IMPROVING THE STATE ESTIMATION ACCURACY OF REAL-TIME VISION BASED MULTIPLE TARGET TRACKING ALGORITHMS WITH UNEQUAL DIMENSION INTERACTIVE MULTIPLE MODEL ESTIMATOR

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YAĞIZ KURT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

JUNE 2023

Approval of the thesis:

IMPROVING THE STATE ESTIMATION ACCURACY OF REAL-TIME VISION BASED MULTIPLE TARGET TRACKING ALGORITHMS WITH UNEQUAL DIMENSION INTERACTIVE MULTIPLE MODEL ESTIMATOR

submitted by **YAĞIZ KURT** in partial fulfillment of the requirements for the degree of **Master of Science in Aerospace Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Serkan Özgen
Head of Department, **Aerospace Engineering** _____

Assoc. Prof. Dr. Halil Ersin Söken
Supervisor, **Aerospace Engineering, METU** _____

Examining Committee Members:

Assist. Prof. Dr. Ali Türker Kutay
Aerospace Engineering, METU _____

Assoc. Prof. Dr. Halil Ersin Söken
Aerospace Engineering, METU _____

Prof. Dr. Umut Orguner
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Nazım Kemal Üre
Artificial Intelligence and Data Engineering, ITU _____

Assist. Prof. Dr. Gökhan Sosyal
Electrical and Electronics Engineering, Ankara University _____

Date:22.06.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Yağız Kurt

Signature :

ABSTRACT

IMPROVING THE STATE ESTIMATION ACCURACY OF REAL-TIME VISION BASED MULTIPLE TARGET TRACKING ALGORITHMS WITH UNEQUAL DIMENSION INTERACTIVE MULTIPLE MODEL ESTIMATOR

Kurt, Yağız
M.S., Department of Aerospace Engineering
Supervisor: Assoc. Prof. Dr. Halil Ersin Söken

JUNE 2023, 140 pages

This study presents methodology for improving the vision-based multiple object motion estimation performance for the maneuvering objects in case of occlusion. Present deep learning-based visual MOT (Multiple Object Tracking) algorithms cannot track the objects when an occlusion exists even for a short period of time. When the measurement is not available, the prediction model propagates the Kalman Filter's motion model, or the tracker maintains the previous estimation until a measurement is received. In either case, during these periods, the object may maneuver or accelerate/decelerate so that the tracker cannot assign the tracklets successfully when the measurement is arrived. Using Unequal dimension Interactive Multiple Models (UDIMM) approach, track loss due to the ID switch is highly reduced compared to the simple constant velocity motion model used in the StrongSort multi-object tracker algorithm. For the concerns on real time computation on onboard aerial vehicles, lightweight object detector and MOT algorithms, YOLOv5 and StrongSort are implemented, respectively. Estimation and performance results of different filters are compared on simulation environment.

Keywords: interactive multiple model filter, vision based multiple object tracking, estimation, deep learning

ÖZ

EŞİT OLMAYAN BOYUTLU ETKİLEŞİMLİ ÇOKLU MODEL KESTİRİM YÖNTEMİ İLE GERÇEK ZAMANLI GÖRÜNTÜ TABANLI ÇOKLU HEDEF TAKİP ALGORİTMALARININ DURUM TAHMİNİ HASSASİYETİNİN İYİLEŞTİRİLMESİ

Kurt, Yağız
Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Halil Ersin Söken

Haziran 2023 , 140 sayfa

Bu çalışma ile, blokaj altında manevra yapan nesnelerin görüntü tabanlı çoklu nesne takibi algoritmalarındaki hareket tahmini performanslarını artırmak için bir yöntem sunulmaktadır. Mevcut derin öğrenme tabanlı Çoklu Nesne Takibi (ing. MOT) algoritmaları, blokaj altında kısa bir süre dahi olsa nesneleri takip etmekte başarılı olamamaktadırlar. Ölçüm mevcut olmadığı durumda ölçüm alınana kadar Kalman Filtresi'nin hareket modeli zaman adımı kadar ötelenmektedir veya ölçüm kaybolmadan önceki tahmin korunmaktadır. Her iki durumda da, blokaj altındaki süre içerisinde nesnenin manevra yapma veya hızlanma/yavaşlama vb. öngörülemez davranış sergileme ihtimali bulunmaktadır. Bu sebeple, ölçüm ulaştığı anda takip algoritması, izleri (ing. tracklets) başarılı bir şekilde ölçüm değerleri ile eşleyememektedir. Eşit Olmayan Boyutlu Etkileşimli Çoklu Modeller yaklaşımını kullanarak, kimlik (ing. ID) değişikliğine bağlı takip kaybı, mevcut takip algoritmalarında kullanılan basit sabit hız hareket modelinin kullanıldığı duruma kıyasla önemli ölçüde azaltılmıştır.

Anahtar Kelimeler: etkileşimli çoklu model filtresi, görüntü tabanlı çok nesne takibi, kestirim, derin öğrenme



Ad somnis ultra astra..

ACKNOWLEDGMENTS

First and foremost, I am immensely grateful to my thesis advisor, Assoc.Prof.Dr. Halil Ersin Soken, for their invaluable guidance, expertise, and unwavering commitment to my academic growth. His insightful feedback, constructive criticism, and constant encouragement have played a pivotal role in shaping this research work. I express my gratitude to him for providing me with precise guidance that has steered my career path. I am thankful to the rest of thesis committee for their insightful questions and comments which encourage me to expand my thesis research from various perspectives.

My appreciation and gratitude extend to my family, whose unwavering support, love, and encouragement have been the bedrock of my journey. To my kindhearted father Şenol Kurt, my dear and beloved sister Öykü Kurt but especially my mother Makbule Pekel, I cannot thank you enough for your endless sacrifices, belief in my abilities, and the opportunities you have provided me. I would like to dedicate a special acknowledgment to my mother, whose unwavering love, support, and sacrifices have been the driving force behind my life. Her guidance and encouragement have shaped not only my academic pursuits but also my character and values. I am truly blessed to have my family as my pillars of support.

Last but not least, I would like to extend my acknowledgments to my dear luminary friend Selim Çam, my trustworthy friend Kutay Özkaratay, my best workmate Şevket Utku Aydınlı, my lodestar Dr. Burak Boyacıoğlu and my dear colleagues from ASELSAN Inc. and TUBITAK SAGE. The countless conversations, study sessions, and shared experiences have not only enriched my graduate experience but have also created lasting bonds that I will cherish for a lifetime.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ABBREVIATIONS xx

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Literature Survey 4

1.3 Contributions and Novelties 7

1.4 The Outline of the Thesis 8

2 MULTIPLE OBJECT DETECTION 9

2.1 Object Detection on Aerial Images 9

2.2 Convolutional Neural Networks (CNNs) 10

2.3 YOLO Series Detection Frameworks 16

2.3.1 YOLOv5, 2020 21

3 MULTIPLE OBJECT TRACKING (MOT) 25

3.1 Classification of the Object Tracking Methods 25

3.2 Simple Online and Real-time Tracking (SORT) 28

3.3 Deep Simple Online and Real-time Tracking (DeepSORT) 30

3.4 Strong Simple Online and Real-time Tracking (StrongSORT) 34

4 SURVEYS, DATASETS AND EVALUATION METRICS ON MULTI OBJECT DETECTION AND TRACKING 39

4.1 Surveys on Object Detection on Aerial Images 39

4.2 Aerial Image Datasets 40

4.2.1 UAVDT Dataset 41

4.2.2 VisDrone Dataset 41

4.2.3 DroneVehicle Dataset 42

4.2.4 DOTA Dataset 43

4.2.5 MOT17 Dataset 43

4.3 MOT Evaluation Metrics 44

4.3.1 MOTA 47

4.3.2 IDF1 48

4.3.3 HOTA 49

5 MOTION AND MEASUREMENT MODEL REPRESENTATIONS 53

5.1 General Representation of Motion Models 53

5.2 Constant Velocity Model 57

5.3 Constant Acceleration Model 57

5.4 Constant Turn Model 58

5.5 Measurement Model 59

6 ESTIMATION FILTERS 61

6.1 Kalman Filter 61

6.1.1 Bayes' Theorem 62

6.1.2 Derivation of the Kalman Filter 65

6.2 Interactive Multiple Model (IMM) Filter 72

6.3 Unequal Dimension Interactive Multiple Model (UDIMM) Filter . . . 75

7 SIMULATION ENVIRONMENT, MODEL FRAMEWORK AND PERFORMANCE PARAMETERS 77

7.1 Test Setups 77

7.1.1 Simulation Scenario 77

7.2 Model Framework 79

7.3 Simulation Performance Parameters 81

8 EVALUATION OF THE RESULTS 85

8.1 Scenario 1: Single Object with No Occlusion 89

8.2 Scenario 2: Single Object with Occlusion 98

8.3 Scenario 3: Multiple Object with Occlusion 109

8.4 Test on MOT17 Validation Dataset 125

9 CONCLUSION AND FUTURE WORK 127

9.1 Conclusion 127

9.2 Future Work 129

REFERENCES 131

LIST OF TABLES

TABLES

Table 1.1 Challanges of real-time visual target tracking on aerial images . . . 3

Table 2.1 Comparison of the components of various YOLO frameworks 19

Table 3.1 Comparison of two initialization method [1] 26

Table 3.2 Comparison of processing mode[2] 27

Table 4.1 A summary of different evaluation metrics for MOT 46

Table 6.1 One cycle of the IMM algorithm 74

Table 7.1 Ground-truth modes and values with durations 79

Table 8.1 Performance of Filters in Scenario 1 98

Table 8.2 Performance of Filters in Scenario 2 109

Table 8.3 Performance of Filters in Scenario 3 125

Table 8.4 Results of applying UDIMM estimator with DeepSORT and StronSORT algorithms ON MOT17 validation dataset 126

LIST OF FIGURES

FIGURES

Figure 1.1 Overview of detection and tracking algorithm interactions with the system 4

Figure 2.1 A classical CNN architecture [3] 11

Figure 2.2 Representation of image as a grid of pixels [4] 12

Figure 2.3 Illustration of a 2-D convolution operation without kernel flipping. The depicted boxes with arrows illustrate the process of generating the top-left element of the output tensor. This is accomplished by applying the kernel to the corresponding top-left region of the input tensor [5]. 13

Figure 2.4 Simple frameworks of one-stage(top) and two-satage(bottom) detectors 16

Figure 2.5 YOLO's development in choronological order 17

Figure 2.6 Single-Stage Detector Architecture [6] 21

Figure 2.7 The YOLOv5 network architecture is comprised of three main components: (1) the backbone, which is CSPDarknet and used for feature extraction, (2) the neck, which is PANet and used for feature fusion, and (3) the head, which is the YOLO Layer and outputs detection results such as class, score, location, and size. [7] 22

Figure 3.1 Overview of SORT algorithm [8] 30

Figure 3.2 Overview of DeepSORT algorithm [8] 31

Figure 3.3 Illustration of StrongSORT improvements over DeepSORT algorithm [9] 35

Figure 4.1 Ground truth annotation in a sample frame from MOT17 dataset 44

Figure 4.2 Comparison of MOT evaluation metrics [10] 46

Figure 4.3 This tracking example illustrates how the use of IDF1's single best trajectory matching approach can lead to unintuitive matches between trajectories [10]. 49

Figure 4.4 Decomposition of HOTA metric. Each sub-metric focuses on a specific type of tracking error, allowing for a more detailed analysis of the tracker's performance. [10]. 50

Figure 5.1 Object motion geometry in 2D plane 56

Figure 5.2 YOLOv5 measurements sample on image coordinates. This frame is also used in annotation format for training the neural network [11]. 60

Figure 6.1 One cycle of the IMM estimation algorithm. Interaction/mode mixing step is illustrated on the left side, filtering is shown on the middle section and final probability evaluations are represented on the right side of the algorithm scheme [12]. 73

Figure 7.1 Ground-truth trajectories of simulated scenario for the two moving objects 78

Figure 7.2 The Model Framework refers to a system that includes algorithms for multiple object detection and tracking, which interact with a two-dimensional simulation environment. 84

Figure 8.1 Simulation and ground truth modes of the scenario 1. 89

Figure 8.2 Position estimation results in x and y directions for CV model (left), CA model (right) in scenario 1. 91

Figure 8.3 IMM estimator (left), UDIMM estimator (right) in scenario 1. . . 92

Figure 8.4 2D trajectory change of the object with respect to time for CV model (upper-left), CA model (upper-right), IMM estimator (bottom-left), UDIMM estimator (bottom-right) in scenario 1. 93

Figure 8.5 Velocity estimation results in x and y directions for CV model (first row), CA model (second row) in scenario 1. 95

Figure 8.6 Velocity estimation results in x and y directions for IMM estimator (first-row), UDIMM estimator (second-row) in scenario 1. 96

Figure 8.7 Mode probability results of IMM (left) and UDIMM (right) estimators for scenario 1. 97

Figure 8.8 Mean Errors of Total Position (left) and Total Velocity (right) for scenario 1. 97

Figure 8.9 Simulation and ground truth modes of the scenario 2. 99

Figure 8.10 Position estimation results in x and y directions for CV model (left), CA model (right) in scenario 2. 101

Figure 8.11 IMM estimator (left), UDIMM estimator (right) in scenario 2. . . 102

Figure 8.12 2D trajectory change of the object with respect to time for CV model (upper-left), CA model(upper-right), IMM estimator (bottom-left), UDIMM estimator (bottom-right) in scenario 2. 103

Figure 8.13 Velocity estimation results in x and y directions for CV model (first row), CA model (second row) in scenario 2. 106

Figure 8.14 Velocity estimation results in x and y directions for IMM estimator (first row), UDIMM estimator (second row) in scenario 2. 107

Figure 8.15 Mode probability results of IMM (left) and UDIMM (right) estimators for scenario 2. 107

Figure 8.16 Mean Errors of Total Position (left) and Total Velocity (right) for scenario 2. 108

Figure 8.17 Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing CV model in scenario 3. . . . 110

Figure 8.18 Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing CV model in scenario 3. 111

Figure 8.19 Velocity estimation results in x and y directions for CAR 1 (top-row), CAR 2 (bottom-row) utilizing CV model in scenario 3. 112

Figure 8.20 Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing CA model in scenario 3. . . . 113

Figure 8.21 Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing CA model in scenario 3. 114

Figure 8.22 Velocity estimation results in x and y directions for CAR 1 (first-row), CAR 2 (second-row) utilizing CA model in scenario 3. 115

Figure 8.23 Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing IMM estimator in scenario 3. . 116

Figure 8.24 Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing IMM estimator in scenario 3. 117

Figure 8.25 Velocity estimation results in x and y directions for CAR 1 (first-row), CAR 2 (second-row) utilizing IMM estimator in scenario 3. . . . 118

Figure 8.26 Mode probabilities changes of IMM filter for CAR 1 (left), CAR 2 (right) in scenario 3. 118

Figure 8.27 Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing UDIMM estimator in scenario 3. 119

Figure 8.28 Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing UDIMM estimator in scenario 3. 120

Figure 8.29 Velocity estimation results in x and y directions for CAR 1 (first-row), CAR 2 (second-row) utilizing UDIMM estimator in scenario 3. . . 121

Figure 8.30 Mode probabilities changes of UDIMM filter for CAR 1 (left), CAR 2 (right) in scenario 3. 122

Figure 8.31 Mean Errors of Total Position and Total Velocity for CAR 1 (first-row) and CAR 2 (second-row) in scenario 3. 123

Figure 8.32 Sample MOT 17 footages with annotated ground truths 126

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
MOT	Multiple Object Tracking
MTT	Multi-Target Tracking
YOLO	You Only Look Once
YOLOv3	You Only Look Once version 3
YOLOv5	You Only Look Once version 5
YOLOX	You Only Look Once version X
KF	Kalman Filter
CBAM	Convolutional Block Attention Model
R-CNN	Region-based Convolutional Neural Networks
SSD	Single Shot Detector
CNN	Convolutional Neural Networks
RGB	Red Green Blue
MMDetection	Open source object detection toolbox
UAVDT	Unmanned Aerial Vehicles Detection Tracking
ReLU	Rectified Linear Unit
CSPNet	Cross Stage Partial Network
PANet	Path Aggregation Network
FPN	Feature Pyramid Network
SiLU	Sigmoid-Weighted Linear Units
MOTA	Multi-Object Tracking Accuracy
MOTP	Multi-Object Tracking Precision
HOTA	Higher-Order Tracking Accuracy

IDF1	The Identification Metrics
DBT	Detection-Based Tracking
DFT	Detection-Free Tracking
PDF	Probabilistic Density Function
NEES	Normalized Estimation Error Squared
OBB	Oriented Bounding Box
HBB	Horizontal Bounding Box
RMSE	Root Mean Square Error

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The variance of applications on MOT keep increasing by involving learning algorithms. This article concentrates on enhancing the accuracy and real-time processing capabilities of the detection-tracking task on quadcopter onboard computers. In this regard, we aimed to focus on motion estimation module that has a significant impact on performance while having minimal impact on computational effort.

MOT is essential in computer vision and machine learning because it allows us to track the motion of multiple objects over a duration in a video sequence. This holds particular significance for aerial imagery, as it possesses the ability to capture extensive regions and offer a top-down perspective of a scene. By tracking multiple objects in such images, we can analyze their interactions and predict their future behavior, which can be critical for decision-making and situational awareness.

One of the main advantages of multi object tracking in aerial images is its ability to detect and track multiple objects simultaneously. This can be useful in various domains, such as surveillance, disaster response, and military applications, where it is important to monitor multiple moving objects in real-time. For instance, in disaster response scenarios, multi-object tracking can be used to monitor the movement of rescue vehicles and personnel, as well as other critical resources such as food and water supplies.

Furthermore, multi object tracking can be employed to detect abnormalities and potential hazards within aerial images. By analyzing the movement patterns of objects,

we can detect any unusual or suspicious behavior, such as vehicles deviating from their usual routes or individuals moving in a restricted area. This information can be used to alert authorities and take preventive measures, such as dispatching law enforcement or blocking off an area.

Multi-object tracking in aerial images can also be used for urban planning and traffic management. By tracking the movement of vehicles, we can identify congested areas and optimize traffic flow. This can help reduce traffic congestion, improve road safety, and minimize environmental pollution.

From the algorithm point of view, visual tracking provides key information for estimating the trajectory of detected objects which is an essential for motion planning. Object motion estimation, computing a similarity/distance score between pairs of detections, data/measurement association, and appearance model networks design are a few of the study areas that are involved in tracking multiple objects [13]. The main focus of our research was to improve the accuracy in estimating the motion of objects. Vision based identification and tracking process mainly consists of two tasks: "Object Detection" and "Multi Object Tracking". The objective of object detection is to identify the objects along with their belonging class within a rectangular bounding box. MOT, also called MTT entails the analysis of videos to assign an individual identification (referred to as a target ID) to each bounding box or detection, enabling the differentiation of objects [13].

The task of detecting objects in aerial images presents several challenges, including arbitrary orientations, variations in scale, highly uneven object densities, and significant differences in aspect ratios (ARs) [14]. Besides, real time object tracking enlarge the scope of the detection and tracking challenges. Some of these challanges and the related consequences are summarized in Table 1.1. The challenges marked with the asteriks(*) are analyzed in the scope of this research. In that sense, any error or disturbance during the tracking process which might occur due to measurements acquired from object detector (YOLOv5) is tried to be eliminated in our simulations.

As mentioned previously, tracking multiple objects involves various research fields such as object motion estimation, computing a score measuring similarity or distance is computed between pairs of detections, data/measurement association, and design

Table 1.1: Challanges of real-time visual target tracking on aerial images

Problems		**Consequences**
Object Occlusion*	$->$	Loss of tracking
Moving Platform	$->$	Object scale varies violently
False Positive Detections	$->$	Misleading the estimation
High Speed Flight	$->$	Motion Blur and Object distinction
Nonlinear Motion During Maneuver*	$->$	ID Switch and Track Loss of a Target
High Computation Power	$->$	Low FPS on onboard GPUs
Dynamic Disturbances(vibration, gust etc.)	$->$	Loss of tracking
Limited Number of Public Aerial Image Datasets	$->$	Low Detection Accuracy

*Research interests in this thesis

of appearance model networks [13]. The researchers made an effort to address the challenges by suggesting possible solutions in these research fields. In our work, we focused on increasing the performance of object motion estimation using multiple model methods in challenging scenarios.

Typically, the prevailing motion models employed in MOT assume linear object motion within brief time intervals and necessitate uninterrupted observation. As a result, these methods are susceptible to non-linear object motion, occlusion, and rely on high frame rate videos [9]. The object which is moving and rotating car in our case may behave stochastically. Therefore, using constant velocity motion models as a prediction model in Kalman Filters [15] introduces large variance which cause track loss when no new object observations (from detection) match existing tracklets due to occlusion or nonlinear motion. To tackle these problems, we argue that using multiple models can reduce the prediction error and provide sustainable tracking when occlusion exists even in maneuvering conditions. To reach the goal, UDIMM filter is implemented.

Blom proposed IMM estimator in 1984 [16] and developed by introducing Markovian switching coefficients in 1988 [17]. The IMM filter is renowned for its ability to achieve a favorable trade-off between estimation accuracy and computational complexity. However, mixing the estimation results of unequal dimension models requires special attention and there is no specific or deterministic way to solve this problem.

But, various methodologies are proposed [18, 19, 20, 12] to deal with mixing the unmatched state variables. Two-dimensional environment, where we can simulate

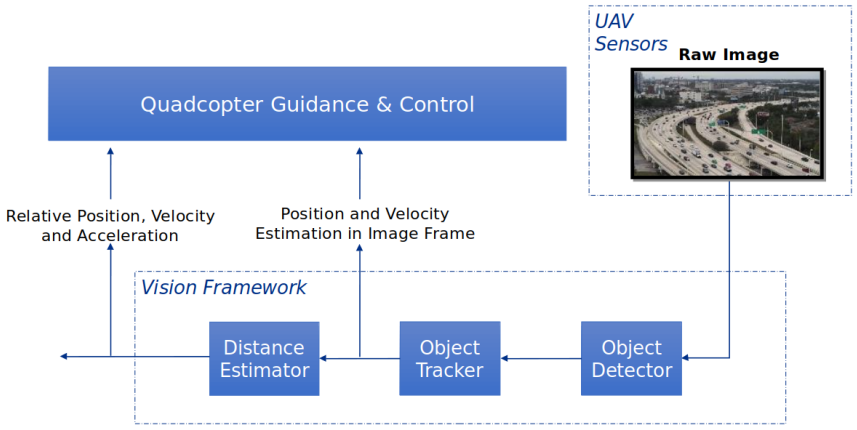


Figure 1.1: Overview of detection and tracking algorithm interactions with the system

the motion and generate the ground-truth values of moving vehicles, is created using *Pygame* library. Sensor data which is a video stream in our case directly comes from the simulation as an input to the object detector, YOLOv5. Then, multi object tracking algorithm (StrongSort) by using various type of Kalman Filters is applied to track two moving ground vehicle simulated in our environment. Figure 1.1 illustrates the usage of the *"Object Detection and Object Tracking"* algorithms in overall system and input-output relations with other system modules. UDIMM filter performs high quality results especially for tracking maneuvering targets in occluded scenarios.

1.2 Literature Survey

There are various researches ongoing to develop vision based MOT. Since there exists many challenging tasks, significant number of algorithms are recently proposed claiming they solved some of the problems. In such a rapidly evolving field of study, surveys about multiple object tracking clear up recent advanced works and points out the challenges needed to be solved. [13] elaborates the main stages of MOT algorithms with the explanation of *deep learning* implementation, additionally compares the performance of MOT algorithms on the three MOTChallenge dataset and also

presents some possible future research directions. [21] gives the broad perspective to tracking algorithms with the challenges and type of categories. [22] is focused on taxonomy of visual multiple target tracking and elaborates the general research areas related to MOT algorithms. [23] is one of the most recent literature review about MOT, they mainly focused on the algorithms trying to solve abrupt appearance changes and severe object occlusion problems. [24] investigates the functionality of deep networks in top-ranked MOT algorithms.

MOT methods are generally analyzed in two types: "Seperate Trackers" and "Joint Trackers". Seperate trackers identify/localize the target first and then try to associate them with the estimation results coming from both appearance and motion model. Because of the nature of its process, they are called tracking-by detection. Since tracking-by-dection requires high computational power due to heavy components (two neural network), several recent methods proposed unified frameworks joining detection, motion, embedding, and association models all together. Recent findings indicate that joint tracker methods demonstrate similar or superior tracking accuracy when compared to tracking-by-detection approaches, while requiring fewer computational resources. But, there exits some problems with joint trackers such as limited data for training these unified components and also there is a competition between components which decreases the stability and reliability of the algorithms in complex scenarios. Tracking-by-detection type MOT algorithms [25, 26, 27, 28, 29] recently get started to be preferred in *real time* applications due to recent progress in lightweight (single stage) object detection performances. A widely used and classic motion model-based MOT algorithm is Simple Online and Real-time Tracking (SORT) [25] published in 2016, which uses a KF [15] to estimate object motion. The authors of DeepSORT [28] recommended incorporating appearance information into the motion estimator to enable tracking of objects for longer periods, even when they are occluded, resulting in fewer instances of identity switches. However, when we look at the present, DeepSORT shows very low performance compared to state-of-the-art methods because of its outdated techniques. In 2022, StrongSORT [9] implemented appearance-free link model (AFLink) to effectively connect short tracklets and form complete trajectories and Gaussian-smoothed interpolation (GSI) algorithm to compensate for missing detections on top of the classic Deep-

SORT tracking algorithm. With the additional improvements, StrongSORT achieved the best performance in MOT17 challanges in terms of Higher Order Tracking Accuracy (HOTA) [10] and Identification F_1 (IDF1) [30] metrics.

To expand on the motion models employed in MOT methods, linear motion models used in KFs limit the performance of tracking algorithms especially in occluded and maneuvering scenarios. To handle non-linear motion, Extended KF [31] and Unscented KF [32] are applied, but they increase the complexity and unstability of the system, and still rely on approximating the Gaussian prior whereas our work shows that probability distribution of measurement errors are not Gaussian. To be able to estimate state variables of the moving target accurately, motion mode of the targets should be known correctly so that we can propagate the motion model of the KF when measurement is unavailable. However, there is no deterministic way of knowing targets next motion mode since the target can behave in many different ways. In the literature, there exist multiple model filters which are proposed to deliver a solution to this problem. The Interacting Multiple Model (IMM) filters have demonstrated their ability to deliver highly effective and accurate state estimation in systems characterized by significant variations in behavior patterns over time. The primary factor behind the enhanced performance of the IMM filter [16, 17] in contrast to the single-model approach lies in its mode-mixing capability, which consistently coordinates the various models [12]. A detail overview of Interactive Multiple Model methods in target tracking is provided in [33]. Additionally, [34] presents a collection of multiple model (MM) filters and compares their performances. Innovative interaction approach for UDIMM filters [18] used in vehicle motion estimation is proposed in [12] by reducing complexity of mode mixing step. [20] presents novel approaches for unequal dimension track to track fusion (T2TF) using Covariance Intersection (CI) to solve the special heterogeneous track-to-track fusion (HT2TF) case which basically covers the fusion of tracks with different state spaces. [18] put forward a systematic methodology for mode mixing, particularly in cases where the modes possess unequal states dimensions. In the literature, there are also similar concept studies to our work which are applied in real life scenarios about moving target tracking. In [35], moving target simulated in Gazebo environment is detected and tracked from a UAV to be able to perform landing on a moving vehicle. They relied on deep

learning-based object detection (Single Shot Detector [36]), stereo image processing algorithm, and kernelized correlation filtering algorithms. [37] use the Camshift algorithm with the Kalman Filter and tried to track linear moving target when occlusion exists. [38] is the demonstration of tracking human target moving randomly in groves. Images are obtained by the onboard camera and SSD [36] algorithm is used to detect the candidate human targets. In order to select the correct target from these candidates, histogram of oriented gradients (HOG) [39] and color histogram features are extracted from the subimages, and then Support Vector Machine (SVM) [40] based classifier is adopted to determine the target. After tracking object initialization, a Tracking-Learning-Detection and Kernelized Correlation Filter (TLD-KCF) [38, 41] approach is presented to perform tracking the moving objects.

There are various approaches studied for vision based moving target tracking. Each of them has advantages within the scenario they used. In our work, we proposed the usage of UDIMM filter for vision based maneuvering target tracking in scenarios where the maneuvering motion behaviour and dense occlusion exists.

1.3 Contributions and Novelties

Our contributions are as follows:

- Introducing multiple model filters to vision based tracking algorithms which can be applied for navigating the robots in indoor environments or in a mission for guiding ground vehicles in real-time from unmanned air vehicles (UAVs).

- Improving state estimation results of maneuvering objects under occluded scenarios when measurements are not available.

- Reducing the ID switch and improving tracking performance of maneuvering objects under dense occluded scenarios.

1.4 The Outline of the Thesis

Chapter 1 provides an overview of the thesis, encompassing the problem definition and motivation, literature review, contributions, and an outline of the thesis structure.

Chapter 2, discusses the multiple object detection on aerial images, gives brief intoroduction about convolutional neural networks and explains YOLO series detection frameworks in detail.

Chapter 3, presents variations of multiple object tracking methods and gives detail explanation of tracking-by-detection type MOT algorithms particularly for advancement stages of SORT algorithm.

Chapter 4, provides researches about surveys, datasets and evaluation metrics on multi object detection and tracking.

Chapter 5, introduces motion and measurement model representations used in the estimation stage of tracking algorithms. The rationale and underlying principles behind the derivations are elucidated, detailing the assumptions and motivations involved.

Chapter 6, elucidates the estimation filters, beginning with the linear Kalman filters, progressing to the interactive multiple model filters, and concluding with the unequal dimension interactive multiple model filters.

Chapter 7, entail an explanation of the simulation environment, model framework, and performance parameters employed in comparing the simulation results.

Chapter 8, evaluates the simulation results and benchmark results on MOT17 dataset.

Chapter 9, concludes the thesis, providing additional discussions and proposing future studies by identifying open problems that need to be addressed.

CHAPTER 2

MULTIPLE OBJECT DETECTION

2.1 Object Detection on Aerial Images

The utilization of autonomous unmanned aerial vehicles (UAVs) is gaining widespread recognition across industries due to their adaptability and rapid deployment capabilities. These UAVs have proven to be valuable assets in various civil and military domains, including applications such as aerial photography for mapping and agriculture, search and rescue missions, and surveillance in urban or sensitive military areas.

There exists a gap for surveillance use cases of UAVs in terms of reliability of algorithms and autonomy level. There are still lots of research proceeding in this field. But UAVs used in surveillance are not still fully adapted to our daily lives since autonomy level is not sufficient and reliability of algorithms are not well verified. But their development keeps increasing by involving learning algorithms with the improvement in computer power. In the future, it is obvious that we are going to encounter surveillance UAVs more often in our daily lives, for example they might be used widely in suspicious and denouncement situations or in criminal cases. Also, surveillance drones play critical role for observation in military operations. They are widely used in observing non-safe areas by providing real time vision with detected potential objects. In both use cases of quad rotor UAVs, taking fast action and planning, precise tracking and accurate detection of object are required and play vital roles for success of the mission.

It is usual to detect relatively large moving objects on a fixed frame, but there are challenges on detecting small objects from real time footage taken by quadrotor UAVs.

These challenges include both software and hardware issues. Detection algorithms must run on the onboard due to latency occurrence in communication with the ground station. For this reason, there is a computation limit due to weight and dimension concerns for small UAVs. Furthermore, drones often operate at different altitudes, resulting in significant variations in object scales. This poses a challenge for network optimization. Additionally, when drones fly at high speeds and low altitudes, motion blur can occur, particularly in densely populated areas, further complicating the task of distinguishing objects [42]. In addition to these, there are lack of public datasets for aerial images to be able to train the quadrotor UAVs. Also, there are some additional limitations of detecting objects from aerial imagery such as varying disturbances (wind, gust etc.), vibration and moisture effecting the camera lens.

Furthermore, detecting objects within high-speed trajectories presents inherent difficulties due to sudden object movements, occlusions, changes in object scale, variations in appearance caused by illumination and viewing angle changes, non-rigid deformations, variability in the shape and posture within the same object category, as well as camera movements [35]. Moreover, when the camera experiences tilts and jerks in conjunction with UAV motion, maintaining a fixed position on a moving object, such as a vehicle, becomes challenging, potentially resulting in the object being lost from the camera's field of view. Additionally, contemporary state-of-the-art detection techniques relying on deep learning demand substantial computational power and dedicated hardware to operate at an acceptable speed [35].

2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are widely used neural networks in computer vision tasks such as image classification, object detection, and segmentation. Their design is focused on automatically extracting significant features from input data using convolutional operations. In recent years, CNNs have exhibited impressive performance, outperforming traditional methods that relied on manually engineered features. Consequently, they have become the prevailing state-of-the-art approach in diverse computer vision applications.

In a CNN, the fundamental operation is convolution, which involves applying a filter to an input image to generate a feature map. Convolution is a mathematical process where a small matrix of weights, known as a filter, is moved across the input image. At each location, the filter performs element-wise multiplication and summation to produce the resulting feature map. The result is a new image that captures local patterns in the input image. The use of multiple layers in CNNs allows for the learning of increasingly complex features and representations of the input data, resulting in significantly improved performance on visual tasks [43].

The composition of a convolutional neural network involves various layers, such as convolutional layers, fully connected layers and pooling layers. Convolutional layers are the most important layers in a CNN and are responsible for learning the features of the input data. A convolutional layer comprises a collection of adaptable filters, known as kernels, which convolve with the input data to generate a feature map. The output of a convolutional layer is fed into a pooling layer, which reduces the spatial dimensionality of the feature maps while retaining the most important information.Subsequently, the output from the pooling layer is inputted into a fully connected layer that maps the extracted features to the respective output classes. The classical CNN architecture can be seen in Figure 2.1.

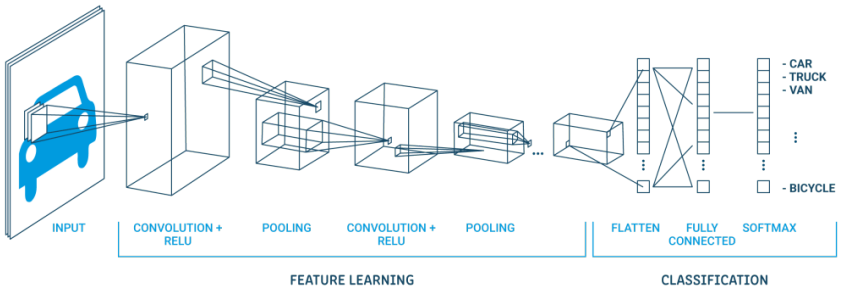


Figure 2.1: A classical CNN architecture [3]

There are several state-of-the-art convolutional neural networks that have been developed for image classification and object detection tasks, such as LeNet-5, AlexNet, VGGNet, GoogLeNet, ResNet, and InceptionNet. Each of these networks has a unique architecture and has been trained on large-scale datasets to achieve high accuracy on image recognition tasks.

The working of a convolutional neural network can be explained using the following steps:

1. Input data: The input data is typically an image, which is represented as a matrix of pixel values. An image is composed of pixels that are organized in a grid-like pattern and have pixel values indicating their brightness and color which is illustrated in Figure 2.2.

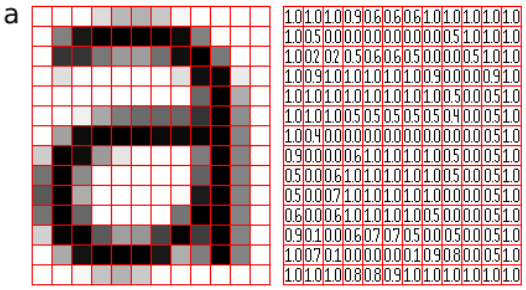


Figure 2.2: Representation of image as a grid of pixels [4]

2. Convolution: The input data is convolved with a set of learnable filters to extract relevant features from the image [5]. Mathematically, the convolution operation in two dimension can be represented as:

$$h_{i,j} = (f * g)_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} f_{m,n} g_{i-m,j-n} \tag{2.1}$$

where h is the output **feature map**, f is the input data which is an **image** as a matrix, g is the learnable filter which is also called **Kernel**, and M and N are the dimensions of the filter [5].

In the case of convolutional neural networks, the input signal is typically a 2D image represented as a matrix and the filter is a smaller matrix that is slid over the input image, performing the convolution operation at each location as can be seen in Figure 2.3. This process results in a feature map that highlights certain features in the image, which can then be used for tasks such as object recognition, segmentation, and classification.

3. Activation: Activation functions serve as mathematical functions in artificial neural networks, enabling the introduction of non-linearity to the neuron's out-

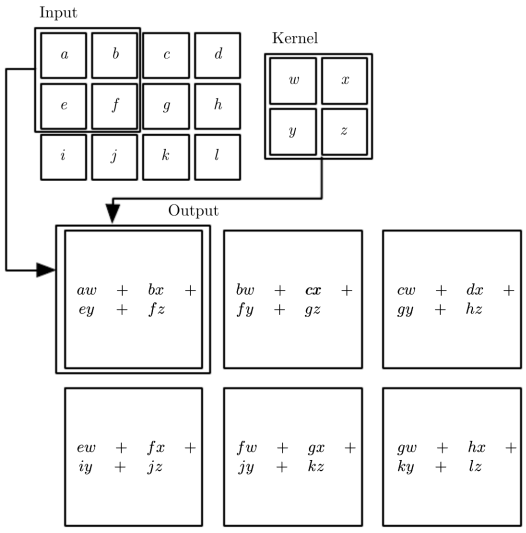


Figure 2.3: Illustration of a 2-D convolution operation without kernel flipping. The depicted boxes with arrows illustrate the process of generating the top-left element of the output tensor. This is accomplished by applying the kernel to the corresponding top-left region of the input tensor [5].

put. Their significance lies in shaping the output of individual neurons and, consequently, the overall output of the neural network [44].

One of the most commonly used activation functions is the Rectified Linear Unit (ReLU), defined as:

$$f(x) = \max(0, x) \tag{2.2}$$

where x is the input to the neuron. The **ReLU** activation function is widely used in deep neural networks due to its computational efficiency and good performance.

Another commonly used activation function is the **Sigmoid** function, which is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

The Sigmoid function outputs a value between 0 and 1, making it useful in binary classification problems. Nevertheless, it is susceptible to the vanishing gradient problem, which can impede the training progress of deep neural networks.

The Hyperbolic Tangent ($tanh$) function is similar to the Sigmoid function but outputs a value between -1 and 1, making it useful in classification problems that have outputs in this range. The $tanh$ function is defined as:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

Another activation function that has gained popularity in recent years is the **Swish** function, defined as:

$$f(x) = x\frac{1}{1 + e^{-x}} \tag{2.5}$$

The Swish function is similar to the ReLU function but introduces a smoothness to the output.

Neural networks offer various alternative activation functions, including the Softmax function, Leaky ReLU, and Parametric ReLU. The selection of an appropriate activation function relies on the specific problem at hand and the underlying architecture of the neural network.

4. Pooling: The pooling operation is employed in convolutional neural networks for computer vision tasks to decrease the spatial resolution of the feature map while retaining crucial information. This process involves sliding a fixed-sized window across the input feature map and selecting the maximum or average value within that window as the output for that region. By implementing this operation, the network becomes more resilient to minor spatial translations in the input, aiding in its robustness [43].

