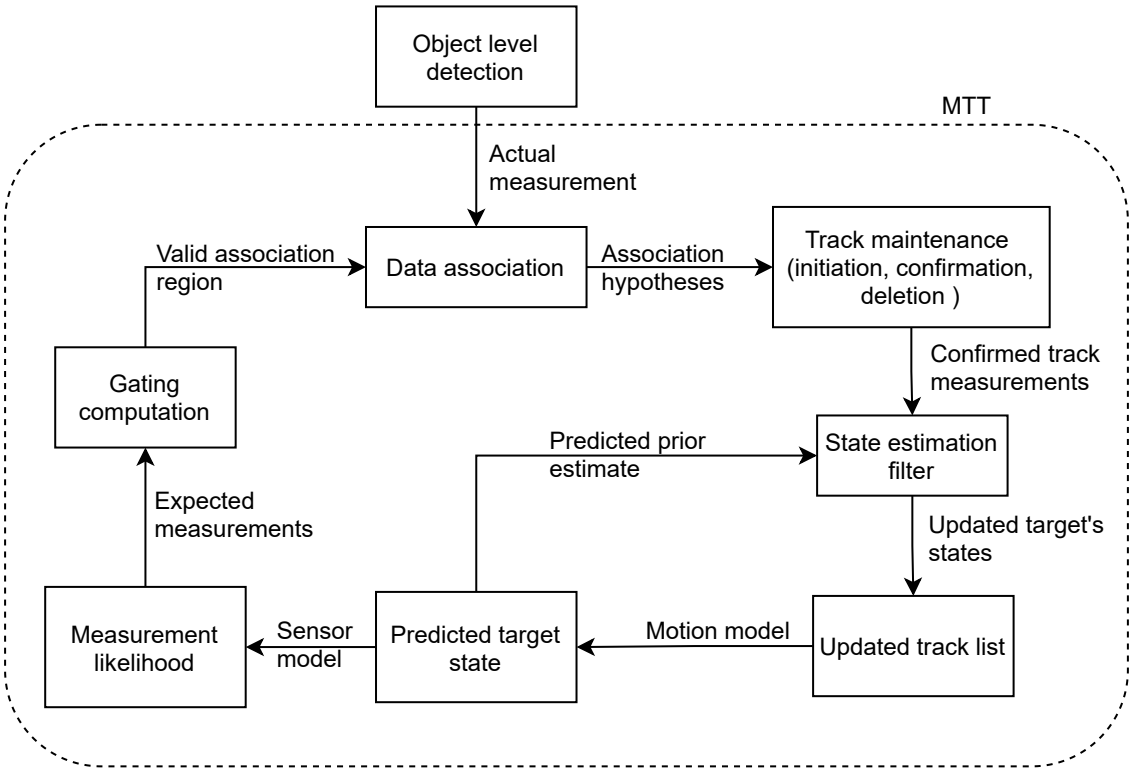


Figure 3.1: General workflow of a statistical MTT algorithm

3.2 Problem setup

Throughout this thesis, a 2D multi-target tracking problem is considered, wherein the goal is to recursively estimate at each time step k, the number of targets and their states represented by the set $X^k = \{x_1^k, x_2^k, ..., x_{n_x}^k\}$ using the measurement set denoted by $Z^k = \{z_1^k, z_2^k, ..., z_{n_z}^k\}$. Due to the possibility of missing a target or having false positive detections, the number of targets, n_x, and the number of measurements, n_z, might be different. The state vector of a target contains its relative position, velocity and acceleration with respect to the center of the ego vehicle, i.e., $x = \begin{bmatrix} p_{x,rel} & p_{y,rel} & v_{x,rel} & v_{y,rel} & a_{x,rel} & a_{y,rel} \end{bmatrix}^T$. Thus, the discrete-time counterpart of the state vector is given by:

$$x_k = \begin{bmatrix} p_{x,rel}^k & p_{y,rel}^k & v_{x,rel}^k & v_{y,rel}^k & a_{x,rel}^k & a_{y,rel}^k \end{bmatrix}^T \tag{3.1}$$

The coordinate system for the simulations is shown in Figure 3.2. The ego vehicle is represented as the origin in this coordinate frame. The states of the targets are measured and estimated relative to this origin as shown in Figure 3.2. There are some assumptions made within the scope of this project to simplify the problem so that a tractable algorithm can be developed.

- Targets are assumed to be the point objects. Thus, their kinematic properties like position, velocity, and acceleration are relevant and therefore, these kinematic properties are treated as target states in this study. Targets size, dimension, and yaw are not relevant with point object assumption.
- No object classification is used in this study.
- One target can produce at most one detection per time step, per sensor. Thus, at most, one measurement per time step, per sensor, can be associated with one target.

- Trajectories of different targets are independent of each other.
- Motion of targets in x and y direction is also independent.
- States are assumed to be normally distributed. Thus, the mean and covariance values of the states are the quantity of interest for state estimation

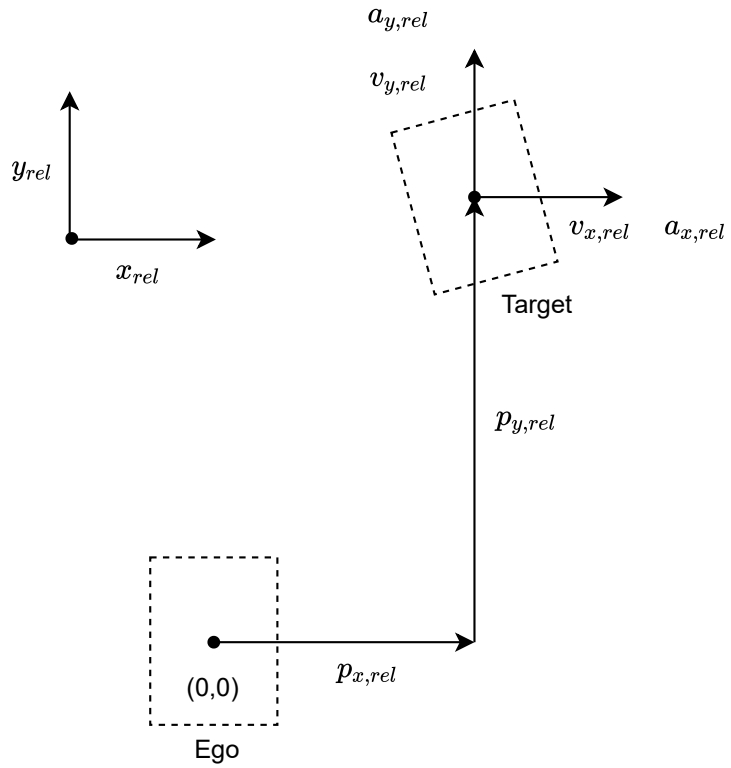


Figure 3.2: Co-ordinate system for the simulations

3.2.1 Measurement setup

A multiple sensor framework is considered in this study to ensure its general applicability. To this end, two measurement sets are generated, simulating camera and radar detections. The camera measures the relative positions, while radar measures both the relative positions and velocities in the x and y directions. The measured states and their assumed normal standard deviation for camera and radar are listed in table 3.1. It should be noted that in practice, the measurement attributes from a radar may not be exactly the positions and velocities, rather range (and range rates) and angles (azimuth and elevation) are available which can result in different standard deviations in x and y directions. However, for simplicity, they are assumed to be the same in both directions for both the sensors. Furthermore, the frequency at which the measurement data is available is different for the two sensors, which typically is the case in practice.

The general measurement model [47] for a sensor is given by:

$$z_k = \mathbf{H}x_k + \mathbf{w}_{meas,k} \tag{3.2}$$

where $\mathbf{H}$ is the measurement matrix and $\mathbf{w}_{meas,k}$ is the measurement noise sequence. The measurement matrix projects the state vector in the measurement space. For a linear measurement

Table 3.1: Simulated sensor parameters [20] [55]

Sensors	States Measured	Standard Deviation	Frequency
Radar	$p_{x,rel}$, $p_{y,rel}$	$\sigma_{p_{x,r}} = \sigma_{p_{x,r}} = 0.55\ m$	$20\ Hz$
	$v_{x,rel}$, $v_{y,rel}$	$\sigma_{v_{x,r}} = \sigma_{v_{x,r}} = 0.28\ m/s$	
Camera	$p_{x,rel}$, $p_{y,rel}$	$\sigma_{p_{x,c}} = \sigma_{p_{x,c}} = 1\ m$	$9.09\ Hz$

model, it is a constant matrix. Thus for the simulated camera and radar measurements, the measurement matrices are given as:

$$\mathbf{H}_{camera} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

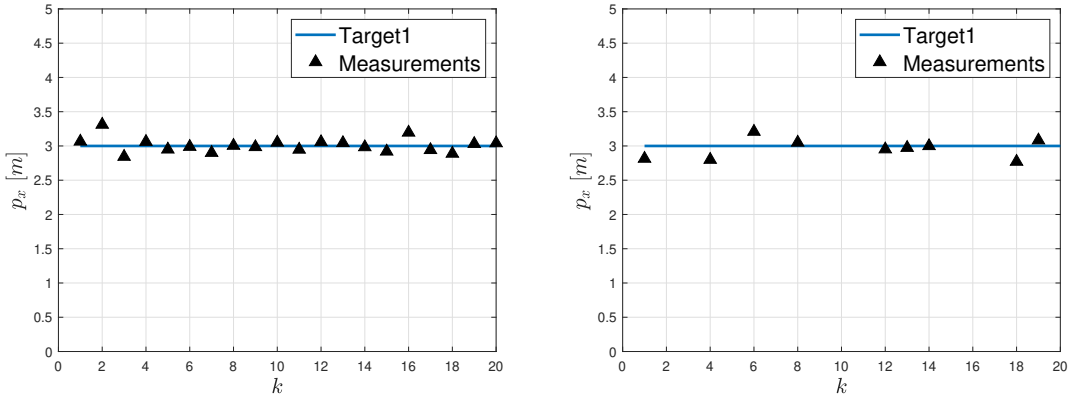
$$\mathbf{H}_{radar} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.4}$$

Furthermore, the measurement noise sequence is assumed to be zero-mean Gaussian white noise with known standard deviation as listed in the table 3.1. The radar measurement standard deviations are approximated from the study presented in [20], while the camera measurement standard deviation is approximated from the study presented in [55].

There is always a finite possibility of a target getting missed by the sensor. To that end, a parameter called, probability of detection (P_D) is used to simulate the possibility of (miss)detecting a true target.

Probability of detection P_D

For every target at each time step, a random number between 0 and 1 is generated, then a check is used such that if the random number is greater than the defined P_D, the target measurement is deleted to mimic the possibility of missing that target in that time step. Figure 3.3 shows the measurements obtained for the x-position of a target with time. The measurement values are not exactly equal to the true positions as random noise is added. Furthermore, for the lower probability of detection ($P_D = 0.5$), it can be seen that the target is not detected randomly for approximately half of the time steps which is expected.



(a) $P_D = 1$ (b) $P_D = 0.5$

Figure 3.3: Effect of P_D on measurement generation

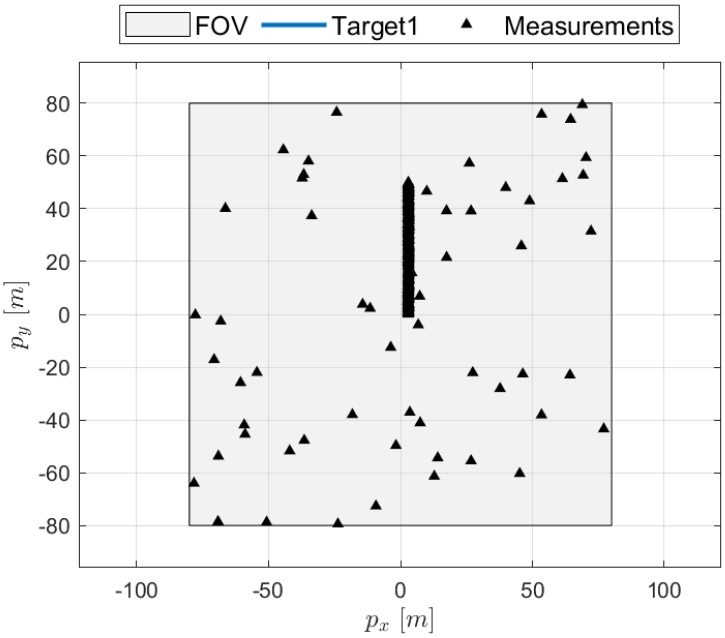
Clutter measurement C_k

In real scenarios, there is also a possibility of false-positive or clutter detections. In multi-target tracking applications, it is common to assume the clutter detections (C_k) to be uniformly distributed within the FOV of the sensor and the number of clutter detections (ϕ) is assumed to be Poisson distributed [9] given by equation (3.5). The intensity function variable λ_c is used to represent the expected number of clutter measurements expected at each time step. Thus, this parameter can be tuned to increase or decrease the clutter detections per time step.

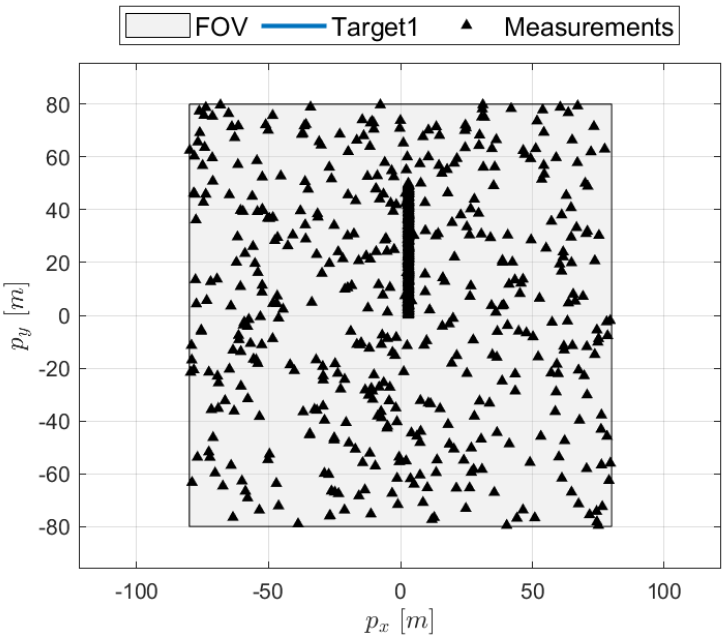
$$P[\phi = i] = \frac{\lambda_c^i \exp(-\lambda_c)}{i!} \text{ for } i = 0, 1, 2, ... \tag{3.5}$$

The generated clutter measurements are then appended to the measurement matrix for that time step (Z_k) to give a complete measurement matrix consisting of measurements generated from the true targets if any, and the clutter, which essentially is the input to the MTT algorithm. It should be noted that this method can generate any clutter measurement values within the FOV of the sensor which is not completely true in actual scenarios as sensors have limited resolution and measurements are reported within that resolution. Furthermore, the uniform distribution assumption of the clutter measurements may not hold in reality as the false positives may be generated from an irrelevant object or erroneous signal processing of the sensor, which typically is not uniformly distributed [37]. However, in this thesis, the sensor resolution is assumed to be infinite and clutter measurements to be uniformly distributed, for simplicity.

Figure 3.4 shows the effect of intensity function variable λ_c on the number of clutter measurements obtained when the measurements are generated for the same number (=500) of time steps for both the cases. The clutter measurements are generated along with the measurements from the true target. It can also be seen that the clutter measurements are uniformly randomly distributed within the FOV of the sensor.



(a) $\lambda_c = 0.1$



(b) $\lambda_c = 1$

Figure 3.4: Clutter detections generated in 500 time steps with different Poisson intensity function variable λ_c. Number of clutter detections are higher with higher λ_c. Furthermore, the distribution of these clutter detections is uniform within FOV of the sensor

3.3 Kalman filter

The Kalman filter is a closed-form solutions of a Bayesian recursive filter that can be used to estimate the target state recursively given that the motion and measurement models are linear and Gaussian [47] [50]. A detailed flowchart of the working of a discrete Kalman filter as shown in Figure 3.5.

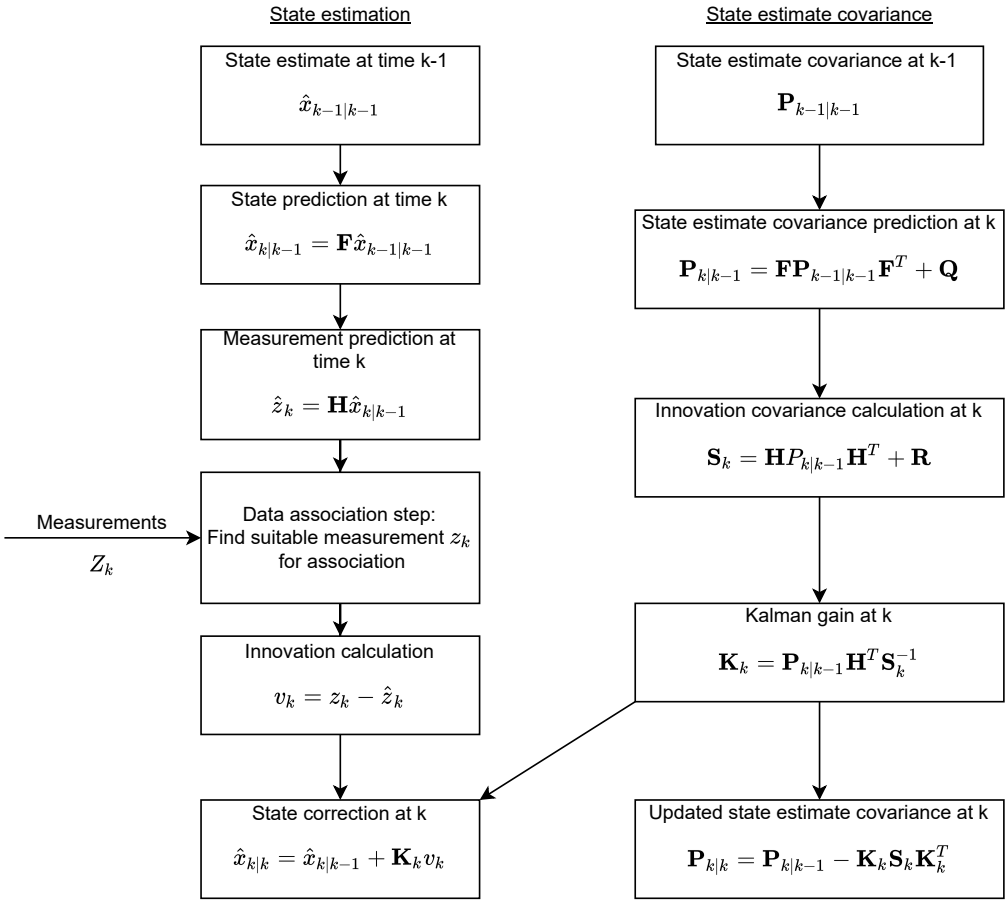


Figure 3.5: Detailed flowchart of Kalman filter recursive steps [20]

3.3.1 Prediction step

The prediction step is used to predict the states of the target at time k, given its state at time $k-1$. The measurement is also predicted in this step.

Target state prediction

The motion of a target in linear continuous state space representation [47] is given by:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{D}\mathbf{w}_{process}(t) \tag{3.6}$$

where x is the state of the target, $\mathbf{A}$ is the state transition matrix, $\mathbf{D}$ is the noise model and $\mathbf{w}_{process}$ is the zero-mean white Gaussian process noise.

Although the target motion is more accurately modeled in continuous time, a discrete-time motion model is used generally to systematically deal with the discrete nature of availability of the sensor measurements [28]. However, the target motion should not depend on how and when

the measurements are taken, thus, motion prediction should be independent of the availability of the measurement. Thus, a mixed-time model is used in this thesis, wherein the prediction step is carried out at a higher frequency while the measurement update steps are carried out at a lower frequency, which is dependent on the frequency at which measurement data is available. Figure 3.6 shows the schematic representation of such a setup. The frequencies of the Kalman prediction, camera update, and radar update are listed in the table 3.2. The prediction step of the Kalman Filter (KF) is performed at 100 Hz. In practice, a higher frequency prediction helps in minimizing the linearization errors in the state prediction when non-linear motion models are considered. The update steps of the filter are performed once the measurement data is available from the sensor. It is assumed that sensors are not faulty and they are providing the measurements at a constant frequency. Furthermore, in the current implementation, it is assumed that the Kalman update step including the data association step is performed instantaneously. However, the data association step involves solving optimal assignment problem, maintaining and evaluating many hypotheses related to association uncertainties (explained in MHT implementation in chapter 4), which is computationally expensive and thus, can not be assumed to be instantaneous in practical implementations.

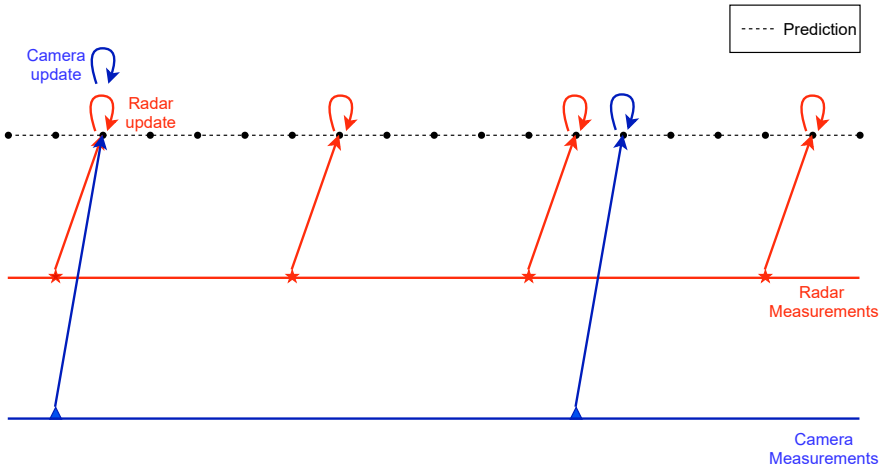


Figure 3.6: Schematic representation of prediction and update step of the Kalman filter

Table 3.2: Kalman filter frequencies

Parameter	Value
KF prediction	100 Hz
Radar update	20 Hz
Camera update	9.09 Hz

The discrete time state space representation [47] of the target motion is described as

$$x_{k|k-1} = \mathbf{F}x_{k-1|k-1} + \Gamma\mathbf{w}_{process,k-1} \tag{3.7}$$

where k represents the discrete time steps, $\mathbf{F}$ is the discrete time state transition matrix and $\Gamma\mathbf{w}_{process,k-1}$ is the discrete time process noise matrix. These matrices are derived from their continuous time counterparts and are provided in [20]

$$\mathbf{F} = e^{\mathbf{A}T} = \sum_{i=0}^{\infty} \frac{(\mathbf{A}T)^i}{i!} \tag{3.8}$$

$$\Gamma\mathbf{w}_{process,k-1} = \int_{t_0}^{t} e^{\mathbf{A}(T-\tau)}\mathbf{D}\mathbf{w}_{process}(\tau)d\tau \tag{3.9}$$

where T is the time difference between the two consecutive time steps. It should also be noted that $\mathbf{w}_{process}(t) \neq \mathbf{w}_{process,k-1}$ as $\mathbf{w}_{process,k-1}$ is the random process noise sequence, the discrete counterpart of the zero-mean Gaussian process noise.

Using the expected value operator on equation (3.7), one can obtain the estimated predicted state

$$E[x_{k|k-1}] = E[\mathbf{F}x_{k-1|k-1}] + E[\Gamma\mathbf{w}_{process,k-1}] \tag{3.10}$$

Since process noise is assumed to be zero-mean Gaussian noise, the second term of the above equation would be 0. Under this assumption, the predicted estimate for the target state is given by:

$$\hat{x}_{k|k-1} = \mathbf{F}\hat{x}_{k-1|k-1} \tag{3.11}$$

Measurement prediction

As shown in Figure 3.5, the predicted target state is used to make a measurement prediction. The measurement equation in discrete time is given by [20]:

$$z_k = \mathbf{H}x_{k|k-1} + \mathbf{w}_{meas,k} \tag{3.12}$$

where $\mathbf{H}$ is the linear constant sensor model matrix and $\mathbf{w}_{meas,k}$ is the zero-mean white noise Gaussian measurement noise sequence, representing the uncertainty involved with the sensor model.

The predicted measurement can be calculated by using the expected operator on equation (3.12)

$$E[z_k] = E[\mathbf{H}x_{k|k-1}] + E[\mathbf{w}_{meas,k}] \tag{3.13}$$

which under the assumption of $\mathbf{w}_{meas,k}$ being zero-mean Gaussian noise would result into the expected value of predicted measurement

$$\hat{z}_k = \mathbf{H}\hat{x}_{k|k-1} \tag{3.14}$$

The dimension of the camera and radar measurement predictions will be different due to their respective measurement matrices given in equation (3.3) and (3.4) respectively.

State estimate covariance

The covariances measure the uncertainty related to the expected value [47]. Thus, in addition to the expected or mean values, the covariances related to the state estimation need to be calculated. By definition, covariance is based on the error between the true and the estimated state [47]. Thus, the covariance of the predicted state in a recursive form is given as

$$\begin{aligned}\mathbf{P}_{k|k-1} &= E[(x_{k|k-1} - \hat{x}_{k|k-1})(x_{k|k-1} - \hat{x}_{k|k-1})^T] \\ &= \mathbf{F}E[(x_{k-1|k-1} - \hat{x}_{k-1|k-1})(x_{k-1|k-1} - \hat{x}_{k-1|k-1})^T]\mathbf{F}^T + E[(\Gamma\mathbf{w}_{process,k-1})(\Gamma\mathbf{w}_{process,k-1})^T] \\ &= \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}\end{aligned} \tag{3.15}$$

where $\mathbf{Q}$ is the additive process noise matrix. This matrix is calculated based on the motion model as discussed in section 3.3.3.

3.3.2 Update step

Once the predicted target state and the related covariance are calculated, they are corrected based on the measurements received. As shown in Figure 3.5, prior to the measurement update, the data association step is used to find a suitable measurement to be associated with the target. Different data association methods will be discussed in the next chapters. For now, we assume that a suitable measurement is selected for the measurement update.

First, the difference between the predicted and actual measurements, called innovation, is calculated.

$$v_k = z_k - \hat{z}_k \tag{3.16}$$

Again, the dimension of the innovation term will be different for the radar and the camera measurements. Along with the innovation, its covariance is needed to be calculated which is given by

$$\begin{aligned} \mathbf{S}_k &= E[(z_k - \hat{z}_k)(z_k - \hat{z}_k)^T] \\ &= \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \end{aligned} \tag{3.17}$$

where $\mathbf{R} = E[\mathbf{w}_{meas,k}\mathbf{w}_{meas,k}^T]$ is the measurement noise matrix which is assumed to be zero-mean white Gaussian noise with known variances. The measurement noise matrices for the camera and the radar measurements are given by:

$$\mathbf{R}_{camera} = \begin{bmatrix} \sigma^2_{p_{x,c}} & 0 \\ 0 & \sigma^2_{p_{y,c}} \end{bmatrix} \tag{3.18}$$

$$\mathbf{R}_{radar} = \begin{bmatrix} \sigma^2_{p_{x,r}} & 0 & 0 & 0 \\ 0 & \sigma^2_{p_{y,r}} & 0 & 0 \\ 0 & 0 & \sigma^2_{v_{x,r}} & 0 \\ 0 & 0 & 0 & \sigma^2_{v_{y,r}} \end{bmatrix} \tag{3.19}$$

where $\sigma_{p_{x,c}} = \sigma_{p_{y,c}} = 1\ m$, $\sigma_{p_{x,r}} = \sigma_{p_{y,r}} = 0.55\ m$ and $\sigma_{v_{x,r}} = \sigma_{v_{y,r}} = 0.28\ m/s$ as mentioned in table 3.1. Note that the covariance term of different measured states, i.e., the non-diagonal terms of measurement matrix, are kept 0 under the assumption that there is no correlation between the measured states.

