TUM School of Computation, Information and Technology
Technical University of Munich

Bachelor's Thesis in Informatics

3D Tracking on Roadside LiDARs

3D Tracking anhand von straßenseitigen LiDARs

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Walter Zimmer, M.Sc.

Author Vitus Becker

Date February 15, 2024 in Munich

Disclaimer

I confirm that this Bachelor's Thesis is my own work and I have documented all sources and material used.

Munich, February 15, 2024

(Vitus Becker)

Abstract

This thesis explores 3D Multi-Object Tracking on Roadside LiDARs within the AUTOtech.agil project, the successor of the Providentia++ project. The goal is to improve tracking performance in urban environments. Using PolyMOT as a baseline, known for its success in the nuScenes tracking challenge, this work adds features like dynamic association threshold determination and an angular difference penalty to enhance robustness. The evaluation on the TUM Traffic Intersection dataset shows that PolyMOT achieves significant improvements of over 16% in HOTA and over 13% in MOTA. Although the added features only slightly improve MOTA and HOTA by 1%, qualitative analysis underscores their impact. PolyMOT demonstrates remarkable performance in managing complex traffic scenarios and mitigating the impact of noisy detections, particularly when tracking smaller objects. However, SORT3D outperforms PolyMOT clearly in inference speed with 700-800 FPS compared to 30 FPS. Proposed enhancements for future work include integrating learning-based approaches and incorporating map information. Overall, the thesis validates PolyMOT's selection, highlights the added features' impact, and suggests areas for future research.

Zusammenfassung

In dieser Arbeit wird 3D-Multi-Objekt Tracking anhand von straßenseitigen LiDARs im Rahmen des AUTOtech.agil-Projekts, dem Nachfolger des Providentia++-Projekts, untersucht. Ziel ist es, die Tracking Performanz in urbanen Umgebungen zu verbessern. Auf der Grundlage von PolyMOT, das für seinen Erfolg bei der nuScenes Tracking Challenge bekannt ist, werden in dieser Arbeit Funktionen wie die dynamische Bestimmung von Assoziationsgrenzwerten und eine Winkeldifferenzstrafe hinzugefügt, um die Robustheit des Tracking Systems zu verbessern. Die Auswertung auf dem TUM Traffic Intersection Dataset zeigt, dass PolyMOT signifikante Verbesserungen von über 16% in HOTA und über 13% in MOTA erreicht. Auch wenn die hinzugefügten Funktionen die Metriken MOTA und HOTA nur geringfügig um 1% verbessern, unterstreicht die qualitative Analyse die Wirkung der hinzugefügten Features. PolyMOT schneidet beim Tracking in komplexen Verkehrssituationen und der Kompensation verrauschter Detections sehr gut ab, vor allem beim Tracking von kleineren Objekten. Die Inferenzgeschwindigkeit von SORT3D übertrifft jedoch die von PolyMOT klar, mit 700-800 FPS im Vergleich zu 30 FPS. Vorgeschlagene Verbesserungen für zukünftige Arbeiten beinhalten die Integration von lernbasierten Ansätzen und die Einbeziehung von Umgebungsinformationen. Insgesamt bestätigt die Arbeit die Wahl von PolyMOT, hebt die Auswirkungen der hinzugefügten Funktionen hervor und schlägt Bereiche für zukünftige Verbesserungsmöglichkeiten vor.

Contents

Chapter 1

Introduction

In recent years, the field of autonomous driving has experienced remarkable progress, holding the potential to transform transportation by reducing or eliminating the requirement for human involvement in vehicle control. Enabled by the latest technology, new cars from BMW or Mercedes-Benz can perform overtaking maneuvers on highways without any help from the driver behind the steering wheel using the Automatic Lane Change assistence system, see here. In the United States, there are even driverless taxis services like Waymo One that can be ordered via smartphone like an average Uber driver, with the difference that there is no driver sitting inside.

This thesis also falls within the field of autonomous driving research. It aims to contribute to the ongoing advancements in autonomous driving technology by addressing critical challenges in multi-object tracking. This chapter introduces the thesis by first explaining the motivation and context of the work, then defining the concrete task that has to be realized in the course of the work, followed by a quick outline of the complete thesis explaining the consecutively conducted steps.

