Ego-motion Aware Multi-object Tracking: An application for a ROS-based Framework

by

Navid Mahdian
B.Sc., Shahid Beheshti University, 2021

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Ego-motion Aware Multi-object Tracking:

An application for a ROS-based Framework

by

Navid Mahdian

B.Sc., Shahid Beheshti University, 2021

Supervisory Committee

Dr. H. Najjaran, Supervisor

(Department of Electrical and Computer Engineering)

Dr. D. Capson, Departmental Member

(Department of Electrical and Computer Engineering)

ABSTRACT

Multi-object tracking (MOT) is a critical step for safe and reliable operations of robotics and autonomous systems in dynamic and cluttered environments which are inherent to real-world applications. This thesis introduces a novel MOT framework designed for the Robot Operating System (ROS), serving as a versatile foundation for the implementation, testing, and evaluation of various MOT algorithms within the realm of robotics and autonomous systems. A key hallmark of this framework is its integration with both simulated environments and real-world robotic platforms, facilitating exhaustive testing and refinement of MOT algorithms under a broad spectrum of conditions. Moreover, this comprehensive framework is distinguished by its capability for automatic ground truth generation enables detailed and systematic evaluation across numerous operational scenarios.

Within this framework, the Ego-motion Aware Target Prediction module (EMAP) is developed, which significantly enhances the performance of detection-based multi-object tracking algorithms. By integrating camera motion and depth information, EMAP effectively decouples camera movement from object trajectories, thereby minimizing tracking disturbances caused by the ego-motion of the observer. EMAP's effectiveness is rigorously demonstrated through evaluations using the KITTI dataset and a custom-generated dataset in the CARLA autonomous driving simulator, showing substantial improvements in tracking performance, especially in scenarios marked by significant camera motion or the absence of detections.

Additionally, this thesis presents a self-supervised multi-object tracking algorithm that incorporates an adaptive track-matching mechanism. This mechanism leverages unlabeled data to refine tracking precision and efficiency, reducing the dependency on extensive manual annotations and thereby enabling more scalable and generalizable tracking applications. Together, these contributions significantly advance the field

of autonomous systems and robotics by paving the way for more robust and reliable multi-object tracking technologies.

Table of Contents

List of Tables

List of Figures

Acronyms

AssA Association Accuracy.

AssR Association Recall.

AW Adaptive Weighting.

CMC Camera Motion Compensation.

CNN Convolutional Neural Network.

DA Dynamic Appearance.

DBT Detection-Based Tracking.

ECC Enhanced Correlation Coefficient.

EMAP Ego-Motion Aware Target Prediction.

FN False Negatives.

FP False Positives.

FPS Frames Per Second.

HOTA Higher Order Tracking Accuracy.

IDF1 ID F1 Score.

IoU Intersection over Union.

KF Kalman Filter.

MOT Multi-object tracking.

MOTA Multi-Object Tracking Accuracy.

OC-SORT Observation-Centric SORT.

OCM Observation-Centric Momentum.

ORU Observation-centric Re-Update.

Re-ID Re-Identification.

RNN Recurrent Neural Network.

ROS Robot Operating System.

SORT Simple Online and Realtime Tracking.

ACKNOWLEDGEMENTS

I would like to thank:

- I am deeply thankful to my family and friends for their unwavering support and encouragement, Dr. Najjaran for his invaluable guidance and mentorship, and my teammate Mohammad Jani, whose collaboration was instrumental in the success of this project. Special appreciation goes to Amir Soufi, a friend who provided mentor-like support and guidance throughout this journey.

- A special thanks to Sentire Co. for their support and resources, which enabled me to pursuit and complete my research.

Chapter 1

Introduction

1.1 Motivation

Multi-object tracking (MOT) is a fundamental task in computer vision, underpinning various applications that significantly impact our daily lives and numerous industries. Primarily, this task involves identifying and tracking multiple objects across a sequence of images, which presents challenges such as occlusions, changes in object appearance, and varying object velocities. This technology is pivotal in many contexts, such as autonomous vehicle navigation, where precise tracking of other vehicles on the road and pedestrians is necessary for the vehicle to operate safely. It also plays a central role in surveillance systems, enabling the surveillance of pedestrians in densely populated areas for security purposes. Additionally, multi-object tracking is instrumental in sports analysis as it provides detailed tracking of players and objects, which offers rich and valuable insights into game dynamics. Furthermore, in robotics applications, it facilitates the interaction between robots and their environment, allowing for object avoidance, navigation, and environment mapping.

Ego-motion awareness is a crucial aspect of enhancing the performance of multi-

object tracking algorithms, mainly because it enables tracking systems to comprehend and account for their own motion within the environment. By incorporating ego-motion awareness, trackers can differentiate between the motion of objects and the motion of themselves. Such integration increases tracking accuracy and robustness; for example, in autonomous vehicle navigation, understating the motion of the vehicle allows for more precise tracking of surrounding objects by compensating for the movements of the ego vehicle. This leads to safer navigation decisions as the system can accurately predict the trajectories of other objects.

Having ego-motion awareness as a modular component within multi-object tracking systems offers multiple benefits. Firstly, it can be integrated with various tracking algorithms and enhance performance without requiring extensive modifications. Secondly, it allows the ego-motion-aware prediction component to be modified and improved independently from the tracking system. Lastly, this isolation helps developers focus on optimizing this aspect for different sensor setups and application needs, further enhancing the versatility and applicability of multi-object tracking algorithms in different domains.

Another approach to multi-object tracking is using self-supervision to use unlabeled data to improve tracking accuracy and efficiency. This method alleviates the intensive labor and resource demands typically associated with annotating large datasets, enabling algorithms to learn and adapt from the data without explicit manual guidance. By employing an adaptive mechanism tuning itself to various conditions and datasets, self-supervised multi-object tracking achieves remarkable flexibility and generalizability. This enhances the system's ability to handle diverse tracking scenarios and reduces the need to fine-tune parameters for different applications constantly. Consequently, self-supervised multi-object tracking paves the way for more scalable, adaptable, and accessible tracking technologies, making it a valuable advancement in

pursuing more robust autonomous vision systems.

To address the multifaceted challenges of multi-object tracking, developing a comprehensive framework that can seamlessly ingest sensor data from simulated environments and real-world robotic platforms is essential. Such a framework significantly enhances the implementation, testing, and evaluation of multi-object tracking algorithms by providing a standardized platform for algorithm application. Integrating advanced simulation environments capable of automatically generating ground truth data across diverse scenarios—including various maps, weather conditions, and dynamic objects—offers invaluable benefits. It enables rigorous and extensive testing of multi-object tracking algorithms under various conditions, ensuring their robustness and adaptability. Such a framework not only streamlines the development process but also propels advancements in the field by facilitating the fine-tuning of algorithms to meet the demands of real-world application scenarios, marking a significant stride towards achieving more autonomous and reliable tracking systems.

1.2 Contributions

The primary contributions of this thesis can be summarized as follows:

1. **Multi-object Tracking Framework:** A comprehensive MOT framework is introduced, characterized by a structured design flow that optimizes the development and debugging process of MOT algorithms. This framework is uniquely designed to facilitate algorithm testing with diverse simulation datasets, featuring automatically generated ground truth data, and enables seamless integration with real-world robotic systems. Through this structured approach, the adaptability and effectiveness of MOT algorithms are significantly improved across both simulated environments and practical applications, addressing crit-

ical challenges in the field.

2. **Self-supervised Multi-object Tracking Algorithm with Adaptive Track-Matching:** Our study pioneers a fully self-supervised yet highly generalizable methodology for multi-object tracking. At the heart of our approach lies an adaptive track-matching threshold, ingeniously crafted from a reliable self-supervisory signal, allowing the algorithm unprecedented adaptability across various datasets. This methodology eschews the traditional pursuit of peak performance on benchmark datasets like MOT20 to develop a versatile solution capable of tackling the broad spectrum of challenges inherent to MOT. Our findings underscore the method's efficacy, showcasing its ability to yield results comparable to those of both supervised and unsupervised counterparts without necessitating the arduous task of hyperparameter re-tuning for new scenarios. Additionally, augmentation of the approach with a post-processing module is envisaged, designed to merge potentially discarded tracks, thereby elevating the overall performance of the algorithm and further cementing its utility as a robust, adaptable solution for challenges in multi-object tracking.

3. **Ego-Motion Aware Target Prediction Module:** Addressing the challenges posed by conventional prediction methods in Detection-based Multi-object Tracking (DBT), particularly in scenarios characterized by significant camera motion or the absence of detections, the Ego-Motion Aware Target Prediction (EMAP) module is introduced. This innovative approach leverages a novel Kalman Filter (KF)-based prediction module that effectively integrates camera motion and depth information with object motion models. By decoupling camera rotational and translational velocity from object trajectories, our method substantially reduces the disturbances caused by camera motion, thereby enhancing the reliability of the object motion model. The EMAP mod-

ule's integration with state-of-the-art base MOT algorithms—OC-SORT, Deep OC-SORT, ByteTrack, and BoT-SORT—demonstrates its efficacy. Specifically, our evaluation on the KITTI MOT dataset reveals a dramatic reduction in the number of identity switches (IDSW), with OC-SORT and Deep OC-SORT experiencing drops of 73% and 21% respectively. Moreover, the integration of EMAP results in an improvement of more than 5% in the HOTA metric for these algorithms, showcasing its potential to enhance MOT performance in autonomous driving and similar applications significantly.

1.3 Thesis Outline

This thesis is structured into several chapters, each dedicated to different facets of multi-object tracking. An outline of these chapters is provided below:

- **Chapter 2: The MOT Framework and Literature Review**

  Focuses on the comprehensive MOT framework designed for testing, evaluation, and deployment of MOT algorithms. It provides insights into the structured design flow, integration with simulated environments and real-world robotic platforms, and discusses the advantages of this comprehensive approach to MOT research. The literature review is also included in this chapter.

- **Chapter 3: Self-supervised Multi-object Tracking Algorithm with Adaptive Track-Matching**

  Explores the development of a fully self-supervised, highly generalizable MOT algorithm that utilizes adaptive track-matching. This chapter delves into the methodological foundations of using a self-supervisory signal for adaptive thresholding and discusses its impact on flexibility and generalizability across datasets.

- **Chapter 4: Ego-motion Aware Target Prediction Module for Multi-object Tracking**

  Introduces the Ego-Motion Aware Target Prediction (EMAP) module, detailing the methodology behind the integration of camera motion and depth information with object motion models. It discusses the challenges faced by conventional prediction methods in autonomous driving scenarios and proposes a novel approach to improve MOT performance.

- **Chapter 5: Conclusions and Future Work**

  Concludes the thesis with a summary of the contributions and the significance of the research within the context of robotics and autonomous systems. It also addresses the limitations encountered during the study and outlines potential directions for future research.

Chapter 2

The MOT Framework and Literature Review

In this chapter, the multi-object tracking (MOT) framework is explained, and a structured design flow for the development of MOT algorithms is introduced. The meticulously organized steps of our design flow are designed to optimize the debugging process for evaluating the efficacy of MOT algorithms. This approach facilitates the testing of algorithms using diverse, specially tailored simulation datasets equipped with automatically generated ground truth data. Furthermore, it allows for seamless integration with real-world robotic systems, necessitating minimal modifications. This comprehensive strategy enhances the adaptability and effectiveness of MOT algorithms in both simulated and practical environments.

2.1 Background

2.1.1 Gazebo Simulator

Gazebo serves as an open-source robotic simulator extensively employed across the developmental lifecycle of robotic systems, encompassing design, testing, and validation phases within simulated environments. Notably, one of Gazebo's primary strengths lies in its capacity to generate intricate robotic scenarios with a heightened level of realism. Moreover, its seamless integration with the Robot Operating System (ROS) offers developers a conducive environment for algorithmic refinement and implementation.

In the context of multi-object tracking, Gazebo's simulation capabilities allow for the detailed modeling of robot sensors, such as LiDAR, radar, and cameras, which are critical for object detection and tracking. By accurately simulating the characteristics and limitations of these sensors, researchers can develop and refine tracking algorithms that are robust to real-world operational challenges. Additionally, Gazebo's support for dynamic interaction between objects and the environment enables the investigation of the performance of tracking algorithms in complex scenarios, including occlusions, varying object speeds, and densities. Figure 2.1 shows an example of a bookstore that is simulated in the Gazebo environment.

2.1.2 CARLA Self-driving Car Simulator

Similar to Gazebo, CARLA [3] stands out as an open-source simulator primarily designed for autonomous driving applications. CARLA adopts the ASAM OpenDRIVE [4] standard to describe its road networks and urban landscapes. This standard, provided by ASAM, furnishes a comprehensive description of road networks, facilitating

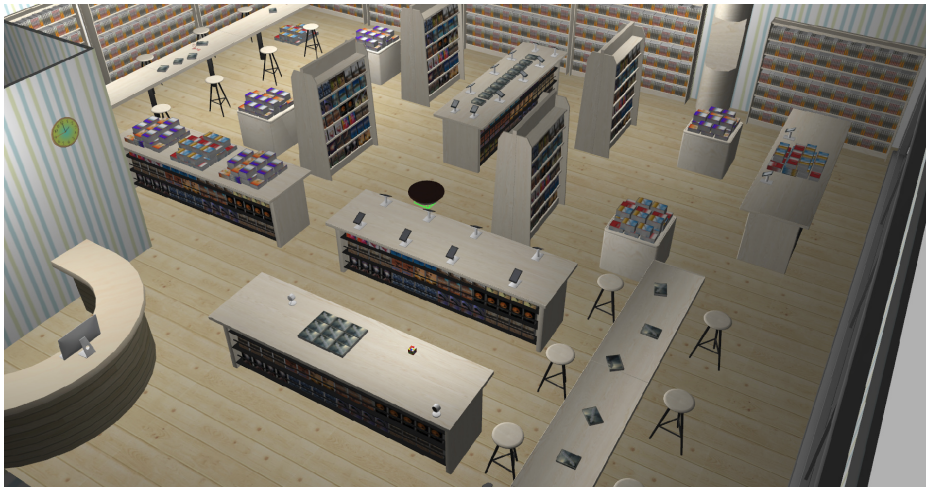

Figure 2.1: Example of a bookstore environment in the Gazebo simulator [1].

the validation of advanced driver-assistance systems (ADAS) and ensuring seamless interoperability among various simulators.

CARLA operates on a server-client architecture paradigm, where the server orchestrates the simulation and renders the virtual environment. Conversely, the client, implemented in Python, manages the interaction between agents and the server. Communication between the server and agents occurs through socket connections, enabling efficient data exchange and synchronization.

Environments that are generated by CARLA contain 3D models of static objects as well as dynamic objects such as ground vehicles and pedestrians. More specifically, the models include 40 different buildings, 16 types of vehicle models as well as 50 pedestrian models.

Maps

CARLA provides eight different types of maps in its ecosystem. Each of these maps can be configured as layered or non-layered, the difference of which is explained below.

In non-layered maps, all of the layers are present in the simulation and cannot be removed. On the other hand, in newer versions of the simulator, CARLA has

introduced the concept of layered maps inside which the user can choose which layer is presented in the environment.

CARLA vs Gazebo

While Gazebo is versatile in its application, providing support for a wide range of robotics projects, CARLA is specifically designed for autonomous driving applications. CARLA particularly focuses on simulating complex urban driving scenarios, including the behavior of vehicles and pedestrians, which is critical for autonomous driving testing. Regarding sensor simulation, Gazebo provides a wide range of sensors such as LiDAR, monocular, stereo, depth camera, infrared, ultrasonic, and GPS sensors, as well as IMU, Force/Torque, and contact sensors. In CARLA, the selection of sensors is tailored specifically for autonomous vehicle applications. For example along with the RGB and depth cameras, sementic segmentation, optical flow, instance segmentation, and event cameras are also provides in CARLA. In terms of detector sensors, collision, lane invasion and obstacle detector sensors are provided in this simulator. table 2.1 provides a detailed comparison between Gazebo and CARLA in terms of their sensors.

2.1.3 Robot Operating System

The Robot Operating System (ROS) [5] is an open-source middleware suite designed to support software development for robotic applications. It provides a flexible framework that enables the integration and orchestration of software components in complex robotic systems. Originating as a project at Stanford University in 2007, ROS has become a pivotal tool in robotics research and development, offering a vast repository of tools, libraries, and conventions that facilitate the rapid development and deployment of robotic functionalities. Figure 2.2 shows an overview of how topics

Table 2.1: Comparison of Sensor Availability in Gazebo and CARLA Simulators.

Sensor Type	**Gazebo**	**CARLA**
RGB Camera	✓	✓
Depth Sensor	✓	✓
Lidar	✓	✓
GPS/GNSS	✓	✓
IMU	✓	✓
Collision Detection	✓	✓
Semantic Segmentation	x	✓
Instance Segmentation	x	✓
Radar	x	✓
Altimeter	✓	x
Wind Sensor	✓	x
Thermal Camera	✓	x
Sonar	✓	x
Magnetometer	✓	x
Lane Invasion Detection	x	✓
Obstacle Detection	x	✓
Optical Flow	x	✓
DVS (Dynamic Vision Sensor)	x	✓
RSS (Responsibility-Sensitive Safety)	x	✓

and nodes are connected to the ROS Master.