The final step in the Kalman filter is to correct or update the predicted states and related covariances. This is done by calculating the Kalman gain. The Kalman gain determines the weight of the correction from the measurement. It depends on the ratio of the state prediction and measurement prediction error covariances. For example, if the state prediction error covariance is relatively higher than the measurement error covariance, the Kalman gain will be higher to correct the state prediction estimate with measurements. The complete derivation of Kalman gain is provided in [47]. The Kalman gain in recursive form is given by:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}\mathbf{S}_k^{-1} \tag{3.20}$$

Based on the Kalman gain and innovation, the state estimate and its covariance are updated as shown in Figure 3.5

$$\begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbf{K}_k v_k \\ \mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^T \end{cases} \tag{3.21}$$

3.3.3 Motion model

The purpose of the motion model is to provide prior information regarding the motion of the target in the Kalman filter. There are many motion models presented in [28] that can be used. However, it is decided to use the constant acceleration motion model in this study as it is one of the most common motion models found in literature [28] and is currently used in TNO implementation also. In this model, the jerk of the target is modeled as zero-mean white noise. As the target is assumed to be a point object, it can move in any direction without any correlation in its states in x and y direction. Thus, the constant acceleration motion model in continuous time state space form would be given by equation (3.6) with

$$x(t) = \begin{bmatrix} p_{x,rel} & p_{y,rel} & v_{x,rel} & v_{y,rel} & a_{x,rel} & a_{y,rel} \end{bmatrix}^T \tag{3.22}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.23}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.24}$$

and,

$$\mathbf{w}_{process} = \begin{bmatrix} w_x \\ w_y \end{bmatrix} \tag{3.25}$$

where w_x and w_y are the known disturbances (in jerk) in x and y direction respectively.

The discrete time state transition matrix $\mathbf{F}$ can be obtained from the equation (3.8)

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

Using equation (3.9), the process noise covariance matrix is also obtained as described in [20]

$$\mathbf{Q} = \sigma_q^2 \begin{bmatrix} \frac{T^5}{20} & 0 & \frac{T^4}{8} & 0 & \frac{T^3}{6} & 0 \\ 0 & \frac{T^5}{20} & 0 & \frac{T^4}{8} & 0 & \frac{T^3}{6} \\ \frac{T^4}{8} & 0 & \frac{T^3}{3} & 0 & \frac{T^2}{2} & 0 \\ 0 & \frac{T^4}{8} & 0 & \frac{T^3}{3} & 0 & \frac{T^2}{2} \\ \frac{T^3}{6} & 0 & \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^3}{6} & 0 & \frac{T^2}{2} & 0 & T \end{bmatrix} \tag{3.27}$$

where $\sigma_q^2 = \sigma_x^2 = \sigma_y^2$ is the known variance of the jerk in x and y direction. It should be noted that σ_x^2 and σ_y^2 can be different. However, under the assumption of no correlation between the states in x and y direction, it is suitable to assume $\sigma_x^2 = \sigma_y^2$ [20].

3.3.4 Kalman filter assumptions and their effects

There are several assumptions made while deriving the recursive form of equations for the Kalman filter as presented in Figure 3.5. These assumptions are listed in section 1.3 of [47]. [47] also deals with the potential performance issues one can encounter if the process and/or measurement noise variances are under/over-estimated. Thus, tuning of the filter becomes essential. A simple methodology suggested in [20] is followed in section 3.4 to tune the process noise variance to achieve the best possible performance from the Kalman filter with the measurement setup (see section 3.2.1) considered in this study.

3.4 Kalman filter tuning

To get the best tracking result from the MTT algorithm, its filter needs to be tuned. From the Kalman gain expressions, (3.17) and (3.20), it can be seen that the Kalman gain depends on the ratio of $\mathbf{R}$ and $\mathbf{Q}$. In practice, however, $\mathbf{R}$ is derived from the sensor's specification sheet and assumed to be known and fixed. With $\mathbf{R}$ fixed, an optimization study can be performed on $\mathbf{Q}$ to tune the Kalman filter. One such optimization study is provided in [20]. The circular motion of the target is chosen for Kalman filter tuning as it does not follow the assumed constant acceleration motion model and hence, can be used as a general tuning scenario. Furthermore, the circular motion is not biased regarding the motion in either of the x or y directions. In this scenario, a target completes a 50 m radius, 360 deg circle, with respect to the ego vehicle. Figure 3.7(a) shows the x-y position plot of the circle scenario, whereas Figure 3.7(b) shows the evolution of the true states of the target over time.

Now, following similar approach as in section 3.5.1 in [20], a simulation study is performed to find process (motion) noise variable σ_q. The idea is to find an optimum value for σ_q such that the sum of the ratio of RMS error to the maximum of absolute error for every state estimate is minimum. Thus, the cost function for the optimization is:

$$J = \sum_{i=1}^{6} \frac{\text{RMS}(e_i)}{\max(|e_i|)},\ i = \{p_x, p_y, v_x, v_y, a_x, a_y\} \tag{3.28}$$

Note that the steady state errors are considered here. The filter reaches the steady state when its estimation errors are within three standard deviations [47]. Different parameters for this simulation are listed in table 3.3. These parameters are chosen such that a realistic but challenging maneuver is captured and thus, a more general tuning of σ_q is achieved.

Table 3.3: Kalman filter frequencies

Parameter	Value	Description
r	50 m	radius of the circle
ω	0.1π rad/s	angular velocity of target
T	0.01 s	prediction step size

Using grid search, the optimum value of σ_q was obtained for which the cost is minimum. Figure 3.8 shows the result. The minimum cost is obtained for $\sigma_q = 1.2\ m/s^3$.

It is observed that the optimum value for the process noise variance depends on the scenario. A study is provided in appendix B wherein the optimum value for σ_q is obtained for different ω of the target. A lower value of optimal σ_q is obtained when the target is moving slow. However, the risk associated with choosing a lower value of σ_q, i.e. underestimating the process noise variance, is more as compared to overestimating the process noise variance. This is visible in the steep slope of the error cost function before the optimal σ_q point as compared to the slow rising slope after the optimal σ_q point as shown in Figure 3.8. In [47] also, boosting process noise is recommended to capture the uncertainty with regards to the assumed motion model. It can be concluded from this study that one can not guarantee the best performance of the Kalman filter in every scenario, especially when the motion of the target is not in accordance with the assumed motion model. However, the maneuver described in this section is considered challenging enough to capture the uncertainty of the underlying motion model. Thus, $\sigma_q = 1.2\ m/s^3$ is fixed for the rest of this thesis.

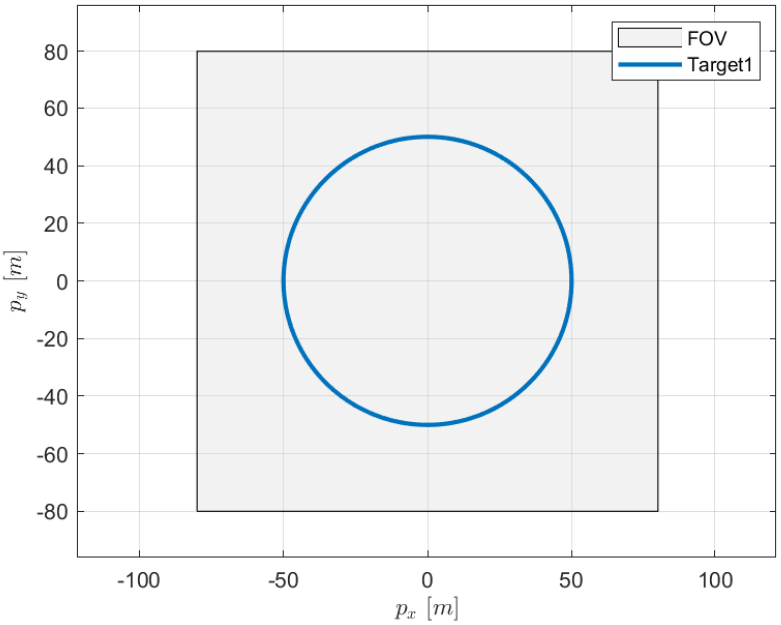
Tracking performance for this scenario with calculated optimal σ_q is shown in the Figure 3.9 and 3.10. It can be seen in Figure 3.9, that the Kalman filter is able to track the motion of the target. Figure 3.10 shows the estimation error for each of the motion state. The errors are higher in the initial period as the filter is in the initialization phase. After approximately 0.8 s, the filter reaches the steady state as the error values converge. The RMS and max absolute errors in steady state for all the individual motion states are presented in the table 3.4. The acceleration

estimation errors are understandably high as the assumed constant acceleration motion is not followed in this scenario.

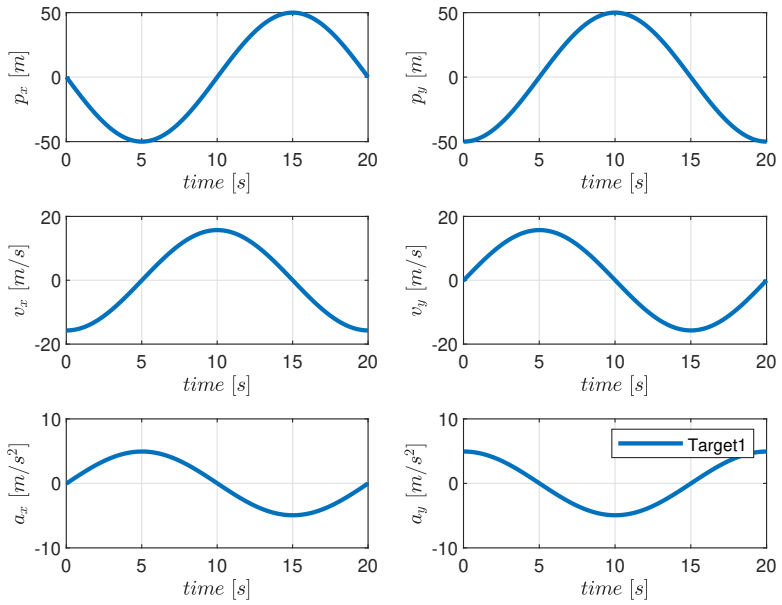
Table 3.4: Kalman filter frequencies

Error	e_{p_x}	e_{p_y}	e_{v_x}	e_{v_y}	e_{a_x}	e_{a_y}
RMS(e_i)	0.097 m	0.099 m	0.136 m/s	0.129 m/s	0.475 m/s^2	0.469 m/s^2
max($\|e_i\|$)	0.220 m	0.258 m	0.436 m/s	0.456 m/s	1.268 m/s^2	1.382 m/s^2

It should also be noted that missed detection, clutter detection, or any kind of ambiguity is not considered in this simulation. Thus, every measurement generated from the sensors described in section 3.2.1 is associated with the given single target. Although in practice, data association steps like gating (explained in chapter 4), affect the performance of the filter, it is not considered in this scenario such that the Kalman filter can be tuned without the influence of the data association step.



(a) x-y plot in relative co-ordinates



(b) Time plot of true states in relative co-ordinates

Figure 3.7: Circle scenario: Target1 starts at (0,-50) and completes a circle of 50 m radius in clockwise direction with a constant angular velocity of 0.1π rad/s

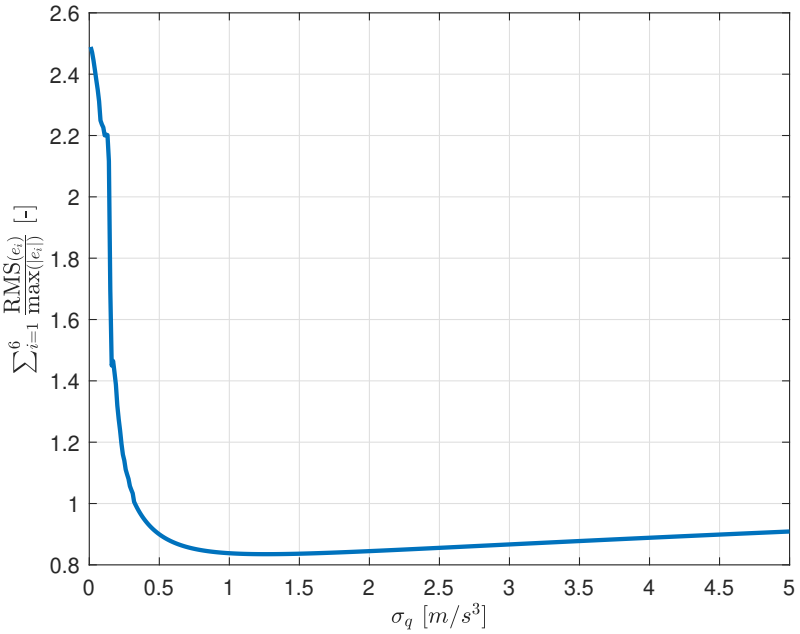


Figure 3.8: Filter tuning

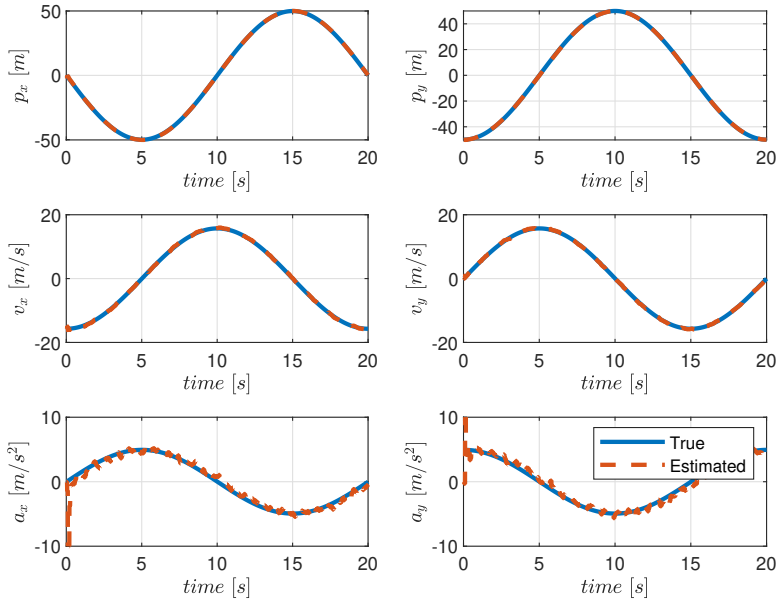


Figure 3.9: Tracking performance: True states vs estimated states

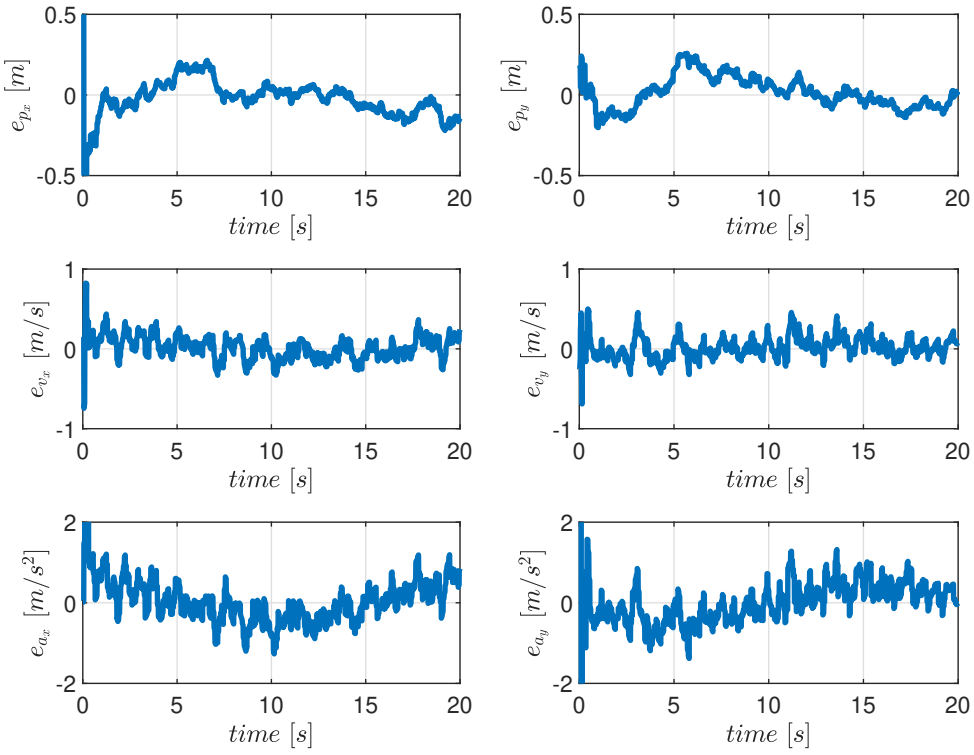


Figure 3.10: Tracking performance: Estimation errors

3.5 Summary

The base for the mathematical representation of the multi-target tracking problem was laid in this chapter. The measurement setup used to generate measurement was also discussed. Practical issues like missing the targets and false-positive measurement generation were also dealt with. A complete description of the Kalman filter and its tuning was also provided. Thus, all aspects of the multi-target tracking problem were fixed in this chapter, except for data association. This is discussed in the next chapter.

Chapter 4

Data association methods

As established earlier, the multi-target tracking (MTT) problem is essentially a filtering problem given one knows how to assign a measurement to a track for its state update. The filter part of the problem is already fixed in chapter 3. Now different data association methodologies are discussed in this chapter. A data association algorithm needs to solve various ambiguities regarding the unknown and varying number of targets, false detections, missed detections, etc in real-time. To solve these ambiguities, the data association step performs the following tasks:

1. Association hypothesis generation: An association hypothesis is the data association sequence that represents which measurement to be associated with which track. With the probability of missed detections and false detections and other uncertainties, there can be many different hypotheses regarding the measurement-to-track association.
2. Track management: This task includes initiating, confirming, and deleting tracks to accommodate an unknown and varying number of targets entering and leaving the FOV or surveillance area of the sensors.
3. Complexity reduction: This step is crucial for making the MTT algorithm tractable. As the number of tracks and measurements increases, the number of association hypotheses increases drastically. To keep computational cost in check, different techniques like gating, pruning, etc. are used to remove unlikely association hypotheses.

Thus, in addition to the measurement-to-track association, the data association step involves gating, hypotheses generation, track initiation, confirmation, and deletion functions, which are necessary to solve these ambiguities. These steps are explained in detail in each of the implementations covered in the next sections. The crucial tuning parameters are also identified and their selection and/or tuning procedure is also explained.

4.1 Global Nearest Neighbour (GNN)

In this section, the complete implementation details of the global nearest neighbour (GNN) data association is presented. The basic idea in GNN is to assign the measurements nearest to the predicted track. This nearness is decided based on the statistical distance called mahalanobis distance, between the actual measurement and the predicted measurement for a particular track [4]. Thus, for the predicted measurements of a set of tracks, GNN solves an optimal assignment problem to associate appropriate measurements to different tracks for their state update.

4.1.1 Data association as an optimal assignment problem

In GNN, only one hypothesis of measurement-to-track association is considered at every measurement update. This hypothesis is corresponding to the optimal 2D assignment problem based on a cost matrix.

Cost matrix

The measurement-to-track association in GNN is determined by solving an optimal assignment problem. This optimal assignment problem is formulated by forming a n_T x n_z dimensional 2D cost matrix **C**. Here, n_T denotes the number of tracks that we have after the prediction step and n_z denotes the number of measurements available at the given time step. The elements of this cost matrix is computed as:

$$C_{i,j} = d^2 + \log\left(|\mathbf{S}_k|\right) \tag{4.1}$$

where $i \in \{1, 2, .., n_z\}$ represents the measurement index, $j \in \{1, 2, .., n_T\}$ represents the predicted track index, d^2 represents the mahalanobis distance between the corresponding measurement prediction ($\hat{z}^k$) and the actual measurement (z^k) given by equation 4.2 and $\mathbf{S}_k$ represents the innovation covariance. These elements essentially represents the cost of associating the i^{th} predicted track to the j^{th} measurement.

$$d^2 = (z_i^k - \hat{z}_j^k)^T \mathbf{S}_k^{-1} (z_i^k - \hat{z}_j^k) \tag{4.2}$$

The idea behind using the mahalanobis distance as the cost of association is related to the likelihood of a measurement to be associated with a (predicted)track [4]. Thus, the farther a measurement is from the prediction, the less likely it is to be considered for the association, and hence its cost of association is higher. Moreover, the term $\log\left(|\mathbf{S}_k|\right)$ is added in the cost to penalize the tracks that have high uncertainty in their prediction from competing with other tracks having lower prediction uncertainty for a measurement association [4]. In practical implementation, all track-measurement pairs are not considered for the valid association. A course selection is used to determine which measurements should be even considered for association with a track. This step is called gating. The cost of association for any measurement outside the gating region of the measurement predicted (for a track) is then taken as ∞.

Gating

Gating is a hard boundary around the predicted measurements that is used to funnel valid measurements for data association consideration [4]. The most optimal gate used is the ellipsoidal gating [24], which is based on the chi-square χ^2 distribution test. A gating threshold γ_G is used to evaluate the 'fitness' of the actual measurements based on their mahalanobis distance from the predicted measurements.

$$d^2 \leq \gamma_G \tag{4.3}$$

This threshold value γ_G needs to be selected with caution as the too large gate can allow highly unlikely erroneous measurements to be considered for data association leading to higher computational load, whereas too small gate can result in rejection of the actual measurement generated from the true target. Generally, γ_G is calculated from the chi-square table based on the defined confidence intervals ($> 95\%$ is common) and appropriate degree of freedom [47] [24]. The degree of freedom is dependent on the dimension of the measurement vector [20] [24]. For example, the degree of freedom for the camera measurement would be 2 since the x and y positions measured by the camera are independent of each other. Thus, for a camera measurement and a required confidence interval of 99.9%, the gating threshold would be ≈ 13.816 [39]. The gating thresholds for camera and radar sensors used in this thesis are presented in table 4.1.

Table 4.1: Gating thresholds for camera and radar sensors

Sensor	γ_G
Camera	13.8
Radar	18.5

It should also be noted that putting a hard gate region around the predicted measurement violates the assumption of innovation being truly Gaussian. However, a sufficiently large gate

ensures the near-Gaussian distribution. Furthermore, gating helps in reducing computational loads significantly which is necessary to design a tractable MTT algorithm working in real-time [4] [12].

Solving the optimal assignment problem

An example is used in this section to explain how the data association problem is converted into a linear optimal assignment problem. Consider a situation shown in Figure 4.1. Here, measurements z_1 and z_2 lie in the gating regions for both the predicted tracks T_1 and T_2. Furthermore, the measurement z_3 lies in the gate for T_2 but not T_1.

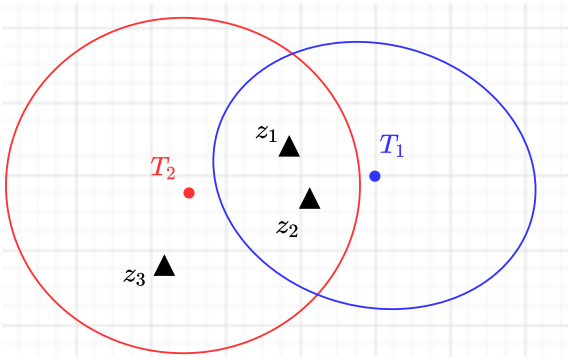


Figure 4.1: Typical ambiguous situation in data association: adapted from [4]

For the given situation, the cost matrix is given by:

$$\mathbf{C} = \begin{array}{c} \begin{matrix} T_1 & T_2 \end{matrix} \\ \begin{pmatrix} 1.1 & 1.2 \\ 0.9 & 1.3 \\ \infty & 1.1 \end{pmatrix} \end{array} \begin{matrix} z_1 \\ z_2 \\ z_3 \end{matrix} \tag{4.4}$$

Now the problem at hand is to find an assignment matrix $\mathbf{A}$ with elements

- $A_{i,j} = 1$, if measurement i is assigned to track j
- $A_{i,j} = 0$, otherwise

such that the global cost of association given by (4.5) is minimum

$$J_{global} = tr(\mathbf{A}^T\mathbf{C}) \tag{4.5}$$

To convert this problem into a linear optimal assignment problem, some constraints are required [4] which are compatible with the point object assumptions listed in section 3.2.

1. $A_{i,j} \in \{0, 1\}, \forall i, j$: Each measurement or track is either assigned or unassigned
2. $\sum_j A_{i,j} \leq 1, \forall i$: At most one measurement is assigned to a track
3. $\sum_i A_{i,j} \leq 1, \forall j$: A measurement can be assigned to at most one track.

Using these constraints, the data association problem is converted into a 2D linear optimal assignment problem to find an optimum assignment matrix $\mathbf{A}^*$

$$\begin{aligned} &\min_{\mathbf{A}^*} tr(\mathbf{A}^T\mathbf{C}) \\ \text{s.t. } &A_{i,j} \in \{0, 1\}, \forall i, j \\ &\sum_j A_{i,j} \leq 1, \forall i \\ &\sum_i A_{i,j} \leq 1, \forall j \end{aligned} \tag{4.6}$$

This optimal assignment problem is then solved using the Jonker-Volgenant (JV) algorithm [4] [12]. The `assignjv` function [17] in the 'sensor fusion and tracking' toolbox of MATLAB is used in this implementation to perform this task.

For the given cost matrix (4.4), the optimal assignment decision would be to assign z_2 to T_1 and z_3 to T_2. The assignment matrix then is given by

$$\mathbf{A}^* = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4.7}$$

resulting into the minimum global cost of $J^{min}_{global} = 2$.

It is common to use association sequence θ_k to represent the assignment hypothesis. For the given number of measurements n_z at time step k, θ_k is given as:

$$\theta^k = \begin{bmatrix} \theta_1 & \theta_2 & ... & \theta_{n_z} \end{bmatrix} \tag{4.8}$$

where $\theta_i \in \{0, 1, ..., n_T\}\ \forall i \in \{1, 2, ..., n_z\}$, represents the index of the track j that is associated with the measurement i. Here element 0 in θ_i represents the possibility of having the measurement being unassigned. In terms of θ^k, the constraints for the optimization problem are given as:

- Each measurement can either be associated or not associated, i.e., $\theta_i \in \{0, 1, ..., n_T\}\ \forall i \in \{1, ..., n_z\}$
- Any pair of tracks cannot share the same measurement, i.e.,
  $\forall i, i' \in \{0, 1, ..., n_T\}$, $i \neq i'$, if $\theta_i \neq 0$, $\theta_{i'} \neq 0 \Rightarrow \theta_i \neq \theta_{i'}$

For the cost matrix given in (4.4), the optimal assignment sequence would be $\theta^k = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$.