   Formally, given an input feature map X of size $H \times W \times C$, where H, W, and C represent the height, width, and number of channels, respectively, max pooling is defined as follows [5]:

$$Y_{i,j,k} = \max_{m=0}^{r-1} \max_{n=0}^{r-1} X_{r.i+m,r.j+n,k} \tag{2.6}$$

   where Y is the output feature map of size $H' \times W' \times C$, and r is the pooling size, i.e., the size of the pooling window. In other words, the output feature map Y is obtained by dividing the input feature map X into non-overlapping regions of size $r \times r$ and taking the maximum value within each region.

Other pooling operations, such as average pooling, can also be used in neural networks, but max pooling is generally preferred due to its ability to preserve the most prominent features while reducing the size of the feature map.

5. Fully connected layer: Fully connected layers, also referred to as dense layers, are a prevalent type of layer employed in CNNs for tasks such as image classification. These layers establish connections between every neuron in the current layer and every neuron in the subsequent layer. This arrangement enables the network to learn intricate nonlinear relationships between features.

   Typically, the output of a convolutional layer is transformed into a flattened one-dimensional vector, which is subsequently inputted into a fully connected layer. The output of the fully connected layer commonly undergoes activation via a function like ReLU before being forwarded to the next layer [45].

   In mathematical terms, the output of a fully connected layer can be expressed as follows:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.7}$$

   where $\mathbf{x}$ is the input vector, $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, and $f(\cdot)$ is the activation function. The weight matrix $\mathbf{W}$ determines the strength of the connections between neurons in the current layer and neurons in the next layer, while the bias vector $\mathbf{b}$ allows for the output of the layer to be shifted [45].

   Fully connected layers are commonly used in CNNs for the final classification layer, where the output is a probability distribution over the different classes. However, they can also be used in intermediate layers for feature extraction.

The development of CNNs has been a major breakthrough in the field of deep learning, and has led to significant advances in a variety of applications. The use of CNNs has also spurred the development of new architectures and techniques, such as residual networks, attention mechanisms, and transfer learning, which have further improved the performance of deep learning models on visual tasks.

2.3 YOLO Series Detection Frameworks

Object detection is an essential task in the field of computer vision and has garnered significant research attention over the years. Deep learning-based object detectors can be broadly categorized into two groups: one-stage detectors and two-stage detectors.

Single-stage object detectors are capable of predicting object bounding boxes directly without requiring a separate region proposal step (unlike two stage detectors), which results in faster inference times, making them suitable for real-time applications. However, due to their emphasis on speed, they may not perform as well as multi-stage detectors in recognizing irregularly shaped objects or detecting small objects within a group. Nonetheless, the single-stage approach has the benefit of being structurally simpler and faster than the multi-stage detectors. Simple illustration of the difference in one-stage and two-stage detectors can be seen in Figure 2.4. This section will discuss the progress of the YOLO series of object detectors, which belong to the category of one-stage object detectors, and their evolutionary progress which is shown in 2.5.

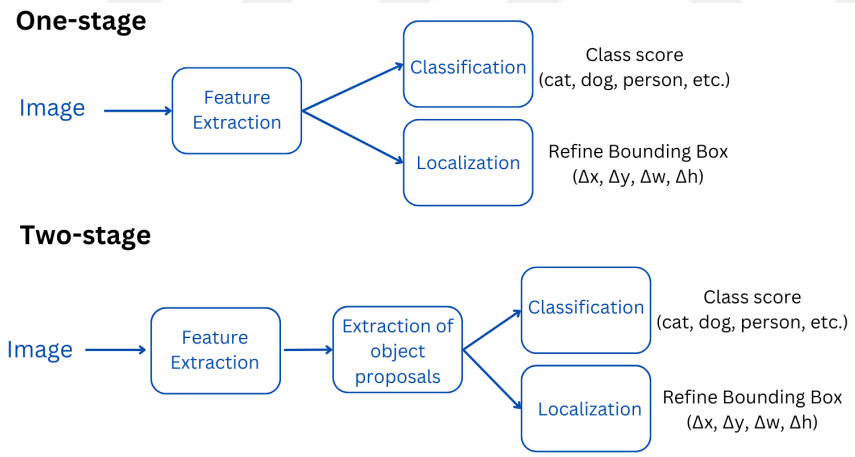


Figure 2.4: Simple frameworks of one-stage(top) and two-satage(bottom) detectors

YOLO (You Only Look Once) is a collection of object detection frameworks that leverage deep neural networks to achieve real-time object detection. The YOLO series has garnered substantial interest in the field of computer vision due to its exceptional speed and accuracy. As a result, it holds great value for a wide range of

applications including autonomous vehicles, surveillance systems, and robotics [46].

The YOLO algorithms employ a grid-based approach where the input image is divided into a grid of cells. These algorithms then predict the probability of each cell containing an object and provide the corresponding bounding box coordinates for that object [46]. This is achieved by utilizing a single neural network to process the entire image, unlike other object detection methods that require a separate region proposal step. The YOLO series has evolved with time, with each new version introducing improvements to the network architecture and training techniques.

The first YOLO model was proposed by Redmon et al. in 2016. It introduced a unified architecture for object detection and classification by treating the problem as a regression problem. YOLO achieves real-time performance, but with lower accuracy compared to other methods at that time.

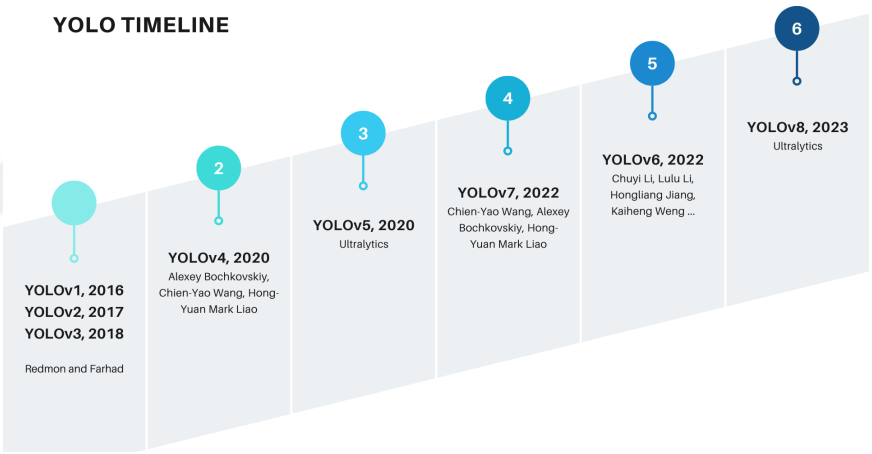


Figure 2.5: YOLO's development in choronological order

In 2017, Redmon and Farhadi introduced YOLOv2 [47]. It improved the first model by adding features such as batch normalization, anchor boxes, and high-resolution classification. Anchor boxes are used to improve the bounding box predictions, while batch normalization improves the model's stability and speed. The authors also introduced the concept of multi-scale training to further improve the model's accuracy.

In 2018, Redmon and Farhadi proposed YOLOv3 [48], which is the most popular and widely used YOLO model. It improved the previous models by introducing a

number of key features such as feature pyramid network, Darknet-53 backbone, and prediction across multiple scales. The feature pyramid network enables the model to detect objects across various scales, while Darknet-53 is a deeper and more powerful backbone network compared to the previous models. YOLOv3 also introduced a technique called "swish" activation, which improved the model's performance.

YOLOv4 [6], released in 2020, introduced new network architecture and training techniques, further improving the performance of the algorithm. It is an important development in the field of computer vision because it represents a significant improvement over its predecessors in terms of detection accuracy and speed. YOLOv4 achieves state-of-the-art performance by incorporating a number of advanced techniques, such as a new backbone network architecture, spatial pyramid pooling, feature fusion, and Mish activation function. It also employs a bag-of-freebies and bag-of-specials training method to improve the overall training process. YOLOv4 has several practical applications, such as real-time surveillance systems, autonomous vehicles, robotics, and human-computer interaction. Its accuracy and speed make it particularly useful for tasks that require real-time object detection and tracking.

YOLOv5 [11] comprises a collection of open-source object detection models and techniques built upon the YOLO model. These models have been pre-trained using the COCO dataset. Glenn Jocher introduced YOLOv5, incorporating several enhancements and distinctions. Notably, Glenn, the creator of mosaic augmentation, contributed this technique to enhance the performance of YOLOv4, which is now included in YOLOv5. It is maintained by Ultralytics and represents their research into the future of computer vision. However, YOLOv5 has not yet been formally discussed in a research paper. An accuracy review is needed to compare YOLOv5's efficiency with other algorithms, but there are not many articles available on YOLOv5's performance. An unofficial article written by Roboflow was relied upon by many people for information on YOLOv5, but it was not a published paper. However, the user-friendly implementation, advanced device support, and comprehensive application manual of YOLOv5 have made it a popular choice for practical applications. The release of YOLOv5 includes five different models sizes: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. The subsequent section of this research delves into a thorough examination of YOLOv5, as it is the primary subject under investigation.

Table 2.1: Comparison of the components of various YOLO frameworks

Layers	YOLOv3	YOLOv4	YOLOv5	YOLOv7	YOLOv6
Backbone	Darknet53	CSPDarknet53	CSPDarknet53	EELAN	RepVGG and CSPRepStack
Neck	FPN	SPP PANet	PANet	PANet	RepPAN
Head	B x (5+C) output layer B: Number of bounding boxes C: Class Score	B x (5+C) output layer B: Number of bounding boxes C: Class Score	B x (5+C) output layer B: Number of bounding boxes C: Class Score	Lead Head for the final output, Auxiliary Head for the middle layer outputs	Decoupled Classification and Detection Head
Loss Function	Binary Cross Entrophy	Binary Cross Entrophy	Binary Cross Entropy and Logit Loss Function	BCE with Focal Loss for Classification and IoU loss for Detection	Varifocal Loss for Classification and Distribution Focal Loss for Detection

In this research, YOLOX [49] is briefly introduced, which is utilized by the authors of StrongSORT [9] to extract the appearance features necessary for tracker evaluation on the MOT17 dataset. YOLOX is a computer vision model used for real-time object detection, and it was introduced in 2021. The model is based on the YOLOv3 SPP model, which uses the Darknet53 backbone as its baseline. YOLOX uses an anchor-free approach, which sets it apart from previous YOLO models. It also incorporates advanced detection techniques such as Decoupled Head and simOTA label assignment strategy. In addition, it employs strong data augmentation techniques like MOSAIC and mixUP to enhance its robustness during training.

In July 2022, WongKinYiu and AlexeyAB released YOLOv7 [50]. This version of YOLO demonstrates top-notch performance and is trained to detect the 80 generic classes in the MS COCO dataset, enabling real-time object detection. The methods proposed in YOLOv7 have a different development direction compared to current real-time object detectors that primarily concentrate on improving inference speed on various GPUs and CPUs. Along with optimizing the architecture, YOLOv7 also emphasizes enhancing the training process through optimized modules and methods. These modules and methods enhance the training cost to enhance the accuracy of object detection without increasing the inference cost. These modules and methods are collectively called "trainable bag-of-freebies". YOLOv7 offers multiple fundamental models, including YOLOv7, YOLOv7-tiny, and YOLOv7-W6. YOLOv7 serves as the standard model tailored for conventional GPU computing. On the other hand, YOLOv7-tiny is a lightweight model optimized for edge AI and deep learning tasks, specifically designed to run efficiently on mobile devices or distributed edge servers. This model is crucial for real-world computer vision applications. Compared to other versions, YOLOv7-tiny uses leaky ReLU as the activation function. YOLOv7-W6 is

a model specifically optimized for cloud GPU computing, catering to extensive AI and deep learning workloads in the cloud environment. This variant eliminates the need for GPUs on user devices. Additionally, there are other variations available, such as YOLOv7-X, YOLOv7-E6, and YOLOv7-D6. These models were obtained by applying a compound scaling method to increase the depth and width of the model, thereby enhancing its capabilities.

The primary objective of YOLOv6 [51] is to expand its application to industrial settings, which requires excellent performance across various hardware and real-world situations. In order to achieve optimal results for a diverse range of industrial applications, speed and accuracy are critical factors. As a result, YOLOv6 offers five different models, ranging from the fastest (YOLOv6-N) to the most accurate and largest (YOLOv6-L), to cater to different requirements and experimentation. The YOLOv6 model adopts several innovative approaches to achieve state-of-the-art results, which can be summarized into four main points. Firstly, it employs anchor-free object detection, which provides better generalizability and saves time in post-processing. Secondly, YOLOv6 features a revised reparameterized backbone and neck for the model architecture. Thirdly, the model uses Varifocal loss (VFL) for classification and Distribution Focal loss (DFL) for detection. Finally, YOLOv6 includes several improvements that make it more suitable for real-time industrial applications, such as longer training epochs, quantization, and knowledge distillation [51].

YOLOv8, the most recent version of the YOLO model [52], is a state-of-the-art model capable of performing tasks such as object detection, image classification, and instance segmentation. YOLOv8n is the smallest model with a 37.3 mAP score on COCO, while YOLOv8x is the largest model with a higher accuracy rate of 53.9 mAP on COCO. It was developed by Ultralytics, the creators of the influential YOLOv5 model, and features several architectural and developer experience improvements over its predecessor. As of writing this research, Ultralytics is actively developing YOLOv8, adding new features and taking feedback from the community into consideration. As an enhancement, YOLOv8 has a significant increase in accuracy as measured by COCO and Roboflow 100. Additionally, YOLOv8 provides numerous convenient features for developers, including an easy-to-use command-line interface (CLI) and a well-structured Python package.

2.3.1 YOLOv5, 2020

This study selects YOLOv5m as the preferred option due to its ease of implementation, comprehensive application manuals, and tutorials. Moreover, YOLOv5m has resolved its bug and error issues and works more reliable and efficient than the latest YOLO models.

YOLOv5 is a computer vision model belonging to the You Only Look Once (YOLO) family, and is primarily used for object detection. As previously mentioned, YOLO series are classified as one/single stage object detectors which is illustrated in Figure 2.6.

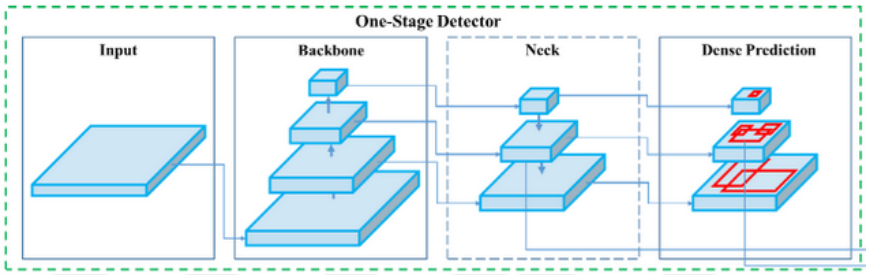


Figure 2.6: Single-Stage Detector Architecture [6]

Backbone refers to a pre-trained network that is utilized for extracting valuable feature representations from images. By doing so, the image's spatial resolution is diminished while its feature (channel) resolution is increased. The role of the **model neck** is to extract feature pyramids, enabling the model to efficiently detect objects with diverse sizes and scales, improving its ability to generalize to various scenarios. The primary responsibility of the **head** is to execute the concluding operations. These operations involve applying anchor boxes on feature maps and producing the final output, which encompasses classes, objectness scores, and bounding boxes.

The YOLOv5 models consist of three components, which are common across all models: CSP-Darknet53 serves as the backbone, SPP and PANet are included in the model neck, and the same head as YOLOv4 is used to perform final stage operations. The network architecture of YOLOv5 is shown in Figure 2.7. Figure 2.7 illustrates the framework of YOLOv5.

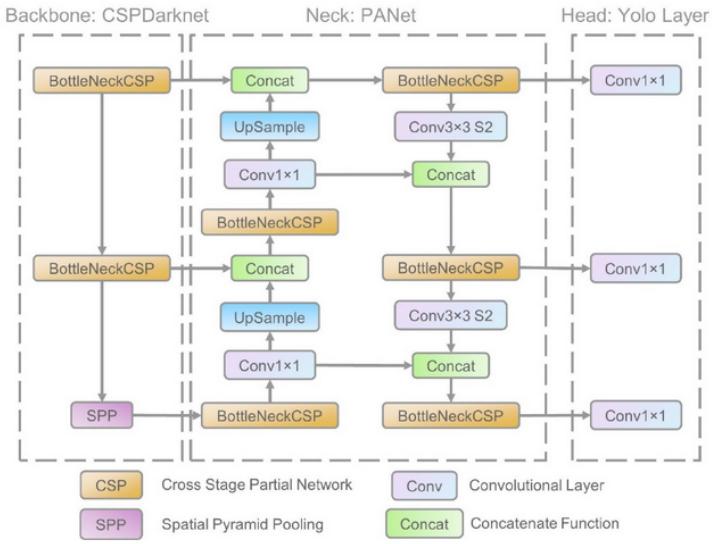


Figure 2.7: The YOLOv5 network architecture is comprised of three main components: (1) the backbone, which is CSPDarknet and used for feature extraction, (2) the neck, which is PANet and used for feature fusion, and (3) the head, which is the YOLO Layer and outputs detection results such as class, score, location, and size. [7]

Yolov5 incorporated **CSPNet** [53] into Darknet which is used as the backbone for YOLOv3. In large-scale backbone networks, CSPNet addresses the issue of redundant gradient information by incorporating gradient changes into the feature map. This approach reduces the number of parameters and FLOPS (floating-point operations per second) required by the model. As a result, the model size is reduced while maintaining inference speed and accuracy. In YOLOv5, the **PANet** [54] is used as the "neck" of the model to enhance the flow of information. PANet implements a new FPN [55] structure that enhances the bottom-up path, allowing for improved propagation of low-level features. The Yolov5 model incorporates adaptive feature pooling, which establishes connections between the feature grid and all feature levels. This enables the direct transmission of valuable information from each feature level to the subsequent subnetwork. PANet enhances the accuracy of object localization by effectively using more precise location information from lower layers. The YOLOv5's head, also known as the YOLO layer, produces feature maps of three different sizes $(18 \times 18, 36 \times 36, 72 \times 72)$ to enable the model to detect small, medium, and large

objects, which achieves multi-scale prediction. Thus, it helps us to accurately detect the objects where object's size and scale changes violently with the change in altitude on aerial vehicles.

Additionally, the selection of an activation function is an important decision for any deep learning model. In the case of YOLOv5, the authors opted to use **SiLU** [56, 57] which is also called the swish activation function. The authors of YOLOv5 used this function with the convolution operations in the hidden layers. On the other hand, they used the **Sigmoid** [57] activation function with the convolution operations in the output layer.

Although no research paper has been published yet to determine the strengths and weaknesses of YOLOv5, some airtificial practitioners have conducted tests to evaluate its performance on various benchmarks and proved its reliability. In this research, YOLOv5 is chosen as detection framework due to its extensive support, tutorials, and a large community of users.

CHAPTER 3

MULTIPLE OBJECT TRACKING (MOT)

3.1 Classification of the Object Tracking Methods

It is challenging to categorize a specific multiple object tracking (MOT) method based on a universal criterion. Admitting this, the categorization of MOT methods is approached from a broad perspective by assessing them based on multiple criteria which are *i) initialization method*, *ii) processing mode* and *iii) type of output* [23].

i) Initialization Method

To categorize MOT algorithms, one of the criteria is the initialization method. Generally, they can be divided into two groups: Detection-Based Tracking (DBT) and Detection-Free Tracking (DFT).

In the DBTapproach, objects are initially detected and subsequently connected to form trajectories. This methodology is commonly referred to as "tracking-by-detection." Within a given sequence, object detection or motion detection techniques, often based on background modeling, are applied to individual frames to obtain object hypotheses. Afterward, tracking is performed (either sequentially or in batches) to connect the detection hypotheses into trajectories. There are two key considerations for Detection-Based Tracking (DBT) algorithms. Firstly, because the object detector is trained beforehand, most DBT algorithms are designed to work with specific types of targets such as pedestrians, vehicles, or faces. Secondly, the effectiveness of DBT is heavily reliant on the accuracy of the object detector used.

On the other hand, DFT approaches [58, 59, 60, 61] do not rely on an object detector to detect objects in each individual frame. Instead, Rather, these approaches necessi-

Table 3.1: Comparison of two initialization method [1]

Item	DBT	DFT
Initialization	automatic, imperfect	manual, perfect
# of objects	varying	fixed
Applications	specific type of objects (in most cases)	any type of objects
Advantages	ability to handle varying number of objects	free of object detector
Drawbacks	performance depends on object detection	manual initialization

tate the manual initialization of a predetermined number of objects in the first frame, followed by the localization of these objects in the next frames. DFT algorithms are free of pre-trained object detectors and can track objects that do appear in the first frame. However, they cannot handle the case when new objects appear.

To put it differently, Detection-Based Tracking (DBT) is commonly preferred because it can automatically detect new objects and terminate disappeared objects. On the other hand, Detection-Free Tracking (DFT) cannot handle the case of appearing objects but does not require pre-trained object detectors. Table 3.1 provides an overview to initializing methods.

The algorithm used in this research, StrongSORT, belongs to the category of Detection-Based Tracking (DBT) methods.

ii) Processing Mode

In terms of processing mode, MOT algorithms can be classified in two categories: Online Tracking and Offline Tracking.

Online tracking is a real-time tracking approach where the tracking process is performed frame-by-frame as new data becomes available [13]. Online tracking requires the algorithm to process each incoming frame as it arrives, and the tracking results are updated on a per-frame basis. Online tracking methods are often used in scenarios where real-time tracking is essential, such as in video surveillance, robotics, and autonomous driving. Online tracking algorithms typically use data association techniques, such as Kalman Filtering [15], particle filtering, or graph-based optimization

Table 3.2: Comparison of processing mode[2]

Item	DBT	DFT
Input	Up-to-time observations	All observations
Methodology	Gradually extend existing trajectories with current observations	Link observations into trajectories
Advantages	Suitable for online tasks	Obtain global optimal solution theoretically
Drawbacks	Suffer from shortage of observation	Delay in outputting final result

methods, to track objects over time.

Offline tracking, on the other hand, is a post-processing approach where tracking is performed on a pre-recorded video sequence after all the data has been captured. Offline tracking methods are often used in scenarios where the highest tracking accuracy is required, such as in sports analysis, scientific research, or in forensic investigations [23, 13]. Offline tracking algorithms typically use batch optimization techniques, such as global optimization or maximum likelihood estimation, to optimize the tracking results over the entire sequence. To overcome the computational and memory limitations, it may not always be possible to process all frames in a sequence at once. One possible approach is to divide the data into shorter video clips and perform inference on each batch hierarchically or sequentially. Table 3.2 provides an overview to processing mode.

It's worth noting that some algorithms can operate both online and offline, depending on how the tracking problem is formulated and solved. The choice of whether to use an online or offline approach will depend on the specific application and the requirements of the task at hand. The algorithm used in this research, StrongSORT, belongs to the category of online tracking processing mode.

iii) Type of Input

In the sense of type of input, MOT algorithms can be divided into two groups: stochastic tracking and deterministic tracking.

Stochastic tracking methods, also known as probabilistic tracking, incorporate probability distributions into the tracking process. These methods assume that object motion and measurements are stochastic processes, and estimate the probability distri-

butions of these processes to predict the future states of the objects. Examples of stochastic tracking methods include Kalman filters, particle filters, and Markov chain Monte Carlo methods [21]. For instance, in detection-free tracking, the bounding box results can differ depending on the inference method used, such as particle filtering, due to the randomness in generating particles during processing. Similarly, some studies in detection-based tracking use single object trackers to improve detection bounding boxes, which can also result in different tracking outputs during different runs [23].

Deterministic tracking methods, on the other hand, do not consider the uncertainty in the tracking process and assume that the object motion and measurements are deterministic. These methods estimate the exact states of the objects and their trajectories by using various techniques such as data association, optimization, and graph-based methods. For example, when using a tracking-by-detection approach, data association methods like the Hungarian algorithm [62] will produce tracking results that are deterministic. Deterministic tracking is usually achieved through deterministic optimization methods used to derive the final output [21].

Both stochastic and deterministic tracking methods have their own advantages and disadvantages. Stochastic methods are more flexible and can handle non-linear and non-Gaussian motion models, but they require more computational resources and can suffer from particle degeneracy and sample impoverishment [21, 2]. Deterministic methods are computationally efficient and can provide exact object states, but they are sensitive to measurement noise and require accurate data association. The algorithm used in this research, StrongSORT, includes both stochastic and deterministic modules such as Kalman Filter and Hungarian algorithm [62], respectively.

3.2 Simple Online and Real-time Tracking (SORT)

SORT [25] is a simple implementation of a tracking method that uses object detection to track multiple objects in a video sequence. The method represents each object as a bounding box and updates their positions in each frame using a tracking-by-detection framework. In the SORT method, multiple object tracking is done by detecting ob-

jects in each frame and representing them as bounding boxes. Kalman filtering is employed to estimate the states of objects, while frame-by-frame data association is performed using the Hungarian method with an association metric that assesses the overlap of bounding boxes. However, the SORT method exhibits limited efficacy in tracking objects during occlusions, which frequently occur in scenes captured by front facing cameras.

The framework SORT algorithm consists of 4 main components: detection, estimation, association and management of track identities.

The accuracy of **object detection** has a significant impact on tracking performance, as it provides measurements to the tracker and functions as a visual sensor. In the **estimation** stage, the algorithm uses a representation and a motion model to continue tracking an object in the next frame. It uses a linear constant velocity model to estimate the object's movement between frames, which is not affected by other objects or camera motion. The state of a target motion model is given in equation (3.1).

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \tag{3.1}$$

The target's position in each frame is represented by the pixel coordinates of its center u and v, as well as its size (scale) and aspect ratio, which are denoted by s and r, respectively. During the tracking process, a constant aspect ratio is assumed. When a detection is successfully matched with a target, the bounding box of the detection is utilized to update the target's state, with the velocity components being optimally calculated through the application of the Kalman filter methodology. However, if there is no detection that corresponds to the target, the target state is predicted without any corrections using the linear velocity model. Figure 3.1 shows the SORT algorithm.

In the stage of **data association**, the cost of assigning each detection to each predicted bounding box is computed using the intersection-over-union (IOU) distance measure. This results in a cost matrix, which is then solved optimally using the Hungarian algorithm to determine the best assignment of detections to existing targets.

To create trackers for objects that enter the image or are not currently being tracked, we use detections that have an overlap with existing targets that is less than a specified

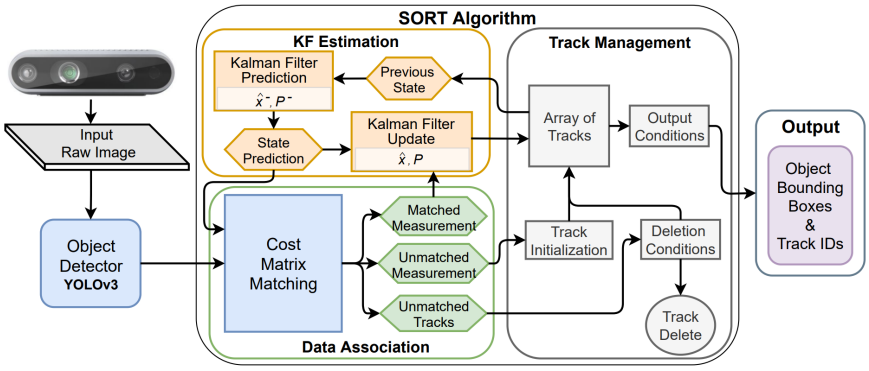


Figure 3.1: Overview of SORT algorithm [8]

threshold, IOU_{min}. After a new tracker is created based on a detection, it undergoes a probationary period. During this period, the tracker needs to be associated with further detections to gather enough evidence to prevent it from mistakenly tracking false positive objects. In order to prevent an uncontrolled increase in the number of trackers and localization errors, tracks are terminated if they are not detected for a certain number of frames, which is denoted by T_{Lost}. In the SORT algorithm, T_{Lost} is set to 1 due to two concerns of authors. First, the constant velocity model used in the tracker is not accurate in predicting the true dynamics of the object. Second, the authors are focused on frame-to-frame tracking rather than object re-identification. Deleting lost targets early also helps with computational efficiency. If an object reappears after being lost, tracking will continue under a new identity. Thus, the SORT algorithm is not effective in handling occlusions, even if the objects are occluded for a short time period.

3.3 Deep Simple Online and Real-time Tracking (DeepSORT)

DeepSORT [28] is a modified version of SORT that utilizes appearance information by means of a pre-trained association metric. The researchers conducted training of a convolutional neural network (CNN) using an extensive dataset specifically designed for person re-identification. The objective was to train the CNN to effectively differentiate between various pedestrians. By incorporating this network, they improved the system's resilience against misses and occlusions, while maintaining its ease of

implementation, efficiency, and suitability for real-time applications.

The DeepSORT tracking scenario is based on an eight-dimensional state space, different than the SORT algorithm, $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$, which includes the center position (u, v), aspect ratio (γ), height (h) of the bounding box, and their corresponding velocities in image coordinates. The DeepSORT algorithm utilizes a conventional Kalman filter that assumes constant velocity motion and linear observation model. In this model, the bounding coordinates (u, v, γ, h) of the object are directly taken as observations of its state.

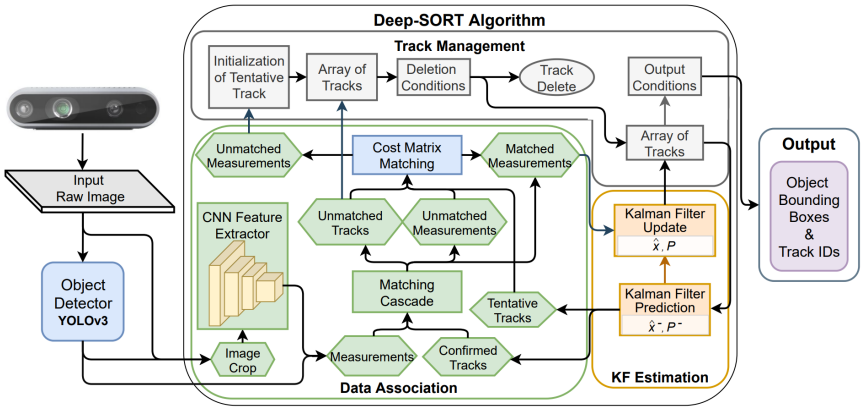


Figure 3.2: Overview of DeepSORT algorithm [8]

The method of managing tracks in DeepSORT is similar to the SORT algorithm, as depicted in Figure 3.2. DeepSORT maintains a record of the elapsed number of frames since the most recent successful measurement association for each track. This counter is incremented during the Kalman filter prediction phase and is reset to zero when a track is successfully associated with a measurement. Tracks surpassing a predetermined maximum age (A_{max}) are eliminated from the track set, as they are deemed to have exited the scene. For detections that cannot be associated with an existing track, new track hypotheses are initiated. During the initial three frames, newly formed tracks are categorized as tentative, and it is anticipated that they will be linked to measurements in each subsequent time step. If a track fails to establish a successful association with a measurement within these first three frames, it is eliminated.

On the other hand, data association part of the DeepSORT algorithm is more complex than the SORT algorithm since the appearance model and related cost calculations

are involved. One common approach for matching predicted Kalman states with new measurements is to formulate an assignment problem that can be solved with the Hungarian algorithm. While doing that, the authors of DeepSORT, at first, they intended to use combination of motion and appearance information by using two suitable metrics [28]. However, the authors decided to remove the impact of motion information on the combined association cost when there is significant camera motion by setting the value of hyperparameter λ to zero in equation (3.4). The detailed explanations are provided in the following paragraphs.

To include information about the motion of objects being tracked, DeepSORT uses the squared Mahalanobis distance (effective metric when dealing with distributions) to measure the distance between the predicted Kalman states and the newly arrived measurements which can be seen in equation (3.2).

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i) \tag{3.2}$$

In equation (3.2), the authors are referring to converting a track distribution in the object state space to the measurement space. The notation "(y_i, S_i)" represents the projection of the $i-th$ track distribution into measurement space, while dj represents the $j-th$ bounding box detection.

Essentially, the Mahalanobis distance takes into account the uncertainty in the estimated state by quantifying the deviation of the detection from the mean track location in terms of standard deviations. By using the Mahalanobis distance metric, it is possible to reject infeasible associations by setting a threshold at a 95% confidence interval. By doing that a $gate\ matrix$ guarantees that only associations within certain gating regions are admissible.

Due to the inherent limitations of the Kalman filtering framework in accurately estimating object location in the image-space problem, the Mahalanobis distance may not be an appropriate association metric when there is a high level of motion uncertainty. In particular, if the camera undergoes unaccounted movements, it can result in abrupt shifts in the image plane, rendering the Mahalanobis distance less informative for effectively tracking objects through occlusions.

Therefore, the authors proposed adding a second metric to the assignment problem to incorporate appearance information. To do that they compute appearance descriptor, r_j which is a unit vector, $||r_j|| = 1$. They maintain a collection of last $L_k = 100$ associated appearance descriptors called "gallery" denoted by $R_k = \{r_k^{(i)}\}_{k=1}^{L_k}$ for each track k. The second metric calculates the smallest cosine distance between the $i-th$ track and the $j-th$ detection in appearance space as given in equation (3.3).

$$d^{(2)}(i,j) = min\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in R_i\} \tag{3.3}$$

Both metrics address distinct aspects of the assignment problem and mutually enhance each other's capabilities. The Mahalanobis distance offers insights into possible object locations by considering motion characteristics, making it valuable for short-term predictions. On the other hand, the cosine distance incorporates appearance information, which becomes particularly valuable for re-establishing identities after prolonged occlusions when motion information becomes less reliable. The problem of association is formulated by combining both metrics through a weighted sum.

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j) \tag{3.4}$$

The combined association cost incorporates a hyperparameter called λ, which allows adjusting the weight assigned to each metric. Empirical observations suggest that setting $\lambda = 0$ is a reasonable choice when substantial camera motion is present.

When two tracks attempt to associate with the same detection, the Mahalanobis distance tends to give preference to tracks with higher levels of uncertainty. However, this behavior is not desirable as it can result in unstable tracks and track fragmentations. To overcome this challenge, the authors propose a matching cascade that prioritizes objects that have been observed more frequently. This approach incorporates their concept of probability distribution into the association likelihood, aiming to mitigate the issues caused by uncertainty and improve the stability of the tracking process.

In the last step of the matching process, they apply the intersection over union association method, which was proposed in the original SORT algorithm, to the un-

confirmed and unmatched tracks that have an age of one. This step helps to handle sudden changes in object appearance, for example, caused by partial occlusion with static scene geometry. Additionally, it improves the tracking performance by making it more robust against incorrect initialization. DeepSORT reduces ID switches and gives better tracking results than the SORT in case of occlusions.

3.4 Strong Simple Online and Real-time Tracking (StrongSORT)

The StrongSORT [9] method enhances DeepSORT with advanced components such as a detector and embedding model, as well as inference techniques. This approach can act as a robust and impartial starting point for other methods in multiple object tracking, which is useful for both academic and industrial purposes.

StrongSORT also provides solutions to the "missing association" and "missing detection" issues by introducing two new techniques: the appearance-free link model (AFLink) and the Gaussian-smoothed interpolation algorithm (GSI). AFLink relies only on spatiotemporal information to predict whether two tracklets are associated with the same ID, thus solving the missing association problem without requiring an appearance model and achieving a better balance between speed and accuracy. The GSI technique addresses the missing detection problem by using Gaussian smoothing to interpolate the missing detections, resulting in more robust tracklets. Both AFLink and GSI are simple, easily integrated, and do not depend on any particular model or appearance-based features. They can be used independently as standalone methods, making them versatile and suitable for a wide range of applications.

DeepSORT employs the Faster-RCNN [63], which has been optimized for performance, as its detector. In addition, a basic CNN is trained and used as the embedding model in the system. To improve detection accuracy, the creators of StrongSORT replace the object detector used in DeepSORT with **YOLOX** [49], recognizing that detection accuracy has a direct impact on tracking performance.

Moreover, StrongSORT utilizes a more powerful appearance feature extractor, **BoT** [64], in place of the original basic CNN as can be seen in 3.3. This allows for the extraction of more distinctive features, resulting in improved performance.

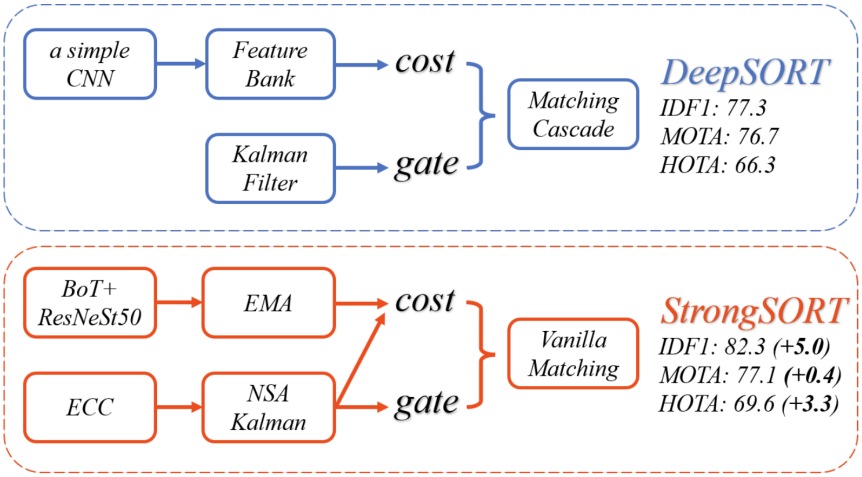


Figure 3.3: Illustration of StrongSORT improvements over DeepSORT algorithm [9]

To mitigate the vulnerability to detection noise in the feature bank mechanism of DeepSORT, StrongSORT introduces an alternative approach called the appearance state updating method. This method, proposed in [65], replaces the feature bank mechanism with an Exponential Moving Average (EMA) technique. The appearance state (e_i^t) of the tracklet at the i-th position in frame t is updated using the EMA method, as expressed in equation (3.5).

$$e_i^t = \alpha e_i^{t-1} + (1 - \alpha) f_i^t \tag{3.5}$$

The appearance embedding of the current matched detection is denoted as f_i^t, and a momentum term of $\alpha = 0.9$ is used in the EMA updating strategy. This strategy leverages inter-frame feature changes to alleviate the impact of detection noise. The experiments conducted by the authors of StrongSORT demonstrate that this approach enhances the accuracy of tracklet matching and reduces processing time.

Besides, camera movements are present in several benchmark datasets [66, 67, 68]. StrongSORT employs the **Enhanced Correlation Coefficient Maximization (ECC)** [69] model to compensate for camera motion. ECC is a method for aligning images parametrically that is capable of estimating the global rotation and translation between adjacent frames. Because of its effectiveness and computational efficiency, ECC is a commonly utilized approach for addressing motion noise introduced by

camera movement in MOT tasks.

Another improvement is that the standard Kalman filter is susceptible to low-quality detections [70] and neglects information related to the scales of detection noise [71]. In order to address this issue, StrongSORT adopts the **NSA Kalman** algorithm utilized in GIAOTracker [71]. The NSA Kalman algorithm adjusts the noise covariance matrix R_k, given in equation (3.6), based on the level of confidence associated with each detection. This enables it to adapt to varying levels of detection quality.

$$R_{k,uptdated} = (1 - c_k)R_k \tag{3.6}$$

where R_k refers to a constant covariance matrix for measurement noise, while c_k represents the detection confidence score associated with the state at time k. In simple terms, when a detection has a high confidence score, it means that it is less noisy and therefore has a low value for the covariance matrix R_k. As a result, the detection is given a higher weight during the state update step, leading to more accurate updates. Conversely, if a detection has a low confidence score and is more noisy, it is assigned a lower weight during the update step.