Nodes

ROS nodes are the basic execution units of a ROS-based system. Each node is a process that performs computation and interacts with other nodes over the ROS network. Nodes can be producers or consumers of information, such as sensor data processors, actuators, controllers, or state estimators, allowing for a modular and distributed approach to system design. This architecture supports the principles of reusability and scalability, crucial for the development of complex robotic applications.

Topics and Messages

Communication among nodes in ROS is primarily handled through topics and messages. Topics are named channels that nodes use to exchange data, operating on a publish-subscribe model. Messages, the structured data sent over these channels, are defined using simple language-independent data types, facilitating interoperability and flexibility. This mechanism allows nodes to communicate asynchronously, sharing information such as sensor readings or control commands without requiring direct knowledge of each other's existence or operation.

Services

For synchronous communication, ROS introduces services, which operate on a request-response mechanism. Services are characterized by a set of message structures, comprising one for the request and another for the response. Nodes can offer services to perform specific operations and respond to requests from other nodes. This model is particularly useful for performing operations that require immediate feedback or action, enhancing the system's capability to handle dynamic and real-time requirements.

2.1.4 ROS Master

The ROS Master plays a central role in the ROS architecture, acting as a name service for nodes. It enables nodes to discover each other and establish communication channels for exchanging messages or invoking services. The ROS Master maintains a registry of topics, services, and active nodes, facilitating the dynamic configuration and coordination of the system's components.

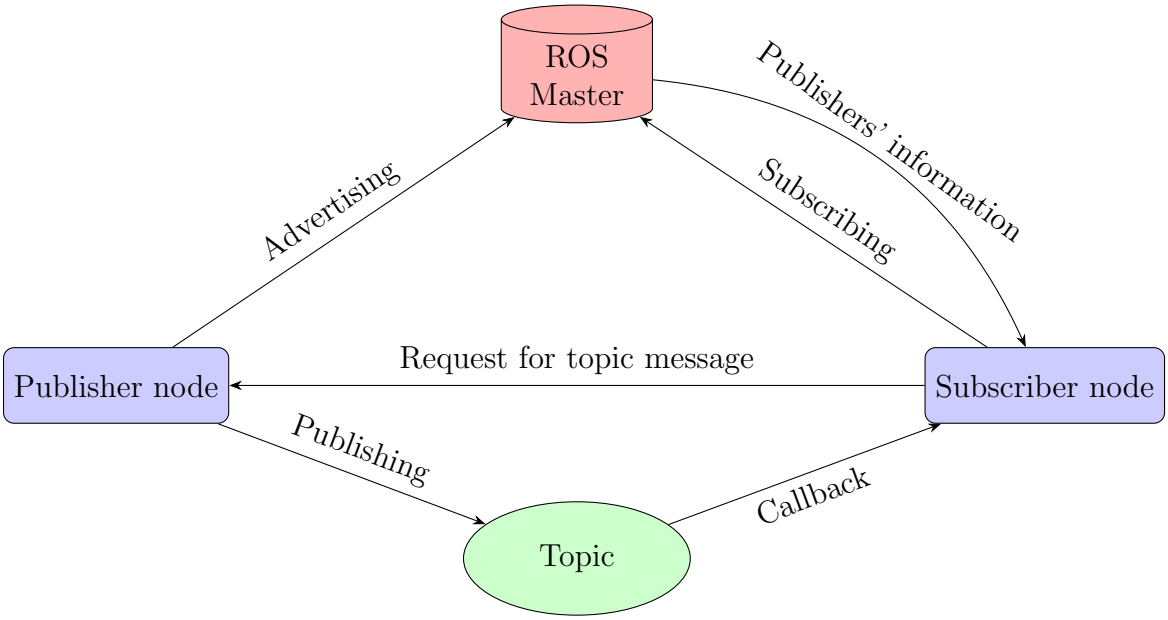

Figure 2.2: Overview of ROS Communication

Message Passing

Message passing in ROS is designed to be both efficient and flexible, supporting diverse communication patterns through topics and services. This framework allows for the decoupled operation of system components, enabling distributed processing and enhancing the system's resilience and scalability. The message-passing interface is a cornerstone of ROS, providing the foundation for the complex interactions that underpin robotic functionalities and behaviors.

2.1.5 The Jackal UGV from Clearpath Robotics

The Jackal Unmanned Ground Vehicle (UGV) [6], developed by Clearpath Robotics, represents a cornerstone in robotics research platforms, combining mobility, versatility, and advanced computational capabilities. Engineered for adaptability across various terrains, the Jackal UGV is an all-terrain, fully programmable platform that facilitates comprehensive research and development efforts, especially within the do-

main of robotics and autonomous systems.

Equipped with a robust suite of sensors and an onboard computer, the Jackal UGV is designed to support complex applications ranging from autonomous navigation to multi-object tracking. Its compatibility with the Robot Operating System (ROS) empowers researchers to leverage an extensive array of tools and libraries, accelerating the development and deployment of new algorithms and functionalities.The platform's integration with various sensors, including LiDAR, cameras, and GPS, enables the capture and analysis of environmental data essential for tracking multiple objects in diverse and challenging scenarios.

Moreover, the Jackal UGV's computational resources facilitate on-the-go data processing, significantly enhancing the efficiency and responsiveness of tracking systems. This capability is critical for developing applications where real-time analysis and decision-making are paramount. Through its advanced features and ROS integration, the Jackal UGV exemplifies a pivotal tool for advancing research in multi-object tracking, offering both the mobility and computational power necessary to tackle the complexities of real-world environments. Figure 2.3 shows our Jackal Clearpath robot I have used to deploy my multi-object tracking algorithms.

2.2 Related Work

2.2.1 Single Object Tracking

The problem of tracking a single object can be defined as following a bounding box of an object that has been marked in the first frame of a video sequence for the duration of the sequence. In classic object tracking, optical flow is used to track the location of the pixels with the assumption that the optical flow is constant in the vicinity of a single object. Lucas-Kankade [7] is one of the most common methods, used for

Figure 2.3: The Jackal UGV equipped with a Velodyne VLP-16 LiDAR, two RGB-D cameras, wheel encoders, and an IMU, demonstrating its comprehensive sensor suite for advanced robotic applications.

optical flow estimation. In this method, the assumption of constant pixel flow in the neighborhood is written for every pixel and the goal is to minimize the sum of the squares of the errors produced in each of the equations.

Given an image $I(x, y, t)$, where x and y represent spatial coordinates and t represents time, the brightness constancy assumption can be expressed as:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.1}$$

Using the Taylor series expansion and ignoring higher-order terms, the following is obtained:

$$I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = I(x, y, t) \tag{2.2}$$

This simplifies to the optical flow constraint equation:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y = -\frac{\partial I}{\partial t} \tag{2.3}$$

where $v_x = \frac{dx}{dt}$ and $v_y = \frac{dy}{dt}$ are the components of optical flow in the x and y directions,

respectively. This can be written in matrix form as:

$$\begin{bmatrix} I_x \\ I_y \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \end{bmatrix} = -I_t \tag{2.4}$$

where $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, and $I_t = \frac{\partial I}{\partial t}$ are the partial derivatives of the image intensity with respect to x, y, and t, respectively.

Another approach to object tracking is using correlation filters [8–15]. After the selection of the object bounding box in the first frame, the object will be tracked by matching the template with the regions in the next image frame, and the most similar region will be selected as the matched region. The template will then be updated based on the new tracked bounding box in the new frame. Correlation-based trackers are computationally efficient and can handle various object appearances and motions. However, they may suffer from challenges such as occlusions, scale changes, and appearance variations.

Deep-learning-based methods: Deep-learning methods have been used in object tracking both for feature extraction and position prediction. The main deep learning architectures used for object tracking are CNN, Autoencoder, and RNN.

Algorithms that are based on RNNs utilize the power of recurrent neural networks in sequence-related tasks. In [16], the authors integrate RNNs into the tracking process to explicitly model the temporal dependencies across video frames. Having the tracking problem regarded as a sequence learning problem, RNN can predict the future state of the object considering the history of the object's position. The main power of this method is to handle occlusions and long-term dependencies as RNNs are able to maintain information about the object even if the object is temporarily lost. [17] combines Siamese networks with LSTM units in order to capture the spatial and temporal features of the target object more effectively. The siamese network

takes pairs of frames as input and extracts features from them, and the LSTM models the temporal relationship between the extracted features. LSTM's ability to remember and utilize historical appearance information results in the enhancement of the tracker's robustness to occlusions.

2.2.2 Detection-based Multi-object Tracking

Most of the leading MOT approaches [18–23] follow the detection-based tracking schema. Providing that the object detectors have improved significantly during the recent years [24–26], these methods rely on the detections as their input and focus on finding the optimal association between the per-frame detection boxes. DBT algorithms are also referred to as graph model-based tracking as the problem can be redefined as a graph optimization problem. In this category of solutions, the MOT problem is divided into three separate tasks: detection, prediction, and association. While this provides the advantage of modularity, the overhead of each task is also considerable. Moreover, the overall performance of the approach is significantly influenced by the detection module. As a result, when the detection module fails to detect objects, the prediction module is responsible for predicting the future location of the missed object until the object is detected again by the object detector. In the prediction part, the goal is to find the best prediction of the object's location in the next frame based on the previous trajectory of the track. Many of the dominant methods [20–22] use Kalman Filter as their prediction model. In [20], the authors proposed OC-SORT, a method to fix the accumulated noise during occlusion by generating a virtual trajectory for the occlusion period while keeping the simple Kalman filter model.

Some methods [27, 28] view the prediction and association phase as a joint problem and try to solve the problem using RNNs. The authors of [27] present a structure of

RNNs that uses motion, appearance and interaction features of the objects to perform prediction and association based on the cues over a temporal window.

Some works [22, 29, 30] concentrated on enriching the association phase by utilizing the appearance features. In [22] the authors propose a deep association metric by training a CNN on MARS [31] person reidentification dataset. [30] introduced an innovative unsupervised re-identification learning module, eliminating the need for object labels. Additionally, they incorporated an occlusion estimation module to predict occluded areas, achieving results on par with supervised approaches.

2.2.3 Motion Modeling

SORT-based tracking approaches leverage a constant velocity Kalman Filter model to represent and predict the motion of objects within the frame. While this simple model is able to represent the movement of objects well in occlusion-free static-camera environments, its generalizability decreases in moving camera scenarios specifically when occlusion occurs. Several trackers [23, 32–35] used motion cues from objects and the ego-camera to provide a sturdier motion model. MAT [23] utilizes the Enhanced Correlation Coefficient Maximization (ECC) [36] model to estimate the camera's rotational and translational motion. Initially, the system employs Kalman Filter to predict object motion. Subsequently, an alignment step is performed using ECC to align the prediction. Furthermore, [35] introduces a motion-aware matching module to match tracks with the new observations based on the motion features.

While in the above motion modeling approaches the authors try to integrate motion clues in their method to increase the reliability of the MOT approach, none of them have directly applied the motion information to the Kalman Filter representation. In this work, SORT's state definition is modified, and the Kalman Filter is reformulated by injecting the ego-motion data into the formulation in order to reject

disturbances and increase the reliability of the model.

2.2.4 Baseline SORT-based Trackers

This section outlines the evolution of tracking algorithms built upon and inspired by the Simple Online and Realtime Tracking (SORT) [18] algorithm. SORT has significantly impacted the field of multi-object tracking, leading to the development of a new category of algorithms that derive from its foundational principles. Within this framework, four of the main state-of-the-art algorithms, developed as extensions of SORT, will be explored, showcasing the advancements and specific contributions each has made to the field.

Simple Online Real-time Tracking (SORT)

In SORT [18], the solution for the MOT problem is provided in three stages. Detection, prediction (estimation model), and data association. For the detection stage FrRCNN has been used to generate detected object bounding boxes from the input image in each frame. In the prediction stage, a linear constant velocity motion model has been used to model the motion of target objects and propagate the identity of targets to the following frame. The state definition of SORT is described in 4.3. where u and v represent the horizontal and vertical pixel coordinates of the center of the target object's bounding box, s denotes the scale (area) of the bounding box, r is the aspect ratio of the bounding box, and $\dot{u}$, $\dot{v}$, and $\dot{s}$ correspond to the first derivatives of u, v, and s with respect to time, respectively. This state vector is used to predict the new locations of existing objects in subsequent frames based on their current states.

In the final stage of SORT, data association, the Hungarian algorithm [37] is utilized to assign detections to existing tracks based on a cost matrix. The cost between each detection and all existing tracks is calculated using the Mahalanobis distance

between predicted states and detected bounding boxes, taking into account the uncertainty of the prediction. This approach allows for efficient tracking of objects across frames, even in cases of temporary occlusion or missed detections, by maintaining identity continuity for each tracked object.

The Hungarian algorithm [37] is a polynomial-time combinatorial optimization algorithm utilized for solving the assignment problem. It is particularly significant in the field of robotics and multi-object tracking, where it optimizes the assignment of objects to trackers in a bipartite graph model. The essence of the algorithm lies in its ability to find a perfect matching of minimum weight in a weighted bipartite graph.

Problem Formulation Consider a bipartite graph $G = (U, V, E)$ with two disjoint sets of vertices U and V, and edges E connecting vertices from U to V. Each edge $(u, v) \in E$ has an associated weight $w(u, v)$, representing the cost of assigning vertex u to vertex v. The goal is to find a perfect matching $M \subseteq E$ that minimizes the total weight $\sum_{(u,v)\in M} w(u, v)$, ensuring that each vertex is involved in exactly one matched pair.

Algorithm Overview The Hungarian algorithm operates in the following steps:

1. **Initial Reduction:** For each row and column of the cost matrix, subtract the smallest element of each row from all its elements, and then do the same for each column. This step reduces the problem to one where the minimum cost is zero.

2. **Covering Zeros:** Cover all zeros of the matrix using a minimum number of vertical and horizontal lines.

3. **Adjusting the Matrix:** If the minimum number of covering lines equals the matrix's order, it indicates the presence of an optimal assignment among the

covered zeros. Otherwise, locate the smallest element not covered by any line. Subtract this element from all uncovered elements and add it to elements covered twice. Proceed to repeat from step 2.

The algorithm proceeds iteratively to adjust the cost matrix and increase the number of independent zeros (zeros that are not in the same row or column as any other zero in the chosen set) until a complete matching is achieved.

Time Complexity The computational efficiency of the Hungarian algorithm is paramount for its application in real-time systems. The algorithm's time complexity is $O(n^3)$, where n is the number of vertices in one set of the bipartite graph. This cubic time complexity is derived from the need to perform matrix adjustments and zero coverings multiple times until the optimal assignment is found. Despite this seemingly high complexity, in practice, the Hungarian algorithm performs efficiently for the sizes of problems encountered in most applications within the robotics domain.

Furthermore, SORT incorporates an aging mechanism for tracks, whereby tracks without associated detections for a certain number of frames are terminated to remove outdated tracks. This mechanism helps in managing the number of tracks and ensures that the tracking process remains focused on currently visible objects.

Overall, the SORT algorithm offers a simple yet effective solution for multi-object tracking (MOT) by combining detection, prediction, and data association stages. Its efficiency and relatively low computational cost make it suitable for real-time tracking applications. However, it's important to note that the performance of SORT can be significantly influenced by the quality of the detection input and the chosen motion model's adequacy for the specific application scenario. Figure 2.4 shows an overview of SORT.

Figure 2.4: Diagram of the SORT (Simple Online and Realtime Tracking) algorithm, illustrating its streamlined process for multi-object tracking. The figure outlines the key phases of detection, state prediction with a Kalman Filter, data association using the Hungarian algorithm, and track management.

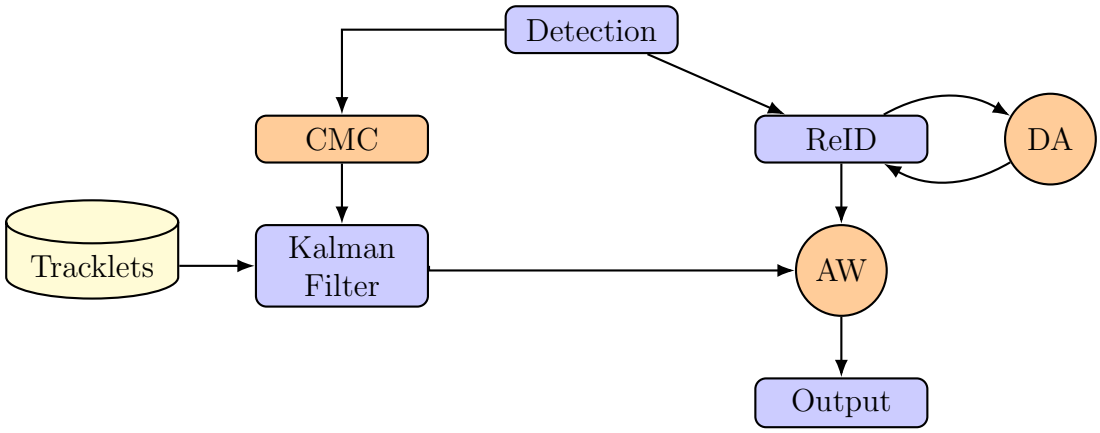

Figure 2.5: Deep OC-SORT Algorithm Architecture: This diagram depicts the workflow for real-time multi-object tracking, highlighting innovations like Camera Motion Compensation (CMC), Dynamic Appearance Modeling (DA), and Adaptive Weighting (AW), integrated with standard features like Re-identification (ReID) and Kalman Filter prediction. These elements collectively improve accuracy and adaptability in complex tracking scenarios. Innovations introduced by Deep OC-SORT are marked in orange.