4.1.2 Track maintenance

In literature, there are two ways in which the task of track maintenance is performed [4]. One method is based on some ad hoc rules which are made based on experience. For example, a track gets confirmed if a measurement is associated with it, say M times out of the last N time steps, with $M \leq N$. Similarly, a track gets deleted if no measurement is assigned to it, say M times out of the last N time steps. This ad hoc rule of maintaining tracks is called the M/N rule. The other method uses a score to determine the likelihood of a track being a true target. The latter is used in this thesis as it avoids any ad hoc rule and the necessity of remembering the assignment history of a track (at least for past N time steps).

Track score function

First developed by [54], a probabilistic score expression, in recursive form, for the evaluation of track formation hypotheses is presented in [4]. This expression takes into account the various aspects of data association problem formulation, such as the probability of detection P_D, the probability density of false alarms β_{FA}, dimension of measurement M, mahalanobis distance d^2 of the actual measurement and the predicted measurement and uncertainty regarding prediction (innovation covariance $\mathbf{S}_k$). The recursive expression is given in equation (4.9).

$$L_k = L_{k-1} + \Delta L_k \tag{4.9}$$

where

$$\Delta L_k = \begin{cases} \log\left[1 - P_D\right] \text{ , when no measurement assigned to the track at } k \\ \log\left[\frac{P_D}{(2\pi)^{M/2}\beta_{FA}\sqrt{|\mathbf{S}_k|}}\right] - \frac{d^2}{2} \text{ , when measurement is assigned to the track at } k \end{cases}$$

The logarithm (log) used in these expressions does not have any physical meaning, rather it is used to avoid numerical problems that can occur when probability values are very small. That

is why these scores are generally referred to as log-likelihood ratio (LLR) . The expression for the change of score ΔL_k when no measurement is assigned to a track includes the probability of miss-detection P_{MD} (=$1 - P_D$). To that end, if P_D is very high, the score for a track reduces significantly if no measurement is associated with that track, suggesting that it is most likely that the target has left the FOV. Furthermore, the expression for ΔL_k, when a measurement is assigned to a track, evaluates the probability of the associated measurement being a detection from a true target against the possibility of it being a false alarm. In this expression, the first part has P_D in the denominator suggesting if P_D is high, it is most likely that the measurement is coming from a true object and vice-versa. The denominator of the first part contains β_{FA} representing the possibility of the measurement being a clutter or false alarm and innovation covariance S representing the uncertainty in the prediction for the given track. The second part of the expression is the mahalanobis distance. Thus, if the associated measurement is statistically far from the prediction, the association likelihood will be lower, and thereby, the increase in track score will be lower.

The initial track score is defined simply as the multiplication of probability of detection and ratio probability of new targets β_{NT} to the false alarms β_{FA}. [4]

$$L_1 = \log\left[\frac{P_D \beta_{NT}}{\beta_{FA}}\right] \tag{4.10}$$

The track scores denote the probability of having a true target in a track [4]. The relation between the probability of existence of a true target (P_T) in a track with track score L is given by [4]:

$$P_T = \frac{e^L}{1 + e^L} \tag{4.11}$$

Using the recursive track scores given in equation (4.9), the status of a track is determined. Some threshold values are defined to change the state of a track. Suppose T_c and T_d are the confirmation and deletion thresholds, then

$$L_k \geq T_c \text{ , confirm track}$$

$$T_d < L_k < T_c \text{ , tentative track}$$

$$L_k \leq T_d \text{ , delete track.}$$

These threshold values are tuned based on the scenario and measurement setup properties. For instance, if a large number of clutter detections are generated, then T_c should be high such that only the prominent tracks are confirmed. The selection of these threshold values is discussed in the next section.

4.1.3 Parameter selection

Some parameters have been introduced in section 4.1.2 which determines the performance of the GNN based MTT algorithm. Some of these parameters are scenario and measurement setup based like the probability of detection P_D, probability of false alarms β_{FT}, probability of new targets β_{NT}, and the dimension of the measurement vector M. These parameters need to be fixed based on some experiments on real-world driving data experience captured using the measurement setup. However, in this study, these parameters are fixed based on the values used in the tracking literature and the measurement setup described in section 3.2.1. The parameters are listed in table 4.2. The subscripts r and c denote camera and radar respectively.

Apart from these scene and measurement setup related parameters, there are some tuning parameters available in GNN formulation which can be tuned to improve the performance of the tracker.

Table 4.2: Scene and measurement setup related parameters [9]

Parameter	Value
P_D	0.999
β_{FA}	0.00002
β_{NT}	0.004
M_c	2
M_r	4

Confirmation threshold T_c

Confirmation threshold T_c is the threshold that is used to confirm a tentative track. This threshold should be chosen carefully as a very low value can lead to confirmation of even the tentative tracks generated from the clutter measurements. Moreover, it should also not have a higher value as it results in keeping a true target unconfirmed for a longer duration, which is also not desirable. Following the standard sequential probability ratio test (SPRT) formulation discussed in [4], the confirmation threshold can be given as:

$$T_c = \log\left[\frac{1-\beta}{\alpha}\right] + L_1 \tag{4.12}$$

where α is false track confirmation probability, β is the true track deletion probability and L_1 is the initial track score. Using the parameters given in table 4.2, L_1 is calculated as per equation (4.10) to obtain the value of ≈5.3. The α can be defined from the system's requirement. Let the measurement setup produces N_{FA} number of false alarms per second and the allowed number of false track confirmation per hour is N_{FC}. Then,

$$\alpha = \frac{N_{FC}}{3600 N_{FA}} \tag{4.13}$$

In this implementation, N_{FC} is arbitrarily taken as 1, i.e., at max, one false target is allowed to be confirmed per hour. The parameter N_{FA} can be calculated from the β_{FA} and the measurement settings. As discussed in section 3.2.1, the operating frequencies of radar and camera sensors are 20 Hz and 9.09 Hz respectively. Thus, a total number of 29 measurement scans available per second collectively from both sensors. Also, the tracking volume for both sensors is assumed to be a square of dimension 80x80 m^2. Thus, $N_{FA} = \beta_{FA} \cdot 29 \cdot (80 \cdot 80) = 3.71$. Using the values of N_{FC} and N_{FA} in equation (4.13),

$$\alpha = 0.00007234$$

Furthermore, β will have lesser effect on the final value T_c as denominator of equation (4.12) will be much bigger. As recommended in [4], its value is taken as $\beta = 0.1$. Finally, the value of T_c obtained using equation (4.12)

$$T_c = 14.7$$

For a track score of 14.7, the probability of having a true target in that track is approximately 1 as per equation (4.11). This also checks as a sanity check.

Deletion threshold for tentative track T_d

The deletion threshold for a tentative track can be given by following the standard SPRT formulation [4],

$$T_d = \log\left[\frac{\beta}{1-\alpha}\right] \tag{4.14}$$

which essentially weighs the possibility of deleting a true target to the possibility of keeping a false track alive. Using the obtained values of α and β, the value of deletion threshold for tentative track is

$$T_d = -2.3 \tag{4.15}$$

For a track score of -2.3, the probability of having a true target in that track as per equation (4.11) is less than 0.1. This again checks as a sanity check.

Deletion threshold for well-established track, T_{drop}

As per equation (4.9), the score of a track will increase monotonically if measurements are associated continuously to it. This means that for a well-established track with a very high track score, it will take longer to delete even if it has left the FOV of the sensors. Thus, a different approach is required to delete a well-established track. One of the most popular ways of choosing a deletion threshold for a well-established track is N_d consecutive miss [4], i.e., if a track is missed consecutively for N_d time steps, it will be deleted. The choice of this consecutive miss threshold N_d is not standard and depends on the application. To fix this threshold, motivation is taken from the overtaking scenario discussed in [15]. It is observed that an overtaking vehicle drives around 1 second parallel to the passing vehicle during an overtake thus occluding the overtaking vehicle behind the vehicle being overtaken. Such scenario is described in section5.1. Thus, in such a scenario, it should be ensured that the track of the occluded vehicle should not be deleted from the ego vehicle's tracker for at least 1 second. Furthermore, within 1 second, there would be around 30 measurement scans collectively for the sensor setup used in this thesis. Thus, it is decided to keep N_d as 30 for this scenario. However, it can be tuned based on the requirements.

It might be possible that a prominent track may never get deleted even if there is no true target present for that track. This can happen due to the clutter measurements occasionally being associated with the track preventing the consecutive miss counts to reach N_d. One possible solution that could be used to solve this problem is to delete a track based on its covariance. But there is no standard way of choosing a threshold on the covariance matrix. An elegant solution for this is suggested in [4]. One can delete a track based on the drop in its track score. If the probability of detection P_D is known, one can calculate by how much track score will drop for $N_d = 30$ consecutive misses. For the assumed values in this thesis, the track score drop threshold

$$T_{drop} = \log\left((1 - P_D)^{30 \cdot t_{occ}}\right) = \log\left((1 - 0.999)^{30 \cdot 1}\right) = -207.2 \tag{4.16}$$

where t_{occ} is the occlusion duration in second(s). A check can be used to see if the drop of the score for a track from its maximum score is more than T_{drop} as a deletion criterion. This formulation ensures that even the tracks that are associated occasionally with the clutter measurements are also deleted. In implementation, the maximum score of a track (L_{max}) is updated and maintained in memory and the following condition is used to delete a track.

$$L_k - L_{max} \leq T_{drop} \text{ , delete track.}$$

4.1.4 Complete algorithm

Merging the Kalman filter with the GNN data association functions results in a complete MTT algorithm capable of tracking unknown and varying numbers of targets. A pseudo-code of the complete algorithm is given in algorithm 4.1.1. For all the updated tracks of the previous time step $k-1$, first the states of the targets and their corresponding covariances are predicted for the current time step k. Using these predicted tracks and the measurements received at k, the optimal assignment problem is solved to determine which measurement is most likely to be associated with which target. Based on this association, the target's state and covariances are updated. For unassigned targets, the state and covariances are not updated and kept the same as that of the prediction but their track scores are updated. The unassigned targets may get deleted if their track score becomes less than the deletion threshold. Furthermore, the unassigned measurements spawn new tentative targets. These new targets are assigned with the initial track score and the state same as that of the measurement. The status of these tentative tracks may get confirmed in the next time steps if their scores become more than the confirmation threshold. The mean values of the states of the confirmed targets in the updated tracklist are essentially the output of the MTT algorithm as estimate at time k. This process is repeated recursively to keep track of targets.

Algorithm 4.1.1 GNN algorithm

Require: List of *tracks* at time step $k-1$
Prediction step:
for all *tracks* **do**
$\mu_k^- = A_{k-1}\mu_{k-1}$
$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1}$
end for
return *pred_tracks*
Update step:
receive measurements z_k
if isempty(*pred_track*) **then**
if isempty(z_k) **then**
$updated\_tracks = pred\_tracks$
else
Initialize new tracks
Add new tracks to the *updated_tracks*
end if
else
Form assignment cost matrix C
Solve assignment problem using JV algorithm
for all *assigned_tracks* **do**
Update mean μ_k and covariance P_k
Update track score T_S
if $T_S >= confirmation\_threshold$ **then**
Update track status to confirmed
end if
end for
for all *unassigned_tracks* **do**
Update track score T_S
if $T_S < deletion\_threshold$ **then**
delete track from the *updated_track* list
end if
end for
for all *unassigned_measurement* **do**
Initialize new tracks corresponding to measurements
Add the new tracks to the *updated_tracks* list
end for
return *updated_tracks*
end if
Estimate:
for all *updated_tracks* **do**
if $status == confirmed$ **then**
$estimated\_states = \mu_k$
end if
end for

4.2 Multi Hypotheses Tracking (MHT)

In this section, the theoretical and implementation details of the multi hypotheses tracking (MHT) data association method are presented. The multi hypothesis tracking (MHT) generates, maintains, and evaluates multiple hypotheses regarding the data association uncertainties. The outline of a complete MHT algorithm based on [9] is presented in section 4.2.1. Subsequent sections then discuss various steps involved in the MHT algorithm using some examples. The tuning parameters for the MHT algorithm are discussed in section 4.2.5. Finally, a parallel is drawn between the GNN and MHT algorithms using a toy example in section 4.3.

4.2.1 MHT outline

Figure 4.2 shows the outline of a hypothesis-oriented MHT algorithm based on the work of [9]. An iteration at time k starts with generating m-best hypotheses from each of the hypotheses present at $k-1$. These hypotheses consist of different sets of tracks that correspond to a possible state of the world. The generated hypotheses are then evaluated based on the LLR scores of the tracks they constitute. Based on these scores, the hypotheses are weighted against each other. These hypothesis weights are then used to determine the likelihood of the hypothesis to be true, which is useful to prune the unlikely hypothesis in the hypothesis management step. To keep the computation load predictable, a cap is also put on the maximum number of hypotheses that can be maintained at each time step. This is called capping. Once the generated hypotheses are pruned and capped, the tracks of the hypothesis with maximum weight are presented as the output to the user. The tracks in each of the survived hypotheses are then used to predict the target's states which in turn are used to predict the measurements for each hypothesis. New hypotheses are generated at the next time step from these predicted hypotheses and the process is iteratively continued. All the steps are explained in detail in the next sections using some toy examples.

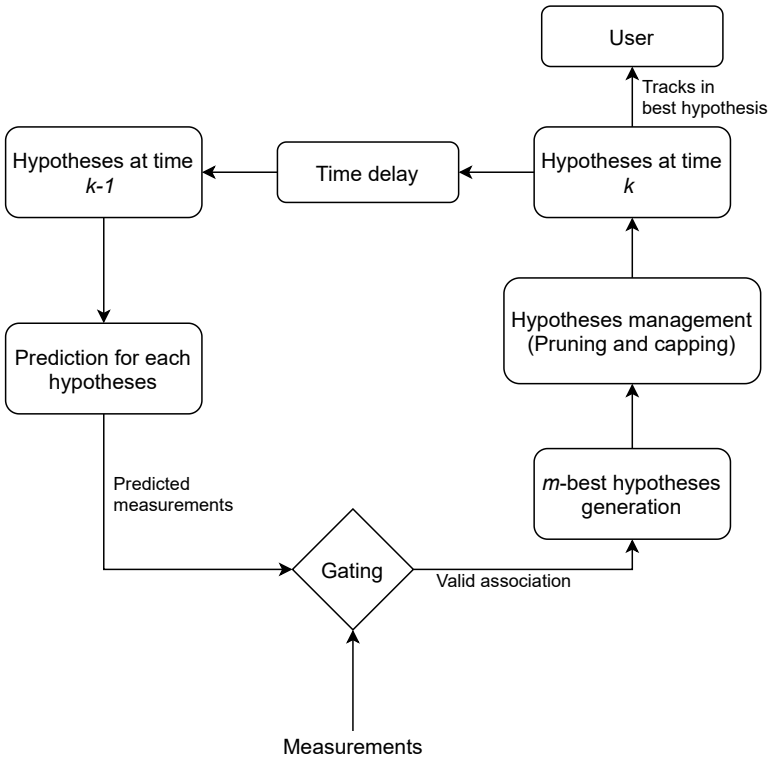


Figure 4.2: MHT algorithm outline: adapted from [9]

4.2.2 Hypothesis generation

A hypothesis represents a set of tracks, which is corresponding to a possible state of the tracked targets in the ego vehicle's environment. Thus, each hypothesis corresponds to a set of tracks resulting from a different measurement-to-track association sequence.

$$H_i^k = \{T_1, T_2, ..T_{n_T}\} \tag{4.17}$$

Consider that there exists only one track, T_1, at initial time step $k = 0$ as shown in Figure 4.3. At time step $k = 1$, a measurement, z_1^1, is received. Now there are three possibilities:

- z_1^1 originates from T_1
- z_1^1 is clutter or false alarm
- z_1^1 originates from a previously unseen or a new track, T_2

These three possibilities are then represented as three hypotheses in Figure 4.3. It should be noted that the track T_1 in hypotheses H_1^1 and H_2^1 are not the same. This is because T_1 in H_1^1 is not updated with the measurement z_1^1 whereas it is updated with z_1^1 in H_2^1. However, the track T_1 in hypotheses H_1^1 and H_3^1 are the same, as in both of these hypotheses, T_1 is not updated with z_1^1. This means that the same copy of a track may be maintained in two different hypotheses. Let another measurement, z_1^2, is received at $k = 2$. Based on the hypotheses at $k = 1$, multiple hypotheses can be generated at $k = 2$ as shown in Figure 4.3. It is easy to see that the growth of the hypotheses tree is exponential even with a single measurement received at each time step. Thus, enumerating all possible hypotheses at each time step is not tractable. To that end, only a few m-best hypotheses are generated [9]. To generate m-best hypotheses, first, a validation cost matrix is formed from the list of tracks present in the hypothesis in the previous iteration $k - 1$ (called the prior hypothesis) and the measurements available at k. This validation cost matrix then is used to translate the association problem into a 2D assignment problem from which m-best solutions are derived.

Ambiguity and validation cost matrix

The formulation of the validation cost matrix is explained using the same situation shown in Figure 4.1. For this situation, the prior hypothesis at $k - 1$ contains two tracks T_1 and T_2. At time k, first an ambiguity matrix (4.18) is formed. The tracks in the prior hypotheses constitute the columns and the measurements available at k constitute the rows in the ambiguity matrix. The two extra columns namely FA and NT represents the possibility of the measurement being a false alarm or generated from a new track respectively. The elements of the ambiguity matrix $\mathbf{\Omega}_{i,j}$ take value 0 if the measurement i lies beyond the valid gating region of track j. It can be verified from Figure 4.1 that measurement z_3 is outside the gating region of track T_1. Furthermore, all measurements can be either false alarms or generated from a new target. Therefore, the corresponding elements of the ambiguity matrix for FA and NT columns are taken as 1.

$$\mathbf{\Omega} = \begin{array}{c} \begin{matrix} FA & T_1 & T_2 & NT \end{matrix} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \end{array} \begin{matrix} \\ z_1 \\ z_2 \\ z_3 \end{matrix} \tag{4.18}$$

The ambiguity matrix can be converted into a validation cost matrix such that a linear optimal assignment (see section 4.1.1) can be formulated. To do so, the 0's in the ambiguity matrix are replaced by ∞ to indicate the impossibility of assigning measurement i with track j, whereas the 1's are replaced by the cost of associating measurement i with track j. This cost of association for the known track (prior hypothesis tracks) and measurement pair is already presented in (4.1). The costs of associating a measurement to a new track (C_{NT}) and the cost of associating a measurement

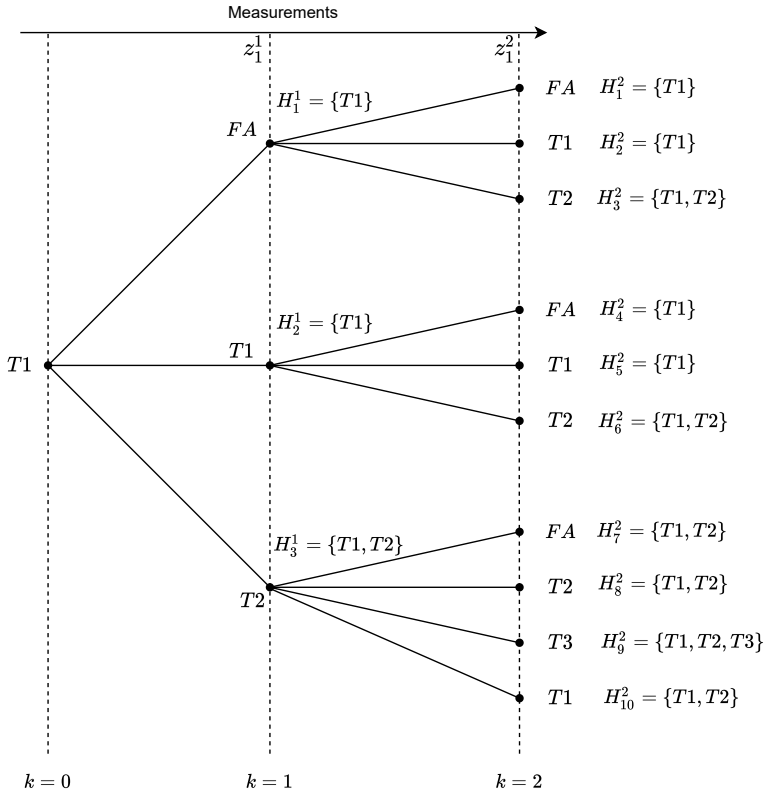


Figure 4.3: Combinatorial growth of a hypotheses tree with only one measurement available at each time step

to a false alarm (C_{FA}) are tunable parameters. For a typical MTT problem setup, these costs are one order higher than the known track and measurement pair association costs [37] [12]. In this implementation, these values are chosen based on the gating region fixed for the camera and the radar sensors respectively in table 4.1. The gate for camera measurement is 13.8 (see table 4.1). Thus, the cost of associating a camera measurement as a new track should be more than 13.8. Furthermore, the cost of treating a measurement as a false alarm should be higher than the cost of treating it as a new track to ensure that the hypothesis of an unassociated measurement being a new track is more likely than the hypothesis of being just a false alarm [37]. Based on this logic, C_{NT} and C_{FA} values are fixed and tabulated in table 4.3.

Table 4.3: C_{NT} and C_{FA} values for camera and radar sensors

Sensors	C_{NT}	C_{FA}
Camera	14	15
Radar	19	20

Now the cost matrix $\mathbf{C}$ corresponding to the ambiguity matrix given in (4.18) assuming a camera measurement update is given as:

$$\mathbf{C} = \begin{matrix} FA & T_1 & T_2 & NT \\ \end{matrix} \begin{pmatrix} 15 & 1.1 & 1.2 & 14 \\ 15 & 0.9 & 1.3 & 14 \\ 15 & \infty & 1.1 & 14 \end{pmatrix} \begin{matrix} z_1 \\ z_2 \\ z_3 \end{matrix} \tag{4.19}$$

To solve a linear optimal assignment problem, it is desired to have the constraints mentioned in equation (4.6). However, these constraints can not be satisfied for the cost matrix of the form shown in equation (4.19), since there is a valid possibility of having either all the measurements originated from the new tracks or all measurements being false alarms. Thus, the structure of the cost matrix is reformed as shown in equation (4.20)

$$\mathbf{C} = \begin{array}{c} \begin{array}{cccccccc} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) \\ FA_1 & FA_2 & FA_3 & T_1 & T_2 & NT_1 & NT_2 & NT_3 \end{array} \\ \begin{pmatrix} 15 & \infty & \infty & 1.1 & 1.2 & \mathbf{14} & \infty & \infty \\ \infty & 15 & \infty & \mathbf{0.9} & 1.3 & \infty & 14 & \infty \\ \infty & \infty & 15 & \infty & \mathbf{1.1} & \infty & \infty & 14 \end{pmatrix} \end{array} \begin{array}{c} z_1 \\ z_2 \\ z_3 \end{array} \tag{4.20}$$

In this structure, a diagonal sub-matrix of size n_zxn_z (non-diagonal elements are ∞) is introduced for both false alarms and the new tracks such that the constraints regarding the disjoint legal assignments are maintained while finding optimal assignment solution for this cost matrix [9]. This structure ensures that each measurement has all the possibility of association with a false alarm, a new track, or a previously known track. In this way, all uncertainties regarding the data association are included in the association problem. This was not the case with the cost matrix formulated in the GNN algorithm though. In GNN, only the previous tracks were considered for the track-to-measurement association. Possibility of new tracks and false alarms were treated in track maintenance.

Furthermore, the prior track index in this structure becomes $n_z + j$. For example, track index of prior track T_1 is now $3 + 1 = 4$ in this cost matrix structure. This cost matrix is then used to generate m-best association hypotheses using Murty's algorithm [4] as explained below.

m-best hypothesis

The crude way of implementing the MHT algorithm is to enumerate all possible hypotheses, calculate their probabilities and prune unlikely hypotheses to keep, the m-best hypotheses. However, enumerating all possible hypotheses is far from trivial as shown in Figure 4.3. Furthermore, if most of the hypotheses are low probable, there is no value in generating them in the first place. To that end, [9] used Murty's algorithm to generated only the m-best hypotheses. This ensures that only high quality hypotheses are generated. The explanation of Murty's algorithm based on [35] is presented next.

The pseudo-code for Murty's algorithm to find m-best solutions to a linear assignment problem, P_0, is shown in algorithm 4.2.1. The association problem is expressed as a bipartite graph having two disjoint sets T and Z representing the set of hypothesized tracks and the measurements respectively. Thus, the association problem is presented as the list of triples $< t, z, c >$. Here t represents a hypothesized track, z represents a measurement and c represents the cost of associating z with t. A solution, S, to such an assignment problem is a list of triples in which each t and each z appears exactly once and the cost of the solution is given by the sum of cost elements of each triple present in S.

The algorithms starts with finding the single best solution, S_0, using the well-known JV algorithm [4] [35]. Subsequent solution to P_0 are then found by solving a succession of assignment problem generated from P_0 by the process called 'partitioning' [35]. A problem, P, with the best solution S and size n (number of elements in set S), is partitioned into a set of sub-problems, $P_0^{'}, P_1^{'}, .., P_n^{'}$, such that [35]:

1. The union of the set of possible solutions to $P_0^{'}$ through $P_n^{'}$ is exactly the set of possible solutions to P minus the solution S.

2. The set of possible solutions to $P_0^{'}$ through $P_n^{'}$ are disjoint.

To formulate the sub-problem $P_a^{'}$, first P is copied to $P_a^{'}$ and the i^{th} triple in S, $< t_j, z_i, c >$ is removed. This is done to ensure that no solution to $P_a^{'}$ can contain $< t_j, z_i, c >$, so S cannot be a

solution to $P_a^{'}$. Furthermore, before making sub-problems $P_{a+1}^{'}$ through $P_n^{'}$, the triple $< t_j, z_i, c >$ is forced to be in the solutions to those sup-problems. To do so, all the triples $< t_j, z_h, c >$ with $h \neq i$, and $< t_h, z_i, c >$ with $h \neq j$ are removed from P. This ensures that every possible solution to this modified P must contain $< t_j, z_i, c >$, so that the sets of possible solutions to any sub-problem $P_b^{'}$, where $b > a$, will be disjoint from the set of the possible solutions to $P_a^{'}$.

Algorithm 4.2.1 Murty's algorithm [35]

1. Find the best solution, S_0, to assignment problem P_0.
2. Initialize a priority queue of problem/solution pairs to contain only $< P_0, S_0 >$. The top pair on this queue will always be the pair with lowest-cost solution.
3. Clear the list of solutions to be returned.
4.

for $i = 1$ to m, or until the priority queue of problem/solution pairs is empty **do**
  4.1. Take the top problem/solution pair, $< P, S >$, off the queue
  4.2. Add S to the list of solutions to be returned.
  4.3.
  for each triple, $< t, z, l >$, found in S **do**
    4.3.1. Let $P' = P$.
    4.3.2. Remove the triple $< t, z, l >$ from P'.
    4.3.3. Find the best solution, S', to P'.
    4.3.4.
    if S' exists **then**
      4.3.4.1. Add $< P', S' >$ onto the queue of problem/solution pairs.
    end if
    4.3.5. From P, remove triples that include t, and all triples that include z, except for $< t, z, l >$ itself.
  end for
end for

After finding the single best solution, S_0, to the original problem P_0, partitioning is done based on S_0. This partitioning generates sub-problems. The best solutions corresponding to these sub-problems are then added to the priority queue of problem/solution pairs. Then, the problem P is found that has the best solution. The solution to this problem P is the second-best solution to P_0. Now, P is removed from the queue and is replaced by its partitioning. The best solution obtained in this queue will be the third-best solution to P_0. This process is repeated till one finds m-best solutions to P_0 or the priority queue of problem/solution pairs is empty.

