Global Tracking based Multi-Object Tracking in Complex Environments

by **Yulong Qin**

Thesis submitted in fulfilment of the requirements for the degree of
Master by Research in Computer Science

under the supervision of Qiang Wu

School of Electrical and Data Engineering
Faculty of Engineering and IT
University of Technology Sydney
April 16, 2025

Certificate of Authorship / Originality

I, Yulong Qin declare that this thesis is submitted in fulfilment of the requirements for the award of degree of Master by Research in Computer Science, in the School of Electrical and Data Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: Thursday 17th April, 2025

Acknowledgments

I sincerely appreciate and thank everyone for their continuous support and efforts upon the completion of this thesis.

First, of course, I would like to thank my fiancée Zoey—thank you for being so patient and supportive during this entire process. I have kept you as my motivation since the first day. I also owe a thank you to my cat, Cyber, who provided warmth and relaxation during those intense study marathons.

My sincerest appreciation goes to A/Prof Qiang Wu, my supervisor. The advice, support and positivity that he offered throughout my research and writing process have been crucial in getting my work to where it is today. He has taught me a lot about making the best of every situation, how everything happens for a reason, and that if you go with the flow your life will be much more enjoyable.

I would like to express my utmost gratitude to my parents and family for standing by me and motivating me, as this has brought the most strength in it all. This would have been impossible without them.

My gratitude is also due to my fellow FEIT students; their companions made this academic journey a lot less lonely.

I also used AI-based tools to improve my language so that my thoughts could flow more naturally.

So, thank you again to everyone who encouraged and supported me along the way! I could not have done this without you.

Yulong Qin

April 16, 2025
Sydney, Australia

Abstract

Multi-object tracking (MOT) is a crucial task in computer vision with widespread applications, such as autonomous driving and intelligent surveillance. Traditional MOT approaches based on convolutional neural networks (CNNs) and Kalman filters are relatively simple to implement but often fail in complex environments with dynamic target motion. Recently, transformer-based methods have shown great promise in improving MOT performance by leveraging global attention. However, frame-by-frame association strategies remain vulnerable to unstable detections, limiting their effectiveness in real-world scenarios.

To address these challenges, the Global Tracking Transformer (GTR) was proposed to associate detection results across multiple frames through similarity computation. While GTR improves global consistency, it still suffers from three key limitations: (1) high dependency on accurate detection results; (2) an inflexible single-anchor query mechanism, which struggles in dynamic environments; and (3) lack of fine-grained attention modeling within the transformer.

This thesis proposes an enhanced transformer-based tracking framework to overcome the above limitations. First, a Kalman filter-based prediction module is introduced to provide reliable location estimates when detections are missing or inaccurate. These predictions, together with valid detections, are used as transformer inputs, improving temporal continuity. Second, a Convolutional Block Attention Module (CBAM) is integrated into the transformer to refine feature extraction by emphasizing informative channels and spatial regions. Third, a multi-query strategy, termed MQTFormer, is developed to capture motion variations by leveraging multiple trajectory queries across time. Furthermore, a multi-weight fusion mechanism combines feature similarity, detection confidence, and adaptive adjustment to enhance robustness in the matching process.

Experimental evaluations on standard MOT benchmarks, including MOT17 and TAO, demonstrate that the proposed method consistently outperforms state-of-the-art approaches in terms of accuracy and stability, particularly in challenging scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents a detailed review of essential definitions, evolution and recent trends in the field of Multi-Object Tracking (MOT). We start with the fundamental concepts of MOT: the common goals of tracking several objects through time in a dynamic environment, and we discuss the wide spectrum of applications where MOT is an important component. From intelligent surveillance and autonomous driving to sports analytics, medical imaging [1], etc., MOT finds various important applications in many domains.

This section goes deeper into the emergence of MOT methods, from traditional approaches that relied more on handcrafted features and optimization based tracking model to recent deep learning-based solutions which build upon neural networks as well as sophisticated data association algorithm. This history of the research shows that, over decades, MOT has evolved in increasingly sophisticated models to deal with complex tracking scenarios (e.g., occlusions, appearance changes or scale variations) and real-time processing.

After having set the historical context, we shift our attention to modern state-of-the-art methods in MOT. We focus specifically on progress made in the last few years, mostly using Transformer-based networks that leverage self-attention mechanisms for enhanced spatial and temporal reasoning, leading to major improvements on tracking performance.We will examine the performance of these methods and analyze their strengths, weaknesses, and key directions for future work — a topic that is parallel to the motivation behind this thesis.

With this context in mind we formulate the research question that will be answered in this thesis. In this tutorial, we describe particular difficulties in Multi-Object Tracking (MOT), including occlusion handling, identity switching between frames and complexity-accuracy trade-off. These challenges also reveal the limitations of current models and motivate developments in new approaches to improve tracking robustness in both stationary and dynamic real-world environments with variable amounts of noise.

Lastly, the chapter ends with a short summary of the thesis structure.We note the structure of later chapters, each building off that background to cumulatively construct, test and validate our method to be presented. Following this clear and structured methodology, we can systematically tackle the research problem to hopefully contribute something novel to the field of Multi-Object Tracking.

1.1 Overview of object tracking

1.1.1 Basic concepts of object tracking

Object tracking is the process of continuously monitoring the position and movement of a specific object within a series of images or video frames [2]. The primary goal is to locate the object in each subsequent frame after it has been initially detected. This technique is widely used in various fields, including surveillance systems, autonomous driving, and video analysis. The main challenge of object tracking lies in consistently identifying the object as it moves, changes appearance, or becomes partially obscured.

The multi-object tracking (MOT) task is tracking multiple objects across a sequence of frames. Differing from the pure single-object tracking (SOT) task, which is track a unique target object, MOT includes the ability to distinguish multiple objects that may be moving along diverse routes or trajectories with different speeds. In crowds, objects might overlap or appear similar and thus erase their track from each outline — making this task particularly onerous. MOT has extensive use cases — ranging from monitoring public spaces, sports events to traffic surveillance

The process of object detection typically begins with processing the input data, starting with object recognition [3] to detect and locate the objects of interest in the image. Subsequently, image classification [4] is performed (for example, using deep learning models like YOLO or SSD [5]) to quickly and accurately identify the object classes. Next, object localization [6] is carried out to determine the precise position of objects in the image, assisting in the subsequent detection stage. These steps collectively constitute the complete object detection process, providing a foundation for tracking tasks, as shown in Figure1.1.

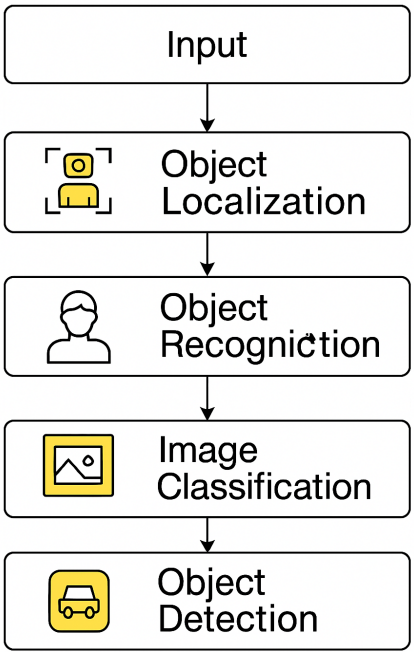

Figure 1.1: A typical object detection process, illustrating how an input image is processed through various stages, including feature extraction, region proposal, classification, and localization. Each stage contributes to identifying and accurately locating objects within the image, providing bounding boxes and class labels for detected objects. This process is fundamental in applications requiring precise object recognition.

After features are extracted, the system moves on to data association, which involves matching detected objects across consecutive frames. This is crucial for ensuring that the correct object is being tracked, especially in situations where multiple objects are present. Algorithms like the Hungarian algorithm [7] are often used to effectively link detections across frames. The core of the tracking process is predicting the object's future position in subsequent frames. Motion models, such as the Kalman filter [8] or particle filter [9], are typically employed to make these predictions based on the object's previous movements.

During tracking, the system needs to incorporate data from new frames to keep updating its model. This assists in refining the predictions whilst still holding accuracy through time. Handling occlusions or the situation when an object is temporarily hidden by another object or moves out of the frame, poses a huge challenge during this process. Once the object disappears due to occlusion, the handling of such situation is vital as this allows our system to re-identify and continue tracking down the same object once it come into view again.

There are many problems in object tracking that should be tackled to obtain good performance. One of the biggest challenges is occlusion, meaning that an object was blocked by another (partial or total). This makes it hard for the tracking algorithm to keep a correct track. Managing changes in appearance is another big difficulty. Appearance of objects can change due to illumination, view angle and deformation, making it difficult to identify and track them consistently.

Real-time processing is another challenge and depending on the application, tracking has to be done for live video streams. Due to this, highly efficient algorithms that can process through frames as rapidly as possible and yet maintain accuracy are needed. Moreover, when there is a scene with many visual similar objects, it will be difficult to distinguish between them which increase probability of tracking error. The most important thing is to find the correct initialization of tracking in the first frame. Mistakes at this early stage can introduce tracking errors that perpetuate through out a sequence.

These adversities exist for most of the object tracking methods and become important focuses in making a tracking algorithm more effective and robust.

1.1.2 Applications of MOT

Smart Surveillance [10] In public areas like airports, train stations, and shopping malls, deploying Multi-Object Tracking (MOT) technology enables continuous real-time monitoring of large crowds, maintaining consistent IDs for each individual. This capability is essential for anomaly detection, such as recognizing if an elderly person has fallen, thereby triggering immediate alerts to medical and security personnel for prompt assistance. As illustrated in Fig.1.2, an intelligent surveillance system at a station gate utilizes MOT in conjunction with thermal imaging to assess passengers' body temperatures. The system instantly identifies and labels individuals with color-coded bounding boxes based on temperature status—green for normal and red for elevated readings. This setup not only facilitates smooth crowd flow but also reinforces public health safety measures by detecting potential health issues and enabling authorities to act swiftly when abnormalities are flagged.

Sports Analysis[12] Multi-Object Tracking (MOT) systems based on behavior recognition are revolutionizing sports analytics by offering in-depth insights into athletes' movements during training and competitions. By capturing precise joint positions, skeletal movements, and postural dynamics, MOT systems provide coaches and athletes with comprehensive biomechanical

Figure 1.2: The intelligent surveillance system at the station gates continuously monitors passengers' body temperatures as they pass through. The system displays each passenger's temperature above their bounding box, with color coding to indicate status: green for normal temperature (e.g., 35.2°C) and red for elevated temperature (e.g., 37.5°C), signaling a potential health concern. This setup allows for efficient, real-time temperature screening at public transit points, enhancing health and safety measures [11].

data for performance enhancement. For instance, as shown in Fig.1.3, a runner on a treadmill is monitored via body sensors, with the data rendered in real-time on the right side as a skeletal model that highlights joint motion. This detailed feedback enables coaches to identify inefficiencies, fine-tune training regimes, and even anticipate injury risks. The combination of MOT and sensor technology allows for scientifically backed, individualized training programs that help athletes optimize performance and unlock their full potential by addressing specific technical areas.

Autonomous Driving[14] Multi-Object Tracking (MOT) plays a pivotal role in autonomous driving technology by enabling vehicles to detect, classify, and monitor various objects on the road, such as other vehicles, pedestrians, and traffic signals. This capability equips autonomous vehicles with situational awareness, allowing them to make informed navigation decisions while ensuring passenger safety. As illustrated in Fig. 1.4, the system identifies multiple road elements in real-time, including cars, traffic lights, and pedestrians, each labeled for clarity. This data helps the vehicle's control system plan safe, efficient routes while adjusting to the surrounding traffic environment. By dynamically monitoring all road users and adjusting driving strategies accordingly, MOT enhances the vehicle's ability to respond to complex road situations.

Drone Surveillance [16] Multi-Object Tracking (MOT) is extensively used in drone surveillance due to drones' capability for broad, flexible monitoring. By leveraging MOT, drones can track multiple ground targets over vast areas, making them highly effective for diverse applications, including agricultural monitoring, wildlife protection, and disaster response. In agriculture, drones equipped with MOT can monitor crop health and livestock movement. For wildlife conservation, MOT aids in tracking endangered species without disturbing them, while in natural disaster scenarios, drones can locate people in need of rescue, monitor changing environmental conditions, and assist relief efforts by providing real-time situational awareness.

Augmented Reality (AR) [17] Augmented Reality (AR) technology benefits significantly from Multi-Object Tracking (MOT) by enhancing users' interaction with their surroundings

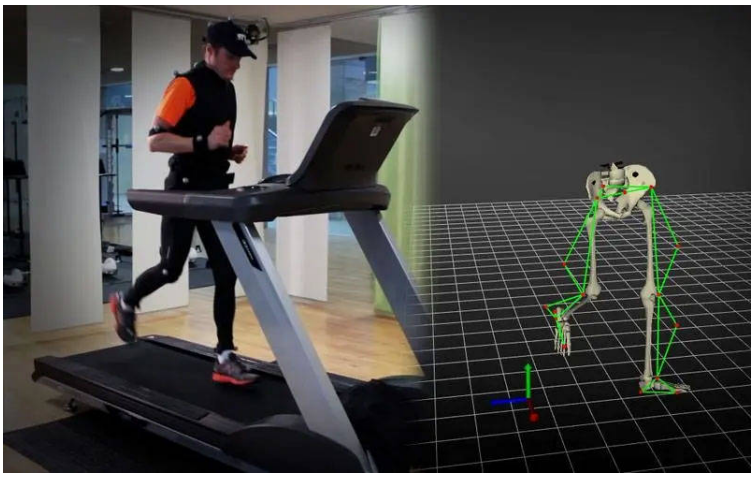

Figure 1.3: A man monitors his physical activity on a treadmill through the use of body sensors and motion tracking technology. The system captures his movement in real-time, rendering a skeletal model on the right that accurately tracks his joint positions and movements, thereby providing valuable biomechanical data for performance analysis and physical health assessment. [13].

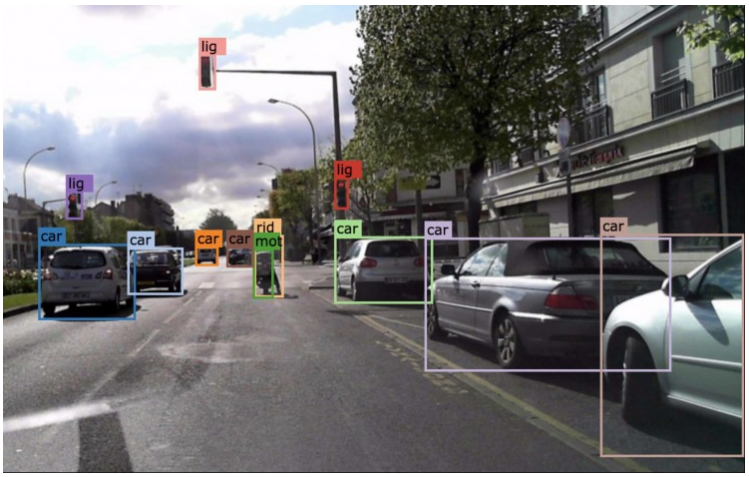

Figure 1.4: An example of autonomous driving technology where multiple vehicles, traffic lights, and road users are detected and tracked in real-time using Multi-Object Tracking (MOT). Each object is identified and labeled, such as cars, traffic lights, and pedestrians, providing the autonomous vehicle with situational awareness to make informed driving decisions [15].

through accurate target identification. For instance, in AR navigation systems, MOT assists users by highlighting key environmental elements, such as road signs and pedestrian indicators, helping to navigate complex routes. This feature is particularly beneficial in crowded urban areas, where MOT technology can maintain a focus on relevant objects, improving the user's orientation and decision-making capabilities. By facilitating a more immersive and precise AR experience, MOT helps bridge the gap between virtual information and the real world, enabling users to interact with digital data intuitively.

1.2 Representative Approaches of Object Tracking

The history of multi-object tracking (MOT) dates back to the 1960s [2], with early work focused on video surveillance and simple detection tasks. Limited computing power led researchers to use models like Kalman and particle filters for efficient tracking in noisy environments.

As hardware improved, methods like Joint Probabilistic Data Association (JPDA) [18] and Multiple Hypothesis Tracking (MHT) [19] emerged to handle challenges like object confusion and complex scenes. However, traditional methods struggled with occlusions, object interactions, and background noise.

With the rise of deep learning, MOT methods shifted to data-driven approaches. Deep learning models, especially Convolutional Neural Networks (CNNs) [20], allowed for better feature extraction and handling of complex scenes [21]. Recently, Transformers have further enhanced tracking with improved global modeling.

Today, MOT methods are split into traditional, probabilistic approaches and deep learning-based methods, each with distinct advantages. Future research may focus on combining these strengths to tackle increasingly complex scenarios.

1.2.1 Traditional object tracking

Non-deep-learning methods for object tracking can be summarized into a few key representative solutions, each with a unique approach to the tracking process.

Background subtraction [22] is a method that identifies moving objects by detecting differences between a reference background image and the current frame. This method is particularly effective in static camera setups, where the background remains largely unchanged. The primary goal is to separate the foreground (the moving objects) from the background, allowing for straightforward detection and tracking.

Optical flow [23] involves calculating the apparent motion of objects between consecutive frames by analyzing the motion of pixels. This method provides detailed motion information, such as the direction and speed of movement, which is crucial for tracking. However, optical flow can be sensitive to noise and may struggle in scenarios with fast-moving objects or complex backgrounds.

Mean-shift [24] is a method that tracks objects by iteratively shifting a search window towards the densest region in the feature space (usually color or texture). This technique is particularly effective for tracking non-rigid objects that may change shape but maintain consistent visual features. The mean-shift algorithm is robust in scenarios where the object's appearance remains relatively constant.

Kalman filter [8] is a probabilistic model that predicts the future position of an object based on its previous states, taking into account both the object's dynamics and the uncertainty in the measurement process. The Kalman filter is well-suited for tracking objects with linear motion in environments with low noise. It is widely used for its efficiency and ability to provide smooth estimates of object trajectories, as illustrated in Figure 1.5.

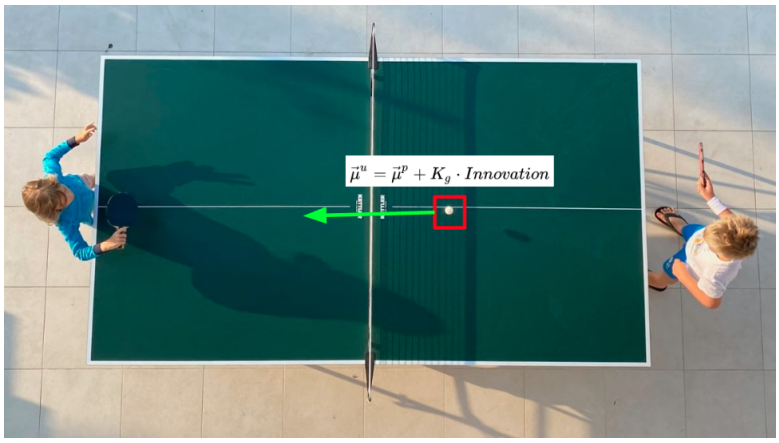

Figure 1.5: An illustration of a table tennis game where the Kalman filter is used to estimate the position and trajectory of the ball. The Kalman filter algorithm combines predictions and observations to calculate a more accurate position (indicated by the red box), allowing the system to anticipate the ball's movement dynamically, enhancing the tracking accuracy in fast-paced environments [12].

1.2.2 Deep-learning based object tracking

Deep learning is a branch of machine learning that uses neural networks with many layers to automatically learn patterns and representations from raw data [25]. The core idea is to allow the model to learn the necessary features for tasks such as classification, detection, or tracking without manual intervention. Each layer in a deep neural network transforms the input data into more abstract and complex representations, making deep learning particularly effective for tasks like image processing and sequence prediction, which are essential in object tracking.

Object tracking using deep learning involves a sequence of steps that work together to track objects accurately across video frames. The process typically starts with detecting objects in the initial frames, where the system identifies and localizes the objects that will be tracked throughout the sequence. This initial detection is crucial because it sets the stage for subsequent tracking.

After detection, the model extracts relevant features [26] from the detected objects. These features capture important aspects of the object's appearance, such as texture, color, and shape, along with more abstract characteristics learned by the neural network. These feature representations are critical for distinguishing objects from one another and maintaining consistent tracking, as they provide the necessary information for the system to follow the objects across frames.

Once features are extracted, the system needs to associate detected objects across consecutive frames. This step involves matching the features of objects from one frame to the next, ensuring continuity in tracking. Temporal association is particularly important when objects move,

change appearance, or when multiple objects interact within the scene. Without effective association, tracking accuracy would degrade over time.

To maintain continuous tracking, the model often predicts the future positions of objects based on their movement patterns. This trajectory prediction [27] step can use techniques like recurrent neural networks (RNNs) [28] or attention mechanisms to model temporal dependencies. These approaches allow the system to anticipate where objects will appear in upcoming frames, even if they are temporarily occluded or move out of the frame, enhancing the robustness of the tracking process.

It does this by continually updating its understanding of all aspects of each object as new frames go through. Such adaptive updating guarantees the accuracy of the tracker and adapts to various appearances/disappearances of object throughout their motions in the scene. This is an adaptive process which without cannot lead to successful object-handing as it grows over time.

Deep learning-based object tracking methods are generally divided into two categories: local tracking and global tracking [29]. Each approach has its distinct methodology and applications, offering different advantages depending on the specific tracking scenario.

Recently, emerging foundation models such as the Segment Anything Model (SAM) [30] have gained significant attention in the computer vision community. SAM enables prompt-based, fine-grained object segmentation, and its strong generalization has inspired new approaches for instance tracking and segmentation-based MOT. Some recent studies have explored integrating SAM features with conventional trackers to enhance robustness in challenging environments, especially in segment-level tracking scenarios. Although SAM-based methods are still in the early stages of MOT applications, they represent a promising direction for combining detection, segmentation, and tracking into a unified paradigm.

1.2.3 Local Tracking

Based on the strategy used to associate object detections across frames, deep learning-based object tracking methods can be generally divided into two categories: local tracking and global tracking. This section first discusses local tracking approaches.

Local tracking focuses on detecting and associating objects locally, frame by frame, under the assumption that objects move smoothly and predictably over short time intervals. These methods rely on the principle that object motion is relatively continuous and deterministic within a narrow temporal window. As such, they are effective when large or abrupt state transitions are unlikely to occur between frames.

One of the key strengths of local tracking lies in its responsiveness to short-term object appearance or position changes. Since the tracking window is short and localized, the system can quickly adapt to visual changes, making local tracking particularly suitable for real-time applications where rapid response is essential.

Another advantage of local tracking is its computational efficiency. Because it only processes a few frames at a time, the method is lightweight and well-suited for resource-constrained environments. This makes local tracking highly applicable in real-time systems such as video surveillance, autonomous vehicles, and embedded devices [31].

However, local tracking also has inherent limitations due to its strong reliance on contextual

information in the immediate vicinity of the object. In complex scenes involving occlusions or dense object interactions, local methods may struggle to maintain accurate associations, as they lack access to broader temporal and spatial context.

1.2.4 Global Tracking

In contrast, global tracking methods adopt a more holistic approach by analyzing longer sequences of frames to construct a comprehensive understanding of object trajectories [29]. Rather than relying solely on short-term motion cues, global tracking integrates information from a larger temporal window, which significantly enhances robustness to occlusions, appearance changes, and complex object interactions.

A key advantage of global tracking is the incorporation of long-term context. By aggregating information across time, these methods can maintain consistent object identities even when targets disappear temporarily or move out of view. When the object reappears, the system can re-identify it based on its accumulated features and motion patterns [32].

Global tracking is particularly useful in dynamic, cluttered environments such as crowded public spaces or traffic intersections, where multiple objects frequently overlap or interact. Its global perspective allows for better scene understanding and more stable tracking performance over extended durations [33], [34].

However, this robustness comes at a cost. Global tracking is generally more computationally intensive than local tracking, as it requires processing and storing information from longer frame sequences. This increased complexity can lead to slower processing speeds, making it less ideal for applications with strict real-time constraints unless optimization techniques are applied.

Despite the trade-offs, global tracking remains a valuable approach for applications demanding long-term stability and resilience, particularly when object occlusions and re-identifications are frequent and unavoidable.

1.2.4.1 Representative Works of Local Tracking

One of the notable methods in local tracking is TransTrack[35], which employs a Transformer-based architecture to connect detected objects across frames. By utilizing attention mechanisms, TransTrack captures both spatial and temporal dependencies, allowing the model to maintain object identities even in challenging scenarios such as occlusions. Despite its effectiveness in handling complex tracking tasks, the complexity of the Transformer architecture leads to increased computational demands, which can limit its efficiency in real-time applications.