2.2.5 OC-SORT

Building upon the foundational principles of SORT, Observation-Centric SORT (OC-SORT) introduces a novel framework designed to enhance robustness in tracking objects, particularly under conditions of occlusion and non-linear motion [20]. OC-SORT addresses the limitations of SORT by rethinking the motion model and data association stages to be more observation-centric. This approach significantly reduces error accumulation and increases tracking accuracy, especially in complex scenarios.

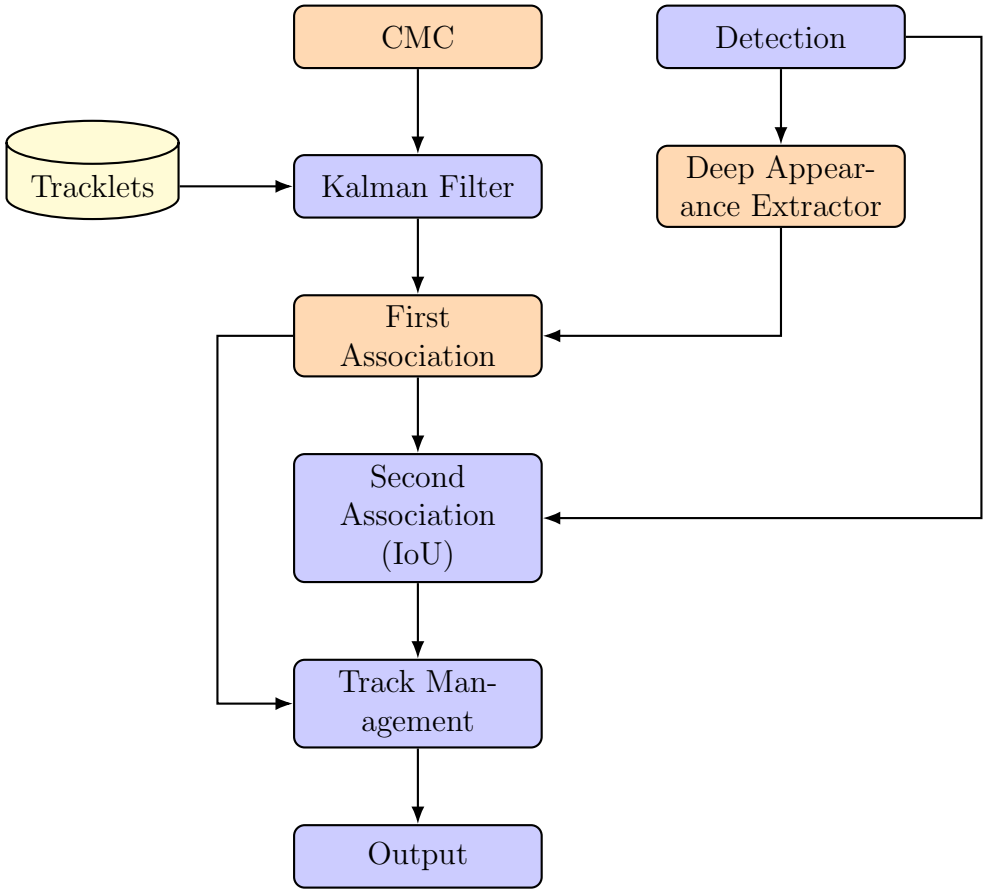

Figure 2.6: Overview of the BoTSORT algorithm for multi-object tracking, highlighting its core components: Detection, CMC, Tracklets, Deep Appearance Extraction, Kalman Filter, Associations, and Track Management, leading to the final Output. Innovations introduced by BoT-SORT are marked in orange.

Observation-centric Re-Update (ORU)

One of the key innovations of OC-SORT is the Observation-centric Re-Update (ORU) mechanism. ORU is activated when a track, previously lost due to occlusion or missed detections, is re-associated with a new detection. This mechanism leverages both historical and newly observed data to generate a virtual trajectory for the occluded period, effectively correcting the accumulated error in the Kalman Filter parameters of the track. The ORU process thus ensures that the track's state is accurately updated, reducing drift and improving the continuity of object tracking across frames.

Observation-Centric Momentum (OCM)

Another significant contribution of OC-SORT is the Observation-Centric Momentum (OCM) strategy, which incorporates the motion consistency of objects into the data association cost calculation. Unlike traditional approaches that rely on estimated motion for association, OCM uses direct observations from object detections to assess motion consistency. This reduces the influence of estimation noise on velocity direction calculation and enhances the reliability of track-detection associations, particularly in scenarios where object motions are non-linear and unpredictable.

Performance and Implementation

OC-SORT achieves state-of-the-art performance on several benchmark datasets, including MOT17 [2], KITTI [38], and DanceTrack [39]. It demonstrates significant improvements over SORT, particularly in handling occlusions and non-linear object motion, without compromising the algorithm's real-time processing capability. OC-SORT maintains a simple, online, and real-time framework, running at over 700 FPS on a single CPU, which underscores its efficiency and suitability for real-time tracking applications.

2.2.6 Deep OC-SORT

Deep OC-SORT [40] is an advanced multi-object tracking (MOT) algorithm that enhances the robustness and accuracy of tracking by adaptively incorporating visual appearance cues into a motion-based tracking framework. Originating from the foundational OC-SORT algorithm, Deep OC-SORT introduces three pivotal innovations: Camera Motion Compensation (CMC), Dynamic Appearance (DA), and Adaptive Weighting (AW), each tailored to mitigate specific challenges inherent in MOT. Fig-

ure 2.5 shows an overview of this method.

Camera Motion Compensation (CMC)

CMC aims to improve object localization in scenes characterized by camera motion. By leveraging a scaled rotation matrix and translation vector, CMC meticulously adjusts the detected object components and Kalman filter states to counteract camera movements. This process significantly enhances the precision of object position prediction and association over time, particularly in dynamic settings.

Dynamic Appearance (DA)

DA revolutionizes the integration of visual appearance information by adjusting the update rate of the appearance embedding based on the confidence level of the detection. This dynamic strategy enables selective incorporation of high-quality appearance information, diminishing the adverse effects of occlusions and motion blur on tracking accuracy. Employing a dynamic exponential moving average (EMA) for appearance embeddings, DA fortifies the tracker's resilience to appearance variations.

Adaptive Weighting (AW)

AW refines the detection-to-track association by augmenting the significance of appearance features according to their distinctiveness. It computes a cost matrix that combines motion and appearance cues, with an adjustment to the appearance weight influenced by the similarity scores between tracks and detections. This method ensures a heavier reliance on appearance information when it proves to be most distinctive, thereby elevating track association accuracy.

Performance and Evaluation

Deep OC-SORT achieves notable scores across various metrics on benchmark datasets such as MOT17, MOT20, and DanceTrack, including HOTA, MOTA, and IDF1. Its effectiveness is particularly evident in scenarios fraught with heavy occlusion and frequent object crossovers.

2.2.7 BoT-SORT

BoT-SORT [41] represents a significant advancement in the realm of Multi-object tracking (MOT), achieving state-of-the-art performance by adeptly combining motion and appearance information with camera motion compensation (CMC) and a refined Kalman Filter state vector. BoT-SORT and its extension, BoT-SORT-ReID, have demonstrated remarkable results on MOTChallenge datasets including MOT17 and MOT20, excelling in major MOT metrics such as MOTA, IDF1, and HOTA. Figure 2.6 provides a schematic representation of BoT-SORT.

Kalman Filter Improvements

At the core of BoT-SORT's motion model is the Kalman Filter (KF), traditionally utilized for predicting object trajectories. BoT-SORT introduces a modified state vector that directly estimates the width and height of bounding boxes, enhancing localization accuracy compared to conventional models that estimate the aspect ratio. This adjustment allows for a more precise fit of bounding boxes to detected objects, contributing to improved tracking performance.

Camera Motion Compensation (CMC)

To address the challenges posed by camera movements, BoT-SORT incorporates Camera Motion Compensation (CMC). By employing image registration techniques, the algorithm compensates for camera motion, thereby ensuring that the predicted locations of objects remain consistent and accurate across frames. This feature is particularly crucial in dynamic scenarios, where camera motion can lead to significant tracking errors.

IoU and Re-ID Fusion for Robust Associations

BoT-SORT introduces a novel method for fusing Intersection over Union (IoU) and Re-Identification (Re-ID) scores to establish robust associations between detections and tracklets. By prioritizing high-confidence detections for appearance feature extraction and employing a dynamic strategy for combining motion and appearance cues, BoT-SORT achieves a delicate balance between detection accuracy (MOTA) and identity preservation (IDF1).

Performance and Benchmark Evaluation

BoT-SORT and BoT-SORT-ReID have set new benchmarks on the MOT17 and MOT20 datasets, outperforming existing methods in all principal MOT metrics. This achievement underscores the effectiveness of BoT-SORT's integrated approach in handling various tracking challenges, including occlusions and crowded scenes.

2.2.8 ByteTrack

ByteTrack [19] introduces a novel paradigm in multi-object tracking (MOT) by valuing every detection box, irrespective of its score. This approach challenges the conven-

tional practice of discarding low-confidence detections, which often results in missing true objects and fragmented trajectories. ByteTrack's core innovation lies in its ability to effectively utilize low-score detections to enhance track continuity and robustness, especially in scenarios with occlusions or non-linear movements.

Association by Every Detection Box

Contrary to traditional methods that rely solely on high-confidence detections for identity association, ByteTrack incorporates nearly every detected bounding box in its association process. This inclusive strategy employs the similarities between low-confidence detections and existing tracklets to recover true objects and eliminate background detections effectively. The association process consists of two phases: initial matching of high-confidence detections to tracklets based on motion and appearance similarities, followed by a secondary matching phase where unmatched tracklets are associated with low-confidence detections using motion-based criteria.

Enhanced Tracking Performance

ByteTrack's association strategy significantly improves tracking performance, as demonstrated by its state-of-the-art results on various benchmark datasets including MOT17, MOT20, HiEve, and BDD100K. The algorithm not only excels in accurately tracking objects across frames but also in maintaining identity consistency even in challenging conditions. ByteTrack's effectiveness is particularly notable in densely populated scenes and in situations with frequent occlusions, where its strategy of utilizing low-confidence detections plays a pivotal role in preserving track integrity.

Simplicity and Efficiency

One of ByteTrack's key advantages is its simplicity and computational efficiency, enabling real-time tracking performance. The tracker's design leverages a high-performance object detector coupled with a straightforward yet effective association method, ensuring both high accuracy and speed. ByteTrack's ability to run at 30 FPS on standard hardware while achieving superior tracking accuracy underscores its potential for practical applications, setting a new benchmark for real-time multi-object tracking.

2.2.9 Evaluation Metrics

In the realm of multi-object tracking, a variety of metrics are employed to evaluate different performance attributes, such as detection accuracy, identity preservation, and overall efficacy of tracking. These metrics collectively shed light on the distinct functionalities and capabilities of the tracking algorithms, as detailed below:

1. **Higher Order Tracking Accuracy (HOTA)**: This metric evaluates multi-object tracking performance by integrating detection, association, and localization accuracies. It calculates the geometric mean of Detection Accuracy ($DetA$) and Association Accuracy ($AssA$) across various localization thresholds:

$$\text{HOTA} = \sqrt{\frac{1}{|Th|} \sum_{\alpha \in Th} \text{DetA}(\alpha) \times \text{AssA}(\alpha)} \tag{2.5}$$

   where Th represents the set of localization thresholds.

   - **Detection Accuracy (DetA)** measures the accuracy of detecting objects:

$$\text{DetA}(\alpha) = \frac{TP(\alpha)}{TP(\alpha) + FP(\alpha) + FN(\alpha)} \tag{2.6}$$

Here, $TP(\alpha)$, $FP(\alpha)$, and $FN(\alpha)$ represent the counts of true positives, false positives, and false negatives at threshold α.

- **Association Accuracy (AssA)** assesses the accuracy of maintaining correct identities across frames:

$$\text{AssA}(\alpha) = \frac{\sum_{i \in TP(\alpha)} A_i}{|TP(\alpha)|} \tag{2.7}$$

A_i is the association score for each true positive, defined as:

$$A_i = \frac{TPA(i)}{TPA(i) + FNA(i) + FPA(i)} \tag{2.8}$$

where $TPA(i)$, $FNA(i)$, and $FPA(i)$ are the True Positive Associations, False Negative Associations, and False Positive Associations, respectively.

Integration Over Localization Thresholds

The final HOTA score is calculated by integrating these accuracies across various localization thresholds to comprehensively evaluate a tracker's performance, reflecting robustness across different scenarios and applications.

2. **Multiple Object Tracking Accuracy (MOTA)**: Aggregates errors from false positives, false negatives, and identity switches to provide an overall accuracy measure:

$$MOTA = 1 - \frac{\sum_t (FP_t + FN_t + IDS_t)}{\sum_t GT_t} \tag{2.9}$$

where FP_t, FN_t, and IDS_t are counts of false positives, false negatives, and identity switches at time t, respectively.

3. **ID F1 Score (IDF1)**: Reflects the accuracy of identity assignments, computed

as the harmonic mean of identity precision and recall:

$$IDF1 = \frac{2 \times IDTP}{2 \times IDTP + IDFP + IDFN} \tag{2.10}$$

with $IDTP$, $IDFP$, and $IDFN$ representing the true positive, false positive, and false negative identity matches.

4. **Identity Switches (IDs)**: Counts the number of times an object is incorrectly reassigned a different identity. Lower values indicate better performance in maintaining identity consistency.

Together, these metrics provide a comprehensive framework for evaluating MOT systems, enabling researchers and developers to benchmark the performance of algorithms under various conditions and use cases. By quantifying different aspects of the tracking process, they help highlight the strengths and weaknesses of specific approaches, guiding future improvements in tracking technology.

2.3 Framework for Testing Multi-object Tracking Algorithms

This section introduces a comprehensive framework developed for the evaluation and testing of Multi-object tracking (MOT) algorithms. The framework is designed to facilitate a structured approach to assessing the performance and robustness of MOT algorithms across various scenarios. Central to the framework's design is its integration with the Robot Operating System (ROS), which serves as the backbone for data transfer and communication between components.

2.3.1 Framework Overview

The framework operates across multiple levels, each designed to progressively increase the complexity and realism of the testing environment. At its core, the framework utilizes two main simulators: Gazebo and CARLA. Gazebo offers a simplified, yet effective environment for initial proof of concept and testing of basic scenarios. CARLA, an advanced autonomous driving simulator, is employed for more complex and dynamic testing environments, particularly those relevant to autonomous driving applications. Figure 2.8 shows an overview of the proposed framework.

2.3.2 Simulation-based Testing

Initial testing is conducted within the Gazebo simulator, providing an accessible platform for basic algorithm validation. Following successful preliminary tests, the framework transitions to the CARLA simulator. In this framework, a script is provided to dynamically generate tailored environments for various autonomous driving tasks. This includes the configuration of different numbers of vehicles, diverse weather conditions, and the selection among various towns available in CARLA. Such versatility allows for comprehensive and realistic scenario testing, essential for evaluating the performance of multi-object tracking algorithms under varied operational conditions.

Furthermore, our framework is designed to produce ground truth data in the MOT17 format. This feature is critical for the systematic evaluation of MOT algorithms, as it aligns with established benchmarks in the field. The MOT17 style ground truth data facilitates a direct comparison of algorithmic performance against well-known datasets, ensuring that our evaluation metrics are both relevant and standardized. This capability significantly enhances our ability to assess the effectiveness of MOT algorithms, providing a robust foundation for identifying areas of improve-

ment and validating algorithmic advancements in real-world conditions. Additionally, Figure 2.7 visualizes the process of automatic ground truth generation within a CARLA environment. This illustration underscores the framework's capability to meticulously simulate complex scenarios and generate precise ground truth data, which is indispensable for the comprehensive evaluation of multi-object tracking algorithms. By leveraging such detailed simulations, it should be ensured that the algorithms are tested against diverse and challenging conditions.

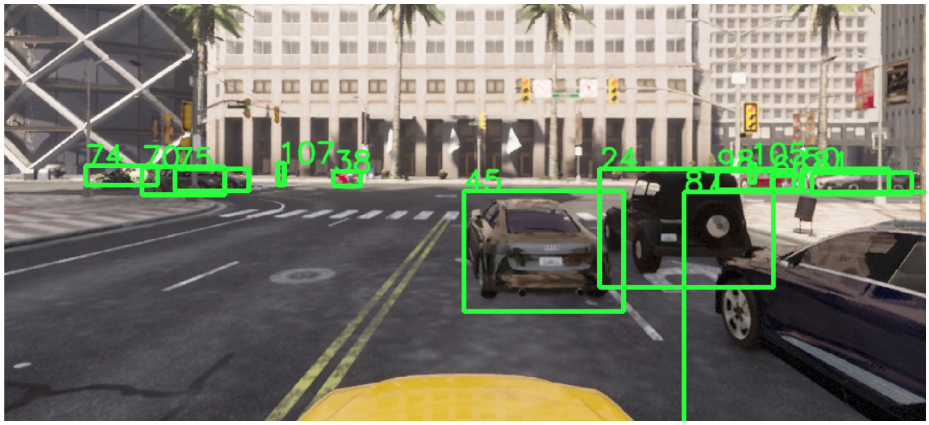

Figure 2.7: Visualization of automatic ground truth generation within a CARLA environment.