1.1 Motivation and Context

One of the critical challenges in achieving fully autonomous vehicles is the reliable detection and tracking of multiple objects in the vehicle's surroundings. As autonomous vehicles navigate complex urban environments, accurately tracking and anticipating the movements of pedestrians, vehicles, and other objects becomes essential for ensuring safety and applicability. Improving and enabling the possibility of autonomous driving was also the target of the Providentia++ project. In the context of this project, the team created a digital twin to replicate the traffic conditions at a section of the Autobahn A9 near Garching (Munich). The specific objective is to facilitate the exchange of traffic-related information among all participants in particular situations. This thesis is part of the successor project AUTOtech.agil, which aims to improve the existing system and, in this way, enable it to be also applicable in more urban areas. For that reason, cameras and LiDAR sensors collect data about the traffic at an urban intersection. This data is then processed to deliver helpful information about traffic participants and their interplay. LiDAR sensors deliver 3D data of the environment in the form of point clouds which are unordered sets of measured 3D points in space, see Figure 1.1. In contrast to camera data in the form of images, LiDAR data has the advantage of being more precise and delivering direct 3D information for which camera images first would have to be processed somehow. In this project, we utilize data recorded from a roadside infrastructure perspective rather than a vehicle perspective, offering several advantages. Data recorded in vehicle-view typically contains many occlusions since the sensors capture their data at the

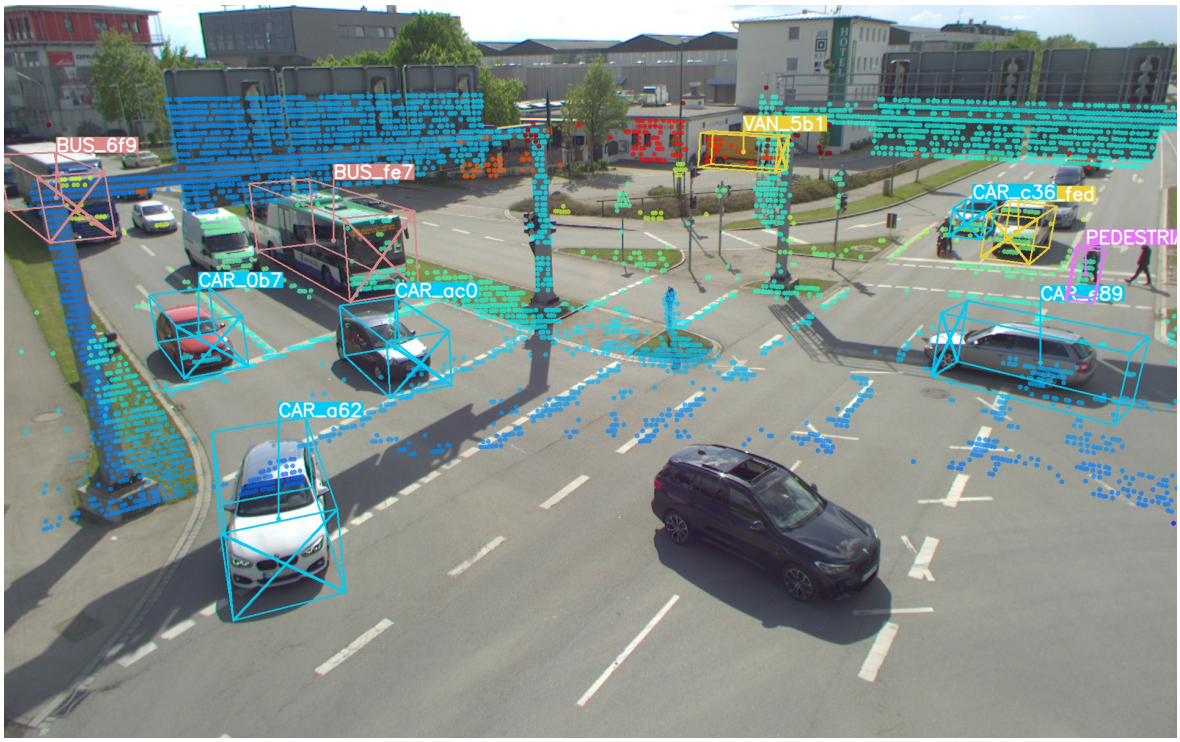


Figure 1.1: This figure presents an illustration of captured image data and projected point cloud data at the mentioned intersection. The visible points represent the 3D structure of the point cloud, projected onto the image for visualization.

same height where all other traffic participants are, while infrastructure-view data typically is recorded from a top-down perspective, decreasing the number of occlusions. Furthermore, infrastructure-view data allows for the incorporation of information about the environment into computations or interpretations. The reason for that is the fixed position of the sensors and cameras, leading to improved performance.