Now, the complete process of obtaining $m = 3$ best solutions corresponding to the cost matrix obtained in equation (4.20) is explained. The bipartite graph representation of $\mathbf{C}$ is given as:

$$P_0 = \{T, Z, C\} \tag{4.21}$$

where $T = \{FA_1, FA_2, .., NT_3\}$ is the set of possible tracks, $Z = \{z_1, z_2, z_3\}$ is the set of measurements and C is the cost of associating i^{th} measurement with j^{th} track which is provided in the matrix representation of the cost in equation (4.20). Note that the matrix representation of the assignment problem is used instead of the bipartite graph representation in this example for better illustration.

1. *Step 1*: Find the single best optimal solution. For the given cost matrix, the optimal solution, indicated by the bold elements in (4.20), is (14, 0.9, 1.1) resulting in the minimum possible cost of 16 (=14+0.9+1.1). For simplicity of representation, the solution is written as an array of the cost of assignment of each of the measurements, i.e., 14 represents the cost of association of measurement z_1, 0.9 represents the cost of association of measurement z_2, and so on. Thus, in terms of triple, the solution would be:

$$S_0 = \{< NT_1, z_1, 14 >, < T_1, z_2, 0.9 >, < T_2, z_3, 1.1 >\}$$

Furthermore, the criterion of having each track and each measurement exactly once is not met in this case. To meet this criterion, padding with dummy rows and columns is proposed in [12] [9] such that the cost matrix is a square matrix and thus the required criterion is met. However, the work presented in [10] relaxes this condition and thus can be used on rectangular cost matrices.

2. *Sweep 1*: Now, the second best solution is obtained by successively removing one of the assignments from the previous step solution as a possibility and use them as constraints. Thus, a total of n_Z number of constraints are possible, one for each measurement. The first constraint that is put is to not have 14 in the solution. This is indicated by putting an underline to 14 in the first row, the first column of table 4.4. This constraint results in a modified cost matrix:

$$\mathbf{C}' = \begin{array}{c} \begin{array}{cccccccc} (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) \\ FA_1 & FA_2 & FA_3 & T_1 & T_2 & NT_1 & NT_2 & NT_3 \end{array} \\ \begin{pmatrix} 15 & \infty & \infty & 1.1 & \mathbf{1.2} & \infty & \infty & \infty \\ \infty & 15 & \infty & \mathbf{0.9} & 1.3 & \infty & 14 & \infty \\ \infty & \infty & 15 & \infty & 1.1 & \infty & \infty & \mathbf{14} \end{pmatrix} \end{array} \begin{array}{c} z_1 \\ z_2 \\ z_3 \end{array} \tag{4.22}$$

Note that the element $\mathbf{C'}_{1,6}$ is replaced by ∞ indicating that it can not enter the solution. The best assignment for this constraint is then obtained which is given by (1.2, 0.9, 14) resulting in the overall cost of 16.1.

Now, for the second constraint, the second element of the best solution (of the previous step) is removed from the possible solution list but ensuring that the first element is there in the solution. This is done because the purpose of this step is to determine the next best solution and the best solution excluding $\mathbf{C}_{1,6}$ (= 14) is already obtained. This constraint is denoted by $(14, \underline{0.9})$. Since 14 has to be included in the solution, the first row and the sixth column of the cost matrix can be excluded, resulting in a smaller submatrix:

$$\mathbf{C'_1} = \begin{array}{c} \begin{array}{ccccccc} (1) & (2) & (3) & (4) & (5) & (6) & (8) \\ FA_1 & FA_2 & FA_3 & T_1 & T_2 & NT_2 & NT_3 \end{array} \\ \begin{pmatrix} \infty & 15 & \infty & \infty & 1.3 & \mathbf{14} & \infty \\ \infty & \infty & 15 & \infty & \mathbf{1.1} & \infty & 14 \end{pmatrix} \end{array} \begin{array}{c} z_2 \\ z_3 \end{array} \tag{4.23}$$

The best assignment for this submatrix is indicated by the boldface elements. This assignment results in an overall cost of 29.1. Using the same logic, the constraint for the last step in this sweep would be $(14, 0.9, \underline{1.1})$ resulting in the cost of 28.9. Out of the three constraints in this sweep, constraint $(\underline{14})$ results in the lowest cost. Thus, the solution obtained corresponding to it is the next best solution after the optimal solution obtained in *Step 1.*

Table 4.4: Sweep 1

Constraints	**Solutions**	**Cost**	**Rank**
$\underline{14}$	(1.2, 0.9, 14)	16.1	1
14, $\underline{0.9}$	(14, 14, 1.1)	29.1	3
14, 0.9, $\underline{1.1}$	(14, 0.9, 14)	28.9	2

3. *Sweep 2*: All the steps in sweep 1 are repeated with (1.2, 0.9, 14) as the optimal solution. Also, the constraint that was used to obtain this solution is carried forward to make sure that the best solution of the previous step(s) does not appear again. Thus the first constraint would be $(\underline{14}, \underline{1.2})$. The complete constraint table for this sweep and corresponding solutions with their costs and rank is provided in table 4.5. From this table, the lowest cost is obtained

Table 4.5: Sweep 2

Constraints	Solutions	Cost	Rank
14, 1.2	(1.1, 14, 1.1)	16.2	1
14, 1.2, 0.9	(1.2, 14, 14)	29.2	3
14, 1.2, 0.9, 14	(1.2, 0.9, 15)	17.1	2

for the solution (1.1, 14, 1.1) corresponding to (14, 1.2) constraint. This is the next best solution to the solution obtained in sweep 1 making it the 3rd overall best solution.

The 3-best data association hypothesis corresponding to the solutions obtained from the above steps are:

$$\theta_1 = \begin{bmatrix} 6 & 4 & 5 \end{bmatrix} \tag{4.24}$$

$$\theta_2 = \begin{bmatrix} 5 & 4 & 8 \end{bmatrix} \tag{4.25}$$

$$\theta_3 = \begin{bmatrix} 4 & 7 & 5 \end{bmatrix} \tag{4.26}$$

with cost of 16, 16.1 and 16.2 respectively.

Even with the m-best hypothesis generated from each previous hypotheses, the number of hypotheses, N_h, will grown exponentially with time k.

$$N_h \geq (m)^k \tag{4.27}$$

Thus, it becomes necessary to manage the number of hypotheses such that the complexity of the problem and thereby the computational load is kept under check. In this thesis pruning and capping, explained in section 4.2.4, are used to keep the growth of the hypothesis tree under check. Hypotheses probabilities are weighted against each other to evaluate their quality. This probabilistic evaluation of hypotheses is presented in the next section 4.2.3. Furthermore, the number of best hypotheses to be generated m is a design parameter. Its selection is explained in section 4.2.5.

4.2.3 Probabilistic evaluation of hypothesis

The probabilistic evaluation of the hypotheses is explained in this section based on the expressions and theory provided in [9]. Using this theoretical background as a base, a more pragmatic approach called sequential probability ratio test (SPRT) [5] of evaluating hypotheses based on the track scores (see section 4.1.2) is presented.

A new hypothesis generated at time k, Θ_l^k, is dependent on the assignment at current time step $\theta_l(k)$, and the parent (previous) hypothesis, $\Theta_{p(l)}^{k-1}$ based on the measurements upto and including time $k-1$

$$\Theta_l^k \triangleq \{\Theta_{p(l)}, \theta_l(k)\} \tag{4.28}$$

The subscript $_{p(l)}$ represents that the hypothesis Θ_l^k is generated from its parent hypothesis Θ_p^{k-1}.

The aim now is to calculate the posterior probability of the new hypothesis generated at k. The conditional probability of Θ_l^k can be calculated using Bayes's rule [9]:

$$P\left(\Theta_l^k \mid Z^{1:k}\right) = \frac{1}{c} \underbrace{p\left(Z^k \mid \theta_l(k), \Theta_{p(l)}^{k-1}, Z^{1:k-1}\right) P\left(\theta_l(k) \mid \Theta_{p(l)}^{k-1}, Z^{1:k-1}\right)}_{\text{Association dependent}} \underbrace{P\left(\Theta_{p(l)}^{k-1} \mid Z^{1:k-1}\right)}_{\text{Prior}} \tag{4.29}$$

where c is the normalization constant and Z is the measurement vector. Note that the superscript used in the expression represents the time step(s). The last term of the expression represents the

probability of the parent hypothesis, which should be known from the prior (probability of the parent hypothesis). The other two terms are dependent on the data association sequence used in the current time step.

The first term of (4.29) represents the measurement likelihood, i.e., probability of having the measurement vector Z^k, given the data association sequence $\theta_l(k)$. It can be calculated using equation (4.30) under the assumptions [9]:

1. If the measurement is a false alarm, then its probability density function is assumed to be uniformly distributed within the FOV of the sensors.

2. Probability of observing a new track is also uniformly distributed throughout the FOV of the sensors.

$$p\left(Z^k \mid \theta_l(k), \Theta_{p(l)}^{k-1}, Z^{1:k-1}\right) = \underbrace{\left(\frac{1}{V}\right)^{\phi}}_{\text{false}} \underbrace{\left(\frac{1}{V}\right)^{\nu}}_{\text{new}} \underbrace{\prod_{i=1}^{n_z} [P_D \mathcal{N}_{t_i}[z_i(k)]]^{\tau_i}}_{\text{previous tracks}} \tag{4.30}$$

where,

- V is the tracking volume representing the area of the FOV of the sensors.
- ϕ is the number of false alarms in the association hypothesis
- ν is the number of new tracks in the association hypothesis
- $\tau_i = \begin{cases} 1 \text{ , if } z_i \text{ is associated with previously known track} \\ 0 \text{ , otherwise} \end{cases}$
- P_D is the probability of detection. In the original work of [9], there is no P_D in the expression as they assumed near unity probability of detection.
- $\mathcal{N}_{t_i}[z_i(k)]$ is the kinematic measurement prediction probability density for the previously observed tracks which is given by equation (4.31) under the assumption of predicted measurement has the normal probability density function.

$$\mathcal{N}_{t_i}[z_i(k)] = \frac{1}{\sqrt{|2\pi \mathbf{S}_i(k)|}} \exp\left(-\frac{1}{2}(z_i(k) - \hat{z}_i(k|k-1))^T (\mathbf{S}_i(k))^{-1} (z_i(k) - \hat{z}_i(k|k-1))\right) \tag{4.31}$$

It is easy to see that equation (4.30) essentially is the conditional probability of measurement likelihoods for a given association hypothesis constituting three possibilities for a measurement, namely, being a false alarm, being originated from a new track, and being originated from a previous track.

The second term of equation (4.29) represents the probability of association sequence $\theta_l(k)$ given its parent hypothesis based on the measurements available till and including time $k-1$. It is calculated using equation (4.32)

$$P\left(\theta_l(k) \mid \Theta_{p(l)}^{k-1}, Z^{1:k-1}\right) = \frac{\phi!\nu!}{n_z!} \lambda_{NT}^{\nu} \lambda_{FA}^{\phi} \prod_{t \in \mathbf{T}^{k-1}} (P_D)^{\delta_t} (1 - P_D)^{1-\delta_t} (P_\chi)^{\chi_t} (1 - P_\chi)^{1-\chi_t} \tag{4.32}$$

where,

- λ_{NT} is the intensity variable of the assumed Poisson distribution for the expected number of new targets [9]

- λ_{FA} is the intensity variable of the assumed Poisson distribution for the expected number of false alarms [9]
- P_χ is the probability of deletion of a track t in previous hypothesis tracklist $\mathbf{T}^{k-1}$
- $\delta_t = \begin{cases} 1 \text{ , if track } t \text{ in previous hypothesis tracklist } \mathbf{T}^{k-1} \text{ is detected at } k \\ 0, \text{ otherwise} \end{cases}$
- $\chi_t = \begin{cases} 1 \text{ , if track } t \text{ in previous hypothesis tracklist } \mathbf{T}^{k-1} \text{ is deleted at } k \\ 0, \text{ otherwise} \end{cases}$

Equation (4.32) represents the probabilities of all the permutations of the tracks in the prior to either be detected or miss-detected at time k, along with the possibility of having false tracks and the new tracks in the current time.

Using the expressions (4.30) and (4.32) in equation (4.29), one can calculate the posterior probability of a hypothesis at k.

$$P\left(\Theta_l^k \mid Z^{1:k}\right) = \frac{1}{c'}\beta_{NT}^{\nu}\beta_{FA}^{\phi}\prod_{i=1}^{n_z}[P_D\mathcal{N}_{t_i}[z_i(k)]]^{\tau_i}\prod_{t\in\mathbf{T}^{k-1}}(P_D)^{\delta_t}(1-P_D)^{1-\delta_t}(P_\chi)^{\chi_t}(1-P_\chi)^{1-\chi_t} P\left(\Theta_{p(l)}^{k-1} \mid Z^{1:k-1}\right) \tag{4.33}$$

where, c' is the new proportionality constant, and, β_{NT} and β_{FA} are the probability density of the new target and the false alarms respectively. These are scenario-based parameters that are fixed in this thesis as discussed in section 4.1.3. It should also be noted that during the derivation of these probabilistic expressions, it is assumed that the numbers of new tracks and false alarms are uniformly distributed within the FOV of the sensors and their numbers are Poisson distributed. These assumptions are very common in the multi-target tracking literature [4] [9] [5]. They are already taken into consideration while designing the measurement setup discussed in section 3.2.1.

A more pragmatic approach called sequential probability ratio testing (SPRT), is used in practice for the evaluation of hypothesis based on the probabilistic expressions derived in this section [4] [5]. The recursive expression of log-likelihood scores of the tracks, discussed in section 4.1.2 equation (4.9), is used in this method to evaluate the quality of the hypothesis recursively [4]. It is computationally cheap to use the recursively updated track scores instead of calculating the hypothesis probabilities directly using the analytical expressions presented in this section [4]. As described earlier, each hypothesis is essentially a collection of tracks representing a particular state of the targets present in the ego vehicle's environment. Thus, the un-normalized score of each hypothesis is essentially the sum of the LLR scores of the tracks it constitutes [4]. These un-normalized scores are then normalized and then used to evaluate one hypothesis against another which is discussed next.

4.2.4 Hypothesis management

As explained earlier, the number of hypotheses grows exponentially in MHT, even if only m-best hypotheses are generated from each of the previous hypotheses. To manage the growth of the tree and keeping the computational load in check, hypothesis pruning and capping are used in this thesis.

Pruning

The basic idea in pruning is to delete or prune the hypotheses that have a lower probability than a pre-defined pruning threshold Γ_{prune}. The pruning threshold is a tunable parameter that needs to be selected carefully as described in section 4.2.5. A toy example is used to explain the complete procedure of pruning the hypotheses. Consider at k, there are 10 new hypotheses generated with track scores shown in table 4.6.

Table 4.6: Hypotheses scores and their weights (before pruning)

Hypothesis	Score	Weight
H_1^k	45	0.1754
H_2^k	31.8	0.1239
H_3^k	21.9	0.0853
H_4^k	8.7	0.0339
H_5^k	46.7	0.1820
H_6^k	51	0.1987
H_7^k	5.5	0.0214
H_8^k	22	0.0857
H_9^k	16	0.0624
H_{10}^k	8	0.0312

To prune the unlikely hypothesis, first, the hypotheses are weighted against each other. The scores of the hypothesis are used to find the normalized weights. For example, the normalized weight of H_1^k would be $\frac{45}{\sum Score} = \frac{45}{256.6} = 0.1754$. The normalization step is important because the probability of each hypothesis should sum up to unity. The normalized weight is then used to decide whether the hypothesis is pruned or not. Let the pruning threshold $\Gamma_{prune} = 0.05$, then hypotheses H_4^k, H_7^k and H_10^k will be pruned. After pruning, the weights of the remaining hypotheses are re-normalized which is given in table 4.7. Usually, log-weights are used to avoid floating-point approximation issues.

Table 4.7: Hypotheses scores and their weights (after pruning)

Hypothesis	Score	Weight
H_1^k	45	0.1919
H_2^k	31.8	0.1357
H_3^k	21.9	0.0934
H_5^k	46.7	0.1992
H_6^k	51	0.2176
H_8^k	22	0.0938
H_9^k	16	0.0682

Capping

To keep the computational load predictable, there is a cap put on the maximum number of hypotheses, N_{max}, that can survive at each time step. In this step, the hypotheses are first sorted in the descending order of their weights. Then only the top $N_{ma}x$ number of hypotheses are kept and the rest are pruned. Let us consider the maximum number of hypotheses for the toy example used in 4.7 is 5. In that case, hypotheses H_3^k and H_9^k are pruned and the surviving hypothesis with their sorted weights are provided in table 4.8. Note that the weights are again re-normalized.

Table 4.8: Hypotheses scores and their weights (after capping)

Hypothesis	Score	Weight
H_6^k	51	0.2595
H_5^k	46.7	0.2376
H_1^k	45	0.2290
H_2^k	31.8	0.1618
H_8^k	22	0.1119

Track evaluation

After the evaluation of hypotheses in pruning and capping, the individual tracks within those hypotheses are evaluated. This is done to make sure that the track having a finite possibility of having a true target in it is only propagated in future hypotheses. This is determined based on the track score as discussed in section 4.1.2 equation (4.11). The thresholds calculated for the confirmation and deletion of the tracks in the GNN case (see section 4.1.3) are also used in the MHT implementation as both follow the standard SPRT formulation [4] [5].

4.2.5 Parameter selection

Some tunable parameters were identified during the MHT problem formulation in the previous sections. The effect of these parameters and the basis of their selection are discussed in this section.

Number of hypotheses to be generated, m

The number of hypotheses to be generated from each of the previous hypotheses, m, is a crucial parameter for the expansion of the hypotheses tree. In many MHT implementations, a fixed m is used [37] [12]. However, this method of generating the same number of hypotheses from each of the previous hypotheses is inefficient. This can be explained using an example. Let there are two hypotheses , H_1^{k-1} and H_2^{k-1} exists at $k-1$ with weights 0.9 and 0.1 respectively (corresponding *log*-weights would be -0.1054 and -2.3026 respectively). Suppose $m^k = 5$. Then 5 hypotheses will be generated from each of the previous hypotheses resulting in a total of 10 hypotheses at k. It is more probable that the hypotheses generated from H_2^{k-1} would be of low quality and probably will get pruned after the pruning step. On the other hand, the hypotheses generated from H_1^{k-1} are more likely to produce high quality hypotheses at k. Thus, it is wise to generate more hypotheses from the high quality hypothesis such that more high quality hypotheses are generated and the ambiguity of the scenario is efficiently captured. Furthermore, the number of hypotheses to be generated from a low quality previous hypothesis should be restricted such that a high number of low quality hypotheses are avoided. To that end, equation (4.34) is used in this thesis to generate a number of hypotheses at k from a previous hypothesis based on its weight [37].

$$m^k = \max(1, \lfloor N_{max} \exp(w_h^{k-1}) \rceil) \tag{4.34}$$

where N_{max} is the maximum number of hypotheses that can be maintained and w_h^{k-1} is the log-weight of the previous hypothesis. This formulation takes into account N_{max} to ensure that an overall N_{max} number of relatively high quality hypotheses are generated. For the example used in this section, assuming $N_{max} = 10$, the number of hypotheses generated from H_1^{k-1} would be 9, whereas the number of hypotheses generated from H_2^{k-1} would be 1.

Pruning threshold, Γ_{prune}

The purpose of the pruning threshold Γ_{prune} is to prune the low quality hypothesis. In most of the MHT implementations, the pruning threshold is treated as a pre-defined fixed number [37] [4]. However, treating Γ_{prune} as a fixed quantity might not be the best solution. The average weight of the hypotheses is dependent on the number of hypotheses. For example, the weights of two equally likely hypotheses would be 0.5 while the weights of hundred equally likely hypotheses would be 0.01. One could argue that the threshold values can be kept to very low values considering the maximum number of hypotheses. However, this will result in an inefficient growth of the hypothesis tree as very low quality hypotheses will be allowed to produce more branches. To that end, an online calculation of Γ_{prune} is proposed in equation (4.35), which considers the number of hypotheses present in the current step to decide the pruning threshold.

$$\Gamma_{prune} = \log\left[\frac{1}{1000 \cdot N_h}\right] \tag{4.35}$$

This formulation ensures that the pruning threshold is low when the number of hypotheses is high. Note that the log is used as weights of the hypotheses are also maintained in the logarithmic scale in the implementation to avoid numerical problems.

Capping threshold, N_{max}

The computational load in MHT increases as the number of hypotheses increases. Thus, a cap on the maximum number of hypotheses can be derived from the system's computational capabilities. Furthermore, the higher the degree of ambiguity, the more number of hypotheses are required to capture different possibilities. In this thesis, the ambiguity simulation scenario described in section 5.1 is used to find an optimum number of hypotheses such that satisfactory performance is achieved from the MHT algorithm at a reasonable computational cost. This is explained in section 5.3.1.

4.2.6 MHT output representation

As explained earlier, MHT maintains multiple hypotheses regarding data association. These hypotheses result in multiple possibilities of the number of targets(number of confirmed tracks in that hypothesis) and their states(obtained after Kalman update). These hypotheses are incompatible with each other. This means that if hypothesis H_1^k is assumed to be true, then H_2^k or any other hypothesis cannot be true at time k. The most common way of representing an output of the MHT algorithm is to present the most likely hypothesis at a given time step k [4].

4.3 Toy example: MHT vs GNN

A toy example is used to show how MHT can perform better than GNN in an ambiguous situation. Consider two tracks T_1 and T_2 are given as prior denoted by the two black dots in Figure 4.4. The true trajectory of the tracks is shown by the dotted black lines. Let at $k = 1$, two measurements $Z^1 = \{z_1^1, z_2^1\}$ are received, represented by the two triangles. Now, there are two possibilities regarding the data association:

$$\begin{cases} H_1^1 \triangleq \theta_1^1 = \begin{bmatrix} 2 & 1 \end{bmatrix} \\ H_2^1 \triangleq \theta_2^1 = \begin{bmatrix} 1 & 2 \end{bmatrix} \end{cases}$$

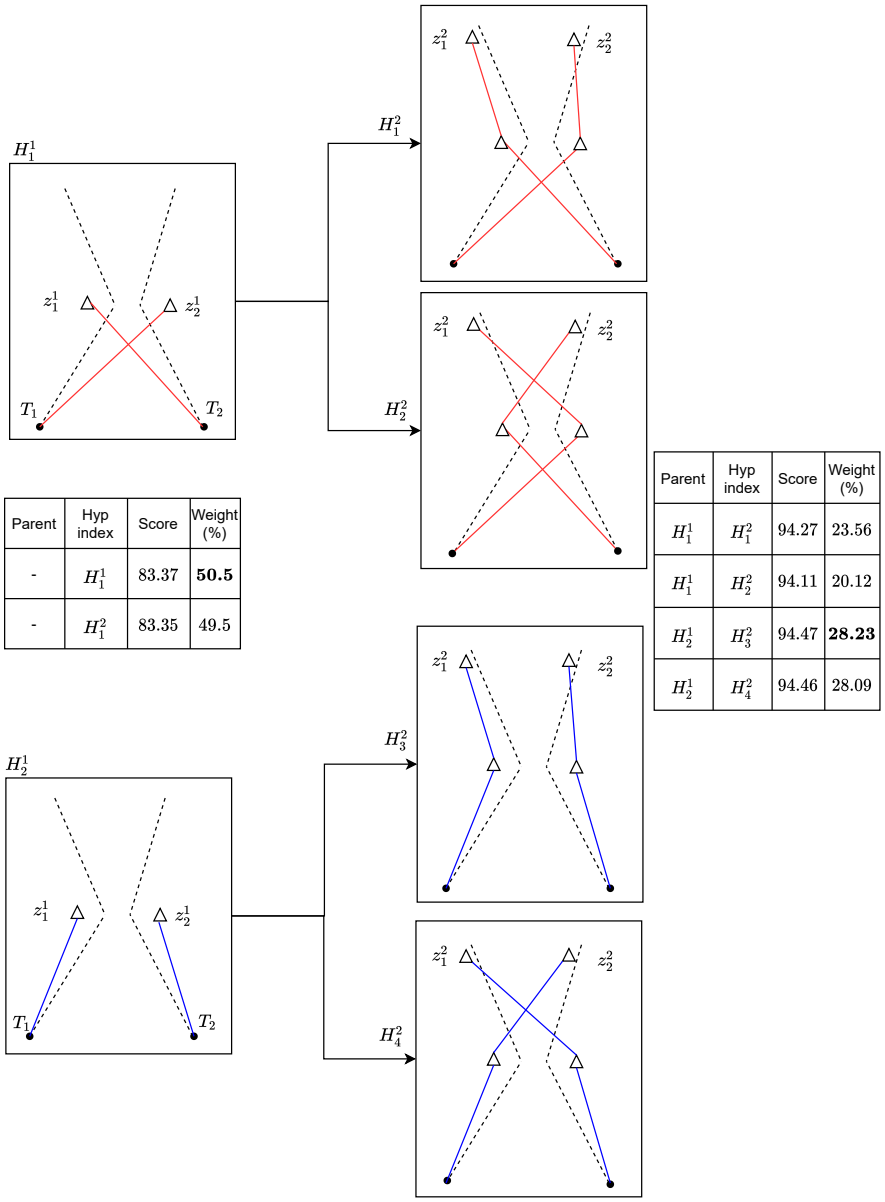
Note that the possibility of false alarm and new track are neglected in this case for simplicity. For these association hypotheses, the track scores are calculated from which the hypotheses probabilities are calculated as shown in Figure 4.4. The score of the hypothesis is given as the sum of the track scores of the two tracks present in that hypothesis. The two association hypotheses are shown by two different colours for the estimated trajectories. The solid red lines represent the trajectories of the two targets for the hypothesis H_1^1 and the solid blue colour line represents hypothesis H_2^1. The best hypothesis at $k = 1$ (H_1^1 with probability 0.505), suggests that the targets cross each other. At this stage, both GNN and MHT produce the same output as shown in Figure 4.6.

At $k = 2$, two new measurement are received $Z^2 = \{z_1^2, z_2^2\}$. From each of the two hypotheses at $k = 1$, two more hypotheses are generated at $k = 2$. The colour of trajectory lines in these hypotheses is kept the same as that of their parent hypothesis. Again the two newly generated association hypothesis for each of the previous hypothesis is as follows:

$$\begin{cases} H_1^2 \triangleq \theta_1^2 = \begin{bmatrix} 2 & 1 \end{bmatrix} & \text{, with parent } H_1^1 \\ H_2^2 \triangleq \theta_2^2 = \begin{bmatrix} 1 & 2 \end{bmatrix} & \text{, with parent } H_1^1 \\ H_3^2 \triangleq \theta_3^2 = \begin{bmatrix} 1 & 2 \end{bmatrix} & \text{, with parent } H_2^1 \\ H_4^2 \triangleq \theta_4^2 = \begin{bmatrix} 2 & 1 \end{bmatrix} & \text{, with parent } H_2^1 \end{cases}$$

The track scores are again calculated for theses association hypotheses and the probabilities of the hypotheses is obtained as shown in the table in Figure 4.4. Note that the hypothesis H_3^2, which is originated from H_2^1, is the most likely hypothesis at $k = 2$, which essentially represents the correct trajectory. In this way, MHT helps in resolving ambiguities. On the contrary, GNN can not solve this ambiguity as it made wrong assignment at $k = 1$ and is not able to solve the ambiguity at $k = 2$. The best hypothesis for GNN at $k = 2$ would be corresponding to H_1^2 which is shown in Figure 4.5.