2.3.3 Input Data and Algorithm Evaluation

Input to the MOT algorithms can be sourced from two primary channels: detections from YOLOv8 or perturbed ground truth data, with configurable noise parameters. This dual-input approach offers flexibility in testing algorithm robustness against varying levels of data fidelity and noise. The automatic generation of ground truth in CARLA plays a pivotal role in benchmarking algorithm performance, facilitating objective comparison and evaluation.

2.3.4 Real-world Deployment

After rigorous testing in simulated environments, algorithms demonstrating high levels of performance and reliability are candidates for deployment on a real-world, ROS-powered robotic platform. This final stage of testing underscores the framework's commitment to developing MOT algorithms that are not only theoretically sound but also practically viable in real-world applications.

2.3.5 Conclusions

The proposed framework represents a structured and comprehensive approach to the testing and evaluation of multi-object tracking algorithms. By leveraging the capabilities of ROS, Gazebo, and CARLA simulators, along with sophisticated detection inputs, the framework ensures a thorough assessment of algorithms across a spectrum of environments, from simulated to real-world applications.

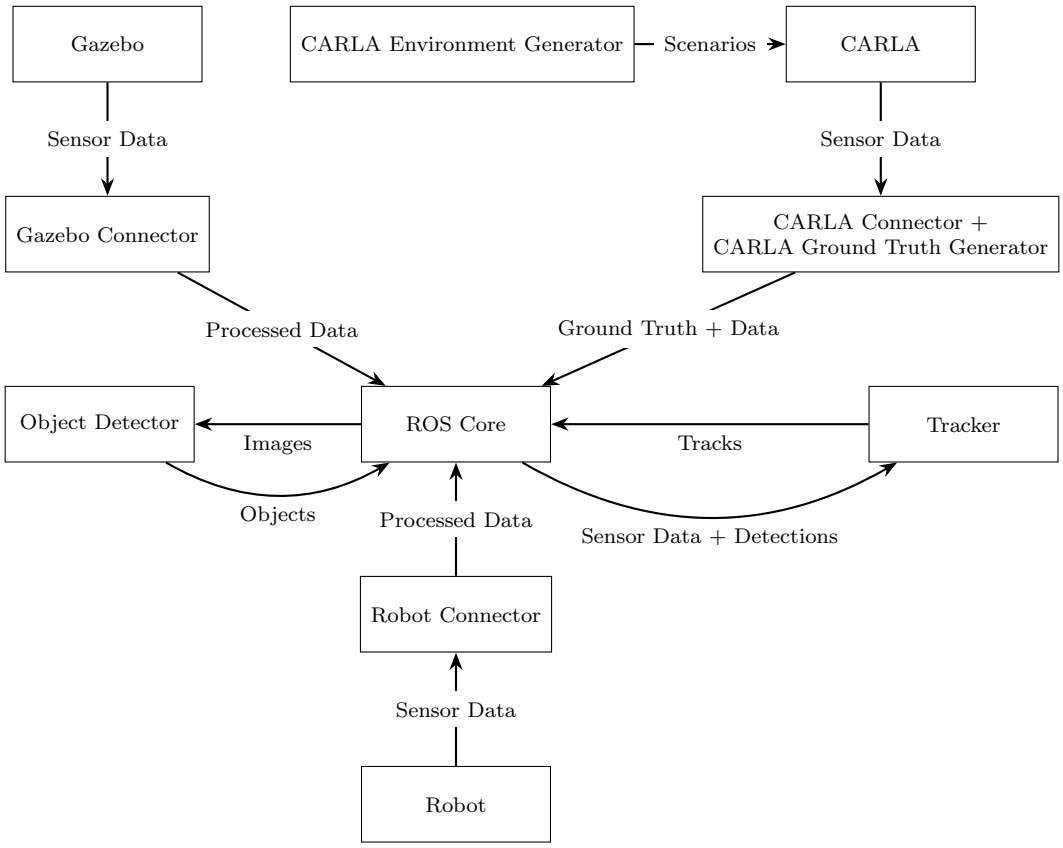

Figure 2.8: Schematic representation of the data flow and connectivity in the ROS-based framework for multi-object tracking algorithm testing. This framework integrates data from Gazebo, CARLA, and a real-world robot through respective connectors, emphasizing the central role of ROS in managing and processing sensor information.

Chapter 3

Self-supervised Multi-object Tracking Algorithm with Adaptive Track-Matching

This chapter proposes a self-supervised algorithm for efficient multi-object tracking using unlabeled image sequences. Thus, the proposed approach omits the need for labor-intensive labeling of video inputs, and its novelty lies in the ability to tune the track-matching threshold parameter through a reliable self-supervisory signal. This enables the method to be generalizable and insensitive to the parameters of a particular dataset. Moreover, multiple post-processing modules are introduced that enhance the robustness and reduce the number of identity switches. The approach is evaluated on the MOT20 dataset. The results show that the proposed method has comparable performance in comparison to other supervised and unsupervised learning methods without the need for fine-tuning the hyperparameters for new applications.

3.1 Introduction

Multi-Object Tracking (MOT) is a prominent application in the field of computer vision. The general task of MOT consists of detecting objects presented in each frame of a given video sequence and maintaining the association between the detected objects and their corresponding tracks through time [42]. MOT has a wide range of applications in various fields. In self-driving cars [43], for example, MOT is used to perform environment mapping and path planning. It can also assist with obstacle avoidance, and help to identify different stationary and dynamic objects such as pedestrians and vehicles. Moreover, MOT is heavily applied in video surveillance applications [44], where the task is to detect and track individuals or moving vehicles with either a stationary or a moving camera. Another application of MOT is video analysis [45], where MOT can be used to perform analysis on, for instance, soccer match videos to evaluate the players' performance.

The primary objective of MOT is to produce trajectories for a varying number of objects in a given image sequence. The developed algorithm should account for any object that may disappear or any new object that may appear at a certain frame. The main challenges of the MOT task can be summarized into (1) identifying the objects in each frame, (2) pairing the detected objects across the frames, and (3) initializing and terminating tracks as objects enter or leave the scene [46].

Existing methods of MOT can be categorized into two groups. Detection-free tracking (DFT) and Detection-based tracking (DBT). in DFT, the position of the objects is assigned manually in the first frame, and the specified objects are tracked throughout the remaining frames. While this approach eliminates the requirement of using an object detector, It may not be feasible to extend it in scenarios where new objects enter the scene. The second category of MOT is DBT, in which the

detections of the objects at each frame are given to the tracker as an input [42]. Due to the advances in object detection algorithms, DBT methods have recently gained more popularity.

Recent work has investigated unsupervised learning approaches for MOT. These approaches are anticipated to overcome the shortcomings of other supervised methods. Despite the high performance of supervised methods, they require a huge amount of tracking annotations, as most of the available MOT datasets are unlabeled.

The main contributions of our work are summarized as follows:

- A novel self-supervised multi-object tracking algorithm is proposed that utilizes both visual and spatial information extracted from image sequences.
- A generic mechanism is proposed that can adapt to new unseen datasets. The feature encoding module is trained independently, and the track association threshold is adaptively adjusted.
- The proposed approach is evaluated through a range of experiments on the MOT20 dataset against other supervised and unsupervised baselines. A detailed ablation study is provided to further clarify the impact of each module.

3.2 Methodology

The multi-object tracking (MOT) problem is aimed at identifying objects in a video and tracking them over time. Given a video of N frames, the i_{th} frame is denoted by F_i, containing M_i bounding boxes of objects in each frame. The input is represented by matrix D_{ij}, where $i \in [1, N]$ and $j \in [1, M_i]$, with each element of the matrix representing a detected bounding box. The goal of the MOT problem is to find the connection between these detected objects across frames and to output a set of

tracking sequences T, where each track represents a ground truth object among L objects denoted by O_1 to O_L.

The proposed model is based on the tracking-by-detection framework where a deep learning-based object detection algorithm is utilized to generate the hypothesis bounding boxes. Each bounding box contains four values that represent the coordinates of the top, left, bottom, and right positions of the detected object such that, $D_{ij} = \left[D_{ij}^{TLx},\ D_{ij}^{TLy}, D_{ij}^{BRx},\ D_{ij}^{BRy}\right]$. In the problem formulation, the hypotheses are regarded as nodes, and the connection between every two hypotheses is considered as an edge. Hence, the MOT problem can be defined as a Constraint Satisfaction Problem (CSP) [47] which is defined on a graph. The basic constraints for this problem enforce three conditions: no backward edge is selected, each node is connected to at most one other node, and no node has multiple incoming nodes. These constraints are formulated as follows:

- $\forall\ i,\ i', j, j'((D_{ij}, D_{i'j'})\ \in\ Edges\ \implies i' > i)$
- $\forall i, i', i'', j, j', j''\ (\ (D_{ij}, D_{i'j'})\ \wedge\ (D_{ij}, D_{i''j''}))\ \implies$
  $(i' = i'',\ j' = j'')$
- $\forall i, i', i'', j, j', j''((D_{ij}, D_{i''j''})\ \wedge\ (D_{i'j'}, D_{i''j''}))\ \implies$
  $(\ i = i',\ j = j')$

In this work, two similarity features between images are used to define the weight parameters of the edges on the graph. The first feature is the Visual Dissimilarity Feature (VDF) which is based on the visual representation of the detected images. The second feature is the Spatial Dissimilarity Feature (SDF) which is based on the relative positions of the bounding boxes with respect to their corresponding frames.

3.2.1 Visual and Spatial Dissimilarity Feature

VDF compares the visual similarity between two detected objects in different frames. The objective is to generate a representation vector for each object bounding box. The representation vector is similar for all the bounding boxes of the same objects, hence it can be used to differentiate dissimilar objects. To compare the visual features of any two images, the visual dissimilarity vector is first found, and the Euclidean distance is then calculated between the two vectors to represent the distance, as shown in Equation 3.1.

$$VDF(D_1, D_2) \;=\; \sqrt{[\overrightarrow{f}(D_1) - \overrightarrow{f}(D_2)]^2} \tag{3.1}$$

where, $\overrightarrow{f(.)}$ is the image encoding vector.

SDF is defined as the distance between the center of the bounding box of two images. Choosing this metric as a dissimilarity factor is justified by the fact that object movement and camera movement are minimal between every two frames from videos taken by most camera devices.

3.2.2 Visual-spatial Fusion

In the field of object tracking, it is crucial to consider both visual and spatial information from bounding boxes in order to accurately generate tracks. While using spatial information alone can provide valuable information about the movement of objects, it may not be sufficient to distinguish between objects that move in close proximity to each other, resulting in lost tracks. Using visual information only, on the other hand, would result in incremental tracking errors, especially in cases with object occlusion. Hence the visual information would be inadequate. However, based on the fact that object movement is limited between consecutive frames, the spatial

information would provide additional helpful information and would compensate for such limitation. To demonstrate the importance of fusing both VDF and SDF, Figure 3.2 illustrates a failing example of a tracker that uses SDF alone. In the specific case of a crowded scene with multiple objects, the spatial information may not be sufficient to differentiate between the objects and maintain accurate tracks. Figure 3.3, on the other hand, shows that VDF alone would not be sufficient for tracking objects in occluded scenes. As a result, utilizing both VDF and SDF information is a good practice for robust object tracking. The pseudocode for generating dissimilarity between two images is shown in Algorithm 1.

Algorithm 1 Dissimilarity Function

Input: Feature vectors a, b
Output: Dissimilarity measure $dissim$

```
dissim ← get_vdf(a.vis, b.vis)
spatial_dissim ← get_sdf(a.pos, b.pos)
if spatial_dissim > Maximum_Distance then
    dissim ← INFINITY
end if
return dissim
```

3.2.3 Identification Algorithm

The presented study proposes a novel self-supervised method for solving the multiple object tracking problem. The algorithm receives the current frame, represented by F_{last}, and the output of a detection algorithm, which is indicated by a set of bounding boxes denoted by bb_i, where i is the i^{th} bounding box. The algorithm maintains several global variables that are updated dynamically throughout its execution. OT represents a list of open tracks and PT represents a list of potential tracks identified so far.

The algorithm employs the defined dissimilarity function to evaluate the similarity between each pair of bounding boxes bb_i extracted from the current frame F_{last}.

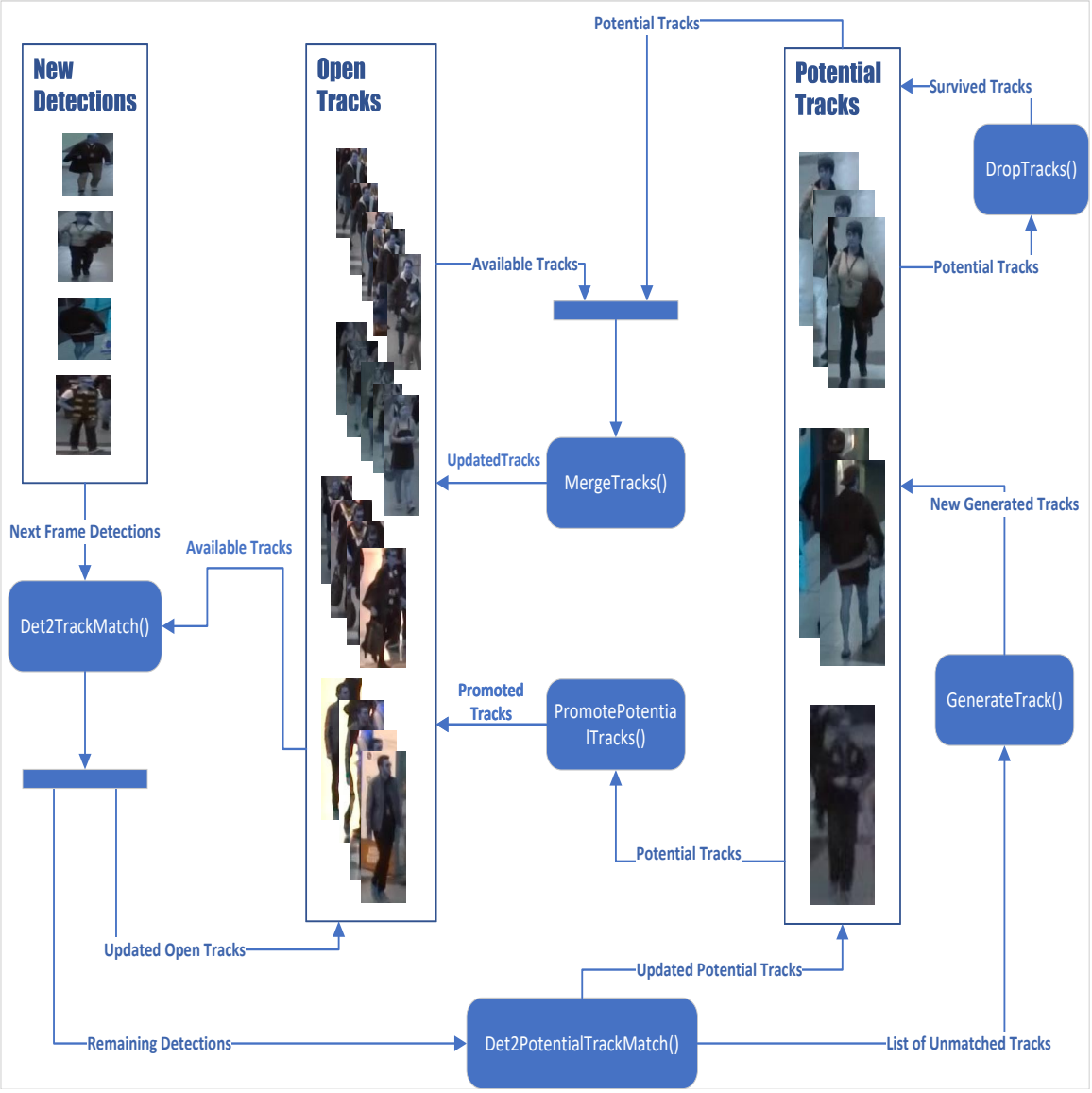

Figure 3.1: The proposed algorithm design flow diagram. Rounded boxes denote the primary system functions.

The resulting dissimilarity scores are stored in a limited capacity pool and are used to determine the association between bounding boxes and existing open tracks OT or to initiate new potential tracks PT. The algorithm begins by calculating the dissimilarity between all pairs of bb_is in the current frame and storing the scores in $pool$. The distribution of the scores in $pool$ is following a normal distribution and a threshold of $(\mu - 2 * std)$ is established for comparison purposes, which has been empirically determined.

For each bb_i, the algorithm calculates the dissimilarity between the current bounding box and the last K frames of each track in OT. The resulting $K * len(OT)$ dissimilarities are aggregated into a median for each open track, and the minimum aggregated dissimilarity score determines the association between the bounding box and an open track. This process continues until all bounding boxes have been assigned to an open track or have initiated a new potential track.

To meet the criteria for association with an open track, two conditions must be met. The first constraint requires that the aggregated dissimilarity score must be below a visual adaptive threshold, which is dynamically updated for each frame based on the distribution of dissimilarity scores in the limited capacity pool. The second constraint requires that the new bounding box must be within a certain distance from the last 5 frames of the open track, as determined by a positional threshold.