In the initial association stage of DeepSORT, only the appearance feature distance is used as the matching cost, while the motion distance serves as a gate. In contrast, StrongSORT addresses the assignment problem by considering both appearance and motion information. They combine the appearance cost (A_a) and motion cost (A_m) to obtain the cost matrix C. This is achieved by taking their weighted sum, as indicated in equation (3.7).

$$C = \lambda A_a + (1 - \lambda)A_m \tag{3.7}$$

The weight factor lambda is assigned a value of 0.98 by authors of StrongSORT, which is consistent with the approach used in [65, 72].

Another a noteworthy observation is that the matching cascade algorithm in DeepSORT can limit the performance of the tracker as it becomes more effective. This is because a stronger tracker is robust to confusing associations, and thus, additional

prior constraints can restrict the matching accuracy. To address this issue, the authors of the paper replaced the matching cascade with a method called **vanilla global linear assignment**. Based on the experiments conducted, it has been observed that the IDF1 score decreases for the DeepSORT algorithm when using vanilla matching instead of the matching cascade. However, for the StrongSORT algorithm, vanilla matching provides an additional score compared to the matching cascade.

CHAPTER 4

SURVEYS, DATASETS AND EVALUATION METRICS ON MULTI OBJECT DETECTION AND TRACKING

4.1 Surveys on Object Detection on Aerial Images

There are several surveys available in the literature. Since improvements in computer vision field increases exponentially in recent years, surveys become very useful to be able to see the overall picture and they enable to compare the recently proposed object detection methods in one place. For this end, I preferred to give place to surveys on this section.

A comprehensive survey conducted in 2019 by [73] presented an extensive analysis of the most recent progress in object detection using deep learning methods. This survey reviewed over 300 research contributions and examined various aspects of generic object detection, including detection frameworks, object feature representation, object proposal generation, context modeling, training strategies, and evaluation metrics. The survey encompassed a thorough assessment of more than 300 research works, by considering different fields of object detection. These fields include type of feature representation, proposal generation mechanisms, training strategies, modeling the context and type of evaluation metrics. The survey primarily focused on progress made in the last five years, limiting attention to still images and leaving video object detection as a separate topic for future consideration. The intention of the survey was to provide a helpful categorization for readers to better understand the similarities and differences between various strategies.

In 2019, the paper referenced as [2] provided a review of lightweight CNN architectures that can be used for object detection on embedded systems such as drones. The

focus of the paper is on CNN-based object detection algorithms and deep learning techniques, as well as commonly used and accurate lightweight CNN architectures.

In 2021, a comprehensive review was conducted by [74] on recently developed deep learning methods for detecting small objects. The main focus of this survey [74] is deep learning techniques for detecting small objects in images and provides a comprehensive review of related object detection and instance segmentation methods, including major challenges and strategies for improving detection performance on small objects.

Published in 2021, the research paper [75] offers a comprehensive survey of modern object detectors employing deep learning approaches. This article specifically focuses on lightweight classification models designed for edge devices, thoroughly assessing the performance of these architectures across multiple metrics. Additionally, the authors provide an in-depth evaluation of prominent backbone architectures and lightweight models used in object detection.

In 2021, [76] evaluated the performance of generic object detection systems under challenging conditions such as occlusions, low-light environments, and objects merged with the background in real-world scenarios. They conducted both quantitative and qualitative analyses of various detection approaches and discussed the available datasets for evaluating detection performance.

4.2 Aerial Image Datasets

This section discusses the performance of various publicly available aerial image datasets. The author emphasizes the importance of these datasets in evaluating the performance of object detectors, as they provide a benchmark for comparison. The author specifically focuses on datasets that contain aerial images, including those captured by drones, UAVs, and satellites, in order to align with the motivation of the research. This section also includes the MOT17 dataset, which is used to compare the tracking performance of the StrongSORT algorithm that has been previously tested and validated on this dataset.

4.2.1 UAVDT Dataset

In 2018, a video dataset named UAVDT [77] was introduced for object tracking and detection tasks using unmanned aerial vehicles (UAVs). The dataset consists of 80,000 horizontally annotated frames from 100 UAV recorded videos, out of which 50 video segments are used for target detection tasks. The dataset contains three object categories: car, truck, and bus, with 95%, 3%, and 2% labeling ratios, respectively. The training set and validation set contain 24,143 and 16,592 annotated images, respectively, with an average frame resolution of 1024×540. The dataset also considered 14 different attributes like weather condition, flying altitude, camera view, vehicle category, and occlusion while selecting the images from raw videos. However, the current state-of-the-art methods have relatively poor performance on the UAVDT dataset due to new challenges encountered in real-world UAV-based scenes, including high object density, small object size, and camera motion. The UAVDT benchmark dataset includes approximately 80,000 annotated frames with 2,700 vehicles and 0.84 million bounding boxes. The R-FCN detector, which is a two-stage detector, achieves an AP score of 34.35, which is relatively low for a region-based method. There is room for improvement in enhancing the accuracy of detecting small objects in challenging scenarios such as occlusions, out-of-view situations, and cluttered backgrounds. Although region-based methods have better performance compared to one-stage object detectors, their running speeds are too slow (<5fps) for practical applications with limited computing resources. On the other hand, region-free approaches eliminate the need for generating region proposals, resulting in significant time savings and enabling nearly real-time processing.

4.2.2 VisDrone Dataset

In 2021, a more advanced version of VisDrone 2018, called VisDrone2021 dataset [78] was released. This benchmark dataset contains 400 video clips, made up of 265,228 frames and 10,209 static images captured by various drone-mounted cameras, covering a wide range of scenarios like location, environment, and objects. The dataset includes more than 2.6 million horizontal bounding boxes or points manually annotated for frequent interests targets like pedestrians, cars, bicycles, and tricycles.

It also provides some important attributes for better data utilization like scene visibility, object class, and occlusion. The best-performing algorithms in terms of accuracy are DBNet, SOLOer, Win-T and TPH-YOLOv5, with scores ranging from 39.18% to 39.43%. However, the best mAP score is still below 40%, which is not good enough for practical applications, and the computational complexity of the submitted algorithms requires further consideration due to limited resources on the UAV platform.

4.2.3 DroneVehicle Dataset

In 2021, a dataset called DroneVehicle [79] was released, which is a large-scale dataset for drone-based RGB-Infrared cross-modality vehicle detection. It includes 28,439 pairs of RGB-Infrared images and provides annotations for 953,087 objects. The dataset covers various scenarios, including urban roads, residential areas, and parking lots, from day to night. However, since the RGB-Infrared cross-modality images provide both effective and redundant information, such as infrared images providing better object locations and categories for RGB images under low-light conditions but introducing false objects with similar appearance to real objects under good lighting conditions, there is ongoing work to improve the performance of object detection using the respective advantages of the RGB and infrared modalities. The issue of redundant and conflicting information in RGB-Infrared cross-modality images has been a topic of great interest to researchers in this field. They are actively working on improving object detection by leveraging the unique strengths of both RGB and infrared modalities. Moreover, infrared images are not very accurate in detecting objects in some well-lit daytime scenarios and some rectangular objects in infrared images can be mistaken for vehicles, which can lead to confusion. The problem of "ghost shadows" appearing in areas without vehicles and the possibility of pixel misalignment between RGB and infrared images introduces uncertainty in the position of detected objects. To address these issues, researchers have proposed the uncertainty-aware cross-modality vehicle detection (UA-CMDet) method, which is designed to be aware of such uncertainties in cross-modality vehicle detection. They removed this module after training to prevent any additional computation during inference. Additionally, they created an "illumination-aware cross-modal non-maximum suppression (IA-NMS)" strategy to combine the detection results from different modalities.

4.2.4 DOTA Dataset

In 2021, a dataset called Dataset of Object Detection in Aerial images (DOTA) was introduced, along with object detection baselines specifically designed for aerial images. This comprehensive dataset comprises 11,268 aerial images and contains a total of 1,793,658 object instances annotated with oriented-bounding-box annotations across 18 different categories [14]. The authors utilized the DOTA dataset to develop baseline models that encompassed a range of 10 cutting-edge algorithms and over 70 configurations. These models were employed to assess the speed and accuracy performance of each algorithm. Additionally, the authors provided a code library for object detection in aerial images (ODAI) and established a dedicated website for the comparison and evaluation of various algorithms. The authors of the study gathered images from different sources, including Google Earth, Gaofen-2 (GF-2) Satellite, Jilin-1 (JL-1) Satellite, and airborne images captured by CycloMedia in Rotterdam, with various resolutions and sensors. Moreover, the updated version of the dataset, DOTAv2.0, includes both RGB and grayscale images. The available code libraries for object detection mainly concentrate on horizontal object detection, with only limited support for oriented object detection, as in the case of Detectron2. However, in [14], they added several important operators to MMDetection to support oriented object detection, and then evaluated 10 different algorithms for object detection in aerial images. By default, MMDetection does not include modules that enable oriented object detection. In [96], the authors introduced significant modules, including rotated RoI Align and rotated position sensitive RoI Align, to extract features from rotated regions. These modules play a crucial role in algorithms like rotated region proposal network (RRPN) and RoI Transformer.

4.2.5 MOT17 Dataset

MOT17 is a popular dataset for evaluating multiple object tracking algorithms, which includes a diverse set of real-world video sequences captured from various cameras with different characteristics, such as different camera viewpoints, illumination conditions, object densities, and occlusion levels. The dataset provides a rich set of annotations, including object bounding boxes, object IDs, and occlusion and visibil-

ity labels, which allows for comprehensive evaluation of tracking performance. Due to its popularity and comprehensive evaluation metrics, the MOT17 dataset has been widely used in the research community to benchmark and compare the performance of different tracking algorithms.

4.3 MOT Evaluation Metrics

In the field of computer vision, Multi-Object Tracking (MOT) evaluation metrics serve as a means of quantitatively evaluating the performance of tracking algorithms used to track multiple objects in a video sequence. These metrics are designed to provide a measurable indicator of the tracking performance, which can be used to compare different tracking algorithms, optimize algorithm parameters, and track the progress of the field over time [80].

MOT refers to the process of taking a video as input and dividing it into individual frames at a designated frame rate. The output of MOT includes the **detection** of objects present in each frame, the **localization** of these objects within each frame, and the **association** of objects across different frames, determining whether they belong to the same object or different ones [80].

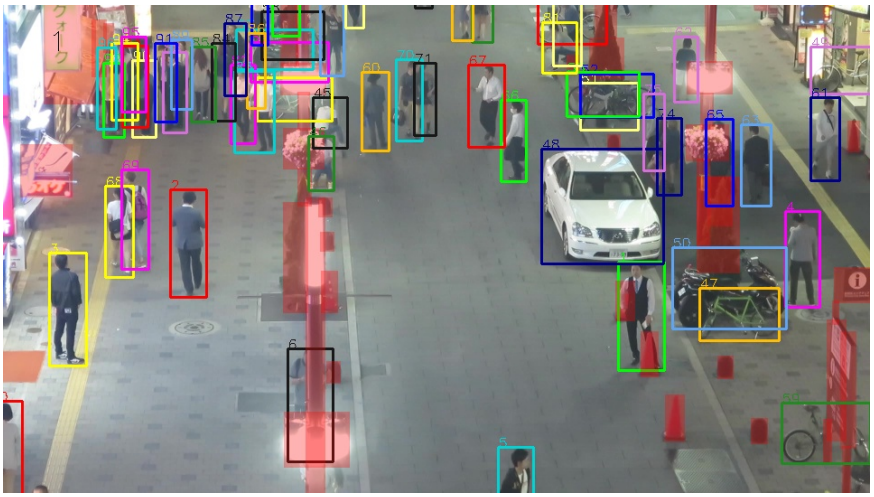


Figure 4.1: Ground truth annotation in a sample frame from MOT17 dataset

[81] propose a theoretical structure for examining the evaluation metrics used for

multi-object tracking (MOT), which is composed of two elements. First, multi-object tracking evaluation metrics should consider five different types of errors that can occur during the tracking process. These errors are False Negatives (FN), False Positives (FP), fragmentation, mergers, and deviation. **False Negatives** occurs when the ground truth exists, but the prediction is missed. **False Positives** happens when the tracker predicts the existence of an object that has no corresponding ground truth tracker. **Merge or ID Switch** takes place when two or more object tracks are swapped as they pass close to each other. **Deviation** happens when an object track is reinitialized with a different track ID. **Fragmentation** occurs when a track suddenly stops getting tracked, but the ground truth track still exists. Second, MOT evaluation metrics should have monotonicity and error type differentiability [81]. Frequently employed MOT evaluation metrics are

- Track-mAP
- Multi-Object Tracking Accuracy - MOTA
- Multi-Object Tracking Precision - MOTP
- IDF1
- Higher-Order Tracking Accuracy - HOTA

4.2 demonstrates how different evaluation metrics can have varying emphasis on different aspects of tracking performance. Three trackers (A, B, C) with different levels of detection and association accuracy are compared, with MOTA and IDF1 placing more weight on accurate detection and association, respectively. On the other hand, HOTA provides a balance between both by incorporating both detection and association scores.

In evaluating the performance of tracking algorithms, metrics such as IDF1, MOTA, and MOTP aim to condense the overall performance into a single number for easy comparison. However, metrics like HOTA provide more detailed information on the specific errors made by the algorithm, which is crucial in improving performance. The choice of metrics will depend on the specific use case, and the selection will significantly affect the focus of improvement. The major differences of these metrics are stated in Table 4.1

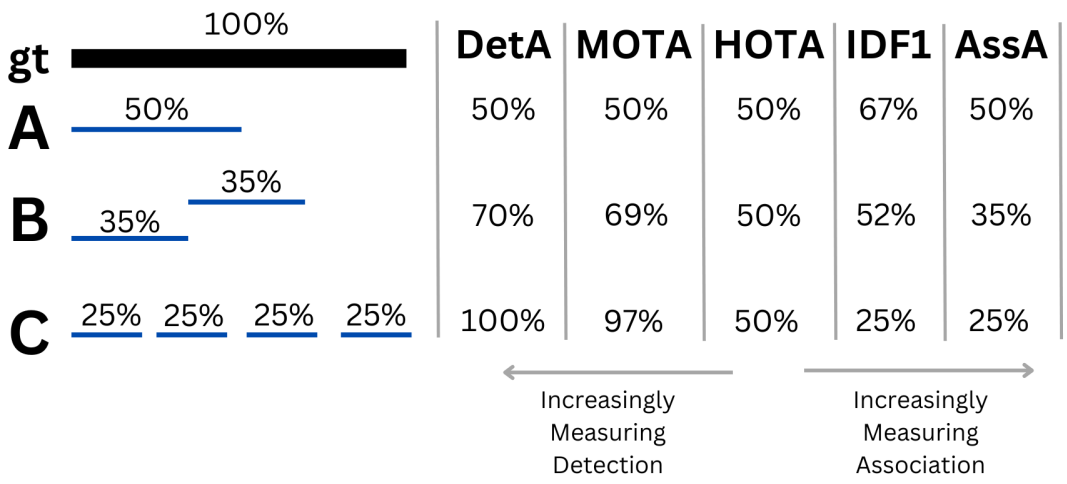


Figure 4.2: Comparison of MOT evaluation metrics [10]

Table 4.1: A summary of different evaluation metrics for MOT

Metrics	MOTA	IDF1	HOTA
Representation	Final Tracks	Final Tracks	Final Tracks
Matching Mechanism	Bijective	Bijective	Bijective
Matching Domain	Detection	Trajectory	Detection
Association Domain	Previous One Detection	Matched Detections	All Detections
Scoring Function	$1 - \frac{Err}{\lvert gtDet \rvert}$	F1 Score	Double Jaccard
Bias Toward	Detection	Association	Balanced

Comprehending the individual evaluation metrics assists in identifying how different types of errors impact the overall score. Awareness of the diverse types of errors that affect the evaluation metrics plays a significant role in enhancing the MOT scores and guiding future research. This study focuses on evaluating the algorithms using specific metrics, namely MOTA, HOTA, and IDF1, and provides a detailed analysis of these metrics.

4.3.1 MOTA

MOTA evaluates the tracking performance by comparing predicted detections (prDets) to ground truth detections (gtDets) at a detection level. The metric constructs a bijective mapping (one-to-one mapping) between these two sets of detections in each frame, based on their spatial similarity. This metric evaluates the tracker's overall performance by accounting for all the errors made in object configuration, including false positives, misses, and mismatches, across all frames [80]. It provides a straightforward measure of the tracker's ability to detect objects and maintain their trajectories, regardless of the accuracy of the object location estimation. Thus, detection performance significantly outweighs association performance.

As we previously discussed, if prDets and gtDets are similar enough, they are matched and considered true positives (correct predictions). Any prDets that are not matched become false positives (extra predictions) while any gtDets that are not matched become false negatives (missing predictions). Only adequately spatially similar prDets and gtDets can be matched. This means that in order to calculate MOTA, it is necessary to establish a measure of similarity between detections, represented by a score called S. Additionally, a threshold value, α, must be set to determine which detections can be considered a match, with a minimum required value of S for two detections (e.g. IoU_{Loc} for 2D bounding boxes) to be considered as the same object, $S \geq \alpha$. The multiple object tracking accuracy (MOTA) is defined as

$$MOTA = 1 - \frac{|FN| + |FP| + |IDSW|}{|gtDet|} \tag{4.1}$$

In Equation 4.1, FN and FP states the detection error while association error is stated with IDSW. It should be noted that MOTA does not take into account the localization error. Instead, another metric called MOTP (Multi-Object Tracking Precision) is defined in the CLEAR MOT metrics [80] specifically for measuring the accuracy of localization.

To further explain the MOTA metric on association error,an ID switch (IDSW) refers to a scenario in which a tracker incorrectly switches the identities of two objects or a track is lost and then re-initialized with a different identity. In other words, an IDSW

occurs when a TP has a predicted identity (prID) that is different from the prID of the previous TP that has the same ground truth identity (gtID). IDSWs only account for association errors between the current and the previous TPs with the same gtID, but not errors in which the same prID switches to a different gtID (known as ID transfer).

4.3.2 IDF1

IDF1 is a metric that places more importance on the accuracy of associating detections with ground truth objects rather than the accuracy of the detections themselves. This metric is considered a secondary measure in the MOTChallenge benchmark because it primarily measures the accuracy of association, rather than detection [10].

To calculate IDF1, a one-to-one mapping is established between the ground truth trajectories (gtTrajs) and the predicted trajectories (prTrajs), which is different from MOTA that matches at the detection level. This approach introduces new types of detection matches [10]. The overlaps between matched trajectories with a similarity score $S \geq \alpha$ are known as Identity True Positives (IDTPs). Identity False Negatives (IDFNs) and Identity False Positives (IDFPs) are the ground truth detections and predicted detections respectively that do not overlap with any matched trajectories, including those that are not matched. The Hungarian algorithm [62] is used to select the trajectories to match in order to minimize the sum of the number of IDFP and IDFN. IDF1, which combines Identity Precision (IDP) and Identity Recall (IDR), is then used to calculate the final score with the Equations given in (4.2, 4.3, 4.4). The Figure 4.3 demonstrates that, in IDF1, the best one-to-one mapping may not always produce the most accurate match between trajectories. In this particular example, the blue ground truth trajectory is matched with the gray predicted trajectory, even though it is a poor fit for both. Despite there being better matches available, the best one-to-one mapping is chosen based on the overall set of trajectories being evaluated.

The ID-Recall, ID-Precision and IDF1 scores are calculated as follows

$$ID - Recall = \frac{|IDTP|}{|IDTP| + |IDFN|} \tag{4.2}$$

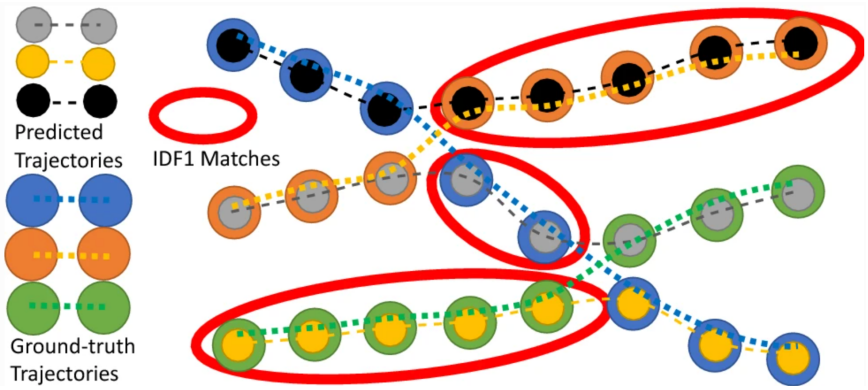


Figure 4.3: This tracking example illustrates how the use of IDF1's single best trajectory matching approach can lead to unintuitive matches between trajectories [10].

$$ID - Precision = \frac{|IDTP|}{|IDTP| + |IDFP|} \tag{4.3}$$

$$IDF1 = \frac{|IDTP|}{|IDTP| + 0.5|IDFN| + 0.5|IDFP|} \tag{4.4}$$

A high IDF1 score does not provide information about good detection or association, but rather estimates the total number of unique objects in a scene. It also does not assess the localization accuracy of trackers.

4.3.3 HOTA

HOTA builds upon MOTA metric that was previously used, and it addresses many of the limitations of MOTA. HOTA has been created to achieve three objectives: (i) give a unified score that combines various aspects of tracking evaluation, (ii) assess higher-order tracking association over a long period, and (iii) divide into sub-metrics that enable analysis of various components of tracker performance [10].

MOTA, IDF1, and HOTA all conduct a one-to-one matching between the ground truth detections and predicted detections using the Jaccard Index, which is given in Equation (4.5), or commonly called IOU score to evaluate the spatial similarity between the two. Any additional or missing predictions are penalized accordingly.

$$Jaccard\ Index = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{4.5}$$

A one-to-one correspondence is established between all the ground truth detections and predicted detections by using the Hungarian algorithm to determine the pairing that maximizes the sum of their matching scores.

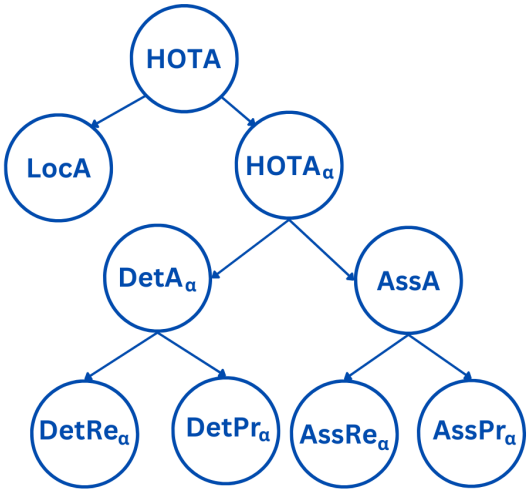


Figure 4.4: Decomposition of HOTA metric. Each sub-metric focuses on a specific type of tracking error, allowing for a more detailed analysis of the tracker's performance. [10].

Breaking down HOTA into sub-metrics has an additional benefit of allowing users to choose or adjust algorithms based on the specific requirements of their use case.

HOTA is computed at various values of a localisation threshold α, and the overall HOTA score is determined by averaging the HOTA$_\alpha$ scores obtained at each threshold. This approach ensures that the final HOTA score accurately reflects the localisation accuracy.

The accuracy of localisation score ($LocA$) can be evaluated independently from other components of tracking as following:

$$LocA = \int_0^1 \frac{1}{|TP_\alpha|} \sum_{c \in \{TP_\alpha\}} S(c)\, d\alpha \tag{4.6}$$

The spatial similarity score between the predicted detection (prDet) and ground truth detection (gtDet) is denoted as $S(c)$. It is calculated using multiple localization thresholds, denoted as α. HOTA can be split into two distinct components: the detection accuracy score ($DetA_\alpha$) and the association accuracy score ($AssA_\alpha$).

$$DetA_\alpha = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{4.7}$$

The Detection Accuracy ($DetA_\alpha$) metric, which is states in equation (4.7), measures how well the predicted detections align with the ground truth detections. A predicted detection is considered a true positive (TP) when the Localization Intersection-over-Union ($Loc - IoU$) score between the ground truth detection (grDet) and predicted detection (prDet) is greater than 0.5.

$$AssA_\alpha = \frac{1}{|TP|} \sum_{c \in \{TP\}} A(c) \tag{4.8}$$

$$A_c = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{4.9}$$

The Association Accuracy ($AssA_\alpha$) is a metric that evaluates the alignment between matched trajectories, taking into account all detections. It is calculated using equations (4.8) and (4.9). By combining the Detection Accuracy ($DetA_\alpha$) and the Association Accuracy ($AssA_\alpha$),

$$\begin{aligned} HOTA_\alpha &= \sqrt{\frac{\sum_{c \in \{TP\}} A(c)}{|TP| + |FN| + |FP|}} \\ &= \sqrt{DetA_\alpha AssA_\alpha} \end{aligned} \tag{4.10}$$

Equation (4.10) demonstrates that HOTA is computed by taking the geometric mean of a detection score and an association score, meaning that both aspects of tracking are equally considered in the final score. This is different from many other tracking metrics that may prioritize one aspect over the other, and the resulting HOTA score falls between the two. By employing this formulation, it ensures that both the detection score and association score possess a consistent structure and that each detection

makes an equal contribution to the overall score. The calculation of both scores is based on a Jaccard index formulation, further enhancing their similarity and coherence.

The HOTA formulation in equation (4.10) considers detection and association accuracy but ignores localization accuracy. To include localization accuracy, the final HOTA score is calculated as the integral of the HOTA score across the valid range of α values between 0 and 1.

$$HOTA = \int_0^1 HOTA_\alpha \, d\alpha \approx \frac{1}{19} \sum_{\alpha \in \{0.05,\ldots,0.95\}} HOTA_\alpha \tag{4.11}$$

The DetA and AssA scores can be computed by integrating over the range of α values, similar to the HOTA score calculation in equation (4.11).

In simple words, HOTA evaluates the quality of trajectory matching by measuring the alignment of detections and averaging it over all the matched detections. Additionally, it penalizes any detections that are not matched.

CHAPTER 5

MOTION AND MEASUREMENT MODEL REPRESENTATIONS

5.1 General Representation of Motion Models

In this section, the motion equations are presented for the discrete-time system. A dynamic model is utilized to describe the evolution of the object state over time. The object is considered as a point object in these dynamic models, disregarding its specific shape.

Before proceeding with the discrete-time models, fundamentals of continuous-time models [82] are briefly introduced because discrete-time models are frequently derived by discretizing their continuous-time models. The continuous-time model is given with the equations (5.1) and (5.2).

$$\dot{x}(t) = A(t)x(t) + D(t)u(t) + B(t)w(t), \quad x(t_o) = x_0 \tag{5.1}$$

$$z(t) = C(t)x(t) + v(t) \tag{5.2}$$

Equation (5.1) and (5.2) refer to the continuous-time version of the model, where the matrices $A(t)$, $D(t)$, $B(t)$, and $C(t)$ correspond to the continuous-time versions of the discrete-time F_k, G_k, E_k, and H_k matrices described in equations (5.11) and (5.12), respectively. The continuous-time process and measurement noise are represented by $w(t)$ and $v(t)$, respectively, and their covariances are denoted by $Q(t)$ and $R(t)$.

The equations for discretizing (sampling) a continuous-time linear system are presented below, which can be used for a time-invariant system with a sampling rate

$\Delta t = t_{k+1} - t_k$, through either Euler discretization or Zero-Order Hold (ZOH) methods [83, 84] are given in (5.1).

$$F_k \triangleq F(t_{k+1}, t_k) = F(t_{k+1} - t_k) = F(\Delta t) = e^{A\Delta t} \tag{5.3}$$

$F_k \in \mathbb{R}^{n \times n}$ is time-varying discrete-time transition matrix in state space model where the $A(t)$ matrix is the continuous time equivalent to the F_k matrix in equation (5.3). Similarly, discrete time input gain is obtained [83] as following

$$G_k \triangleq G(t_{k+1}, t_k) = \int_0^{\Delta t} e^{A(\Delta t - \tau)} D d\tau \tag{5.4}$$

and the noise in the process that occurs at discrete intervals is connected to the noise that occurs continuously over time can be written as

$$w_k \triangleq w(t_k) = \int_0^{\Delta t} e^{A(\Delta t - \tau)} E w(\tau) d\tau \tag{5.5}$$

With the zero-mean and white assumption (Gaussian) on $w(t)$ [83], the attributes of the process noise that has been discretized, w_k, shows the same characteristics

$$E[w_k] = 0, \quad E[w_k w_j'] = Q_k \delta_{kj} \tag{5.6}$$

where δ_{kj} is the Kronecker delta function. The covariance of the discrete-time process noise, Q_k, is given by

$$Q_k = \int_0^{\Delta t} e^{A(\Delta t - \tau)} B Q B' e^{A'(\Delta t - \tau)} d\tau \tag{5.7}$$

v_k is the measurement noise is obtained as similar to the equation (5.6) with zero mean and white noise assumption which can be seen in equation (5.8)

$$E[v_k] = 0, \quad E[v_k v_j'] = R_k \delta_{kj} \tag{5.8}$$

The discrete-time linear state-space models are generally represented in state space form with additive noise as represented in equation (5.9) and (5.10).

$$X_{k+1} = f_k(X_k, u_k) + w_k \tag{5.9}$$

$$Z_k = h_k(X_k) + \nu_k \tag{5.10}$$

At discrete time t_k, x_k, z_k, and u_k represent the object state, observation (measurement), and control input vectors, respectively. The variables w_k and v_k correspond to the process and measurement noise, respectively. In object tracking problem where the object in concern belongs to the third part, the control input *u* is not known. Equation (5.9) and (5.10) can be written in more popular form for 2D scenarios and it is a representation of discrete-time motion models.

$$X_{k+1} = F_k X_k + G_k u_k + E_k \omega_k \tag{5.11}$$

$$Z_k = H_k X_k + \nu_k \tag{5.12}$$

At discrete time, x_k represents the object state vector $\in \mathbb{R}^n$, z_k represents the observation vector $\in \mathbb{R}^m$, and u_k represents the control input vector $\in \mathbb{R}^p$. The process noise sequence $\omega_k \in \mathbb{R}^q$ with covariance $Q_k \in \mathbb{R}^{q \times q}$, while the measurement noise sequence $v_k \in \mathbb{R}^m$ with covariance $\mathbb{R}_k \in \mathbb{R}^{m \times m}$. The time-varying matrices $F_k \in \mathbb{R}^{n \times n}$, $G_k \in \mathbb{R}^{n \times q}$, $E_k \in \mathbb{R}^{n \times p}$, and $H_k \in \mathbb{R}^{m \times n}$ are the discrete-time transition matrix, input gain matrix, noise gain matrix, and measurement matrix, respectively which are derived through equations (5.3)-(5.8).

Object dynamics in horizontal plane used for state estimation and tracking problems can be derived from a standard continuous curvilinear-motion model [82].

$$\dot{x}(t) = v(t)\cos(\phi)(t) \tag{5.13}$$

$$\dot{y}(t) = v(t)\sin(\phi)(t) \tag{5.14}$$

$$\dot{v}(t) = a_t(t) \tag{5.15}$$

$$\dot{\phi}(t) = a_n(t)/v(t) = \omega(t) \tag{5.16}$$

where (x,y) is the object position, (v_x,v_y) is the linear velocity of the object in Cartesian coordinates along *x*-axis and *y*-axis. The variable a_t and a_n are the object tangential and normal accelerations in the horizontal plane, respectively and the values of them specify the motion model of a object as depicted in Figure 5.1. There are three different type of motion models used in estimation filters which are derived from equation (5.13)-(5.16), (nearly) constant velocity (CV) model, (nearly) constant acceleration model (CA), (nearly) constant turn model (CT) [82]. The *nearly* term states that velocity/acceleration/turn is nearly constant with the noise.

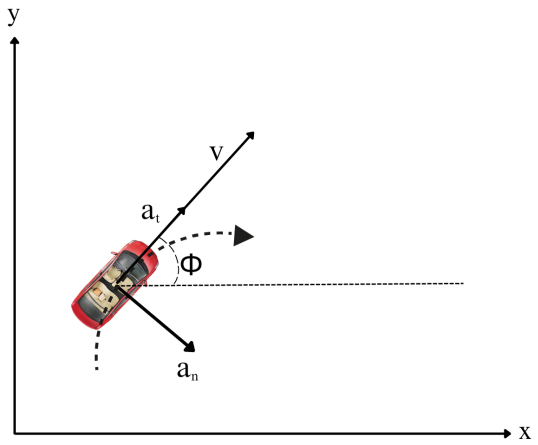


Figure 5.1: Object motion geometry in 2D plane

In deep learning based object tracking algorithms, motion models have 8-dimensional state vector for non-acceleration motion (X_k^{na}), and 10-dimensional state vector for accelerating motion (X_k^a) models. Those state vectors are written as

$$Non-accelerating : x_k^{na} = [x, y, a, h, \dot{x}, \dot{y}, \dot{a}, \dot{h}] \tag{5.17}$$

$$Accelerating : x_k^a = [x_k^{na}, \ddot{x}, \ddot{y}] \tag{5.18}$$

where $(x, y, \dot{x}, \dot{y})$ represents the object position and related velocities in Cartesian coordinates along *x*-axis and *y*-axis, respectively. Aspect ratio ($a = width/height$), height, and their respective velocities $(\dot{a}, \dot{h})$ are stated as $(a, h, \dot{a}, \dot{h})$, respectively.

5.2 Constant Velocity Model

From equation (5.13)-(5.16), (nearly) constant velocity motion model(CV) can be derived when $a_n = a_t = 0$. The state vector $x^{CV} \in \mathbb{R}^{8\times 1}$ can be written in Cartesian coordinates as stated similar to equation (5.17).

$$x_k^{CV} = [x, y, a, h, \dot{x}, \dot{y}, \dot{a}, \dot{h}] \tag{5.19}$$

The corresponding state equation for discrete white noise acceleration(DWNA) model [85] can be written as

$$x_{k+1}^{CV} = F^{CV} x_k^{CV} + \Gamma^{CV} w_k^{CV} \tag{5.20}$$

where F^{CV} is the state transition matrix, Γ^{CV} is the process noise gain and ω^{CV} is a scalar-valued zero-mean white acceleration process noise [85]. They are defined as

$$\begin{aligned} F^{CV} &= \begin{bmatrix} I_{4\times 4} & dt_{4\times 4} \\ 0_{4\times 4} & I_{4\times 4} \end{bmatrix}, \quad \Gamma^{CV} = \begin{bmatrix} \frac{1}{2} dt_{4\times 1}^2 \\ dt_{4\times 1} \end{bmatrix} \\ \omega_k^{CV} &\sim \mathcal{N}(0, Q^{CV}), \quad Q^{CV} = E\left[\Gamma w_k^{CV} w_k^{CV} \Gamma'\right] = \Gamma \sigma_\omega^2 \Gamma' \end{aligned} \tag{5.21}$$

The covariance of the process noise Q^{CV} is an 8×8 dimensional matrix which is obtained as shown in equation (5.21) [85]. The physical dimension of ω_k^{CV} and $\sigma_{a\omega}^2$ is $[pixel/time^2]$. The process noise is modeled as an acceleration noise in all states. It is qualified by a mean value of zero since its a type of white noise and standard deviation. Sampling time, *dt*, indicates the time difference between time step *k+1* and *k* which is constant during the experiments.

5.3 Constant Acceleration Model

From equation (5.13)-(5.16), (nearly) constant acceleration motion model(CA) can be derived when $a_n = 0, a_t =$const. The state vector $x^{CA} \in \mathbb{R}^{10\times 1}$ can be written in Cartesian coordinates as stated similar to equation (5.18).

$$x_k^{CA} = [x, y, a, h, \dot{x}, \dot{y}, \dot{a}, \dot{h}, \ddot{x}, \ddot{y}] \tag{5.22}$$

The corresponding state equation for discrete Wiener process acceleration(DWPA) model [85] can be written as

$$x_{k+1}^{CA} = F^{CA} x_k^{CA} + \Gamma^{CA} w_k^{CA} \tag{5.23}$$

They are defined as

$$F^{CA} = \begin{bmatrix} I_{4\times4} & dt_{4\times4} & \begin{bmatrix} diag(0.5dt^2)_{2\times2} \\ 0_{2\times2} \end{bmatrix} \\ 0_{4\times4} & I_{4\times4} & \begin{bmatrix} diag(dt)_{2\times2} \\ 0_{2\times2} \end{bmatrix} \\ 0_{2\times4} & 0_{2\times4} & I_{2\times2} \end{bmatrix}$$
$$\Gamma^{CA} = \begin{bmatrix} \frac{1}{2}dt^2_{4\times1} \\ dt_{4\times1} \\ I_{2\times1} \end{bmatrix} \tag{5.24}$$
$$\omega_k^{CA} \sim \mathcal{N}(0, Q^{CA}), \quad Q^{CA} = E\left[\Gamma w_k^{CA} w_k^{CA} \Gamma'\right] = \Gamma \sigma_\omega^2 \Gamma'$$

The covariance of the process noise Q^{CA} is an 10×10 dimensional matrix which is obtained as shown in equation (5.24) [85]. The physical dimension of ω_k^{CA} and $\sigma_{j\omega}^2$ is $[pixel/time^3]$. The process noise is modeled as an acceleration increment(jerk) noise for the states (x,y).

5.4 Constant Turn Model

From equation (5.13)-(5.16), (nearly) constant acceleration motion model (CT) can be derived when $a_t = 0, a_n =$const. The state vector $x^{CT} \in \mathbb{R}^{8x1}$ can be written in Cartesian coordinates as following

$$x_k^{CT} = [x, y, a, h, \dot{x}, \dot{y}, \dot{a}, \dot{h}] \tag{5.25}$$

The corresponding state equation for discrete white noise acceleration (DWNA) model [85] can be written as

$$x_{k+1}^{CT} = F^{CT} x_k^{CT} + \Gamma^{CT} w_k^{CT} \tag{5.26}$$

They are defined as

$$
\begin{aligned}
F^{CT} &= \begin{bmatrix} I_{4\times4} & \begin{bmatrix} \frac{\sin(\omega dt)}{\omega} & -\frac{1-\cos(\omega dt)}{\omega} \\ \frac{1-\cos(\omega dt)}{\omega} & \frac{\sin(\omega dt)}{\omega} \\ 0_{2\times1} & 0_{2\times1} \end{bmatrix} & \begin{bmatrix} 0_{2\times2} \\ dt_{2\times2} \end{bmatrix} \\ 0_{4x4} & \begin{bmatrix} \cos(\omega dt) & -\sin(\omega dt) \\ \sin(\omega dt) & \cos(\omega dt) \\ 0_{2\times1} & 0_{2\times1} \end{bmatrix} & \begin{bmatrix} 0_{2\times2} \\ dt_{2\times2} \end{bmatrix} \end{bmatrix} \\
\Gamma^{CT} &= \begin{bmatrix} \frac{1}{2}dt^2_{4\times1} \\ dt_{4\times1} \end{bmatrix} \\
\omega_k^{CT} &\sim \mathcal{N}(0, Q^{CT}), \quad Q^{CT} = E\left[\Gamma w_k^{CT} w_k^{CT} \Gamma'\right] = \Gamma \sigma_\omega^2 \Gamma'
\end{aligned}
\tag{5.27}
$$

The covariance of the process noise Q^{CT} is an 8×8 dimensional matrix which is obtained as shown in equation (5.27) [85]. The physical dimension of ω_k^{CT} and σ_ω^2 is $[pixel/time^2]$. The process noise is modeled as an acceleration increment noise for the all states. Turn rate, ω is taken as a constant value which is (-5,5) deg/s for left and right turn models so that our system model become a linear model. This model is used in UDIMM filter with the other motion models analyzed in previous sections 5.2 and 5.3.