Another significant work is MOTR[36]. MOTR integrates detection and tracking into a unified framework by leveraging the power of Transformers. The model utilizes learnable queries to track multiple objects simultaneously, making it adept at addressing challenges like occlusions and sudden movements. However, the complexity of the Transformer model used in MOTR results in substantial computational resource requirements, which poses a challenge for deployment in resource-constrained environments.

1.2.4.2 Representative Works of Global Tracking

In the domain of global tracking, GTR (Global Tracking Transformer)[29]stands out as a prominent method. GTR begins by obtaining all detection results from the initial detection stage. It

then employs trajectory queries as input to the decoder, where these queries are matched with features of all detected objects within the Transformer architecture[37].This approach assigns a unique trajectory to each object, ensuring precise tracking across the entire video sequence. However, GTR's performance is highly dependent on the accuracy of the initial detection results, which can be a limitation in less reliable detection systems. Additionally, the method relies on a query mechanism that is overly anchored to the last frame of a sliding window, and the Transformer's attention mechanisms[37] do not sufficiently account for channel and spatial dimensions, which could affect the model's performance in handling complex visual features.

1.2.4.3 Key Differences Between Local Tracking and Global Tracking

Local and global tracking methods differ in several key aspects. Each has its own advantages and challenges.

Local tracking focuses on short-term, frame-by-frame analysis. It mainly considers the immediate context around the object. This allows for quick and efficient tracking but limits its ability to understand long-term object behavior. In contrast, global tracking takes a broader view. It analyzes longer sequences of frames, which helps build a more complete understanding of object trajectories and interactions.

When dealing with occlusions, local tracking often struggles[38]. If an object is temporarily out of view, it can lose track easily. Global tracking, however, is more robust. It maintains long-term associations, making it better at re-identifying objects when they reappear.

In terms of computational demands, local tracking is more efficient. It processes only a few frames at a time, making it suitable for real-time applications. Global tracking, on the other hand, is more computationally intensive. It processes a larger volume of data, which can slow it down but improves accuracy in challenging scenarios.

Local tracking also depends heavily on immediate visual context. This makes it more prone to errors during rapid scene changes or when objects interact closely. Global tracking, by considering the entire scene, handles complex interactions and changes more effectively.

Finally, local tracking, while fast, is less adaptable in dynamic environments. It can struggle when objects change appearance or behavior. Global tracking is more flexible[39]. It integrates information over time, making it more resilient to occlusions, interactions, and other challenges.

1.3 Research Problems

In this thesis, we target a line of studies and enhancements based on Global Tracking Transformer (GTR). In particular, it focuses on the following problems in GTR model: over-reliance on detection results, we have to hold last frame of a sliding window as query-frame, and finally without channel and spatial attention mechanism for feature extraction. In summary, the problems are:

- **Dependence on Detection Results, Lacking the Ability to Supplement Failed Detections**
  Tracking in GTR heavily depends on detections outputs. GTR does not have a means of predicting or filling in the gaps where targets would be when detectors fail due to occlusion, lighting changes, or large amounts of appearance change on the target. This

dependence leads to broken paths, especially in cases where continuity matters and reveals one of GTRs biggest weaknesses when it comes to recovering from detection failures.

- **Anchoring the Last Frame as the Query Frame, Limiting Flexibility**
  In GTR, the last frame within a sliding window acts as the query frame. This design assumes that the final frame carries the most relevant information. However, such a fixed mechanism may hinder its applicability in dynamic scenarios, especially when objects move rapidly or unpredictably. If the scene changes suddenly, the system may fail to correctly match objects across frames and compromise temporal continuity.

- **Inability to Use Channel and Spatial Attention to Aid in Feature Extraction**
  GTR also does not integrate one of the most important mechanisms, which is channel and spatial attention for feature elicitation. GTR has no mechanism to focus on relevant target features from a background scene, so that it is difficult to generalize the GTR in complex backgrounds or multiple object tracking. Neglecting low-contrast is introducing the system imprecise features which make it difficult to differentiate between foreground and background thus reducing the precision and robustness of object tracker.

The solutions proposed in this thesis for each problem are as follows:

- **Problem 1: Dependence on Detection Results, Lacking the Ability to Supplement Failed Detections**
  Solution: Prediction Module
  This thesis proposes an easy-to-deploy prediction module based on the Kalman filter[8]. This module is capable of providing more accurate proposals during detection failures, thus extending and improving the accuracy of the original trajectory.
  Rationale:
  One of the most common models in target tracking which is used for linear dynamic systems is Kalman filter. It is able to predict a state of target with noise and uncertainties, and correct it. Hence, whenever the detector fails, the Kalman filter can use previous target trajectory information to effectively predict reasonable future positions of a target compensating for its failed discovery.

- **Problem 2: Anchoring the Last Frame as the Query Frame, Limiting Flexibility**
  Solution: Enhanced Query Design
  This thesis proposes a more rigorous and refined query design, enhancing the robustness of similarity calculations through a multi-query design, with further optimization of the specific calculation mode.
  Rationale:
  A multi-query[29] design allows for target matching and identification from multiple perspectives, improving adaptability to complex scenarios. By introducing more feature dimensions and optimizing similarity calculations, this design can more accurately distinguish targets, reducing ID confusion and tracking errors.

- **Problem 3: Inability to Use Channel and Spatial Attention to Aid in Feature Extraction**
  Solution: Transformer with Enhanced Attention Mechanisms
  This thesis redesigns the Transformer structure, integrating the Convolutional Block Attention Module (CBAM)[40] to provide stronger channel and spatial attention mechanisms.
  Rationale:

CBAM enhances feature extraction by applying channel and spatial attention[41], allowing the model to focus more effectively on key features of the target while ignoring irrelevant background information. This improved attention mechanism significantly increases the accuracy and robustness of the tracking process.

1.4 Structure of Thesis

The overall structure of the following thesis chapters is:

Related work is presented in Chapter 2, followed by three thematically structured chapters. Each of these chapters begins by introducing key theoretical foundations, which are then examined in the context of specific studies and publications.

The prediction module and multi-query information are introduced in Chapter 3, including but not limited to: motivations, methodology, experiments etc.

Another technical chapter, Chapter 4 delves into the reasons for CBAM as well as its approach and experiments. To this end, each technical chapter will be followed by a summary section to review and summarize the methodology.

In Chapter 5, we summarize the contents of the whole thesis and give an outlook on future work.

Chapter 2

Related Work

In this section, we explore the fundamental ideas that lay the foundation for the research presented in this thesis. This foundational review introduces key theories and terminologies in a sequential manner, aiming to enhance understanding of the methodologies and analyses discussed in subsequent chapters. Our goal in covering these fundamentals is to provide readers with sufficient context to understand the rationale behind our method design, and to enable them to follow the technical details more effectively.

Chapter 2 then presents a review of the literature, highlighting major studies, theories, and methods that have shaped the field. Rather than adhering to a conventional taxonomy based solely on model types, this review organizes the literature according to task stages and conceptual themes identified in advance. This structure aims to facilitate a clearer understanding of how various innovations have emerged and evolved to form the current state of knowledge. The theoretical foundations, advantages, and limitations of each study are examined, particularly in relation to the techniques adopted in this thesis.

Together, this introduction and the literature review that follows are intended to provide readers with a comprehensive overview of the field. This foundation supports a smoother transition to the technical discussions in the later chapters, which will build upon both the core concepts introduced here and the relevant scholarly work in the domain. While related work is often presented by categorizing methods based on model types or algorithmic families, this chapter adopts a task-oriented structure. Specifically, the literature is organized around the functional components and conceptual stages of the object tracking pipeline, which better reflects the logical progression of design principles used in this thesis. This approach is intended to provide readers with a more intuitive understanding of how different methods address specific challenges, and how these contributions are related to the development of the proposed framework.

2.1 Kalman Filter Theory in Object Tracking

The Kalman filter is one of the most widely used models in computer vision, particularly in object tracking within dynamic environments. Although it was initially developed for engineering applications such as navigation and control, it has since been adapted for visual tracking tasks, allowing systems to track pedestrians, vehicles, or athletes by estimating their positions—even when detections are noisy or partially occluded.

The filtering process begins with an *initial estimate* of the object's state, such as position or

velocity, as shown in Figure 2.1. This initial state is defined as

$$\hat{\mathbf{x}}_0 = \text{Initial guess}, \tag{2.1}$$

which provides the baseline that initializes the tracking procedure. In vision-based applications, this initial guess may be based on the first observed location of the object.

Each iteration of the Kalman filter involves a *prediction* step that forecasts the object's state in the next frame based on its current state. The predicted state at time $k+1$ is calculated by applying the state transition model:

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k, \tag{2.2}$$

where $\mathbf{A}$ is the state transition matrix representing the assumed motion model (e.g., constant velocity), and $\mathbf{B}\mathbf{u}_k$ accounts for any external control inputs. This prediction helps maintain trajectory continuity, even when the object is temporarily lost due to occlusion.

To correct this prediction, the Kalman filter incorporates new observations of the object, denoted as $\mathbf{z}_k$. These measurements are typically obtained from object detection results in frame k, and are represented as

$$\mathbf{z}_k = \text{New data from object detection in frame } k. \tag{2.3}$$

Since visual measurements are often noisy or partially missing, the filter applies a correction by computing the *Kalman Gain*, which determines how much the new measurement should influence the final estimate. The Kalman Gain is computed as

$$\mathbf{K}_k = \mathbf{P}_k^{-}\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^{-}\mathbf{H}^T + \mathbf{R})^{-1}, \tag{2.4}$$

where $\mathbf{P}_k^{-}$ is the prior estimate of the state covariance, $\mathbf{H}$ is the observation model, and $\mathbf{R}$ denotes the measurement noise covariance. The Kalman Gain effectively balances the confidence in the prediction against the reliability of the new measurement.

Using this gain, the filter proceeds to update the predicted state. The *state update* step refines the estimated position by incorporating the measurement information as follows:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^{-}), \tag{2.5}$$

where $\hat{\mathbf{x}}_k^{-}$ is the prior prediction, and the term $\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^{-}$ represents the innovation or residual between prediction and measurement.

Following the state update, the filter also updates the estimate of the error covariance matrix to reflect improved certainty. The updated covariance is given by

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^{-}, \tag{2.6}$$

where $\mathbf{I}$ is the identity matrix. This updated error covariance provides a refined measure of uncertainty associated with the new estimate.

In computer vision tracking applications, this recursive estimation process is repeated at every frame, allowing the filter to adapt to the motion of objects over time. Even when an object is temporarily occluded or exits the frame, the Kalman filter can maintain a reliable prediction of its position, effectively "bridging the gap" until the object reappears.

Thanks to this robustness and efficiency, the Kalman filter has become a cornerstone technique in real-time tracking systems used in autonomous driving, surveillance, and human activity analysis. It enables smooth and accurate tracking even under imperfect observation conditions.

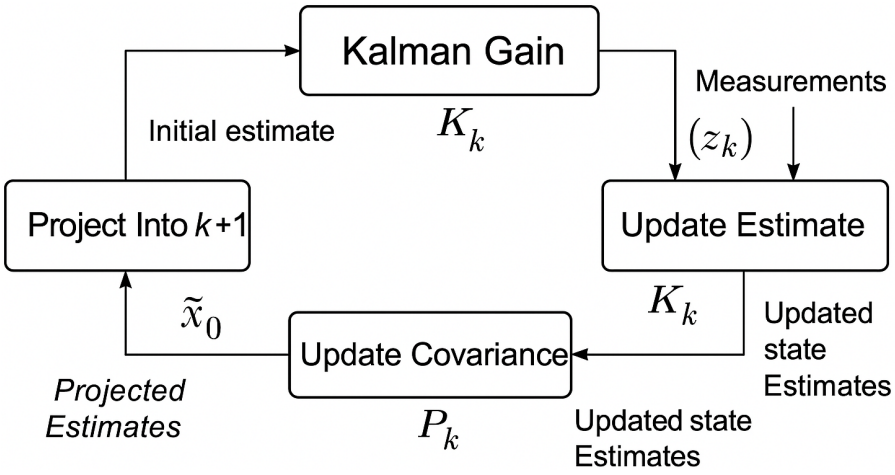

Figure 2.1: The Kalman Filter process in object tracking, illustrating each step of the recursive estimation cycle. The cycle starts from an initial state estimate $\hat{\mathbf{x}}_0$, and proceeds through prediction, measurement update, and error covariance adjustment. Symbols include: $\hat{\mathbf{x}}_k^-$ (predicted state), $\hat{\mathbf{x}}_k$ (updated state), $\mathbf{z}_k$ (observation), $\mathbf{K}_k$ (Kalman gain), and $\mathbf{P}_k$ (error covariance). This pipeline enables robust tracking under uncertainty and partial observations[8].

2.2 Tracking Based on Kalman Filter in Contemporary Object Tracking Solutions

2.2.1 SORT

SORT (Simple Online and Realtime Tracking)[42] is a multi-object tracking algorithm introduced by Alex Bewley in 2016. Its primary goal is to simplify the tracking process and improve real-time performance. As illustrated in Figure 2.2, the SORT algorithm begins with an input video sequence, processed by an object detector[2] (such as YOLO or Faster R-CNN), which detects objects in each frame and generates bounding boxes. Since SORT relies entirely on the detector's output, the accuracy of detections directly influences tracking performance.

Once objects are detected, the data association module[2] matches detected objects across frames to maintain consistent identities over time. SORT uses the Hungarian algorithm for matching, minimizing costs based on the Euclidean distance between predicted positions and new detections.

The core of SORT's tracker is the Kalman filter[8], comprising two key modules: prediction and update.In the prediction module, the Kalman filter estimates the next state (position and velocity) of each object, maintaining continuity even during detection failures, such as occlusions. In the update module, new detections are used to adjust the estimated state, refining the predicted positions and velocities to reduce errors. Unmatched detections initialize new Kalman filters, while unmatched predictions are removed if they persist for several frames.

SORT has been widely used in real-time applications like traffic monitoring and sports analysis. However, its limitations include susceptibility to identity switches in crowded scenes, as it relies solely on object positions without appearance-based features. Detection issues, such as missed detections or false positives, also impact its performance.

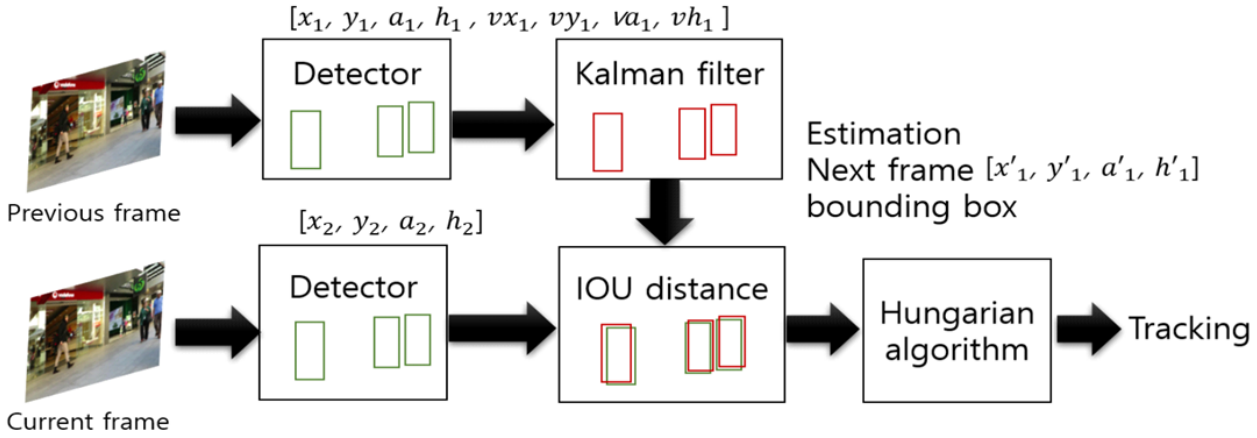

Figure 2.2: The pipeline of the Simple Online and Realtime Tracking (SORT) algorithm. Given a video sequence, each frame is processed by an object detector to obtain bounding boxes. A Kalman Filter predicts the positions of previously tracked objects in the next frame. These predicted positions are matched with current detections using Intersection-over-Union (IOU) distance. The Hungarian algorithm solves the assignment problem and links detections to existing tracks, enabling real-time multi-object tracking.

2.2.2 DeepSORT

To address SORT's limitations in handling occlusions and distinguishing similar objects, Wojke et al. introduced DeepSORT in 2017[43]. As shown in Figure 2.3, DeepSORT extends SORT by adding deep learning-based appearance features, enhancing robustness and accuracy. The main difference is the addition of a deep neural network for appearance feature extraction, allowing DeepSORT to manage complex tracking scenarios more effectively.

At this stage, the object detector generates a bounding box for every frame in exactly the same way that SORT does. Unlike SORT, DeepSORT adds an additional appearance feature extraction module that offers a convolutional neural network (CNN) based encoding of each detected object into a high dimensional feature vector [8]. Sub-types that describe visual features such as color, texture and shape — which allows data to be linked where things may look similar or come in close proximity.

Positional and appearance info are then fused, and the combined information is passed to Kalman filter that keeps the state of each tracked object[44]. For example, during the prediction step, the Kalman filter predicts the object's next position based on previous positions to provide continuous tracking over barriers. The update step is the second part of the Kalman filter, where with new detections the predicted state is corrected by reducing noise in position and velocity.

In DeepSORT, data association matches observations to tracklets by fusing positional and appearance information with the Hungarian algorithm and a matching cascade. DeepSORT, which uses a cost matrix with Euclidean and cosine distances between feature vectors in order to enable more precise distinguishing of similar objects as well as tracking in crowded scenes.

For matched objects, the Kalman filter updates their state with both position and visual information. Kalman filters are initialized new objects and unmatched frames over a threshold will remove lost objects.

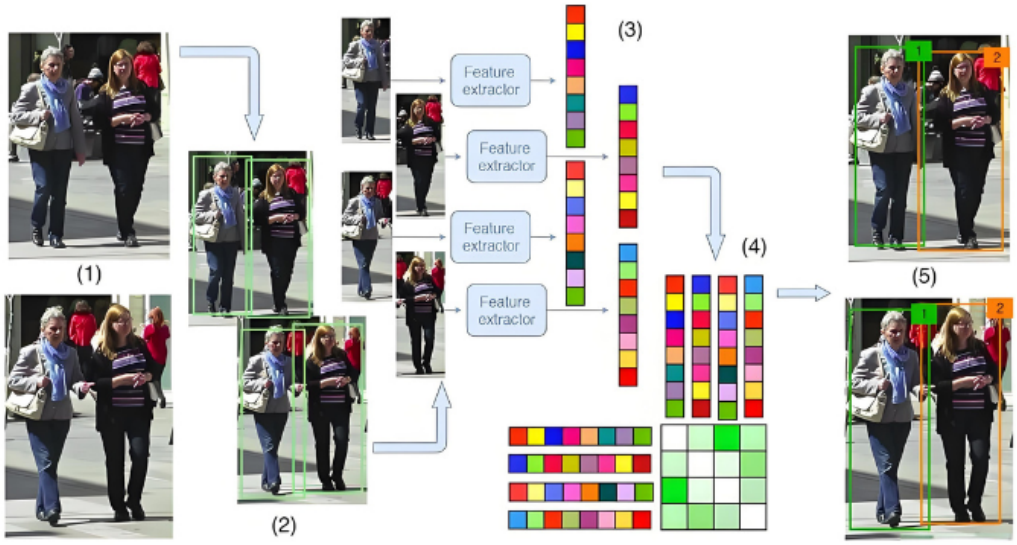

Figure 2.3: The process of DeepSORT begins with a video sequence input, where each frame is analyzed by an Object Detector to generate Detections. Simultaneously, a Deep Appearance Descriptor module extracts Appearance Features to enhance object re-identification across frames. The Cost & Gate module combines detection data with appearance features, passing it to the Matching Cascade, which links detections to existing tracks in the Features Bank. The Kalman Filter predicts object positions for unmatched tracks, and the Gate module updates the tracks accordingly, maintaining accurate tracking across frames.

2.2.3 ByteTrack

As shown in Figure 2.4, ByteTrack[45] operates through a multi-step process that handles both high-confidence and low-confidence detections in two stages of data association, ensuring accurate and robust multi-object tracking.

Input is then the image sequence that indicates continuous video frames. In the first module of ByteTrack, raw video frames are passed onto the next Module which is detection.

The Detector module detects the objects from each frame of the image sequence. The detections received out of the detector include the object bounding boxes and their respective confidence scores. These will then be used in the next association step to track frames.

After detections are acquired, they are forwarded to the First Association module. Here, the system is required to associate tracklets (predicted) with high score detections from the last frame. The detections with confidence score above a certain threshold are considered as the high score detections which are true in nature with better reliability. First Association – The first association tries to match all tracklets with high-confidence detections as well as possible.

The output of the First Association splits off into one of two paths. Matched Tracklets (Path-1): These are tracklets that have been successfully matched to high-confidence detections. After performing a detection, this results in the former tracklets (updated) being updated with their new position and detection data. The second path contains the remain detections, those high-confidence detections that do not match with any predicted tracklets. All the remaining detections are treated as new objects by the system, and tracklets (new) is initialized for them resulting in creating new tracks.

The next step is the Second Association module, where it takes care of all unmatched pre-

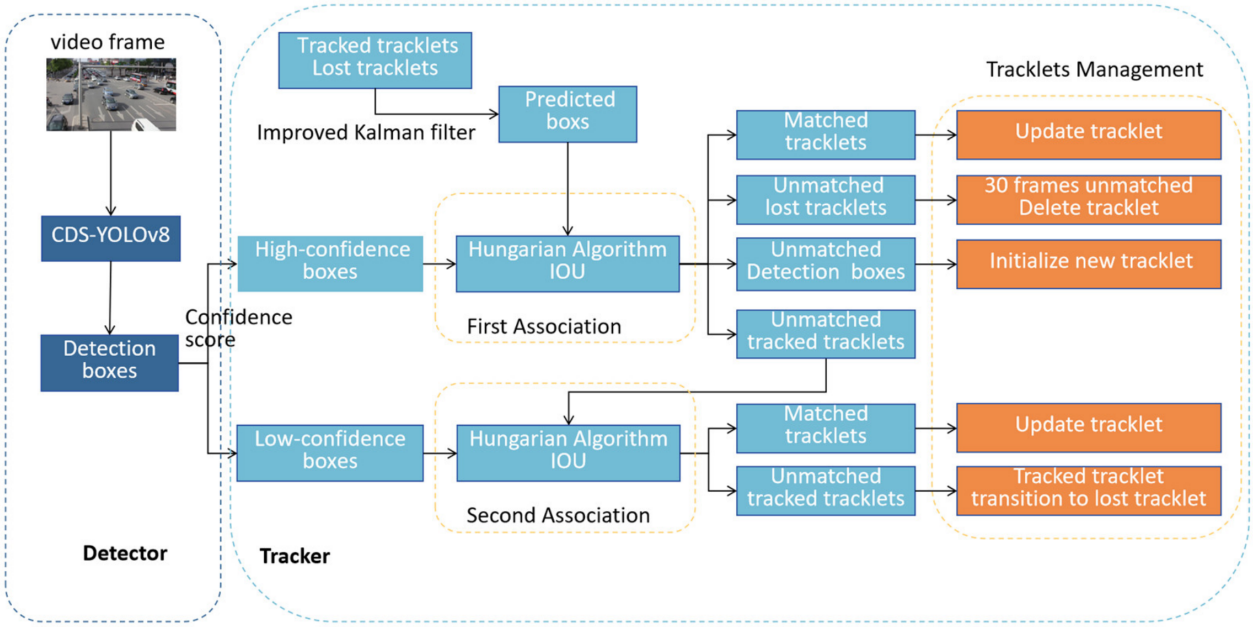

Figure 2.4: The ByteTrack pipeline applies CDS-YOLOv8 to obtain detection boxes with confidence scores. High-confidence boxes are first matched with predicted tracklets using IOU and the Hungarian algorithm. Unmatched tracks are then associated with low-confidence boxes in a second round. Tracklets are updated, initialized, or deleted based on the association results, improving robustness in crowded scenes.

dicted tracklets from the First Association. In this, the remain kallakts (unmatched predicted tracklets) are associated with low score detections. Detections with a lower score have confidence scores below the threshold defined but can still be valid objects. The Missed match Recovery is attempted by Second Association, where it associates these remaining tracklets with low-confidence detections.