Resolving ambiguities is important in target tracking applications. Apart from improved tracking accuracy, it helps in having a correct understanding of the environment. For instance, in a vehicle following application, the ability to maintain the correct trajectory of the lead vehicle is of paramount importance. In the toy example presented in this section, the trajectory of the targets got swapped after $k = 2$ in the GNN case. This would have resulted in the wrong trajectory following for the ego vehicle. In such cases, MHT may prove to be a better method. The output presented to the user from both the algorithms are shown in Figure 4.6.



Parent	Hyp index	Score	Weight (%)
-	H_1^1	83.37	**50.5**
-	H_1^2	83.35	49.5

Parent	Hyp index	Score	Weight (%)
H_1^1	H_1^2	94.27	23.56
H_1^1	H_2^2	94.11	20.12
H_2^1	H_3^2	94.47	**28.23**
H_2^1	H_4^2	94.46	28.09

Figure 4.4: Toy example: Ambiguity resolution in MHT

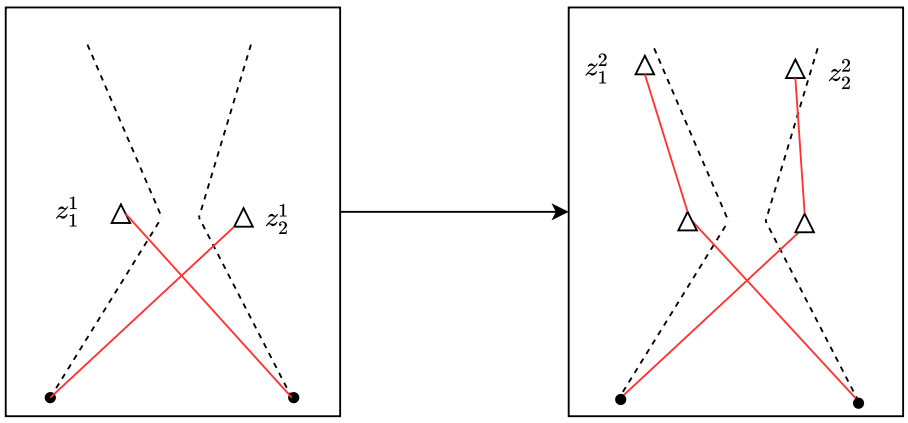


Figure 4.5: GNN data association: Ambiguity unresolved as tracks crossed each other

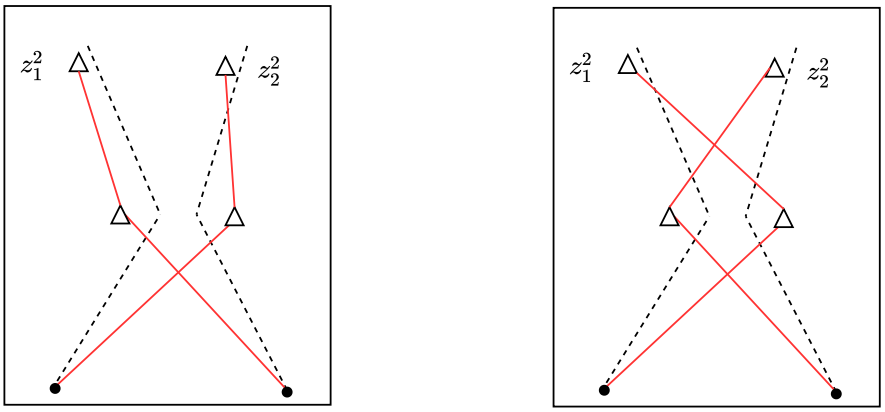


Figure 4.6: Output presented to the user

4.4 Summary

In this chapter, the theoretical and implementation details of the GNN and MHT data association methods were presented. Furthermore, the crucial parameters were identified and their selection procedure was described. The potential advantage of MHT with respect to GNN was also presented using a toy example. A comprehensive performance comparison between the two data association methods is presented in the next chapter.

Chapter 5

Simulation results

In this chapter, the tracking performances of the GNN and MHT data association based multi-target tracking algorithms are evaluated in the simulation scenarios described in section 5.1. Some general and scenario-specific performance metrics are used to compare the performance of the two algorithms. These metrics are introduced in section 5.2. Conclusions from the simulation study for each scenario are also drawn. Finally, a general discussion over the comparison between the two algorithms is provided in section 5.5.

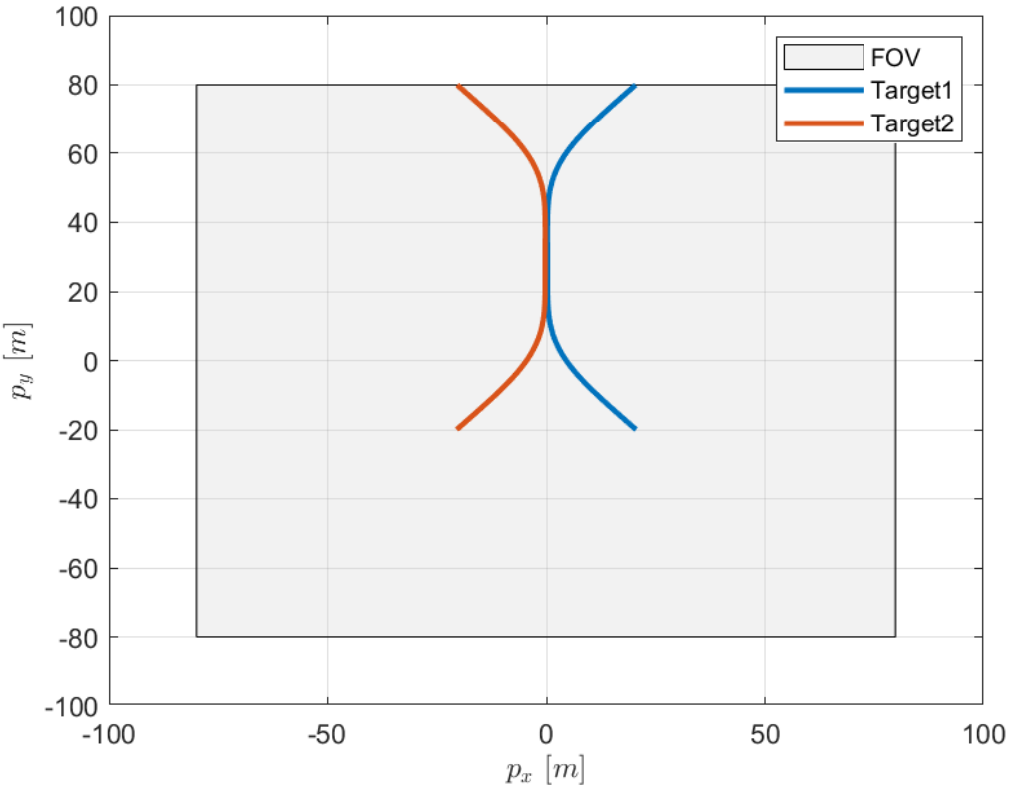
5.1 Simulation scenarios

The basic idea of creating a scenario is based on different requirements of the multi-target tracking mentioned in 1.3.1. It is important to note that every scenario created in this section is in relative coordinates as shown in Figure 3.2. Thus, the ego vehicle is represented as the origin (0,0) and all other trajectories of the targets are plotted relative to the ego vehicle's state. The subscript rel is, however, dropped from the state variables to improve the legibility of the text and the notations. The measurements are generated as described in section 3.2.1 for both the sensors. The field of view (FOV) of both the sensors is assumed to be a square of dimension 80x80 m^2.

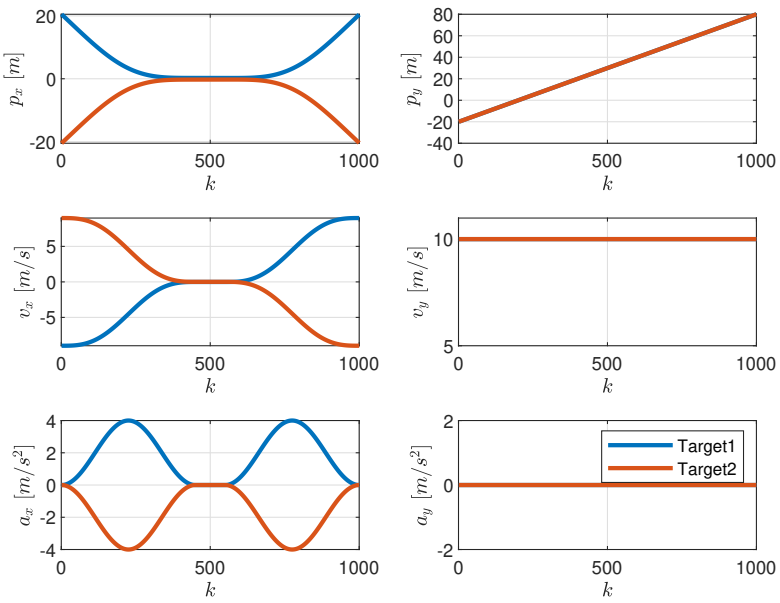
5.1.1 Ambiguity scenario

The purpose of this scenario is to test and tune different data association algorithms when an ambiguous situation arises. The trajectory of the two targets in this scenario is shown in Figure 5.1(a). The two vehicles moving on the right and left of the ego vehicle are moving at a constant velocity in the positive y-direction (with respect to the ego vehicle) merge together and then separate out again. As seen from the time plots of the states of the two targets in Figure 5.1(b), all the measured states, i.e. position and velocity in both x and y position are almost equal for both the objects in the merging phase. This is a highly ambiguous situation that the data association algorithm needs to deal with.

The ambiguity region is defined as the region in which the respective states of the targets are almost the same (within the measurement variance of the sensors). The gap between the two targets in the merging phase in x direction, d_{gap} can be used as a parameter to change the degree of ambiguity. However, it can not be 0 as the true position of the two targets can not be the same at any given time step. Furthermore, the duration of the merging or ambiguity phase, t_{amb} can also be used as a parameter to change the degree of ambiguity. The plots shown in Figure 5.1 are with $d_{gap} = 0.5m$ and $t_{amb} = 1s$.



(a) x-y plot in relative co-ordinates



(b) Time plot of true states in relative co-ordinates. 1 unit of k represents 0.01 s

Figure 5.1: Ambiguity scenario: Two targets moving in positive y-direction merging in and then out

5.1.2 Occlusion scenario

The purpose of this scenario is to test and tune data association algorithms to be able to cope up with the short-term occlusions and/or blind spots. A schematic representation of such a scenario is shown in Figure 5.2. In this scenario, Target2 is occluded from the ego vehicle when it runs parallel to Target1 which is stationary with respect to the ego vehicle (depicted by the blue dot in Figure 5.3(a)). The trajectory of Target2 is shown by the red line in Figure 5.3(a). The dashed line in the trajectory of Target2 represents the occluded region. The true states of the two targets with time are shown in Figure 5.3(b). The duration of occlusion t_{occ} is dependent on the scenario parameters like the length of the vehicles, relative velocity, etc. For the scenario shown in Figure 5.3, $t_{occ} = 1s$ (from time step 450 to 550). The reason behind choosing the occlusion period as 1 second is based on the overtaking scenario study presented in [15].

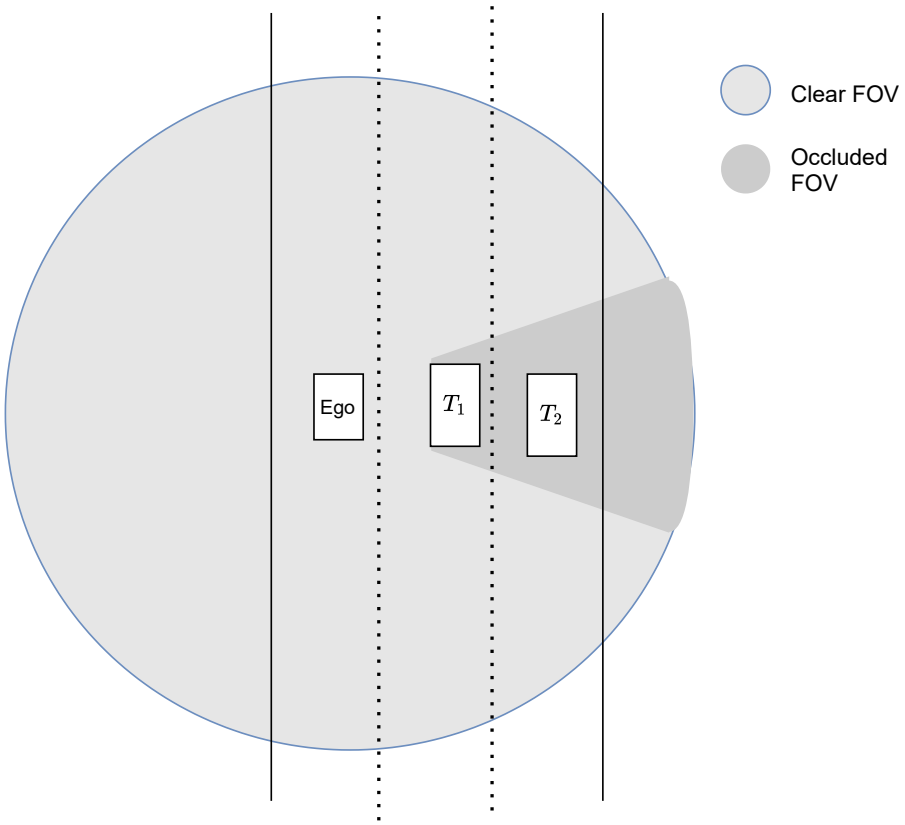
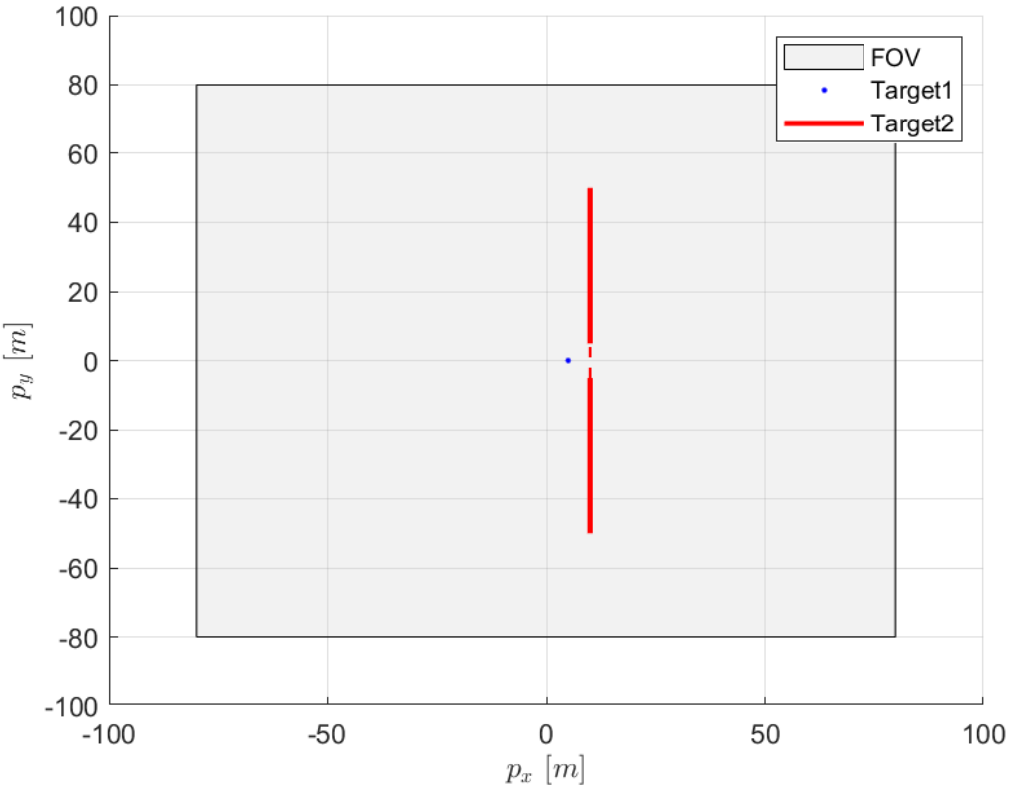
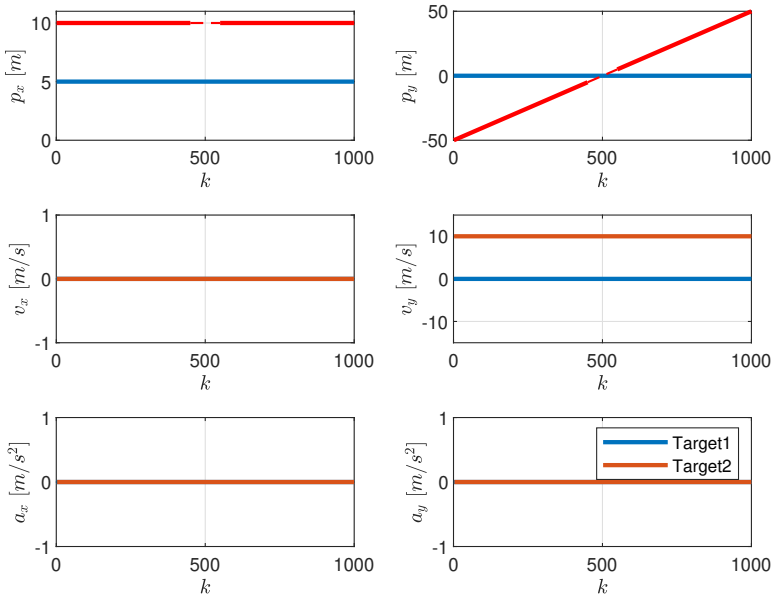


Figure 5.2: Schematic representation of occlusion of target T_2 due to obstructed FOV from target T_1



(a) x-y plot in relative co-ordinates



(b) Time plot of true states in relative co-ordinates. 1 unit of k represents 0.01 s

Figure 5.3: Occlusion scenario: Target2 is occluded from Target1. Occlusion of Target2 due to Target1 is depicted by the dashed line in its position trajectory

5.2 Performance metrics

5.2.1 GOSPA

The GOSPA metric is used to quantify the error between two finite sets. In the multi-target tracking (MTT) problem, the set of estimated tracks and the set of true targets are the two finite sets. The goal is to find error between these two sets to provide objective values to evaluate the performance of different algorithms. This metric provides an elegant way to include three sources of errors, namely: localization, missed target, false track error. These errors can be calculated at each time step, thus, helps in gauging the performance of the algorithms in real-time [45]. However, in reality, the ground truth is not known and thus GOSPA cannot be used as the on-board performance metric. Nevertheless, it can be used in simulation studies to evaluate the performance of the MTT algorithms.

Let the set of ground truths and estimates be $\mathbf{G}$ and $\mathbf{E}$. Then, the GOSPA metric error can be expressed as an optimization problem to minimize the cost of truth to assignment sets [45],

$$e(\mathbf{G},\mathbf{E}) = [\min_{\gamma\in\Gamma}(\sum_{(i,j)\in\gamma} d(G_i,E_j)^p + \frac{c^p}{2}(|\mathbf{G}|+|\mathbf{E}|-2|\gamma|))]^{\frac{1}{p}} \tag{5.1}$$

where γ is the set that contains the ground truth and estimate pairs, c denotes the cut-off distance between the truth and estimate pair for consideration of making a pair, i.e., if the distance between a truth and an estimate is more than c, they will not be considered to make a pair (not included in γ), and p is the order of the metric which is usually taken as 2 such that the error is represented in root mean square(RMS) form [45]. The value of c depends on the application and should be tuned. It is fixed to a value of 20 in this thesis based on the value used in [60]. An example is used next to show how GOSPA error is calculated.

Consider at an instance k, the true and estimated point targets are depicted in a 2D state space as shown in Figure 5.4. Here, $\mathbf{G} = \{G1, G2\}$ and $\mathbf{E} = \{E1, E2, E3\}$. The states of these points (not shown in the figure) are as follows:

- G1: (15,30)
- G2: (55,25)
- E1: (30,55)
- E2: (15,35)
- E3: (28,10)

In this case, only $G1$ and $E2$ made a pair as the euclidean distance between the two, $d = 5$, is less than c. Thus, assignment pair $\gamma = \{(G1, E2)\}$. Apart from this localization error between the assigned pairs, the false track and missed target are penalized by the same factor, i.e., $\frac{c^p}{2} = 200$. The number of unassigned ground truth or missed targets is given by

$$\#\text{Missed} = |\mathbf{G}| - |\gamma| \tag{5.2}$$

Here $|.|$ represents the cardinality operator to give the number of elements in a set. Similarly, the number of unassigned estimates of false tracks is given by

$$\#\text{False} = |\mathbf{E}| - |\gamma| \tag{5.3}$$

Thus, in this case, the total number of missed targets and false tracks is given by $|\mathbf{G}| + |\mathbf{E}| - 2|\gamma| = 2 + 3 - 2 \cdot 1 = 3$. Using these values, the total GOSPA error at k is calculated using (5.1):

$$GOSPA_k = (5^2 + 200(3))^{1/2} = 25 \tag{5.4}$$

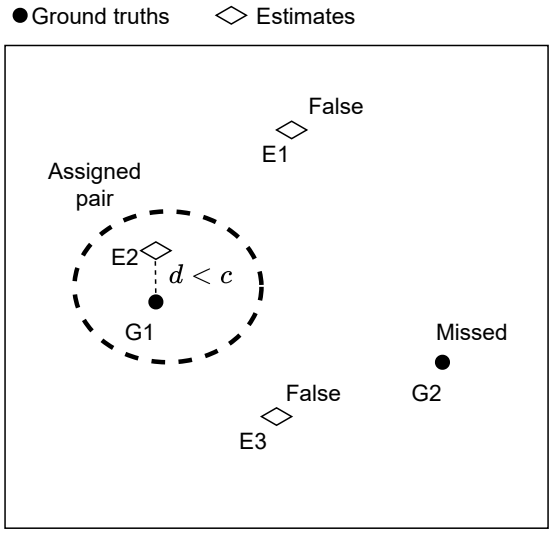


Figure 5.4: True vs estimated states at time instance k. Figure not to scale. G1-E2 make an assignment pair. E1 and E3 are false tracks, whereas G2 is a missed target.

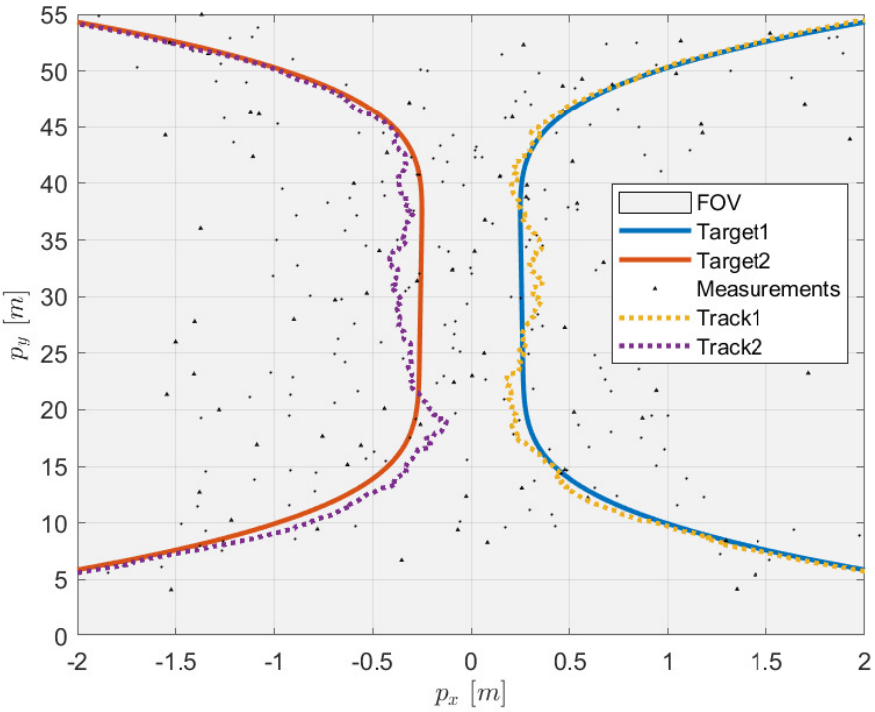
5.2.2 Ambiguity Resolution Success Indicator (ARSI)

The ambiguity resolution success indicator (ARSI) is an indicator developed for the ambiguity scenario considered in this thesis. It is developed to automate the process of judging the success of the tracker in resolving the ambiguity without having to look at the tracking result plots manually.

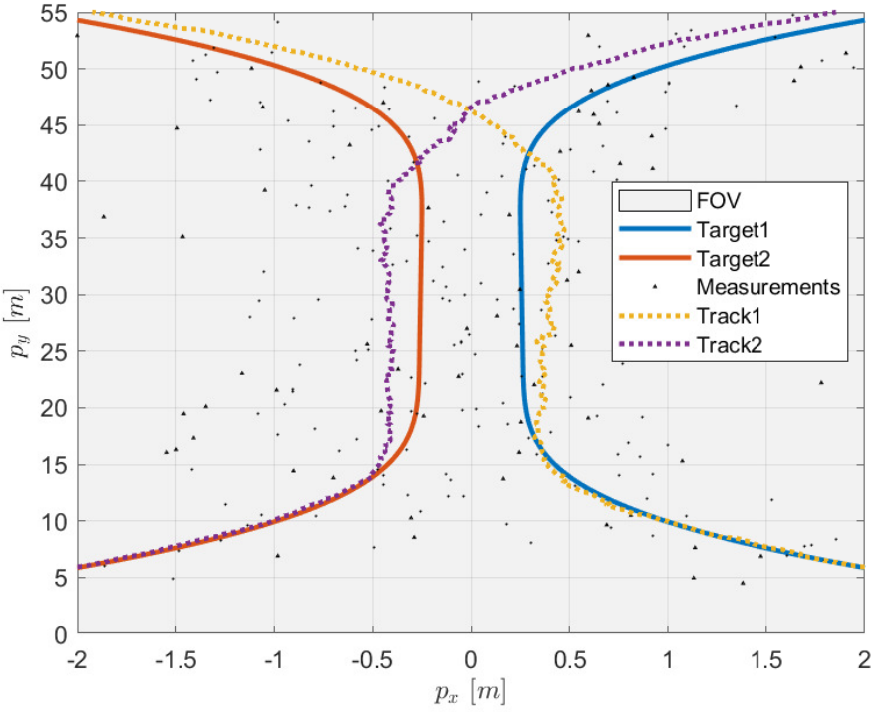
As shown in Figure 5.1, the initial and final x-position of both the targets have the same sign, i.e., Target1 starts and ends on the left of the ego vehicle, whereas Target2 starts and ends on the right of the ego vehicle. In estimating the trajectories of these two targets, there are two possibilities. The first possibility is that the ambiguity is resolved and Track1 turns left after the ambiguous region as shown in Figure 5.5(a). This is considered a success. The other possibility is that the tracks get swapped after the ambiguous region as shown in Figure 5.5(b). This is considered a failure. Thus, the ambiguity resolution is declared successful if the sign of the initial and final estimate of each of the tracks remains the same. This forms the basis of ARSI.