5.5 Measurement Model

In the experiments, our measurements are directly obtained from the object detector (YOLOv5) which are the center coordinates of the bounding box with height and width (x, y, w, h). Therefore, YOLOv5 can be thought as a vision sensor which is linear model and gives the measurements in image coordinates frame. Before stepping into the tracking algorithm, aspect ratio($a = w/h$) is calculated to be used in filters. Thus, our measurement vector is $[x, y, a, h]$. The linear measurement model is represented by a matrix $H \in \mathbb{R}^{mxn^j}$ where $m = 4$ since x, y, a, h are measurements, n^j is the state dimension of j^{th} motion model which is in our case $n^{CV} = n^{CT} = 8$

and $n^{CA} = 10$. As in equation (5.12), measurement vector can be written as

$$z_k^j = H^j x_k^j + \nu_k = \begin{bmatrix} x & y & a & h \end{bmatrix}_k^T + \nu_k$$
$$\nu_k \sim \mathcal{N}(0, R_k),\ R_k = (1 - c_k)\ diag(\sigma_x^2, \sigma_y^2, \sigma_a^2, \sigma_h^2)_k \tag{5.28}$$

Measurement covariance matrix, R_k is adaptively modulated so that the noise is scaled during the state update procedure for time step, k. The covariance matrix of the measurements, denoted as R_k, remains the same across all motion modes. Due to varying levels of noise in different measurements, the scale of measurement noise ($\mathbb{R}_k$) should be adjusted based on the detection confidence (C_k) associated with each measurement. This covariance matrix is called Noise Scale Adaptive (NSA) covariance [71] and applying it to the Kalman Filters increase the performance of trackers in terms of HOTA metric [10, 9].



Figure 5.2: YOLOv5 measurements sample on image coordinates. This frame is also used in annotation format for training the neural network [11].

CHAPTER 6

ESTIMATION FILTERS

6.1 Kalman Filter

Kalman Filters are powerful estimation algorithms used in various fields, including aerospace and object tracking. They were developed by Rudolf Kalman and have become a fundamental tool in the field of control systems and state estimation [86].

At its core, a KF is a recursive algorithm that enables us to estimate the state of a system based on a series of measurements affected by noise. It combines the information from these measurements with a mathematical model of the system to provide an optimal estimate of the true state.

The key idea behind the KF is to maintain a belief about the system's state by iteratively updating it with new measurements. It operates in two steps: prediction and correction. In **prediction step**, the KF uses the system's mathematical model and the previous state estimate to predict the current state. It also predicts the uncertainty or covariance of the state estimate. In **correction step**, the KF incorporates the actual measurement into the prediction. It calculates the KF, which determines the relative contribution of the prediction and the measurement. The Kalman Gain is adjusted based on the covariance of the prediction and the measurement noise. The corrected state estimate is obtained by combining the prediction and the measurement using the Kalman Gain [15]. By iteratively performing prediction and correction steps, the KF effectively fuses information from multiple measurements over time, refining the state estimate and reducing the impact of noise. KFs are widely used in object tracking applications, such as tracking the position and velocity of moving targets.

In this section, Bayes' theorem and derivation of KF with the related assumptions are provided.

6.1.1 Bayes' Theorem

Bayes' theorem represents a concept that involves integrating prior and current information by employing the principles of conditional probability in probability theory [87]. When examining two interconnected events, denoted as x and y, the conditional probability of event x given the observation of event y can be mathematically expressed as follows:

$$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(y|x)p(x)}{p(y)} \tag{6.1}$$

Equation (6.1) represents the the joint probability of events x and y, which is denoted as $p(x,y)$, is normalized by the unconditional probability of event y, stated with $p(y)$.

Various aspects of the object tracking problem, including the number and states of objects, can be represented as events denoted by x, while different types of sensor outputs such as radar signals or infrared images can be regarded as events represented by y. By applying Bayes' theorem as shown in equation (6.1), it becomes possible to compute the conditional probabilities $p(x|y)$, which provide probabilistic solutions for the object tracking problem [87].

In the context of object tracking, the object states are represented by the joint PDF, $p(X_k) = p(X_k, X_{k-1}, ..., X_0)$. At time t_k, the variable X_k represents a general object state, which can encompass single or multiple object states, the count of objects, the identity of one or more objects, or a mixture of these factors. Additionally, $Y_k = (y_1, y_2, ..., y_k)$ denotes the measurements, y_i at time i. When a known physical relationship exists between the variable Y_k and the object states X_k, and information about the object states is accessible, it becomes feasible to establish a procedure for updating the knowledge of these states using Bayes' theorem. as given in equation (6.1). So, it can be written as

$$p(X_k|Y_k) = \frac{p(Y_k|X_k)p(X_k)}{p(Y_k)} \tag{6.2}$$

In equation (6.2), $p(X_k|Y_k)$ denotes the posterior distribution of the object state.The expression $p(Y_k|X_k)$ represents the likelihood function, which quantifies the probability of observing the sequence of measurements Y_k given the underlying sequence of states X_k. Th term in the numerator $p(X_k)$ is the prior distribution of X_k. The denominator $p(Y_k)$ serves as a normalization factor, ensuring that the resulting probability distribution $p(X_k|Y_k)$ satisfies the fundamental principles of probability and sums up to a total probability of 1.

Equation (6.2) offers a solution for calculating the distribution of X_k using the known sequence of measurements Y_k. However, since the measurements are received sequentially over time, it becomes necessary to update the distribution of the object state at each stage of the measurement process. This process is implemented to ensure that all available measurement is utilized as soon as it is received, enabling the object tracking problem to make the most accurate estimations and predictions.

To expand Equation (6.2) in recursive manner, $p(X_k|Y_k)$, which is the conditional distribution calculated at time k, is obtained after the final measurement y_k is received, by updating the conditional distribution of the object state $p(X_{k-1}|Y_{k-1})$ evaluated at time $k-1$. Thus, sequence of measurements can be represented as $Y_k = (y_k, Y_{k-1})$. By using the eqaution (6.1), we can rearrange the equation (6.2) with the following probabilities:

$$\begin{aligned} p(Y_k|X_k) &= p(y_k, Y_{k-1}|X_k) = p(y_k|Y_{k-1}, X_k)\; p(Y_{k-1}|X_k) \\ p(X_k) &= p(x_k, X_{k-1}) = p(x_k|X_{k-1})\; p(X_{k-1}) \\ p(Y_k) &= p(y_k, Y_{k-1}) = p(y_k|Y_{k-1})\; p(Y_{k-1}) \end{aligned} \tag{6.3}$$

Based on the $principle\ of\ causality$, it is evident that the measurements at time $k-1$ are independent of the object states at $times \geq k$ [87]. According to this principle, likelihood function is simplified, $p(Y_{k-1}|X_k) = p(Y_{k-1}|X_{k-1})$, and by substituting the probabilities derived in eqaution (6.3) into the equation (6.2) and with

some mathematical additional arrangements, recursive form of Bayesian solution is obtained as

$$p(X_k|Y_k) = \frac{p(y_k|Y_{k-1}, X_k)p(x_k|X_{k-1})}{p(y_k|Y_{k-1})} p(X_{k-1}|Y_{k-1}) \tag{6.4}$$

Assuming that the measurements at a specific time depend only on the object states at that time. Thus, it can be stated that the measurements obtained at time k are conditionally independent of the previous measurements obtained at times $\leq k-1$. Furthermore, these measurements depend only on the objects states at that instant time represented by x_k, rather than the entire sequence of object states, X_k. Considering that the measurement likelihood can be simplified to $p(y_k|Y_{k-1}, X_k) = p(y_k|x_k)$.

In the field of object tracking, it is frequently necessary to make simplifying assumptions in order to effectively model the problem using a probabilistic approach. One common assumption is based on the **Markov** property, which states that in many real-world systems, the current state is independent of previous states, given the most recent state. Therefore, on the world of object tracking, this means that x_k depends only on x_{k-1} and as a results, $p(x_k|X_{k-1}) = p(x_k|x_{k-1})$ is obtained. Reflecting those assumptions and the outcomes to the equation (6.4), we obtain the **Recursive Bayes solution**:

$$p(X_k|Y_k) = \frac{p(y_k|x_k)}{p(y_k|Y_{k-1})} p(x_k|x_{k-1}) p(X_{k-1}|Y_{k-1}) \tag{6.5}$$

The recursive Bayesian solution in equation (6.5) yields the posterior conditional distribution $p(X_k|Y_k)$ at time k, following the reception of the final measurement y_k. In the context of object tracking, it is crucial to determine the sequence of objects (tracklets) and their respective states at a particular time k. This information can be obtained from $p(X_k|Y_k)$ by integrating over the objects and their states. By applying multiple integral to the term $p(X_k|Y_k)$ through $dx_0 \ldots dx_{k-1}$, the **state conditional density** is obtained as

$$p(x_k|Y_k) = \frac{1}{p(y_k|Y_{k-1})} p(y_k|x_k) \int_{x_{k-1}} p(x_k|x_{k-1})\, p(x_{k-1}|Y_{k-1}) dx_{k-1} \tag{6.6}$$

Most object tracking algorithms calculate or strive to approximate equation (6.5), which represents the conditional density. The integral term has a special name called **"Chapman-Kolmogorov"** equation. By solving this integral, it becomes possible to obtain the predicted state of the elements in x_k, taking into account all the measurements up to time $k-1$ and the state at time $k-1$. The likelihood function is denoted with the $p(y_k|x_k)$ and while the transition density is stated with the notation $p(x_k|x_{k-1})$. The solution to the recursive relation described in equation (6.6) forms the essence of solving object tracking problems.

6.1.2 Derivation of the Kalman Filter

The Kalman filter is a well-established recursive technique in Bayesian estimation that is known for minimizing the estimated error covariance [86]. It is considered an optimal method for prediction and correction because it effectively combines prior knowledge with measurements. Within the Bayesian framework, estimation problems involve the prior density, which is determined by the object's dynamic equation in the case of object tracking, and the likelihood function, which is derived from the measurement equation. By recursively computing the posterior density of the object state, the Kalman filter enables efficient estimation in object tracking scenarios [87].

Prior to delving into the derivation of the Kalman Filter, it is necessary to derive the optimal recursion for the object tracking filter. To achieve that "object dynamics equation", "sensor measurement equation", "prediction and filtering", "transition density" and "likelihood function" are elaborated.

Object Dynamics Equation

Consider $x_k \in \mathbb{R}_x^n$ as the object state at time k. The object dynamics are commonly represented by a stochastic difference equation that can be expressed as follows

$$x_k = g(x_{k-1}, v_k) \tag{6.7}$$

where $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\omega} \rightarrow \mathbb{R}^{n_x}$ is a continuous and differentiable (twice) function with respect to its arguments. And the random noise input to the system is represented

with the variable ω_k, which is incorporated to model the errors. The prior density for the object state is given with the equation (6.7).

Sensor Measurement Equation

On the other hand, consider measurement $y_k \in \mathbb{R}^{n_y}$ which is observed at time k.

$$y_k = I(x_k, v_k) \tag{6.8}$$

where $\mathbf{I} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_x}$ is a continuous and differentiable (twice) function with respect to its arguments. v_k, which is the random variable, represents the measurement error in the model. Sensor measurement equation enables us to obtain the likelihood function $p(y_k|x_k)$.

g and **I** functions are linear, and the random noise variables v_k and ω_k are assumed to follow a Gaussian distribution. To compute the Gaussian posterior density of x_k using the Kalman Filter, certain approximations and numerical techniques are necessary.

Prediction and Filtering

Remind from state conditional density function,

$$p(x_k|Y_k) = \frac{1}{p(y_k|Y_{k-1})} p(y_k|x_k) \int_{x_{k-1}} p(x_k|x_{k-1})\, p(x_{k-1}|Y_{k-1}) dx_{k-1} \tag{6.9}$$

In Equation (6.9),

- $p(x_k|x_{k-1})$ is the transition density function which is derived from the object dynamics equation.
- $p(y_k|x_k)$ is the likelihood function which is derived from the sensor measurement equation.
- $p(x_{k-1}|Y_{k-1})$ denotes the posterior density at time $k-1$.
- $p(y_k|Y_{k-1})$ denotes the normalization factor.
- $p(x_{k-1}|Y_{k-1})$ states the prior probability density function.

- $p(x_k|Y_k)$ is the ultimate goal of the filter which is posterior probability density function.

After the sensor measurement y_k is observed, the conditional probability density function $p(x_k|Y_k)$ of the object state at time k, considering all the observed sensor measurements $Y_k = (y_1, ..., y_k)$, can be obtained using equation (6.9).

The recursive process described in equation (6.9) calculates the posterior density of the object state in two steps:

1. In the process of *prediction*, the conditional probability distribution from the previous stage is used with the transition probability distribution to generate an estimated probability distribution using the Chapman-Kolmogorov equation (CKE). The predicted density is given in equation (6.10).

$$p(x_k|Y_{k-1}) = \int_{x_{k-1}} p(x_k|x_{k-1})\, p(x_{k-1}|Y_{k-1}) dx_{k-1} \tag{6.10}$$

   The primary objective in object tracking is to determine the value of x_k. To achieve this, the predicted mean value of x_k is calculated using $p(x_k|Y_{k-1})$, and the accuracy of this estimation is measured by the covariance matrix of x_k which is also the outcome of the predicted density.

2. The process of *filtering* involves using the new data y_k with the likelihood function to create the filtering distribution $p(x_k|Y_k)$, as shown in equation (6.9).

Transition Density

In the majority of object tracking algorithms, it is common to assume additive noise in the object dynamics equation, resulting in an equation of the following form.

$$x_k = f(x_{k-1}) + \omega_k \tag{6.11}$$

Using the fundamental principle of transformation of random variables, the transition density function can be expressed in the following manner with the inverse of the equation (6.7), $\mathbf{g}^{-1}(x_k, x_{k-1}) = x_k - f(x_{k-1})$.

$$p(x_k|x_{k-1}) = p_{\omega_k}(g^{-1}(x_k, x_{k-1})) \left|\nabla_{x_k} g^{-1}(x_k, x_{k-1})\right| = p_{\omega_k}(x_k - f(x_{k-1})) \quad (6.12)$$

The Likelihood Function

The majority of object tracking algorithms make use of an additive assumption regarding the measurement noise. This assumption leads to a measurement equation that takes the following form:

$$y_k = h(x_k) + v_k \quad (6.13)$$

Using the fundamental principle of transformation of random variables, the likelihood function can be expressed in the following manner with the inverse of the equation (6.8), $\mathbf{I}^{-1}(y_k, x_k) = y_k - h(x_k)$.

$$p(y_k|x_k) = p_{v_k}(I^{-1}(y_k, x_k)) \left|\nabla_{y_k} I^{-1}(y_k, x_k)\right| = p_{v_k}(y_k - h(x_k)) \quad (6.14)$$

The Filter Recursion

Finally, by replacing the transition density (eq. (6.12)) and likelihood function (eq. (6.14)) within equation (6.9), we obtain the posterior density of the object state as follows:

$$p(x_k|Y_k) = \frac{p_{v_k}(y_k - h(x_k)) \int p_{\omega_k}(x_k - f(x_{k-1}))p(x_{k-1}|Y_{k-1})dx_{k-1}}{\int p_{v_k}(y_k - h(x_k))p(x_k|Y_{k-1})dx_k} \quad (6.15)$$

Equation (6.15) provides a concise mathematical representation of the Bayesian method for tracking non-maneuvering objects. Since the constant turn equations, stated in Section 5.4, are derived with a constant turn rate value, motion of equations become linear and the non-maneuvering case is also valid for our maneuvering case as well. So the equation (6.15) serves as the foundation for Bayesian tracking algorithms that

have been suggested in this research study. Distinctions among these algorithms stem from the varying assumptions made during the evaluation of equation (6.15).

The Kalman Filter is derived with the following **three assumptions**:

A1: The object dynamics and measurement equations are linear.

$$\begin{aligned} x_k &= Fx_{k-1} + \omega_k \\ y_k &= Hx_k + v_k \end{aligned} \tag{6.16}$$

A2: The sequences of noise variables ω_k and v_k are white, uncorrelated, and follow a Gaussian distribution with zero mean. The covariance of ω_k is denoted as Q_k, while the covariance of v_k is denoted as R_k.

A3: At time $k-1$, the posterior density of the object state $p(x_{k-1}|Y_{k-1})$ follows a Gaussian distribution with a mean of $\hat{x}_{k-1|k-1}$ and a covariance of $P_{k-1|k-1}$.

In the following equations Gaussian densities and Gaussian product will appear regularly in the derivations so its proof is explained in [88, 87]

Using equation (6.12) with the knowledge of p_{w_k} is Gaussian density with zero mean and covariance, the **transition density** is written as

By utilizing equation (6.12) and considering that p_{w_k} represents a Gaussian density with a mean of zero and a given covariance, the **transition density** can express as

$$p(x_k|x_{k-1}) = N(x_k; Fx_{k-1}, Q_k) \tag{6.17}$$

Additionally, the **prior density** of the object state at time $k-1$ is Gaussian as well, so it can be written as

$$p(x_{k-1}|Y_{k-1}) = N(x_k; \hat{x}_{k-1|k-1}, P_{k-1|k-1}) \tag{6.18}$$

By using the equations (6.17) and (6.18), the **predicted density** in (6.10) can be written as

$$p(x_k|Y_{k-1}) = \int N(x_k; Fx_{k-1}, Q_k)N(x_{k-1}; \hat{x}_{k-1|k-1}, P_{k-1|k-1})dx_{k-1} \tag{6.19}$$

The predicted density is calculated as by applying the Theorem 2.1 (Gaussian Product) [87] to the equation (6.19),

$$p(x_k|Y_{k-1}) = N(x_k; \hat{x}_{k|k-1}, P_{k|k-1}) \tag{6.20}$$

where $\hat{x}_{k|k-1}$ and $P_{k|k-1}$ denotes the **prediction step of Kalman Filter** with the following expression derived within the result of Theorem 2.1.

$$\begin{aligned} \hat{x}_{k|k-1} &= F\hat{x}_{k-1|k-1} \\ P_{k|k-1} &= FP_{k-1|k-1}F^T + Q_k \end{aligned} \tag{6.21}$$

By examining equation (6.21), it becomes evident that the **prediction density** assumes a Gaussian distribution when the prior density at time $k-1$ follows a Gaussian distribution and the object dynamic equation adheres to linearity or Gaussian properties.

Similarly, using equation (6.14) with the knowledge of p_{v_k} is Gaussian density with zero mean and covariance, the **likelihood function** is written as

$$p(y_k|x_k) = N(y_k; Hx_k, R_k) \tag{6.22}$$

The normalization factor in equation (6.15), which is integral term at the denominator, can be found using Theorem 2.1.

$$\begin{aligned} p(y_k|Y_{k-1}) &= \int p(y_k|x_k)\, p(x_k|Y_{k-1})dx_k \\ &= \int N(y_k; Hx_k, R_k)\; N(x_k; \hat{x}_{k|k-1}, P_{k|k-1})dx_k \\ &= N(y_k; \hat{y}_{k|k-1}, S_k) \end{aligned} \tag{6.23}$$

where $\hat{y}_{k|k-1}$ and S_k denotes the **measurement prediction step** of Kalman Filter with the following expression, eq. (6.24), derived within the result of Theorem 2.1.

$$\begin{aligned} \hat{y}_{k|k-1} &= H\hat{x}_{k|k-1} \\ S_k &= HP_{k|k-1}H^T + R_k \end{aligned} \tag{6.24}$$

Finally, the posterior density is calculated by incorporating the predicted density, the likelihood function, and the normalization factor derived from earlier equations into the Bayesian solution (6.9).

$$p(x_k|Y_K) = \frac{N(y_k; Hx_k, R_k)\, N(x_k; \hat{x}_{k|k-1}, P_{k|k-1})}{N(y_k; \hat{y}_{k|k-1}, S_k)} \tag{6.25}$$

Using the Theorem 2.1 once again,

$$p(x_k|Y_K) = N(x_k; \hat{x}_{k|k}, P_{k|k}) \tag{6.26}$$

where $\hat{x}_{k|k}$ and $P_{k|k}$ denotes the **update step** of Kalman Filter with the following expression, eq. (6.27), derived within the result of Theorem 2.1.

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + P_{k|k-1}H^T S_k^{-1}(y_k - \hat{y}_{k|k-1}) \\ P_{k|k} &= P_{k|k-1} - P_{k|k-1}H^T S_k^{-1} H P_{k|k-1} \end{aligned} \tag{6.27}$$

where the **Kalman Gain** can be denoted as

$$K_k = P_{k|k-1}H^T S_k^{-1} \tag{6.28}$$

Therefore, when provided with a Gaussian prediction density and a linear/Gaussian measurement equation, the resulting posterior density is also Gaussian. In Bayesian terms, this Gaussian prediction density is referred to as conjugate to the Gaussian likelihood.

To sum up, the Kalman Filter algorithm is given by

1. Predicted mean and covariance matrix:

$$\begin{aligned} \hat{x}_{k|k-1} &= F\hat{x}_{k-1|k-1} \\ P_{k|k-1} &= FP_{k-1|k-1}F^T + Q_k \end{aligned} \tag{6.29}$$

2. Predicted measurement, innovation covariance matrix and Kalman gain:

$$\begin{aligned} \hat{y}_{k|k-1} &= H\hat{x}_{k|k-1} \\ S_k &= HP_{k|k-1}H^T + R_k \\ K_k &= P_{k|k-1}H^T S_k^{-1} \end{aligned} \tag{6.30}$$

3. Posterior mean and covariance matrix:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_{k|k-1}) \\ P_{k|k} &= P_{k|k-1} - K_k H P_{k|k-1} \end{aligned} \tag{6.31}$$

For the notice, the covariance update $P_{k|k}$ in equation (6.31) is valid when the Kalman gain (K_k), given in equation (6.28), is optimal. If it is desired to use any K_k, not just optimal K_k, the following calculation for the posterior covariance matrix should be used. In equation (6.32), I is an identity matrix with appropriate dimension.

$$P_{k|k} = (I - K_k H)P_{k|k-1}(I - K_k H)^T + K_k R_k K_k^T \tag{6.32}$$

In practical applications, the computed Kalman gain by the filter is not the optimal value due to two main factors: the real world is rarely perfectly linear and Gaussian, and there can be floating-point errors during computation. By using the equation (6.32), the Kalman filter is less likely to diverge when dealing with real-world conditions [89]. In this research, it is chosen to utilize equation (6.32) as a covariance update due to the non-Gaussian characteristics observed in the measurements obtained from the YOLOv5 object detector.

6.2 Interactive Multiple Model (IMM) Filter

In many application scenarios, it is inadequate to represent an estimation problem using only one process model or one measurement model. The inclusion of multiple models becomes necessary, such as when capturing different types of motion,

where one model accurately represents straight line motion while another model captures curved motion. A mode in object tracking refers to a combination of a process model and a measurement model, where the switching between these modes follows a Markov process. However, when the mode switches are unknown, which is often the case in practice, achieving the optimal Bayesian solution for the problem becomes exponentially complex. Therefore, approximations are typically employed to address this challenge [90].One of the approximation method is the Interacting Multiple Model (IMM) estimator [17].

The Interacting Multiple Model (IMM) filter is considered to be a highly efficient class of estimators for tracking maneuvering targets. It offers a good balance between estimation accuracy and computational complexity, making it a cost-effective choice for handling targets with changing dynamics. The reason behind the impressive effectiveness of the IMM estimator lies in its strategic approach of mixing right before updating the mode-conditioned estimates, similar to the optimal Bayesian estimator [90].

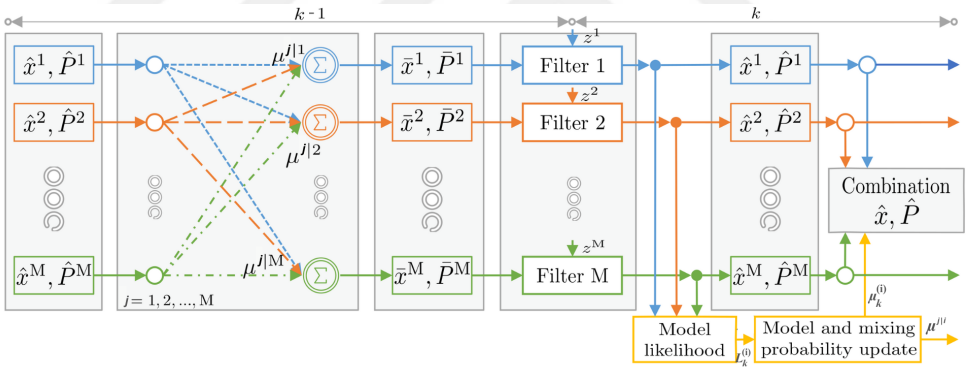


Figure 6.1: One cycle of the IMM estimation algorithm. Interaction/mode mixing step is illustrated on the left side, filtering is shown on the middle section and final probability evaluations are represented on the right side of the algorithm scheme [12].

The Interacting Multiple Model (IMM) holds a unique position because of its ability to be integrated with other algorithms to address the challenge of tracking multiple targets. Filters used in IMM filter are based on different models as well as different types, e.g. KF, UKF, EKF or particle filter as can be seen in Figure 6.1. IMM is built upon the two assumptions:

1. The actual sequence of modes, denoted as s_k, follows a Markov process, meaning that the mode at each time step depends only on the previous mode. This can be described by the probability $\pi_{ji,k} = P\{s_k^{(j)}|s_{k-1}^{(i)}\}$, which represents the transition probability from mode j to mode i at time step k. These transition probabilities are denoted as $\pi_{ij,k}$ and hold true for all combinations of modes i and j.

2. The true mode s at any given time belongs to a mode space denoted as $\mathbb{S}$. This mode space $\mathbb{S}$ is time-invariant and remains the same throughout the entire process. It is identical to the finite model set $\mathbb{M}$ that is also time-invariant, meaning that $\mathbb{S}_k = \mathbb{M}$ holds true for all time steps k.

Table 6.1: One cycle of the IMM algorithm

Model re-initialization ($\forall m_i \in \mathbb{M}$):	
Predicted model probability:	$\bar{c}^{(i)} = \sum_j \pi_{ji}\mu_{k-1}^{(j)}$
Mixing Probabilities:	$\mu_{k-1}^{j\|i} = \pi_{ji}\mu_{k-1}^{(j)}/\bar{c}^{(i)}$
Mixed State:	$\bar{x}_{k-1\|k-1}^{(i)} = \sum_j \hat{x}_{k-1\|k-1}^{(j)}\mu_{k-1}^{j\|i}$
Mixed Covariance:	$\bar{P}_{k-1\|k-1}^{(i)} = \sum_j [P_{k-1\|k-1}^{(j)} + (\bar{x}_{k-1\|k-1}^{(i)} - \hat{x}_{k-1\|k-1}^{(j)})$ $(\bar{x}_{k-1\|k-1}^{(i)} - \hat{x}_{k-1\|k-1}^{(j)})^T]\mu_{k-1}^{j\|i}$
Model-conditioned filtering ($\forall m_i \in \mathbb{M}$):	
Predicted state:	$\hat{x}_{k\|k-1}^{(i)} = F_{k-1}^{(i)}\bar{x}_{k-1\|k-1}^{(i)}$
Predicted covariance:	$P_{k\|k-1}^{(i)} = F_{k-1}^{(i)}\bar{P}_{k-1\|k-1}^{(i)}\left(F_{k-1}^{(i)}\right)^T + Q_{k-1}^{(i)}$
Innovation residual:	$y_k^{(i)} = z_k - H_k^{(i)}\hat{x}_{k\|k-1}^{(i)}$
Innovation covariance:	$S_k^{(i)} = H_k^{(i)}P_{k\|k-1}^{(i)}(H_k^{(i)})^T + R_k^{(i)}$
Filter gain:	$K_k^{(i)} = P_{k\|k-1}^{(i)}(H_k^{(i)})^T(S_k^{(i)})^{-1}$
Update state:	$x_{k\|k}^{(i)} = x_{k\|k-1}^{(i)} + K_k^{(i)}y_k^{(i)}$
Update covariance:	$P_{k\|k}^{(i)} = P_{k\|k-1}^{(i)} - K_k^{(i)}S_k^{(i)}(K_k^{(i)})^T$
Model probability evaluation ($\forall m_i, m_j \in \mathbb{M}$):	
Model likelihood:	$L_k^{(i)} = \mathcal{N}(y_k^{(i)}; 0, S_k^{(i)})$
Model probability:	$\mu_k^{(i)} = \frac{\bar{c}^{(i)}L_k^{(i)}}{\sum_j \bar{c}^{(j)}L_k^{(j)}}$
Estimate Fusion ($\forall m_i \in \mathbb{M}$):	
Overall state:	$\hat{x}_{k\|k} = \sum_i \hat{x}_{k\|k}^{(i)}\mu_k^{(i)}$
Overall covariance:	$P_{k\|k} = \sum_i \left[P_{k\|k}^{(i)} + (\hat{x}_{k\|k} - \hat{x}_{k\|k}^{(i)})(\hat{x}_{k\|k} - \hat{x}_{k\|k}^{(i)})^T\right]\mu_k^{(i)}$

The idea is to have single filter for each possible mode of behavior of the system. At each epoch filters interact with each other according to Markov state transition

probabilities π_{ji}. The state estimates of all filters (characterized by the mean $\hat{x}^i$ and covariance $\hat{P}^{(i)}$) are mixed using a set of mixing probabilities $\mu^{j|i}$. These mixing probabilities are computed based on the model probabilities $\mu^{(i)}$ which are updated using likelihoods $L_k^{(i)}$ of each filter, prior model probabilities $\mu^{(i)}$ and the state transition probabilities π_{ji}. In the final fusion step, the updated state estimates from all model filters are combined by taking a weighted sum. This fusion process considers the likelihood of each filter, allowing the more probable filters to influence and modify the estimates of the less probable filters [12, 34]. In Table 6.1, a complete cycle of IMM filter is presented.

The design parameters of the IMM estimator, which have a direct impact on performance, can be expressed as follows:

- The model set, $\mathbb{M}$.
- The system noise covariance, Q and measurement noises covariances,R.
- The initial state $x_{0|0}$ and the state covariance $P_{0|0}$.
- Transition probabilities π_{ji} that govern the transitions between the models.

6.3 Unequal Dimension Interactive Multiple Model (UDIMM) Filter

In the context of applications of the standard IMM estimator and the Unequal Dimension Interactive Multiple Model (UDIMM) estimator, they are nearly identical except for the mixing step. In this study, UDIMM incorporates four modes with varying dimensions due to the inclusion of an additional constant acceleration model (as discussed in Section 5.3). Researches [18, 12, 20, 19] indicates that the choice of mixing or augmentation approach has a considerable influence on the performance of the IMM algorithm.

In the literature, the proposed state augmentation approaches are given as follows. The a represents the augmented states vector and covariance.

1. *A simple approach (biased)*, e.g. [91] involves augmenting with zero mean and

variance.

$$\hat{x}^{(i,a)} = \begin{bmatrix} \hat{x}^{(i)} \\ 0 \end{bmatrix}, \quad P^{(i,a)} = \begin{bmatrix} P^{(i)} & 0 \\ 0 & 0 \end{bmatrix} \tag{6.33}$$

2. The *unbiased approach* [92] augments the smaller state vector by incorporating the mean and variance values from the larger state vector.

$$\hat{x}^{(i,a)} = \begin{bmatrix} \hat{x}^{(i)} \\ \hat{x}_\theta \end{bmatrix}, \quad P^{(i,a)} = \begin{bmatrix} P^{(i)} & 0 \\ 0 & P_{\hat{x}_\theta} \end{bmatrix} \tag{6.34}$$

where $\hat{x}_\theta$ is the additional state which can, e.g., be model acceleration, turn-rate, steering angle, or thrust and $P_{\hat{x}_\theta}$ is the related covariance which both are the same as the larger state vector. So, bias does not exist after the modified mixing.

3. The *no information approach* involves augmenting the information matrix with zeros, indicating that there is no available information regarding the augmented state. In this case, the scalar standard deviation is chosen as large as the programming language would allow where the zero information corresponds to infinite covariance.

$$\hat{x}^{(i,a)} = \begin{bmatrix} \hat{x}^{(i)} \\ 0 \end{bmatrix}, \quad P^{(i,a)} = \begin{bmatrix} P^{(i)} & 0 \\ 0 & \infty \end{bmatrix} \tag{6.35}$$

4. *The uniform approach*, e.g. [18], involves the augmented states in the given scenario which are represented using a uniform distribution. To define each augmented state, two parameters $[a, b]$ are required, which determine the expected value and variance of the uniform distribution.

$$\hat{x}^{(i,a)} = \begin{bmatrix} \hat{x}^{(i)} \\ E_q[\theta] \end{bmatrix}, \quad P^{(i,a)} = \begin{bmatrix} P^{(i)} & 0 \\ 0 & Cov_q(\theta) \end{bmatrix} \tag{6.36}$$

$$\begin{aligned} E[\theta] &= \begin{bmatrix} \frac{a_1+b_1}{2} & \dots & \frac{a_{n_\theta}+b_{n_\theta}}{2} \end{bmatrix}^T \\ Cov(\theta) &= diag\left(\begin{bmatrix} \frac{(b_1-a_1)^2}{12} & \dots & \frac{(b_{n_\theta}-a_{n_\theta})^2}{12} \end{bmatrix} \right) \end{aligned} \tag{6.37}$$

The values of $[a, b]$ strongly affects the performance and final results of the filter estimations, so they should be chosen studiously.

CHAPTER 7

SIMULATION ENVIRONMENT, MODEL FRAMEWORK AND PERFORMANCE PARAMETERS

7.1 Test Setups

This study presents the results obtained through simulations and tests using real-life footage captured by a quadcopter. These results serve to demonstrate the effectiveness and reliability of the approach being investigated.

7.1.1 Simulation Scenario

The experiments are conducted in Python, utilizing a computer with an i7 12800HX 3.40 GHz processor and an RTX3080Ti GPU, while running on the Linux Ubuntu 20.04 operating system. Simulation environment is prepared using *Pygame* library in Python. There are three simulation scenarios to be evaluated: 1) Single object with no occlusion, 2) Single object with occlusion, 3) Multiple objects with occlusion.

For the evaluation phase, a challenging simulation scenario is created with multiple U-shaped trajectories, accelerating motion and occlusions involving obstacles and intersections of moving two objects. Table 7.1 provides a comprehensive record of the actual trajectory with the physical value of the state and duration of each mode. Additionally, it encompasses the periods of time in which the measurements are unavailable. The simulated scenario comprises three distinct motion modes with varying durations. Sampling time is set to $dt = 1$. The trajectories of two objects depicted on simulation environment can be seen in Figure 7.1. There are 10 mode switch in total for both moving objects during 193 time step. The three type of ground truth motion

mode is illustrated in Figure 7.1 with different colors; CV mode is depicted with blue line, CA mode is shown with the red line and CT mode is described with the green line. Additionally, the boundaries of the simulation environment is set to 1950×720 pixel where the origin is located on the upper left corner regarding the image frame. The starting points are depicted in Figure 7.1 using blue and red circles, along with a black circle inside. The initial points points are also visualized in the graphs in Figure 7.1 along with the car image.

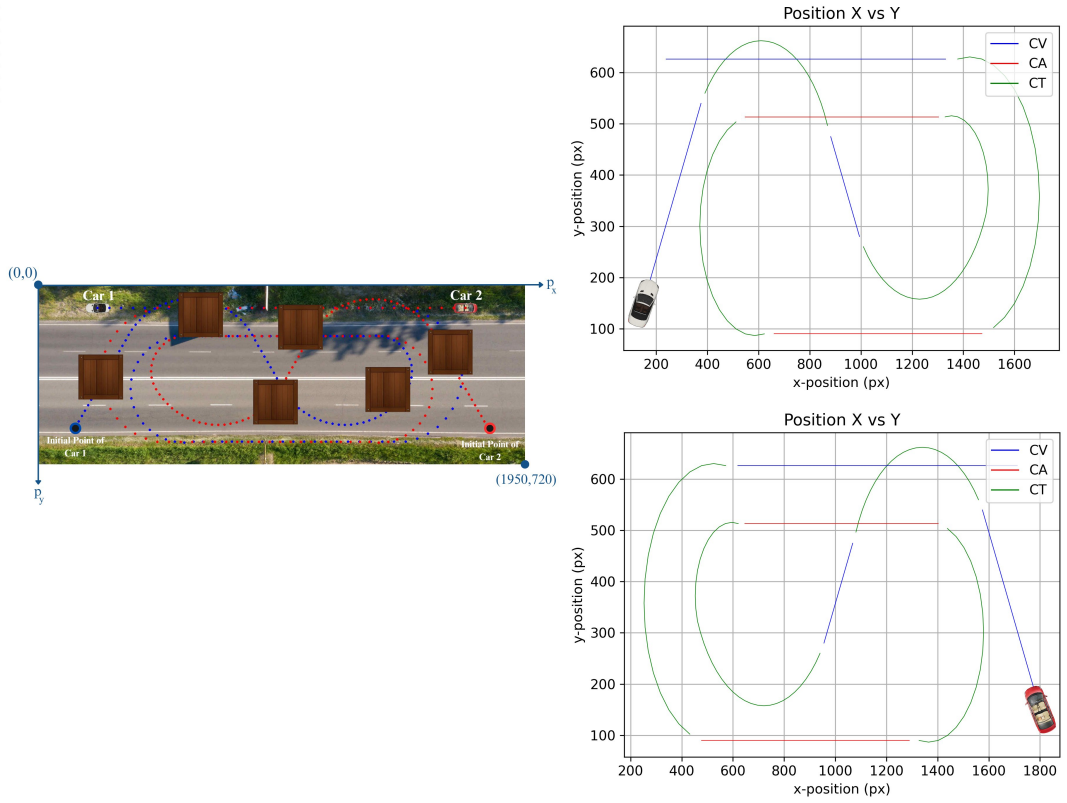


Figure 7.1: Ground-truth trajectories of simulated scenario for the two moving objects

The performance of estimation filters is being evaluated on simulation scenarios based on several factors such as the normalized estimated squared error, covariance bound coverage and track percentage.

In particular, the fourth, fifth, sixth and ninth modes illustrated in Table 7.1 present significant challenges as the motion mode transitions from CT motion to CA motion when occlusion occurs. Due to the similarity between CT and CA motion modes,

multiple model estimator encounter difficulties in accurately calculating the likelihood of the models. Furthermore, the transition moment between the fourth and fifth modes introduces substantial noise into the system, as the turn rate changes while the motion mode remains the same.