The Second Association can also have two results. One is a matched tracklets that are previously unmatched tracklets associated with low-confidence detections. We then update those tracklets with the detections resulting in some tracklets (updated). The other is to the re remain tracklets, which are those tracklets that cannot find one matched in the second association stage. The system therefore also labels these tracklets as tracklets (lost).

2.2.4 Conclusion

The reviewed methods, SORT, DeepSORT, and ByteTrack, all show how the Kalman filter supports deep learning-based multi-object tracking. These methods use the filter to improve tracking continuity and accuracy, but each one also has some drawbacks when dealing with complex tracking scenarios, such as heavy occlusions.

SORT primarily uses the Kalman filter to predict object states between frames, which is helpful when detections are inconsistent due to occlusions or temporary failures of the detector. This predictive capability is key for real-time applications; without it, the system would lose track of an object as soon as it goes out of sight. However, SORT only makes simple motion predictions and does not incorporate appearance information from the data, making it prone to identity switches in occluded scenes or when objects look similar. This limitation can be problematic

in complex environments, where additional cues may be needed to track objects accurately.

DeepSORT addresses some limitations of SORT by adding appearance features along with predictions from the Kalman filter. In this case, the filter predicts object positions, and these predictions are corrected when new detections are available to improve the object's position and speed accuracy. Using appearance characteristics, DeepSORT can associate detections across frames more robustly, reducing identity confusion in crowded scenes or when objects look similar. However, DeepSORT still faces challenges in highly dynamic situations, such as when fast-moving objects pass behind an obstacle and briefly vanish or change direction, even with accurate appearance features.

ByteTrack takes a more sophisticated approach by using a two-phase association mechanism, associating high-score detections first, followed by low-score detections. This approach helps ByteTrack maintain more consistent associations in cases where objects may only be intermittently detected. Although the Kalman filter's predictive and corrective functions remain core to ByteTrack, this method has a drawback. By relying heavily on confidence-based detections instead of identity-based associations, ByteTrack can be biased towards high-confidence detections. In difficult scenarios involving persistent low-confidence detections or background clutter, ByteTrack's performance may degrade, as it cannot completely avoid identity switches or track breaks.

Each of these methods builds on the Kalman filter's predictive power and correction capabilities. Yet, each has limitations: SORT struggles without appearance information, DeepSORT encounters issues in highly dynamic scenes, and ByteTrack's two-phase approach still depends heavily on high-confidence detections, making it less robust in uncertain situations.

Based on these observations, this thesis aims to enhance the Global Tracking Transformer (GTR) model by using predictions generated with Kalman filtering. In real-world applications, detections often fail due to occlusions, motion blur, or other environmental factors, causing tracking continuity to be lost. The Kalman filter's predictive abilities provide a recovery mechanism, allowing tracking to continue even without accurate detections. The GTR model addresses continuity by predicting an object's likely position when detections are missing and correcting its movement once reliable detections are available. This research aims to reduce the limitations of detectors by integrating the Kalman filter into the GTR model.

This research also explores more efficient data association methods to improve GTR performance by better using the prediction information from the Kalman filter, resulting in a more robust and accurate GTR model. The improved GTR model aims to achieve continuous and accurate tracking in high-speed, complex scenes with heavy occlusions and appearance variations by leveraging the advantages and addressing the weaknesses of existing trackers like SORT, DeepSORT, and ByteTrack.

2.3 Transformer and Its application for object tracking

2.3.1 Transformer-Based Tracking Formulation

Transformer-based models have recently emerged as powerful architectures in visual tracking tasks due to their ability to capture long-range dependencies and contextual information. Originally introduced by Vaswani et al. in 2017[37], the Transformer was designed for sequence modeling tasks such as machine translation. Its core innovation lies in the use of a self-attention

mechanism, which allows the model to attend to all parts of the input sequence simultaneously, offering significant advantages over recurrent models in both accuracy and computational efficiency.

In the context of object tracking, the Transformer is adapted to model temporal and spatial dependencies across video frames. As illustrated in Figure 2.5, a standard Transformer-based tracking framework consists of an encoder-decoder architecture, where the encoder aggregates multi-frame information from input feature sequences, and the decoder predicts object trajectories by attending to both the temporal context and current observations. Specifically,

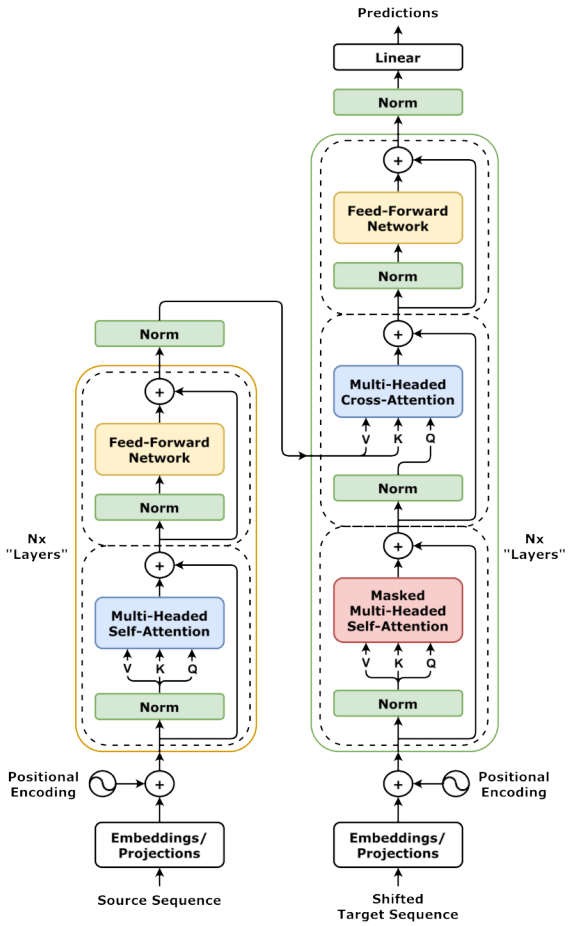

Figure 2.5: The Transformer architecture, adapted for tracking applications. The encoder encodes spatiotemporal features from input frames, while the decoder predicts object trajectories through cross-attention mechanisms[30].

the encoder receives a sequence of frame-wise visual features extracted by a backbone network. Through stacked layers of multi-head self-attention and feed-forward networks, it builds a rich representation of object movement and appearance over time. The decoder takes query embeddings (such as positional tokens representing object locations) and uses cross-attention mechanisms to align them with the encoder output, yielding refined predictions for object positions or identities.

One crucial advantage of using Transformers for tracking lies in their capacity to perform association implicitly, by directly learning query-to-object assignments across time without relying on explicit matching heuristics. This enables robust tracking even under challenging conditions like occlusion, appearance variation, or crowded scenes.

Moreover, positional encodings are added to the input features to preserve the temporal order of the frames, which is essential for modeling object trajectories. These encodings can be either fixed (e.g., sinusoidal) or learned during training, and they ensure that the model distinguishes between objects across different time steps.

Compared to traditional tracking pipelines that separate detection, association, and motion modeling, Transformer-based trackers offer an end-to-end paradigm. They unify detection and tracking tasks, enabling global optimization of identity assignment and bounding box regression in a single architecture.

2.3.2 Vision Transformer

In the field of computer vision, traditional convolutional neural networks (CNNs) have dominated image recognition tasks. However, with the success of Transformers in natural language processing, researchers aimed to adapt the Transformer architecture to improve image recognition performance. The primary purpose of migrating the standard Transformer to the Vision Transformer (ViT) is to leverage the Transformer's strength in capturing long-range dependencies and global information, addressing limitations in feature extraction commonly associated with CNNs.

In this thesis, the Vision Transformer (ViT)[46] applies the Transformer architecture to image recognition tasks, marking a significant shift from traditional convolutional neural networks (CNNs). As illustrated in Figure 2.6, ViT, introduced by Dosovitskiy et al. in 2020, adapts the Transformer model to process image patches as tokens, which are subsequently handled by a Transformer encoder.

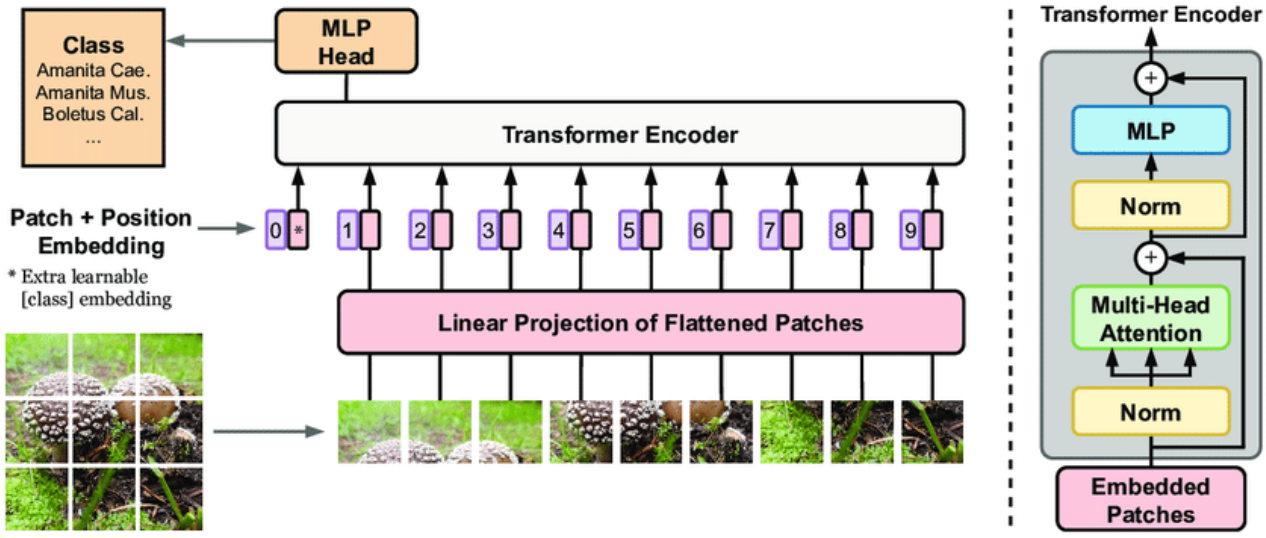

Figure 2.6: The Vision Transformer (ViT) architecture. The image is divided into patches, which are embedded and processed by the Transformer encoder using multi-headed self-attention. A class token is added for classification, and the final prediction is generated by an MLP head[47].

In the field of computer vision, traditional convolutional neural networks (CNNs) have dominated image recognition tasks. However, with the success of Transformers in natural language

processing, researchers aimed to adapt the Transformer architecture to improve image recognition performance. The primary purpose of migrating the standard Transformer to the Vision Transformer (ViT) is to leverage the Transformer's strength in capturing long-range dependencies and global information, addressing limitations in feature extraction commonly associated with CNNs.

In this thesis, the Vision Transformer (ViT)[46] applies the Transformer architecture to image recognition tasks, marking a significant shift from traditional convolutional neural networks (CNNs). As illustrated in Figure 2.6, ViT, introduced by Dosovitskiy et al. in 2020, adapts the Transformer model to process image patches as tokens, which are subsequently handled by a Transformer encoder.

The model retains the core elements of the Transformer encoder but adapts them for processing images rather than text sequences.

The first major component is the use of image patches as input tokens. An image is divided into fixed-size patches, for example, 16×16 pixels. These patches are flattened into vectors that serve as tokens for the Transformer. For instance, if the original image has a resolution of 224×224, dividing it into 16×16 patches results in 196 patches, calculated as:

Each patch is passed through a linear projection layer to convert it into a fixed-dimensional embedding vector. This process is analogous to the word embedding in natural language processing (NLP).

To retain the spatial structure of the image, positional encodings are added to the patch embeddings, much like in the original Transformer architecture for text. These positional encodings ensure that the model is aware of the relative positions of patches within the image.

The core of ViT is its Transformer encoder, which processes the sequence of patch embeddings. The encoder consists of multiple layers of multi-head self-attention and feed-forward networks, similar to the structure in the original Transformer model used for text. The self-attention mechanism enables the model to capture long-range dependencies across the image, as each patch can attend to all other patches in the image. This is a key difference from CNNs, where the receptive field is limited and primarily local.

In ViT, the Transformer architecture is adapted for image recognition by processing image patches as tokens. The queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$) are derived from these patch embeddings, and d_k is the dimensionality of the keys.

A special `[CLS]` token is prepended to the patch embeddings to aggregate information across all patches. During the self-attention process in the Transformer encoder, each patch embedding interacts with every other patch embedding and the `[CLS]` token. This enables the `[CLS]` token to accumulate a holistic representation of the image's features, effectively summarizing the global context of the image by integrating spatial information across all patches. The output of this token, representing the entire image, is then passed through a classification head to produce logits. These logits are processed by a softmax function, which converts them into probabilities over the image classes, enabling classification. This approach allows the `[CLS]` token to encapsulate the learned features of the image, leveraging the Vision Transformer's ability to capture long-range dependencies and spatial relationships across the image.

The differences between ViT and CNNs also bring some key advantages, especially when considering the global context by self-attention, using image feature vectors with access to all pixels gives a big picture of what is in the image. Moreover, given large datasets and large model

sizes, ViT beats CNNs by using simple multi-layer perceptrons (MLPs) instead of convolutions + scaled since they generalize quite nice in this settings.

Still, ViT has its downfalls; most importantly the data efficiency as ViT is very data-hungry due to absence of any kind of inductive biases from CNN domain (locality and translation invariance assumption are the common ones). Also, as its self-attention mechanism has complexity $O(n^2)$ in sequence length, it can be computationally expensive for high-resolution images.

The self-attention mechanism has recently been successful in NLP[48], and shown to be a potential way of replacing convolutions for computer vision, but does come with issues regarding data efficiency and computational demands in real-world scenarios.

2.3.3 Transformer for object tracking

2.3.4 TrackerFomrer

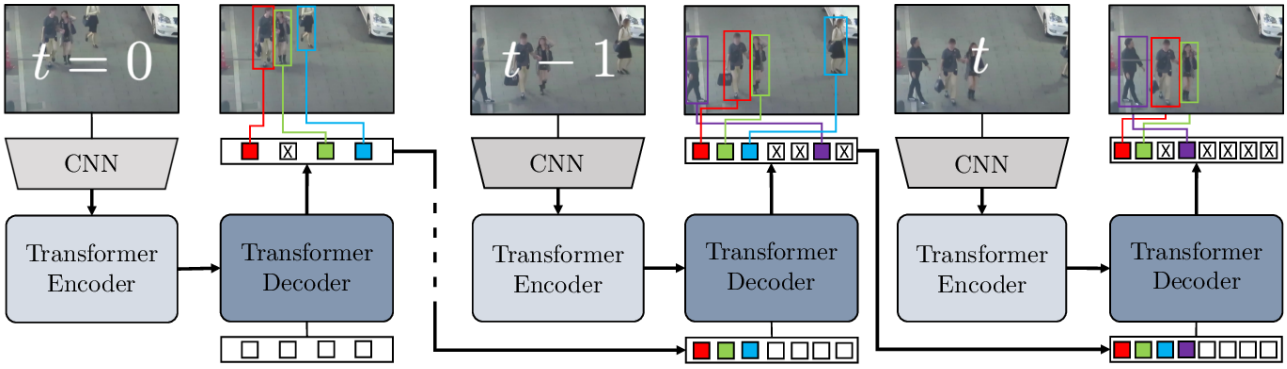

Figure 2.7: The TrackFormer process: each frame in the video sequence is processed by a CNN to extract features, which are then fed into a Transformer Encoder-Decoder architecture. The system uses query boxes (colored boxes) to match objects across frames, updating their identities and locations over time [49].

TrackFormer[50] is a multi-object tracking algorithm introduced to address the limitations of previous tracking methods by utilizing transformers. TrackFormer leverages the attention mechanism of transformers to improve tracking performance, particularly in handling occlusions and complex interactions between objects.

TrackFormer uses object detection results and encodes these detections into a sequence of feature embeddings. The core of TrackFormer is the transformer encoder-decoder architecture[37], which processes these embeddings to model the relationships between objects across frames. The transformer's self-attention mechanism allows TrackFormer to capture long-range dependencies and interactions between objects, which is crucial for maintaining accurate tracking in crowded and dynamic scenes, as illustrated in Figure 2.7.

The main steps of TrackFormer are as follows: First, an object detector like YOLO[51] or Faster R-CNN[46] provides bounding boxes for objects in each frame. These detections are then encoded into feature embeddings using a neural network. These embeddings are fed into the transformer encoder, which applies self-attention to model the relationships between all detected objects in a frame.

A transformer decoder then processes these encoded features and predicts the locations of objects for the following frame. This is based on both what the observers see now and the

historical paths of the objects. Model uses the attention mechanism of the decoder to make sure it focuses on relevant features present in previous frames enabling better continuityd accuracy for tracks.

Another step called data association[52] is also included in TrackFormer where it associates predicted positions with the found positions on next frame. This matching is based on a cost matrix balancing the spatial proximity and feature similarity of objects to ensure robust association in case of occlusions / missed detections.

These losses helped TrackFormer to achieve great improvements on multi-object tracking benchmarks such as MOT17. This allows Asfamal to capture complex interactions and dependencies amongst the objects resulting in more accurate predictions and stable tracking results especially in densely populated scenes with constant occlusions.

Although TrackFormer holds higher performance as compared to other methods, still there are few limitations of it.Transformer architecture is expensive to operate high computational complexity, particularly when real-time applications are required. Moreover, the TrackFormer performance deeply relies on object detections and feature embeddings.

2.3.5 TransTrack

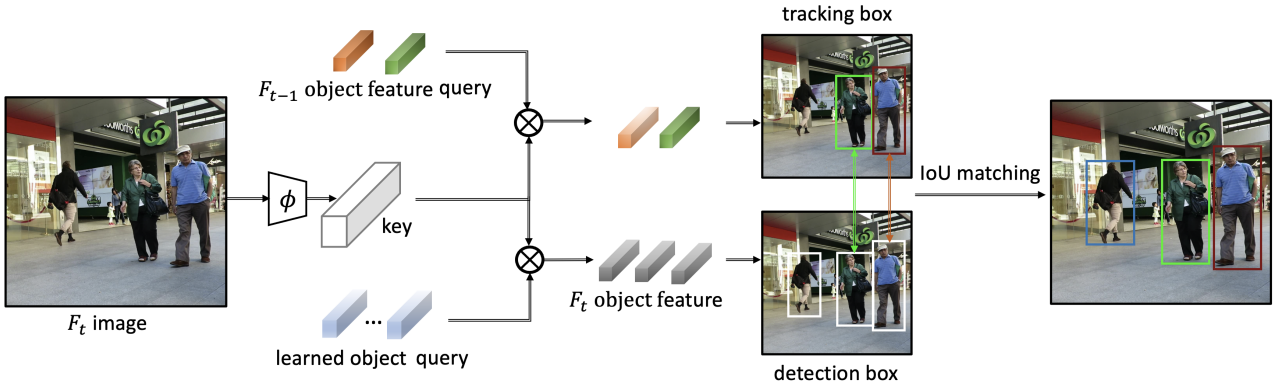

Figure 2.8: TransTrack process: frame F_t generates object features, which are matched to previous frame detections (F_{t-1}) via IoU, linking objects across frames [53].

TransTrack [35] is a multi-object tracking algorithm based on transformers that deals with occlusion and the association of object identity over time. TransTrack is an extension of conventional tracking methods that uses a transformer-based framework, improving associations between objects across frames, thus making it more robust to complex situations.

TransTrack combines object detection and tracking in a single end-to-end framework. By modeling spatial and temporal relations between objects using transformers, TransTrack can address the challenges involved with multi-object tracking better. In highly cluttered scenes, where depth due to occlusion is common, this single framework is beneficial as shown in Fig 2.8.

For every image frame F_t, initial detections are first extracted. These detections come from an object detector like DETR (DEtection TRansformers) [54], which are then encoded as feature embeddings via a convolutional neural network (CNN). These embeddings consist of object constructions from the present frame F_t as well as the earlier frame F_{t-1}. The features are fed to a feature encoding module (ϕ) which outputs a transformer key. This key is fed into the transformer to model frame associations together with learned object queries [36].

In the decoding phase of the transformer, the model retrieves information from both current and previous frames' latent features encoded. Via cross-attention, the transformer builds temporal relationships between frames and is able to merge object features from multiple time points with learned object queries. The combined outcome then yields tracking and detection boxes to output object position predictions in the respective boxes.

TransTrack utilizes IoU matching [55] in data association, associating tracking boxes to detection boxes—essentially linking predicted positions to those in the current frame. This algorithm is based on the components of spatial proximity and feature similarity and will reliably match objects over frames, providing strong tracking. The states for matched objects are updated, whereas unmatched detections initialize new tracking instances.

On benchmarks such as MOT17 [56], TransTrack has demonstrated considerable performance gains. It leverages transformers to handle occlusions and complicated interactions between multiple objects, leading to improved tracking precision by limiting identity switches over existing methods.

While TransTrack possesses these strengths, it also has limitations. The transformer-based architecture is resource-hungry and may hinder achieving real-time results, particularly in the case of low-end devices. TransTrack also relies on the initial detection quality and the quality of the feature embeddings. These dependencies make the results more susceptible to variation for low detection quality, indicating points in which further improvements may be considered to enhance transferability.

2.3.6 MOTR

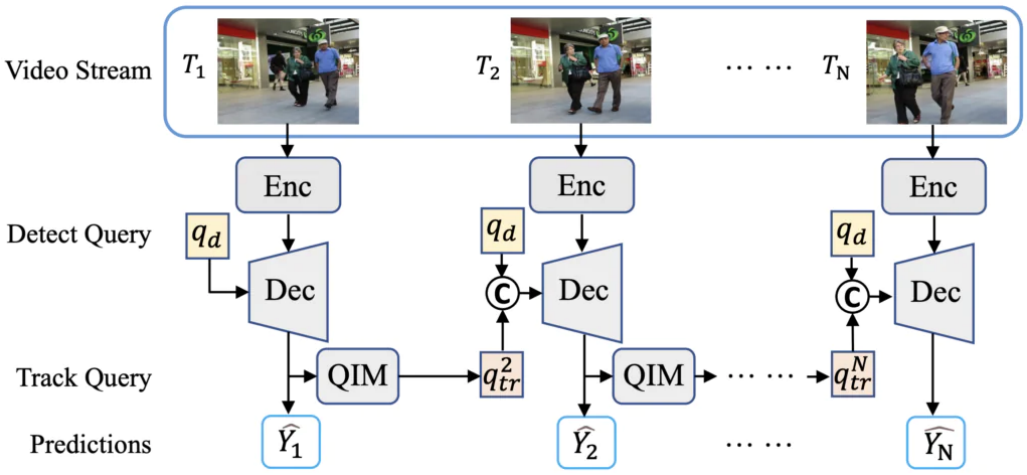

Figure 2.9: The MOTR process: each frame T_n in the video stream is processed through an encoder-decoder structure. Detect queries q_d and track queries q_{tr} are updated iteratively to maintain object tracking across frames, yielding predictions Y_n [53].

MOTR (End-to-end multiple object tracking with transformer) [36] is a cutting-edge algorithm designed to leverage the transformer architecture to address the complexities of multi-object tracking (MOT). MOTR integrates object detection and tracking into a single, unified framework that excels in scenarios with frequent occlusions and complex object interactions.

MOTR uses the powerful self-attention mechanism of transformers to model long-range dependencies and interactions between objects across frames. This approach enables more accurate

and robust tracking, particularly in crowded and dynamic environments. By processing sequences of frames together, MOTR can maintain consistent object identities and handle occlusions effectively, as illustrated in Figure 2.9.

The workflow of MOTR begins with processing a video stream consisting of frames $T_1, T_2, \ldots, T_N$. For each frame T_i, an encoder, denoted as Enc, extracts features from the frame, encoding spatial relationships within the current frame. MOTR utilizes two types of queries: the detection query q_d and the track query. The detection query q_d is used by the decoder, denoted as Dec, to identify objects in each frame. The decoder applies cross-attention to the encoded features, allowing the model to incorporate temporal dependencies from previous frames.

In each frame T_i, the predictions Y_i are generated based on the output of a Query Interaction Module (QIM). This module processes both the detection query q_d and the track query q_i^{tr}, where q_i^{tr} represents the track query specific to frame T_i. The QIM ensures continuity in object identities by associating current frame detections with predictions from previous frames, which helps in preserving object identities over time and reduces identity switches.