5.2.3 Track Continuity Indicator (TCI)

The track continuity indicator (TCI) is a performance metric developed for this thesis to indicate the success of a data association algorithm in avoiding the discontinuity of an occluded track. If the occluded track is deleted and registered as a new track after the occlusion period, it is judged as a failure. Thus, for the occlusion scenario presented in section 5.1, TCI represents an overall success/fail criteria in maintaining the track continuity of the occluded target.



(a) Success



(b) Failure

Figure 5.5: Ambiguity scenario: $x - y$ plot. Note the difference in the scales of x and y directions

5.3 Tuning and evaluation: Ambiguity scenario

As discussed in section 4.2.5, the ambiguity scenario is first used to find the optimum capping threshold to have an optimal balance between performance and computation cost. Once this parameter is fixed, the performance of the MHT algorithm is compared with the baseline GNN algorithm with varying degrees of ambiguity in section 5.3.2. Since tracking performance is very much dependent on the quality of the measurement data. To that end, the same random number generators were used for both GNN and MHT algorithms for a fair comparison. In addition, the simulations were iterated for a sufficient number of different random number generators to draw statistically significant conclusions. The sample size calculated for a confidence level of 99%, and a 5% error margin is approximately 650 [49]. See appendix D for further details regarding this.

5.3.1 Optimal capping threshold N_{max} for MHT

As explained in section 4.2.5, selecting a general hypothesis capping threshold is not trivial. More hypotheses are generally required to be maintained as a higher degree ambiguity situation arises [37]. However, maintaining more hypotheses results in increased computational cost. Thus, a performance versus computation time study is proposed in this section to chose a suitable capping threshold N_{max} for the given ambiguity scenario.

A reasonable ambiguous situation is considered in this simulation study with $t_{amb} = 1s$ and $d_{gap} = 0.5m$. The ambiguity duration is chosen arbitrarily whereas the gap is chosen such that the targets remain within one standard deviation of the position measured by the radar sensor. This scenario is then simulated for 650 different iterations for different values of N_{max}. The success ratio in terms of ARSI is defined as:

$$\text{Success ratio} = \frac{\text{Number of successful ambiguity resolution}}{\text{Total number of iterations}} \tag{5.5}$$

Figure 5.6 shows the success ratio with different N_{max}. As expected, the success ratio certainly improves with increasing N_{max}. On the other hand, the average computation time for the simulation increases exponentially with increasing the capping threshold N_{max} as shown in Figure 5.7.

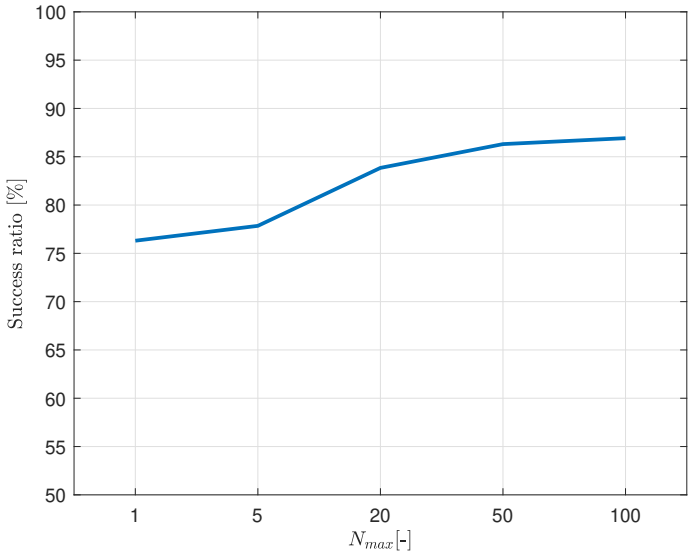


Figure 5.6: Success ratio with different capping thresholds N_{max} values

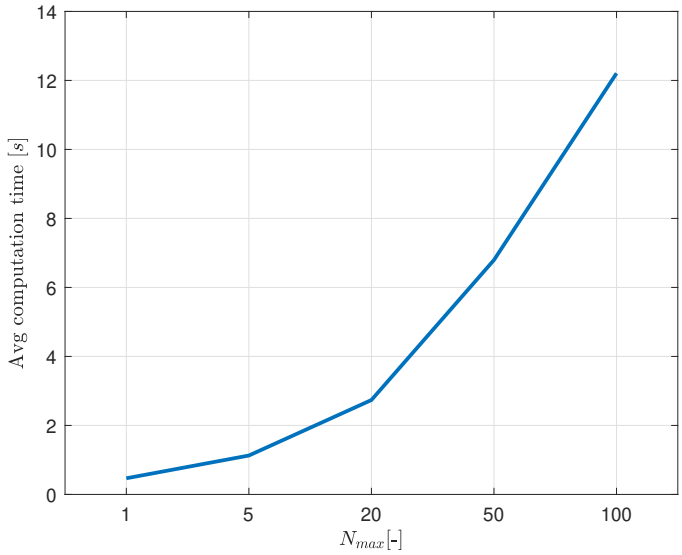


Figure 5.7: Average computation time with different capping threshold values

In Figure 5.8, the cumulative moving average of the average GOSPA error values, $\texttt{GOSPA}_{\texttt{CMA}}$, for each of the 650 iterations is plotted for different N_{max} values. Since error in GOSPA metric is calculated in terms of euclidean distance at each time instance, its value will be non-negative. Thus, average of the GOSPA error values is taken as quantitative performance measure for each iteration. Using these average GOSPA values, the $\texttt{GOSPA}_{\texttt{CMA}}$ plot is plotted. A sudden jump in the $\texttt{GOSPA}_{\texttt{CMA}}$ error is observed at iteration number 247 for lower values of N_{max} in Figure 5.8. In this case, three tracks were generated resulting in a high false track penalty in the GOSPA metric. This false track generation is avoided in MHT with $N_{max} = 50$ or higher as seen from the $\texttt{GOSPA}_{\texttt{CMA}}$ plot. The reason for this observation is explained in appendix C. It is one of the nice test cases showing the benefit of MHT over GNN.

The GOSPA metric severely penalizes the false track or missed track instances. This can cause sudden jumps in the $\texttt{GOSPA}_{\texttt{CMA}}$ plots. Thus, localization error can be considered to draw more general conclusions. The cumulative moving average of the average localization error values, $\texttt{local}_{\texttt{CMA}}$ is shown in Figure 5.9. The average values of the localization error seem to have an insignificant difference for different values of N_{max}. The root-cause of this observation is the dominance of the acceleration error component. The bar plots of the average values of position, velocity, and acceleration and total errors over all 650 iterations are provided in Figure 5.10. The order of acceleration error is around 30 times more than the position or velocity error. This is expected since the constant acceleration motion model is not followed in the majority of this scenario (see Figure 5.1). Furthermore, the acceleration errors for all the N_{max} values are practically constant. Thus, it is decided to consider only the position and velocity states in the localization error calculations for the rest of the analysis in this scenario. The cumulative localization error considering only the position and velocity states, $\texttt{localposvel}_{\texttt{CMA}}$ is shown in Figure 5.11. The localization error decreases with increasing N_{max}.

Thus, considering the success ratio, computation time, and localization error plots, $N_{max} = 20$ is fixed for this thesis. This value is not general as it is based on a particular scenario. In practice, multiple scenarios should be considered to fix it.

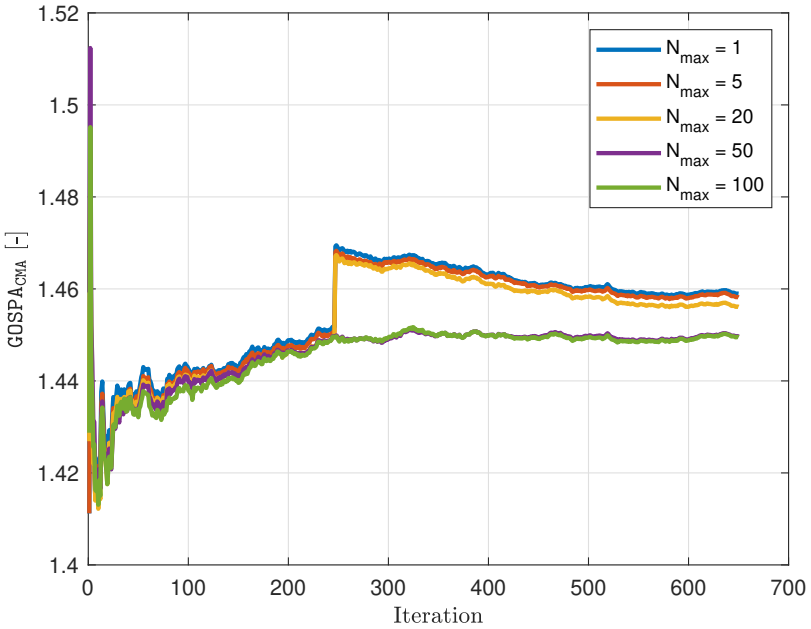


Figure 5.8: Cumulative moving average of the average GOSPA error values for each iterations with different capping threshold values

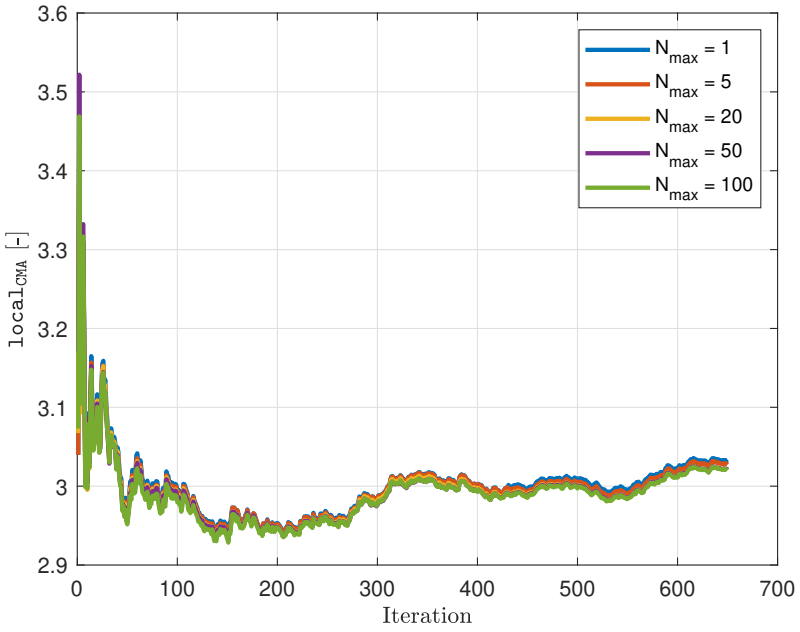


Figure 5.9: Cumulative moving average of the average localization error values for each iterations with different capping threshold values

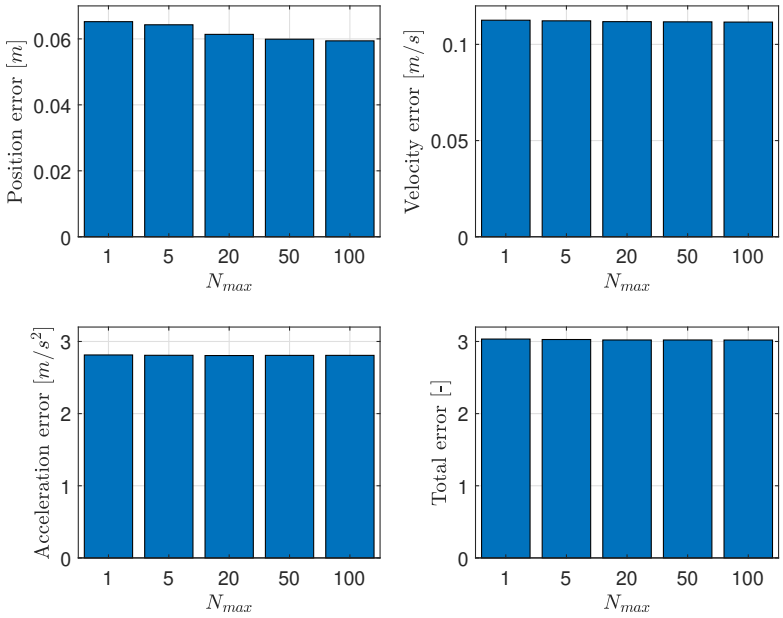


Figure 5.10: Average position, velocity, acceleration and total localization errors over all iterations for different capping threshold values

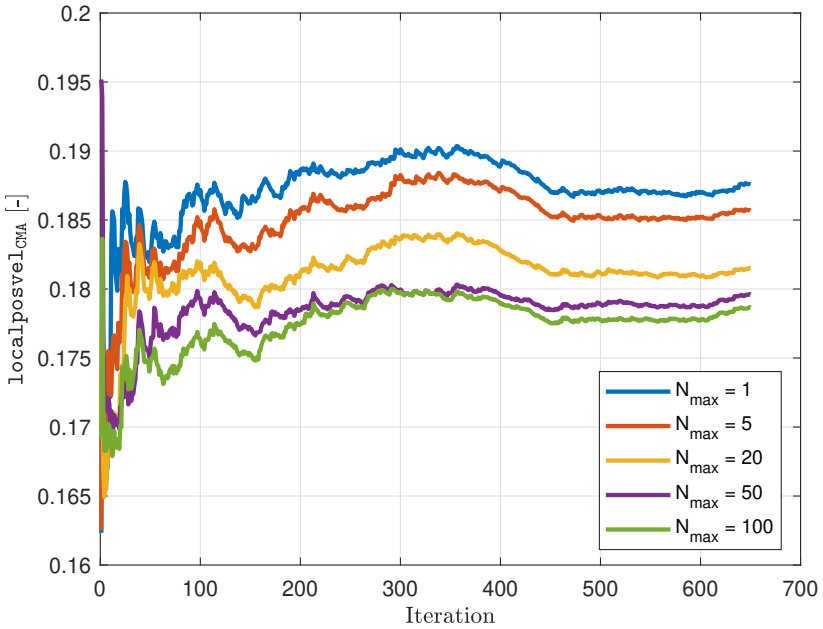


Figure 5.11: Cumulative moving average of the average localization error(only position and velocity considered) values for each iterations with different capping threshold values

5.3.2 GNN vs MHT

With all the parameters fixed for both GNN and MHT, a comparative study of both algorithms is presented in this section for different degrees of ambiguity. The comparison is done based on the success ratio and the localization error component of the GOSPA metric. All other relevant plots are provided in appendix C.

Effect of ambiguity duration t_{amb}

The success ratio for both GNN and MHT with different ambiguity duration (t_{amb}) is shown in Figure 5.12. The gap between the two targets, $d_{gap} = 0.5m$ is kept the same for all cases. The MHT shows on an average 8% more success ratio than GNN. Furthermore, as the ambiguity duration increases, the success ratio for both algorithms decreases. This is expected as the number of hypotheses required for resolving the high degree of ambiguity is not sufficient.

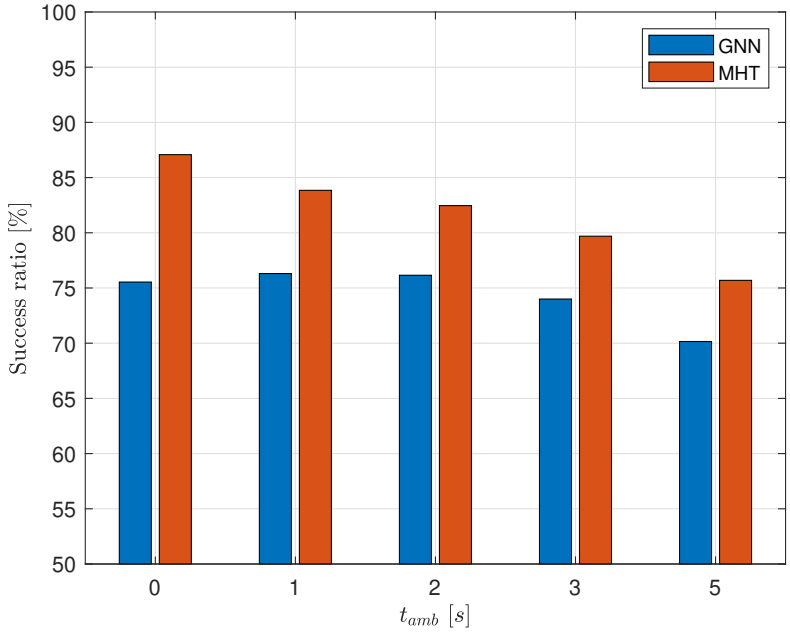


Figure 5.12: Success ratio for different ambiguity duration

The effect of increasing ambiguity duration on the average localization error is presented in table 5.1. As expected, the error values for both algorithms increase with the increase in ambiguity duration. Furthermore, MHT shows better performance than GNN which is expected since it resolves more ambiguities, resulting in better localization. The average improvement in localization error with MHT with respect to GNN in terms of percentage is calculated to be 3% approximately.

Table 5.1: Average localization error for GNN and MHT algorithms with different t_{amb}

t_{amb} [s]	Avg. local error		% improvement
	GNN	MHT	
0	0.1803	0.1688	6.4
1	0.1876	0.1815	3.3
2	0.1991	0.1941	2.5
3	0.2093	0.2061	1.5
5	0.2334	0.2284	2.1

Effect of gap d_{gap}

The tracking performance of MHT and GNN algorithms in terms of success ratio for different values of d_{gap} is shown in Figure 5.13. The duration of ambiguity t_{amb} is kept at $1s$ for all the simulations. Both GNN and MHT have comparable performances when the gap is more than the unit standard deviation of distance measurement of both camera and radar sensors ($> 1\ m$). This confirms that GNN performs well in less ambiguous situations. However, once the gap between the two targets is reduced such that it came inside the unit standard deviation of distance measurement for the sensors, GNN's performance deteriorated significantly in terms of success ratio. Further reduction of the gap resulted in poor performance from the MHT algorithm also. This suggests that the capping threshold N_{max} chosen for the simulation is not enough to resolve the ambiguity successfully. In terms of average localization error presented in table 5.2, MHT seems to perform better than GNN for the reasonable gap between the two targets. However, for very low values of d_{gap}, GNN seems to be at par with MHT.

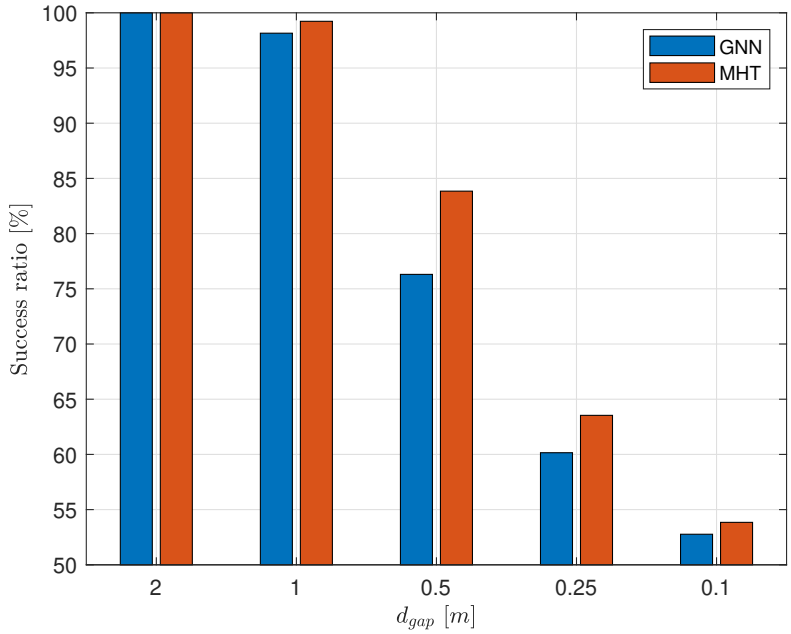


Figure 5.13: Success ratio for different gap values

Table 5.2: Average localization error for GNN and MHT algorithms with different d_{gap}

d_{gap} [m]	Avg. local error		% improvement
	GNN	MHT	
2	0.1424	0.1424	0.0
1	0.1547	0.152	1.7
0.5	0.1876	0.1815	3.3
0.25	0.2018	0.2001	0.8
0.1	0.2022	0.2024	-0.1

5.3.3 Conclusion

From the simulation results obtained in this section for the ambiguity scenario, it can be concluded that:

1. In less ambiguous situations, GNN performs at par with MHT.
2. The performance of both GNN and MHT algorithms degrades with the increase in the degree of ambiguity. However, GNN's performance deteriorates quite abruptly with an increase in the degree of ambiguity. MHT performs relatively well as compared to GNN in a moderate degree of ambiguity. For the given simulation setup and nominal degree of ambiguity, MHT is $\approx 8\%$ more successful than GNN in terms of ambiguity resolution and has reduced the localization error by $\approx 3\%$.
3. MHT allows the possibility to improve the performance by maintaining more number of hypotheses. This is not possible in GNN though.

5.4 Tuning and evaluation: Occlusion scenario

The occlusion scenario described in section 5.1 is used in this section to study the effect of occlusion duration t_{occ} on the performance of GNN and MHT algorithms. It is desirable to keep a track alive for the occluded period and avoid discontinuity in its trajectory. As discussed in section 4.1.3, one can tune the T_{drop} threshold to keep an occluded track alive for the desired duration. To have a fair comparison between GNN and MHT, the T_{drop} threshold is kept the same for both the algorithms. The track continuity and GOSPA error metric are used to compare the two algorithms. The simulations are repeated 500 times with different random number generators to generate different measurement signals. The sample size of 500 is chosen arbitrarily. As shown in Figure C.5 in appendix C.2, the cumulative moving average of the average GOSPA errors seems to have stabilized after 500 iterations. Thus, it is deemed sufficient to draw statistically significant conclusions from this study. Furthermore, the track continuity success ratio (TCSR) in terms of track continuity indicator (TCI) is defined as:

$$\text{TCSR} = \frac{\text{Number of successful TCI}}{\text{Total number of iterations}} \tag{5.6}$$

5.4.1 Case 1: $t_{occ} = 1s$

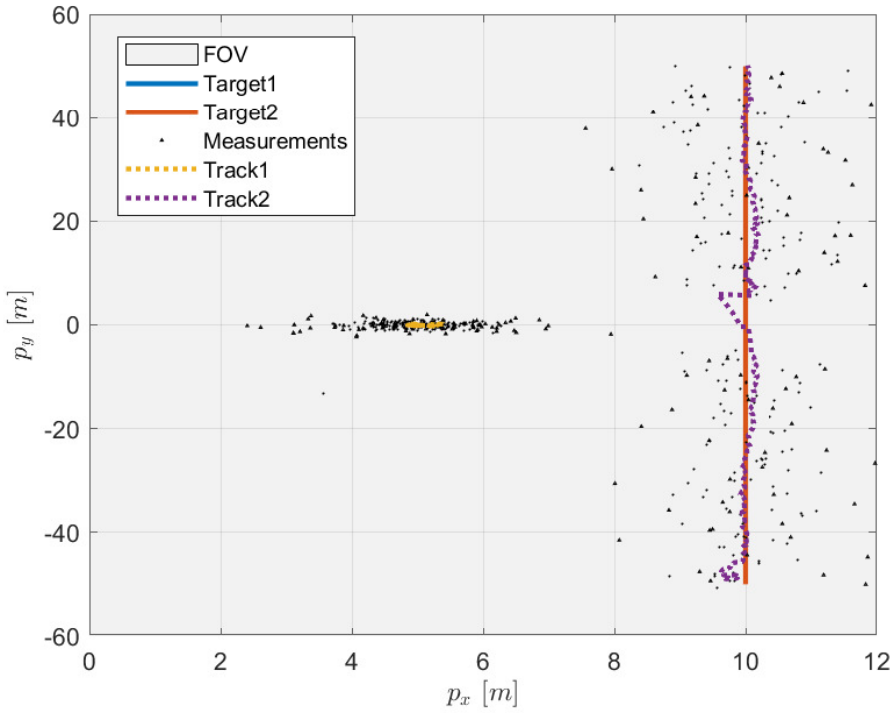
In this case, both GNN and MHT kept the target alive for the occluded period in all iterations. The average GOSPA error for GNN and MHT is shown in table 5.3. Both methods produce the same tracking performance which is expected since the tracks are far apart from each other and there is only one most likely hypothesis for the two tracks.

5.4.2 Case 2: $t_{occ} = 1.1s$

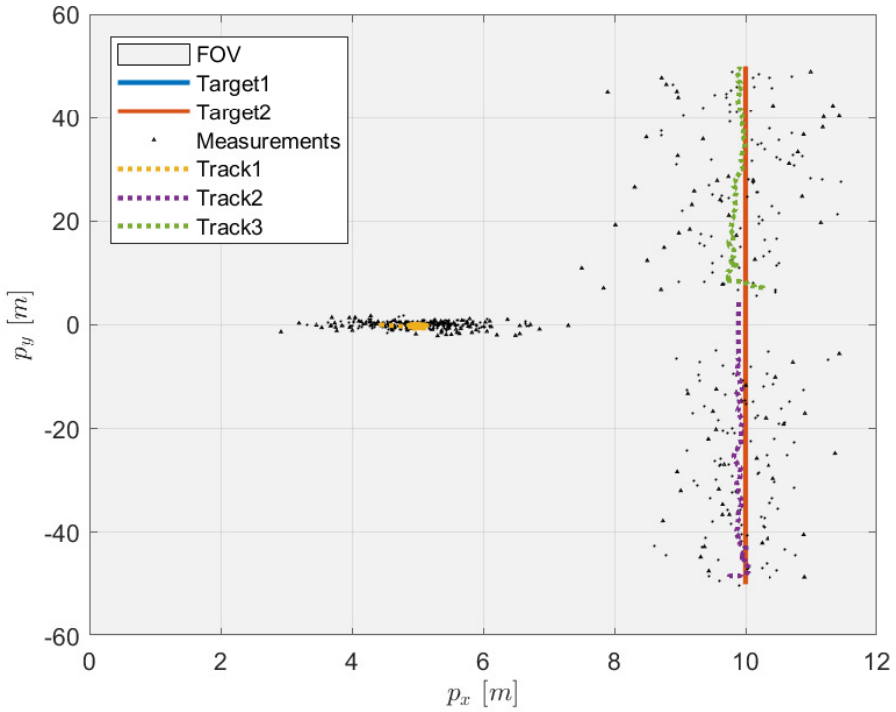
For this case, the occlusion period is increased to 1.1 seconds, which is 0.1 seconds more than the corresponding T_{drop} threshold value for track deletion. The motivation behind this simulation is to see if MHT helps in the track continuity of Target2, even if the T_{drop} threshold is kept below the requirement. The simulation result for this case is also presented in table 5.3. It is observed that out of 500 iterations, only 17 times (3.4%) the track for Target2 is continued with GNN whereas MHT is able to continue the track for 90 times out of 500 iterations (18%). The tracking result for a successful and an unsuccessful iteration, in terms of continuity, is shown in Figure 5.14. Target1 in these figures is not visible as it is hidden behind the measurements and the estimated Track1. Analyzing the successful cases of GNN, it is observed that the occluded track is updated with the clutter measurements resulting in less drop in its track score and thus, is not deleted just after the occlusion period. However, in most of the MHT successful cases, the occluded track is updated with the measurements from Target1 in some hypotheses. Even though the probabilities of such hypotheses were low, they helped in retaining the track once measurements are available again for the occluded target.