Table 7.1: Ground-truth modes and values with durations

		Car 1			**Car 2**		
No	**Mode**	**Value**	**Duration (s)**	**N/A Meas.**	**Value**	**Duration (s)**	**N/A Meas.**
1.	CV	25 px/s	18	6 - 13 sec	25 px/s	18	11 - 19 sec
2.	CT	-5 $°/s$	24	28 - 40 sec	5 $°/s$	24	39 - 43 sec
3.	CV	25 px/s	10	49 - 52 sec	25 px/s	10	-
4.	CT	5 $°/s$	24	53 - 56 sec 71- 75 sec	-5 $°/s$	24	-
5.	CT	10 $°/s$	12	76 - 80 sec	-10 $°/s$	12	-
6.	CA	0.5 px/s^2	25	95 - 102 sec 109 - 113 sec	0.5 px/s^2	25	88 - 91 sec 101 - 106 sec
7.	CT	10 $°/s$	18	-	-10 $°/s$	18	-
8.	CA	0.5 px/s^2	20	139 - 140 sec	0.5 px/s^2	20	-
9.	CT	10 $°/s$	18	160 - 168 sec	-10 $°/s$	18	-
10.	CV	47.5 px/s	25	183 - 186 sec	47.5 px/s	25	170 - 174 sec 178 - 180 sec

The period of unavailable measurements (N/A measurements), as indicated in Table 7.1, is obtained when two objects are simulated simultaneously, thus the occlusion due the one another is also taken into account.

7.2 Model Framework

It encompasses several areas of study such as object motion estimation, calculating similarity or distance between detection pairs, data association, and designing appearance model networks. The objective of this research was to enhance the precision of object motion estimation by employing the Unequal Dimension Interactive Multiple Model (UDIMM) [18] approach.

The algorithm framework utilized in this research is depicted in Figure 7.2. In this

context, first the environment is initiated so that the video stream is provided to the object detector (YOLOv5) which works as a vision sensor providing measurements to the system. Measurements includes the information of center coordinates, aspect ratio and height of the bounding box which is used in data association and UDIMM estimator stage. Detections are provided to the Matching Cascade algorithm (Cosine Distance + Mahalanobis Distance + Hungarian algorithm) along with the appearance features provided by feature extractor [64]. The deep features are subjected to the cosine distance metric in order to compute the cost matrix, which is used to solve the linear assignment problem known as the Hungarian algorithm. Prior to inputting this cost matrix into the Hungarian algorithm [62], the squared Mahalanobis distance, also referred to as the *gating distance*, is calculated between the state distribution and measurements. This gating distance helps eliminate infeasible entries in the cost matrix by considering the state distributions obtained through Kalman filtering. After applying the Hungarian algorithm, the unmatched tracks and unmatched detections are used as input for the intersection over union (IoU) distance metric. This metric is employed to associate the remaining tracks along with the unconfirmed tracks together.

Following the association stage, there are three potential outputs: 1) Unmatched Tracks, 2) Unmatched Detections, and 3) Matched Tracks. If an unmatched track is unconfirmed (having fewer than three associated detections) or if it is confirmed but exceeds the maximum age (the limit for predicted steps with the motion model, without measurement updates), the track is removed. Conversely, if the unmatched track is confirmed and its age is within the maximum age threshold, it undergoes the prediction step of the UDIMM estimator. During the prediction stage, it is crucial to note that unmatched yet confirmed tracks are continuously updated using the motion model until they either reach the maximum age threshold or receive a measurement update. These tracks are stored in memory and are utilized during the data association stage, where they attempt to obtain a measurement update. Unmatched detections are employed to initiate new tracks, while matched tracks are updated using the UDIMM estimator. Subsequently, the new states, class types, and IDs are conveyed to the simulation environment in order to visually update the object states.

7.3 Simulation Performance Parameters

This section introduces three different types of performance metrics to evaluate tracking results in the simulation environment. The performance results of the filters are provided with their Normalized Estimation Error Squared (NEES) and percentage of ground-truth states remaining within the $1\sigma, 2\sigma, 3\sigma$ bounds around estimation results. Additionally, the percentage of successful tracking over the simulation period and the success achievement of different filters are provided. In this regard, Kalman Filter based single&linear constant velocity (CV), constant acceleration (CA) motion models and besides simple IMM with three motion model (CV, CT-left, CT-right) and UDIMM with four motion model (CV, CA, CT-left, CT-right) are evaluated. The goal is to maintain the initial track number with no ID switch. For the notice, the age (hyperparameter of StrongSORT) of the multiple tracker, which is maximum number of misses before a track is deleted, is set to "30". Consequently, once a track is eliminated, it becomes impossible to allocate that particular ID to the track. The subsequent subsections provide a comprehensive explanation of the performance parameters which are calculated in result of 20 Monte Carlo simulations. The only uncertainty of the Monte Carlo simulations comes from the additionaly measurement uncertainity. The error graphs of filter estimation results are the mean values of Monte Carlo simulations. However, position, velocity and mode estimation graphs are provided for a single run without additional measurement uncertainty which is naturally eliminated using simulation environment. Measurement uncertainty values are determined by taking square root of the measurement noise covariance matrix where the derivation strategy is explained in Section 8.

Covariance Bound Coverage: Covariance bounds are derived from the estimator employed in the tracking system. These bounds are then visualized by plotting them around the estimation results and the zero mean for the residuals, considering different standard deviations such as $1\sigma, 2\sigma, 3\sigma$. The success rate of the estimation results is determined by the percentage of ground truth values falling within these bounds around the mean estimation value. This metric offers valuable insights into the stability of the estimator.

Track Percentage: The track percentage indicates the percentage of successfully

tracked objects, where the assigned track IDs of the objects are maintained throughout the simulation. In this case, the track IDs "1" and "2" are specifically targeted to be preserved for the first and second objects, respectively. The track percentage is then calculated as a performance metric, representing the proportion of successfully tracked individuals.

Normalized Estimation Error Squared: NEES is an excellent measure of the filter's performance and it is possible to use when the ground-truth states are available. NEES metric is a measurement evaluation technique commonly used in various fields, particularly in the context of object tracking and estimation tasks. The NEES metric assesses the accuracy of estimated states or parameters by comparing them to ground truth or reference values. It provides a quantitative measure of the discrepancy between the estimated values and the true values, enabling the assessment of the quality and reliability of the estimation process.

The NEES metric incorporates both the estimated error and uncertainty associated with the estimation. By normalizing the squared error by the uncertainty, the NEES metric takes into account the confidence or reliability of the estimation. This normalization enables fair comparisons between different estimation methods or algorithms, regardless of their inherent uncertainties or noise characteristics.

The NEES metric is particularly useful in evaluating the performance of tracking algorithms that involve state estimation over time. It provides insights into the accuracy and reliability of the estimated states, helping researchers and practitioners understand the effectiveness of their tracking systems. By analyzing the NEES metric, one can identify areas of improvement, compare different tracking approaches, and fine-tune algorithm parameters to enhance tracking performance. As a rule of thumb, the average $NEES$, given in equation (7.1), should be the dimension of the state vector which is 4 in this study, since the only positions and velocities in x-y directions are considered in calculation of NEES metric. When NEES values are less than 4, the estimations are considered underconfident, indicating a tendency to be conservative or pessimistic. On the other hand, when the NEES values are greater than 4, the estimations are labeled as overconfident, suggesting a tendency to be optimistic or excessively confident. The $\tilde{x}$ is the estimation error and P is the related

covariance matrix.

$$NEES = \tilde{x}^T P^{-1} \tilde{x} \tag{7.1}$$

Overall, the NEES metric serves as a valuable tool in quantitatively evaluating the quality of estimations in various applications, providing a comprehensive measure that takes into account both estimation errors and uncertainties.

Root Mean Square Error: To assess the effectiveness of state estimators, the estimation error is analyzed as a means of evaluation. The root mean square error (RMSE), a commonly employed metric for measuring accuracy, is well-suited for this task. It calculates the average estimation error across each time step. In order to attain a robust statistical significance, the RMSE is averaged over multiple Monte Carlo runs for a simulation spanning a certain number of time increments.

$$RMSE(\tilde{x}(t)) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\tilde{x}_1^i(t)^2 + ... + \tilde{x}_n^i(t)^2)} \tag{7.2}$$

The N states the total Monte Carlo runs and n is the dimension of the estimation error $\tilde{x}(t)$ which is total simulation time.

It is essential to calculate the RMSE exclusively for elements with the same unit, as it would be incorrect to combine, for instance, velocity and acceleration. Therefore, RMSE values are determined separately for each state or unit. In our case, the position and velocity are chosen as a state vector, the resulting metric is referred to as the RMSE of position and RMSE of velocity, respectively. In general, a smaller RMSE indicates a better performance of the state estimator.

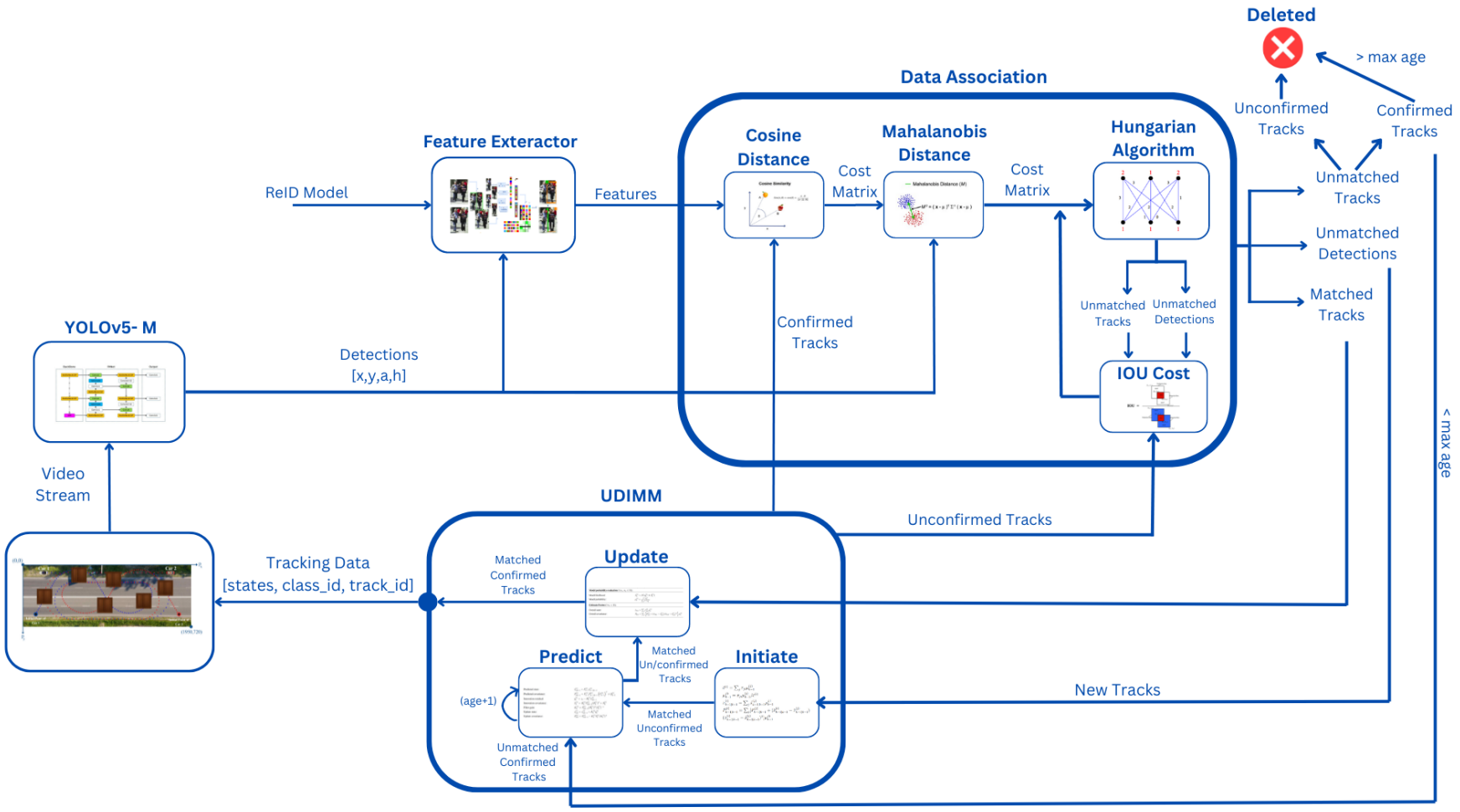


Figure 7.2: The Model Framework refers to a system that includes algorithms for multiple object detection and tracking, which interact with a two-dimensional simulation environment.

CHAPTER 8

EVALUATION OF THE RESULTS

This section utilizes four different types of filters to evaluate tracking results. In this regard, Kalman Filter based single and linear constant velocity (CV), constant acceleration (CA) motion models and besides simple IMM with three motion model (CV, CT-left, CT-right) and UDIMM with four motion model (CV, CA, CT-left, CT-right) are evaluated. The goal is to maintain the initial track number with no ID switch. The performance parameters are explained in Section 7. There are three simulation scenarios to be evaluated: 1) Single object with no occlusion, 2) Single object with occlusion, 3) Multiple objects with occlusion.

Besides, the evaluation of real aerial images is conducted using specific footage. In the final subsection, the performance results on the MOT17 validation dataset are presented for comparison purposes with the original StrongSORT and its variations.

Now, the outline of the settings of the motion model and filter parameters utilized for evaluating the mentioned scenarios are designed as following. The time step for all estimators is set to $dt = 1$. The measurement noise covariance matrix used for the filters is determined by analyzing the probability distribution of measurement errors. In total 50 simulations for different scenarios are conducted to ensure a precise distribution of differences between the measurements obtained from YOLOv5 and the ground truth values. Subsequently, the mean values of the error distributions of the bounding box's center coordinates, aspect ratio, and height are extracted to compute the measurement noise covariance matrix. In the process noise covariance matrix matrix, discrete Wiener process acceleration (DWPA) and the discrete white noise acceleration (DWNA) models are used, and the varying variance value is determined through trial and error. The values found in performance tables, such as Tables 8.1,

8.2, and 8.3, are calculated based on 20 Monte Carlo simulations. These simulations take into account additional measurement uncertainty, which is represented by white noise. The standard deviations of this noise are determined by the square root of the measurement noise covariance matrix.

Constant Velocity Model

The CV model is already incorporated in the original StrongSORT algorithm. Therefore, the parameters mentioned in the original article and code are utilized without any modifications.

Constant Acceleration Model

The CA model is represented with 10 states as stated in Section 5.3. For the corresponding state equation, the discrete Wiener process acceleration (DWPA) model [85] is used to derive process noise matrix, Q^{CA}. To do that $Q_discrete_white_noise()$ function from $filterpy$ [89] library is used. Additionally, it is realized that in the state equation of constant acceleration model, the only acceleration terms added to the positional states of the objects, in x and y direction. Because of that, the process noise matrix derived with the function is modified by considering the related acceleration terms. Thus, discrete white noise acceleration (DWNA) model is used for the states, $aspectratio$ and $width$ whereas DWPA model is used for the states, x and$y$. Process noise covariance matrices of DWNA and DWPA models are given in equation 8.1.

$$Q_{DWNA} = \begin{bmatrix} \frac{1}{4}dt^4 & \frac{1}{2}dt^3 \\ \frac{1}{2}dt^3 & dt^2 \end{bmatrix} \sigma_\omega^2, \quad Q_{DWPA} = \begin{bmatrix} \frac{1}{4}dt^4 & \frac{1}{2}dt^3 & \frac{1}{2}dt^2 \\ \frac{1}{2}dt^3 & dt^2 & dt \\ \frac{1}{2}dt^2 & dt & 1 \end{bmatrix} \sigma_\omega^2 \tag{8.1}$$

σ_ω^2 for the constant acceleration model is chosen as 3 px/s^2 for DWNA and px/s^3 for DWPA models. The measurement noise matrix is calculated as explained previously

$$R^{CA} = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 9 \end{bmatrix} \tag{8.2}$$

The constant acceleration model is initialized using the measurements for the first four states (x, y, a, h), while the unknown states $(v_x, v_y, v_a, v_h, a_x, a_y)$ are set to zero.

Interactive Multiple Model Estimator

This estimator comprises three motion models: CV, CT-right, and CT-left. The process model noise matrix, denoted as Q^{IMM}, is constructed using the DWNA approach, as explained in the CA model. However, since there are multiple motion models in this case, it is necessary to design multiple process noise covariance matrices. For the CV model, σ_ω^2 is set at 2 px/s^2, while for the CT-left and CT-right models, σ_ω is selected as 3 px/s^2.

The measurement noise matrix is chosen the same as for three model which is given in equation (8.3).

$$R^{IMM} = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \tag{8.3}$$

The transition probability matrix for the IMM filter $\pi_{ji}^{[1]}$ is chosen as

$$\pi_{ji}^{[1]} = \begin{cases} 0.99, & if \quad i = j \\ 0.005, & if \quad i \neq j \end{cases} \tag{8.4}$$

And the mode probability, μ_k is initiated with the following values,

$$\mu_k = [0.33, 0.33, 0.33]$$

The turn rates in CT-left and CT-right motion models are taken as constant with the value of 5 and -5 deg/s, respectively.

Unequal Dimension Interactive Multiple Model Estimator

This estimator comprises four motion models: CV, CA, CT-right, and CT-left. The

process model noise matrix, is constructed using the DWNA approach for the CV, CT-right, and CT-left motion models and using DWNA and DWPA approach for the CA motion model as explained in CA section. Since there are multiple motion models in this case as well, it is necessary to design multiple process noise covariance matrices. For the CV model, σ_ω^2 is set at 2 px/s^2, while for the CT-left and CT-right models, σ_ω^2 is selected as 3 px/s^2 and for the CA model σ_ω^2 is chosen as 5 px/s^3.

The measurement noise matrix is chosen the same as for three model which is given in equation (8.5).

$$R^{UDIMM} = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \tag{8.5}$$

Transition probability matrix UDIMM $\pi_{ji}^{[2]}$ is chosen as

$$\pi_{ji}^{[2]} = \begin{cases} 0.97, & if \quad i = j \\ 0.01, & if \quad i \neq j \end{cases} \tag{8.6}$$

And the mode probability, μ_k is initiated with the following values,

$$\mu_k = [0.25, 0.25, 0.25, 0.25]$$

The turn rates in CT-left and CT-right motion models are taken as constant with the value of $5\ deg/s$ and $-5\ deg/s$, respectively.

The uniform distribution parameters $[a, b]$ which identify the augmented states are chosen for the underlying scenario where the maximum acceleration is assumed to be $0.5\ px/s^2$. Therefore, $[a, b]$ are chosen as $b = -a = 0.5\ px/s^2$ for a_x and a_y.

In the gating threshold section, Mahalanobis distance calculation is performed using mixed x_{prior} and P_{prior} values. This enables the utilization of multiple modes and facilitates gating based on the probability of the multiple models, leading to an improvement in data association.

8.1 Scenario 1: Single Object with No Occlusion

In this scenario, it is only being taken into account to keep track of single object when there is no occlusion. Simulating a single object simplifies the problem by reducing the ID switch possibility of similar looking objects belonging to the same class. There are three different type of motion models as discussed in Section 5. The tracking performance is assessed using a single CV and CA motion model. Additionally, the tracking performances are evaluated by employing multiple model mixing methods that mix multiple models (CV, CA and CT) with equal and unequal dimensions. The illustration and ground truth modes of scenario 1 can be seen in Figure 8.1.

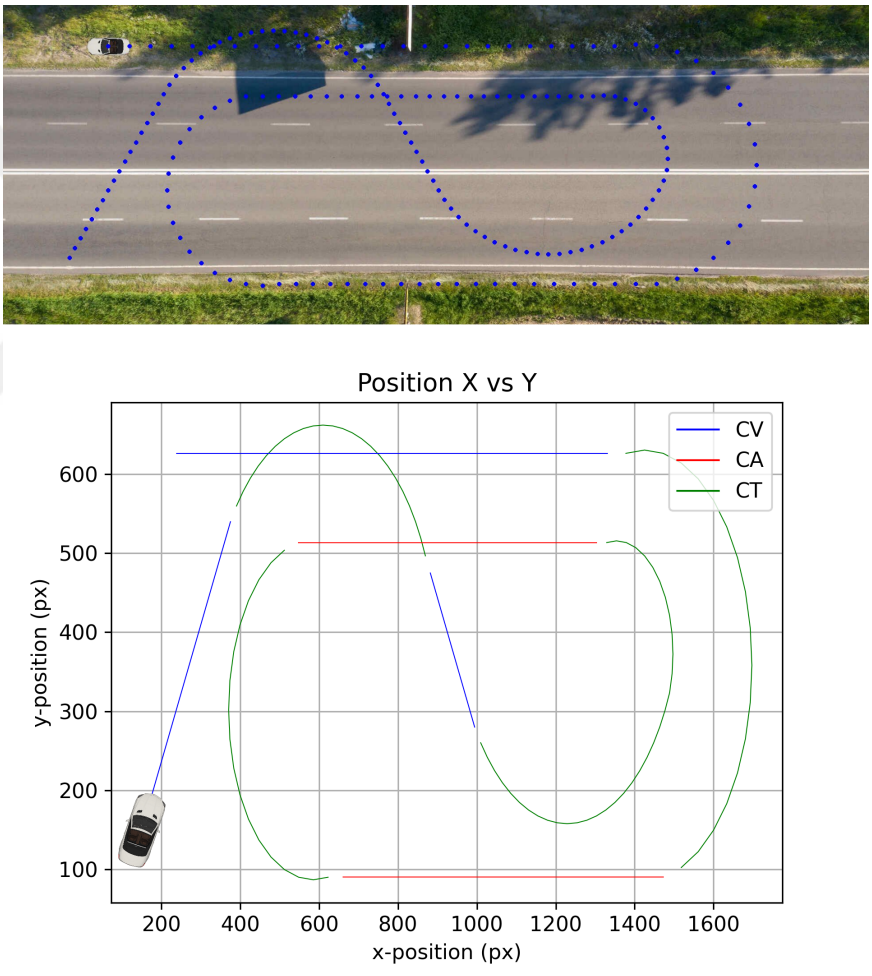


Figure 8.1: Simulation and ground truth modes of the scenario 1.

In the position and velocity estimation figures, solid blue line shows the ground truth

trajectory. Scatter points represents the estimation results and different colours belongs to the different tracked IDs of a moving object. Dashed line around the estimation results is the covariance bounds with considered standard deviation.

Figure 8.2 (left-column) represents the position estimation results when employing the CV motion model. Dashed line around the estimation results is the covariance bounds with 3σ standard deviation. The loss of tracking for a single object is evident. The first occurrence of an ID switch happens when the object transitions from the CT to the CA model, approximately at the 130 second mark. The second ID switch is occurred around 170 seconds where the object change its mode from CT to CV model. Additionaly, it can be seen that covariance is getting larger where the ID switch occurs.

By employing the CA model in place of CV, the estimations and tracking achieve enhanced accuracy and continuity. As can be seen in the Figure 8.2 (right-column), tracking is maintained for 173 second until the last mode change from CT motion to CV motion. All the maneuvers and accelerations are captured by the filter, the states of the object is accurately estimated with comparably low covariance with respect to the Kalman Filter with CV motion model.

The IMM method exhibits excellent performance in terms of estimation and tracking outcomes. The positions in the x and y directions are estimated with high precision and minimal error, as demonstrated in Figure 8.3 (left-column). Throughout the simulation, the object's identity is preserved, indicating the successful results of the IMM method, particularly in scenarios without occlusions.

The UDIMM method produces position estimation results similar to those of the IMM method when there are no occlusions. However, the UDIMM method exhibits lower covariance values, as illustrated in Figure 8.3 (right-column). As a result, UDIMM achieves significantly improved and more reliable performance, particularly within the 1σ covariance bounds.

The upper-left section of Figure 8.4 depicts the trajectory variation of an object when the CV motion model is utilized. Particularly, in this depiction, it is evident that none of the tracklets within the time range of 155 seconds to 170 seconds have been con-

firmed. This is because, in order for tracking to begin, at least three measurements should match with the tracklet during the data association stage. However, in this situation, the filter estimations become inconsistent during the maneuvering motion, and the upcoming measurements do not match, resulting in the initiation of new tracklets instead of using them to update the filters. Consequently, no object has been tracked since none of the tracklets have been confirmed within this specific time frame.

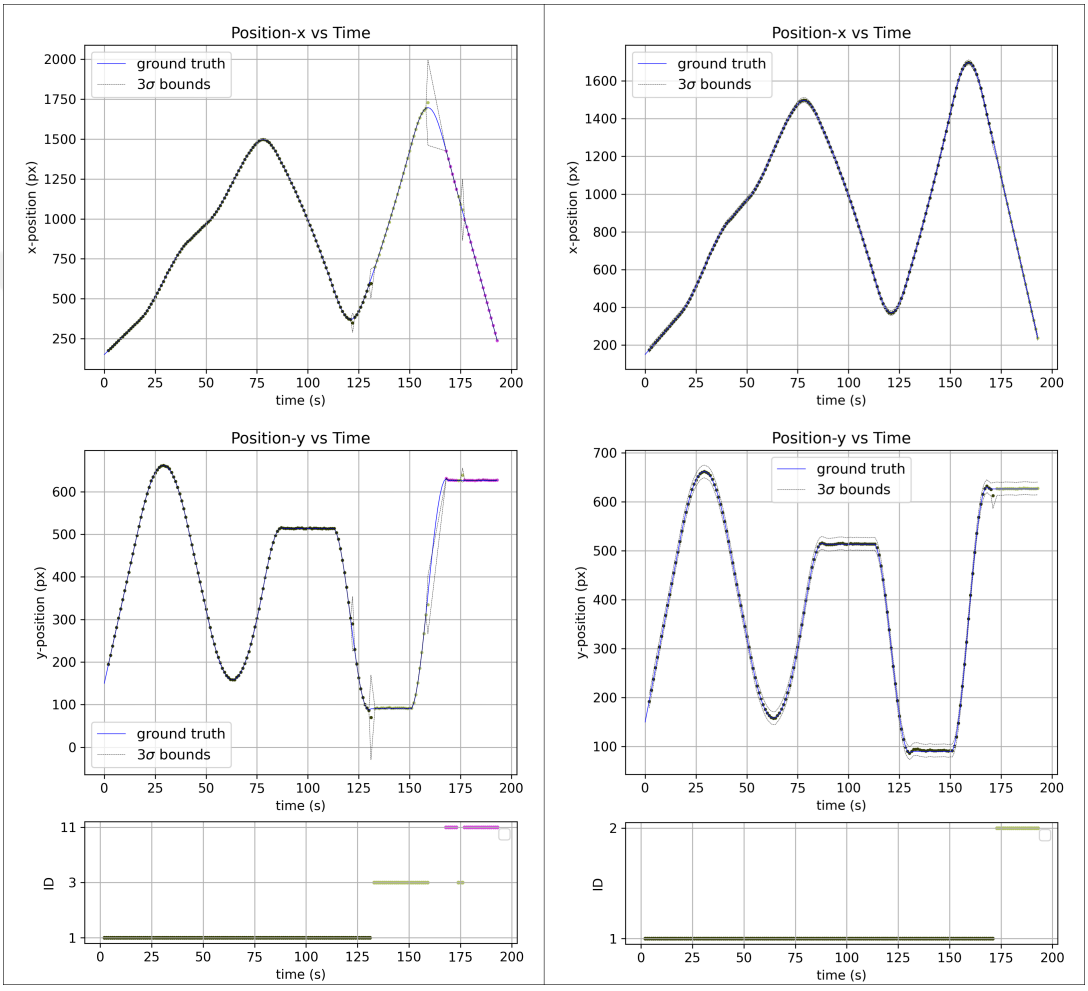


Figure 8.2: Position estimation results in x and y directions for CV model (left), CA model (right) in scenario 1.

Figure 8.4 (upper-right) illustrates the variation in the two-dimensional trajectory of an object over time when CA motion model is implemented. Unlike the CV model, the tracking remains uninterrupted, resulting in continuous estimation results with-

out any missing data. However, in the final mode switch of the ground truth object (represented by light green dots in Figure 8.4), an ID switch occurs and the track is lost.

Utilizing multiple model estimators, including IMM and UDIMM, yields highly satisfactory outcomes when estimating the trajectory of objects, as illustrated in Figure 8.4 (lower-left and lower-right, respectively). The tracking remains consistent throughout the simulation, without any instances of ID switch occurring.

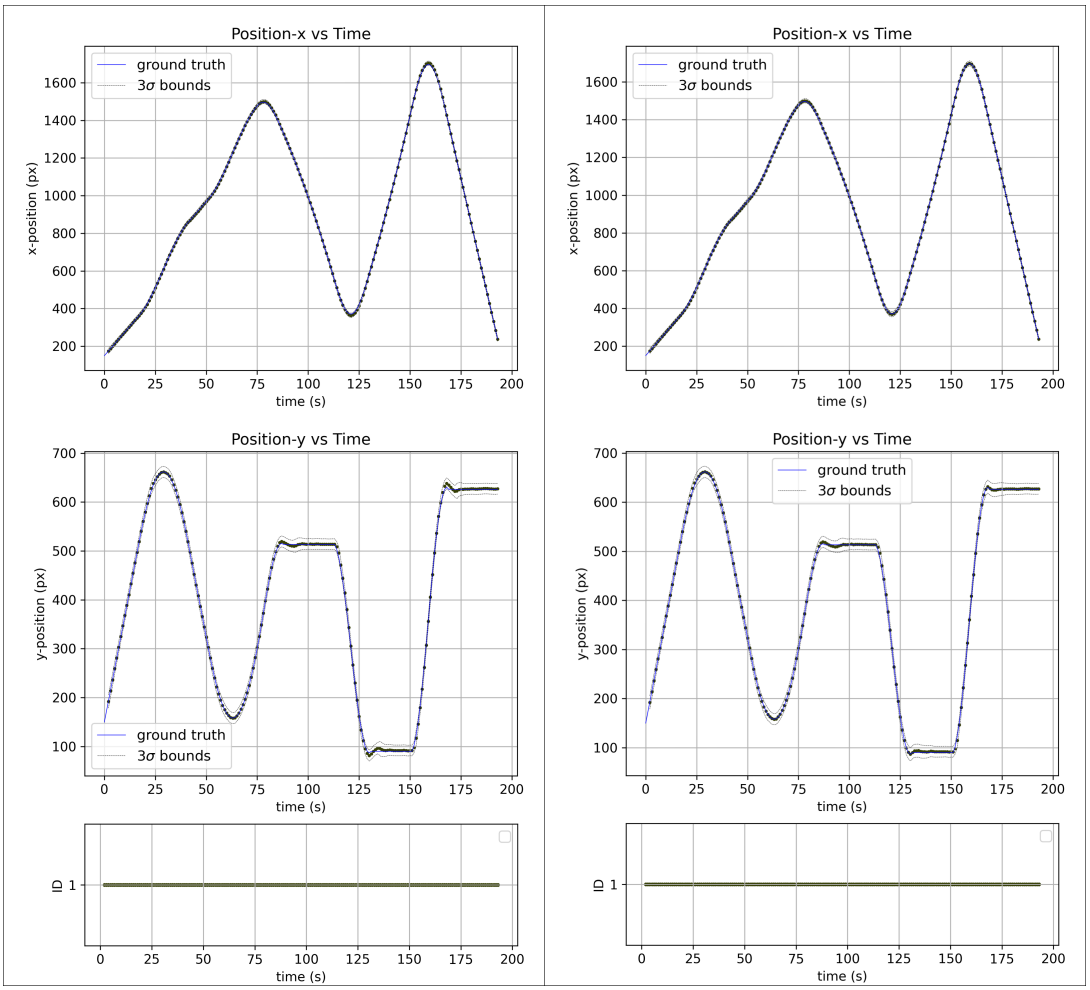


Figure 8.3: IMM estimator (left), UDIMM estimator (right) in scenario 1.

The first row of Figure 8.5 presents the results of velocity estimation in the x and y axes, obtained by implementing the Kalman filter with the CV motion model. Dashed line around the estimation results is the covariance bounds with 1σ standard deviation.

It can be concluded that using the CV motion model yields unsatisfactory outcomes for maneuvering and accelerating motions. The main reason is that the CV model lag the input (especially when the target maneuvers or accelerate) because the filter is not modeling acceleration. When there is a mismatch between the filter motion model and the ground truth motion model, it leads to significant increase in error magnitude, which can undermine the reliability of the tracking.

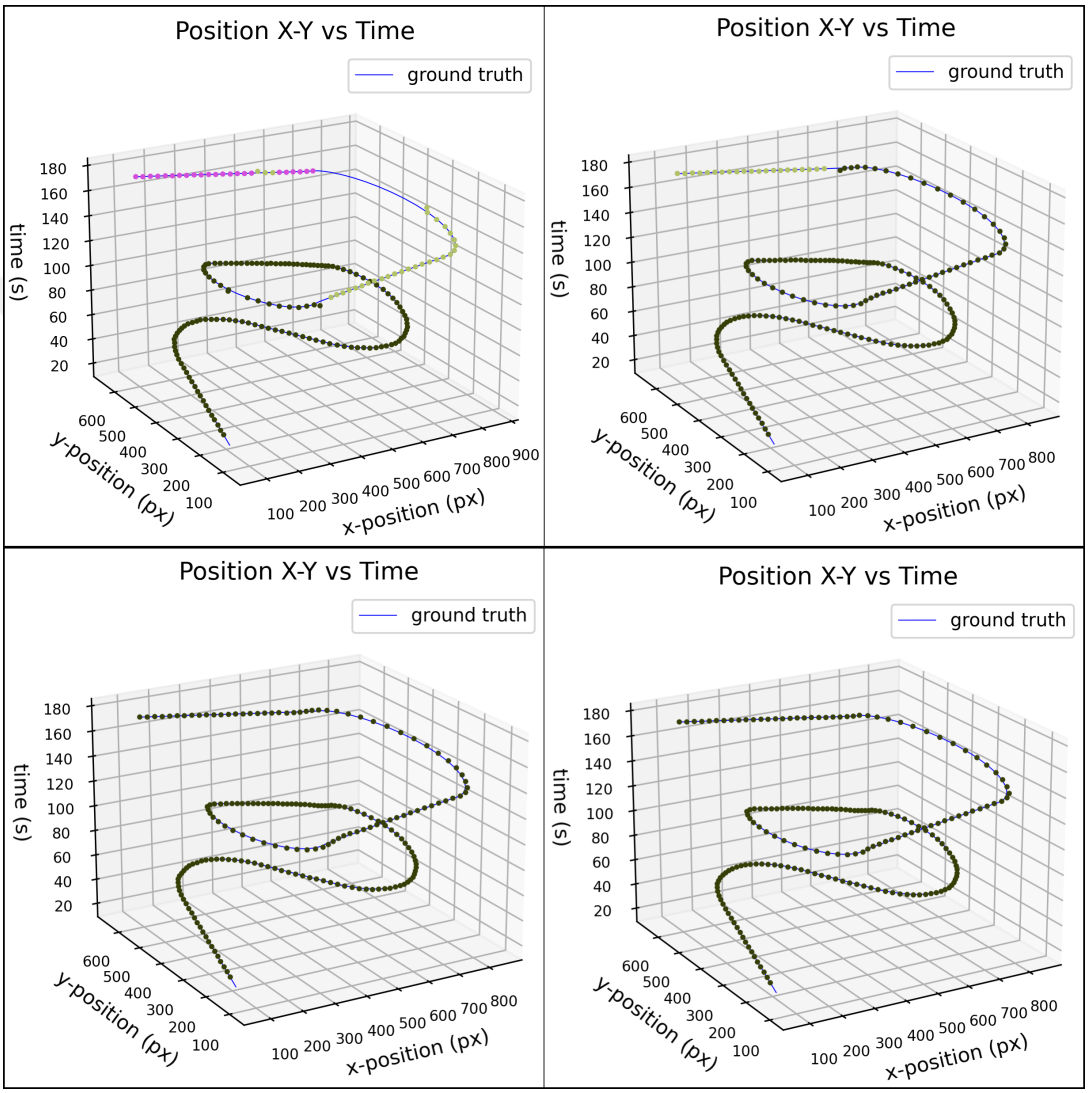


Figure 8.4: 2D trajectory change of the object with respect to time for CV model (upper-left), CA model (upper-right), IMM estimator (bottom-left), UDIMM estimator (bottom-right) in scenario 1.

It is deduced that in Figure 8.5 (first row), the velocity estimations are not capable

of tracking the ground truth velocity, especially at maneuvering motion which causes a significant direction of a velocity change. Figure 8.5 (second row) displays the results of velocity estimation in both the x and y directions, obtained by implementing the Kalman filter with the CA motion model. It can be inferred that the velocity is accurately estimated even during maneuvering behaviors with the narrow covariance bounds. Only minor peaks are observed when there are abrupt changes in velocity direction, such as around times $t = 85$ seconds, $t = 130$ seconds, $t = 160$ seconds, and $t = 170$ seconds. Multiple model estimators, such as IMM and UDIMM, exhibit excellent performance in estimating the velocity states along the horizontal (x) and vertical (y) axes. Using the multiple model filters improves the accuracy of velocity estimation with tight covariance bounds and small overshoots during sudden changes in velocity direction as can be seen in Figure 8.6.

The Figure 8.7 demonstrates the mode probabilities employed to combine the estimated states from multiple filters. The below figures with blue line denotes the ground truth modes changes of the object with respect to time. The IMM estimator effectively captures changes in motion modes without any delay. However, during the time interval from $t = 95$ seconds to $t = 115$ seconds, the IMM estimator estimates the results using the CV motion model instead of the CA motion model, as the CA motion model is not included in the IMM estimator. And since the magnitude of the acceleration, $a_x = 0.5px/s^2$ and $a_y = 0.5px/s^2$, is not relatively high compared to velocity, IMM estimator provides accurate results in terms of estimating the position and the velocity. However, the UDIMM estimator faces difficulty in accurately estimating the mode when the ground truth mode transitions to a CA motion model. This confusion can be attributed to two potential reasons. Firstly, the available acceleration samples, which consist of 25 and 19 time steps for two different mode durations, are insufficient to significantly enhance the likelihood of the CA motion mode. Consequently, the CA mode is mistakenly estimated as the CT mode. Secondly, there is a similarity between the CA and CT motion models, particularly when the CT modes have very short durations, as observed during specific intervals such as between 113 and 130 seconds, and between 151 and 168 seconds. The research [93] demonstrates that in various scenarios, UDIMM accurately identifies the CA mode without any ambiguity when the duration is sufficiently long. Furthermore, UDIMM

allows for immediate transitions between modes, accurately reflecting the true mode of the object.