As the process advances to the next frame, the detection query q_d and the updated track query q_{i+1}^{tr} are fed into the decoder. This iterative process continues for each frame until the final frame T_N, where predictions Y_N are generated with object identities maintained consistently across the video.

MOTR uses a matching network for data association, so it is also quick to get matched based on spatial distance and feature-based descriptor similarity of detected objects in consecutive frames. Such strong matching helps to maintain identity preservation since there will be few identity switches, and thus better tracking quality.

MOTR has shown significant improvements within multi-object tracking benchmarks like MOT17. By utilizing a transformer-based architecture, it can model the relationships and dependencies between objects effectively which results in great tracking accuracy and also better occlusion handling than traditional approaches.

However, MOTR is also with some drawbacks. However, this transformer architecture is also computationally complex which can affect real-time performance on low power devices. Also, the performance of MOTR is highly dependent on the quality of initial object detections and feature embeddings.

2.3.7 Conclusion

TrackFormer, TransTrack, and MOTR are all advanced multi-object tracking (MOT) algorithms that leverage the transformer architecture to improve tracking performance in challenging scenarios, particularly those involving occlusions or requiring consistent object identities across long sequences. Although they all utilize transformers, each approach has its own design philosophy, strengths, and limitations, making them suitable for different applications.

TrackFormer was initially developed to capture strong intra-frame relationships through a set-based prediction approach. It relies heavily on high-performance external detectors, and its design is particularly effective in static or low-motion scenes with accurate detections. By leveraging associations within a single frame, TrackFormer improves detection quality in sparsely populated scenarios. However, it lacks deep temporal modeling, which limits its ability to maintain object identities across frames. This makes TrackFormer vulnerable to identity switches in dynamic scenes, where continuity over time is crucial.

TransTrack addresses this limitation by tightly coupling detection and tracking within the DEtection TRansformer (DETR) [54] framework. It models both spatial and temporal relationships through joint object-query embeddings and bipartite matching. This enables better identity preservation across frames and reduces association errors. Despite its improved robustness, the method incurs high computational overhead due to DETR's iterative attention and the cost of pairwise matching, making it less suitable for real-time or resource-constrained environments.

MOTR takes an even more integrated approach, unifying detection and tracking through deep spatio-temporal attention. It introduces a Query Interaction Module (QIM) to iteratively update detection and tracking queries across frames, allowing it to handle long-term occlusions and consistently associate objects even under severe clutter. MOTR achieves strong performance in terms of identity preservation and occlusion recovery. However, it is computationally expensive and highly dependent on the quality of initial detections; inaccurate first-frame detections may lead to cascading errors in subsequent frames.

From a data association perspective, the three methods differ significantly. TrackFormer employs a one-frame set matching strategy based on visual proximity and appearance similarity, which can result in identity switches for overlapping or similarly appearing objects. TransTrack uses global bipartite matching to improve association accuracy, albeit at the cost of computation. MOTR, by incorporating attention across time, builds on both spatial and temporal cues to maintain identities with high accuracy, especially under occlusion. Nevertheless, this comes with an even higher computational burden.

Despite their innovations, all three methods fundamentally operate in a local tracking paradigm, where frame-to-frame association dominates and global trajectory optimization is absent. This limits their generalizability to long videos or complex real-world environments that demand consistent, holistic object modeling.

2.4 GTR

2.4.1 Principles and Workflow

Since all the research in this thesis is implemented and developed based on GTR, it will be presented as a separate section. GTR (Global Tracking Transformers)[29] is a multi-object tracking (MOT) algorithm designed to leverage the powerful sequence modeling capabilities of transformers, specifically to address the limitations of traditional tracking methods. Unlike conventional methods that separate detection and tracking into distinct stages, GTR integrates these processes into a unified framework. This approach allows GTR to capture complex object interactions and dependencies over long periods, as illustrated in Figure 2.10.

GTR is fundamentally based on a transformer-style architecture modeling spatial and temporal relations of objects across frames. GTR uses the self-attention mechanism inside transformers to focus on key features and maintain the identity of each object over time, despite being difficult in dense crowds and frequent occlusions. With the use of self-attention, GTR is able to take into consideration important features and context that local information-based approaches tend to miss out on.

One significant feature in GTR is the introduction of the concept of a *trajectory query*. The trajectory query acts as the query (Q) in the attention mechanism, representing each object's

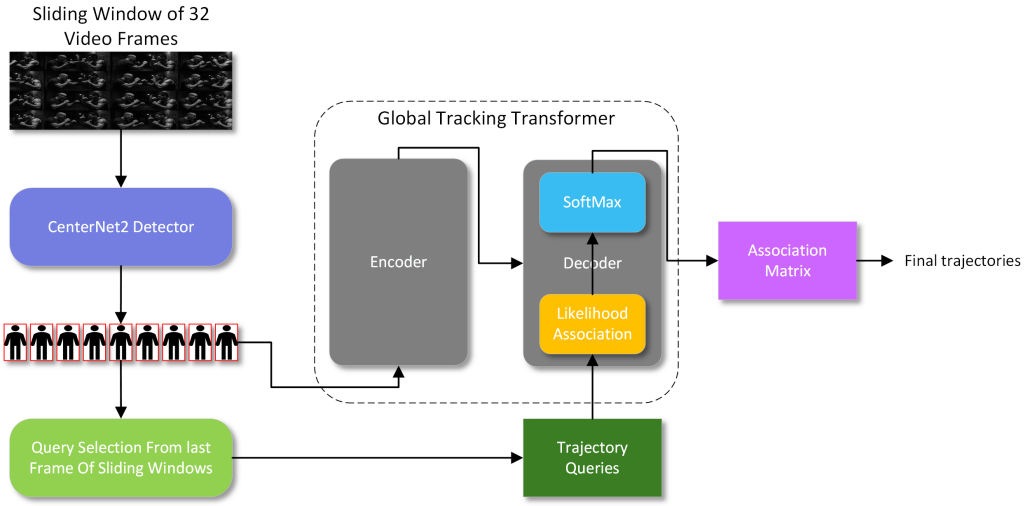

Figure 2.10: Overview of the Global Tracking Transformer (GTR) framework. The GTR model takes all-frame detections as input and employs trajectory queries, selected from a reference frame, to associate and track objects across time. The Transformer architecture consists of an encoder that processes global detection features and a decoder that performs cross-attention between trajectory queries and encoded features. A likelihood-based association followed by a SoftMax operation generates the association matrix, which is used to output consistent object trajectories throughout the video sequence [57].

trajectory across frames. This enables GTR to more effectively capture and maintain the trajectory of each object over time, providing a robust mechanism for identity retention even in the presence of frequent occlusions or when objects move in and out of the frame. By leveraging the trajectory query, GTR can accurately track object identities over extended periods, particularly in densely populated scenes where objects often interact or overlap.

GTR has a sequential workflow as described below. This starts with an object detector, usually DETR (DEtection TRansformers)[54], on every frame which provides bounding boxes for each identified object. DETR uses a transformer-based detection model which gives contextual embeddings for each detected object, which fits naturally with the tracking paradigm of GTR. These bounding boxes surrounding the object concerns provide position information and act as detections, or initializing locations for tracking tasks.

After detection, each bounding box is converted to feature embeddings with the help of a convolutional neural network (CNN). This CNN generates spatial features on every identified object, outputting them as high-dimensional vectors that serve as tokens for the transformer. These embeddings preserve important high-level properties, e.g., shape, texture, and spatial characteristics of the object, which enables GTR to keep accurate identities of objects through frames. Having a contextualized set of inputs before feeding it into a transformer is key to GTR's performance as this allows it to identify different objects even in the case where raw positional data would not suffice.

With token embeddings ready to go, they look into named feature embeddings that were passed through the train-data-embedding using the transformer encoder. The self-attention in a transformer encoder is applied over all the embeddings within a single frame, thereby capturing spatial relationships between objects. This would make the encoder pay attention to the interaction patterns given a frame, which mainly matters in higher object density scenarios

where objects often interact with each other. GTR has the prerogative of incorporating self-attention for dynamically weighing essentials of objects with respect to one another, thereby capturing subtle dependencies and preventing identity switches in dense environments.

The transformer decoder will therefore use these encoded features from the current frame and also combine them with features of previous frames. In the decoder, we use cross-attention to model inter-frame temporal dependencies and predict the position of each object in the next frame using class type annotation. Hence, GTR can retain the object id for temporal frames, no matter whether an object is occluded or out of sight from time to time. The spatial-temporal information is then combined using a cross-attention mechanism, allowing GTR to consider both the current spatial configuration as well as previous trajectories of objects for prediction.

GTR employs a bipartite matching algorithm for data association, facilitating accurate associations of objects between frames. To mitigate the possibility of identity switches, our matching algorithm takes into account spatial distance in addition to feature similarity. GTR matches objects based on identifiers like color, size, and shape, allowing it to keep track of the same object over time (tracking only one familiar object using multiple features), reducing false positives in complex environments. This means unmatched detections will start new tracks, and if there are no matches for a track over a certain number of frames, it is removed from the tracker to conserve resources.

2.4.2 Strengths of GTR

As such, GTR achieves considerable improvements on tracking benchmarks like MOT17, where it shines in scenarios with high occlusions and aggressive object interactions. GTR does the majority of magic through its transformer-based architecture, which helps it model long-term dependencies spanning frames, as well as catch complex spatial interactions within one frame. GTR takes advantage of self-attention to obtain the most relevant features on each frame and retain accurate and consistent identities of the objects, even when there are many occlusions or a lot of densely populated scenes. This ability to attend on the relevant features for each frame also lowers GTR's reliance on error-free initial detections, thereby increasing its robustness against detection errors.

Furthermore, GTR has a relatively robustness data association process for identity preservation. GTR alleviates the issue of identity switches by correctly associating objects across frames through matching based on spatial proximity and feature similarity. Such robustness is critical for applications where continuous and precise tracking of multiple objects over long periods of time is required such as autonomous driving or surveillance.

2.4.3 Limitations of GTR

While GTR is powerful, it is also not as flexible and effective in tracking tasks. A major drawback is the dependency of GTR on the quality at which it was initially detected. As GTR relies on the object detector to have correct bounding boxes in every frame, any detection error, e.g. due to occlusion, motion blur or adverse illumination conditions will profoundly break the tracking flow. GTR does not have an internal recovery for missed detections, meaning that a failure to detect the object can lead to identity switches or complete loss of track continuity.

Another disadvantage is querying mechanism of GTR, where it can only use the last frame in a sliding window as query frame. Such a single-frame stance is less flexible for dynamic scenes,

where objects may move in unexpected or non-linear ways. The reliance on features solely from the most recent frame can reduce tracking accuracy due to GTR's inability to adjust effectively when objects are moving quickly, particularly in fast-moving scenarios [58].

Also, GTR does not have channel or spatial attention mechanism in its feature extraction pipeline. Such mechanisms are widely employed to adjust feature representation by emphasizing important regions or channels, enabling GTR to learn more subtle features in challenging tracking scenarios. Lacking these attention mechanisms, GTR struggles with complex spatial details, which means it may not be robust in multi-object tracking scenarios where intricate feature extraction is necessary for effective tracking [59].

This shortcoming indicates directions for improvement, such as adding detection failure handling mechanisms, increasing flexibility to dynamic scenes, and including another attention module to improve feature accuracy. Optimizing such areas mentioned above can make GTR more adaptive to various and difficult tracking scenarios.

2.5 Summary and Link to Proposed Methods

Despite the progress made in both Kalman filter-based and Transformer-based tracking approaches, several challenges remain unresolved. Kalman-based methods, while efficient and simple, struggle in complex environments with frequent occlusions or abrupt motion. Transformer-based approaches, such as MOTR and GTR, improve association accuracy but still suffer from limited prediction ability and reliance on single-frame or single-query input.

To address these issues, this thesis proposes two novel methods. The first method, GPAT, augments the global tracking pipeline with a Kalman predictor and a CBAM module to better handle missed detections and enhance spatial feature learning. The second method, MQT-Former, introduces a multi-query mechanism to improve robustness and temporal consistency. These proposed solutions aim to tackle the identified weaknesses and are detailed in the following chapters.

Chapter 3

Enhancing GTR with Additional Prediction Module and CBAM

3.1 Introduction

In this chapter, we present GPAT (Global Prediction Attention-enhanced Transformer), as illustrated in Figure 3.1. GPAT is designed to process video sequences for multi-object tracking through a workflow that includes object detection, prediction of missed targets, attention-based feature enhancement, and robust association across frames.

To address detection failures caused by occlusion, fast motion, or illumination changes, we introduce a prediction module that estimates object positions based on historical trajectories. This preserves track continuity and improves robustness when detections are missing.

In parallel, we integrate the Convolutional Block Attention Module (CBAM) to refine feature representation. CBAM applies both channel attention—to emphasize discriminative features—and spatial attention—to localize targets more precisely by suppressing irrelevant background. This dual attention enhances tracking accuracy and stability, particularly in cluttered scenes.

Together, the prediction and CBAM modules improve tracking consistency and accuracy under real-world conditions. The following sections and figures detail the implementation and effectiveness of these components in improving GPAT's performance.

3.2 Method

3.2.1 Overview of Network Architecture

In the GPAT (Global Prediction Attention-enhanced Transformer) framework, each component is designed to collaboratively process video input, generate detection and prediction outputs, encode enhanced features, and associate object trajectories across time for robust multi-object tracking.

The pipeline begins with a *Video Sequence* input, which is fed into the *YOLOv8 Detector*. This module extracts object-level information from each frame, including bounding box coordinates,

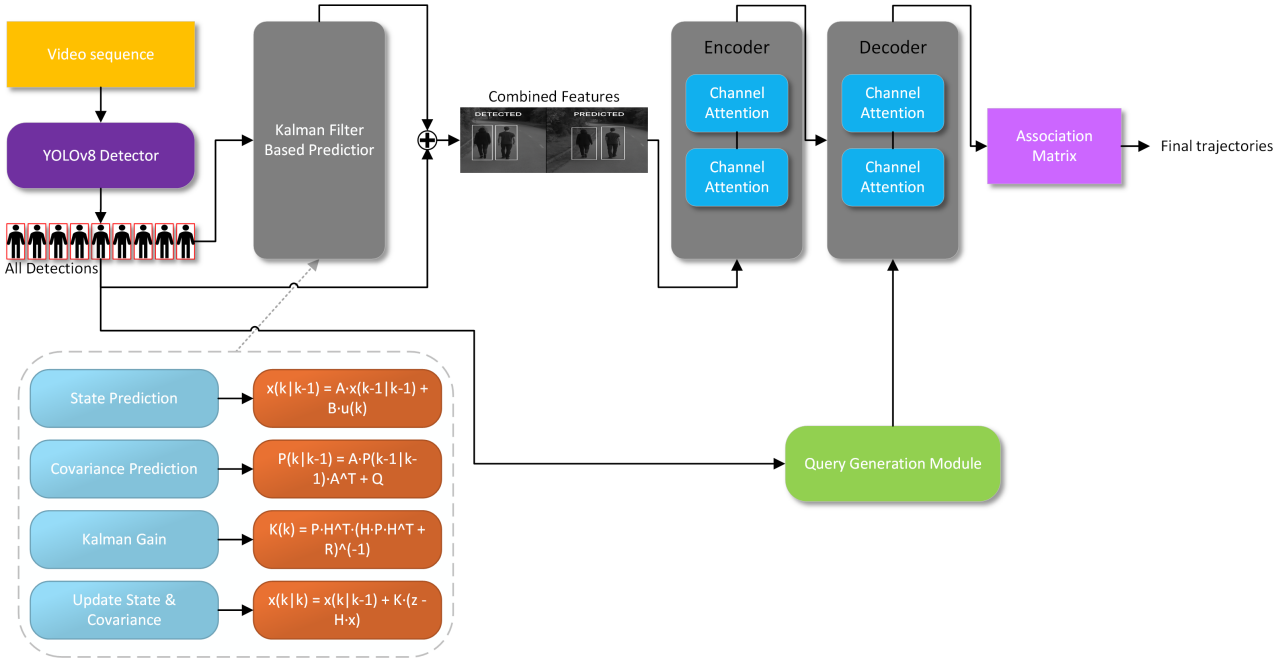

Figure 3.1: The overall architecture of GPAT (Global Prediction Attention-enhanced Transformer). The framework integrates YOLOv8 for object detection, a Kalman Filter-Based Predictor for handling detection failures, a CBAM-Enhanced Transformer for robust spatiotemporal modeling, and a query generation mechanism to produce accurate and consistent trajectory associations.

category labels, and detection confidence scores. These outputs are stored as *All Detections*, serving as the primary input for subsequent tracking operations.

To address temporary detection failures caused by occlusion, motion blur, or detector uncertainty, a *Kalman Filter-Based Predictor* is introduced. It estimates the current locations of tracked objects based on prior trajectory states using a four-step recursive process: *State Prediction*, *Covariance Prediction*, *Kalman Gain Computation*, and *State Update*. When a detection is found, the predictor performs a confidence-based update. When detections are missing, it directly outputs predicted positions, which are stored as *All Predictions*.

A *Combined Features* module then fuses information from both detection and prediction branches. This merged representation ensures continuity of object information flow, enabling the system to maintain trajectory consistency even under partial detection loss.

The fused feature map is passed into the *CBAM-Enhanced Transformer*, which consists of encoder and decoder blocks. Each block is augmented with *Channel Attention* and *Spatial Attention* modules (i.e., CBAM) to enhance discriminative features and suppress irrelevant noise. The encoder focuses on learning contextual relations between all objects, while the decoder operates on *Trajectory Queries* generated from a *Query Generation Module* that selects frames from the tail of a sliding window.

After passing through decoder layers, the refined features are fed into an *Association Matrix* module that constructs pairwise similarity relationships between current observations and past trajectories. This results in the final object assignments across frames, outputting *Final Trajectories* with high temporal consistency.

GPAT's design enables strong performance in complex environments by integrating prediction-

driven resilience with attention-enhanced global modeling.

3.2.2 Kalman filter based Predictor

The integration of the Kalman filter into the Global Predictor with Attention and Tracking (GPAT) framework, as depicted in Figure 3.2, significantly enhances the robustness and reliability of the tracking process. This modification is particularly effective in scenarios where the YOLOv8 detector may fail due to occlusions, motion blur, or other challenging conditions. By incorporating the Kalman filter, GPAT maintains continuous and accurate tracking of objects, even in the face of detection failures.

The tracking process within GPAT begins with an initial set of detections labeled as "All Detections," which are processed by the Kalman filter-based predictor. The predictor determines whether detections are available. When detections are found, a selection step is performed based on confidence scores, followed by a Kalman update. This update utilizes the detected object positions to correct the state estimates, improving accuracy by integrating the observed measurements with the Kalman filter's predictions.

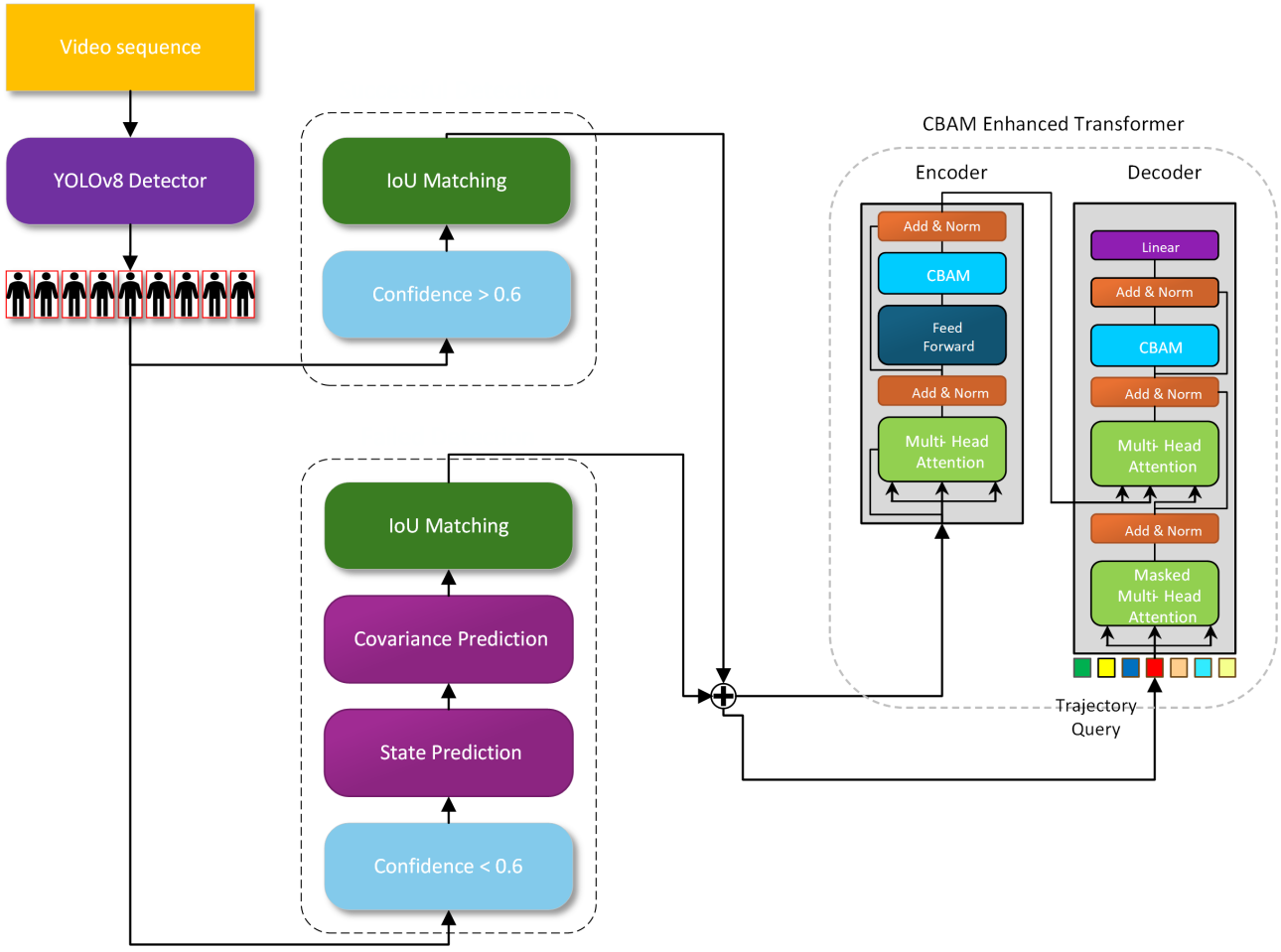

Figure 3.2: Main loop of the Kalman filter-based predictor in GPAT. High-confidence detections (confidence > 0.6) are associated via IoU matching and updated through Kalman filter correction. In the absence of confident detections, the predictor performs state and covariance prediction to estimate object positions. Both updated and predicted results are subsequently forwarded to the CBAM Enhanced Transformer for global feature integration.

Mathematically, when detections are available, the Kalman update equations are:

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}^\top \left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R}\right)^{-1}, \\ \mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{H}\mathbf{x}_{k|k-1}\right), \\ \mathbf{P}_{k|k} &= \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_{k|k-1}. \end{aligned} \tag{3.1}$$

In these equations, $\mathbf{x}_{k|k}$ represents the updated state estimate at time k, incorporating the current detection $\mathbf{z}_k$. The state vector $\mathbf{x}$ typically includes the object's position and velocity in the image plane. The matrix $\mathbf{P}_{k|k}$ is the updated error covariance, reflecting the uncertainty in the state estimate after considering the new measurement. The Kalman gain $\mathbf{K}_k$ balances the influence of the prediction and the measurement. The observation matrix $\mathbf{H}$ relates the internal state variables to the observed positions from the detector, and $\mathbf{R}$ is the measurement noise covariance.

From a computer vision perspective, these updates adjust the predicted positions and velocities of tracked objects based on new visual detections, effectively correcting deviations caused by model inaccuracies or unexpected object movements. This ensures that tracking remains accurate even when the object's motion deviates from the assumed model.

In situations where detections are missing—such as when YOLOv8 fails due to occlusion or motion blur—the Kalman filter's prediction mechanism becomes essential. The predictor generates an estimated position based on the object's last known state and the state transition model, predicting future positions in the absence of new observations. The prediction equations are:

$$\begin{aligned} \mathbf{x}_{k|k-1} &= \mathbf{F}\mathbf{x}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k, \\ \mathbf{P}_{k|k-1} &= \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\top + \mathbf{Q}. \end{aligned} \tag{3.2}$$

Here, $\mathbf{x}_{k|k-1}$ is the predicted state estimate at time k, based on the previous estimate $\mathbf{x}_{k-1|k-1}$. The state transition matrix $\mathbf{F}$ models the object's motion (e.g., constant velocity), and $\mathbf{Q}$ is the process noise covariance accounting for motion uncertainties. In many vision applications, the control input $\mathbf{B}\mathbf{u}_k$ is zero.