Table 5.3: GNN vs MHT: Occlusion scenario with T_{drop} fixed corresponding to $t_{occ} = 1s$

t_{occ} $[s]$	TCSR %		Avg. GOSPA	
	GNN	MHT	GNN	MHT
1	100	100	1.057	1.057
1.1	3.4	18	1.322	1.287



(a) Success



(b) Failure

Figure 5.14: Occlusion scenario: x-y plot. $t_{occ} = 1.1s$. Note the missing measurements for Target2 in the occluded region.

5.4.3 Effect of increasing T_{drop}

One could argue to increase the T_{drop} threshold such that a track can be kept alive for the longer occluded duration. A simulation study is presented in this section where the T_{drop} is fixed corresponding to $t_{occ} = 3s$. The tracking result of one such case for GNN and MHT data association methods is shown in Figure 5.15. For both GNN and MHT, Track2 makes an erratic jump towards Track1. This is because the prediction uncertainty of Track2 became so large that it started to steal the measurements from Target1 for its update. The Kalman gain for this update is very high (as Kalman gain depends on the ratio of state prediction error covariance and measurement error covariance). This causes the sudden jumps in the trajectory of Track2. Furthermore, a new track, Track3, is created when Target2 is observed again while Track2 is deleted, whereas the same Track2 is continued in the MHT case. This is because MHT maintains a hypothesis that does not update Track2 with Target1's measurements and thereby continues Track2 on its predicted trajectory. The probability of this hypothesis was lower than the hypothesis of stealing the measurements. That is why the erratic jumps are presented to the user. However, once the measurements for Target2 are available again (after the occlusion period), the probability of that hypothesis (which did not update the Track2 with Target1's measurement) became dominant and thus Track2 was continued. Such recovery is not possible in GNN which may lead to the deletion of Track2 and the creation of a new track once Target2 is observed after the occlusion period. However, the prediction uncertainty for the distracted Track2 grows so large that the measurement generated by the Target2 after the occlusion period may also lie in its gating region, and hence it is continued even in the GNN case. The TCSR and average GOSPA for 500 iterations for this case is provided in table 5.4.

Table 5.4: GNN vs MHT: Occlusion scenario with T_{drop} fixed corresponding to $t_{occ} = 3s$

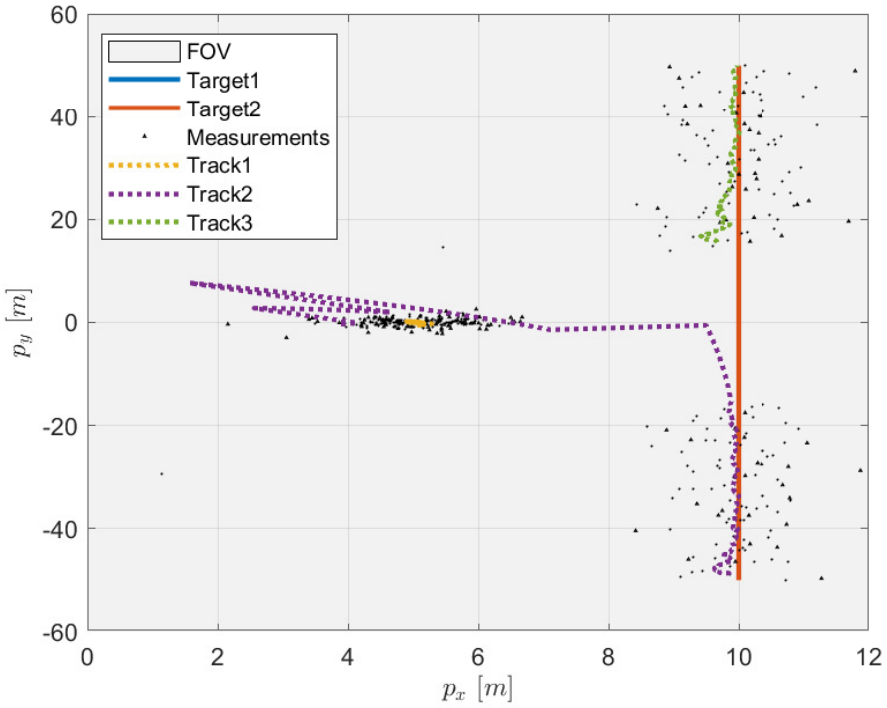
t_{occ} $[s]$	TCSR %		Avg. GOSPA	
	GNN	MHT	GNN	MHT
3	25.6	34.2	2.745	2.737

5.4.4 Conclusions

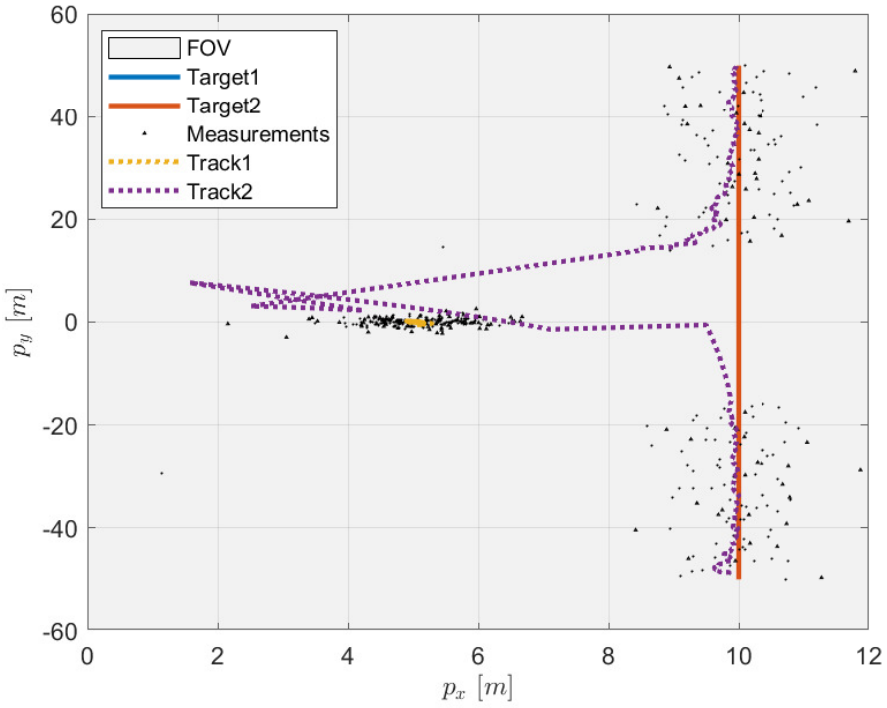
The main conclusions that can be drawn from the presented simulation study for the occlusion scenario:

1. MHT can outperform GNN in terms of track continuity as it may maintain some low probable hypothesis which prevents deletion of the occluded track.

2. Increasing track deletion thresholds to accommodate longer periods of occlusion t_{occ} is not the best solution as it leads to the undesirable data association hypothesis because of the increased prediction uncertainty of the missed target.



(a) GNN



(b) MHT

Figure 5.15: Occlusion scenario: x-y plot. $t_{occ} = 3s$. Note the missing measurements for Target2 in the occluded region.

5.5 Discussion

Based on the comparative study presented in this chapter, it seems that MHT outperforms GNN in ambiguous situations. The performance gain of MHT over GNN is visible in various performance metrics. It is also seen that the performance of the MHT algorithm can be improved by maintaining more hypotheses. However, the computation cost will be a limiting factor, and one has to find an optimum balance between performance and computational expense. Furthermore, the probabilistic nature of MHT allows the target tracker to output the confidence it has in the understanding of the environment. It might be possible that the trajectory of the targets in two comparable hypotheses is significantly different. This may result in a sudden jump in the trajectories when the best hypothesis in the next time steps originates from the different hypothesis in the previous time step. Thus, it is important to present the confidence that MHT has in its best hypothesis to the user. For instance, if the ratio between the best and the second-best hypothesis probabilities is high, the confidence in the best hypothesis is high. Similarly, if this ratio is close to unity and the states of the targets in these hypotheses are significantly different, the confidence in the best hypothesis would be low. This information could be vital to inform the user about the ambiguous situation and thereby take necessary actions. Such a feature however is not possible with GNN. It is also observed that the tracking error is significantly influenced by the choice of motion model being used. In the ambiguity scenario, the acceleration error component is most dominant as constant acceleration motion is not followed by the targets. This observation confirms that the motion model plays a significant part in overall tracking accuracy.

The occlusion scenario study shows that MHT can retain tracks for a longer duration than GNN even if the track deletion threshold is kept the same for both methods. This is because MHT maintains some low probable hypotheses that help in retaining the occluded tracks once they are observed again. However, longer occlusion duration results in undesirable data association hypotheses due to increased uncertainty of the occluded track. This calls for a design decision of balance between the desire to retain a track or delete a track if its uncertainty grows. Furthermore, the basic assumption taken in this thesis of having a constant probability of detection (P_D) is not met in this scenario. The probability of detection in the occluded area will be less than the probability of detection in the clear area of FOV. This consideration can be used to partition the measurement space online to deduce the appropriate value of P_D as depicted in Figure 5.2.

Chapter 6

Conclusions and recommendations

6.1 Conclusions

The main aim of this thesis is to find a data association algorithm that can perform better than GNN in a dynamic environment for a multi-target tracking application. To achieve this aim, two research questions were posed:

1. Which data association algorithm works best in a dynamic environment and what is the performance gain achieved with the newly developed algorithm with respect to the baseline GNN method?

2. What are the crucial tuning parameters that can improve the overall performance of the target tracker?

To answer the first question, a probabilistic framework is chosen to simulate various ambiguities and uncertainties involved in dynamic driving situations. Various performance metrics suggest that MHT outperforms GNN in such ambiguous situations. MHT is $\approx 8\%$ more successful in resolving ambiguities resulting in an improvement of $\approx 3\%$ GOSPA localization error with respect to the baseline GNN method. Furthermore, MHT is $\approx 12\%$ more probable to continue an occluded track for the same track deletion thresholds used for both GNN and MHT. Even though the results obtained in this study are based on simulations that are limited in nature and number, the comparison study still holds as the simulation framework including the Kalman filter and measurement setup is kept same for both the data association methodologies. Thus, it is concluded that MHT can be a better data association method than GNN in a dynamic environment.

To address the second question, a mathematical base for both the algorithms are presented from which the tuning parameters were identified during their implementations. It is seen that, as opposed to GNN, the performance of the MHT algorithm can be improved by tuning some design parameters like the number of hypotheses to be generated(m^k), capping threshold(N_{max}), and pruning threshold(Γ_{prune}). Out of these parameters, m^k and Γ_{prune} can be calculated online and hence their tuning can be avoided. On the other hand, the proper choice of N_{max} seems to be dependent on the degree of the ambiguity present in the scenario. A higher N_{max} is required to deal with a higher degree of ambiguity. The ambiguity scenario simulation study suggests that N_{max} has a significant influence on the overall tracking performance of the tracker. However, computation cost will be a limiting factor in deciding this threshold to find an optimum balance between performance and computational expense.

6.2 Recommendations for future work

1. Scenario (in)dependent tuning: In the current implementation, several scenario-based parameters were assumed to be constant such as the probability of detection, probability of

having a false alarm, probability of new tracks, etc. These parameters are vital in calculating the thresholds for track evaluation. However, these parameters will not be constant in dynamic driving situations. For example, these parameters will be very different when driving in the city while compared to driving on a highway. This calls for a methodology to tune these parameters online based on the driving situation. One possible way is to use real-world data from different scenarios to obtain these parameters. Then, use the obtained parameters according to the driving situations. The driving situation can be deduced from GPS location, road signs, number of detections obtained, etc.

2. Effect of communication and computation latency: There will be latencies involved with measurement data arrival and its transmission, along with the computation time delays in practical MTT implementations. These latencies are neglected in this thesis. However, they might be crucial in determining the performance of the tracker. Thus, it is recommended to study the effect of these latencies on the overall performance of the tracker.

3. Tuning and evaluation with real-world data: It is recommended to use the real-world traffic data set to further test and tune the MHT algorithm. However, generating ground truth data from such data set would be challenging.

4. Performance metric capturing the complete trajectory: The GOSPA metric used in this thesis treats the ground truth and estimated states at each time step as two random finite sets and calculates the euclidean distance between these sets. It does not take the trajectories of the ground truth and estimated tracks into account. Thus, the swap of the trajectories of the two tracks in the ambiguity scenario is not penalized heavily, resulting in a very similar average GOSPA scores for both the resolved and unresolved ambiguity cases. It would be interesting to include the estimated track IDs and the ground truth IDs to calculate the errors.

5. Efficient implementation of the MHT algorithm: The implementation of MHT presented in this study is based on the theoretical base of hypothesis-oriented MHT presented in [9] and [5]. As explained in section 4.2.2, the same copies of a track may be present in different hypotheses. To avoid such inefficiencies, track-oriented MHT [5] can be used wherein, single copies tracks are maintained and they are used to form multiple hypotheses using a global look-up table. Even though this will not have any effect on tracking performance since the same probability expressions will be used for the evaluation of the tracks and hypotheses, it can be computationally cheap. It is recommended to use graph theory for such implementations as it will provide a better data structure to maintain look-up tables.

6. Programming language: All simulations and implementation presented in this thesis are done on MATLAB which is a high-level interpreted programming language. Significant computational savings can be achieved by implementing these algorithms on a compiled programming language like C or C++. The study presented in [10] suggests that solving assignment problems in C++ is 379 times faster than the MATLAB implementation.

References

[1] Joachim Benjaminsson and Emil Rosenberg. "Multiple Object Tracking with Camera Data and Deep Learning Detectors". In: *M.Sc. thesis, Chalmers university of technology* (2017).

[2] N. Bergman and A. Doucet. "Markov chain Monte Carlo data association for target tracking". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2 (2000), pp. 705–708. ISSN: 15206149. DOI: 10.1109/ICASSP.2000.859057.

[3] Keni Bernardin and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: The CLEAR MOT metrics". In: *Eurasip Journal on Image and Video Processing* (2008). DOI: 10.1155/2008/246309.

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

[5] Samuel S. Blackman. "Multiple hypothesis tracking for multiple target tracking". In: *IEEE Aerospace and Electronic Systems Magazine* 19 (2004). DOI: 10.1109/MAES.2004.1263228.

[6] David S. Bolme et al. "Visual object tracking using adaptive correlation filters". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 2544–2550. ISSN: 10636919. DOI: 10.1109/CVPR.2010.5539960.

[7] Katie Burke. *Why Perception Is Key to Safe Self-Driving: NVIDIA Blog*. May 2019. URL: https://blogs.nvidia.com/blog/2018/08/10/autonomous-vehicles-perception-layer/.

[8] Ingemar J Cox. "A Review of Statistical Data Correspondence". In: *International Journal of Computer Vision* 10 (1993), pp. 53–66.

[9] Ingemar J. Cox and Sunita L. Hingorani. "An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18 (1996), pp. 148–150. ISSN: 01628828.

[10] David F. Crouse. "On implementing 2D rectangular assignment algorithms". In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696. ISSN: 00189251. DOI: 10.1109/TAES.2016.140952.

[11] J. Elfring et al. "Semantic world modeling using probabilistic multiple hypothesis anchoring". In: *Robotics and Autonomous Systems* 61 (2013). ISSN: 09218890. DOI: 10.1016/j.robot.2012.11.005. URL: http://dx.doi.org/10.1016/j.robot.2012.11.005.

[12] F. Evers. "A Situational Awareness System for Autonomous Vehicles Based on Multiple Hypothesis Tracking". In: *M.Sc. thesis, TU Eindhoven* (2015).

[13] Ángel F. García-Fernández et al. "Poisson Multi-Bernoulli Mixture Filter: Direct Derivation and Implementation". In: *IEEE Transactions on Aerospace and Electronic Systems* 54 (2018), pp. 1883–1901. ISSN: 00189251. DOI: 10.1109/TAES.2018.2805153.

[14] Karl Granström et al. "Poisson Multi-Bernoulli Mixture Trackers: Continuity Through Random Finite Sets of Trajectories". In: *2018 21st International Conference on Information Fusion, FUSION 2018* (2018), pp. 973–981. DOI: 10.23919/ICIF.2018.8455849.

[15] Geertje Hegeman. "Assisted Overtaking: An Assessment of Overtaking on Two-lane Rural Roads". In: *Technische Universiteit Delft* (2008).

[16] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735.

[17] *Jonker-Volgenant global nearest neighbor assignment algorithm*. URL: https://nl.mathworks.com/help/fusion/ref/assignjv.html.

[18] Jehangir Khan et al. "Data association techniques for advanced driver assistance systems using embedded soft-core processors". In: *2011 11th International Conference on ITS Telecommunications, ITST 2011* (2011). DOI: 10.1109/ITST.2011.6060112.

[19] Zia Khan, Tucker Balch and Frank Dellaert. "MCMC-based particle filtering for tracking a variable number of interacting targets". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), pp. 1805–1819.

[20] Maarten Klompmakers. "Object tracking for autonomous and cooperative driving". In: *M.Sc. thesis, TU Eindhoven* (2020).

[21] W. Koch. *Tracking and Sensor Data Fusion: Methodological Frame- work and Selected Applications*. Springer, 2016.

[22] S. Konatowski, P. Kaniewski and J. Matuszewski. "Comparison of Estimation Accuracy of EKF, UKF and PF Filters". In: *Annual of Navigation* 23 (2017). DOI: 10.1515/aon-2016-0005.

[23] Pavlina Konstantinova, Alexander Udvarev and Tzvetan Semerdjiev. "A study of a target tracking algorithm using global nearest neighbor approach". In: *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech)* (2003), pp. 290–295. DOI: 10.1145/973620.973668.

[24] Y Kosuge and T Matsuzaki. "The optimum gate shape and threshold for target tracking". In: *SICE 2003 Annual Conference* 2 (2003), 2152–2157 Vol.2.

[25] Günter Last and Mathew Penrose. "Lectures on the Poisson Process". In: *Lectures on the Poisson Process* August (2017). DOI: 10.1017/9781316104477.

[26] Eui Hyuk Lee, Qian Zhang and Taek Lyul Song. "Markov Chain realization of joint integrated probabilistic data association". In: *Sensors (Switzerland)* 17.12 (2017), pp. 1–17. ISSN: 14248220. DOI: 10.3390/s17122865.

[27] Weiqiang Li, Jiatong Mu and Guizhong Liu. "Multiple object tracking with motion and appearance cues". In: *CoRR* (2019).

[28] X. Rong Li and Vesselin P. Jilkov. "Survey of Maneuvering Target Tracking. Part I: Dynamic Models". In: *IEEE Transactions on Aerospace and Electronic Systems* 39 (2003), pp. 1333–1364. ISSN: 00189251. DOI: 10.1109/TAES.2003.1261132.

[29] Wenhan Luo et al. "Multiple Object Tracking: A Literature Review". In: *CoRR* (2014). URL: http://arxiv.org/abs/1409.7618.

[30] Xinzhu Ma et al. "Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). DOI: 10.1109/iccv.2019.00695.

[31] Patrick Maher. "Bayesian probability". In: *Synthese* 172 (2010), pp. 119–127. ISSN: 15730964. DOI: 10.1007/s11229-009-9471-6.

[32] R. P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, 2007.

[33] Ronald P.S. Mahler. "Multitarget Bayes Filtering via First-Order Multitarget Moments". In: 39 (2003), pp. 1152–1178. ISSN: 00189251. DOI: 10.1109/TAES.2003.1261119.

[34] Anton Milan et al. "Online multi-target tracking using recurrent neural networks". In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (2017), pp. 4225–4232.

[35] ML Miller and HS Stone. "Optimizing Murty's Ranked Assignment Method". In: *Aerospace and Electronic* (1995), pp. 1–17. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=599256.

[36] Michael Motro and Joydeep Ghosh. "Vehicular Multi-Object Tracking with Persistent Detector Failures". In: *CoRR* (2019).

[37] *Multi Object Tracking*. URL: https://www.youtube.com/watch?v=vBW37s0VXF4&ab_channel=MultipleObjectTracking.

[38] Darko Mušicki and Robin Evans. "Joint Integrated Probabilistic Data Association: JIPDA". In: *IEEE Transactions on Aerospace and Electronic Systems* 40 (2004), pp. 1093–1099. DOI: 10.1109/TAES.2004.1337482.

[39] *NS Table d - Chi-square*. URL: https://www.mun.ca/biology/scarr/4250_Chi-square_critical_values.html.

[40] Songhwai Oh, S. Russell and S. Sastry. "Markov chain Monte Carlo data association for general multiple-target tracking problems". In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)* (2004). DOI: 10.1109/cdc.2004.1428740.

[41] Peter Ondrúška and Ondrúška. "Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks". In: *AAAI-16 conference* (2016). URL: http://mrg.robots.ox.ac.uk/mrg.

[42] Jeremy Orloff and Jonathan Bloom. "Comparison of frequentist and Bayesian inference". In: *MIT OpenCourseWare* (2014), pp. 1–7.

[43] "OSPA(2): Using the OSPA metric to evaluate multi-target tracking performance". In: *2017 International Conference on Control, Automation and Information Sciences, ICCAIS 2017* 2017-January (2017), pp. 86–91. DOI: 10.1109/ICCAIS.2017.8217598.

[44] *Performance Evaluation of Tracking and Surveillance*. URL: https://motchallenge.net/workshops/bmtt-pets2017/.

[45] Abu Sajana Rahmathullah, Angel F. Garcia-Fernandez and Lennart Svensson. "Generalized optimal sub-pattern assignment metric". In: *20th International Conference on Information Fusion, Fusion 2017 - Proceedings* (2017). DOI: 10.23919/ICIF.2017.8009645.

[46] Donald B. Reid. "An Algorithm for Tracking Multiple Targets". In: *IEEE Transactions on Automatic Control* 24.6 (1979), pp. 843–854. ISSN: 15582523. DOI: 10.1109/TAC.1979.1102177.

[47] I. Reid. "Estimation II". In: *University of Oxford* (2001). URL: https://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes2.pdf.

[48] Seyed Hamid Rezatofighi et al. "Joint probabilistic data association revisited". In: *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 3047–3055. ISSN: 15505499. DOI: 10.1109/ICCV.2015.349.

[49] *Sample Size Calculatot by Raosoft*. URL: http://www.raosoft.com/samplesize.html.

[50] Simo Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139344203.

[51] H.K. Sarmah. "An Investigation on Effect of Bias on Determination of Sample Size on the Basis of Data Related To the Students of Schools of Guwahati". In: *International Journal of Applied Mathematics Statistical Sciences (IJAMSS)* 2.1 (2013), pp. 33–48. URL: https://www.researchgate.net/publication/303014899.

[52] Dominic Schuhmacher, Ba Tuong Vo and Ba Ngu Vo. "A consistent metric for performance evaluation of multi-object filters". In: *IEEE Transactions on Signal Processing* 56 (2008), pp. 3447–3457. DOI: 10.1109/TSP.2008.920469.

[53] A. Siagkris-Lekkos. "Automatic flower counting using Multiple Hypothesis Tracking". In: *M.Sc. thesis, TU Eindhoven* (2018).

[54] Robert W. Sittler. "An Optimal Data Association Problem in Surveillance Theory". In: *IEEE Transactions on Military Electronics* 8 (1964). ISSN: 05361559. DOI: 10.1109/TME.1964.4323129.

[55] T.P.A. van der Smagt. "Design of a Track Detection Algorithm for a Driverless Formula Student Racing Car". In: *M.Sc. thesis, TU Eindhoven* (2019).

[56] Ba-Tuong Vo, Ba-Ngu Vo and Antonio Cantoni. "Analytic implementations of the cardinalized probability hypothesis density filter". In: *IEEE Transactions on Signal Processing* 55 (2007), pp. 3553–3567. ISSN: 1053587X. DOI: 10.1109/TSP.2007.894241.

[57] Christian Walck. "Hand-book on statistical distributions for experimentalists". In: *Internal Report SUF–PFY/96–01, University of Stockholm* (2007).

[58] John G. Webster et al. *Multitarget Tracking*. Wiley Encyclopedia of Electrical and Electronics Engineering, 2015. ISBN: 047134608X. DOI: 10.1002/047134608x.w8275.

[59] Jason Williams and L. A.U. Roslyn. "Approximate evaluation of marginal association probabilities with belief propagation". In: *IEEE Transactions on Aerospace and Electronic Systems* 50 (2014), pp. 2942–2959. ISSN: 00189251. DOI: 10.1109/TAES.2014.120568.

[60] Yuxuan Xia et al. "An Implementation of the Poisson Multi-Bernoulli Mixture Trajectory Filter via Dual Decomposition". In: *2018 21st International Conference on Information Fusion, FUSION 2018* (2018), pp. 2453–2460. DOI: 10.23919/ICIF.2018.8455236.

[61] Yuxuan Xia et al. "Multiscan implementation of the trajectory poisson multi-Bernoulli mixture filter". In: *Journal of Advances in Information Fusion* 14.2 (2019), pp. 213–235. ISSN: 15576418. arXiv: 1912.01748.

[62] Yao Yao et al. "Deep-learning method for data association in particle tracking". In: *Bioinformatics* 36 (2020). ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa597.

[63] Jungen Zhang, Hongbing Ji and Cheng Ouyang. "A new Gaussian mixture particle CPHD filter for multitarget tracking". In: *ISPACS 2010 - 2010 International Symposium on Intelligent Signal Processing and Communication Systems, Proceedings* (2010), pp. 10–13. DOI: 10.1109/ISPACS.2010.5704760.

[64] Dawei Zhao et al. "Multi-object tracking with correlation filter for autonomous vehicle". In: *Sensors (Switzerland)* 18 (2018), pp. 1–17. ISSN: 14248220. DOI: 10.3390/s18072004.

Appendix A

Probability theory

Probability theory, in its most general sense, deals with modeling the uncertainties associated with any event. There are two points of view for dealing with these uncertainties related to the quantity of interest [42]. The first being the classical frequentist approach, which treats the uncertainty as unknown and deterministic. It relies on the observations about the frequency at which an event is occurring. Thus, in the case of unrepeatable events, the frequentist approach becomes impractical. The other approach is called Bayesian which treats the uncertainty as stochastic and random. It relies on both, the prior information of the event as well as the likelihood of its observation [31]. Thus, it can be used to model many problems including those which are not repeatable. Due to this property, the Bayesian approach is relevant to the MTT problem as tracking scenarios are usually random and stochastic. Some definitions are mentioned below to introduce the concept of probability.

Probability density function

A probability distribution function (PDF) is a function which describes how a continuous random variable is distributed [31]. Integrating the PDF of a random variable in an interval represents the probability of that variable being in that interval, thus, the PDF yields 1 after integrating it over the whole domain space D. For a random continuous variable x, the PDF is denoted as $p(x)$ with properties:

$$p(x) \geq 0 \text{ for all } x, \text{ and } \int_D p(x)dx = 1 \tag{A.1}$$

Probability distributions are used to model different facets of an MTT algorithm. Some of the important distributions are defined below.