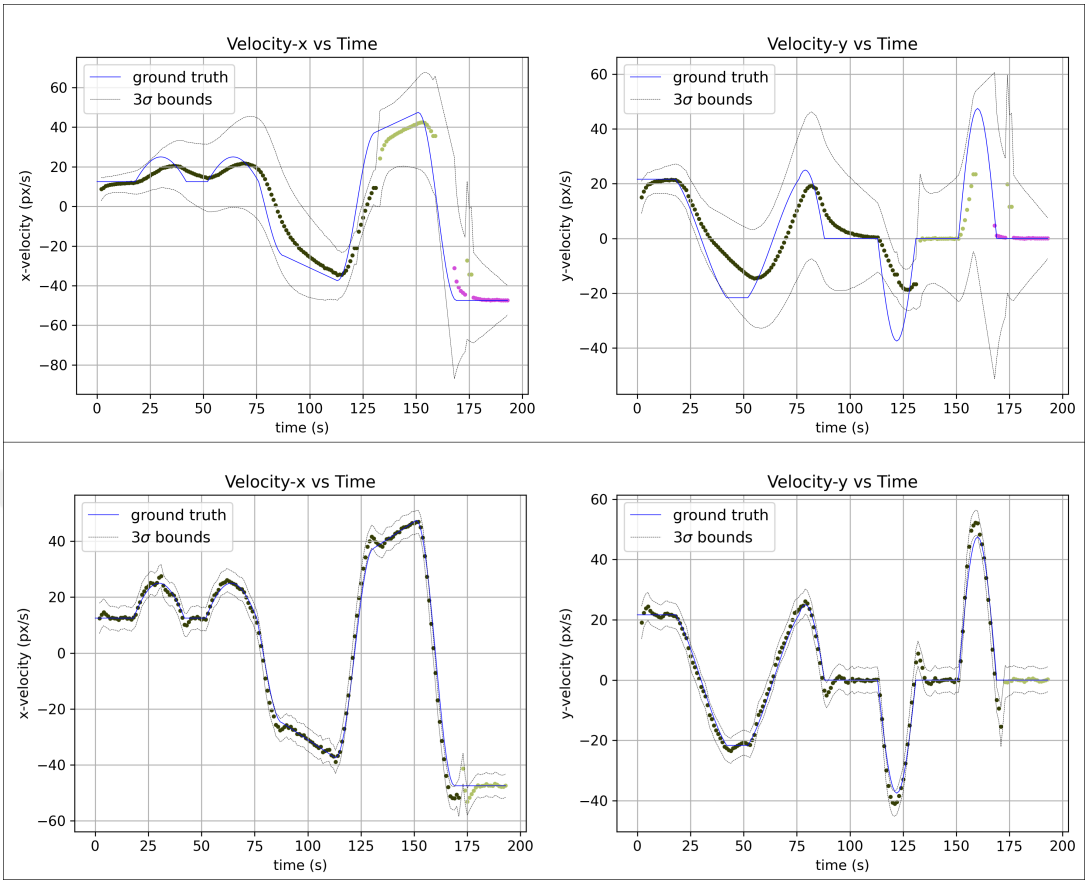


Figure 8.5: Velocity estimation results in x and y directions for CV model (first row), CA model (second row) in scenario 1.

Figure 8.8 illustrates the mean error values resulting from Monte Carlo simulations, specifically regarding total position and total velocity. From the graph, it can be observed that the mean position estimation error is generally small and acceptable for all estimators in terms of tracking, with the exception of the CV motion model. The CV model exhibits significant position errors at certain time points, such as $t = 130, 160, 175$ seconds, indicating a consistent total velocity error when utilizing the CV motion model. The estimation results obtained from the UDIMM estimator lie between the results of the CA and IMM estimators. This is because the UDIMM estimator incorporates the combined motion models used in both the CA and IMM

filters.

In Table 8.1, the performance of filters is presented in relation to the performance parameters discussed in Section 7.3. The evaluation of filter performance is based on the tracked tracklets, which are visualized in the simulation environment. However, tracklets that are updated in the memory but not actively tracked are excluded from consideration. Table 8.1 reveals an interesting observation for the scenario 1 when there is no occlusion exists. The CA model demonstrates excellent performance in estimating the positional states of tracklets themselves, particularly when the ID switch of an object is ignored.

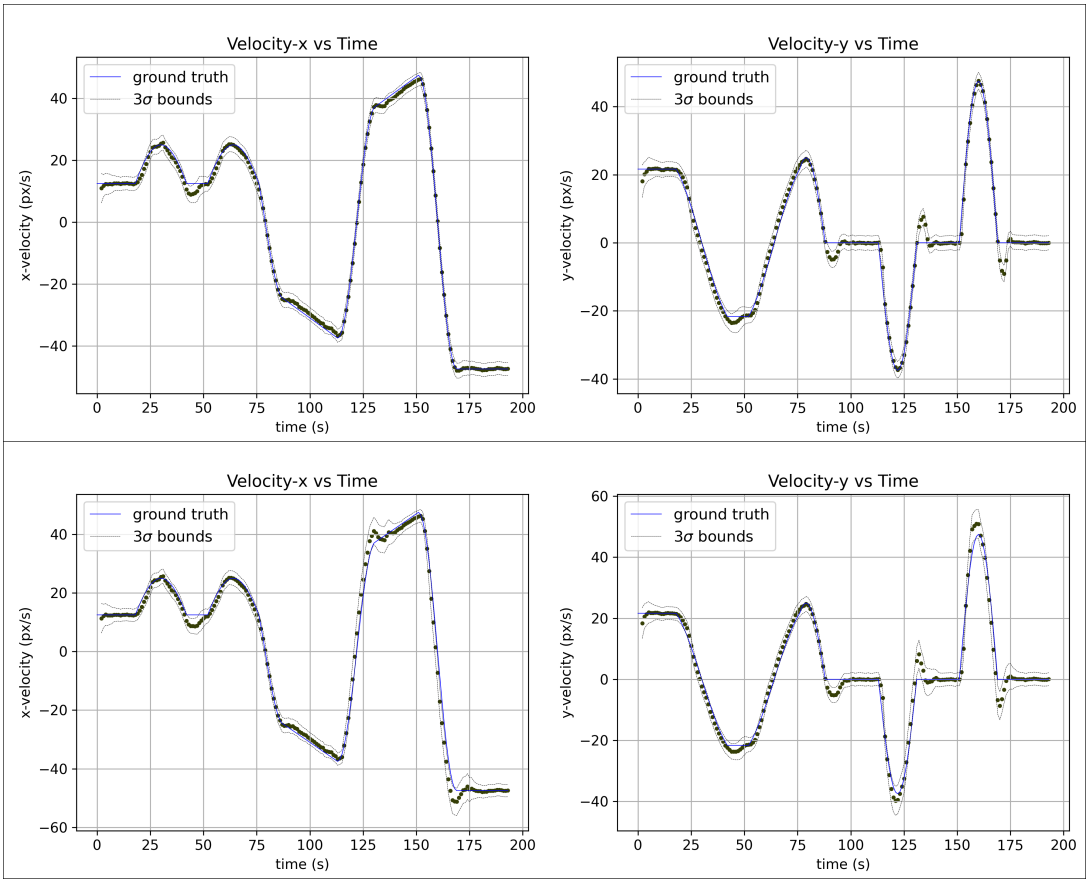


Figure 8.6: Velocity estimation results in x and y directions for IMM estimator (first-row), UDIMM estimator (second-row) in scenario 1.

The CA model, IMM and UDIMM estimators achieve the similar NEES values which is supposed to be 4. However, in terms of velocity estimations, multiple model es-

timators have better accuracy within the 1σ bound. Furthermore, when taking into account the RMSE of position and velocity, it is observed that the CA model exhibits better accuracy in predicting the position along the x and y directions. However, employing multiple model filters yields superior outcomes when it comes to forecasting the dimensions (width and height) of the bounding box. Moreover, the estimation outcomes obtained from employing multiple model filters perform reduced covariance values, which explains their improved performance in tracking percentage.

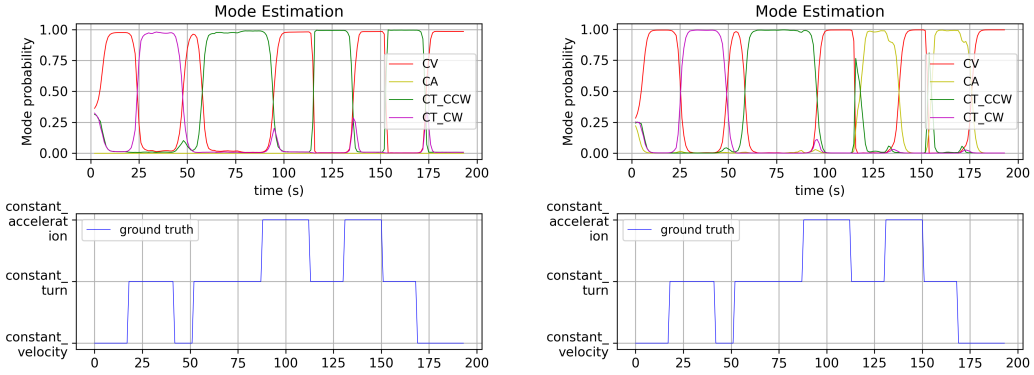


Figure 8.7: Mode probability results of IMM (left) and UDIMM (right) estimators for scenario 1.

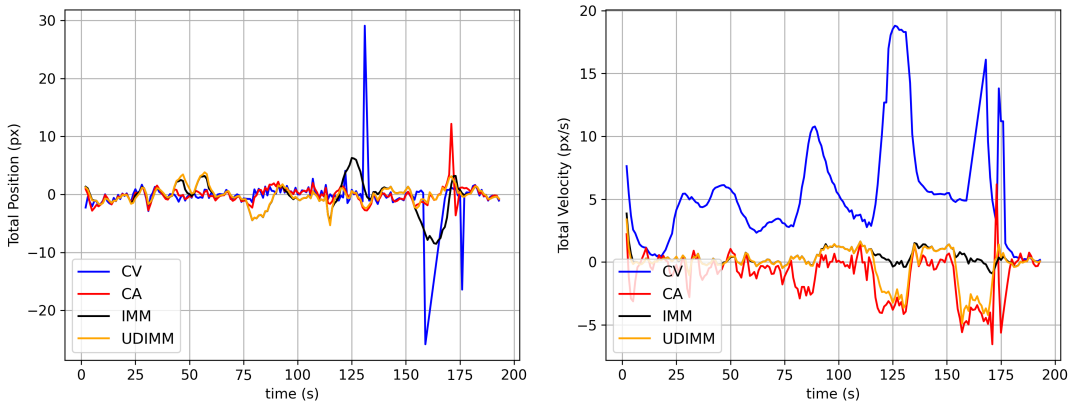


Figure 8.8: Mean Errors of Total Position (left) and Total Velocity (right) for scenario 1.

Multiple model filters provide balanced results between estimation error and covariance values. Overall, UDIMM and IMM estimators performs consistent and reliable results while dealing with mixing the multiple models. CA, IMM and UDIMM filters achieve close results to the desired NEES ($= 4$) value where the NEES is diverged with the CV model due to large estimation errors compared to covariance value. ID switch counts and tracking results of filters with different motion models for a single run without including measurement uncertainty are obtained as,

- CV model: 2 ID switches, tracking is not sustained
- CA model: 1 ID switches, is not sustained
- IMM filter: 0 ID switches, tracking is **sustained**
- UDIMM filter: 0 ID switches, tracking is **sustained**

Table 8.1: Performance of Filters in Scenario 1

Filter	State	1σ | 2σ | 3σ (%) ↑	RMSE POS.(px) ↓	RMSE VEL.(px/s) ↓	NEES($= 4$)	Track Perc.(%) ↑
CV	Est. Pos. (px)	14.23 | 22.65 | 27.82	58.85	37.44	264.24	61.04
	Est. Vel. (px/s)	78.12| 84.33 | 89.95				
CA	Est. Pos. (px)	**88.62** | **92.43** | 96.25	**34.23**	**14.78**	3.39	82.17
	Est. Vel. (px/s)	89.23 | **95.49**| **98.95**				
IMM	Est. Pos. (px)	79.22 | 88.46 | 94.83	42.36	21.86	**3.88**	96.43
	Est. Vel. (px/s)	89.33 | 91.32 | 95.75				
UDIMM	Est. Pos. (px)	84.82 | 91.35 | **97.83**	35.34	15.17	3.75	**98.86**
	Est. Vel. (px/s)	**90.69** | 95.45 | 98.24				

8.2 Scenario 2: Single Object with Occlusion

In this particular scenario, the focus is solely on monitoring a single object in case of occlusion. The simplification of the problem is achieved by simulating the tracking of a single object, which reduces the likelihood of switching the identification of similar-looking objects within the same class similar to the approach employed in scenario 1. The tracking performance is evaluated using both a single CV and CA motion model. Furthermore, the tracking performance is assessed through the utilization of multiple model mixing methods, which involve combining multiple models (CV, CA, and CT)

with equal and unequal dimensions. Figure 8.9 provides visual representations of the illustration and ground truth modes and trajectory with occlusions for scenario 2. The scatter points depict the object's trajectory, while square boxes serve as occlusion objects or interruptions to the measurements obtained by the YOLOv5 object detector.

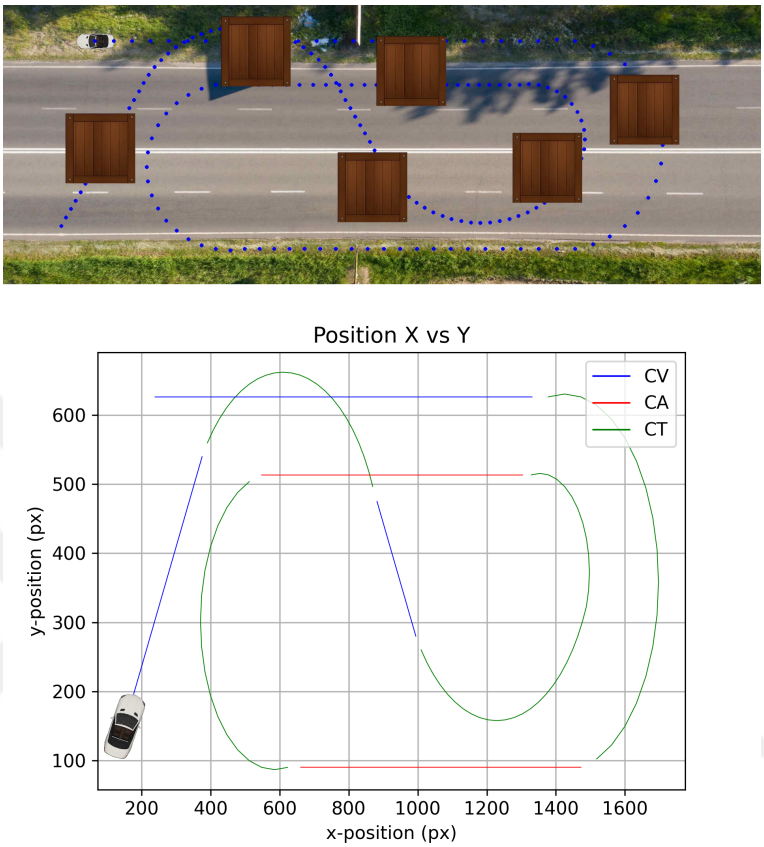


Figure 8.9: Simulation and ground truth modes of the scenario 2.

Figure 8.10 and 8.11 represents the position estimation results for four different filters. Dashed line around the estimation results is the covariance bounds with 3σ standard deviation. Figure 8.10 (left-column) shows the position estimation results when employing the CV motion model. It is evident that in a densely occluded scenario, the occurrence of ID switching and tracking loss is more frequent when utilizing the CV model. When the measurement is interrupted, the covariance bounds expand during the prediction stage of the filter, where only motion model predictions from the Kalman Filter are used. Once the measurement becomes available again, it is unable to be associated with the previous tracklets due to the cost matrix exceeding the gating threshold. As a result, a new track is initiated with this measurement. During the

track confirmation period, which requires at least three measurement associations, estimation data is not available, leading to data gaps such as at time $t = 13 - 14$ seconds in Figure 8.10 (left-column). The limitations of using a single model, such as CV, become apparent when observing the y-axis in Figure 8.10 (left-column) at time $t = 35$ seconds. During the object's occlusion in the maneuvering state, the CV prediction model deviates from the actual curved path.

To consider the CA motion model in Figure 8.10 (right-column), it is obvious that tracking of an object is not maintained due to multiple ID switches. The acceleration term in the model causes large covariance at every occlusion so that the ID of an object changes at every occlusion as can be seen in the Figure. Because of that CA model is not a good choice for densely occluded scenarios, in contrast, it provides very accurate estimation results within the tracklet which shows its benefits as in scenario 1. The presence of the acceleration term in the model leads to significant covariance during occlusions, resulting in frequent changes (at every occlusion in scenario 2) in the object's ID, as observed in the Figure 8.10 (right-column). Consequently, the CA model is not an ideal choice for densely occluded scenarios. On the other hand, the CA model demonstrates its advantages by providing highly accurate position estimation results within the tracklet, as exemplified in scenario 1.

The IMM estimator significantly decreases the occurrence of ID switching for the object and manages to sustain tracking until $t = 165$ seconds, achieving a certain level of success which is depicted in Figure 8.11 (left-column). By utilizing multiple models, the covariance terms in occluded scenarios are decreased in magnitude compared to using a single CV or CA model. One drawback of the multiple model filter is its susceptibility to being deceived by tilted measurements that occur just before the object disappears from the scene. These tilted measurements can potentially mislead the multiple model filter into switching its mode prematurely, prior to the occurrence of occlusion. When the measurement is tilted, multiple model algorithms keep propagating the prediction models according to likelihood of the last estimation just before the occlusion. Therefore, predictions start diverging from the ground-truth path during prediction period and if this period is long enough, tracking of the object is probably lost. It is certain that some measurements pertaining to the upcoming motion mode need to be received right before the occlusion begins, so that the IMM

filter can update its prediction models accordingly even in the absence of any available measurements. The measurement misleading, in which the the CT motion mode is more likely before the occlusion, can be seen in Figure 8.11 (left-column) at times $t = 100$ seconds and $t = 175$ seconds.

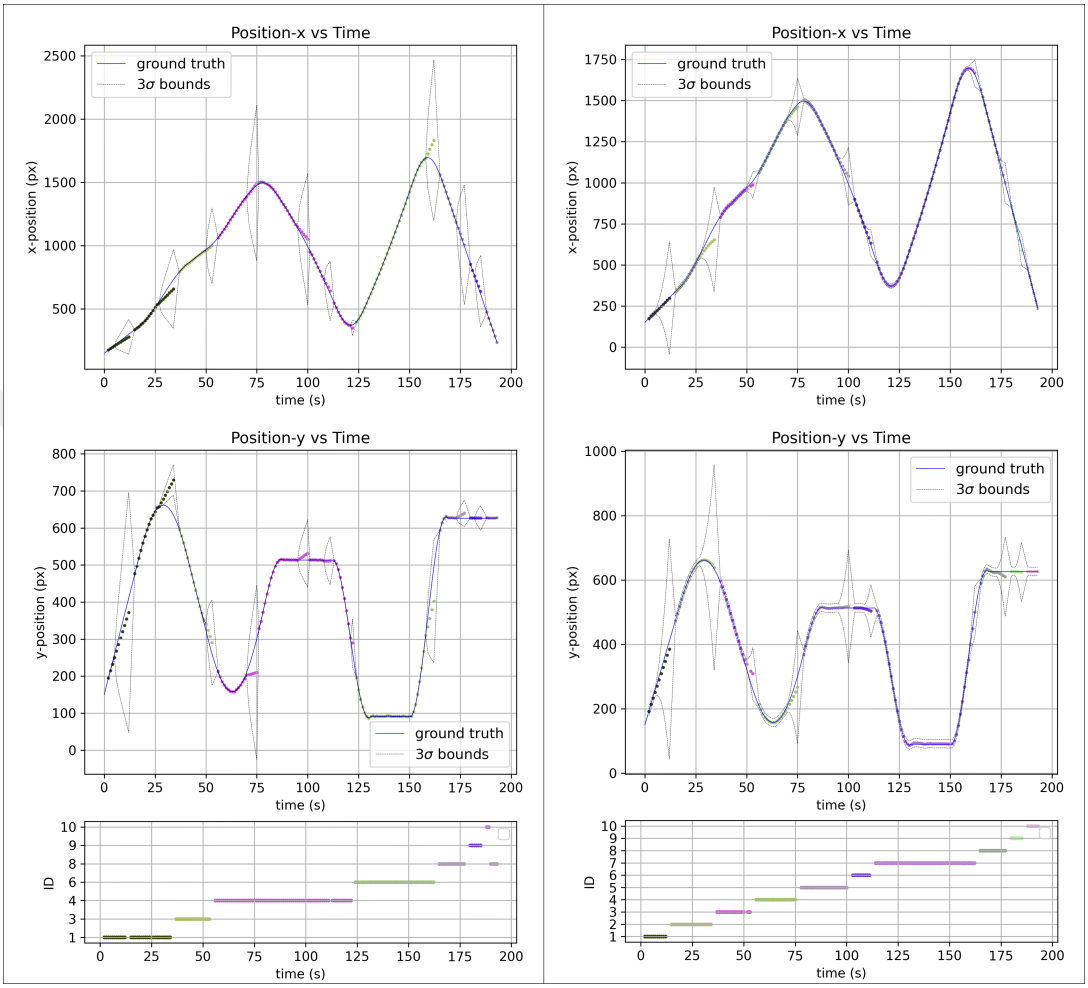


Figure 8.10: Position estimation results in x and y directions for CV model (left), CA model (right) in scenario 2.

The issue of measurement misleading also arises with the UDIMM estimator. However, the UDIMM estimator outperforms other filters in occluded scenarios. Figure 8.11 (right-column) displays the position estimation results of the UDIMM estimator. Dealing with measurement misleading is a challenging problem, especially during prolonged periods of occlusion. In scenario 2, this problem occurs at times $t = 35$ seconds, $t = 55$ seconds and $t = 75$ seconds, resulting in ID switches similar to

other filters. Nevertheless, due to the improved prediction models of the UDIMM estimator, the correct ID of the object is recovered before the tracklet's age reaches the maximum threshold. There is no ID switch at time $t = 100$ seconds because the cost matrix remains below the gating threshold. In short, UDIMM estimator maintained the tracking and ID of an object throughout the scenario 2.

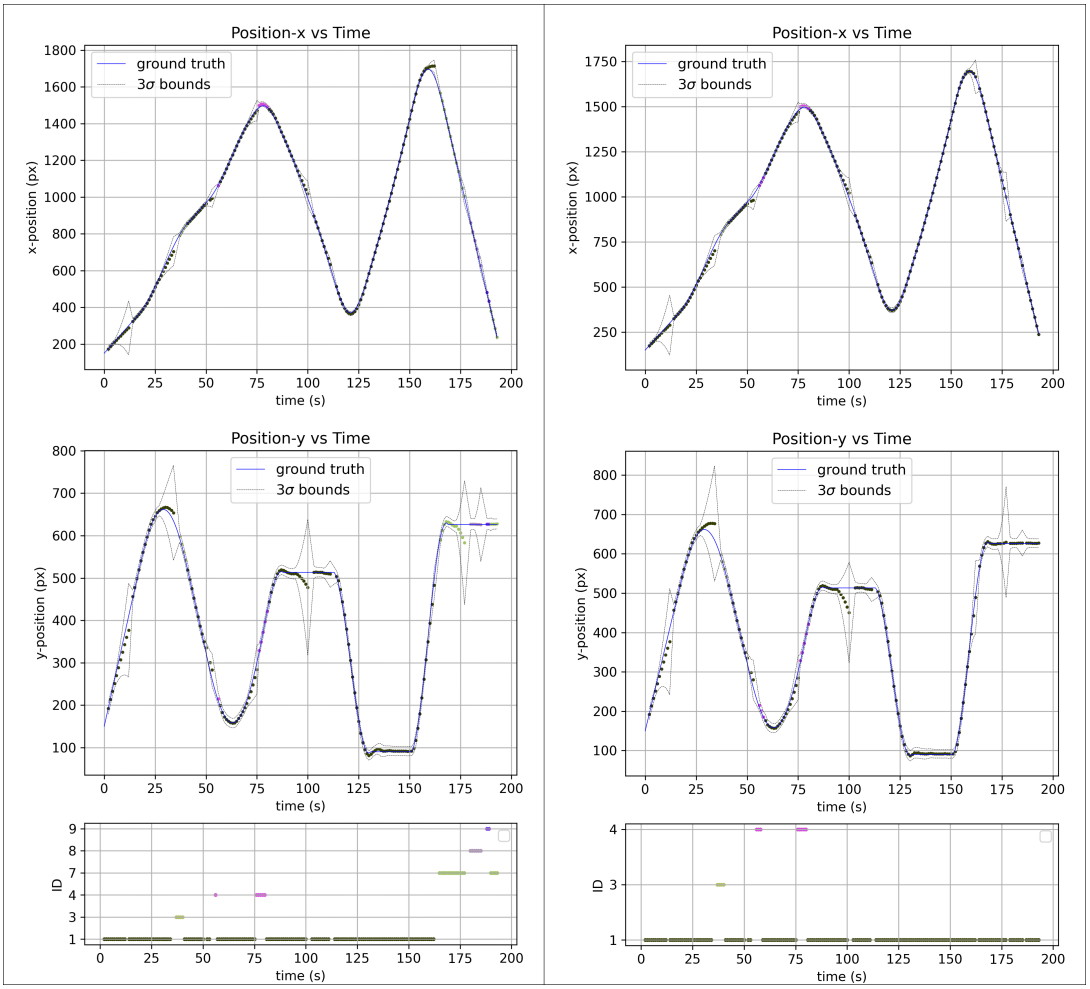


Figure 8.11: IMM estimator (left), UDIMM estimator (right) in scenario 2.

In the upper-left section of Figure 8.12, the trajectory variation of an object is shown when the CV motion model is employed. This depiction highlights cases where tracklets are not confirmed, resulting in unavailable estimation data, especially in manevuering motions. As a result, aside from the frequent ID switching, the utilization of the CV motion model leads to a reduction in the estimation data provided by the tracker, as no object is being tracked. This situation can have significant implications

for guidance or path planning algorithms [94].

Figure 8.12 (upper-right) illustrates the variation in the two-dimensional trajectory of an object over time when CA motion model is implemented. Unlike the CV model, the estimation of tracklets are more accurate and gives reliable estimations when measurement is available. However, in case of occlusions, multiple ID switch occurs and the track is lost.

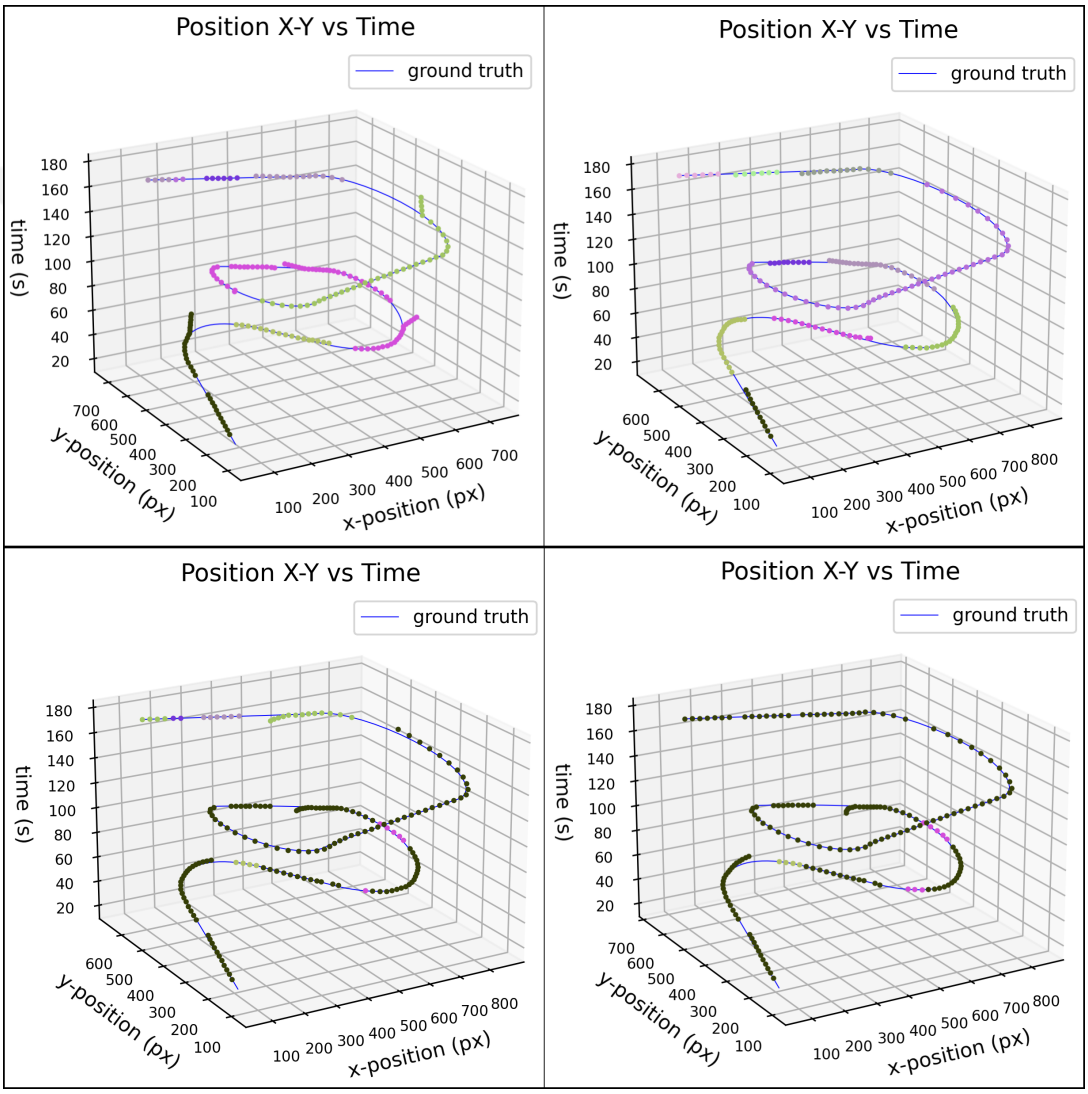


Figure 8.12: 2D trajectory change of the object with respect to time for CV model (upper-left), CA model(upper-right), IMM estimator (bottom-left), UDIMM estimator (bottom-right) in scenario 2.

To emphasize a crucial point, after the time $t = 150$ seconds, IMM becomes confused by mixing the modes, assigning higher probabilities to incorrect modes while decreasing the likelihood of the actual mode at $t = 165$. This is the moment when an ID switch takes place. Following that time, IMM encounters difficulties, leading to multiple ID switches. Conversely, UDIMM exhibits greater accuracy in mode estimation. While the estimated mode may not be the actual mode, UDIMM avoids confusion with likelihoods, resulting in more robust estimation outcomes and maintaining the object's ID.

Utilizing multiple model estimators, including IMM and UDIMM, yields highly satisfactory outcomes when estimating the trajectory of objects, as illustrated in Figure 8.12 (lower-left and lower-right, respectively). For the UDIMM estimation the tracking remains consistent throughout the simulation whereas the multiple ID switch occurs when utilizing the IMM estimator at the final maneuver.

The results of velocity estimation in the x and y axes, obtained by applying the Kalman filter with the CV motion model, are depicted in the first row of Figure 8.13. The dashed line surrounding the estimation results represents the covariance bounds with a 1σ standard deviation. It can be concluded that utilizing the CV motion model produces unsatisfactory outcomes for maneuvering and accelerating motions in occluded scenarios, failing to accurately represent the true velocity of the object. When there is a mismatch between the motion model used in the filter and the actual motion model of the object, it results in a significant increase in the magnitude of the covariance, which undermines the reliability of the tracking. And when the occlusion is involved, the performance of CV model deviates from the actual reality. In Figure 8.5 (first-row), it is evident that the velocity estimations are incapable of accurately tracking the ground truth velocity, particularly during maneuvering motions that involve a significant change in direction. This situation leads to ID switches and poor estimation results with large covariance bounds.

Utilizing CA motion model, reduce the error of estimations with low covariance values compared to CV model and provides acceptable estimation results in occluded scenarios but within the tracklet. However, since the ID switch is prevalent with the usage of CA model, the velocity estimations may not be reliable for the path planning

or guidance applications.

IMM and UDIMM demonstrate outstanding performance when estimating velocity states in the horizontal (x) and vertical (y) axes. The precision of velocity estimation is enhanced with low covariance values and minimizing overshoots during abrupt changes in velocity direction, as evident in Figure 8.14. Nonetheless, there are instances where the multiple model estimator is misled by the measurements, resulting in a tilted velocity direction and deceptive outcomes, such as at time $t = 100$ seconds.

The Figure 8.15 demonstrates the mode probabilities employed to combine the estimated states from multiple filters. Blue line in Figure 8.15 denotes the ground truth modes changes of the object with respect to time. In scenario 2, the mode estimation accuracy is decreased due to more occlusion and sudden measurement interruptions. In this case, it is realized that sudden measurement arrivals and interruptions confuse the IMM and UDIMM filters while mixing the mode probabilities. This confusion with mixing the mode probabilities cause ID switch of the tracked object as in between times $t = 25$ seconds and $t = 60$ seconds.To emphasize a crucial point, after the time $t = 150$ seconds, IMM becomes confused by mixing the modes, assigning higher probabilities to incorrect modes while decreasing the likelihood of the actual mode at $t = 165$. This is the moment when an ID switch takes place. Following that time, IMM encounters difficulties, leading to multiple ID switches. Conversely, UDIMM exhibits greater accuracy in mode estimation. While the estimated mode may not be the actual mode, UDIMM avoids confusion with likelihoods, resulting in more robust estimation outcomes and maintaining the object's ID, ultimately leading to better tracking result.

Figure 8.16 illustrates the mean error values resulting from Monte Carlo simulations, specifically regarding total position and total velocity. From the graph, it can be observed that the mean position estimation error oscillated around zero. At some moment where the occlusion occurs, UDIMM estimator provides better estimation results with low covariance values which is a crucial factor to maintain tracking. The CV model exhibits significant position errors at certain time points, such as $t = 160$ second, indicating a consistent large total velocity error when utilizing the CV motion model. The estimation results produced by the UDIMM estimator do not exhibit

any unstable behaviors and are relatively small compared to other estimators. This is attributed to the UDIMM estimator's ability to improve prediction results when occlusion occurs. CV, CA and IMM filters perform unstable behaviours where the both maneuvering motion and occlusion occurs between time $t = 150 - 190$ seconds.

In Table 8.2, the performance of filters is presented in relation to the performance parameters discussed in Section 7.3. The evaluation of filter performance is based on the tracked tracklets, which are visualized (unmatched tracklets are not included) in the simulation environment. Table 8.2 offers a measure of the estimation quality within the tracklets; however, it does not reflect the accuracy of the entire tracking process. The CA model demonstrates excellent performance in estimating the positional states of tracklets themselves, particularly when the ID switch of an object is ignored. This results in lower RMSE value in terms of position compared to other models.

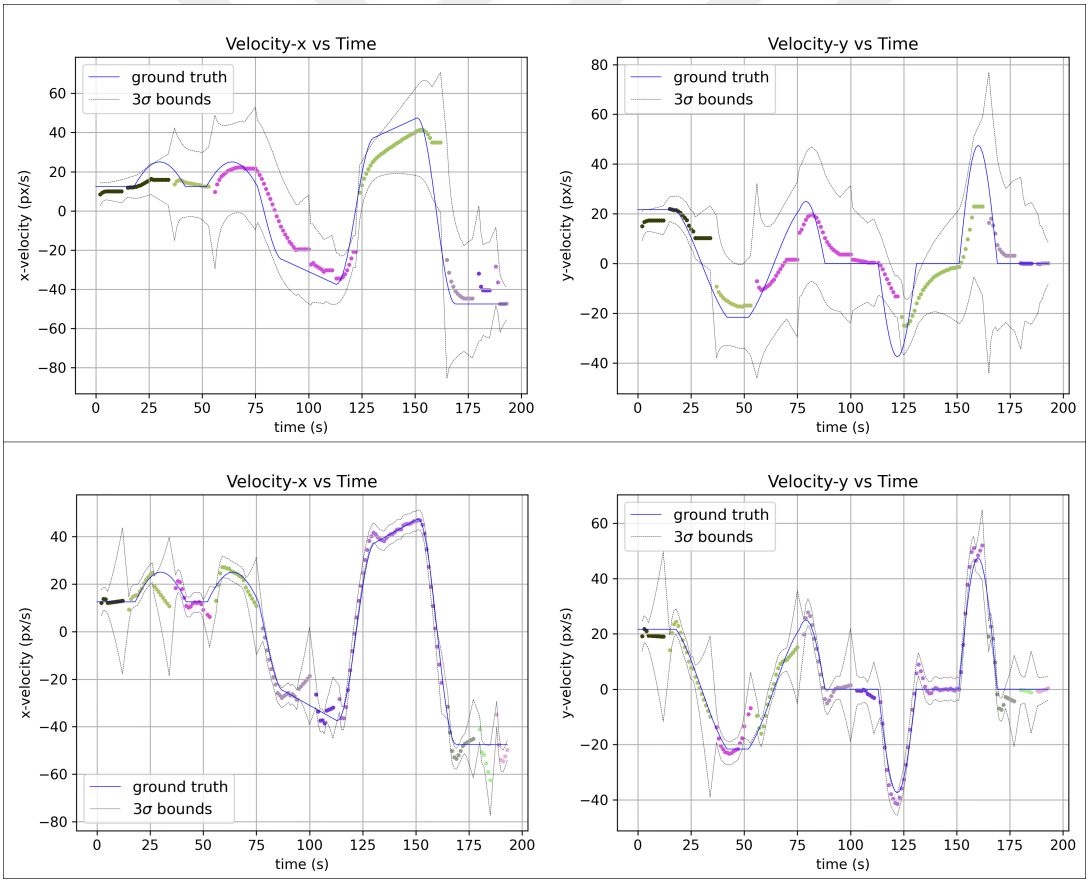


Figure 8.13: Velocity estimation results in x and y directions for CV model (first row), CA model (second row) in scenario 2.

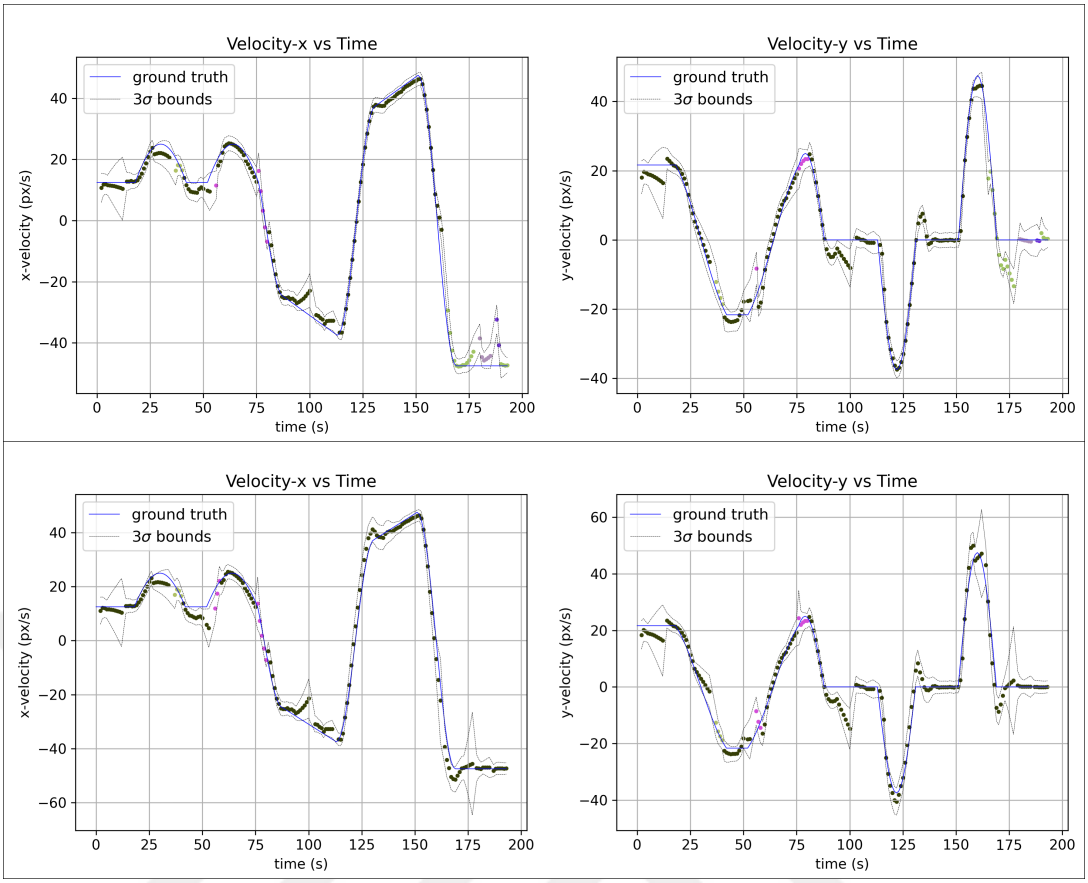


Figure 8.14: Velocity estimation results in x and y directions for IMM estimator (first row), UDIMM estimator (second row) in scenario 2.