In computer vision, these predictions enable the system to continue estimating object positions even when they are not detected in the current frame, maintaining trajectory continuity. The error covariance matrix $\mathbf{P}_{k|k-1}$ is updated to reflect increased uncertainty due to the lack of new detections, ensuring that the system properly weighs predictions against new measurements when detections resume.

The Kalman filter's prediction mechanism addresses a core challenge in multi-object tracking: maintaining consistent object trajectories despite detection failures[60]. By integrating the Kalman filter, GPAT mitigates risks like broken trajectories or identity switches, providing a mathematical framework for estimating positions based on prior motion models and observed data.

3.2.3 CBAM Enhanced Transformer

The CBAM (Convolutional Block Attention Module) as shown in Figure 3.3 is a critical component that enhances feature representation by applying attention mechanisms across both the channel and spatial dimensions. This module helps the model focus on the most informative

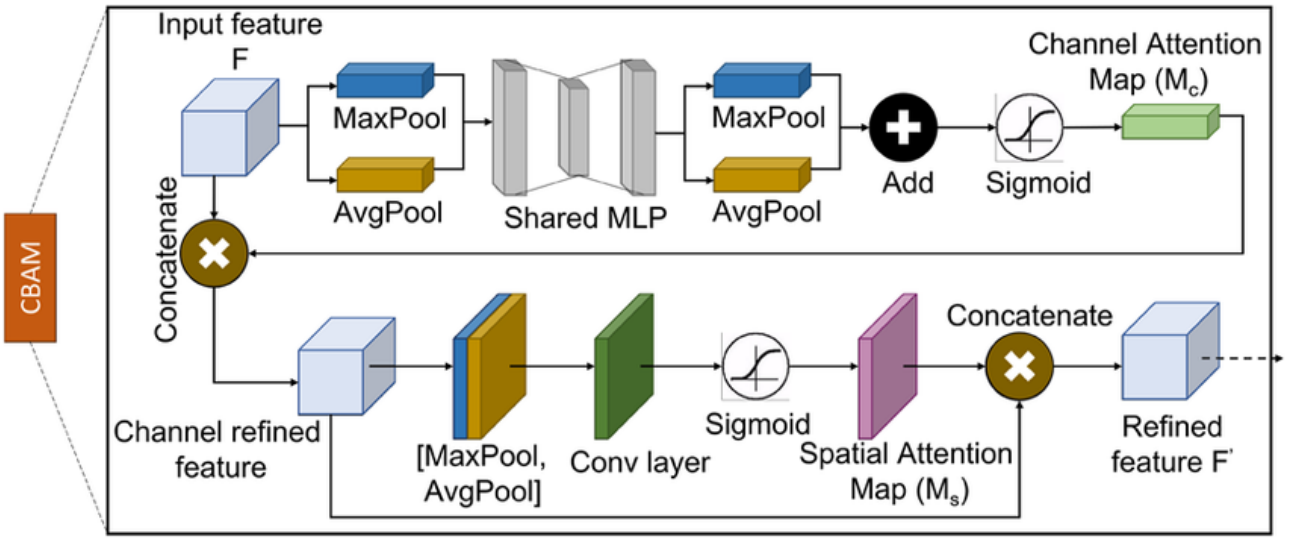

Figure 3.3: The structure of the Convolutional Block Attention Module (CBAM), which sequentially applies channel and spatial attention. The Channel Attention Module utilizes global max and average pooling followed by a shared MLP to generate channel attention maps. The Spatial Attention Module then refines the output by applying pooling and a convolutional layer to generate spatial attention maps. The input feature is adaptively refined in both dimensions to enhance representational capability.

parts of the input data, significantly improving performance in tasks such as object detection and tracking. Below is a detailed breakdown of the CBAM's internal mechanism.

The first part of CBAM is the Channel Attention Module, which emphasizes important feature channels while suppressing less relevant ones.

First, the input feature map F is processed through two types of pooling operations: Max Pooling and Average Pooling, across the spatial dimensions. These operations generate two different channel descriptors. Max Pooling captures the most salient features, while Average Pooling provides an overall summary of the features across the spatial dimensions:

$$F_{max} = \text{MaxPool}(F) \tag{3.3}$$

$$F_{avg} = \text{AvgPool}(F) \tag{3.4}$$

Next, the outputs from the Max Pooling and Average Pooling operations are passed through a shared Multilayer Perceptron (MLP), which consists of a few fully connected layers. The MLP transforms these pooled features into two separate channel attention maps:

$$M_{max} = \sigma(\text{MLP}(F_{max})) \tag{3.5}$$

$$M_{avg} = \sigma(\text{MLP}(F_{avg})) \tag{3.6}$$

where σ represents the Sigmoid activation function.

The two attention maps are then combined by element-wise summation to form the final Channel Attention Mask M_{ch}:

$$M_{ch} = \sigma(\text{MLP}(F_{max}) + \text{MLP}(F_{avg})) \tag{3.7}$$

Finally, the refined feature map is obtained by applying the Channel Attention Mask:

$$F_{ch} = M_{ch} \times F \tag{3.8}$$

Following the Channel Attention Module, the Spatial Attention Module further enhances the feature map by focusing on important spatial locations.

The output from the Channel Attention Module F_{ch} serves as the input to the Spatial Attention Module. Similar to the Channel Attention Module, the Spatial Attention Module first applies Max Pooling and Average Pooling, but this time across the channel dimensions, generating two spatial attention maps:

$$F_{ch\_max} = \text{MaxPool}(F_{ch}) \tag{3.9}$$

$$F_{ch\_avg} = \text{AvgPool}(F_{ch}) \tag{3.10}$$

These pooled maps are concatenated along the channel dimension and passed through a convolutional layer, which combines the spatial attention information into a unified attention map:

$$F_{sp} = \sigma(\text{Conv2D}([F_{ch\_max}; F_{ch\_avg}])) \tag{3.11}$$

where σ is the Sigmoid activation function, and the result is the Spatial Attention Mask M_{sp}.

The Spatial Attention Mask is then applied to the channel-refined feature map F_{ch}, yielding the final output of the CBAM F_{cbam}:

$$F_{cbam} = M_{sp} \times F_{ch} \tag{3.12}$$

Thus, the CBAM modules integrated with the transformer architecture which improvesthe feature extraction and processing of features before sent to Encoder and Decoder. CBAM enables the Transformer to attend for what (i.e, the most informative feature channels) and where (i.e., the most useful spatial regions) to highlight by applying channel-wise and spatial-wise attention in a sequential manner.

CBAM enhances aggregated features obtained from detector, predictor or other preprocessing modules in the Encoder. This triggers the High-Level Narrowing of Features and it's consequent layer to operate on the already narrowed features making sure that it is always a step ahead(in terms of highlighting) in cririciazing only the important information thereby creating an overall better representation.

CBAM improves the features matched by trajectory queries in Decoder. Since CBAM adds attention only over the relevant channels and spatial locations, it helps the Transformer to

make more reliable associations between trajectory queries and encoded features by generating more accurate similarity vectors with more precise decisions during tracking.

Thus, the effect of CBAM incorporated into Transformer is more pronounced in those tasks that emphasis fine-grained attention, like object detection and tracking. CBAM addresses this issue by focusing on important regions of the input data, improving the Transformer's overall accuracy and robustness in challenging situations involving multiple objects and occlusions while keeping the network architecture unchanged.

3.3 Experiments

In this section, the implementation and details of the experiments will be discussed. The datasets used will be provided, and the evaluation metrics, along with the calculation methods, will be described in detail. The design of ablation studies will also be thoroughly explained.

3.3.1 Experiment Setup

To exploit the distinct advantages of different platforms, all experiments were conducted on two separate systems: a high-end personal computer and the iHPC server at the University of Technology Sydney (UTS). The personal computer is equipped with an Intel Core i5-13600F CPU (6 performance cores and 12 threads, up to 4.4 GHz) and an Inno3D GeForce RTX 4080 GPU, which is well-suited for GPU-intensive workloads such as deep learning inference and testing.

The iHPC server, powered by NVIDIA A100 and H100 GPUs designed for large-scale machine learning and high-performance computing tasks, was primarily used during the training phase. These powerful resources allowed for significantly accelerated training time and large-batch experiments on extensive datasets. In contrast, the local PC was leveraged for the inference stage due to its flexibility and ability to support real-time testing and deployment scenarios.

The model was trained for 150 epochs using the Adam optimizer with a base learning rate of 2e-4, weight decay of 1e-4, and a cosine learning rate scheduler with warm-up during the first 5 epochs. A batch size of 16 was used on the A100 GPU. The loss function followed the Hungarian matching-based approach, incorporating Focal Loss for classification and L1 loss + GIoU loss for bounding box regression. To enhance model generalization, data augmentations including random horizontal flipping and multi-scale resizing were applied. The training pipeline was adapted from the official GTR repository with minor changes to accommodate our proposed prediction and attention modules.

On the system level, a Linux operating system was employed for its stability and efficient resource management. The personal computer also runs a dual-system configuration with Linux operating in a virtual machine, allowing convenient switching between environments to accommodate multi-tasking and experimental flexibility.

In terms of software, PyCharm was used as the primary development environment due to its strong Python support and advanced debugging features. The experiments were conducted using PyTorch, which supports dynamic computational graphs and is particularly well-suited for complex model architectures. The PyTorch environment was configured with CUDA 11.7 and cuDNN to ensure full GPU acceleration. Anaconda was used for package and environment management, allowing dependencies to be cleanly isolated across experiments.

3.3.2 Experiment Data

In this experiment, we utilized two publicly available datasets: MOT17 and TAO. Below are the specific details of these datasets:

The MOT17 (Multiple Object Tracking 2017) dataset is a benchmark widely used in multi-object tracking research. It includes 14 video sequences divided into training and testing sets. Each video provides meticulous annotations, such as pedestrian detection boxes and corresponding trajectory IDs. The scenes encompass a variety of complex real-world environments like city streets, shopping malls, and subway stations. The scenarios vary by camera angles, lighting conditions, and object densities. MOT17 offers outputs from three different detectors (DPM, FRCNN, SDP), allowing for varied evaluations of tracking algorithm performance. Standard evaluation metrics such as MOTA (Multiple Object Tracking Accuracy) and IDF1 scores are included. A sample frame is shown in Figure 3.4, illustrating dense urban pedestrian trajectories and consistent identity annotations.

The TAO (Tracking Any Object) dataset is designed to advance research in general object tracking, accommodating a wide range of object types. It comprises 2,907 video segments from diverse sources such as COCO, LVIS, and YFCC100M [61]. This dataset is particularly challenging due to its coverage of 80 different object categories, including people, animals, vehicles, furniture, and tools. Each video segment in TAO is annotated with the object's category, bounding box, and trajectory throughout the sequence. The dataset emphasizes not only the physical location of objects but also the diversity and sparsity of object categories, enhancing its utility for comprehensive tracking applications. Figure 3.5 presents a sample frame from the TAO dataset, showcasing a variety of object categories in complex scenes.

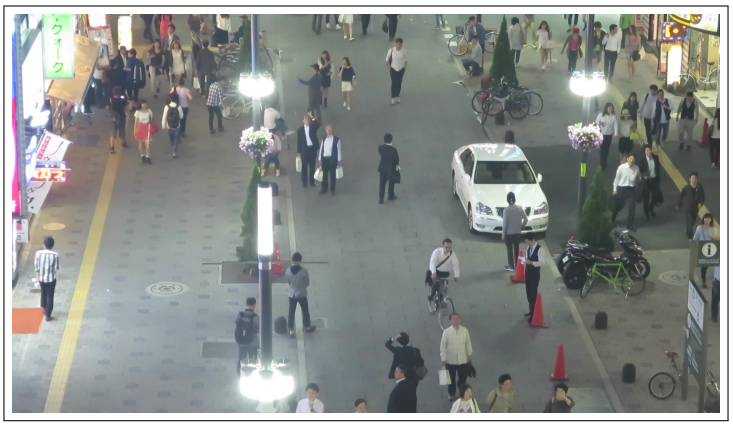

Figure 3.4: Sample frame from the MOT17 dataset, illustrating pedestrian trajectories in urban scenes with dense object distributions and identity annotations.

3.3.3 Evaluation Metrics

MOTA (Multiple Object Tracking Accuracy) is one of the most comprehensive metrics for evaluating tracking performance. It includes three types of errors: False Positives (FP), where detections are mistakenly added; False Negatives (FN), where objects are missed; and Identity Switches (IDSW), where the assigned identity of an object changes. MOTA is defined as:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FP}_t + \text{FN}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t} \tag{3.13}$$

Figure 3.5: Sample frame from the TAO dataset, showing diverse object categories and annotation sparsity across domains such as animals, vehicles, and tools.

where GT_t represents the total ground truths at time t.

IDF1 (ID F1 Score) focuses on the accuracy of identity preservation. It is the harmonic mean of precision and recall for identified detections, calculated as:

$$\mathrm{IDF1} = \frac{2 \cdot \mathrm{IDTP}}{2 \cdot \mathrm{IDTP} + \mathrm{IDFP} + \mathrm{IDFN}} \tag{3.14}$$

Here, IDTP, IDFP, and IDFN represent True Positives, False Positives, and False Negatives for identity, respectively.

DetA (Detection Accuracy) assesses the accuracy of object detection within the tracking context:

$$\mathrm{DetA} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives} + \text{False Positives}} \tag{3.15}$$

AssA (Association Accuracy) evaluates how well a tracking system associates detected objects across frames:

$$\mathrm{AssA} = \frac{\text{Correct Associations}}{\text{Total Possible Associations}} \tag{3.16}$$

HOTA (Higher Order Tracking Accuracy) provides a balanced measure of tracking performance by combining detection and association accuracy:

$$\mathrm{HOTA} = \sqrt{\mathrm{DetA} \times \mathrm{AssA}} \tag{3.17}$$

FPS (Frames Per Second) indicates the processing speed of the tracking system, showing how many frames can be processed per second. This metric is crucial for real-time applications where processing speed is a key factor.

3.3.4 Experimental Results

As shown in Table 3.1, the performance of the GPAT method on the MOT17 dataset is compared with several early than global tracking methods, including LPT[62] and MOT-GM[58]. These earlier methods, despite offering certain performance advantages in their respective development periods, are limited in their effectiveness on the more complex MOT17 dataset. GPAT

demonstrates substantial superiority across multiple key metrics, particularly in MOTA, IDF1, HOTA, DetA, and AssA, which collectively reflect the comprehensive capabilities of tracking algorithms.

Table 3.1: Comparison to the GLOBAL tracking methods earlier than GTR on the MOT17 test set. The metrics include MOTA, IDF1, HOTA, DetA, and AssA.

Method	**MOTA**	**IDF1**	**HOTA**	**DetA**	**AssA**
LPT[62]	57.4	58.7	—	—	—
MOT-GM[58]	43.2	51.5	—	—	—
GPAT (ours)	**77.1**	**73.9**	**59.9**	**60.4**	**59.8**

As shown in Table 3.2, the GPAT method is compared with several traditional multi-object tracking methods on the MOT17 dataset, covering key performance metrics such as MOTA, IDF1, HOTA, DetA, and AssA. These comparisons highlight GPAT's overall performance advantages while also acknowledging areas where certain existing methods perform better. This discussion aims to provide a more comprehensive evaluation and respond to the reviewer's concern about cases where other methods outperform GPAT.

First, in terms of MOTA (Multiple Object Tracking Accuracy), GPAT achieves a score of 77.1. While this score is slightly lower than ByteTrack [45] (80.3) and BoT-SORT [63] (80.6), GPAT still significantly outperforms earlier methods like Tracktor++ [64] (53.5), CenterTrack [65] (67.8), and GSDT [66] (66.2). MOTA reflects the balance between false positives, false negatives, and ID switches. ByteTrack and BoT-SORT benefit from optimized association strategies and enhanced heuristics that can reduce tracking error in cleaner or less crowded scenes. In contrast, GPAT prioritizes robustness through feature-guided prediction and attention-enhanced association, making it better suited for more dynamic or occlusion-heavy environments, even if it slightly sacrifices MOTA in optimal conditions.

For IDF1 (Identity F1 Score), GPAT obtains 73.9, which is slightly lower than ByteTrack [45] (77.3) and BoT-SORT [63] (79.5), but remains competitive. Notably, GPAT outperforms CorrTracker [67] (73.6) and FairMOT [68] (72.3). IDF1 is critical for maintaining identity consistency across frames. ByteTrack's frame-to-frame heuristic linking often excels in maintaining identities in simple contexts, but its reliance on immediate-frame matching can break down under longer-term occlusions. GPAT's multi-level attention and prediction strategies allow for better identity preservation in crowded or long-term scenarios, even if not achieving the absolute highest IDF1 on all sequences.

Regarding HOTA (Higher Order Tracking Accuracy), GPAT reaches 59.9. Although this is slightly below BoT-SORT (64.6) and OC-SORT (63.2), GPAT maintains a strong balance between detection and association accuracy. OC-SORT and BoT-SORT are designed with lightweight, fast association in mind and can sometimes achieve higher HOTA through aggressive matching strategies. However, these methods may falter in maintaining robustness under challenging occlusions or abrupt motion. GPAT offers more stable high-order tracking accuracy by integrating trajectory prediction and CBAM-enhanced Transformer encoding.

On DetA and AssA, GPAT achieves 60.4 and 59.8 respectively. While methods such as MAATrack [69] and ByteTrack [45] may slightly outperform GPAT on one of these metrics, they often trade off balance—excelling in detection or association, but not both. GPAT's strength

lies in maintaining high accuracy across both aspects, demonstrating generalization across diverse scenes without depending on heavy post-processing.

In terms of computational efficiency, the proposed GPAT method achieves an average inference speed of approximately **32 FPS** on an RTX 4080 GPU. While slightly slower than lightweight methods such as **OC-SORT (42 FPS)** and **ByteTrack (38 FPS)**, it significantly outperforms Transformer-based global trackers like **TrackFormer (15 FPS)** and **GTR (18 FPS)**. This demonstrates that GPAT is capable of real-time performance while maintaining superior accuracy, validating its practical deployment value.

In summary, while GPAT may not always achieve the highest value on every individual metric, it offers a consistently strong and balanced performance. Combined with competitive runtime efficiency, GPAT proves to be a robust and practical solution for real-world multi-object tracking challenges.

Table 3.2: Comparison to the traditional methods on MOT17 dataset

Method	**MOTA**	**IDF1**	**HOTA**	**DetA**	**AssA**
Tracktor++[64]	53.5	52.3	44.8	44.9	45.1
CenterTrack[65]	67.8	64.7	52.2	53.8	51.0
TraDeS[70]	69.1	63.9	52.7	55.2	50.8
QuasiDense[71]	68.7	66.3	53.9	55.6	52.7
GSDT[66]	66.2	66.5	55.2	56.4	51.0
FairMOT[68]	73.7	72.3	59.3	60.9	58.0
CorrTracker[67]	76.5	73.6	60.7	62.8	58.9
Unicorn[72]	77.2	75.5	61.7	—	—
GRTU[73]	74.9	75.0	62.0	62.1	62.1
MAATrack[69]	79.4	75.9	62.0	64.2	60.2
ByteTrack[45]	80.3	77.3	63.1	64.5	62.0
OC-SORT[74]	78.0	77.5	63.2	—	63.2
BoT-SORT[63]	80.6	79.5	64.6	—	—
ChainedTracker[75]	66.6	57.4	49.0	53.6	45.2
GPAT (ours)	**77.1**	**73.9**	**59.9**	**60.4**	**59.8**

As shown in Table 3.3, the GPAT method is compared with various Transformer-based multi-object tracking methods on the MOT17 dataset, evaluating key metrics such as MOTA, IDF1, HOTA, DetA, and AssA. Overall, GPAT demonstrates outstanding performance across multiple metrics. Even though it falls slightly short in some specific metrics, GPAT's balanced and stable performance shows comprehensive advantages.

First, in terms of MOTA (Multiple Object Tracking Accuracy), GPAT achieves a score of 77.1. Although this score is slightly lower than that of P3AFormer [76] (81.2) and MOTRv2 [77] (78.6), it significantly outperforms TrackFormer [50] (65.0) and MOTR [36] (65.1). MOTA is a core metric in multi-object tracking that reflects an algorithm's ability to reduce missed and false detections. TrackFormer [50] and MOTR [36] are earlier Transformer-based tracking methods, primarily utilizing a single-layer Transformer architecture for detection and association. These methods often lack a more refined association mechanism, leading to missed or false detections in dynamic scenarios. In contrast, GPAT leverages multi-feature fusion and multi-level association strategies, effectively reducing such errors across complex scenarios.

For IDF1 (Identity F1 Score), GPAT achieves a score of 73.9, which is significantly higher than TransCenter [78] (62.2) and CSTrack [79] (72.6). IDF1 reflects the algorithm's ability to maintain consistent identities across frames, particularly crucial in scenarios with target

interactions or dense environments. Although TransTrack [35] and FUFET [80] also perform well in identity preservation, they mainly rely on a single association strategy, which can result in identity switches in challenging occlusion scenarios. GPAT enhances identity consistency by integrating multi-level association mechanisms and rich feature representations, providing greater robustness, especially when handling frequent occlusions and interactions.

For the HOTA (Higher Order Tracking Accuracy) metric, GPAT scores 59.9, slightly higher than CSTrack [79] (59.3) and FUFET [80] (57.9), indicating a balance in detection and association tasks. HOTA evaluates the combined accuracy of detection and association, making it suitable for assessing performance in complex scenarios. TransCenter [78] and TransTrack [35] have lower scores in this area due to their reliance on single association strategies, which limits their adaptability to the dynamic changes in multi-object scenarios. In contrast, GPAT's multi-feature fusion strategy improves its balance in detection and association, maintaining high accuracy even in dense or dynamic scenes.

DetA (Detection Accuracy) and AssA (Association Accuracy) metrics respectively measure the accuracy in detection and association. GPAT scores 60.4 in DetA and 59.8 in AssA, showing a strong balance between detection and association. Although MOTRv2 [77] and TransMOT [81] show slightly better performance in these metrics, GPAT achieves robust detection and association performance without compromising any critical aspects. TransCenter [78] and FUFET [80] tend to perform less effectively in detection or association tasks, largely due to their difficulty in balancing target identification and identity association in complex scenarios. GPAT's multi-layer detection and multi-feature association strategy enables it to excel in both detection and association tasks.

Table 3.3: Comparison to the Transformer-based methods on MOT17 dataset

Method	**MOTA**	**IDF1**	**HOTA**	**DetA**	**AssA**
TrackFormer[50]	65.0	63.9	—	—	—
TransTrack[35]	74.5	63.9	54.1	60.5	47.9
MOTR[36]	65.1	66.4	—	—	—
P3AFormer[76]	81.2	78.1	—	—	—
MOTRv2[77]	78.6	75.0	62.0	63.8	60.6
TransCenter[78]	73.2	62.2	54.5	60.1	49.7
CSTrack[79]	74.9	72.6	59.3	61.1	57.9
FUFET[80]	76.2	68.0	57.9	62.9	53.6
TransMOT[81]	76.7	75.1	—	—	—
GPAT (ours)	**77.1**	**73.9**	**59.9**	**60.4**	**59.8**

Table 3.4 provides a comparative analysis of various cutting-edge tracking methods applied to the TAO dataset, particularly emphasizing the mAP50 (mean Average Precision at 50% IoU) and FPS (frames per second) metrics. This table highlights the effectiveness of the GPAT method introduced in this thesis, especially when compared against the baseline model, GTR, and other prominent tracking approaches.

GPAT records a commendable mAP50 score of 23.1, notably outperforming the baseline GTR's score of 20.1 [29]. This substantial improvement in mAP50 is indicative of GPAT's superior ability to accurately detect and track objects across the varied and challenging scenarios presented by the TAO dataset. mAP50 is an essential metric because it reflects the precision with which the model identifies and localizes objects, a critical feature in environments where accuracy is paramount.

Moreover, GPAT excels in operational efficiency, achieving a processing speed of 13.2 FPS, which is significantly higher than the 11.2 FPS achieved by GTR [29]. This enhancement in speed, coupled with high tracking accuracy, is particularly advantageous for real-time applications where both rapid response and precision are required. The ability of GPAT to deliver high performance without sacrificing speed or accuracy ensures its suitability for scenarios that demand quick and reliable tracking solutions, such as autonomous driving or real-time surveillance.