If the spatial constraint is not satisfied, the bounding box is not associated with any existing open track and is reserved for the next phase of the algorithm. If the visual constraint is not met, the algorithm moves on to the next minimum dissimilarity score and repeats the process for the next candidate potential track.

The next phase of the algorithm proceeds with any remaining bounding boxes and follows a similar process. If a bounding box is still not associated with any open track or potential track after this second phase, it initiates a new potential track.

In the context of the proposed algorithm, a potential track is a temporarily designated tracking sequence that has not yet satisfied the requirements to be classified as an open track. Once the potential track's length surpasses 10 frames, the algorithm reassesses its status and, if deemed appropriate, removes it from the list of potential tracks PT and adds it to the list of confirmed open tracks OT. If no new track is added to the potential track for 40 frames, the potential track with all the included frames drops.

In an effort to enhance the accuracy of the tracking results, the proposed algorithm incorporates a merging phase to mitigate the issue of identity switching and avoid losing meaningful bounding boxes. This merging process occurs in two parts of the algorithm. The first instance is when a potential track is being considered for promotion to an open track. At this stage, the merging function is employed. The second instance is when the recency of the potential track is too high, and a decision must be made as to whether the track should be dropped or not. In this phase, the merging process decides whether the potential track should be merged with an existing open track or be assigned a new independent ID or drop.

The algorithm employs two criteria to determine if a potential track should be merged with an open track. The first criterion, referred to as "puzzling", checks the frame IDs of the bounding boxes in the potential track to ensure that they are complementary to the frame IDs of the open track. If there is a single overlap between the two, the merging cannot occur and the potential track is assigned a new open track ID. If the puzzling constraint is met, the visual constraint is then evaluated. This constraint is calculated by determining the dissimilarity between the objects in the potential track and the last 15 frames of the open track. The median of these dissimilarities is then compared to a relaxed empirically determined visual threshold of $(\mu - 0.5 * \sigma)$, which is derived from the adaptive visual threshold.

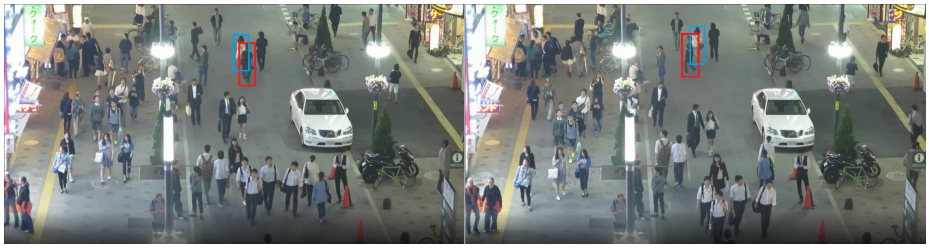

Figure 3.2: The two objects inside the blue and red bounding boxes are walking close to each other. If tracking is only based on the spatial information, tracking these two objects would be unreliable [2].

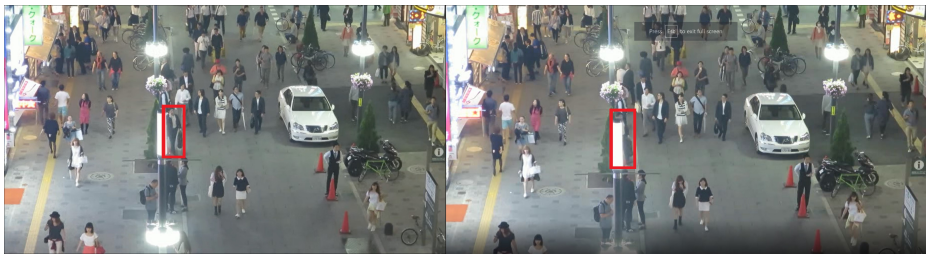

Figure 3.3: The object in the red bounding box gets occluded by the lamp in this example. In this case, relying only on visual information would not be enough to track the object [2].

3.3 Results and Discussion

3.3.1 Dataset

The proposed method is evaluated on MOT20 [48] dataset. The dataset contains 8 sequences, half of which construct the training set, and the remaining four sequences form the test set. The main focus of the MOT challenge is pedestrian tracking which has been introduced in MOT15 [49] by the same team. MOT20 is the most recent version of this dataset, consisting of highly populated and crowded scenes.

The MOT20 dataset includes tracking ground truth labels for the sequences in the training set. However, our approach does not involve a training phase, so these tracking labels are not utilized. In order to generate the VDF, the MobileNetV2 network [50] is used. This network encodes every image of size $224 \times 224 \times 3$ and

returns a 1280 feature vector. The network has a convolutional layer which is followed by 19 residual bottleneck layers.

3.3.2 Evaluation Criteria

In the work, the method is being evaluated by following the current well-known object tracking metrics. In computer vision, the Multi-Object Tracking Precision (MOTP) metric is used to evaluate the accuracy of the estimated positions of objects. It is computed by assessing the cumulative error in the estimated position for matched ground truth-hypothesis pairs across all frames and then averaging it by the total number of matches formed. However, it should be noted that MOTP does not take into account the recognition of object configurations or the evaluation of object trajectories.

On the other hand, the Multi-Object Tracking Accuracy (MOTA) metric measures the number of errors made by the tracking system, such as misses, false positives, and mismatch errors. This metric is calculated by determining the ratio of these errors to the total number of frames, including the ratios of misses, false positives, and mismatches.

To evaluate the performance of an object tracking model, it is important to consider three key criteria. These criteria are essential in determining the effectiveness of the model, and in identifying areas for improvement. To develop a more robust method, it is crucial to delve into these criteria in detail, thoroughly analyzing each one to understand how they impact the overall performance of the model.

- A "miss" in object tracking refers to a situation where an object is present in the ground truth data but no corresponding hypothesis is output by the tracking system. To calculate the number of misses, all objects for which no hypothesis was output must be counted.

- A "false positive" in object tracking refers to a situation where a hypothesis is an output by the tracking system but there is no corresponding object in the ground truth data. To calculate the number of false positives, all tracker hypotheses for which no real object exists must be counted.

- A "mismatch error" occurs when the track ID for an object changes through the sequence. This can happen in situations such as when multiple objects exchange positions in close proximity or when an object's trajectory is reset with a new track ID following its prior loss due to occlusion. To determine the number of mismatch errors, it is imperative to enumerate all instances where the tracking ID for an object has undergone alteration.

3.3.3 Decreasing The Mismatch Rate

Decreasing the mismatch error and the number of identity switches is the primary challenge in the evaluation of the MOT task. Imposing a strict constraint on the track matching results in an increased number of disconnected tracks. On the other hand, relaxing the matching threshold value and the matching constraints would increase the number of identity switches. The choice of an adaptive matching threshold automates the process of finding an appropriate threshold for each specific situation. The sorted matching module reduces the possibility of assigning new detections to one of the open tracks by prioritizing object-to-track matches that have less uncertainty. Finally, by applying the two phases of merging, lost tracks will be reidentified and the puzzle effect will prevent the matching of tracks that have conflicting detections.

3.3.4 The Self-supervised Paradigm

The proposed method possesses strong generalizability due to its adaptive matching thresholds, which play a significant role in ensuring the approach remains effective when faced with changes in datasets. Furthermore, the empirical variables utilized in the algorithm were calibrated using the MOT17 dataset and were not influenced by the MOT20 training dataset.

3.3.5 Results

This work's primary objective is not to achieve state-of-the-art performance on the MOT20 dataset, but instead, to provide a generalizable and adaptable solution to the multi-object tracking problem. Nevertheless, the tracking performance was evaluated on the MOT20 dataset and the results illustrate that the achieved MOTA accuracy outperforms SORT [51]. Additionally, the method demonstrates competitive performance when compared to other supervised approaches. The evaluation results are shown in Table 3.1.

Table 3.2 shows an ablation study of the proposed method. Specifically, the impact of each module is reported using the MOTA metric on the MOT20 training dataset. The results suggest that an adaptive matching threshold has a significant effect on the overall performance compared to the case of a fixed threshold. Moreover, the number of wrong detection to track assignments increases remarkably when the puzzle module is removed.

3.4 Conclusions

In this study, a generalizable and self-supervised approach for solving the multi-object tracking problem is proposed. Our method introduces an adaptive track-matching

Table 3.1: Performance on the MOT20 test set. Metrics: MOTA (Multi-Object Tracking Accuracy), IDF1 (ID F1 Score), MT (Mostly Tracked Trajectories Ratio), ML (Mostly Lost Trajectories Ratio), FP (False Positives), FN (False Negatives), AssPr (Association Precision), Frag (Track Fragmentations).

	Method	**MOTA**	**IDF1**	**MT**	**ML**	**FP**	**FN**	**AssPr**	**Frag**
Unsupervised	SORT[51]	42.7	45.1	208	326	27.5K	265K	66.9	18K
	Ours	43.1	31.6	238	367	6K	270K	69.7	6.8K
Supervised	GMPHD[52]	44.7	43.5	293	274	43K	126K	72	11.1K
	CT[53]	45.1	35.6	409	235	69K	208K	75.4	6.3K

Table 3.2: The impact of each module of the approach evaluated on the MOT20 training set

Method Components					Evaluation Metrics							
Puzzle	Adaptive Threshold	Sorted Matching	Positional Elimination	Merging	MOTA	IDF1	MT	ML	FP	FN	IDSw	Frag
✓	✓	✓	✓	✓	52.73	50.8	14	12	3228	12K	552	256
	✓	✓	✓	✓	50.7	49.6	11	13	2996	12K	421	357
✓	✓	✓	✓		48.2	51.0	12	13	2369	12K	339	573
		✓	✓	✓	33.4	30.8	7	33	2800	16K	215	242

threshold derived from a reliable signal, which enables the algorithm to be adaptive to different datasets. The main goal of this work is not to attain the highest level of performance on the MOT20 dataset, but rather to create a solution that is versatile and can be adapted to address the multi-object tracking challenge in general. The outcomes demonstrate that the suggested method delivers comparable results to other supervised and unsupervised learning approaches, and does not require hyperparameter fine-tuning for new applications. In the future, the replacement of the VDF calculation process with a Siamese network, which can automatically detect image similarity between different frames, is intended. Moreover, the process of merging discarded potential tracks as a post-processing module is planned to be extended to further enhance the performance of the algorithm.

Chapter 4

Ego-motion Aware Target Prediction Module for Multi Object Tracking

Multi-object tracking (MOT) is a prominent task in computer vision with application in autonomous driving, responsible for the simultaneous tracking of multiple object trajectories. Detection-based multi-object tracking (DBT) algorithms detect objects using an independent object detector and predict the imminent location of each target. Conventional prediction methods in DBT utilize Kalman Filter (KF) to extrapolate the target location in the upcoming frames by supposing a constant velocity motion model. These methods are especially hindered in autonomous driving applications due to dramatic camera motion or unavailable detections. Such limitations lead to tracking failures manifested by numerous identity switches and disrupted trajectories. In this chapter, a novel KF-based prediction module called the **E**go-**M**otion **A**ware Target **P**rediction (EMAP) module is introduced by focusing on the integration of camera motion and depth information with object motion models. Our proposed

method decouples the impact of camera rotational and translational velocity from the object trajectories by reformulating Kalman Filter. This reformulation enables us to reject the disturbances caused by camera motion and maximizes the reliability of the object motion model. Our module is integrated with four state-of-the-art base MOT algorithms, namely OC-SORT, Deep OC-SORT, ByteTrack, and BoT-SORT. In particular, our evaluation on the KITTI MOT dataset demonstrates that EMAP remarkably drops the number of identity switches (IDSW) of OC-SORT and Deep OC-SORT by 73% and 21%, respectively. At the same time, it elevates other performance metrics such as HOTA by more than 5%.

4.1 Introduction

Multi-object tracking stands out as a pivotal task in computer vision, finding diverse applications, ranging from monitoring crowds via surveillance cameras to analyzing the intricate movements of soccer players on the field to tracking vehicles on the road and following marine life in the ocean [54]. Additionally, MOT serves as a critical subsystem in autonomous vehicle systems, providing a medium-level representation integral to path-planning processes [55].

Detection-Based Tracking is commonly employed in these applications due to its efficacy in scenarios where new objects may emerge or existing objects may exit the frame. Within a DBT framework, MOT can be represented as a graph optimization problem. In this context, an object detector generates bounding boxes of detected objects in every frame, which serve as graph nodes, and two nodes are connected if and only if they belong to the track of the same object. The basic constraints for this problem enforce three conditions: no backward edge is selected, each node is connected to at most one other node, and no node has multiple incoming nodes. Thus,

the MOT problem is reduced to determining the optimal association between nodes in the graph. Approaches like Tracktor++ [56] perform such graph optimization under the assumption of a high frame rate and available detections in every frame.

(a) (b)

Figure 4.1: In this scenario, the object detection is missed, leading to a failure in predicting the location of the object during lane-changing for Vanilla ByteTrack (a). ByteTrack + EMAP (b) significantly improves the prediction, successfully tracking the object in the next two frames.

However, restricting association to every pair of consecutive frames may be overly simplistic since it barely reflects real-world scenarios where ego-vehicle/camera motion and occlusion exist. In situations where detections are not consistently available

in every frame, it becomes necessary to either abandon the assumption of association between adjacent frames or substitute the unavailable detection with object state predictions. The illustrative example depicted in Figure 4.2 showcases three objects being tracked over four consecutive frames along with their corresponding graph representation. Since the objects are not detected in every frame, the track prediction is employed in lieu of the missing detection.

In Kalman-filter-based methods [57], a motion model is assumed for the objects in the frame and is described by a state transition matrix, $\mathbf{F}$, an observation matrix, $\mathbf{H}$, a process noise, $\mathbf{Q}$, an observation noise, $\mathbf{R}$, and an optional disturbance matrix, $\mathbf{G}$. The object location in the image frame is regularly predicted at each time step using the *predict* stage of Kalman Filter:

$$\begin{cases} \hat{\mathbf{x}}_{n+1|n} = \mathbf{F}_n \hat{\mathbf{x}}_{n|n} + [\mathbf{G}_n \mathbf{w}_n] \\ \mathbf{P}_{n+1|n} = \mathbf{F}_n \mathbf{P}_{n|n} \mathbf{F}_n^\top + \mathbf{Q}_n \end{cases} \tag{4.1}$$

Afterwards, newly detected objects are matched with the predicted positions of existing objects, resulting in new observations denoted as $\mathbf{z}$. In the next stage of the Kalman Filter known as the *update* stage, the Kalman gain, state estimate, and estimate covariance are all recalibrated:

$$\begin{cases} \mathbf{K}_n = \mathbf{P}_{n|n-1} \mathbf{H}^\top (\mathbf{H} \mathbf{P}_{n|n-1} \mathbf{H}^\top + \mathbf{R}_n)^{-1} \\ \hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{K}_n (\mathbf{z}_n - \mathbf{H} \hat{\mathbf{x}}_{n|n-1}) \\ \mathbf{P}_{n|n} = (\mathbf{I} - \mathbf{K}_n \mathbf{H}) \mathbf{P}_{n|n-1} \end{cases} \tag{4.2}$$

Many Kalman-filter-based approaches are constructed upon the foundation of SORT [18], which defines the state as follows:

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^\top \tag{4.3}$$

In this definition of the state, u and v denote the horizontal and vertical pixel positions of the target object's center, while s and r denote the area and height of the target object's bounding box, respectively. Notably, this definition operates under the assumption of a constant velocity model for the tracked objects. While the constant velocity model assumption may not hold for all objects, it serves as a reasonable estimation in scenarios where the camera is stationary, as exemplified by the MOT20 dataset [58]. In cases where the camera is affixed to a mobile vehicle, the addition of the motion of the ego vehicle leads to an increased failure rate in the constant velocity assumption. Specifically, as the motion of the camera becomes more dynamic, the accuracy of predicted $\dot{u}$ and $\dot{v}$ diminishes, even when the tracked object maintains a constant velocity. In Figure 4.1 (a), despite the tracked vehicle moving with a constant velocity, the performance of Kalman Filter is compromised due to the non-uniform motion of the camera. This discrepancy becomes particularly pronounced during lane changes, where the constant-velocity Kalman Filter struggles to make accurate predictions without the availability of new observations to account for the changing dynamics.

In this work, the integration of an Ego-Motion Aware Target Prediction (EMAP) module into Kalman-filter-based systems is proposed. This module serves to incorporate the motion of the ego-vehicle into the Kalman Filter, thereby enhancing the performance of MOT systems. Our contributions can be summarized as follows:

- An ego-motion aware target prediction module is designed, aimed at enhancing the performance of detection-based Multi-Object Tracking (MOT) algorithms.
- The Kalman Filter state is redefined to decouple camera motion from object trajectories, thereby rejecting disturbances caused by camera motion and maximizing the reliability of the object motion model.