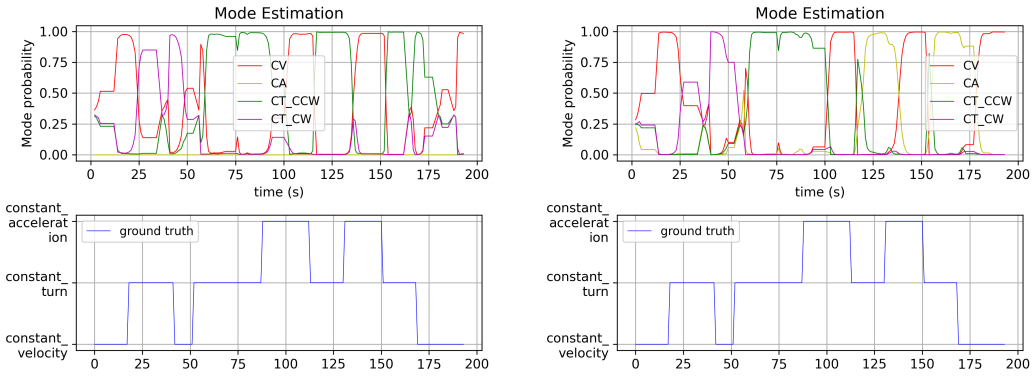


Figure 8.15: Mode probability results of IMM (left) and UDIMM (right) estimators for scenario 2.

However, in terms of velocity estimations, UDIMM has better accuracy within the 1σ bound. Furthermore, when considering the RMSE velocity result, it is observed that the UDIMM model exhibits better accuracy in estimating the total velocity. The key observation is that in the presence of occlusion, the CA model deviates and loses track of the object, initiating a new track based on the measurement. As a result, the position estimation performance appears higher in this table. However, UDIMM, by sustaining tracking and reducing covariance, achieves better performance compared to the CA model. It is evident that utilizing the multiple model filters increase the tracking duration of the objects. IMM and UDIMM estimator yields robust and reliable results with tracking percentage 78.42% and 92.21%, respectively. Overall, UDIMM estimator performs consistent and reliable results while dealing with mixing the multiple models and utilizing their benefits.

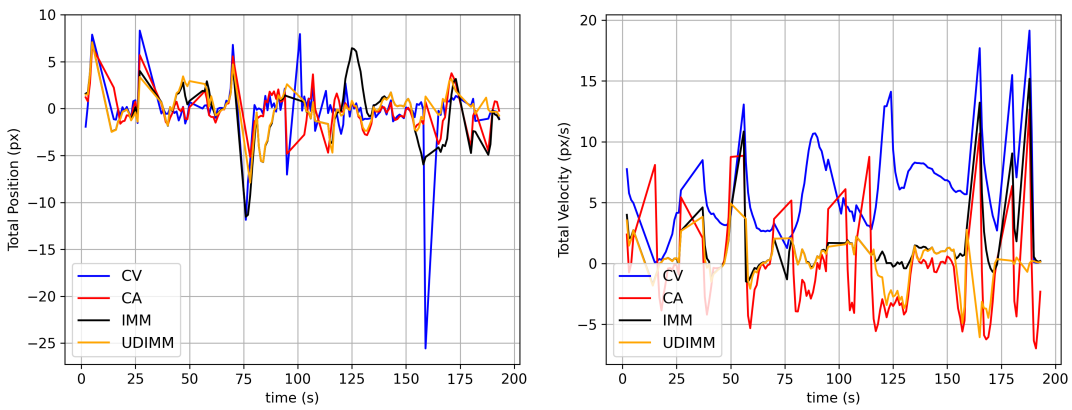


Figure 8.16: Mean Errors of Total Position (left) and Total Velocity (right) for scenario 2.

Multiple model estimators such as IMM and UDIMM, achieve the closer results to the desired NEES ($= 4$) compared to CV and CA models. Even though CA model provides lower position errors, difference between desired and the achieved NEES values are higher than the multiple model estimators due to estimations with high covariance values. ID switch counts and tracking results of filters with different motion models for a single run without including measurement uncertainty are obtained as,

- CV model: 8 ID switches, tracking is not sustained

- CA model: 10 ID switches, tracking is not sustained
- IMM filter: 10 ID switches, tracking is not sustained
- UDIMM filter: 6 ID switches, tracking is **sustained**

Table 8.2: Performance of Filters in Scenario 2

Filter	State	1σ | 2σ | 3σ (%) ↑	RMSE POS.(px) ↓	RMSE VEL.(px/s) ↓	NEES($=4$)	Track Perc.(%) ↑
CV	Est. Pos. (px)	12.74 | 21.67 | 25.32	64.23	48.89	315.58	21.55
	Est. Vel. (px/s)	68.75 | 79.89 | 86.75				
CA	Est. Pos. (px)	**82.78** | **91.78** | 94.24	**38.68**	35.66	2.87	5.27
	Est. Vel. (px/s)	83.65 | 91.43 | 95.46				
IMM	Est. Pos. (px)	76.35 | 85.76 | 91.52	47.51	30.62	**4.19**	78.42
	Est. Vel. (px/s)	82.14 | 86.43 | 94.68				
UDIMM	Est. Pos. (px)	81.22 | 90.34 | **95.66**	39.22	**18.79**	3.45	**92.21**
	Est. Vel. (px/s)	**85.41** | **92.76** | **96.69**				

8.3 Scenario 3: Multiple Object with Occlusion

In scenario 3, the focus is on monitoring two moving objects in case of occlusion. The new challenge is arise when two similar looking occludes each other at a complex scenario. The tracking performance is evaluated using a single CV motion model, a single CA motion model, IMM estimator and UDIMM estimator. Figure 7.1 provides visual representations of the illustration and ground truth modes and trajectory with occlusions for scenario 3. The blue and red scatter points depict the CAR 1's and the CAR 2's trajectories, respectively, while square boxes serve as occlusion objects or interruptions to the measurements obtained by the YOLOv5 object detector.

The image shown in Figure 8.17 displays the outcomes of position estimation for CAR 1 (left-column) and CAR 2 (right-column) using the CV model. The results indicate that CAR 1 experienced a higher number of ID switches compared to scenario 2, mainly because of increased occlusion caused by CAR 2. Besides, the position estimation accuracy is not satisfactory within the tracklets for both objects especially when the both maneuvering motion and occlusion exist.

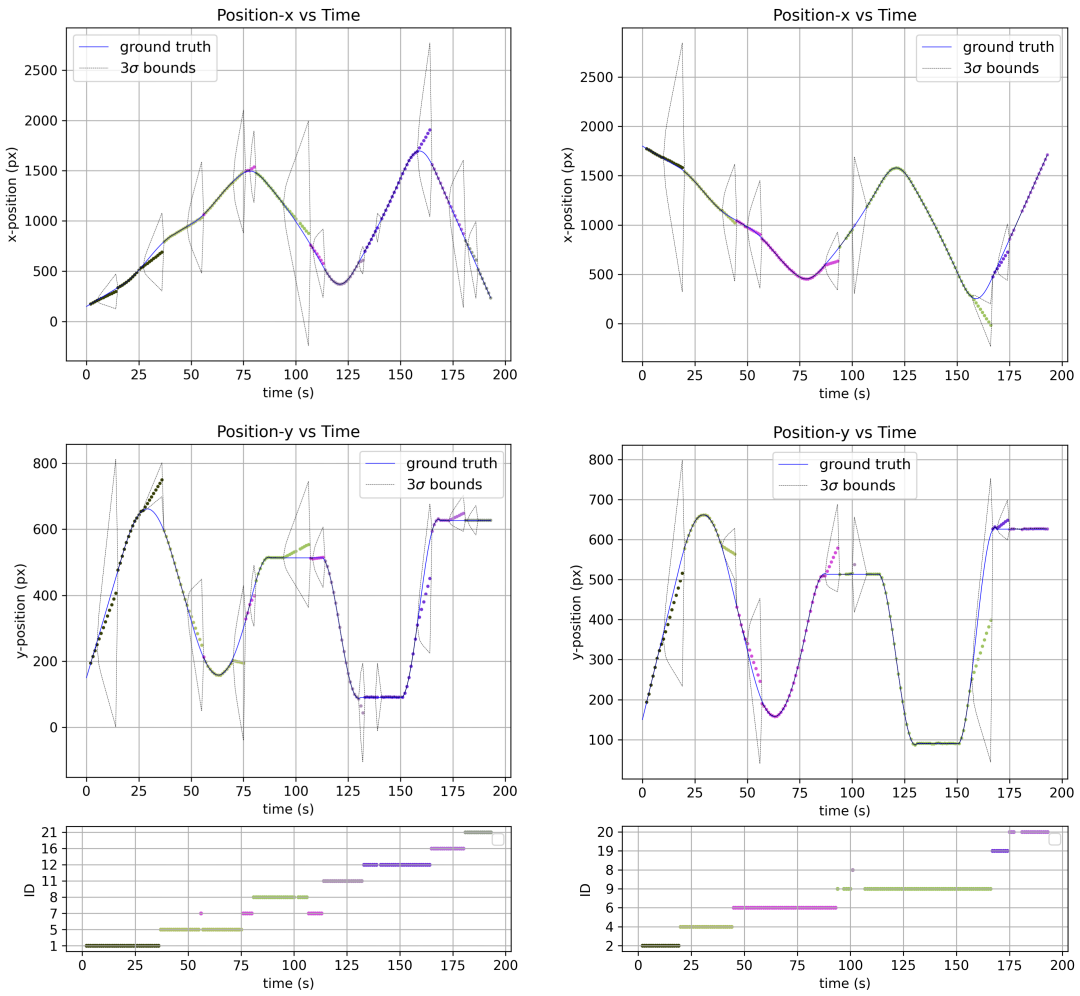


Figure 8.17: Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing CV model in scenario 3.

The trajectory of two objects is depicted in Figure 8.18. The figure reveals that there are significant data gaps, which can be attributed to the reasons explained in scenarios 1 and 2. These data gaps are more pronounced compared to the situation where only a single object is present in scenario 2.

The velocity estimation outcomes for CAR 1 and CAR 2 using the CV model are depicted in Figure 8.19. It is evident that the CV model does not accurately capture the true velocity, particularly during instances where there are changes in the direction of the velocity vector, such as at times $t = 25, 75, 115, 150$ seconds for CAR 1 and at times $t = 35, 65, 125, 165$ seconds for CAR 2. Moreover, the covariance bounds

exhibit significant magnitudes, indicating instability and unreliability in the tracking algorithm. Such large covariance bounds are not suitable for real-life applications.

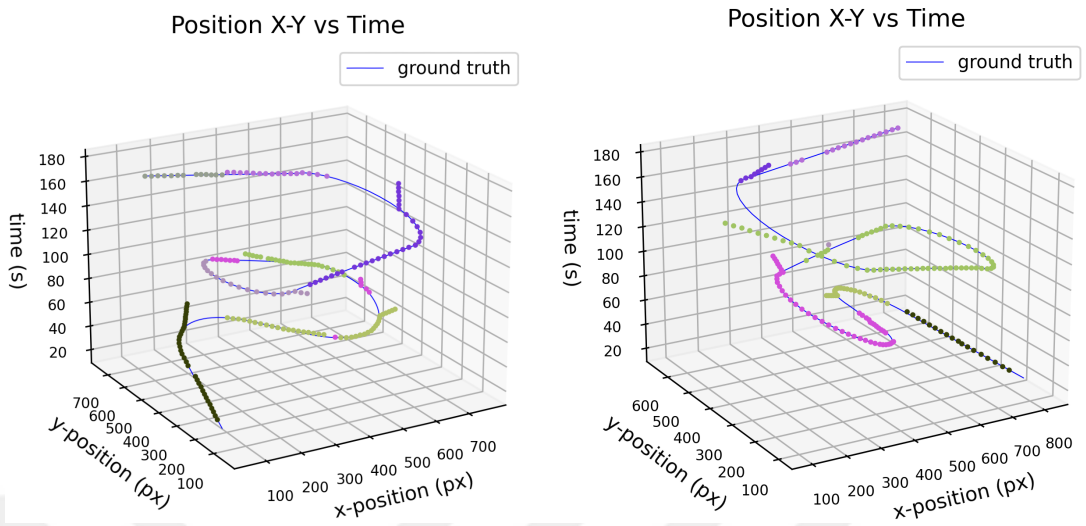


Figure 8.18: Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing CV model in scenario 3.

In contrast, employing the CA model improves the position estimation results with relatively smaller covariance bounds. However, despite having the same number of ID switches as in scenario 2, the total number of initiated IDs is twice as much for both CAR 1 (23 IDs) and CAR 2 (22 IDs), as shown in Figure 8.20. The initiated IDs can be viewed as candidate tracklets that are awaiting potential measurements. However, this introduces a challenge in terms of stability, as it increases the likelihood of erroneous data associations where a measurement intended for the true tracklet may be mistakenly assigned to one of the initiated IDs. And additional measurement coming from the second object also contribute to the complexity of data association which results in early estimation interruption as in time $t = 30$ seconds in Figure 8.20.

CA model very well captures the maneuvering motions and provides smooth and continuous estimation results within the tracklets. Therefore, incorporating the CA model in the UDIMM estimator enhances performance compared to the IMM estimator. However, it should be noted that the CA model introduces noise to the system, resulting in an increased probability of ID switches.

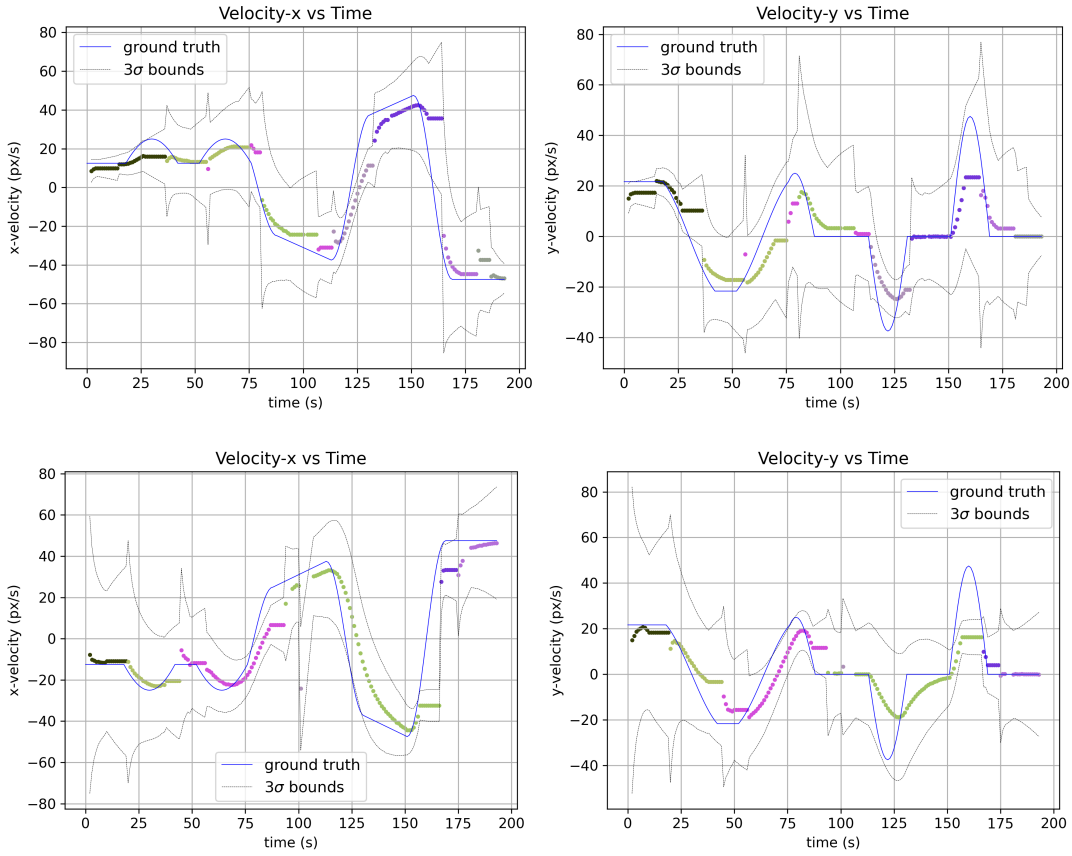


Figure 8.19: Velocity estimation results in x and y directions for CAR 1 (top-row), CAR 2 (bottom-row) utilizing CV model in scenario 3.

The occurrence of data gaps with the CA model is similar to that of the CV model, as illustrated in Figure 8.21. However, the number of missing estimations significantly increases in comparison to scenario 1 due to the presence of the second moving object, CAR 2, which introduces additional complexities to the tracking process. To consider the velocity estimation accuracy of CA model, the covariance bounds are relatively small compared to results implied with CV model. The true velocity remains within that the 3σ covariance bounds at high percentage as given in Figure 8.22. The estimations are not reliable and does not reflect the true velocity direction at some points such as at time $t = 27, 50, 110, 180, 190$ seconds for CAR 1 in the Figure 8.22 (first-row) and at time $t = 10, 15, 60, 90, 160, 175, 180$ seconds for CAR 2 in the Figure 8.22 (second-row).

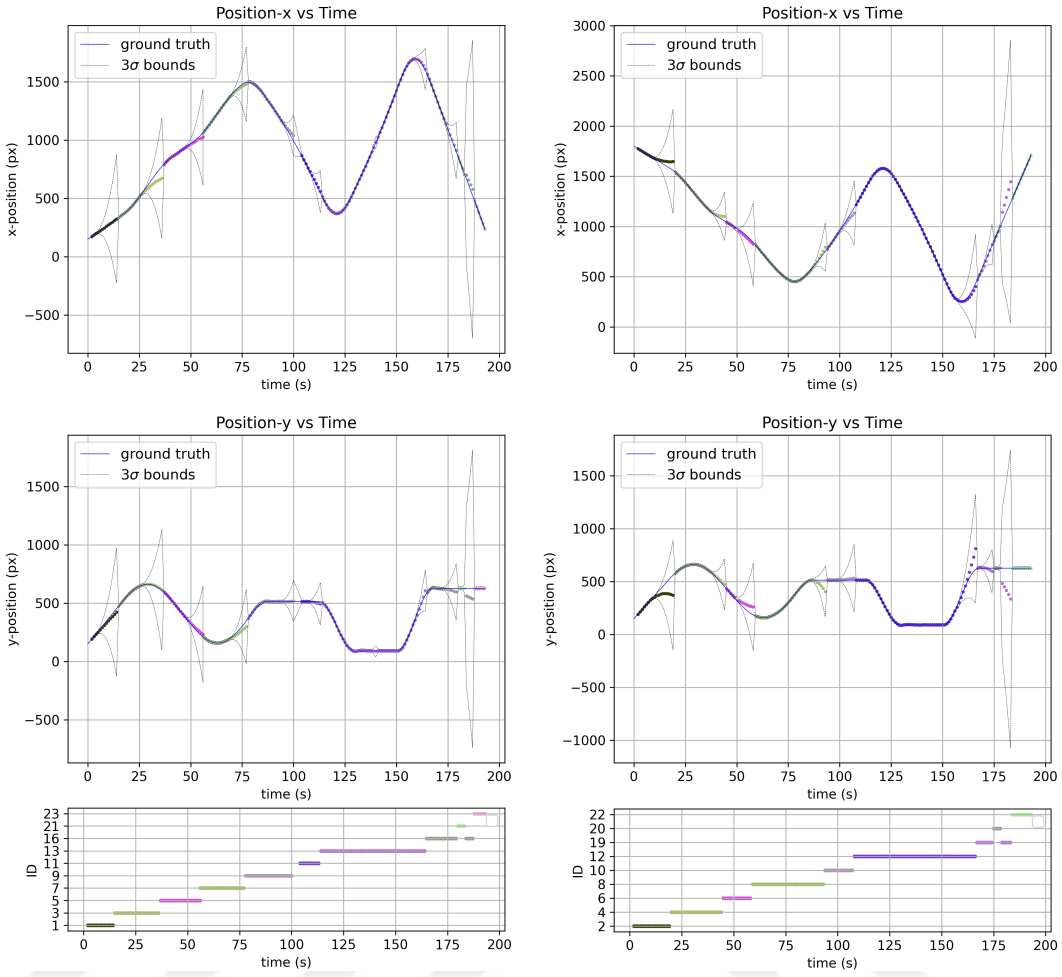


Figure 8.20: Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing CA model in scenario 3.

In scenario 3, when compared to scenario 2, the occurrence of high covariance values in the velocity estimates of CAR 1 disappears at certain times ($t = 15, 100, 185$ seconds) in the x-direction by employing CA model. This happens because the track is not considered as a match with the measurement shown. This could be a result of reducing the presence of unrelated tracks earlier by introducing a second object that affects the cost matrix.

By employing the IMM filter, the tracking duration of the initially detected track increases, similar to what occurs in scenario 2. However, despite having the same number of ID switches as scenario 2, the total count of initiated IDs almost doubles,

rising from 9 to 19, as shown in Figure 8.23. The positional estimation accuracy of the objects remain similar to scenario 2. However, in scenario 3, where a second object is introduced, data gaps begin to increase even in the case of using multiple model estimator which is more obvious in Figure 8.24.

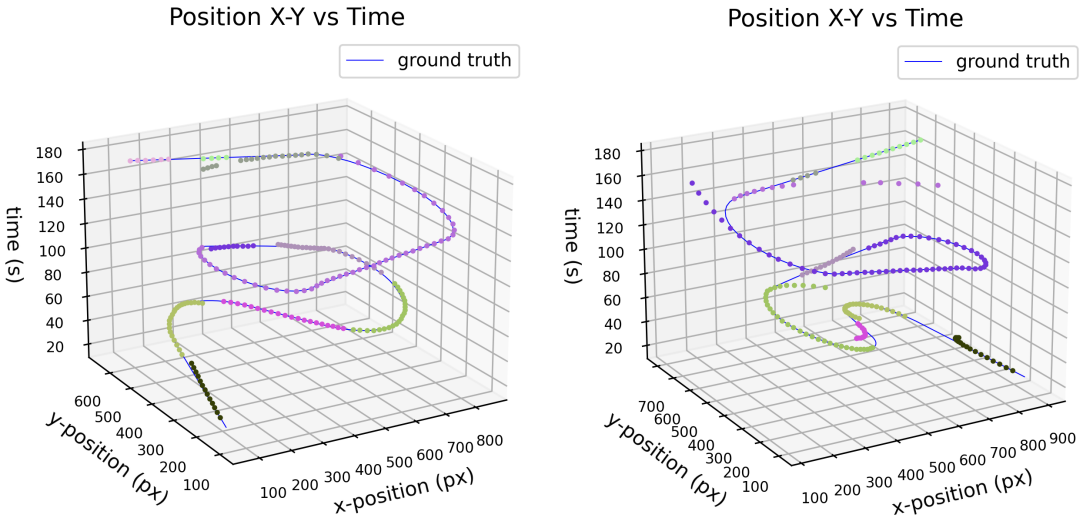


Figure 8.21: Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing CA model in scenario 3.

The experiments shows that as the number of objects increases, the available estimation information for each individual track decreases because of occlusions which reduces the reliability of the tracker in real time applications. Besides, IMM still provides highly accurate position estimations with small covariance and maintain the tracking for a longer period compared to CV and CA models.

Figure 8.25 illustrates the outcomes of velocity estimation using the IMM filter. The accuracy of velocity estimation for CAR 1 is significantly poorer compared to scenario 2. There are instances, such as at times $t = 50$ and $t = 180$, where the estimated velocities deviate significantly from the true values. These discrepancies arise due to a combination of occlusion by other moving objects and mode switching. The velocity estimates for CAR 1 are no longer trustworthy compared to scenario 2. Nonetheless, when it comes to tracking, the use of the IMM filter still enables better tracking performance compared to the CV and CA motion models.

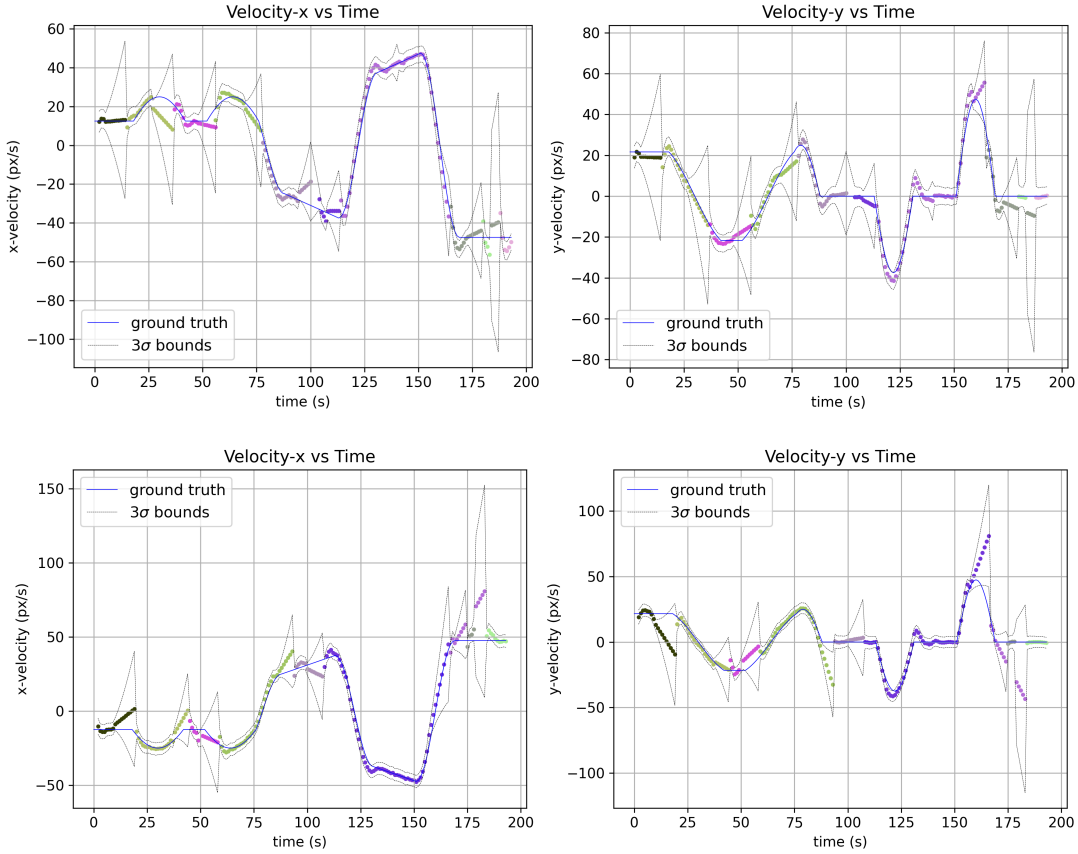


Figure 8.22: Velocity estimation results in x and y directions for CAR 1 (first-row), CAR 2 (second-row) utilizing CA model in scenario 3.

As for CAR 2, the estimation results are significantly superior to CAR 1, exhibiting high accuracy and low covariance values, with no divergence issues except for missing data between the time instances of $t = 160 - 170$ seconds in Figure 8.25 (second-row).

The objects overlap or occlude each other at four different times: at the beginning of the second CT motion mode ($t = 50sec$), during each of the two instances of CA motion mode ($t = 100, 135\ secs$) and during the final CV motion mode ($t = 180sec$).

These specific moments represent the highest likelihood of identity switching or losing track, as depicted in Figure 8.26. During these instances, the object detector can be easily confused by the similarity between the two objects, leading to erro-

neous measurements. Similarly, during the data association process, the Hungarian algorithm may produce incorrect results due to the closely spaced measurements or predicted states derived from the Kalman Filter's motion model leading similar cost values. This can result in the wrong assignment of measurements to the other object and a potential switch in identification. Consequently, tracking algorithms face the greatest challenges when the objects are in close positions or occlude each other. The IMM filter mostly capture the motion mode of the object, but approximating the CV model when the true mode is turned to CA model, as the CA model is not available as it is depicted in Figure 8.26 for both objects.

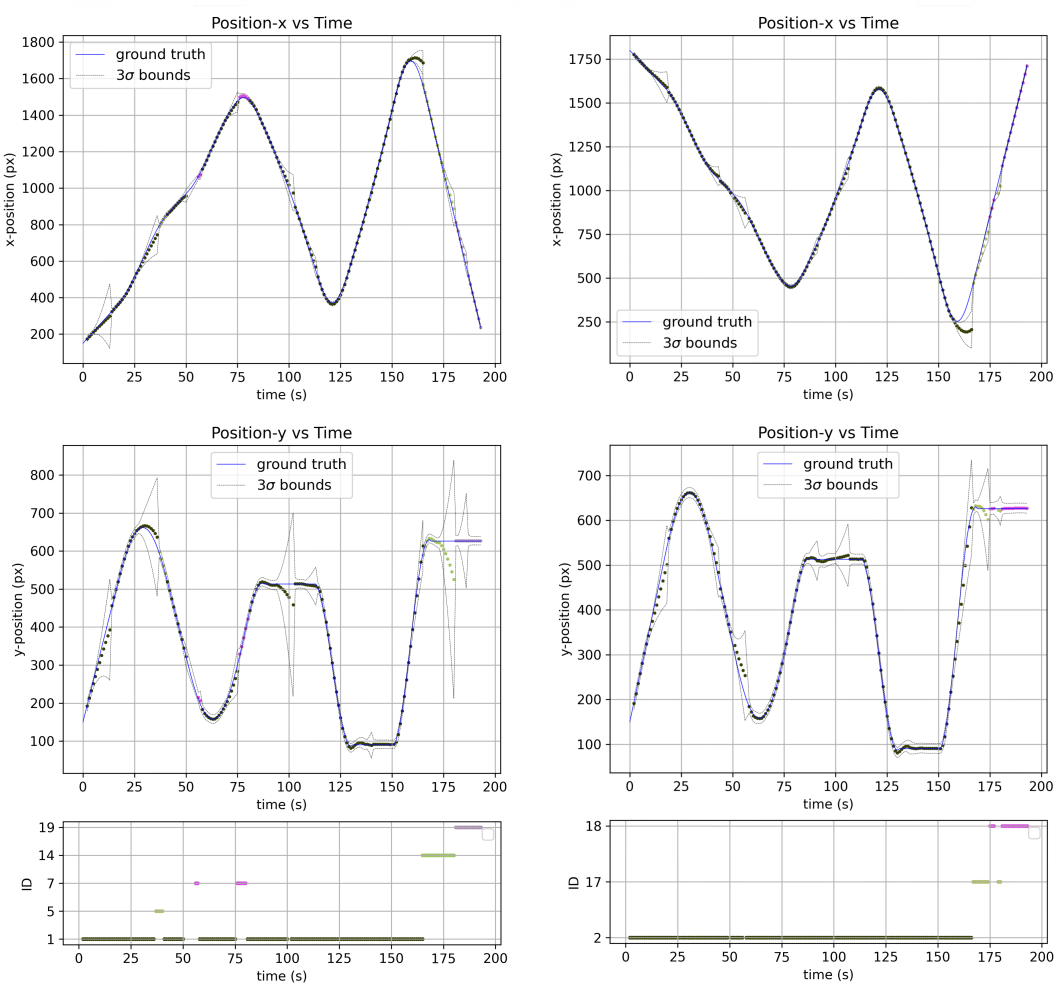


Figure 8.23: Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing IMM estimator in scenario 3.

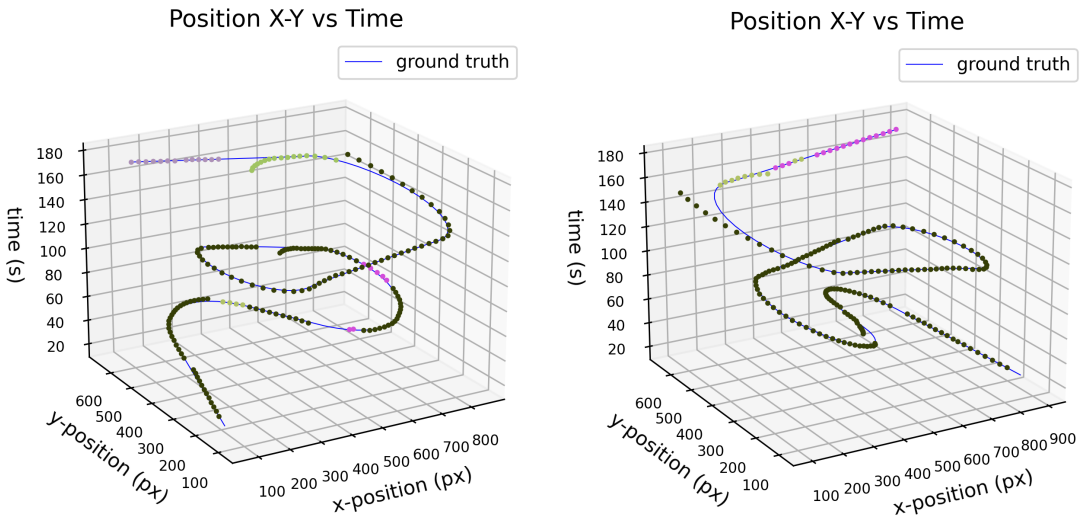


Figure 8.24: Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing IMM estimator in scenario 3.

Through the utilization of UDIMM model, consistent tracking is maintained during the simulation, as depicted in Figure 8.27. Comparatively, in scenario 3, the total count of initiated IDs is higher than in scenario 2, as anticipated. Nonetheless, for CAR 1 in scenario 3, the employment of UDIMM results in a significantly lower number of initiated IDs (12 IDs) compared to when the IMM filter is utilized (19 IDs). The positional estimation accuracy of the objects remain similar to scenario 2. However, in scenario 3, where a second object is introduced, data gaps begin to increase even in the case of using multiple model estimator which is more obvious in Figure 8.28. he experiments shows that as the number of objects increases, the available estimation information for each individual track decreases because of occlusions which reduces the reliability of the tracker in real time applications. Besides, IMM still provides highly accurate position estimations with small covariance and maintain the tracking for a longer period compared to CV and CA models.

In scenario 3, the accuracy of positional estimation for the objects remains comparable to scenario 2 when employing UDIMM filter as shown in Figure 8.27. However, in scenario 3, where a second object is introduced, the occurrence of data gaps starts to rise (similar to the other filter results) even when employing a UDIMM estimator, as evident in Figure 8.28.

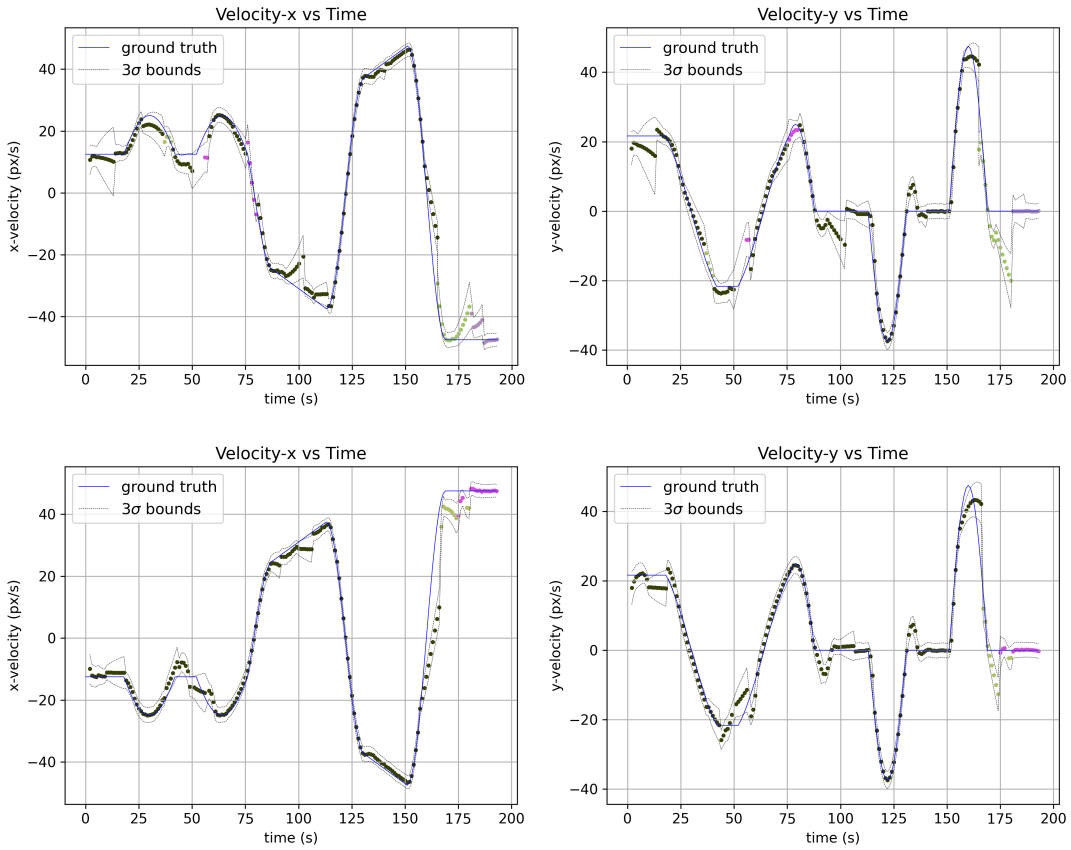


Figure 8.25: Velocity estimation results in x and y directions for CAR 1 (first-row), CAR 2 (second-row) utilizing IMM estimator in scenario 3.

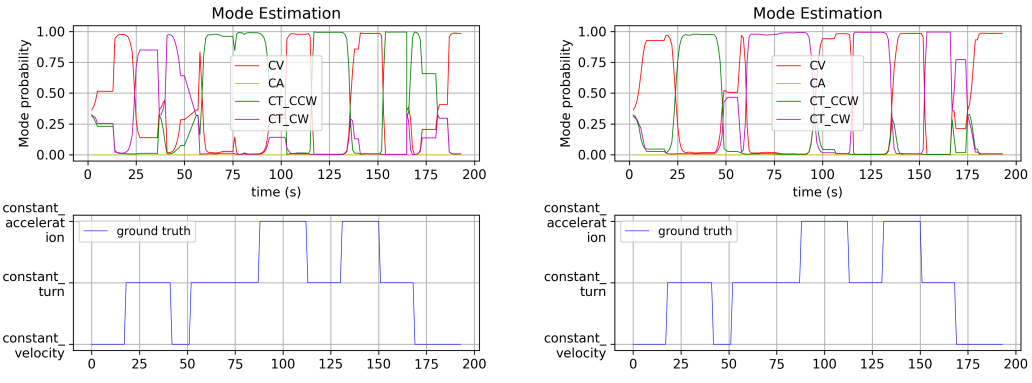


Figure 8.26: Mode probabilities changes of IMM filter for CAR 1 (left), CAR 2 (right) in scenario 3.