When compared to other methods like SORT_TAO [82] and QDTrack [71], GPAT not only demonstrates superior mAP50 scores but also maintains a competitive edge in processing speed, solidifying its status as a leading solution for multi-object tracking on the TAO dataset. Notably, while AOA [83] achieves an even higher mAP50 of 27.5, its processing speed of only 1.0 FPS significantly restricts its usability in real-time applications, illustrating a common trade-off in tracking technology. GPAT's balanced performance thus stands out, offering an optimal mix of accuracy and speed.

The data presented in Table 3.4 not only substantiates the effectiveness of GPAT but also underscores its capability to maintain a balanced performance across crucial metrics. This balance is key to its utility in practical applications, where both the quality of tracking and the responsiveness of the system are equally important. These results corroborate the innovations detailed in Chapter 3 of this thesis, showcasing GPAT's robustness and suitability for tackling the complexities inherent in diverse tracking environments. GPAT's performance thus provides significant value in advancing the field of multi-object tracking, ensuring its relevance amidst a landscape of highly competitive alternatives.

Table 3.4: Comparison to the state-of-the-art on the TAO dataset.

Methods	**mAP50**	**FPS**
SORT_TAO[82]	10.2	15.2
QDTrack[71]	12.4	5.4
AOA[83]	27.5	1.0
GTR[29]	20.1	11.2
GPAT (ours)	**23.1**	**13.2**

3.3.5 Ablation Study

3.3.5.1 Effect of Different Detectors on Tracking Performance

The choice of detector has a profound impact on the performance of GPAT. As demonstrated in Table 3.5, different detection heads result in significant variations in tracking accuracy. YOLOv8, in particular, outperforms the other detectors, achieving a HOTA of 59.9, which is a significant improvement over CenterNet's 58.7 and YOLOv5's 55.1. This highlights YOLOv8's enhanced ability to detect and track objects accurately, which in turn improves overall system performance.

The performance improvements seen with YOLOv8 validate our original motivation for designing GPAT: to address the limitations of the baseline GTR, which heavily relies on the detector's performance. In scenarios where the detector fails, GTR struggles to maintain accurate trajectories, whereas GPAT shows marked resilience. The results indicate that GPAT's improvements

are rooted in a more robust detection mechanism that can better handle challenging tracking scenarios.

Table 3.5: Tracking performance comparison of different detectors. * denotes the default settings.

Detector	**HOTA**	**DetA**	**AssA**	**MOTA**
YOLOv5	55.1	55.4	53.9	70.5
CenterNet	58.7	58.5	57.9	73.6
YOLOv8*	59.9	60.4	59.8	77.1

3.3.5.2 Impact of Scenario Density on Tracking Performance

The experimental results in Table 3.6 illustrate how GPAT's performance varies across scenarios with different object densities, providing insights into its effectiveness under varying levels of complexity. The densities, measured as the average number of objects per frame, range from sparse environments (5.2 objects per frame) to extremely dense settings (69.8 objects per frame).

In low-density environments (5.2 and 9.6 objects per frame), GPAT achieves high Detection Accuracy (DetA) values of 68.7 and 67.4, respectively, along with MOTA scores of 79.2 and 78.5. This suggests strong detection performance when objects are well-separated. However, the Association Accuracy (AssA) at these levels is 52.4 and 48.8, indicating that maintaining consistent identities across frames can still be challenging in sparse scenes due to the lack of inter-object interactions, which limits temporal consistency cues.

As density increases to 14.3 objects per frame, GPAT achieves improved balance across all metrics—HOTA reaches 61.3, DetA is 64.3, and AssA climbs to 59.6. This suggests that moderate density provides richer contextual cues and interactions that enhance both detection and identity preservation, reflecting the model's capacity to leverage spatial relationships effectively.

At medium-to-high density (24.7 and 43.5), GPAT continues to perform robustly, with steady HOTA scores of 63.5 and 67.2, and MOTA values of 75.0 and 73.1. Notably, the AssA remains strong at 62.0 and 60.2, indicating reliable tracking in moderately crowded scenarios. These results reinforce GPAT's strength in maintaining both detection and association accuracy under increasing complexity.

In extremely dense settings (69.8 objects per frame), GPAT achieves its highest HOTA of 72.1 and DetA of 70.7, demonstrating outstanding detection performance. However, the AssA drops sharply to 25.4, revealing a significant weakness in identity consistency in highly crowded scenes. This drop is likely due to severe occlusion, overlaps, and appearance similarity, all of which complicate trajectory association despite accurate detection.

These findings from the extended density analysis highlight both the robustness and limitations of GPAT under varying complexity. The model maintains consistently high detection performance across densities, showcasing a reliable detection backbone. However, its association performance is more sensitive to crowding and scene complexity. This underscores the necessity of further enhancing GPAT's association module to sustain identity tracking in highly congested environments.

Ultimately, this analysis demonstrates the critical balance between detection and association in multi-object tracking. GPAT's performance confirms that while detection mechanisms may

generalize well across scenarios, effective identity association still requires improvements to address the challenges presented by extreme object density and crowd dynamics.

Table 3.6: Performance differences of GPAT in scenarios with varying density levels.

Density	**HOTA**	**DetA**	**AssA**	**MOTA**
5.2	60.1	68.7	52.4	79.2
9.6	57.2	67.4	48.8	78.5
14.3	61.3	64.3	59.6	76.3
24.7	63.5	62.1	62.0	75.0
43.5	67.2	65.8	60.2	73.1
69.8	**72.1**	**70.7**	**25.4**	**74.6**

3.3.5.3 Effect of the CBAM Mechanism on Tracking Accuracy

Table 3.7 presents the results of ablation experiments evaluating the impact of the Convolutional Block Attention Module (CBAM) in different components of a Transformer-based architecture for multi-object tracking. Each experimental setup explores how enabling or disabling CBAM in the encoder, decoder, or both affects key tracking metrics, including Higher Order Tracking Accuracy (HOTA), Detection Accuracy (DetA), Association Accuracy (AssA), and Multiple Object Tracking Accuracy (MOTA). The table includes clear indicators (✓/✗) to denote the presence or absence of CBAM in the encoder and decoder, facilitating an intuitive comparison between configurations.

In Case 1, where CBAM is enabled in both the encoder and decoder, the model achieves the highest MOTA of 77.1, along with balanced values for HOTA (59.9), DetA (60.4), and AssA (59.8). This demonstrates that dual-stage CBAM integration enhances the model's ability to extract and propagate refined features, leading to both accurate detection and robust identity tracking.

Case 2, with CBAM disabled in both components, results in a drop in detection metrics (DetA 55.4, MOTA 75.6), though AssA slightly improves to 60.7. This suggests that while removing CBAM may reduce detection precision, it might introduce some stability in identity association, possibly due to simpler or less fluctuating attention patterns. Nonetheless, the overall tracking performance degrades, confirming CBAM's importance in capturing relevant spatial-channel dependencies.

Case 3 shows the effect of enabling CBAM only in the encoder. It achieves the highest AssA score of 65.8, with DetA and MOTA at 60.5 and 72.6 respectively. The elevated AssA implies that encoder-level attention helps the model extract features that preserve identity over time, improving association. However, the absence of CBAM in the decoder limits the model's adaptability in handling varying detection outputs, affecting its end-to-end tracking efficiency.

Case 4, where CBAM is applied only in the decoder, results in balanced scores: MOTA 71.7, DetA 60.3, and AssA 59.8. While not outperforming the dual-CBAM setup, it performs comparably, showing that decoder-side attention still significantly contributes to refining tracking by enhancing focus during output decoding.

Overall, the study underscores the complementary role of CBAM in both encoding and decoding stages. Case 1 demonstrates the optimal configuration, where CBAM in both stages leads to

the best overall performance. These findings highlight the importance of strategically deploying attention enhancement modules like CBAM to improve both detection and association aspects of multi-object tracking systems.

Table 3.7: Ablation experiments for CBAM placement. The tick (✓) and cross (✗) indicate whether CBAM is enabled in the encoder or decoder.

Case	Encoder CBAM	Decoder CBAM	HOTA	DetA	AssA	MOTA
Case 1	✓	✓	59.9	60.4	59.8	77.1
Case 2	✗	✗	57.9	55.4	60.7	75.6
Case 3	✓	✗	62.9	60.5	65.8	72.6
Case 4	✗	✓	59.9	60.3	59.8	71.7

3.4 Conclusion

In this chapter, we introduced a new multi-object tracking algorithm called GPAT (Global Prediction Attention-enhanced Transformer). GPAT integrates a prediction module and the Convolutional Block Attention Module (CBAM) into the Global Tracking Transformer (GTR) framework, bringing significant enhancements to tracking robustness, accuracy, and adaptability in complex environments. These improvements address some of the key challenges faced by traditional GTR, making GPAT more effective for real-world multi-object tracking applications.

The addition of the prediction module is critical for handling missed detections, a common issue in multi-object tracking due to occlusions, lighting variations, and rapid object movements. In scenarios where the YOLOv8 detector fails to detect objects, GPAT leverages the Kalman filter-based prediction system to estimate the positions of undetected objects based on their historical trajectories. This capability allows GPAT to maintain continuous tracking by predicting the object's next position when detections are unavailable, effectively filling in gaps in object trajectories. The Kalman filter dynamically adjusts the uncertainty of these predictions, ensuring that the system remains robust even under challenging conditions. This enhancement directly addresses the limitations of traditional GTR, which heavily relies on detector output and struggles to maintain tracking continuity when detections are inconsistent. By integrating the prediction module, GPAT significantly improves trajectory stability and reduces tracking interruptions caused by missed detections.

CBAM further strengthens GPAT's feature extraction capabilities by applying channel and spatial attention mechanisms, enabling the model to focus on the most relevant features for tracking. Channel attention selectively amplifies important feature channels, enhancing the model's ability to prioritize critical information in the feature representation. This mechanism helps GPAT distinguish between foreground objects and background noise, a crucial advantage in dense or cluttered environments. Spatial attention, on the other hand, directs the model's focus to specific regions within the feature map where targets are likely located. By emphasizing spatially relevant areas, CBAM improves GPAT's object localization accuracy and reduces the influence of irrelevant background features. Together, these two attention mechanisms enable GPAT to perform well in varied and complex scenarios, where selective feature refinement is essential for accurate and consistent tracking. This dual-attention approach allows GPAT to generalize effectively across different scenes and maintain high tracking accuracy, even in challenging conditions with overlapping objects or occlusions.

The experimental results on the MOT17 and TAO datasets demonstrate GPAT's superior performance compared to both traditional and Transformer-based tracking methods. On the MOT17 dataset, GPAT achieves a MOTA (Multiple Object Tracking Accuracy) score of 77.1, a marked improvement over the baseline GTR model's score. This improvement indicates that GPAT's enhancements in handling missed detections and refining feature attention directly contribute to better tracking accuracy and reliability. Additionally, GPAT's IDF1 (Identity F1 Score) score of 73.9 showcases its improved ability to maintain consistent object identities across frames, a key advantage in scenarios with frequent occlusions and high object densities. The increased Association Accuracy (AssA) of 59.8 further underscores GPAT's robustness in maintaining object associations throughout the video sequence, reducing identity switches and ensuring more stable tracking.

The ablation studies conducted provide further insights into the contributions of different components in GPAT. The choice of detector, for instance, has a significant impact on tracking performance. Experiments reveal that using YOLOv8 as the detector yields higher HOTA (Higher Order Tracking Accuracy) and MOTA scores compared to other detectors such as YOLOv5 and CenterNet. This finding confirms that YOLOv8's improved detection accuracy directly enhances GPAT's overall tracking robustness, especially in scenarios with complex backgrounds or dense object interactions. The study highlights the importance of robust detection mechanisms in maintaining accurate object trajectories and minimizing identity switches in challenging environments.

The impact of scene density on GPAT's performance is also notable. In low-density scenarios, GPAT achieves a HOTA score of 57.2, reflecting strong detection performance. However, in high-density environments with up to 69.8 objects per frame, the HOTA score increases to 72.1, while the Association Accuracy (AssA) decreases to 25.4. These results indicate that GPAT can effectively detect objects in dense scenes, but maintaining consistent associations in such environments remains challenging. This suggests a potential area for future optimization in GPAT's association mechanisms, as improving consistency in highly dense scenes could further enhance its applicability to crowded environments.

The role of CBAM in GPAT's performance is evident from the ablation experiments. When CBAM is enabled in both the encoder and decoder stages, GPAT achieves the highest MOTA score of 77.1 and a HOTA score of 59.9, underscoring the impact of attention mechanisms in refining feature representations. Without CBAM, GPAT's performance declines, particularly in terms of detection accuracy and MOTA, indicating that CBAM's feature refinement plays a crucial role in enhancing GPAT's tracking capabilities. The study also reveals that enabling CBAM solely in the encoder or decoder results in suboptimal performance, suggesting that both stages benefit from CBAM's selective attention mechanisms. This finding emphasizes the value of strategically integrating attention modules within the Transformer architecture to maximize tracking accuracy and robustness.

In conclusion, GPAT represents a significant advancement in multi-object tracking by integrating a prediction module and CBAM into the GTR framework. These enhancements enable GPAT to achieve high accuracy and reliability in dynamic and complex environments, addressing key challenges faced by traditional tracking methods. The prediction module mitigates the impact of detector failures by maintaining trajectory continuity during missed detections, while CBAM's dual-attention mechanism provides selective feature refinement for improved object localization and identity preservation. The experimental results and ablation studies validate GPAT's effectiveness, highlighting its balanced performance in detection, association, and real-

time processing. GPAT's robust performance on both the MOT17 and TAO datasets, combined with its efficient processing speed, makes it a valuable tool for a wide range of real-world applications. The insights gained from this chapter underscore the importance of prediction and attention mechanisms in achieving state-of-the-art multi-object tracking, paving the way for further innovations in the field.

Chapter 4

Multi Queries and Multi-Weight Computation

4.1 Introduction

The Global Tracking Transformer (GTR) has notable limitations that affect its performance in complex multi-object tracking scenarios. Firstly, GTR is highly dependent on detection accuracy. This reliance means that any errors or missed detections in the initial detector stage have a direct impact on the entire tracking process. Even a minor detection failure can lead to substantial errors downstream, causing GTR to lose track of objects or misassociate identities, especially in scenes with occlusions or frequent interactions among objects. Consequently, GTR's effectiveness can be significantly compromised in real-world scenarios where detection is often imperfect.

Furthermore, GTR's query mechanism, which is anchored to specific reference points, is relatively inflexible. This anchored query setup makes it challenging for GTR to adapt to dynamic scenes where object positions and appearances can vary significantly. As a result, GTR struggles in tracking objects that exhibit unpredictable movements or undergo rapid appearance changes, such as in crowded or fast-paced environments. The anchored approach constrains the model's ability to generate flexible hypotheses for object locations, reducing its adaptability and robustness.

To overcome these limitations, we introduce an advanced approach, the Multi Queries Tracking Transformer (MQTFormer). This method leverages multi-queries and multi-weight computations to significantly enhance tracking robustness and mitigate GTR's inherent weaknesses. MQTFormer's multi-query framework addresses the rigidity of GTR's anchored queries by introducing a diverse set of queries that allow the model to generate multiple hypotheses for each object's trajectory. This diversity in hypotheses helps the model to better capture variations in object movement, effectively handling scenes where objects exhibit complex and dynamic behaviors. By enabling multiple perspectives on object positions, MQTFormer can dynamically adapt to changes in scene structure and object motion, enhancing its adaptability across a range of challenging tracking environments.

In addition to multi-queries, MQTFormer incorporates multi-weight computations, which bring a nuanced approach to feature aggregation. Unlike GTR's static weighting, multi-weight computations allow MQTFormer to assign varying levels of importance to different aspects of the

tracking process. For instance, weights can be dynamically adjusted based on feature matching, confidence levels, and temporal consistency, leading to a more balanced and context-aware feature representation. This flexibility allows MQTFormer to respond intelligently to different tracking conditions, prioritizing reliable features and minimizing the impact of uncertain ones. Multi-weight computation thus mitigates the impact of detection failures, allowing the model to maintain tracking accuracy even when detection signals are imperfect.

Together, multi-queries and multi-weight computation significantly improve MQTFormer's performance over GTR. The multi-query mechanism reduces the model's dependency on any single hypothesis, providing more robust tracking in unpredictable scenes. Meanwhile, multi-weight computation enhances the model's decision-making capacity, allowing it to weigh various factors in real time to improve overall accuracy and stability. By addressing GTR's limitations in detection dependency, query rigidity, and limited spatial-channel processing, MQTFormer achieves a higher level of tracking reliability and robustness, making it more suitable for complex, real-world tracking applications.

In this section, we will further explore the technical details and benefits of multi-queries and multi-weight computation in the context of MQTFormer, illustrating how these innovations contribute to an enhanced tracking framework that overcomes GTR's core challenges.

This Work is established and ready for publication.

4.2 Method

One of the key innovations is the use of multiple queries for tracking. The multi-query approach involves selecting more frames as query frames within a sliding window, whose size is adjustable. By doing so, the system captures a broader temporal context, which enhances the robustness of the tracking task. The sliding window allows the model to take into account not just the current frame but also adjacent frames, helping the model better understand the movement and behavior of the tracked objects across multiple frames. This reduces the likelihood of losing track of an object due to short-term occlusions or detection failures, as the system can reference information from neighboring frames to maintain consistent tracking.

This method is particularly useful in scenarios where objects may be temporarily obscured or when the tracking environment is highly dynamic. By relying on multiple query frames, the system can maintain a more consistent tracking performance, ensuring that the trajectory of each object is accurately followed throughout the sequence.

To further enhance tracking performance, the method incorporates a weighted average approach for processing multi-queries. This process includes three key types of weighting: feature matching weighting, confidence weighting, and dynamic adjustment weighting.

Feature matching weighting is based on the similarity between the features of each query frame and the current frame. This similarity is often measured using metrics such as cosine similarity or Euclidean distance, allowing the system to determine how closely the features from a given query frame align with those from the current frame. By placing more emphasis on query frames with higher similarity, the method prioritizes temporal information that is more relevant for tracking the target. This ensures that the model can better capture the target's consistent appearance across frames, even when there are small variations due to noise or minor occlusions.

Confidence weighting considers the detection confidence of each query frame. Each query frame typically has an associated detection score, which reflects how confidently the system detected the object in that frame. Frames with higher detection confidence are assigned more weight, ensuring that the most reliable information is given priority during the tracking process. This helps reduce the influence of frames where the detection was less certain, minimizing the risk of using inaccurate or noisy data for tracking.

Dynamic adjustment weighting takes into account the dynamic behavior of the target, such as its speed, acceleration, and any environmental changes, which can affect the appearance or position of the object. This component adjusts the weighting to account for changes in the target's motion characteristics, ensuring that the system remains responsive to rapid changes in speed or direction. By adapting to these factors, the system can maintain tracking accuracy in more complex and rapidly changing environments, such as when the target moves unpredictably or when the scene undergoes significant transformations.

The combination of these weights allows the system to compute a weighted average of the features across all selected query frames. This ensures that the tracking process is both flexible and robust, able to incorporate reliable and relevant temporal information while dynamically adjusting to the tracking environment. The final weighted feature set is used to update the trajectory of the tracked object, providing a more accurate and reliable tracking output. This multi-query, multi-weight approach, with its focus on leveraging multiple temporal contexts and adapting to dynamic tracking conditions, represents a significant advancement over traditional single-query methods. As a result, it offers improved performance, particularly in challenging scenarios where occlusions, sudden movements, or environmental changes can negatively impact tracking accuracy.

4.2.1 Overview of Network Architecture

The workflow of the proposed multi-weight computation mechanism within the Transformer architecture is illustrated in Figure 4.1. The process begins with the input of a video sequence and proceeds through various stages, including detection, encoding, and decoding, ultimately resulting in the generation of a similarity vector. The detailed steps are as follows.

First, the system starts by receiving a video sequence as input. The video is segmented into individual frames, which serve as the primary data input for the subsequent stages of processing.

Next, each frame from the video sequence is passed through the detector. The role of the detector is to identify and locate objects within each frame, producing bounding boxes and associated feature representations for the detected objects. These detected features are then forwarded to the next stage of the pipeline.

Unlike the original GTR approach, which selects only the last frame within a sliding window to generate queries, the proposed method selects multiple frames as query frames. These frames are highlighted with red boxes in the multi-queries section. By selecting frames from different points within the sliding window, the system can create more robust queries that account for temporal variations and provide a more comprehensive representation of the target. This approach significantly enhances the robustness and accuracy of the tracking task.

The core of the proposed method is the Decoder, which incorporates the multi-weight computation mechanism. The Decoder is composed of several key components. First, the multi-queries are processed through a multi-head attention mechanism. This step allows the Decoder to

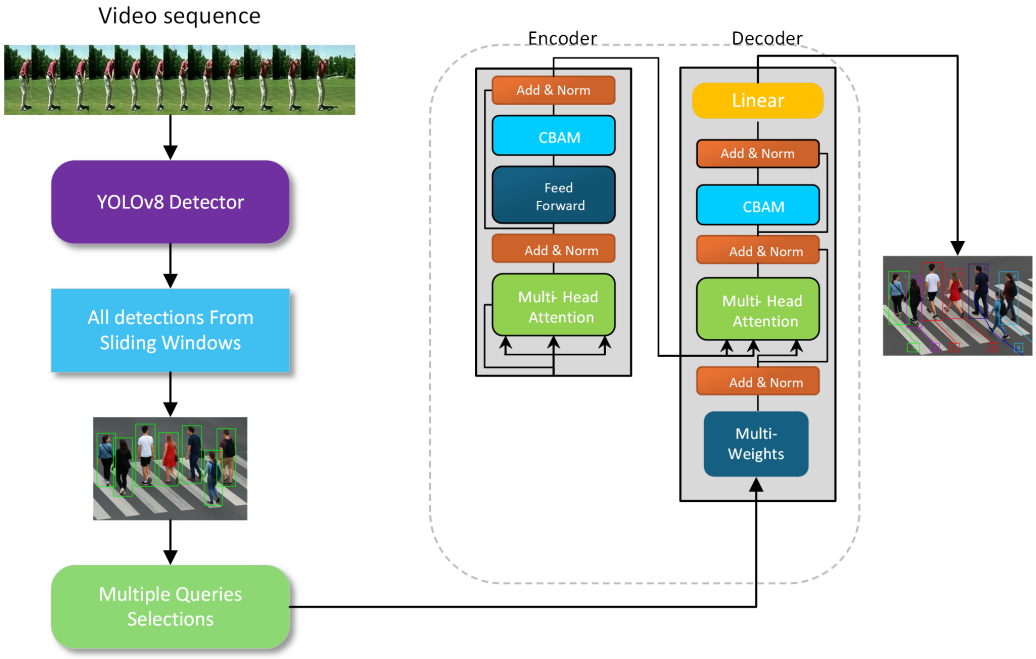

Figure 4.1: The MQTFormer pipeline, illustrating the multi-query tracking mechanism which integrates multiple query frames to enhance tracking robustness.

attend to different aspects of the input features simultaneously, enabling it to capture complex relationships between the queries and the encoded features, which are the output of the Encoder. These encoded features represent the input data in a more abstract, high-dimensional space, capturing important object attributes such as appearance and context, which the Decoder uses to inform its decisions.

One of the critical weighting mechanisms applied is feature matching weighting, which evaluates the similarity between the query features and the current frame features. Another key component is confidence weighting, which adjusts the weight based on the confidence level of the detection in each frame. This ensures that the most reliable detections are given more importance during the attention computation. The final weighting mechanism is dynamic adjustment, which accounts for changes in the target's motion dynamics, such as speed, acceleration, and environmental factors. This dynamic adjustment ensures that the tracking system remains adaptable to varying conditions.

The three weightings—feature matching weighting (FM), confidence weighting (CW), and dynamic adjustment weighting (DA)—are then combined into a single composite weight, which is used to modulate the attention mechanism within the multi-head attention. This composite weight enhances the accuracy and robustness of the similarity vector computation, allowing the Decoder to more effectively track objects across varying frames and conditions.

After applying the weighted attention mechanism, the Decoder outputs a similarity vector. This vector represents the likelihood that the trajectory queries match the current frame features, and it is used for making final tracking decisions.

Finally, the similarity vectors are further refined through multiple layers of cross-attention within the Decoder. Each layer refines the attention based on the composite weights, contributing to the final decision of object tracking.

4.2.2 Multi Queries in MQTFormer

The multi-queries approach enhances the robustness of the tracking task by selecting multiple frames within a sliding window to generate queries[84], rather than relying on the final frame alone, as done in traditional methods like GTR. By doing so, this method captures more comprehensive temporal information, leading to better performance in complex tracking scenarios.