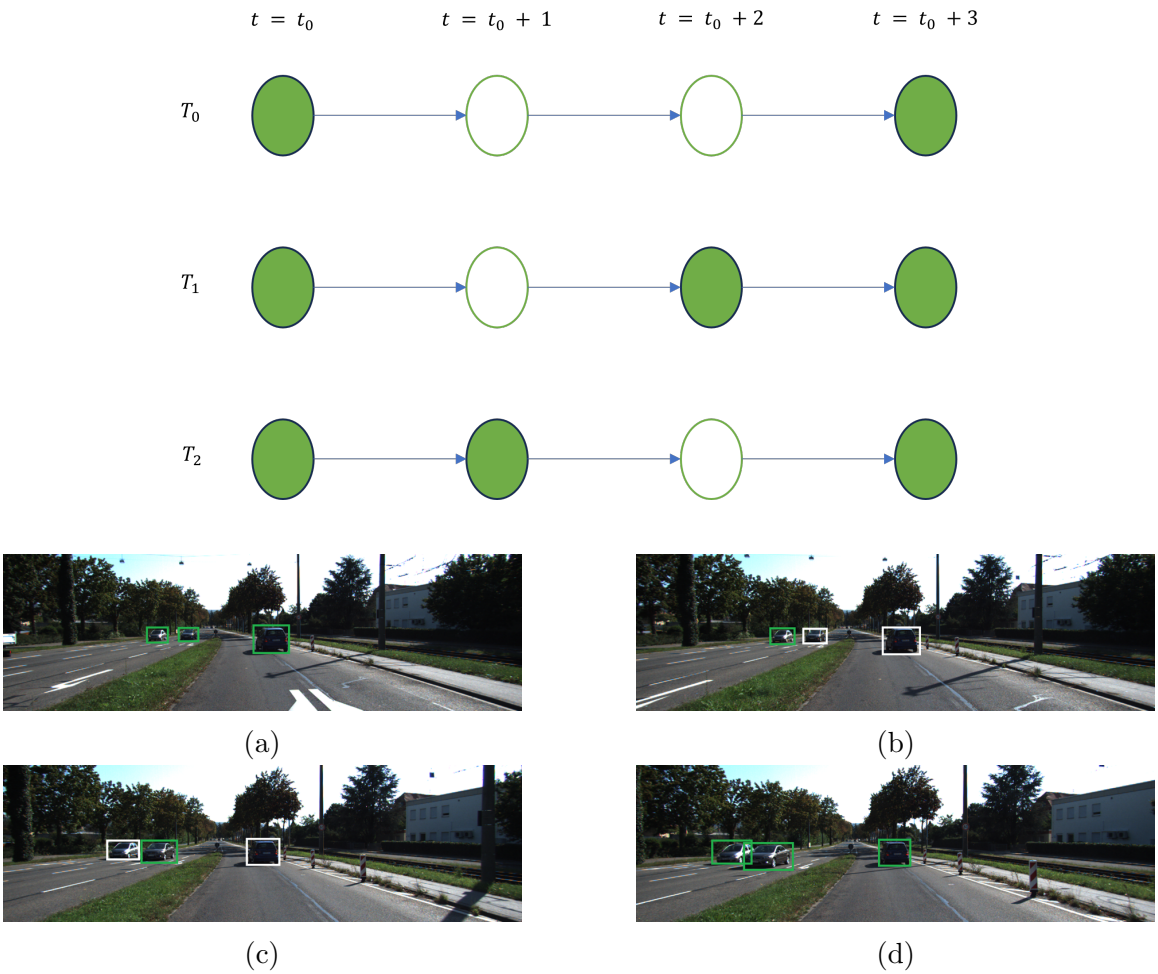

Figure 4.2: Illustrative example of multi-object tracking over four consecutive frames, highlighting the tracking of three objects and their graph representation. Green nodes represent available detections, whereas white nodes indicate unavailable detection and track prediction is employed instead.

- Our comprehensive experimental assessments on the KITTI dataset [38] as well as our tailored CARLA [3] autonomous driving dataset reveal a substantial improvement in tracking performance when EMAP is integrated into the state-of-the-art MOT algorithms.

4.2 Methodology

4.2.1 Camera Motion Projection

Assuming a rectified image in hand, using the pinhole camera model, ego-vehicle translational movement affects the target object location on the image frame by the equations

$$\hat{u}_{trans} = f\frac{d\sin\left(\tan^{-1}\left(\frac{u}{f}\right)\right)}{d\cos\left(\tan^{-1}\left(\frac{u}{f}\right)\right) - \Delta t\dot{D}} \tag{4.4}$$

$$\hat{v}_{trans} = f\frac{d\sin\left(\tan^{-1}\left(\frac{v}{f}\right)\right)}{d\cos\left(\tan^{-1}\left(\frac{v}{f}\right)\right) - \Delta t\dot{D}} \tag{4.5}$$

where f is the camera focal length, d is the Euclidean distance between the target object and the vehicle, $\dot{D}$ is the ego-vehicle translational velocity toward the image plane normal vector, u and v represent the horizontal and vertical pixel coordinates of the target object from the center of the image, while $\hat{u}$ and $\hat{v}$ denote the predicted horizontal and vertical pixel coordinates, respectively. The impact of ego-vehicle rotation on the pixel location of an object is purely horizontal and governed by

$$\hat{u}_{rot} = f\frac{\tan(\Delta t\dot{\psi}) + \frac{u}{f}}{1 - \frac{u}{f}\tan(\Delta t\dot{\psi})} \tag{4.6}$$

where $\dot{\psi}$ is the ego-vehicle yaw angular velocity. Assuming a small time period between each two consecutive frames leads to linearized formulations of the form

$$\hat{u}_{trans} \approx \frac{u\sqrt{u^2+f^2}}{fd}\cdot\dot{D}\Delta t + u \tag{4.7}$$

$$\hat{v}_{trans} \approx \frac{v\sqrt{v^2+f^2}}{fd}\cdot\dot{D}\Delta t + v \tag{4.8}$$

$$\hat{u}_{rot} \approx f\left(1 + \frac{u^2}{f^2}\right) \cdot \dot{\psi}\Delta t + u \tag{4.9}$$

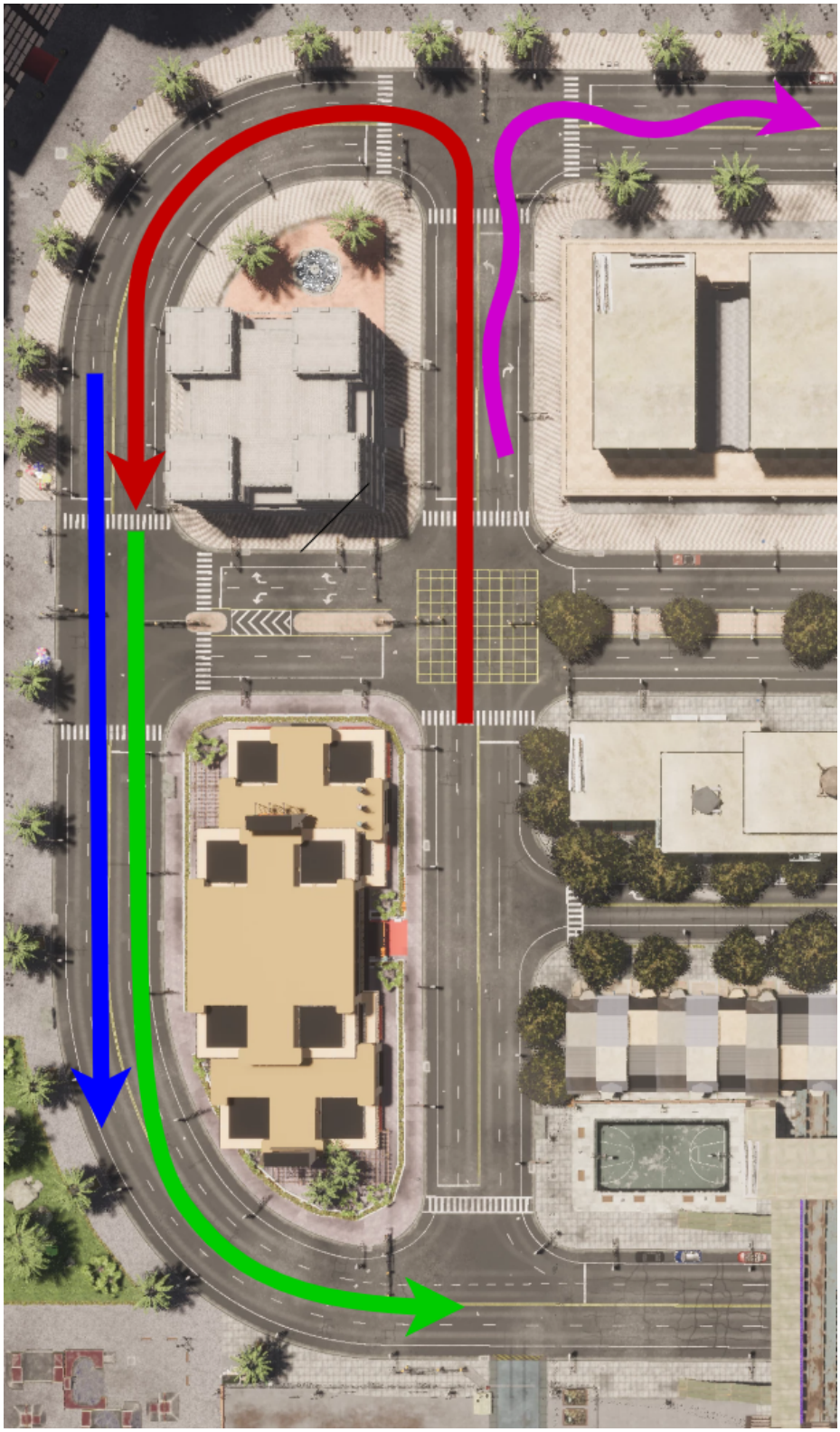

Figure 4.3: Visualization of TOWN #10 in CARLA simulator, featuring four distinct simulation scenarios superimposed on the map. The paths illustrate the diverse trajectories taken in our dataset, capturing a range of scenarios for comprehensive analysis.

4.2.2 Camera Motion Integration with Kalman Filter

Considering the availability of scene depth and camera motion information, the Kalman Filter state definition can now be reformulated to integrate this information into a base model.

Following the similar format used by SORT in equation 4.3, the object's motion is estimated by a constant velocity model; however, the effect of camera motion is decoupled in this motion model.

Each target object is then modeled using the following state definition:

$$\mathbf{x} = [u^l, v^t, u^r, v^b, \dot{x}^l, \dot{y}^t, \dot{x}^r, \dot{y}^b]^\top \tag{4.10}$$

where (u^l, v^t) indicates the top-left corner of a bounding box, and (u^r, v^b) represents the bottom-right corner of that. Unlike SORT, which uses derivatives of (u, v) as noted in 4.3, $\dot{x}^l, \dot{y}^t, \dot{x}^r$, and $\dot{y}^b$ are used. These variables denote the projected horizontal and vertical pixel velocities of the corners of the target object bounding box caused solely by the object's motion in the world frame. Assuming that the movement of the ego-vehicle can be approximated by two independent parameters, i.e., orientation and forward displacement, the state equation of Kalman Filter is now defined as follows:

$$\mathbf{x_{n+1}} = \begin{bmatrix} I_{4\times4} & dtI_{4\times4} \\ 0_{4\times4} & I_{4\times4} \end{bmatrix} \mathbf{x_n} + \begin{bmatrix} G_n^\psi & G_n^D \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{D} \end{bmatrix} dt \tag{4.11}$$

where

$$G_n^{\psi} = \begin{bmatrix} f\left(1+\left(\frac{u_n^l}{f}\right)^2\right) \\ 0 \\ f\left(1+\left(\frac{u_n^r}{f}\right)^2\right) \\ 0 \\ 0_{4\times 1} \end{bmatrix}, G_n^D = \begin{bmatrix} \frac{u_n^l\sqrt{(u_n^l)^2+f^2}}{fd_n} \\ \frac{v_n^t\sqrt{(v_n^t)^2+f^2}}{fd_n} \\ \frac{u_n^l\sqrt{(u_n^l)^2+f^2}}{fd_n} \\ \frac{v_n^b\sqrt{(v_n^b)^2+f^2}}{fd_n} \\ 0_{4\times 1} \end{bmatrix} \tag{4.12}$$

When there exists an associated detection of an object, Kalman Filter estimation phase is used to update the state variables. Then, the prediction is performed to extrapolate the object's position into the subsequent frame.

Figure 4.4 shows an overview of the EMAP module integrated into a DBT multi-object tracking framework. In the depicted figure, the input image originating from the camera is directed into the detection module. Simultaneously, the synchronized odometry data and depth image are routed into the EMAP module. Following the generation of predictions by EMAP for the subsequent location of each existing track, the tracks are then associated with the newly detected objects. Finally, the iteration ends when the tracks are updated with their new associated detection, if available.

4.2.3 Justification for isolating the camera motion

As mentioned earlier in the Introduction section, most Kalman Filter-based DBT algorithms approximate the motion of an object by constant velocity models independent of other objects and camera motion. However, this assumption can easily be violated in scenarios where camera movements introduce unexpected disturbances

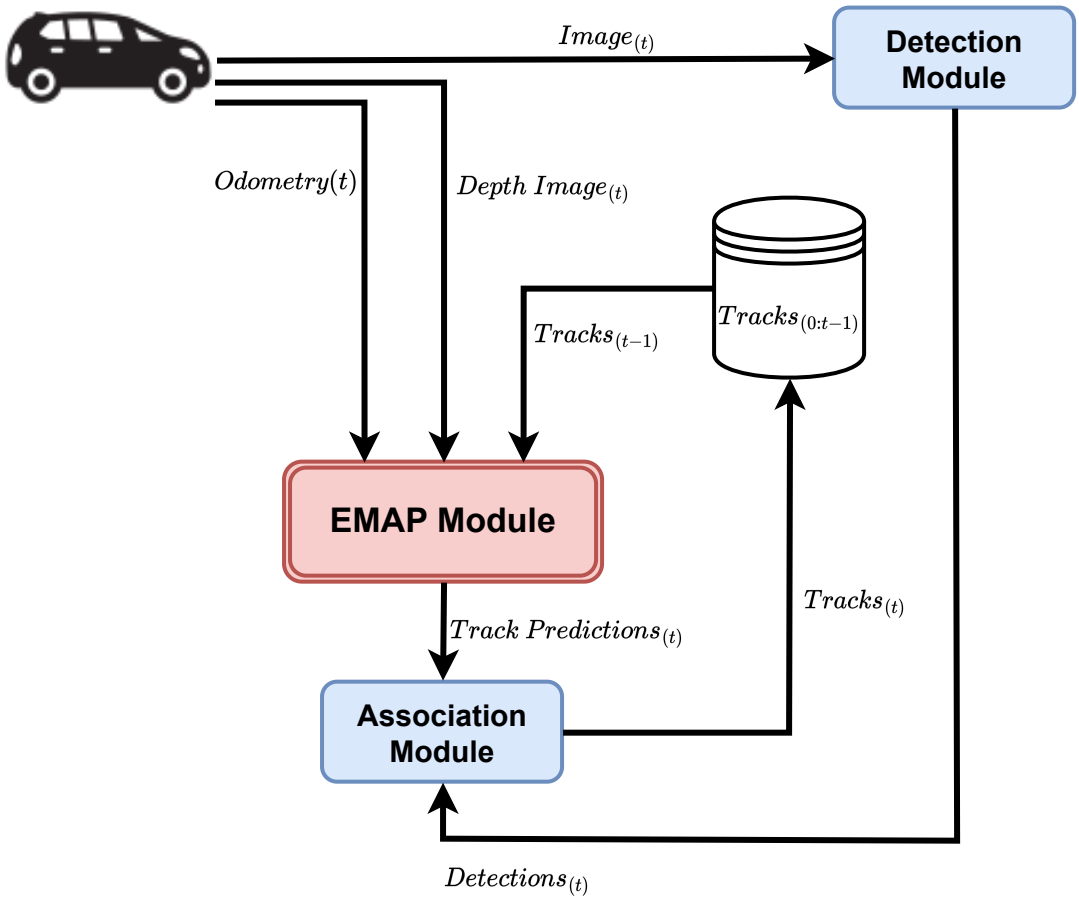

Figure 4.4: Diagram illustrating the three phases of a detection-based multi-object tracking algorithm. The figure highlights the integration of the EMAP module within the system.

to the bounding box location within the image frame. The situation is intensified in the presence of detection gaps, as the constant velocity model fails to accurately predict the track of the trajectories. Consequently, inaccurate predictions give rise to frequent identity switches, ultimately compromising the overall MOT performance. To address this problem, in the EMAP module, camera motion is taken into account separately from the object motion model.