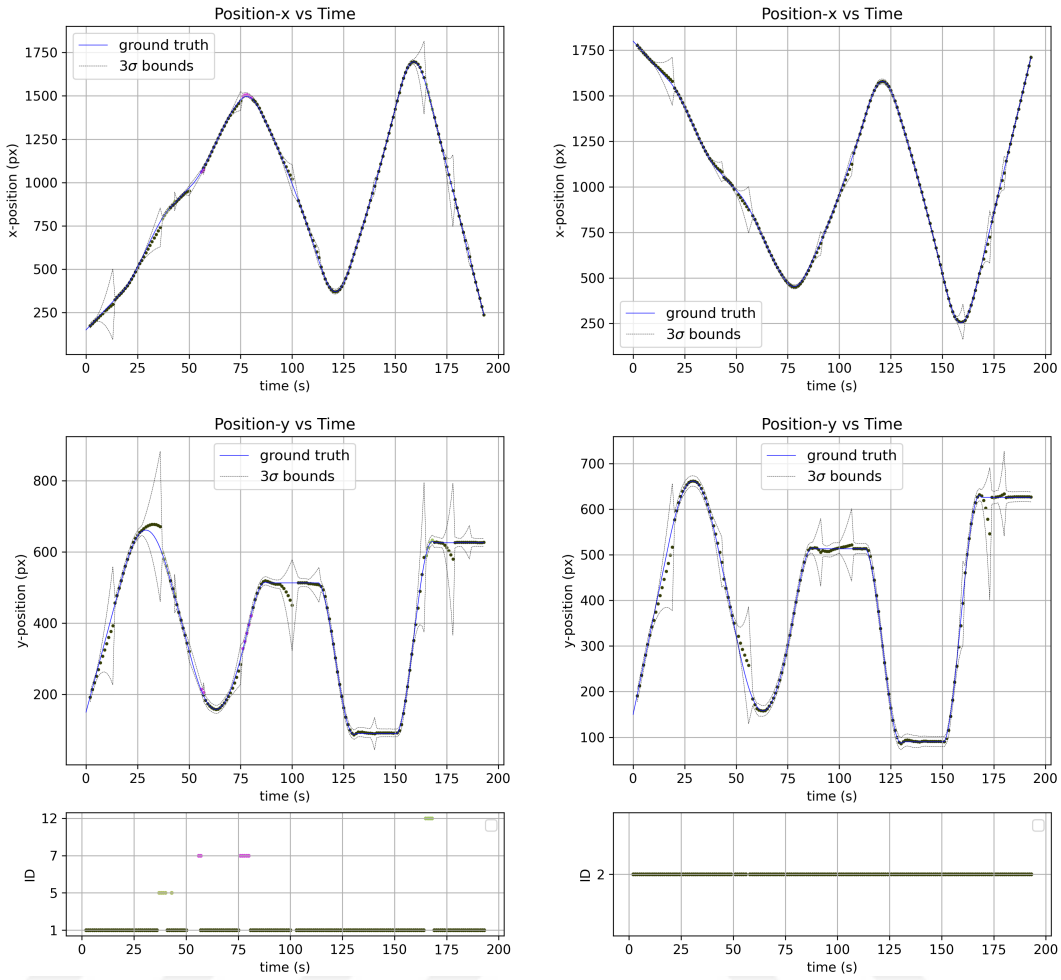


Figure 8.27: Position estimation results in x and y directions for CAR 1 (left-column), CAR 2 (right-column) utilizing UDIMM estimator in scenario 3.

As previously stated, in these circumstances, the reliability of the tracker in real-time applications is reduced. Nonetheless, the UDIMM method consistently delivers precise position estimations characterized by a small covariance, and it effectively maintains tracking throughout the entire simulation. Remarkably, it is the only estimator maintain the tracking in demonstrated simulation without any issues. The reliability and resilience of the UDIMM estimator can be verified by examining the position estimation and frequency of ID switches for CAR 2, as illustrated in Figure 8.27. It is evident that CAR 2 successfully completes scenario 3 without experiencing any ID switches or estimation deviations that would result in spikes in covariance or unstable outcomes. This reinforces the robustness and reliability of the UDIMM estimator.

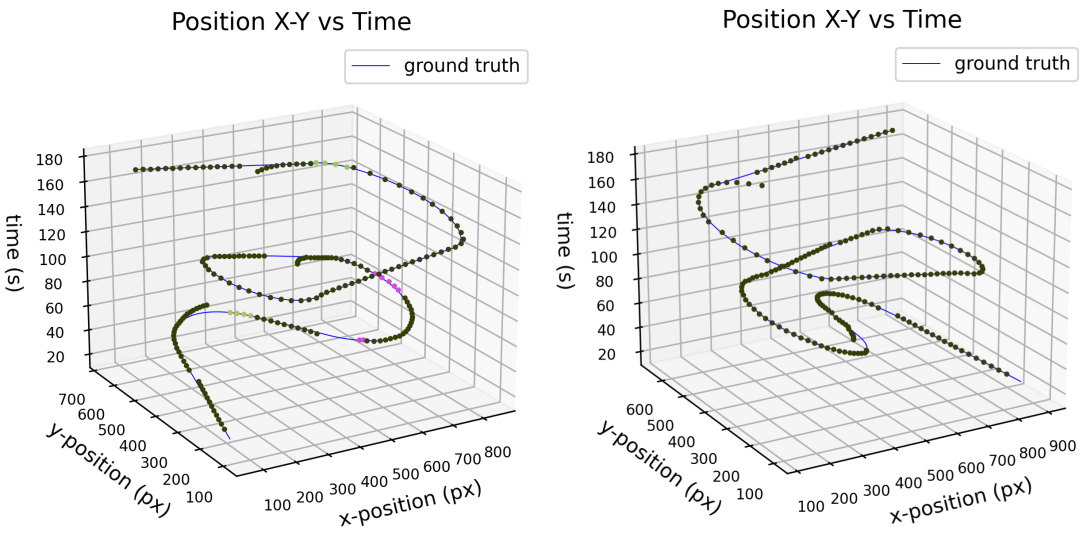


Figure 8.28: Trajectory of CAR 1 (left) and CAR 2 (right) in x and y directions w.r.t time utilizing UDIMM estimator in scenario 3.

To examine the outcomes of velocity estimation achieved by the UDIMM estimator, a comprehensive depiction of the results is presented in Figure 8.29. This figure provides a detailed analysis of the velocity estimation results for further evaluation. At certain time points, specifically at t = 50 and t = 170, the estimated velocities from the UDIMM estimator exhibit notable deviations from the true values, similar to what is observed with the IMM estimator in scenario 3. Given that the velocity states are not factored in when computing the Mahalanobis distance for data association, these inconsistent estimations do not immediately lead to ID switches unless they are rectified by the measurements. This is because velocity estimations have an impact on subsequent position estimations, which subsequently affects the calculations for data association. Nevertheless, these types of inconsistent estimation outcomes are deemed unsuitable for utilization in motion planning or guidance algorithms. Furthermore, the velocity estimation results for CAR 2 exhibit superiority, characterized by tight covariance boundaries, even in this demanding scenario. It is depicted that by preserving ID switches, the estimation data becomes more consistent and uninterrupted, devoid of any abrupt peaks or missing data/states.

As stated previously, in scenario 3, the objects overlap or occlude each other at four

different times ($t = 50, 100, 135, 180$ secs). In these challenging moments, UDIMM achieves better tracking results by weighting the models to keep the estimations on track for both objects which can be seen in Figure 8.30. Following the occlusion of two objects and with the other boxes, instead of relying on a single model with a high probability, the final estimation is combined with various model weights such at times $t = 40, 60, 115, 160$ secs for the object, CAR 1.

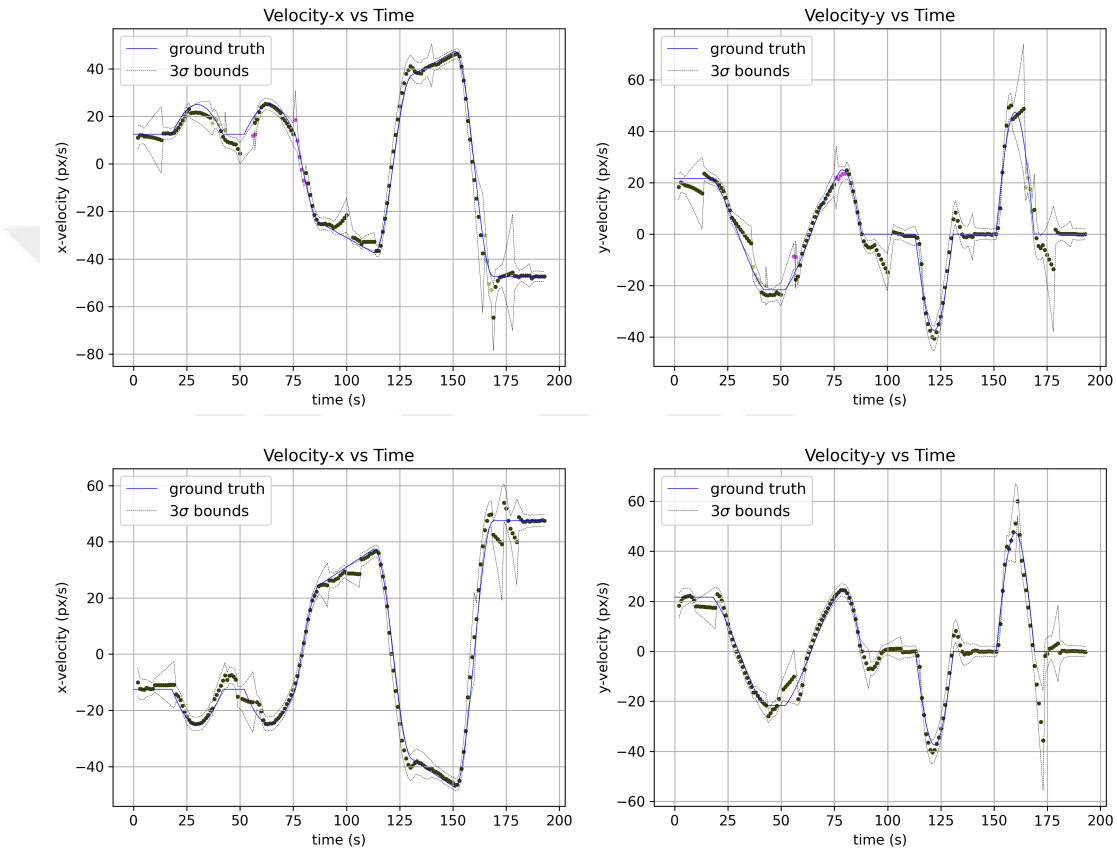


Figure 8.29: Velocity estimation results in x and y directions for CAR 1 (first-row), CAR 2 (second-row) utilizing UDIMM estimator in scenario 3.

On the other hand, mode probabilities for CAR 2 is more stable and accurate with a higher weight for a particular model. Therefore, the tracking results are finalized without any ID switch for the CAR 2 object. Again in the scenario 3, true CA modes at $t = 100$ sec and $t = 135$ sec are estimated with the CV model, and true CT modes at $t = 120$ sec and $t = 160$ sec are estimated with the CA model based on the

explanations provided in scenario 1 and scenario 2.

Figure 8.31 illustrates the mean error values resulting from Monte Carlo simulations, specifically regarding total position and total velocity for CAR 1 (first-row) and CAR 2 (second-row). The figure reveals that the mean position estimation error of the UDIMM estimator fluctuated near zero, with only two instances of unstable behavior for CAR 2 at time $t = 100, 175$ seconds. Figure 8.31 provides a clear demonstration of how the UDIMM estimator successfully maintains tracking while other estimators fail. Notably, the velocity errors are significantly elevated and exhibit high-frequency oscillations when using linear filters like CV and CA models. The CV model exhibits significant position errors at certain time points for CAR 1, such as $t = 100, 115, 130, 160, 180$ seconds. The estimation errors produced by the multiple model estimators do not exhibit any unstable behaviors and are relatively small compared to other estimators. The only moment where the IMM estimator produce large position estimation error which cause loss of tracking is at the time $t = 160, 180$ seconds for both objects. Moreover, the CA model exhibits remarkably high velocity errors that fail to accurately represent the actual velocities of CAR 1 and CAR 2, particularly during instances of frequent occlusions at between times $t = 160 - 180$ seconds.

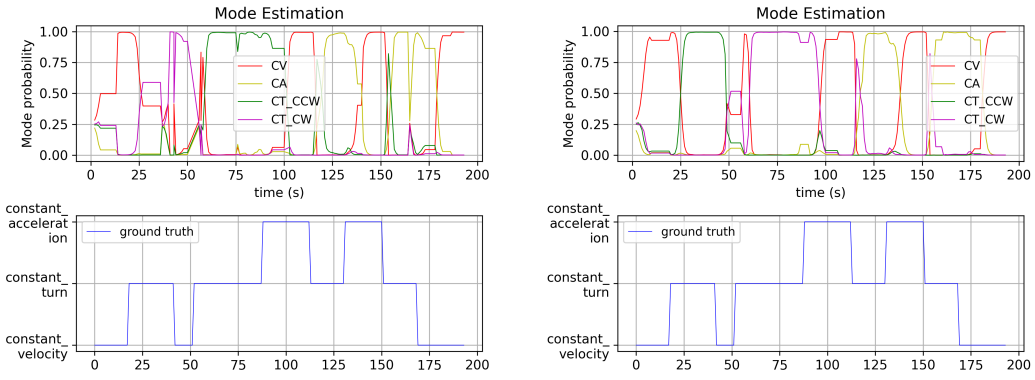


Figure 8.30: Mode probabilities changes of UDIMM filter for CAR 1 (left), CAR 2 (right) in scenario 3.

In Table 8.3, the performance of filters is presented in relation to the performance parameters discussed in Section 7.3. The assessment of filter performance relies on

the examination of matched tracklets within the simulation environment, excluding tracklets that are not matched but are propagated with motion models. In the presence of occlusion, the prediction outcomes of the recent matched tracks are incorporated into the performance table results. Additionally, different than the other scenarios, the Table 8.3 presents the combined performance results of the two object.

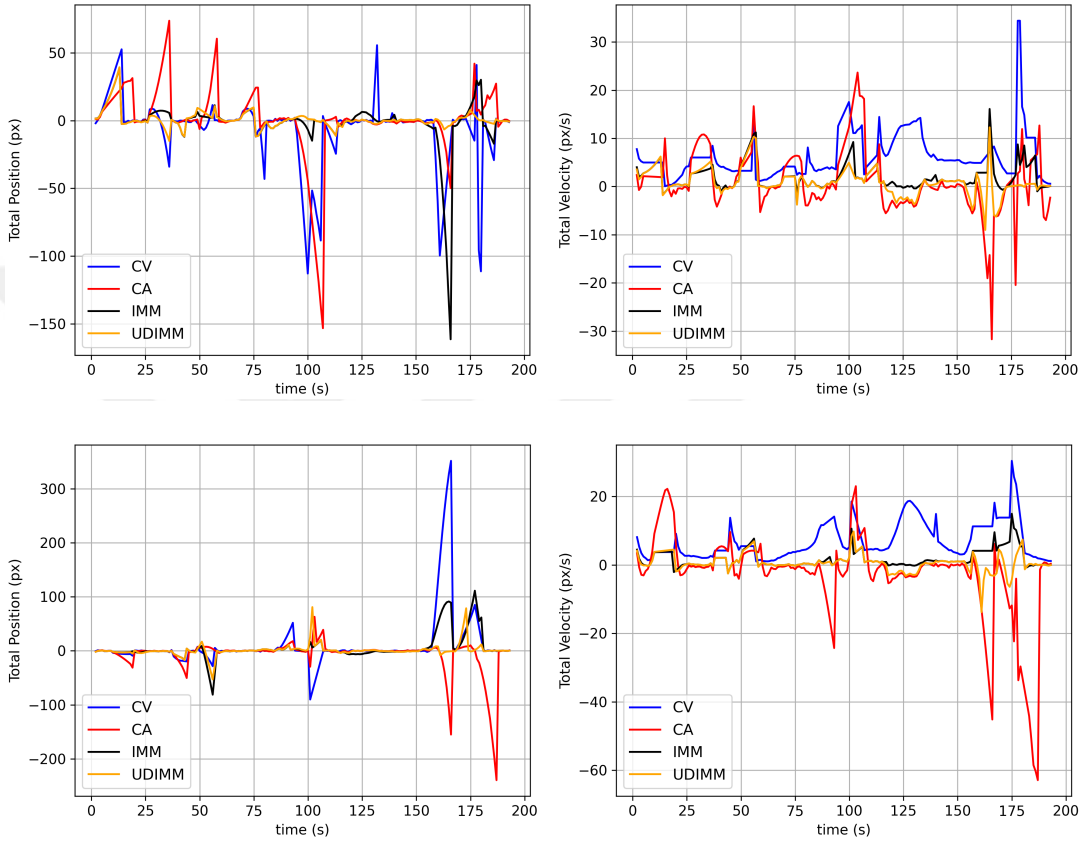


Figure 8.31: Mean Errors of Total Position and Total Velocity for CAR 1 (first-row) and CAR 2 (second-row) in scenario 3.

The estimation results obtained from the CV model, as indicated in Table 8.3, demonstrate a high degree of unreliability and produce the poorest outcomes, leading to an ID switch at the first occlusion. As in the other scenarios, CA model provides quite good estimation results. Nevertheless, because of the accumulation of high covariance values during occlusions, tracking is not maintained when measurements arrive. This is because the tracklets are eliminated during the process of applying a gating

threshold while calculating the Mahalanobis distance. Therefore, even CA model provides satisfactory mean estimation results, they are not reliable and not acceptable with the high covariance values in multiple object tracking algorithms. MOT methods primarily concentrate on resolving the data association problem within a dense environment. However, if a tracklet possesses large covariance boundaries that conflict with other tracklets, it can lead to unstable and unreliable tracking results. The primary factor contributing to the CA model's better NEES value of 2.55 compared to IMM filter is the high covariance associated with the estimated states in relation to the estimation errors. However, it is important to note that the NEES value alone does not sufficiently represent the performance parameter for maintaining the tracking.

Furthermore, multiple model estimators offer estimations with small covariance values, which effectively extend the tracking duration even in challenging scenarios. While it is still possible to encounter an ID switch when employing multiple model estimators, the presence of relatively smaller covariances makes it easier to revert back to the original ID. This is because the convergence time of the covariances to a small value requires only a few time steps. In terms of the state estimation accuracy such as estimated position and estimated velocity and both the CA and UDIMM estimators achieve excellent results with an estimation accuracy exceeding 80% for the 1σ bound. However, the only filter that maintains the tracking throughout the simulation period is the UDIMM estimator. This is attributed to its ability to deliver over 80% accuracy for the 1σ bound and over 95% accuracy for the 3σ bound, while maintaining low covariance values. This assertion is further supported by the NEES value of 4.98 which is very close to the desired NEES ($= 4$) value. The NEES test reveals that the IMM filter performs worse results compared to scenario 2 when the second object is introduced in scenario 3. UDIMM demonstrates superior performance in terms of the root mean square error of position and velocity compared to other models. Furthermore, it is clear that the UDIMM estimator's ability to produce a low estimated velocity error plays a significant role in achieving a low Mahalanobis distance. Consequently, UDIMM demonstrates superior tracking accuracy compared to other filters. As a result, UDIMM maintains a tracking percentage above 90% even when a second object is introduced.

The UDIMM estimator shows a minimal deviation from the desired NEES ($= 4$)

value. In contrast, the other estimators exhibit larger deviations from the desired NEES value. A significantly smaller NEES value indicates that the covariance values are relatively large compared to the estimation errors. Conversely, a considerably higher NEES value suggests that the estimation errors are relatively large in relation to the covariance values. ID switch counts and tracking results of filters with different motion models for a single run without including measurement uncertainty are obtained as a summation of both CAR 1 and CAR 2,

- CV model: 18 ID switches, tracking is not sustained
- CA model: 19 ID switches, tracking is not sustained
- IMM filter: 10 ID switches, tracking is not sustained
- UDIMM filter: 8 ID switches, tracking is **sustained**

Table 8.3: Performance of Filters in Scenario 3

Filter	State	1σ | 2σ | 3σ (%) ↑	RMSE POS.(px) ↓	RMSE VEL.(px/s) ↓	NEES($=4$)	Track Perc.(%) ↑
CV	Est. Pos. (px)	10.34 | 19.77 | 24.44	73.78	57.44	351.65	12.64
	Est. Vel. (px/s)	66.78 | 77.17 | 82.45				
CA	Est. Pos. (px)	**85.16** | 90.22 | 93.33	40.89	37.11	2.55	5.15
	Est. Vel. (px/s)	82.01 | 90.34 | 93.11				
IMM	Est. Pos. (px)	73.31 | 81.23 | 89.21	49.11	33.55	6.17	76.57
	Est. Vel. (px/s)	81.11 | **91.98** | 93.88				
UDIMM	Est. Pos. (px)	83.39 | **90.52** | **95.11**	**40.56**	**22.56**	**4.98**	**90.31**
	Est. Vel. (px/s)	**84.89** | 91.81 | **95.22**				

8.4 Test on MOT17 Validation Dataset

In this section, the HOTA, MOTA, and IDF1 metric results of DeepSORT, StrongSORT, and StrongSORT on the MOT 17 dataset are presented when employing the UDIMM filter. StongSORT algorithm depart from the DeepSORT algorithm with the additional modifications such as BoT (modified feature extractor), ECC (enhanced correlation coefficient maximization) model for the camera motion compensation, NSA (noise scale adaptive) measurement covariance, EMA (exponential moving average) for the feature updating strategy, MC (matching with motion cost) and (woC)

abandoning matching cascade. StrongSORT++ algorithm includes AFlink and GSI modules. The algorithms are explained in detail in Sections 3.3 and 3.4.

Table 8.4: Results of applying UDIMM estimator with DeepSORT and StronSORT algorithms ON MOT17 validation dataset

Method	**HOTA**	**MOTA**	**IDF1**
DeepSORT	66.3	76.7	77.3
DeepSORT with UDIMM	66.7	75.9	77.2
StrongSORT	69.6	77.1	82.3
StrongSORT with UDIMM	68.7	76.25	81.4
StrongSORT++	70.8	78.7	83.3
StrongSORT++ with UDIMM	69.97	77.5	82.5

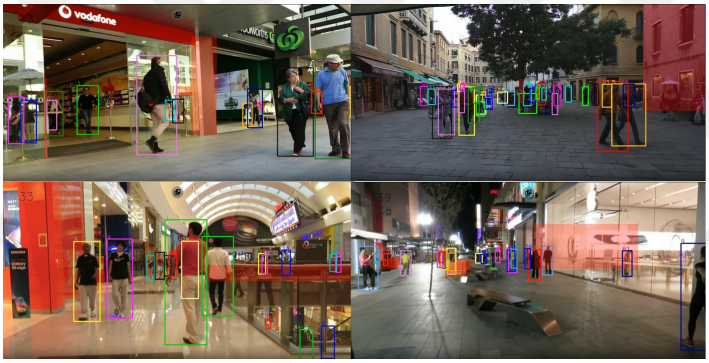


Figure 8.32: Sample MOT 17 footages with annotated ground truths

Upon analyzing the results presented in Table 8.4, it is evident that the only metric in which the application of the UDIMM estimator yields improved results is the HOTA metric, specifically when used with the DeepSORT. The primary explanation for this is that the MOT17 dataset comprises images captured from a human eye-level perspective, where the objects being tracked are pedestrians. These pedestrians exhibit motion patterns that differ from the motion models utilized in the UDIMM estimator. The Figure 8.32 represents the sample footages from MOT17 validation dataset. It can be inferred that even when considering the human eye perspective, the UDIMM estimator performs sufficiently well and does not exhibit any unstable behavior.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

This thesis aims to enhance the accuracy of motion estimation for maneuvering objects in vision-based tracking algorithms, especially for the situations where there is occlusion. In the existing MOT algorithms, such as StrongSORT, the motion prediction modules are based on a CV motion model. However, in this thesis, the UDIMM filter is utilized to modify the motion model of the MOT algorithm to improve its performance. The effectiveness of the UDIMM method was tested in three distinct simulated scenarios and compared with single model filters and the traditional IMM method. The evaluation results indicated that the UDIMM filter yields more robust and accurate estimation outcomes when the object is executing maneuvers and is occluded by either static or similar-looking dynamic objects. In both of these scenarios, the UDIMM method was able to track the object with a high level of performance even in situations where other filters failed to do so or lose track of the object, particularly when the object is performing complex maneuvers and there is dense occlusion.

The results indicate that multiple model estimators provide estimations with small covariance values, which effectively prolong the tracking duration even in challenging scenarios. On the other hand, the CV model demonstrates a high level of unreliability and generates the poorest outcomes, resulting in an ID switches when occlusion occurs. In all scenarios, the CA model yields favorable estimation results. However, due to the accumulation of high covariance values during occlusions, tracking is not sustained upon the arrival of measurements. This is a result of tracklets being eliminated during the application of a gating threshold in the result of calculating the

Mahalanobis distance. Therefore, even CA model provides satisfactory mean estimation results, they are not reliable and not acceptable with the high covariance values in multiple object tracking algorithms. On the other hand, while it is still possible to encounter an ID switch when employing multiple model estimators, the presence of relatively smaller covariances makes it easier to revert back to the original ID. This is because the covariance values quickly converge to small values within a few time steps.

As illustrated in the model framework, appearance models are used in conjunction with kinematic models to reduce ID switch. In fact, these two models are used to improve the performance of tracking algorithms. While appearance models try to capture the similarity between pixels in two frames, kinematic models aim to capture the similarity of trajectories between frames. In this thesis, existing deep learning models from literature are adopted without modification, while motion models are improved using unequal dimension multiple model estimators.

It is evident that in the simulation, objects are passing over each other, which would not typically occur for the vehicles in aerial images. However, in our simulations, this aspect has been disregarded, deliberately introducing overlapping situations and augmenting complexity. Consequently, under normal circumstances, the frequency of mode switches would be significantly lower, leading to improved estimation outcomes such that less number of ID switches are observed.

The another point to emphasize is that in the constant turn model, a constant angular velocity is employed. While this assumption is crucial, simulations were conducted with different ground truth angular velocity values in various constant turn modes to mitigate assumptions impact. However, several methods exist for more realistic estimation predictions. For instance, the model set can be expanded by adding multiple constant turn models with different constant angular velocity values. Another solution is to incorporate nonlinear estimators like Extended Kalman Filters (EKF) into multiple models to estimate the angular velocity as well. However, when this method was attempted, unstable estimation results were encountered during occlusion, and with large covariance values, tracking was completely lost. Another potential solution is to measure the object's heading angle, which can be achieved using Oriented

Bounding Boxes (OBBs), as mentioned in the future work section. With the angle measurement, the angular velocity can be predicted with high accuracy since they are highly correlated states.

In all three scenarios, the UDIMM estimator is the only filter that consistently maintains tracking throughout the simulation period. Moreover, when considering the human eye perspective, the UDIMM estimator provides satisfactory estimation results and remains stable, as confirmed by the test results on the MOT 17 validation dataset.

9.2 Future Work

In the following studies, there are four main attributes that will be considered.

- The existing detection frameworks currently employ horizontal bounding boxes that do not take into account the orientation of objects. This poses a particular challenge in aerial images, where object orientation plays a crucial role in achieving accurate tracking. As a result, additional information regarding the orientation becomes necessary to enhance estimation results. To address this, oriented bounding boxes (OBBs) will be integrated into YOLO frameworks, enabling the inclusion of the heading angle, θ, as an input to the system. This integration aims to improve velocity and acceleration estimations.

- Since the substructure of the StrongSORT algorithm is not suitable for the benchmark evaluation on aerial images datasets e.g. DOTA, Visdrone, UAVDT and DroneVehicle. Further efforts are needed to extract features that are compatible with the specific data type and maintain consistency in the input/output relationship. This involves training a new version of the YOLO framework using an aerial image dataset to adapt it to the specific requirements of aerial imagery.

- Creating a cuboid shaped box around the detected object helps to estimate the dimensions of the object in non-dimensional unit. If estimation of a distance can be added to the calculation, the object dimensions including depth even from different perspectives can be determined. This contribution would be very

useful in the future of MOT by providing three dimensional estimation results for the moving objects.

- Additional constraints will be established, falling into two categories: 1) Dynamical constraints, which involve behaviors that objects are unable to perform, and 2) Environmental constraints, which indicate areas that objects cannot pass through. These constraints can be incorporated into the prediction models based on the class types of the objects determined by the object detector. This implies that class types can be utilized as supplementary measurements for the filter models.

REFERENCES

[1] B. Yang and R. Nevatia, "Online learned discriminative part-based appearance models for multi-human tracking," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 484–498, Springer Berlin Heidelberg, 2012.

[2] A. Bouguettaya, A. Kechıda, and A. M. Taberkıt, "A survey on lightweight cnn-based object detection algorithms for platforms with limited computational resources," *International Journal of Informatics and Applied Mathematics*, vol. 2, no. 2, pp. 28 – 44, 2019.

[3] "A classical CNN architecture." `https://www.run.ai`. Accessed: 27/05/2023.

[4] Øyvind Kolås, "Image." `https://pippin.gimp.org`. Accessed: 27/05/2023.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020.

[7] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, no. 2, 2021.

[8] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, "SORT and Deep-SORT based multi-object tracking for mobile robotics: Evaluation with new data association metrics," *Applied Sciences*, vol. 12, no. 3, 2022.

[9] Y. Du, J. Wan, Y. Zhao, B. Zhang, Z. Tong, and J. Dong, "Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 2809–2819, October 2021.

[10] J. Luiten, A. Os, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," *International Journal of Computer Vision*, vol. 129, pp. 548–571, 2021.

[11] Ultralytics, "YOLOv5." `https://github.com/ultralytics/yolov5`. Accessed: 08/05/2023.

[12] J. Zubača, M. Stolz, R. Seeber, M. Schratter, and D. Watzenig, "Innovative interaction approach in IMM filtering for vehicle motion models with unequal states dimension," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3579–3594, 2022.

[13] G. Ciaparrone, S. T. Francisco Luque Sánchez, R. T. Luigi Troiano, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[14] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Y. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Object detection in aerial images: A large-scale benchmark and challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7778–7796, 2022.

[15] R. E. Kalman, "Contributions to the theory of optimal control," *Boletín de la Sociedad Matemática Mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[16] H. A. P. Blom, "An efficient filter for abruptly changing systems," in *The 23rd IEEE Conference on Decision and Control*, pp. 656–658, 1984.

[17] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[18] K. Granström, P. Willett, and Y. Bar-Shalom, "Systematic approach to IMM mixing for unequal dimension states," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 4, pp. 2975–2986, 2015.

[19] S. H. Park, T. L. Song, R. Oh, and J. W. Choi, "A new IMM interacting approach for unequal dimension states for multitarget tracking in cluttered environments," in *2021 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 28–33, 2021.

[20] C. Allig and G. Wanielik, "Unequal dimension track-to-track fusion approaches using covariance intersection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5881–5886, 2022.

[21] S. B. Jr. and A. Belangour, "Multi object tracking: A Survey," in *Thirteenth International Conference on Digital Image Processing (ICDIP 2021)* (X. Jiang and H. Fujita, eds.), vol. 11878, p. 118780I, International Society for Optics and Photonics, SPIE, 2021.

[22] L. Fan, Z. Wang, B. Cail, C. Tao, Z. Zhang, Y. Wang, S. Li, F. Huang, S. Fu, and F. Zhang, "A survey on multiple object tracking algorithm," in *2016 IEEE International Conference on Information and Automation (ICIA)*, pp. 1855–1862, 2016.

[23] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, p. 103448, 2021.

[24] Y. Xu, X. Zhou, S. Chen, and F. Li, "Deep learning for multiple object tracking: A Survey," *IET Computer Vision*, vol. 13, no. 4, pp. 355–368, 2019.

[25] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.

[26] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, 2017.

[27] M. A. Naiel, M. O. Ahmad, M. Swamy, J. Lim, and M.-H. Yang, "Online multi-object tracking via robust collaborative model and sample selection," *Computer Vision and Image Understanding*, vol. 154, pp. 94–107, 2017.

[28] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, 2017.

[29] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, "Poi: Multiple object tracking with high performance detection and appearance feature," in *Computer Vision*

– *ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 36–42, Springer International Publishing, 2016.

[30] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 17–35, Springer International Publishing, 2016.

[31] G. L. Smith and L. A. M. Stanley F Schmidt, "Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle," *National Aeronautics and Space Administration*, 1962.

[32] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI* (I. Kadar, ed.), vol. 3068, pp. 182–193, International Society for Optics and Photonics, SPIE, 1997.

[33] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: A Survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 103–123, 1998.

[34] J. B. B. Gomes, "An overview on target tracking using multiple model methods," Master's thesis, Instituto Superior Tecnico, Universidade de Lisboa, 2008.

[35] A. K. Shastry, H. Sinha, and M. Kothari, *Autonomous Detection and Tracking of a High-Speed Ground Vehicle using a Quadrotor UAV*.

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 21–37, Springer International Publishing, 2016.

[37] Q. Xiangdong and T. Wang, "Visual-based tracking and control algorithm design for quadcopter UAV," in *2019 Chinese Control And Decision Conference (CCDC)*, pp. 3564–3569, 2019.

[38] Y. Liu, Q. Wang, H. Hu, and Y. He, "A novel real-time moving target tracking and path planning system for a quadrotor uav in unknown unstructured out-

door scenes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2362–2372, 2019.

[39] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[40] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), p. 144–152, Association for Computing Machinery, 1992.

[41] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[42] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2778–2788, 2021.

[43] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[44] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2019.

[45] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[46] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[47] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, 2017.

[48] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[49] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding yolo series in 2021," 2021.

[50] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022.

[51] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "YOLOv6: A single-stage object detection framework for industrial applications," 2022.

[52] Ultralytics, "YOLOv8." `https://github.com/ultralytics/ultralytics`. Accessed: 08/05/2023.

[53] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of CNN," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1571–1580, 2020.

[54] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "Panet: Few-shot image semantic segmentation with prototype alignment," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9196–9205, 2019.

[55] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[56] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *arXiv preprint arXiv:1702.03118*, 2017.

[57] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *arXiv preprint arXiv:2109.14545*, 2022.

[58] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, "Single and multiple object tracking using Log-Euclidean Riemannian subspace and block-division appearance model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2420–2440, 2012.

[59] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1838–1845, 2013.

[60] L. Zhang and L. van der Maaten, "Preserving structure in model-free tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 756–769, 2014.

[61] M. Yang, T. Yu, and Y. Wu, "Game-theoretic multiple target tracking," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[62] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[63] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, (Cambridge, MA, USA), p. 91–99, MIT Press, 2015.

[64] H. Luo, W. Jiang, Y. Gu, F. Liu, X. Liao, S. Lai, and J. Gu, "A strong baseline and batch normalization neck for deep person re-identification," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2597–2609, 2020.

[65] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 107–122, Springer International Publishing, 2020.

[66] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," 2016.

[67] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[68] P. Sun, J. Cao, Y. Jiang, Z. Yuan, S. Bai, K. Kitani, and P. Luo, "Dancetrack: Multi-object tracking in uniform appearance and diverse motion," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20961–20970, 2022.

[69] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1858–1865, 2008.

[70] D. Stadler and J. Beyerer, "Modelling ambiguous assignments for multi-person tracking in crowds," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pp. 133–142, 2022.

[71] Y. Du, J. Wan, Y. Zhao, B. Zhang, Z. Tong, and J. Dong, "Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2809–2819, 2021.

[72] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3069–3087, 2021.

[73] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," 2019.

[74] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Systems with Applications*, vol. 172, p. 114602, 2021.

[75] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," 2021.

[76] M. Ahmed, K. A. Hashmi, A. Pagani, M. Liwicki, D. Stricker, and M. Z. Afzal, "Survey and performance analysis of deep learning based object detection in challenging environments," *Sensors*, vol. 21, no. 15, 2021.

[77] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking," 2018.

[78] Y. Cao, Z. He, L. Wang, W. Wang, Y. Yuan, D. Zhang, J. Zhang, P. Zhu, L. Van Gool, J. Han, S. Hoi, Q. Hu, M. Liu, C. Cheng, F. Liu, G. Cao, G. Li, H. Wang, J. He, J. Wan, Q. Wan, Q. Zhao, S. Lyu, W. Zhao, X. Lu, X. Zhu, Y. Liu, Y. Lv, Y. Ma, Y. Yang, Z. Wang, Z. Xu, Z. Luo, Z. Zhang, Z. Zhang, Z. Li, and Z. Zhang, "Visdrone-det2021: The vision meets drone object detection challenge results," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2847–2854, 2021.

[79] Y. Sun, B. Cao, P. Zhu, and Q. Hu, "Drone-based rgb-infrared cross-modality vehicle detection via uncertainty-aware learning," 2021.

[80] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, p. 246309, 2008.

[81] I. Leichter and E. Krupka, "Monotonicity and error type differentiability in performance measures for target detection and tracking in video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2553–2560, 2013.

[82] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[83] *State Estimation in Discrete-Time Linear Dynamic Systems*, ch. 5, pp. 199–266. John Wiley Sons, Ltd.

[84] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises, 4th Edition*. Wiley, 4th ed., 2012.

[85] *Estimation for Kinematic Models*, ch. 6, pp. 267–299. John Wiley Sons, Ltd.

[86] R. Gurajala, P. B. Choppala, J. S. Meka, and P. D. Teal, "Derivation of the kalman filter in a bayesian filtering perspective," in *2021 2nd International Conference on Range Technology (ICORT)*, pp. 1–5, 2021.

[87] S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans, *Introduction to object tracking*, p. 1–21. Cambridge University Press, 2011.

[88] Y. Ho and R. Lee, "A bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, no. 4, pp. 333–339, 1964.

[89] R. R. Labbe Jr., *Kalman and Bayesian Filters in Python*. The MIT License, 2020.

[90] Y. Bar-Shalom, S. Challa, and H. Blom, "IMM estimator versus optimal estimator for hybrid systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 986–991, 2005.

[91] Y. Bar-Shalom, P. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing, 2011.

[92] T. Yuan, Y. Bar-Shalom, P. Willett, E. Mozeson, S. Pollak, and D. Hardiman, "A multiple IMM estimation approach with unbiased mixing for thrusting projectiles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 3250–3267, 2012.

[93] Y. Kurt and H. E. Soken, "Vision-based real-time motion estimation of ground vehicles from aerial imagery," *Journal of Physics: Conference Series*, 2023.

[94] Y. Kurt, S. U. Aydinli, and E. Soken, "Analyzing the effects of visual based tracking algorithms on classical guidance methods," in *2023 10th International Conference on Recent Advances in Space Technologies (RAST)*, 2023.