To understand the multi-queries approach, we start by considering the likelihood of assigning an object to a specific track using a single query. In traditional tracking methods, such as in GTR, the likelihood L_A is calculated based on the similarity between features derived from a single query frame and the target object. This is represented as:

$$L_A(\alpha_t = i \mid q_k, F) = \frac{\exp(g_{ti}(q_k, F))}{\sum_{j \in \{0,1,\ldots,N_t\}} \exp(g_{tj}(q_k, F))} \tag{4.1}$$

Here:

- α_t denotes the assignment of the object at time t to track i.
- q_k represents the object feature derived from the target features in the detection window of the k-th query frame within the sliding window.
- F represents the features extracted from all detection windows of the current frame.
- $g_{ti}(f_{\text{detection}}, f_{\text{target}})$ is the function that computes the similarity between the features $f_{\text{detection}}$ extracted from the detection windows in the current frame, which capture the visual characteristics of objects (such as shape, texture, and appearance), and the target f_{target} of i.
- The denominator sums over all possible track assignments j at time t, including the possibility of assigning the object to a new track or leaving it unassigned (denoted by $j = 0$).

However, relying on a single query may be suboptimal in cases where the frame q_k does not capture the most informative features due to occlusions, motion blur, or other factors. Therefore, we extend this approach to consider multiple queries from different frames within the sliding window.

Let us now derive the likelihood function when multiple queries are used. In this context, k represents the index of the k-th query target. The queries $\{q_{k1}, q_{k2}, \ldots, q_{km}\}$ are features selected from m different frames within the sliding window, each corresponding to the same target k. Each query q_{ki} captures the target's features from a different frame in the sliding window, reflecting the target's appearance or motion across multiple frames. The joint likelihood of assigning the object to track i using these multiple queries can be formulated as:

$$L_A(\alpha_t = i \mid \{q_{k1}, q_{k2}, \ldots, q_{km}\}, F) = \prod_{n=1}^{m} L_A(\alpha_t = i \mid q_{kn}, F) \tag{4.2}$$

Substituting the likelihood from the single query formula, we get:

$$L_A(\alpha_t = i \mid \{q_{k1}, q_{k2}, \ldots, q_{km}\}, F) = \prod_{n=1}^{m} \frac{\exp(g_{ti}(q_{kn}, F))}{\sum_{j \in \{0,1,\ldots,N_t\}} \exp(g_{tj}(q_{kn}, F))} \tag{4.3}$$

Taking the logarithm on both sides to simplify the computation, we obtain:

$$\log L_A(\alpha_t = i \mid \{q_{k1}, q_{k2}, \ldots, q_{km}\}, F) = \sum_{n=1}^{m} \left[g_{ti}(q_{kn}, F) - \log \left(\sum_{j \in \{0,1,\ldots,N_t\}} \exp(g_{tj}(q_{kn}, F)) \right) \right] \tag{4.4}$$

Recognizing that the second term in the product corresponds to the normalization factor (since the sum of probabilities must equal one), we simplify this to:

$$L_A(\alpha_t = i \mid \{q_{k1}, q_{k2}, \ldots, q_{km}\}, F) = \frac{\exp\left(\sum_{n=1}^{m} g_{ti}(q_{kn}, F)\right)}{\sum_{j \in \{0,1,\ldots,N_t\}} \exp\left(\sum_{n=1}^{m} g_{tj}(q_{kn}, F)\right)} \tag{4.5}$$

Finally, we express this in a more intuitive form:

$$L_A(\alpha_t = i \mid \{q_{k1}, q_{k2}, \ldots, q_{km}\}, F) = \frac{\exp\left(g_{ti}(\{q_{k1}, q_{k2}, \ldots, q_{km}\}, F)\right)}{\sum_{j \in \{0,1,\ldots,N_t\}} \exp\left(g_{tj}(\{q_{k1}, q_{k2}, \ldots, q_{km}\}, F)\right)} \tag{4.6}$$

This final equation represents the likelihood of assigning an object to a specific track using multiple queries. The key advantage of this approach lies in its ability to integrate information from multiple frames, thereby leading to a more robust and accurate tracking performance.

4.2.3 Multi-weight computation

The multi-weight computation involves three key types of weights: feature matching weight, confidence weight, and dynamic adjustment weight. First, for each query frame in the sliding window, the feature matching weight is computed to measure the similarity between the current detection window and the query features. Next, the confidence weight evaluates the reliability of the detection in the current frame, while the dynamic adjustment weight accounts for changes in the target's motion dynamics, such as speed and acceleration. These weights are combined to form a composite weight for each frame, as illustrated in Figure 4.2.

The feature matching weight is used to measure the similarity between the current frame and multiple query frames. By using similarity metrics (e.g., cosine similarity), we determine the matching degree between each query frame and the target object. This similarity score is directly related to the probability calculation, as this probability function uses the similarity score to estimate the likelihood of a match between the target and the query. Specifically, higher similarity increases the probability of correctly associating the current frame with the query frame.

The formula for the feature matching weight is:

$$w_{\text{feature},i} = \text{sim}(f_i, f_{\text{current}}) \tag{4.7}$$

where the similarity function $\text{sim}(f_i, f_{\text{current}})$ is calculated using cosine similarity as:

$$\text{sim}(f_i, f_{\text{current}}) = \frac{f_i \cdot f_{\text{current}}}{\|f_i\| \|f_{\text{current}}\|} \tag{4.8}$$

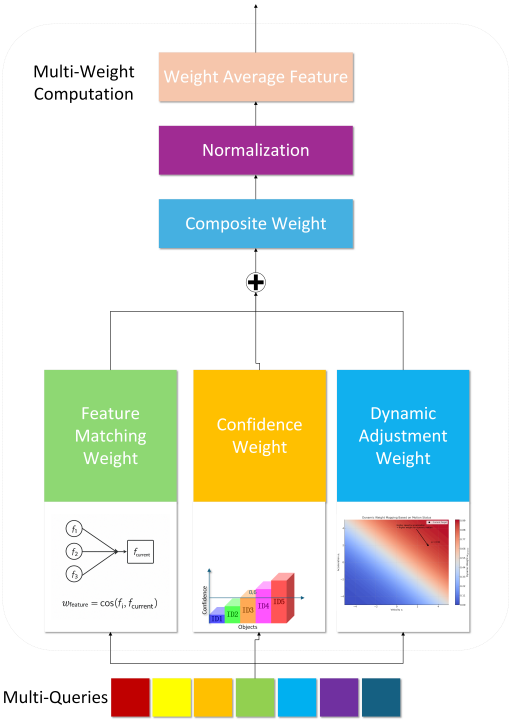

Figure 4.2: The pipeline of Multi-weight computation. This diagram illustrates the flow of the multi-weight computation process, including feature matching, confidence evaluation, and dynamic adjustments for object tracking. The steps are linked with multi-query frames to enhance temporal and spatial consistency in the tracking task.

Here, f_i represents the object feature of the i-th query object in the sliding window, and f_{current} denotes the object feature extracted from the detection window in the current frame. The feature matching weight evaluates the similarity between the query frames and the object feature of the current frame, assigning higher weights to query frames with object features that more closely match those of the current frame.

The confidence weight is assigned based on the detection confidence of each query frame. Higher confidence indicates a more reliable detection result, thereby increasing the frame's weight in the overall tracking computation. The formula is:

$$w_{\text{conf},i} = \text{conf}_i \tag{4.9}$$

where conf_i represents the detection confidence of the i-th query frame. This weight ensures that frames with high detection confidence have a larger impact in tracking computation, enhancing tracking accuracy.

The dynamic adjustment weight reflects the target's motion characteristics, such as speed and acceleration, allowing the tracking system to adapt to variations in the target's movement. The formula is:

$$w_{\text{dynamic},i} = \frac{1}{1 + \exp(-(\alpha v_i + \beta a_i))} \tag{4.10}$$

where:

- v_i represents the speed of the target in the i-th query frame,
- a_i represents the acceleration of the target in the i-th query frame,
- α and β are coefficients that control the contribution of speed and acceleration to the dynamic adjustment weight.

This formula uses a sigmoid function applied to a weighted combination of speed and acceleration, ensuring the resulting weight is between 0 and 1. The dynamic adjustment weight helps the system maintain flexibility and accuracy in dynamically changing environments.

The three weights above are combined through addition to form a composite weight:

$$w_i = w_{\text{feature},i} + w_{\text{conf},i} + w_{\text{dynamic},i} \tag{4.11}$$

To ensure the sum of all weights is 1, the composite weight is normalized as follows:

$$w_{\text{normalized},i} = \frac{w_i}{\sum_{j=1}^{S} w_j} \tag{4.12}$$

Here, S is the total number of frames in the sliding window, w_i represents the composite weight of the i-th query frame, and $w_{\text{normalized},i}$ denotes the normalized weight for the i-th query frame. The index j iterates over all frames within the sliding window, so $\sum_{j=1}^{S} w_j$ is the sum of all composite weights in the sliding window. This normalization ensures that all weights sum to 1, making each normalized weight range between 0 and 1.

Based on the normalized weights, we calculate the weighted average feature for subsequent tracking tasks:

$$f_{\text{weighted}} = \sum_{i=1}^{S} (w_{\text{normalized},i} \times f_i) \tag{4.13}$$

Formula (4.13) applies specifically to the query frames because these frames are selected from the sliding window to represent the target at different time points. By performing a weighted average on these selected query frames, the system can capture the temporal dynamics of the target, thereby improving tracking accuracy.

The integration of multi-weight computation and multi-queries significantly enhances tracking robustness, particularly in calculating the final likelihood $L_A(\alpha_t = i \mid q_k, F)$. Traditional GTR methods often rely on the last frame in the sliding window as the sole query frame for feature comparison. However, using only the last frame may lead to limitations, such as motion blur or occlusion, which affect feature comparison accuracy.

With the introduction of the multi-queries concept, the system selects multiple representative frames as query frames from the sliding window, capturing the target's dynamic changes over time. These query frames provide multiple reference points for feature comparison, increasing the robustness of the feature comparison and reducing tracking errors caused by anomalies in a single frame.

When calculating the final likelihood, the similarity function $g_{ti}(q_{\text{multi}}, F)$ is based on the weighted average feature f_{weighted}, derived from multi-queries:

$$L_A(\alpha_t = i \mid q_{\text{multi}}, F) = \frac{\exp(g_{ti}(q_{\text{multi}}, F))}{\sum_{j\in\{0,1,\ldots,N_t\}} \exp(g_{tj}(q_{\text{multi}}, F))} \tag{4.14}$$

where q_{multi} represents the weighted feature result of multiple query frames, capturing spatial and temporal information from the sliding window. This approach improves system robustness in dynamic or occluded environments.

During feature comparison with multi-queries, each detection bounding box undergoes multi-weight computation. This process considers not only feature matching similarity but also the confidence of each bounding box and the dynamic adjustment for the target's motion. As a result, each detection bounding box contributes differently to the likelihood calculation, with its weight dynamically adjusted based on its reliability and importance.

The final likelihood calculation is given by:

$$L_A(\alpha_t = i \mid q_k, F) = \frac{\exp(g_{ti}(q_k, F))}{\sum_{j\in\{0,1,\ldots,N_t\}} \exp(g_{tj}(q_k, F))} \tag{4.15}$$

After introducing multi-queries and multi-weight computation, $g_{ti}(q_{\text{multi}}, F)$ is calculated based on the weighted average feature of multiple query boxes, rather than relying on a single query box.

The multi-queries q_{multi} are selected from different frames in the sliding window, capturing the target's temporal dynamics. The set of multi-queries is represented as:

$$q_{\text{multi}} = \{q_1, q_2, \ldots, q_S\} \tag{4.16}$$

where q_i represents the query box extracted from the i-th frame within the sliding window, and S is the total number of frames. This formulation shows that q_{multi} consists of query boxes from multiple frames, providing richer temporal context.

Specifically, through multi-weight computation, we obtain a stable, representative weighted average feature f_{weighted} from multiple query boxes for similarity calculation:

$$g_{ti}(q_{\text{multi}}, F) = \text{similarity}(f_{\text{weighted}}, F) \tag{4.17}$$

The final likelihood calculation formula is updated as:

$$L_A(\alpha_t = i \mid q_{\text{multi}}, F) = \frac{\exp(g_{ti}(q_{\text{multi}}, F))}{\sum_{j\in\{0,1,\ldots,N_t\}} \exp(g_{tj}(q_{\text{multi}}, F))} \tag{4.18}$$

Here, q_{multi} represents the weighted feature result of multiple query boxes. The combination of multi-queries and multi-weight computation enables the system to more effectively utilize temporal information, enhancing tracking robustness and reducing accuracy degradation caused by single-frame anomalies.

In the Decoder, the multi-weight computation process integrates seamlessly with the multiple queries. The three types of weights—feature matching, confidence, and dynamic adjustment—each play a critical role in refining the composite weight used in the cross-attention

mechanism. The feature matching weight ensures that query frames that most resemble the current frame contribute more, the confidence weight prioritizes frames with high detection reliability, and the dynamic adjustment weight adapts based on the target's motion characteristics.

These composite weights are normalized and used to compute a weighted average feature, which is then employed in the Decoder's multi-head attention mechanism to generate a similarity vector that accurately reflects the dynamic behavior of the target across multiple frames. This process enhances the Decoder's ability to associate the correct trajectory with each target, leading to improved performance in challenging tracking scenarios such as occlusion and sudden motion changes.

This visualization illustrates the workflow and highlights the stages in which each weight type contributes to the final similarity vector. By structuring the Decoder in this way, the system can make more robust and accurate tracking decisions based on the temporal and spatial context accumulated from multiple frames.

4.3 Experiment

This section presents the experimental results and analysis of the proposed methods. We evaluate the performance of GPAT and MQTFormer using standard multi-object tracking benchmarks and compare them with existing approaches. The experiments aim to validate the effectiveness of our designs under various tracking scenarios, including occlusions, complex environments, and different levels of object density.

4.3.1 Experiment Setup

The experimental setup follows the same configuration described in Section 3.3.1 of Chapter 3.

4.3.2 Experiment Data

The datasets used and their details are consistent with those introduced in Section 3.3.2.

4.3.3 Evaluation Metrics

To avoid redundancy in the evaluation metrics, please refer to 3.3.3 for further details.

4.3.4 Experimental Results

In this detailed comparative study shown in Table 4.1, the proposed MQTFormer (Multi-Queries Tracking Transformer) is evaluated against a diverse set of state-of-the-art multi-object tracking methods on the MOT17 test set. The evaluation covers five key metrics—MOTA, IDF1, HOTA, DetA, and AssA—to provide a holistic view of performance in accuracy, identity consistency, and association reliability.

MQTFormer achieves a MOTA of 79.2, placing it among the top-performing methods such as ByteTrack (80.3) and BoT-SORT (80.6). This high score indicates that MQTFormer effectively minimizes false negatives and false positives while handling identity switches adeptly, which are critical factors in dynamic and densely populated scenes.

The IDF1 score of 75.3 reflects its strong ability to preserve object identities across frames, which is crucial in scenarios where targets frequently occlude or interact. The method's success in maintaining identity consistency demonstrates the effectiveness of its multi-query tracking mechanism in modeling temporal dependencies.

Notably, MQTFormer excels in HOTA, achieving a score of 63.1. This performance metric evaluates the method's ability to balance detection and association accuracy, suggesting that MQTFormer is particularly effective in scenarios involving complex movements and interactions among multiple objects. Its success in this area can be attributed to its novel approach of utilizing multiple queries and a sophisticated weighting system that enhances its adaptability to different tracking challenges.

Nevertheless, we acknowledge that some existing methods surpass MQTFormer in specific metrics.For example, MQTFormer achieves a DetA of 58.2, which is somewhat lower than ByteTrack [45] and MAATrack [69], which score 64.5 and 64.2, respectively. This observation suggests that while MQTFormer is highly effective at modeling temporal associations, there is room for improvement in frame-level detection accuracy. A likely cause is that classical methods such as ByteTrack utilize highly optimized detectors with aggressive filtering and straightforward matching, which favor high detection precision. In contrast, MQTFormer prioritizes the robustness of identity modeling through temporal consistency and multi-weight attention, which may result in slightly lower raw detection accuracy in exchange for more stable long-term tracking.

On the other hand, MQTFormer achieves the highest AssA score of 68.8 among all compared methods, highlighting its strength in robust association under motion variation and occlusion. This performance is attributed to the dynamic adjustment weighting module, which allows the model to adapt its attention distribution according to the object's motion pattern and detection confidence.

Overall, the results validate the effectiveness of MQTFormer's design philosophy, which emphasizes temporal context, multi-query fusion, and adaptive weighting. While some trade-offs exist, particularly in detection alignment, MQTFormer demonstrates superior identity association and holistic tracking capabilities, marking a significant advancement over conventional single-query Transformer-based models. This analysis also provides valuable insights for future work, such as integrating stronger detection backbones or hybrid matching schemes to further improve detection accuracy without compromising association robustness.

4.3.5 Ablation Study

4.3.5.1 Number of Queries

In this ablation experiment, the objective is to evaluate the effect of different numbers of query frames on tracking performance and highlight how increasing the number of queries enhances the system's robustness. This study explores four configurations: single-query, 2-queries, 3-queries, and 4-queries, aiming to determine the optimal balance between tracking accuracy and computational efficiency.

The significance of this experiment lies in its relationship to global tracking principles, where it is critical to capture temporal context across multiple frames. In traditional tracking systems, single-query methods face challenges, particularly in scenarios involving occlusions or identity switches, where the system may lose track of an object when it is momentarily hidden or when detection fails. Single-query approaches rely on information from only one frame, which limits

Table 4.1: Categorized comparison of MQTFormer to state-of-the-art methods on the MOT17 test set.

Method	MOTA	IDF1	HOTA	DetA	AssA
Classical Methods					
Tracktor++ [64]	53.5	52.3	44.8	44.9	45.1
CenterTrack [65]	67.8	64.7	52.2	53.8	51.0
GSDT [66]	66.2	66.5	55.2	56.4	51.0
FairMOT [68]	73.7	72.3	59.3	60.9	58.0
ByteTrack [45]	80.3	77.3	63.1	64.5	62.0
OC-SORT [74]	78.0	77.5	63.2	—	63.2
BoT-SORT [63]	80.6	79.5	64.6	—	—
SORT [42]	59.8	53.8	—	—	—
DeepSORT [43]	61.4	62.2	—	—	—
PermaTrack [85]	73.8	68.9	55.5	62.8	51.2
Embedding Methods					
TraDeS [70]	69.1	63.9	52.7	55.2	50.8
QuasiDense [71]	68.7	66.3	53.9	55.6	52.7
CorrTracker [67]	76.5	73.6	60.7	62.8	58.9
MAATrack [69]	79.4	75.9	62.0	64.2	60.2
CSTrack [79]	74.9	72.6	59.3	61.1	57.9
ChainedTracker [75]	66.6	57.4	49.0	53.6	45.2
Unicorn [72]	77.2	75.5	61.7	—	—
DeepMOT [86]	61.8	62.4	—	—	—
JDE [87]	64.4	55.8	—	—	—
NCT [31]	69.5	68.5	54.6	64.7	53.7
Transformer-based Methods					
TrackFormer [50]	65.0	63.9	—	—	—
TransTrack [35]	74.5	63.9	54.1	60.5	47.9
MOTR [36]	65.1	66.4	—	—	—
MOTRv2 [77]	78.6	75.0	62.0	63.8	60.6
P3AFormer [76]	81.2	78.1	—	—	—
TransCenter [78]	73.2	62.2	54.5	60.1	49.7
TransMOT [81]	76.7	75.1	—	—	—
MeMOT [88]	72.5	69.0	56.9	—	55.2
GTR [29]	75.3	71.5	59.1	61.6	57.0
MQTFormer (ours)	**79.2**	**75.3**	**63.1**	**58.2**	**68.8**

their ability to handle dynamic and complex environments.

The results of this experiment demonstrate the advantage of using multiple queries. The single-query approach shows lower tracking accuracy (with a HOTA of 55.6, DetA of 54.3, AssA of 61.7, and MOTA of 72.4), reflecting the inherent limitations of relying on limited temporal information. Adding more query frames significantly enhances the model's ability to maintain accurate object associations and reduce the impact of occlusions.

With 2-queries, the results improve noticeably (HOTA 60.5, DetA 56.7, AssA 65.8, and MOTA 76.2), as incorporating additional temporal context allows the system to perform more robustly. The tracking process becomes more stable, particularly in scenarios where objects move in and out of view or interact closely with others.

The 3-query setting, which yields the best results (HOTA 63.1, DetA 58.2, AssA 68.8, and MOTA 79.2), strikes the optimal balance. This configuration provides enough temporal context to capture object movements across frames while avoiding the risk of introducing too much noise or redundancy. It demonstrates that using three query frames is sufficient to ensure reliable tracking without unnecessary computational overhead.

With 4-queries, the performance remains strong (HOTA 61.9, DetA 57.6, AssA 67.5, and MOTA 78.1), but the results suggest diminishing returns. Although adding more queries enhances robustness, the improvements become marginal compared to the 3-query setup. This indicates that beyond a certain point, adding more queries does not provide significant additional benefits and may lead to redundant information being processed.

As shown in Table 4.2, the experiment underscores the importance of fine-tuning the number of query frames for optimal tracking performance. The multi-query approach, particularly with 3-queries, ensures a broader temporal context is captured without introducing excessive computational costs. It highlights how this strategy can significantly improve tracking robustness in real-world applications such as video surveillance, sports analysis, and autonomous driving, where continuous and reliable object tracking is essential for overall system performance.

Table 4.2: Comparison of Single-query, 2-queries, 3-queries, and 4-queries settings on key tracking metrics.

Number of Queries	**HOTA**	**DetA**	**AssA**	**MOTA**
Single-query	55.6	54.3	61.7	72.4
2-queries	60.5	56.7	65.8	76.2
3-queries	**63.1**	**58.2**	**68.8**	**79.2**
4-queries	61.9	57.6	67.5	78.1

4.3.5.2 Weight Selection

In this ablation study, as shown in Table 4.3, we evaluated the impact of various weighting mechanisms—both individually and in combination—on the performance of the multi-object tracking system. The goal of this experiment was to assess the effectiveness of each weighting strategy and to understand how these mechanisms influence the overall tracking accuracy.

Initially, we tested the three individual weighting mechanisms: Feature Matching Only, Confidence Only, and Dynamic Adjustment Only. The results show that Dynamic Adjustment weighting achieved the best performance across all metrics, with a HOTA score of 61.8, DetA of 56.2, AssA of 64.5, and MOTA of 77.7. This indicates that Dynamic Adjustment weighting is particularly effective at adapting to changes in the scene, thereby providing stronger association capability and detection accuracy.

We then examined combinations of two weighting mechanisms. The combination of Confidence and Dynamic Adjustment weighting showed relatively better results among the combinations, achieving a HOTA score of 60.5, DetA of 55.2, AssA of 61.9, and MOTA of 75.1. Overall, the combined weighting schemes provided stable performance, though certain individual weighting mechanisms were able to yield higher performance for specific metrics.

Interestingly, some combined mechanisms underperform compared to the best-performing single mechanism. This counterintuitive result may be explained by potential redundancy or conflicts between the combined weighting signals. For instance, while each mechanism captures distinct aspects of object tracking (such as visual similarity, detection confidence, or motion dynamics), combining them without proper balancing can lead to diluted attention focus or competing influences within the attention map. As a result, the strengths of an effective individual mechanism, such as Dynamic Adjustment, may be partially suppressed when merged with less complementary cues. This observation emphasizes the need for careful integration design when combining multiple weighting sources in attention-based tracking systems.

The significance of this ablation study lies in quantifying the contributions of each weighting mechanism, allowing us to identify the most suitable weighting strategies for different tracking requirements. The results suggest that Dynamic Adjustment weighting plays a critical role in enhancing the overall robustness of the tracking system, while combinations of other weighting mechanisms also show potential advantages for specific metrics. This analysis provides guidance for selecting and designing more effective weighting strategies, laying the foundation for improved system robustness.

Table 4.3: Ablation of individual and combined weighting mechanisms on key tracking metrics.