Algorithm 2 Integrating EMAP with a sample base tracker

Input: Existing tracks buffer $\mathcal{T}$, New Detections $\mathcal{D}$, Camera translational and rotational motion (C_t, C_r), Depth image I

Output: Updated tracks buffer $\mathcal{T}^{\text{updated}}$

Initialize $\mathcal{T}^{\text{pred}} \leftarrow \varnothing$
for track t in $\mathcal{T}$ **do**
  if `RotationOnly` **then**
    $\mathcal{T}^{\text{pred}} \leftarrow \mathcal{T}^{\text{pred}} \cup$ `Pred(`t`,` C_r`)`
  else if `TranslationOnly` **then**
    $\mathcal{T}^{\text{pred}} \leftarrow \mathcal{T}^{\text{pred}} \cup$ `Pred(`t`,` C_t`,` I`)`
  else
    $\mathcal{T}^{\text{pred}} \leftarrow \mathcal{T}^{\text{pred}} \cup$ `Pred(`t`,` C_t`,` C_r`,` I`)`
  end if
end for
$(\mathcal{T}^{\text{matched}}, \mathcal{D}^{\text{matched}}) \leftarrow$ `Associate(`$\mathcal{T}^{\text{pred}}, \mathcal{D}$`)`
$\mathcal{T}^{\text{unmatched}} \leftarrow \mathcal{T} \setminus \mathcal{T}^{\text{matched}}$
$\mathcal{D}^{\text{unmatched}} \leftarrow \mathcal{D} \setminus \mathcal{D}^{\text{matched}}$
`UpdateKFstates(`$\mathcal{T}^{\text{matched}}$`,` $\mathcal{D}^{\text{matched}}$`)`
Identify $\mathcal{T}^{\text{lost}}$ from the $\mathcal{T}^{\text{unmatched}}$
Create $\mathcal{T}^{\text{new}}$ from the $\mathcal{D}^{\text{unmatched}}$
$\mathcal{T}^{\text{updated}} \leftarrow \mathcal{T}^{\text{matched}} \cup \mathcal{T}^{\text{new}} \cup \{\mathcal{T}^{\text{unmatched}} \setminus \mathcal{T}^{\text{lost}}\}$

4.3 Experiments

4.3.1 Dataset

KITTI

The KITTI dataset serves as a cornerstone in benchmarking computer vision algorithms for autonomous driving tasks. It consists of real-world data captured from a car equipped with sensors such as LiDAR, camera, and GPS. For the experiments, the KITTI training dataset is used, which consists of 21 sequences encompassing diverse driving scenarios such as urban, highway, and rural environments. For the evaluation, focus is placed on the 'Car' and 'Pedestrian' classes, for which detected bounding

boxes are taken from PermaTrack [59] and also the YOLOv8 object detector [24].

CARLA Simulation Dataset

In addition to the KITTI dataset, experiments were conducted using a custom-generated dataset within the CARLA autonomous driving simulator. The CARLA simulator offers a realistic virtual environment for testing autonomous driving algorithms, allowing for controlled and diverse scenarios. In the dataset, there are four sequences, each representing distinct scenarios. In Scenario #1, a car navigates a straight road. Scenario #2 involves the car initiating movement from a static position, traveling along a straight path, encountering a curve, and concluding the video upon completing the curve. Scenario #3 features the car starting from a stationary position, progressing along a straight trajectory, reaching a curve, and coming to a stop afterward. In Scenario #4, the vehicle travels a path with abrupt movements. Figure 4.3 illustrates the paths on the CARLA map. CARLA integrates seamlessly with MOT algorithms through the Robot Operating System (ROS) [5], enabling communication between Python and CARLA. Synchronized messages, including RGB images, depth cloud (aligned with RGB), ego-vehicle odometry, and automatically generated ground truth tracking bounding boxes from CARLA are streamed to Python via ROS for integrated evaluation.

4.3.2 Evaluation Metrics

HOTA [60] has been selected as the primary metric for its ability to strike a more equitable balance between the accuracy of both object detection and association in multi-object tracking. Our emphasis extends to Association Accuracy (AssA) for evaluating association performance, along with IDF1 as an additional metric in the same context. Metrics such as MOTA predominantly gauge detection performance,

and to ensure fairness, they are utilized only when all methods rely on identical detections for tracking, a condition that is met in the evaluation.

4.3.3 Baseline SORT-based Algorithms

In order to evaluate the impact of EMAP on existing MOT algorithms, four state-of-the-art SORT-based algorithms were selected, and EMAP was integrated with their prediction module to evaluate the effect of EMAP on their performance. The algorithms selected as baselines are OC-SORT, Deep OC-SORT, ByteTrack, and BOT-SORT. All four baselines are Kalman Filter-based algorithms. However, in BOT-SORT authors use image registration to approximate the camera motion and compensate for the effect of this motion.

4.4 Results

In this section, the performance of the solution on both the CARLA and the KITTI datasets is presented. Table 4.1 displays the HOTA scores and identity switch counts for each sequence in the CARLA simulation dataset. Notably, EMAP significantly impacts Sequence #4, characterized by substantial vehicle rotational and translational movements. This sequence shows a decrease in identity switches and an increase in HOTA values for all trackers, highlighting EMAP's effectiveness in complex motion scenarios.

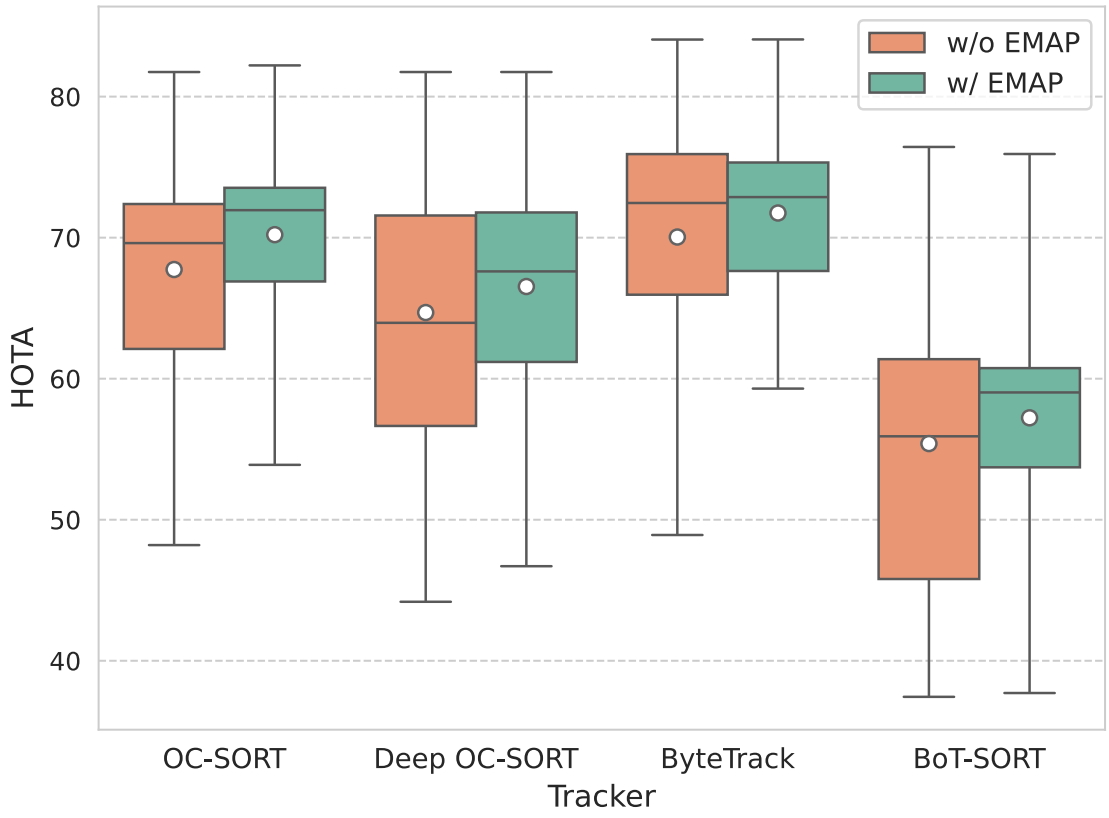

Figure 4.5: Distribution of Higher Order Tracking Accuracy (HOTA) scores across four advanced multi-object tracking algorithms (OCSORT, Deep OCSORT, ByteTrack, and BotSort) with and without the EMAP module over 21 sequences of the KITTI-train dataset.

Table 4.2 presents the average metric values across the 21 sequences of the KITTI train dataset when detections are taken from PermaTrack. EMAP notably reduces the number of identity switches across all four baselines, with the most significant impact observed in OC-SORT, where it decreases by 76%. Furthermore, EMAP integration improves various metrics such as HOTA, MOTA, IDF1, FP, FN, AssA, and AssR in all four baseline trackers. Figure 4.6 illustrates that integration of the EMAP module notably decreases identity switches and enhances HOTA scores compared to baseline performance. Additionally, it demonstrates that EMAP contributes to the improved robustness of baseline trackers, as evidenced by the reduced variance in IDSW and

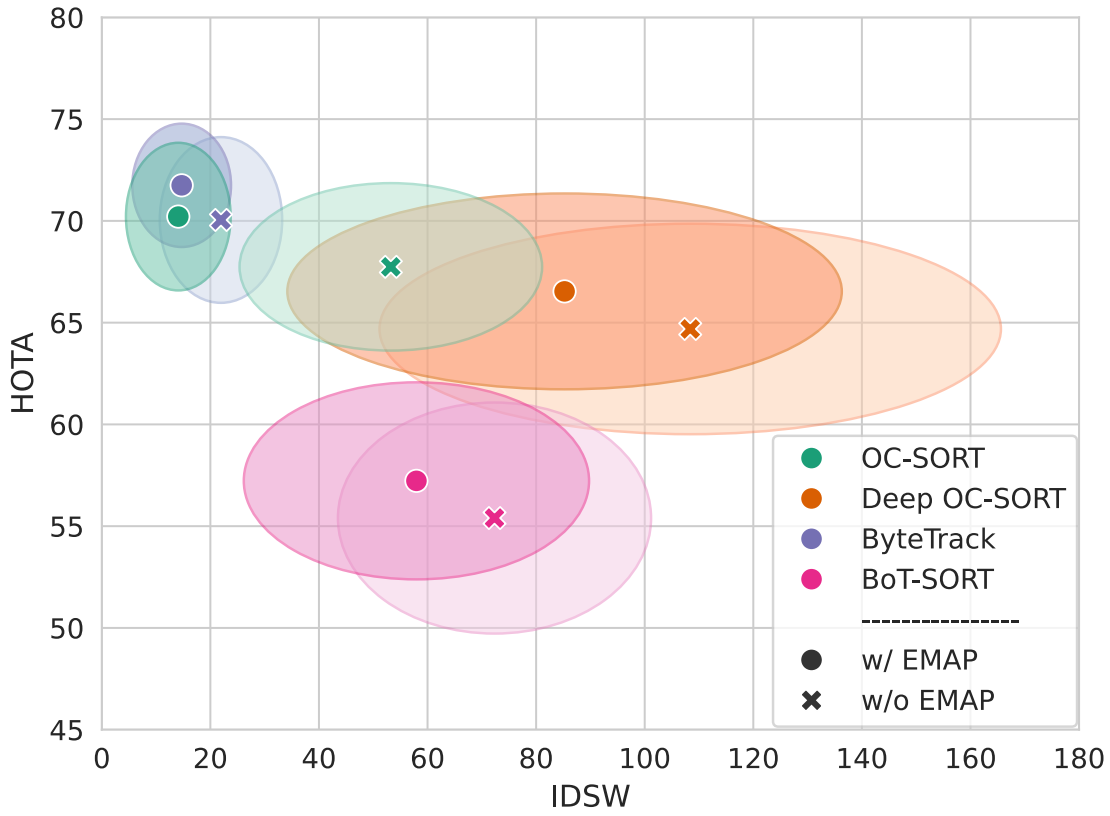

Figure 4.6: HOTA vs IDSW comparisons of OC-SORT, Deep OC-SORT, ByteTrack, and BoT-SORT on 21 KITTI train sequences with or without EMAP module. The height and width of the ellipses are the standard deviation of the distribution.

HOTA across all sequences.

In Table 4.3, the evaluation on the KITTI train dataset using YOLOv8x as the object detector is presented. Despite the fact that YOLOv8x exhibits weaker performance compared to PermaTrack indicated by higher missed detections and lower detection accuracy, the addition of EMAP still positively affects the performance of all baseline trackers.

Figure 4.5 comparing HOTA scores for OCSORT, Deep OCSORT, ByteTrack, and BotSort demonstrates that adding the EMAP module significantly improves tracking performance. Specifically, it shows an increase in mean HOTA scores and a decrease in their variance for all four algorithms with EMAP, indicating enhanced

accuracy and consistency in tracking outcomes. This highlights EMAP's role in bolstering multi-object tracking effectiveness. Figure 4.7 highlights a decrease in identity switches (IDSW) for all four trackers—OCSORT, Deep OCSORT, ByteTrack, and BotSort—upon integrating the EMAP module, with OCSORT experiencing the most significant reduction. This demonstrates EMAP's effectiveness in enhancing tracking continuity and reducing identity errors, thus improving the reliability of the tracking algorithms.

To assess the individual and combined impact of the translation and rotational submodules, an ablation study is conducted comparing the results of EMAP with each submodule added separately and together, as shown in Table 4.4. The largest reduction in the number of identity switches occurs when both submodules are active. However, the maximum HOTA is attained when only the rotational module is activated.

Table 4.1: Performance comparison of MOT algorithms on our CARLA dataset split by sequence with YOLOv8x as the object detector. The best results are shown in **bold**.

Tracker	Sequence #1		Sequence #2		Sequence #3		Sequence #4	
	HOTA↑	IDs↓	HOTA↑	IDs↓	HOTA↑	IDs↓	HOTA↑	IDs↓
OC-SORT [20]	36.15	5	24.84	11	36.3	15	24.09	19
OC-SORT + EMAP [20]	**36.24**	**1**	**25.06**	**3**	**37.04**	15	**29.09**	**7**
BoT-SORT [41]	**35.08**	7	21.88	35	**32.64**	30	23.44	24
BoT-SORT + EMAP [41]	34.94	7	**22.14**	35	31.65	30	**24.54**	**18**
Deep OC-SORT [40]	**35.71**	**4**	**23.65**	**55**	**36.33**	**36**	25.17	67
Deep OC-SORT + EMAP [40]	35.55	9	20.75	129	35.55	61	**26.66**	**38**
ByteTrack [19]	34.94	0	21.14	1	34.26	5	25.71	8
ByteTrack + EMAP	**25.05**	0	**21.7**	**0**	**34.91**	**4**	**28.39**	**2**

4.4.1 Visualization Results

To showcase the impact of the EMAP module, a visualization is presented of the Bytetrack MOT algorithm operating within the CARLA simulator under a scenario where detection is lost. As depicted in Figure 4.1 (a), the prediction module of the

Table 4.2: Average performance (on 21 sequences) on KITTI-train dataset, detections are taken from PermaTrack. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
OC-SORT [20]	67.74	64.53	78.24	709.33	234.19	53.24	68.38	79.34
OC-SORT + EMAP	**70.21**	**67.87**	**82.17**	**662.19**	**187.05**	**14.10**	**73.11**	**85.32**
Deep OC-SORT [40]	64.69	62.59	74.20	893.67	418.52	108.38	62.84	74.41
Deep OC-SORT + EMAP	**66.54**	**64.38**	**76.37**	**863.14**	**388.00**	**85.24**	**66.22**	**77.80**
ByteTrack [19]	70.05	74.98	84.60	339.10	305.00	21.95	72.62	80.28
ByteTrack + EMAP	**71.75**	**75.82**	**85.78**	**324.33**	**295.33**	**14.71**	**74.71**	**81.73**
BoT-SORT [41]	55.40	47.74	65.89	602.95	687.29	72.33	56.36	62.33
BoT-SORT + EMAP	**57.23**	**50.11**	**68.49**	**578.29**	**670.76**	**57.95**	**59.58**	**66.05**

Table 4.3: Average performance (on 21 sequences) on KITTI-train dataset, detections are taken from YOLOv8. The best results are shown in **bold**.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP↓	FN↓	IDs↓	AssA↑	AssR↑
OC-SORT [20]	39.35	37.0	53.04	163.14	1187.9	18.86	46.16	48.82
OC-SORT + EMAP	**40.28**	**38.0**	**54.4**	168.05	**1179.95**	**12.29**	**47.68**	**50.25**
Deep OC-SORT [40]	38.22	35.77	50.08	270.67	1236.38	69.38	42.21	45.36
Deep OC-SORT + EMAP	**39.6**	**37.2**	**52.88**	**225.48**	**1191.19**	**41.33**	**45.15**	**48.41**
ByteTrack [19]	33.48	31.29	44.66	102.76	1368.76	14.24	42.11	44.2
ByteTrack + EMAP	**33.78**	**31.32**	**45.81**	**83.95**	**1356.81**	**11.9**	**42.69**	44.12
BoT-SORT [41]	25.16	23.5	34.57	138.0	1584.1	46.05	29.17	29.81
BoT-SORT + EMAP	**28.06**	**25.49**	**39.21**	**123.7**	**1425.65**	**35.2**	**33.82**	**34.55**

Table 4.4: Ablation on KITTI dataset with Translational and Rotational submodules. The best results are shown in **bold**.

Method	Tracker	HOTA↑	MOTA↑	IDF1↑	IDs↓
Baseline	OC-SORT	67.74	64.53	78.24	53.24
	Deep OC-SORT	64.69	62.59	74.2	108.38
	ByteTrack	70.05	74.98	84.6	21.95
	BoT-SORT	55.4	47.74	65.89	72.33
Translational-Only EMAP	OC-SORT	71.45	74.59	83.35	24.38
	Deep OC-SORT	65.33	63.08	74.91	99.43
	ByteTrack	70.25	74.77	84.4	23.0
	BoT-SORT	55.28	48.65	65.58	76.71
Rotational-Only EMAP	OC-SORT	**72.63**	75.1	85.05	15.43
	Deep OC-SORT	65.57	63.72	75.62	97.14
	ByteTrack	71.18	**75.86**	85.69	15.1
	BoT-SORT	55.41	48.65	65.59	85.81
EMAP	OC-SORT	70.21	67.87	82.17	**14.1**
	Deep OC-SORT	66.54	64.38	76.37	85.24
	ByteTrack	71.75	75.82	**85.78**	14.71
	BoT-SORT	57.23	50.11	68.49	57.95

vanilla ByteTrack struggles to accurately anticipate the position of the target after the detection becomes unavailable (denoted by black bounding boxes). In contrast,

Figure 4.1 (b) illustrates the enhanced performance achieved by incorporating our module. The predicted bounding box location aligns more closely with the actual location, underscoring the efficacy of the ego-motion aware prediction module.