Normal distribution

A normal distribution is a probability distribution which is defined by two variables, namely mean μ and covariance Q. It is also called univariate Gaussian distribution. The normal distribution of a random variable x is denoted by $x \sim \mathcal{N}(\mu, Q)$ and its PDF is given by [31]:

$$p(x) = \mathcal{N}(x; \mu, Q) = \frac{1}{\sqrt{|2\pi Q|}} \exp\left(-\frac{1}{2}(x-\mu)^T Q^{-1}(x-\mu)\right) \tag{A.2}$$

For this thesis work, the normal distribution is used to represent various states of an object like position, velocity, etc. with its mean representing the expected value for that state and covariance matrix representing the uncertainty regarding that expected value.

Poisson distribution

It is a discrete distribution used to describe the rate at which an event occurs [25]. If a random discrete variable x is Poisson distributed, then

$$P[x = i] = Po[i; \lambda] = \frac{\lambda^i \exp(-\lambda)}{i!} \text{ for } i = 0, 1, 2, ... \tag{A.3}$$

where λ is called intensity and it represents the average rate of events occurring. Poisson distribution can be used to model different things like the expected number of objects present, clutter detections, new objects, etc. in the FOV.

Uniform distribution

The uniform distribution of a continuous random variable x is given by [57]:

$$p(x) = \begin{cases} \frac{1}{b-a}, \text{ for } a \leq x \leq b \\ 0 \text{ , for } x < a \text{ or } x > b \end{cases} \tag{A.4}$$

Prior distribution

As its name suggests, prior distribution refers to the prior knowledge that we have of a random variable x, before observing any measurement data. It is simply denoted by $p(x)$.

Measurement likelihood

Measurement likelihood represents a function of measurement data z, given the random variable x. It is represented by $p(z|x)$. It should be noted that likelihood does not represent distribution with respect to the random variable x, rather it is a function on x which provides likelihood of the measurement z.

Posterior distribution

The posterior distribution represents the probability distribution of the random variable x, after observing the measurement data z and it is denoted by $p(x|z)$.

Bayes' theorem

Bayes' theorem is used to evaluate the conditional probability of a random variable x given observation z, i.e., $p(x|z)$ using the prior knowledge of the random variable $p(x)$ and the measurement likelihood $p(z|x)$

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} \tag{A.5}$$

A.1 Bayesian filtering

In the context of state estimation, filtering is defined as a process of estimating the states of a dynamic object or target at a given time instance using the measurements observed till that time instance. For dynamic targets, the filtering problem is solved in real-time using a recursive approach of predicting the motion of the target and then correcting that prediction using the measurement data. Bayesian filtering is one of the widely used framework to tackle the general filtering problem. Many closed-form algorithms or filters like Kalman Filter, Extended Kalman Filter, Unscented Kalman Filter, etc. are available to solve this problem. It should be noted that the choice of the filter depends on the type of distribution and/or non-linearities involved in the problem.

Mathematically, the Bayesian filtering problem entails the task of recursively estimating target state x_k, at time k, from observations up to and including time k, denoted by $z_{1:k}$. Evolution of the states of the dynamic object is represented by a Markov chain which essentially means that the state at a given time step x_k is only dependent on the state at previous time step x_{k-1} and is independent of anything that happened before time step $k-1$ [50]. Furthermore, the current measurement z_k given current state x_k is conditionally independent of the measurement and state histories. The dependency of the states and measurements in a Markov chain is depicted in Figure A.1. The arrows between the states and/or measurement show their dependencies. As mentioned earlier, the Bayesian filtering setup has two steps, prediction of the state using a motion model and correction or updation of the estimate using the current measurement.

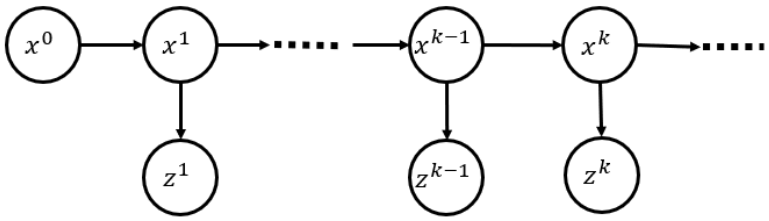


Figure A.1: Schematic showing relation between states and measurements under Markov chain assumption

A.1.1 Prediction step

The aim of the prediction step is to predict the state x_k, given the measurements upto time $k-1$, i.e., $z_{1:k-1}$. This is done by employing Chapman-Kolmogorov equation [1],

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{A.6}$$

where, $p(x_k|x_{k-1})$ represents the transition probability density which is a result of a motion or state transition model and $p(x_{k-1}|z_{1:k-1})$ is the posterior of the state at $k-1$ (which should be available after update step at time $k-1$). Since the prediction is made about the state x_k without observing any new measurement, $p(x_k|z_{1:k-1})$ is called the *prior* distribution.

Motion model

Motion model is a function that is defined to predict the state x_k using the state at previous time step x_{k-1}. The general version of the motion model also include the input signals, however, as discussed above, the dynamics of the target are assumed to be governed by the Markov chain, thus, x_k is only a function of x_{k-1}.

$$x_k = f(x_{k-1}, w_{process,k-1}) \tag{A.7}$$

where $w_{process,k-1}$ is random process noise used to model the error and uncertainties related to the motion model.

A.1.2 Update step

Given the prior prediction of state x_k, one can obtain measurement likelihood by using a sensor or measurement model. The measurement model represents the relation between the state x_k and the measurement z_k and is given by

$$z_k = h(x_k, w_{meas,k}) \tag{A.8}$$

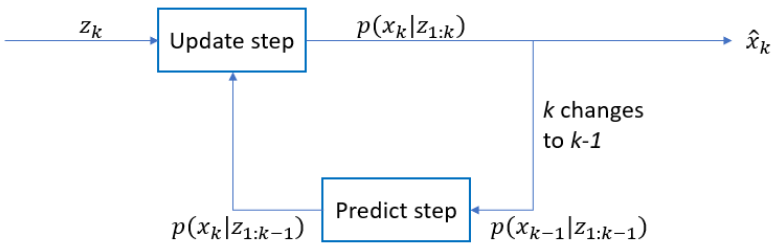


Figure A.2: Schematic showing recursive steps in a Bayesian filter

where, $w_{meas,k}$ is random noise included to capture the uncertainties and error in modelling the sensor output. In distribution terms, the measurement likelihood is denoted as $p(z_k|x_k)$. Now, using Bayes' theorem (A.5), the conditional probability distribution of state x_k after observing the measurement z_k, also called the *posterior*, is given by:

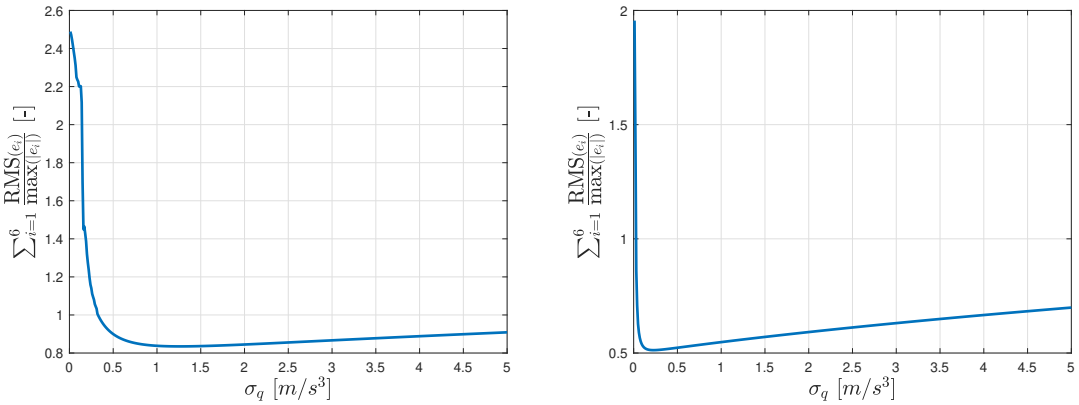
$$p_{k|k}(x_k|z_k) \propto p_{k|k-1}(z_k|x_k)p_{k|k-1}(x_k|z_{1:k-1}) \tag{A.9}$$

The subscript $k|k$ denote the probability distribution calculated at k using the measurements up to time k. This is used to differentiate the posterior distribution from prior and likelihood distributions which are calculated for time k using the measurements till time $k-1$. The overall recursive flow of a Bayesian filter is shown in Figure A.2. The $\hat{x}_k$ represents the extracted state estimated at each time step from the posterior density $p(x_k|z_{1:k})$. For a linear and Gaussian model, $\hat{x}$ essentially is its mean value.

Appendix B

Effect of target motion on filter Tuning

The optimum process noise variable $\sigma_{q,opt}$ value depends on the scenario itself. In this study, the procedure of finding $\sigma_{q,opt}$ (as discussed in section 3.4) is repeated for two different rotational speeds of the target. It is observed that at low rotational speed, $\sigma_{q,opt}$ has a lower value as shown in Figure B.1. Furthermore, from the L-shaped nature of the curve, it can be seen that the risk of using lower value of σ_q than the optimum value (underestimating) is significantly high as compared to overestimating it. Thus, a higher value of $\sigma_{q,opt}$ is opted i.e., $\sigma_q = 1.2m/s^3$ is fixed for this thesis.



(a) For $\omega = 0.1\pi$ rad/s, $\sigma_{q,opt} = 1.2m/s^3$

(b) For $\omega = 0.05\pi$ rad/s, $\sigma_{q,opt} = 0.2m/s^3$

Figure B.1: Effect of rotational speed ω of the target on process noise variable tuning

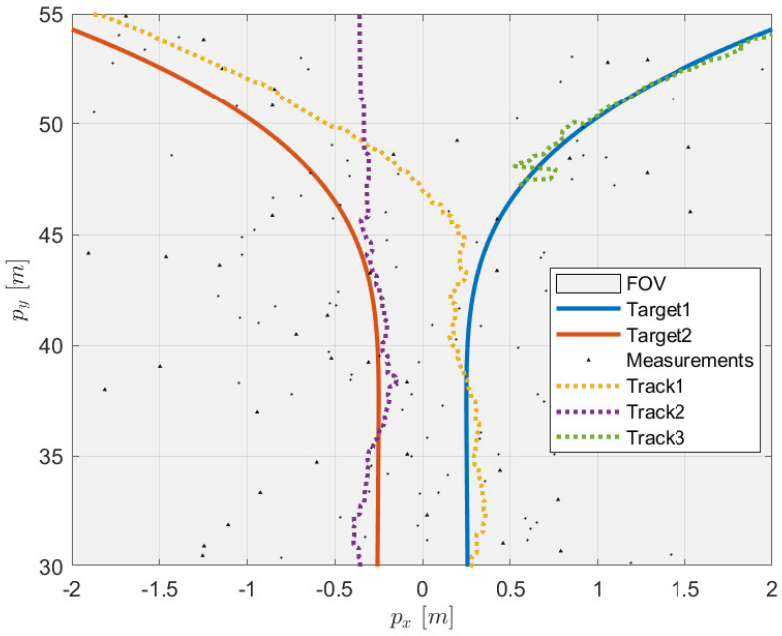
Appendix C

GNN vs MHT: Additional simulation plots

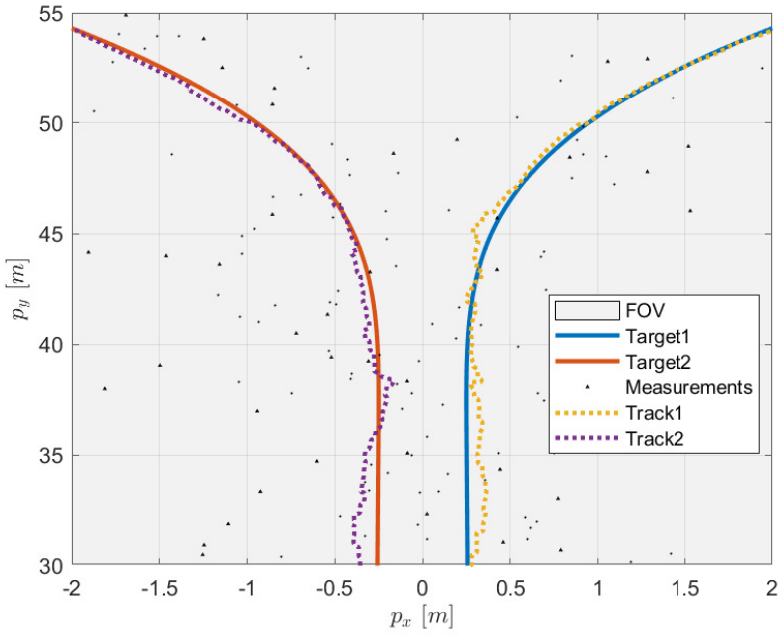
C.1 Ambiguity scenario

C.1.1 Iteration 247: $d_{gap} = 0.5m$, $t_{amb} = 1s$

For iteration number 247, the estimated trajectories are shown in Figure C.1. In GNN case, Track1 takes a left turn at the end of the ambiguous region, i.e., it is associated with the measurements pertaining to Traget2. Furthermore, the Track2 is not associated with any measurement at the end of the ambiguity period (visible from the straight trajectory of the Track2). In the meantime, a new track, Track3, is created for the measurements pertaining to Target1 after the end of the ambiguity period. This results into three estimated tracks for the two actual targets which is also visible in the GOSPA error shown in Figure C.2. Track2 is continued for some time and it is deleted when it fulfils the track deletion criteria explained in section 4.1.3. For the MHT case, the trajectories of the two tracks were slightly different and both tracks were pushed towards the correct directions after the end of the ambiguity period resulting in the resolution of the ambiguity and better tracking performance. The main reason for the success of MHT in this case is that it maintained multiple hypotheses, which resulted into a more likely hypothesis of having two tracks at the end of the ambiguity period, instead of a new track.

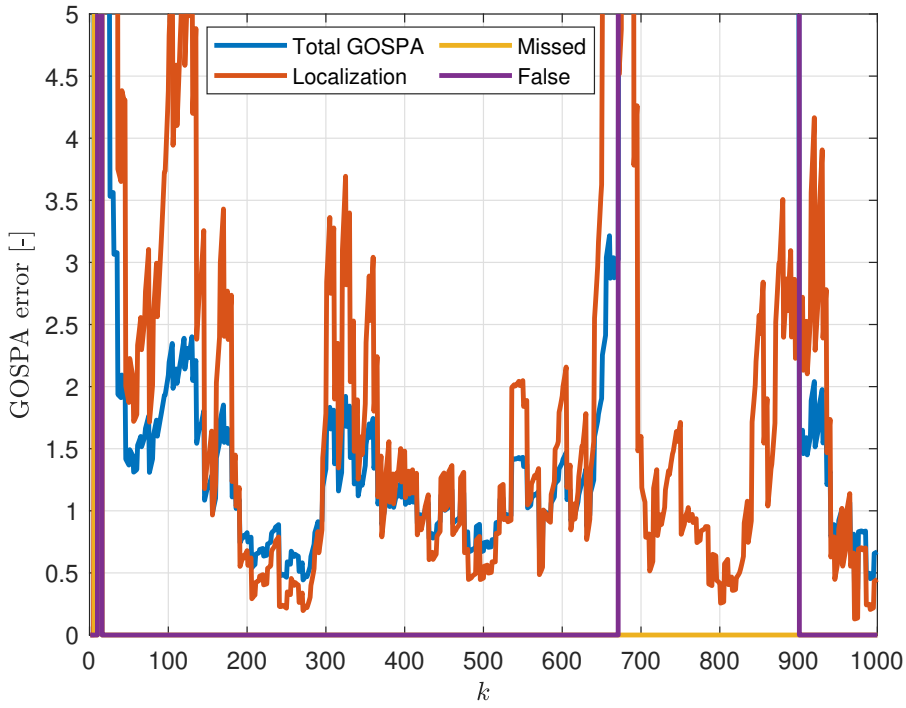


(a) GNN

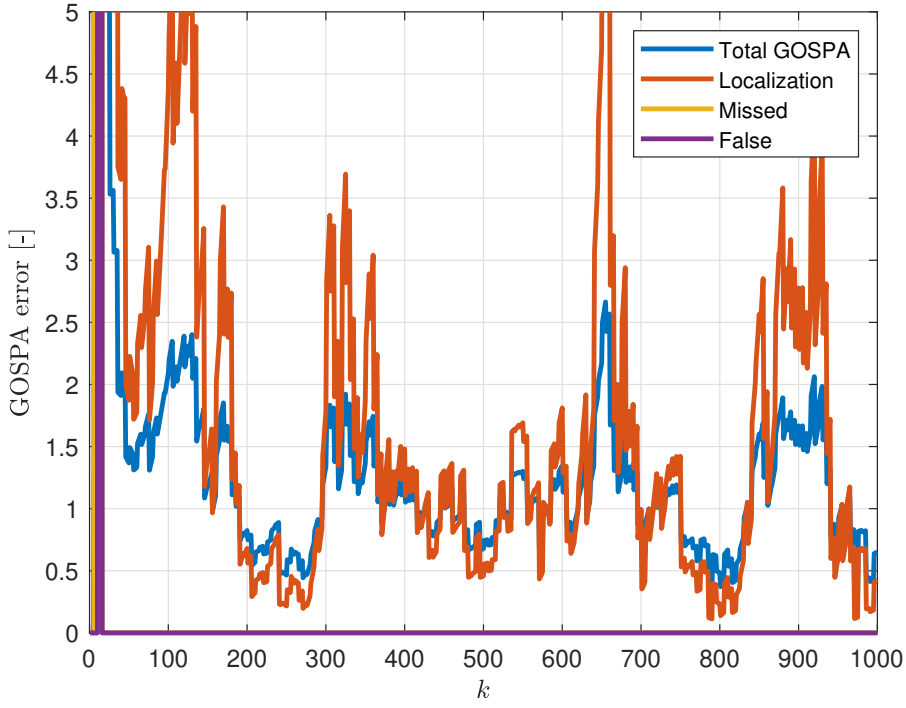


(b) MHT with $N_{max} = 50$

Figure C.1: GNN vs MHT: $x - y$ plot for iteration number 247 in ambiguity scenario. Note that the trajectory shown in this figure is for time steps k from 500 to 750.



(a) GNN



(b) MHT

Figure C.2: GNN vs MHT: GOSPA error for iteration number 247 in ambiguity scenario

C.1.2 Effect of t_{amb}

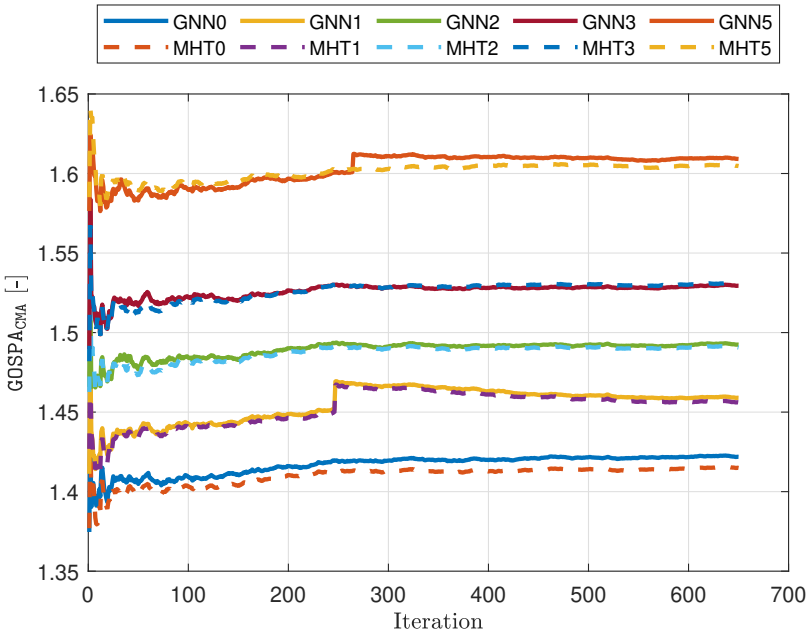


Figure C.3: Cumulative moving average of the average GOSPA error values for each iterations with ambiguity duration. The ambiguity duration is indicated with the number used after the GNN or MHT in the legend box. Thus, MHT1 represents the case of $t_{amb} = 1s$ with MHT data association.

C.1.3 Effect of d_{gap}

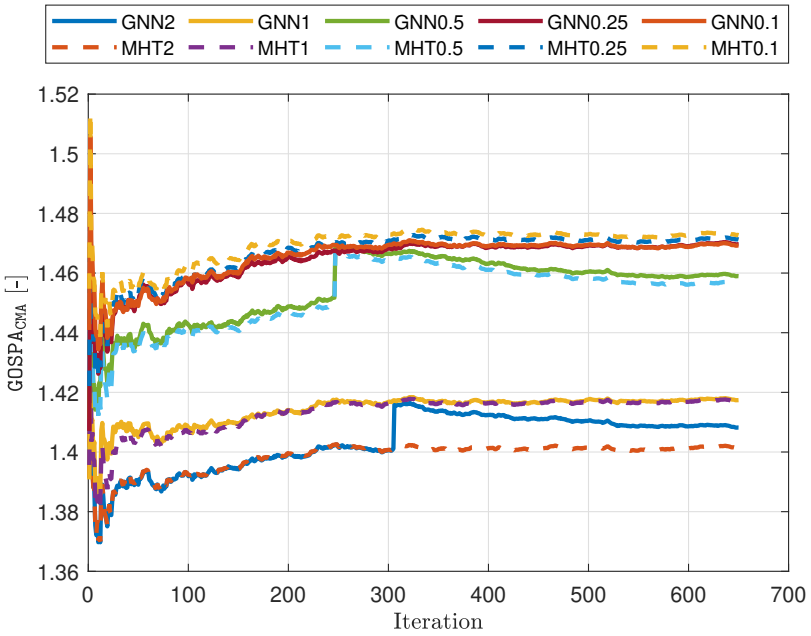


Figure C.4: Cumulative moving average of the average GOSPA error values for each iterations with gap. The gap is indicated with the number used after the GNN or MHT in the legend box. Thus, MHT1 represents the case of $d_{gap} = 1m$ with MHT data association.

C.2 Occlusion scenario

C.2.1 $\texttt{GOSPA}_{\texttt{CMA}}$ for $t_{occ} = 1s$, $T_{drop} = -207.2$

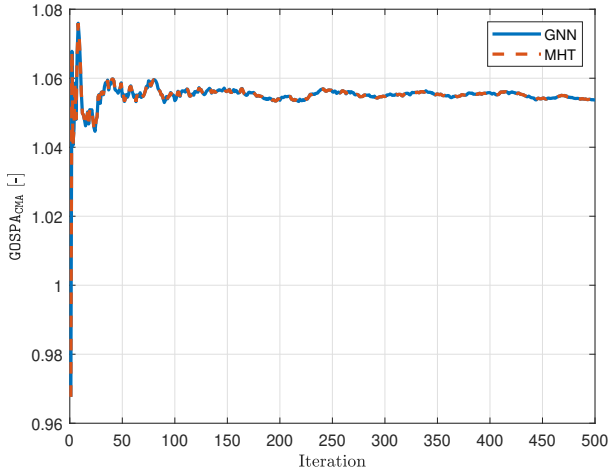


Figure C.5: Cumulative moving average of the average GOSPA error, $\texttt{GOSPA}_{\texttt{CMA}}$ for occlusion scenario with $t_{occ} = 1s$

C.2.2 $\texttt{GOSPA}_{\texttt{CMA}}$ for $t_{occ} = 1.1s$, $T_{drop} = -207.2$

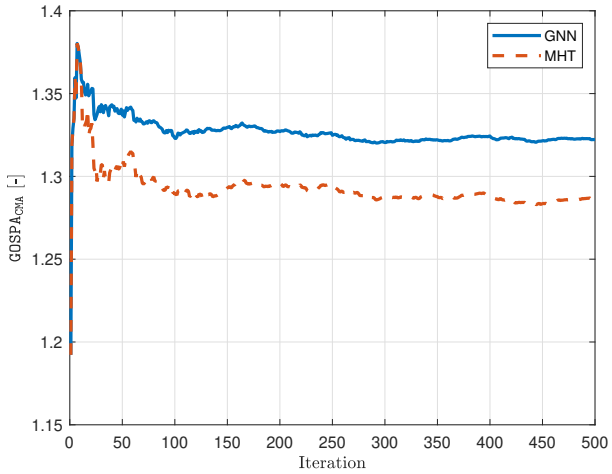


Figure C.6: Cumulative moving average of the average GOSPA error, $\texttt{GOSPA}_{\texttt{CMA}}$ for occlusion scenario with $t_{occ} = 1.1s$

C.2.3 $\texttt{GOSPA}_{\texttt{CMA}}$ for $t_{occ} = 3s$, $T_{drop} = -621.6$

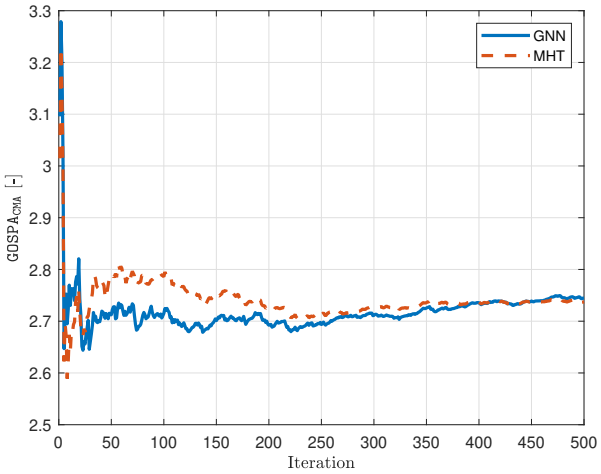


Figure C.7: Cumulative moving average of the average GOSPA error, $\texttt{GOSPA}_{\texttt{CMA}}$ for occlusion scenario with $t_{occ} = 3s$

Appendix D

Sample size calculation for statistically significant results

To draw statistically significant inferences from the simulations that are dependent on the random measurements, it is important to repeat the simulations for enough number of times with different random samples. The sample size n for the number of repetitions or iterations can be calculated from Cochran's formula [51]

$$n = \frac{n_0}{1 + \frac{(n_0 - 1)}{N}} \tag{D.1}$$

where n_0 is the sample size when population size, i.e., pool of the possible number of random numbers is infinite and N is the finite size of the population. For random number generation in this thesis, population size, N is assumed to be 20000. Population size of 20000 is statistically assumed to be a representative of an infinite population [49].

The value of n_0 is calculated using the following formula:

$$n_0 = \frac{z^2 p(1-p)}{e^2} \tag{D.2}$$

where,

- z is the selected critical value for the desired confidence level
- p is the estimated proportion of an attribute present in the population
- e is the acceptable margin of error.

For the ambiguity scenario in terms of ambiguity resolution success indicator (ARSI) value, the outcome is binary. Thus, the estimated p value is taken as 0.5. The z value for a confidence interval of 99% is 2.58 [51]. 5% is commonly used as the acceptable margin of error. For these values, n_0 and n are calculated as follows

$$n_0 = \frac{(2.58)^2 \cdot (0.5) \cdot (1 - 0.5)}{(0.05)^2} = 665$$

$$n = \frac{665}{1 + \frac{665-1}{20000}} = 645$$

Thus, the simulations in the ambiguity scenarios are iterated for 650 times.