Weighting Mechanism	**HOTA**	**DetA**	**AssA**	**MOTA**
Feature Matching Only	60.3	54.9	63.7	75.5
Confidence Only	61.1	55.5	62.3	76.8
Dynamic Adjustment Only	**61.8**	**56.2**	**64.5**	**77.7**
Feature Matching + Confidence	59.1	53.7	61.1	73.5
Feature Matching + Dynamic Adjustment	58.8	54.1	60.5	74.1
Confidence + Dynamic Adjustment	60.5	55.2	61.9	75.1

4.3.5.3 Sliding Window Size and Minimum Track Length

As demonstrated in Table 4.4, the configuration with a 22-frame sliding window size and a 5-frame minimum track length shows the best overall performance, with a HOTA of 63.1, DetA of 58.2, AssA of 68.8, and MOTA of 79.2. This configuration strikes a balance by leveraging enough temporal context through the sliding window while maintaining flexibility with shorter minimum track lengths.

Other configurations, such as a sliding window size of 10 frames with a minimum track length of 5, achieve higher MOTA values (80.1), but they perform lower in other metrics, particularly HOTA and AssA. This suggests that while these configurations may excel in minimizing missed detections and identity switches, they compromise overall tracking accuracy and object association quality.

The inclusion of multi-query and multi-weight computation aligns well with this experimental design. Multi-query strategies, combined with carefully tuned sliding window sizes, ensure that the model captures sufficient temporal information to maintain robust tracking, even in complex and dynamic environments. On the other hand, the multi-weight computation approach helps balance feature matching and confidence, providing adaptability to different sliding window sizes.

Table 4.4: Evaluation of different sliding window sizes and minimum track lengths on key tracking metrics. The combination of 22 sliding window size and 5 minimum track length shows the best overall performance.

Sliding Window Size	Minimum Track Length	HOTA	DetA	AssA	MOTA
10	5	57.3	53.9	60.1	80.1
	10	59.6	54.5	61.8	78.2
	15	60.9	56.1	63.2	77.1
15	5	60.5	55.8	64.1	78.8
	10	61.5	57.0	65.1	77.6
	15	62.1	57.5	66.1	76.9
22	**5**	**63.1**	**58.2**	**68.8**	**79.2**
	10	62.2	57.5	67.2	78.3
	15	62.8	58.8	66.4	78.9
32	5	61.5	58.6	66.7	81.2
	10	62.2	58.2	66.7	80.4
	15	62.5	57.8	67.3	79.5

4.3.5.4 Query Selection Strategy

The table presents a comparison between two different query selection strategies: Uniformly Spaced and Most Recent Frames. The purpose of this ablation study is to investigate the impact of these strategies on the key tracking metrics, including HOTA, DetA, AssA, and MOTA.

In the context of multi-queries, the Uniformly Spaced strategy significantly outperforms the Most Recent Frames strategy across all metrics, as shown in Table 4.5. The Uniformly Spaced approach achieves a HOTA of 63.1, DetA of 58.2, AssA of 68.8, and MOTA of 79.2. In contrast, the Most Recent Frames strategy falls short with a HOTA of 58.7, DetA of 55.3, AssA of 60.8, and MOTA of 74.5.

The performance difference can be attributed to the nature of the temporal information captured by each approach. The Uniformly Spaced strategy selects queries that cover a wider temporal window, allowing the model to capture more comprehensive temporal dynamics. This approach mitigates the risk of relying too heavily on recent frames, which may not reflect long-

term motion patterns and object behaviors. By spreading queries across the entire temporal range, the system ensures more stable and robust tracking, as indicated by the higher HOTA, DetA, AssA, and MOTA scores.

On the other hand, the Most Recent Frames strategy tends to focus on recent frames, which may fail to capture important historical context, leading to poorer performance. This is especially true in scenarios with occlusions, identity switches, or complex interactions, where long-term tracking information is crucial.

The design of this experiment is aligned with the concept of multi-queries and multi-weight computation. Both strategies leverage multiple frames to extract features and associate objects across time, but the Uniformly Spaced method shows better synergy with the multi-query and multi-weight approach. This is because multi-queries aim to broaden the temporal context, and using Uniformly Spaced queries aligns well with that objective, leading to improved tracking accuracy and robustness.

Table 4.5: Impact of query selection strategies on key tracking metrics.

Query Selection Strategy	**HOTA**	**DetA**	**AssA**	**MOTA**
Uniformly Spaced	**63.1**	**58.2**	**68.8**	**79.2**
Most Recent Frames	58.7	55.3	60.8	74.5

4.4 Conclusion

The Multi-Queries Tracking Transformer (MQTFormer) framework represents a significant advance in multi-object tracking, surpassing traditional GTR-based and other state-of-the-art models through three main innovations: multi-queries, multiple weights, and advanced tracking measures. These features make MQTFormer flexible, stable, and more accurate, especially in complex or dynamic environments.

One key feature of MQTFormer is its use of multi-queries, which allows the model to fuse information across multiple frames in a temporal sliding window, unlike traditional single-frame-based systems. This approach enhances tracking accuracy in scenarios with occlusions, rapid movements, and trajectory switches by capturing a deeper temporal context, enabling the model to interpolate object positions more accurately and reduce identity loss.

Experimental results show that a 22-frame sliding window and a minimum track length of 5 yield the best performance across metrics, with scores of HOTA 63.1, DetA 58.2, AssA 68.8, and MOTA 79.2. This configuration balances temporal context processing with adaptability in complex scenarios, particularly in situations where objects intersect or move rapidly around each other.

MQTFormer employs three weighting mechanisms—feature matching, confidence, and dynamic adjustment—to improve tracking reliability. Feature matching enhances differentiation between objects, confidence weighting reduces false positives, and dynamic adjustment allows the model to adapt to scene changes, such as varying speeds and brief occlusions. Together, these weighting strategies ensure that MQTFormer adapts effectively to each tracking challenge.

Tests also show that uniformly spaced queries, rather than focusing on the most recent frames, capture a broader temporal picture, which significantly improves tracking in situations with abrupt changes in object position or appearance.

Although MQTFormer achieves high association accuracy, with an AssA score of 68.8, its detection performance can still be refined. Future adjustments in detector strength or in the weighting calculations could help to enhance object classification accuracy and improve localization. In crowded scenes, such refinements would further strengthen MQTFormer's robustness. With its modular design and incorporation of richer temporal data and dynamic weighting strategies, MQTFormer provides a powerful solution for the demands of multi-object tracking in diverse real-world applications.

Chapter 5

Conclusions

In this dissertation, two new frameworks were introduced, representing a substantial advancement over the Global Tracking Transformer (GTR) baseline. These are the Global Prediction Attention-enhanced Transformer (GPAT) and the Multi-Queries Tracking Transformer (MQTFormer). By addressing several limitations of the original GTR and complementing each other's strengths, these frameworks improve performance and provide more stable accuracy for multi-object tracking systems in complex environments.

5.1 Summary of Contributions

GPAT enhances the GTR architecture by integrating a prediction module and a Convolutional Block Attention Module (CBAM). The prediction module alleviates the issue of missed detections by utilizing Kalman filtering, which incorporates temporal priors to predict object positions even when detections are missing or unreliable. This enables GPAT to maintain smooth trajectories under occlusion or adverse visual conditions. Meanwhile, CBAM improves feature extraction by focusing attention on the most relevant channels and spatial features, thus enhancing detection accuracy.

MQTFormer, based on the GTR framework, introduces a novel multi-query strategy combined with a dynamic multi-weight computation mechanism. This approach enables the model to process multiple temporal cues and adaptively weigh feature similarity, confidence, and motion cues. As a result, MQTFormer is more robust to occlusions, abrupt motion changes, and identity switches. These innovations collectively allow MQTFormer to outperform single-query models on major tracking metrics, including MOTA and HOTA.

5.2 Limitations

Despite their strengths, both GPAT and MQTFormer still suffer from certain limitations in complex or highly dynamic tracking scenarios.

For GPAT, one key limitation lies in its dependency on high-quality initial detections. If early detection results are significantly flawed, the prediction module may propagate errors through the trajectory, leading to identity switches or object loss. This effect becomes more pronounced in crowded scenes where overlapping and interaction among objects can lead to incorrect asso-

ciations. Furthermore, the CBAM module adds computational overhead, potentially limiting GPAT's applicability in real-time scenarios.

MQTFormer, while robust in modeling temporal contexts, has higher computational demands due to the multi-query input and the cost of maintaining memory across multiple frames. In scenarios requiring low-latency processing, such as autonomous driving or real-time surveillance, this could pose a bottleneck. In addition, MQTFormer's reliance on accurate similarity vectors makes it vulnerable in conditions with heavily occluded or overlapping objects, where detection noise can significantly degrade association performance.

5.3 Future Work

To further enhance the robustness, generalization, and efficiency of GPAT and MQTFormer, several promising directions for future work are outlined below. Each addresses a specific limitation or opportunity for improvement identified during experimentation.

5.3.1 Detection-Tracking Integration and Reliability

Although the proposed frameworks rely on external detectors, one potential improvement lies in exploring more tightly coupled detector-tracker architectures. Instead of passively accepting detection results, the tracker could actively incorporate detection confidence or uncertainty information to filter unreliable inputs. In addition, a joint detection-tracking model may help reduce latency and improve accuracy under occlusions or cluttered scenes. Such integration could be achieved using shared feature extractors or multitask learning frameworks that align the objectives of detection and tracking.

5.3.2 Learning-Based Multi-Weight Computation

The current implementation of the multi-weight module uses predefined strategies to combine feature similarity, detection confidence, and dynamic adjustment. A natural extension is to train a lightweight module, such as a multi-layer perceptron, to learn optimal weight combinations in a data-driven manner. The network could dynamically adjust the importance of different weights depending on object motion patterns, frame-level confidence, or scene context. This flexibility could improve performance in unseen or variable tracking conditions and reduce manual hyperparameter tuning.

5.3.3 Adaptive Query Frame Selection

In MQTFormer, query frames are currently selected in a fixed or uniform manner. A more adaptive query selection mechanism could prioritize frames with higher information content, such as those with clear object views or minimal motion blur. This can be achieved through attention-based scoring, motion change detection, or reinforcement learning strategies. Reducing redundant queries while focusing on informative ones could lower computational overhead and improve temporal modeling efficiency, especially in long video sequences.

5.3.4 Advanced Motion Prediction Models

GPAT relies on a Kalman filter to predict object motion, which may not capture long-term dependencies or complex dynamics. Future work may explore more expressive motion mod-

els, including recurrent neural networks (RNNs), graph neural networks (GNNs), or temporal transformers. These models could better handle irregular object motion and interactions between multiple targets, making tracking more robust under scenarios such as group movement, pedestrian crossing, or dynamic camera motion.

5.3.5 Cross-Dataset Evaluation and Generalization

While the current models are tested on MOT17, their generalization ability across datasets remains an open question. Evaluating GPAT and MQTFormer on more challenging benchmarks such as DanceTrack, TAO, or KITTI-MOT can provide insights into their limitations. Techniques such as domain adaptation, test-time fine-tuning, or self-supervised learning could be investigated to enhance performance across varied visual domains without requiring extensive re-training.

5.3.6 Real-Time Optimization and System Deployment

Finally, reducing computational latency is essential for real-world deployment. The attention modules in both GPAT and MQTFormer, while effective, incur additional processing cost. Future work can explore lightweight attention mechanisms, adaptive query memory updates, or frame skipping strategies to improve efficiency. Hardware-aware optimization, including GPU-specific acceleration and parallelization, may also make these frameworks more practical for real-time applications such as autonomous driving, robotics, or smart surveillance systems.

Bibliography

[1] P. Suetens, *Fundamentals of medical imaging*. Cambridge University Press, 2017 (cited on page 2).

[2] 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. 103 448, 2021 (cited on pages 3, 7, 16).

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015 (cited on page 3).

[4] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, 2007 (cited on page 3).

[5] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Springer, 2016, pp. 21–37 (cited on page 3).

[6] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 648–656 (cited on page 3).

[7] P. Sun, J. Cao, Y. Jiang, R. Zhang, P. Luo, and W. Liu, "Pixel-guided image generation: Recurrent inference for multi-object tracking," *arXiv preprint arXiv:2008.07087*, 2020 (cited on page 4).

[8] G. Welch, "An introduction to the kalman filter," 1995 (cited on pages 4, 8, 12, 16).

[9] F. Gustafsson, "Particle filter theory and practice with positioning applications," 7, vol. 25, IEEE, 2010, pp. 53–82 (cited on page 4).

[10] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020 (cited on page 4).

[11] DIY Photography, "Step up in privacy takeover? this cctv camera says it can detect coronavirus carriers," Accessed: 2024-11-10, 2020. [Online]. Available: https://www.diyphotography.net/step-up-in-privacy-takeover-this-cctv-camera-says-it-can-detect-coronavirus-carriers/ (cited on page 5).

[12] Ordinarily Rare, "From my favourite sport to sports medicine," Accessed: 2024-11-10, 2023. [Online]. Available: https://ordinarilyrare.com/from-my-favourite-sport-to-sports-medicine/ (cited on pages 4, 8).

[13] INF News, "Fitness: What you should know before starting a new workout routine," Accessed: 2024-11-10, 2023. [Online]. Available: https://inf.news/en/fitness/e4fa780010bb7049e4ec081478dfafe5.html (cited on page 6).

[14] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020 (cited on page 5).

[15] Center for Data Innovation, "Improving autonomous driving technology," Accessed: 2024-11-10, 2018. [Online]. Available: https://datainnovation.org/2018/06/improving-autonomous-driving-technology/ (cited on page 6).
[16] P. Zhu, L. Wen, D. Du, *et al.*, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7380–7399, 2021 (cited on page 5).
[17] A. B. Craig, "Understanding augmented reality: Concepts and applications," Newnes, 2013 (cited on page 5).
[18] J. Roecker and G. Phillis, "Suboptimal joint probabilistic data association," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 2, pp. 510–517, 1993 (cited on page 7).
[19] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004 (cited on page 7).
[20] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, 2018 (cited on page 7).
[21] X. Han, B. Jin, and Y. Wang, "Transctr: Transformer-based center track for multi-object tracking," *arXiv preprint arXiv:2202.04013*, 2022 (cited on page 7).
[22] M. Piccardi, "Background subtraction techniques: A review," in *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, IEEE, vol. 4, 2004, pp. 3099–3104 (cited on page 7).
[23] I. Papakis, A. Sarkar, and A. Karpatne, "A graph convolutional neural network based approach for traffic monitoring using augmented detections with optical flow," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 2980–2986 (cited on page 7).
[24] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, IEEE, 1999, pp. 1197–1203 (cited on page 7).
[25] Z. Pang, Z. Li, and N. Wang, "Simpletrack: Understanding and rethinking 3d multi-object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 680–696 (cited on page 8).
[26] I. Guyon and A. Elisseeff, "An introduction to feature extraction," in *Feature extraction: Foundations and Applications*, Springer, 2006, pp. 1–25 (cited on page 8).
[27] I. Teeti, S. Khan, A. Shahbaz, A. Bradley, F. Cuzzolin, and L. De Raedt, "Vision-based intention and trajectory prediction in autonomous vehicles: A survey," in *IJCAI*, 2022, pp. 5630–5637 (cited on page 9).
[28] S. Grossberg, "Recurrent neural networks," *Scholarpedia*, vol. 8, no. 2, p. 1888, 2013 (cited on page 9).
[29] X. Zhou, T. Yin, V. Koltun, and P. Krähenbühl, "Global tracking transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8771–8780 (cited on pages 9, 10, 12, 28, 43, 44, 61).
[30] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026 (cited on pages 9, 21).
[31] Y. Wang, X. Zhang, and W. Wang, "Nct: Neural collaborative tracking," *arXiv preprint arXiv:2104.00380*, 2021 (cited on pages 9, 61).
[32] Y. Cai and G. Medioni, "Exploring context information for inter-camera multiple target tracking," in *IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2014, pp. 761–768 (cited on page 10).

[33] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4660–4669 (cited on page 10).

[34] P. Chu, H. Fan, C. C. Tan, and H. Ling, "Online multi-object tracking with instance-aware tracker and dynamic model refreshment," in *Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2019, pp. 161–170 (cited on page 10).

[35] P. Sun, J. Cao, Y. Jiang, *et al.*, "Transtrack: Multiple object tracking with transformer," *arXiv preprint arXiv:2012.15460*, 2020 (cited on pages 10, 25, 43, 61).

[36] F. Zeng, B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei, "Motr: End-to-end multiple-object tracking with transformer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 659–675 (cited on pages 10, 25, 26, 42, 43, 61).

[37] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017 (cited on pages 11, 20, 24).

[38] L. Zheng, M. Tang, Y. Chen, G. Zhu, J. Wang, and H. Lu, "Improving multiple object tracking with single object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 2453–2462 (cited on page 11).

[39] I. Daramouskas and N. Patrinopoulou, "Camera-based local and global target detection, tracking, and localization techniques for uavs," *MDPI Machines*, vol. 11, no. 2, p. 315, 2023. [Online]. Available: https://doi.org/10.3390/machines11020315 (cited on page 11).

[40] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19 (cited on page 12).

[41] H. Zhao, Y. Zhang, S. Liu, *et al.*, "Psanet: Point-wise spatial attention network for scene parsing," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 267–283 (cited on page 13).

[42] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3464–3468 (cited on pages 16, 61).

[43] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017, pp. 3645–3649 (cited on pages 17, 61).

[44] P. Liao, F. Yang, D. Wu, J. Yu, W. Zhao, and B. Liu, "Fasttracktr: Towards fast multi-object tracking with transformers," *arXiv preprint arXiv:2411.15811*, 2024 (cited on page 17).

[45] Y. Zhang, P. Sun, Y. Jiang, *et al.*, "Bytetrack: Multi-object tracking by associating every detection box," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 1–21 (cited on pages 18, 41, 42, 60, 61).

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016 (cited on pages 22–24).

[47] ResearchGate contributors, "Structure and processing method of visual transformer," Accessed: 2024-11-11, 2023. [Online]. Available: https://www.researchgate.net/figure/Structure-and-Processing-Method-of-Visual-Transformer_fig4_371428193 (cited on page 22).

[48] E. Cambria and B. White, "Jumping nlp curves: A review of natural language processing research," *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, 2014 (cited on page 24).

[49] AI-SCHOLAR, *Trackformer*, Accessed: 2024-11-14, 2021. [Online]. Available: https://ai-scholar.tech/zh/articles/transformer/track_former (cited on page 24).

[50] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8844–8854 (cited on pages 24, 42, 43, 61).

[51] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," 4, vol. 5, MDPI, 2023, pp. 1680–1716 (cited on page 24).

[52] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587584 (cited on page 25).

[53] "Https://mint-lab.github.io/mint-lab/papers/seo24$_i cros_m ot.pdf$," Accessed: 2024-11-11, 2024. [Online]. Available: https://mint-lab.github.io/mint-lab/papers/Seo24_icros_mot.pdf (cited on pages 25, 26).

[54] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 213–229 (cited on pages 25, 28, 29).

[55] J. Yan, H. Wang, M. Yan, W. Diao, X. Sun, and H. Li, "Iou-adaptive deformable r-cnn: Make full use of iou for multi-class object detection in remote sensing imagery," 3, vol. 11, MDPI, 2019, p. 286 (cited on page 26).

[56] P. Dendorfer, A. Osep, A. Milan, *et al.*, "Motchallenge: A benchmark for single-camera multiple target tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, vol. 129, 2021, pp. 845–881 (cited on page 26).

[57] arXiv, "Arxiv:2203.13250," Accessed: 2024-11-11, 2022. arXiv: 2203.13250. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2203.13250 (cited on page 29).

[58] A. Roshan Zamir, A. Dehghan, and M. Shah, "Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs," in *Proceedings of the IEEE/CVF Conference on Computer Vision (ECCV) 2012, Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*, Springer, 2012, pp. 343–356 (cited on pages 31, 40, 41).

[59] S. Pal, W. Luo, *et al.*, "Multiple object tracking in deep learning approaches: A survey," *MDPI Electronics*, vol. 12, no. 10, pp. 6887–6901, 2023. [Online]. Available: https://doi.org/10.3390/s23156887 (cited on page 31).

[60] M. Adžemović, P. Tadić, A. Petrović, and M. Nikolić, "Beyond kalman filters: Deep learning-based filters for improved object tracking," *Machine Vision and Applications*, vol. 36, no. 1, pp. 1–22, 2025 (cited on page 35).

[61] B. Thomee, D. A. Shamma, G. Friedland, *et al.*, "Yfcc100m: The new data in multimedia research," *Communications of the ACM*, vol. 59, no. 2, pp. 64–73, 2016 (cited on page 39).

[62] S. Li, Y. Kong, and H. Rezatofighi, "Learning of global objective for network flow in multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8855–8865 (cited on pages 40, 41).

[63] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022 (cited on pages 41, 42, 61).

[64] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 941–951 (cited on pages 41, 42, 61).

[65] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 474–490 (cited on pages 41, 42, 61).

[66] Y. Wang, K. Kitani, and X. Weng, "Joint object detection and multi-object tracking with graph neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 708–13 715 (cited on pages 41, 42, 61).

[67] C. Wang, Y. Zhang, X. Wang, W. Zeng, and W. Liu, "Corrtracker: Towards reliable end-to-end multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 393–12 402 (cited on pages 41, 42, 61).

[68] 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, pp. 3069–3087, 2021 (cited on pages 41, 42, 61).

[69] S. Zhang and L. Wang, "Maatrack: Multi-object tracking with appearance and affinity model," *arXiv preprint arXiv:2104.00380*, 2021 (cited on pages 41, 42, 60, 61).

[70] J. Wu, J. Cao, L. Song, Y. Wang, M. Yang, and J. Yuan, "Track to detect and segment: An online multi-object tracker," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12 352–12 361 (cited on pages 42, 61).

[71] J. Pang, L. Qiu, X. Li, *et al.*, "Quasi-dense similarity learning for multiple object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 164–173 (cited on pages 42, 44, 61).

[72] B. Yan, Y. Jiang, P. Sun, *et al.*, "Towards grand unification of object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2022, pp. 733–751 (cited on pages 42, 61).

[73] S. Wang, H. Sheng, Y. Zhang, Y. Wu, and Z. Xiong, "A general recurrent tracking framework without real data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 219–13 228 (cited on page 42).

[74] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 9686–9696 (cited on pages 42, 61).

[75] J. Peng, C. Wang, F. Wan, *et al.*, "Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 145–161 (cited on pages 42, 61).

[76] S. Gao, Z. Wang, and L. Wang, "P3aformer: Tracking objects as pixel-wise distributions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022 (cited on pages 42, 43, 61).

[77] F. Zeng, B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei, "Motrv2: Bootstrapping end-to-end multi-object tracking with a pretrained object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023 (cited on pages 42, 43, 61).

[78] Y. Xu, J. Cao, Z. Zhang, H. Zhang, H. Hu, and J. Wang, "Transcenter: Transformers with dense queries for multiple-object tracking," *arXiv preprint arXiv:2103.15145*, 2021 (cited on pages 42, 43, 61).

[79] H. Liang, Y. Zhang, and H. Zhang, "Cstrack: Convolutional siamese tracker for object tracking," *arXiv preprint arXiv:2010.12138*, 2020 (cited on pages 42, 43, 61).

[80] Y. Zhang, P. Sun, Y. Jiang, D. Yu, P. Luo, and X. Wang, "Fufet: Fully unsupervised few-shot object tracking," *arXiv preprint arXiv:2103.17103*, 2021 (cited on page 43).

[81] P. Chu and H. Ling, "Transmot: Spatial-temporal graph transformer for multiple object tracking," *arXiv preprint arXiv:2104.00194*, 2021 (cited on pages 43, 61).

[82] D. Du, L. Zhu, S. Zhang, Y. He, X. Li, G. Ding, *et al.*, "1st place solution to eccv-tao-2020: Detect and represent any object for tracking," *arXiv preprint arXiv:2101.08040*, 2021 (cited on page 44).

[83] D. Stadler and J. Beyerer, "Modelling ambiguous assignments for multi-person tracking in crowds," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 133–142 (cited on page 44).

[84] Y. Chen, Z. Li, and A. Song, "Multi-query person search with transformers," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2024, pp. 116–128 (cited on page 54).

[85] P. Tokmakov, X. Wang, and M. Hebert, "Permatrack: Tracking with object permanence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1433–1442 (cited on page 61).

[86] Y. Xu, Y. Ban, A. Del Giorno, H. Bilen, and S. Gong, "How to train your deep multi-object tracker," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6787–6796 (cited on page 61).

[87] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 107–122 (cited on page 61).

[88] S. Zhang, B. Jiang, P. Sun, and P. Luo, "Memot: Multi-object tracking with memory-enhanced transformer," *arXiv preprint arXiv:2206.05896*, 2022 (cited on page 61).