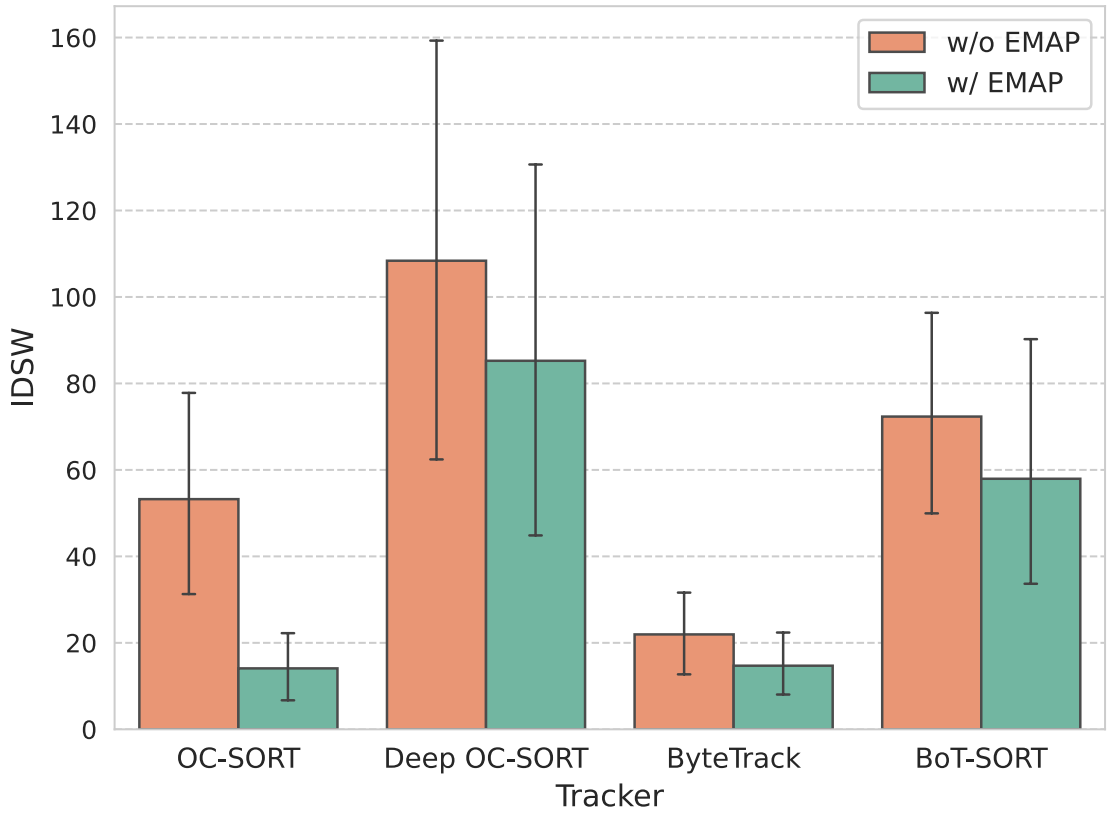

Figure 4.7: Distribution of Identity Switches (IDSW) values across four advanced multi-object tracking algorithms (OCSORT, Deep OCSORT, ByteTrack, and Bot-Sort) with and without the EMAP module over 21 sequences of the KITTI-train dataset.

4.5 Conclusions and Future Work

In this work, the performance of detection-based MOT algorithms is enhanced by leveraging both the motion of the camera and depth information. Our contribution, the Ego-Motion Aware Prediction (EMAP) module, serves as a predictive component seamlessly integrated into DBT multi-object tracking algorithms. EMAP adeptly re-

jects disturbances introduced by camera motion, thereby boosting the reliability of the object motion model. Through the reformulation of the Kalman Filter, EMAP effectively decouples the impact of the rotational and translational velocities of the camera from the target object location on the image frame. In our experiments on KITTI MOT dataset, EMAP significantly enhances the performance of the base trackers, evident in HOTA and notable reductions in the number of identity switches. Our in-depth investigation in CARLA simulator further reveals the efficacy of EMAP where ego-camera motion dominantly impacts the target location. Our module depicts how the reliability of existing MOT algorithms can be improved in real-world scenarios especially autonomous driving.

As a future direction, the retrieval of motion information using Visual Odometry algorithms and the extraction of depth maps solely from RGB cameras are aimed to be explored, with the ultimate goal of extending the versatility of the module to systems relying exclusively on an RGB camera as their primary sensor.

Chapter 5

Conclusions and Future Work

5.1 Summary

This thesis has presented an in-depth study of Ego-motion Aware Multi-object Tracking (MOT) within a ROS-based framework, aiming to address the complexities and challenges inherent in accurate and efficient object tracking in dynamic environments, particularly in the context of autonomous driving and robotics applications. The research introduced the Ego-Motion Aware Target Prediction (EMAP) module, a novel component designed to enhance the accuracy of tracking algorithms by integrating ego-motion awareness into the prediction phase of MOT. This integration allows for a more nuanced understanding of object dynamics relative to the observer's movement, significantly reducing identity switches and improving tracking robustness across various scenarios.

Three primary contributions have been underscored throughout this work:

1. **Ego-Motion Aware Target Prediction Module:** The development and integration of the EMAP module into existing SORT-based trackers demonstrated a marked improvement in MOT performance. By accounting for the

ego-motion of the tracking platform, this module significantly reduced identity switches and enhanced tracking accuracy, as validated on standard datasets like KITTI and custom scenarios within the CARLA simulator.

2. **Self-supervised MOT Algorithm with Adaptive Track-Matching:** A novel approach to MOT was explored through the development of a self-supervised algorithm that adapts its track-matching threshold based on the self-supervisory signal. This methodology broadens the applicability and flexibility of MOT algorithms, enabling them to learn and adapt without extensive manually labeled datasets.

3. **Comprehensive MOT Framework:** The establishment of a structured framework for the development, testing, and evaluation of MOT algorithms has been a crucial outcome of this research. By providing a seamless interface with both simulated and real-world data, this framework facilitates rigorous algorithm testing and refinement, propelling advancements in the MOT domain.

5.2 Limitations

This research, while pioneering in the realm of Multi-object Tracking (MOT) within a ROS-based framework, acknowledges certain limitations that warrant further exploration:

- **Simplistic Motion Model:** EMAP primarily assumes a constant velocity model for target objects, which may oversimplify the complexities of real-world object dynamics. This model, while effective for a broad range of scenarios, may not accurately capture the nuanced movements observed in highly dynamic environments. Exploring more sophisticated motion models could potentially enhance the predictive accuracy and robustness of the tracking system.

- **Reliance on Single Data Sources:** The current implementation of the Ego-motion Aware Tracking system depends on a singular source of input for each type of data, such as depth information, camera motion, and object detection. This design leaves the system vulnerable to inaccuracies or corruption in the input data, potentially compromising the tracking performance. Incorporating redundancy or multiple independent data sources could mitigate this risk, ensuring more reliable system operation under varying conditions.

- **Self-Supervised Learning Model Adaptability:** The self-supervised algorithm proposed for adaptive track-matching currently utilizes a probabilistic distribution to determine the matching threshold. While this approach has shown promise, employing machine learning models such as a Siamese network to adjust the threshold dynamically may offer improved adaptability and generalizability across more complex tracking scenarios. This could further refine the algorithm's performance, especially in environments where object appearances and movements exhibit significant variability.

These limitations highlight areas for future research and development, aiming to further refine and enhance the capabilities of multi-object tracking systems. Addressing these concerns would not only improve the accuracy and reliability of tracking in complex scenarios but also extend the applicability of these algorithms to a wider array of real-world applications.

5.3 Future Work

The research conducted in this thesis lays the groundwork for several promising directions for future exploration within the domain of Ego-motion Aware Multi-Object Tracking (MOT). Addressing the identified limitations and building upon the current

contributions, the following areas have been earmarked for subsequent development:

- **Incorporation of Complex Motion Models:** To overcome the limitations posed by the constant velocity target model, future efforts will focus on integrating more sophisticated motion models into the EMAP module. These advanced models will aim to more accurately reflect the intricate dynamics of object movements, particularly in environments characterized by rapid and unpredictable motion patterns. Exploring non-linear and adaptive motion models could significantly enhance the predictive accuracy and robustness of the tracking system.

- **Support for Multiple Data Sources:** Recognizing the vulnerability of the system to inaccuracies in input data, a key area of future work will involve augmenting the system's architecture to accommodate multiple independent sources of data. This enhancement will provide redundancy, reduce the risk of system failure due to corrupted data, and ensure more reliable tracking under diverse operational conditions. Implementing a fusion mechanism to intelligently integrate data from various sources will be critical to realizing this objective.

- **Integration with TrackFormer:** An exciting avenue for extending the capabilities of the EMAP module involves its application to end-to-end transformer-based multi-object tracking algorithms, such as TrackFormer. By adapting EMAP for use with TrackFormer, the aim is to leverage the transformer architecture's inherent strengths in handling complex spatial-temporal relationships. This integration promises to elevate the performance of transformer-based tracking algorithms by enhancing their awareness of ego-motion and improving their ability to maintain consistent object identities across sequences.

Chapter 6

Additional Information

6.1 Preface

Chapter 4 of this thesis corresponds to work that has been submitted to the IROS 2024 conference and subsequently published as a pre-print on ArXiv. The paper, entitled "Ego-Motion Aware Target Prediction Module for Robust Multi-Object Tracking," appears under the citation [61]. As the lead researcher, I spearheaded the conceptualization, algorithm design, programming, visualization, and manuscript preparation. Mohammad Jani, a fellow graduate student at the ACIS lab at the University of Victoria, contributed significantly to algorithm development, programming, visualization, and manuscript writing. Amir M. Soufi provided invaluable assistance with the visualizations and participated actively in the ideation process. Homayoun Najjaran, my supervisor, provided oversight and critical feedback during the revision stages of the manuscript.

Bibliography

[1] "aws-robotics/aws-robomaker-bookstore-world," Feb. 2024. original-date: 2019-03-02T01:25:46Z.

[2] "MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking | SpringerLink."

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[4] D. Bassermann, "Standards."

[5] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation Workshop on Open Source Software*, 2009.

[6] Clearpath Robotics, *Jackal UGV User Manual*. Clearpath Robotics, Ontario, Canada, 2023. Accessed: 2023-04-26.

[7] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint*

conference on Artificial intelligence - Volume 2, IJCAI'81, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., Aug. 1981.

[8] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE international conference on computer vision*, pp. 4310–4318, 2015.

[9] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Adaptive correlation filters with long-term and short-term memory for object tracking," *International Journal of Computer Vision*, vol. 126, pp. 771–796, 2018.

[10] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 8, pp. 1561–1575, 2016.

[11] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6638–6646, 2017.

[12] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pp. 472–488, Springer, 2016.

[13] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 8, pp. 1561–1575, 2016.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking

with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[15] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 2544–2550, IEEE, 2010.

[16] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang, "Object tracking via dual linear structured svm and explicit feature map," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4266–4274, 2016.

[17] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, pp. 850–865, Springer, 2016.

[18] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, Sept. 2016. ISSN: 2381-8549.

[19] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-object Tracking by Associating Every Detection Box," in *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, (Berlin, Heidelberg), pp. 1–21, Springer-Verlag, Oct. 2022.

[20] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Vancouver, BC, Canada), pp. 9686–9696, IEEE, June 2023.

[21] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "StrongSORT: Make DeepSORT Great Again," Feb. 2023. arXiv:2202.13514 [cs].

[22] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," Mar. 2017. arXiv:1703.07402 [cs].

[23] S. Han, P. Huang, H. Wang, E. Yu, D. Liu, and X. Pan, "MAT: Motion-aware multi-object tracking," *Neurocomputing*, vol. 476, pp. 75–86, Mar. 2022.

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

[25] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10778–10787, 2020.

[26] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[27] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies," in *2017 IEEE International Conference on Computer Vision (ICCV)*, (Venice), pp. 300–311, IEEE, Oct. 2017.

[28] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, (San Francisco, California, USA), pp. 4225–4232, AAAI Press, Feb. 2017.

[29] N. L. Baisa, "Occlusion-robust online multi-object visual tracking using a GM-PHD filter with CNN-based re-identification," *Journal of Visual Communication and Image Representation*, vol. 80, p. 103279, Oct. 2021.

[30] Q. Liu, D. Chen, Q. Chu, L. Yuan, B. Liu, L. Zhang, and N. Yu, "Online multi-object tracking with unsupervised re-identification learning and occlusion estimation," *Neurocomputing*, vol. 483, pp. 333–347, Apr. 2022.

[31] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pp. 868–884, Springer, 2016.

[32] G. Wang, Y. Wang, H. Zhang, R. Gu, and J.-N. Hwang, "Exploit the Connectivity: Multi-Object Tracking with TrackletNet," in *Proceedings of the 27th ACM International Conference on Multimedia*, (Nice France), pp. 482–490, ACM, Oct. 2019.

[33] H. Liu, T. Xu, and X. Wu, "MMOT: Motion-Aware Multi-Object Tracking with Optical Flow," in *Proceedings of the 2022 11th International Conference on Computing and Pattern Recognition*, (Beijing China), pp. 115–120, ACM, Nov. 2022.

[34] B. Shuai, A. Berneshawi, X. Li, D. Modolo, and J. Tighe, "SiamMOT: Siamese Multi-Object Tracking," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Nashville, TN, USA), pp. 12367–12377, IEEE, June 2021.

[35] K.-C. Huang, M.-H. Yang, and Y.-H. Tsai, "Delving into motion-aware matching for monocular 3d object tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6909–6918, 2023.

[36] G. D. Evangelidis and E. Z. Psarakis, "Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1858–1865, Oct. 2008. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[37] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[38] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[39] P. Sun, J. Cao, Y. Jiang, Z. Yuan, S. Bai, K. Kitani, and P. Luo, "Dancetrack: Multi-object tracking in uniform appearance and diverse motion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20993–21002, 2022.

[40] G. Maggiolino, A. Ahmad, J. Cao, and K. Kitani, "Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification," Feb. 2023. arXiv:2302.11813 [cs].

[41] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust Associations Multi-Pedestrian Tracking," July 2022. arXiv:2206.14651 [cs].

[42] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple Object Tracking: A Literature Review," *Artificial Intelligence*, vol. 293, p. 103448, Apr. 2021. arXiv:1409.7618 [cs].

[43] S. Devi, P. Malarvezhi, R. Dayana, and K. Vadivukkarasi, "A Comprehensive Survey on Autonomous Driving Cars: A Perspective View," *Wireless Personal Communications*, vol. 114, pp. 2121–2133, Oct. 2020.

[44] R. C. Joshi, M. Joshi, A. G. Singh, and S. Mathur, "Object Detection, Classification and Tracking Methods for Video Surveillance: A Review," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pp. 1–7, Dec. 2018. ISSN: 2642-7354.

[45] D. Xie, W. Hu, T. Tan, and J. Peng, "A multi-object tracking system for surveillance video analysis," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 4, pp. 767–770 Vol.4, Aug. 2004. ISSN: 1051-4651.

[46] X. Lin, L. Girin, and X. Alameda-Pineda, "Unsupervised Multiple-Object Tracking with a Dynamical Variational Autoencoder," Feb. 2022. arXiv:2202.09315 [cs].

[47] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. USA: Prentice Hall Press, 3rd ed., 2009.

[48] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[49] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[51] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with

a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, Sept. 2017. ISSN: 2381-8549.

[52] N. L. Baisa, "Occlusion-robust online multi-object visual tracking using a GM-PHD filter with CNN-based re-identification," *Journal of Visual Communication and Image Representation*, vol. 80, p. 103279, Oct. 2021.

[53] J. Lohn-Jaramillo, L. Ray, R. Granger, and E. Bowen, "Clustertracker: An Efficiency-Focused Multiple Object Tracking Method," May 2022.

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

[55] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. d. P. Veronese, T. Oliveira-Santos, and A. F. D. Souza, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[56] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking Without Bells and Whistles," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (Seoul, Korea (South)), pp. 941–951, IEEE, Oct. 2019.

[57] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.

[58] P. Dendorfer, H. Rezatofighi, A. Milan, J. Q. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taix'e, "MOT20: A benchmark for multi object tracking in crowded scenes," *ArXiv*, Mar. 2020.

[59] P. Tokmakov, J. Li, W. Burgard, and A. Gaidon, "Learning to track with object permanence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10860–10869, 2021.

[60] J. Luiten, A. Ossep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixe, and B. Leibe, "HOTA: A Higher Order Metric for Evaluating Multi-object Tracking," *International Journal of Computer Vision*, vol. 129, pp. 548–578, Feb. 2021.

[61] N. Mahdian, M. Jani, A. M. S. Enayati, and H. Najjaran, "Ego-motion aware target prediction module for robust multi-object tracking," *arXiv preprint arXiv:2404.03110*, 2024.