1.2 Problem Statement

Using the data mentioned in the paragraph above, performing tasks like object detection, object segmentation or object tracking is then possible. Even if object detection is deeply involved in object tracking, we direct the primary focus of this research towards the task of multi-object tracking based on 3D LiDAR data. More specifically, the task is to get the detections produced by an object detector, track them, and output them. Section 2.1 describes the task of Multi-Object Tracking in detail. Performing these tasks in the 3D domain instead of the 2D domain delivers many advantages:

- 3D data delivers direct depth information, as already mentioned.
- It helps to get an improved understanding of the scene.
- It allows the correct computation of real-world velocities.
- Occlusions occurring during the tracking task can be handled more easily.
- One same object captured from different sensors at different positions in space can be associated easily with each other

A more straightforward tracker is already included and used in the urban environment in the currently existing project this thesis is part of. However, the basis of this tracker is the reliance on simple physical properties, which can lead to weaknesses in urban areas. Tracking

in urban environments comes with more complex challenges due to the increased complexity and the dynamic environment than e.g. tracking on highways. Therefore, the ultimate target is to find another tracking approach that is performing better than the current one, especially in urban environments. However, additionally to its application on the intersection data, in the optimal case the tracker is also able to produce good results when applied to data from highway scenarios. Ultimately, we should be able to integrate the working tracking system into the project's existing toolchain pipeline. Including a well-performing 3D tracking system will help the AUTOtech.agil project improve performance and enable it to create a digital twin in the urban environment. Additionally, object tracking can also improve existing features in the existing project, like object detection and segmentation.

1.3 Outline

This thesis is structured as follows to provide a comprehensive exploration of multi-object tracking within the realm of autonomous driving. Chapter 2 introduces the necessary background to correctly understand the topics covered in the following chapters. It explains the general task of Multi-Object Tracking and different approaches, introduces mathematical concepts used for computations necessary for tracking, and specifies the metrics used for evaluating the tracking system. In the following Chapter 3, many related works are presented structured according to the used approach and the used model architectures, like physical models, graph models, or attention-based models. It shows the diversity of possible solutions for the problem of Multi-Object Tracking. After that, in Chapter 4, the chosen approach we want to use to solve the problem statement is explained in detail. Following is a description of the features that expand upon the chosen approach. Having explained how we realize the tracking task, we execute it on data recorded in more urban and less urban areas in Chapter 5. This chapter compares the results of the existing tracker used in the project to those from the proposed tracker. In Chapter 6, we give an outlook on the current system's weaknesses. Future work can include addressing these issues or further improving already well-performing features. Finally, a conclusion is drawn in Chapter 7 to summarize the thesis results.

Chapter 2

Background

This chapter gives the reader an overview of the basic knowledge of things used or applied in the following parts of the thesis. This knowledge includes different general and concrete concepts, including explanation of the performance metrics used for evaluating the performance of the implementation.

2.1 Multi-Object Tracking

Problem Formulation

One can generally subdivide object Tracking into Single-object Tracking and Multi-Object Tracking. In both cases, the input is a video sequence in the form of consecutive snapshots of the observed environment at consistently distanced timestamps. These snapshots are primarily in the form of images or point clouds, depending on whether 2D or 3D Tracking is performed. In **Single Object Tracking**, the tracking algorithm fixes its target on the same object through the whole sequence. An additional input or an arbitrary choice of an object determines the object to track. This object is then tracked across a given sequence of frames, which means that the algorithm returns the exact position of the considered object at every timestep. Single-object Tracking is simple and efficient, but, as the name already says, it can only track one object at a time. Therefore, the more exciting problem formulation is the task of **Multi-Object Tracking**. In Multi-Object Tracking, we consider an arbitrary number of objects at each frame, and the target is to assign these objects a consistent ID across multiple frames if they already occurred in a previous frame. In Figure 2.1, we consider two timesteps and present the objects at every timestep. In this concrete case, the object's IDs correspond to the colors of the surrounding bounding boxes. The target is now, e.g., that the woman with the red jacket in the right image is assigned the same bounding box color as in the left image, and the two other men in the left image are also assigned the same color in the right image. The man with the purple bounding box in the right image has not been there before. Therefore, we assign him an arbitrary color. So, the goal is to find the correct positions of the objects already identified in previous frames. An alternate formulation of the goal is to find the optimal sequence of states for each occurring object. In the concrete case of following the Tracking by detection approach, see Section 2.2.2, we can formulate the problem mathematically as a maximum-a-posteriori (MAP) estimation problem[16]:

$$\hat{S}_{1:t} = \arg\max_{S_{1:t}} P(S_{1:t}|O_{1:t})$$

Here, $O_{1:t}$ represents all the sequential observations of all objects from frame F_1 up until frame F_t, i.e. the detected objects. S denotes all possible sequential states of all objects from

the first Frame F_1 until Frame F_t. $\hat{S}_{1:t}$ are the predicted states for the objects.
As mentioned earlier, we can also distinguish between 2D and 3D Tracking. The actual difference is mostly simply the data format of the given input. If the input data is in 2D format, i.e., images, we typically perform 2D Object Tracking. If the input data is in 3D format, i.e., in point clouds, we usually employ 3D Object Tracking. Additionally the output is different, of course. 2D Tracking returns bounding boxes describing an area of the image in the form of positions in the images, while 3D Tracking returns cuboid bounding boxes including 3D coordinates.

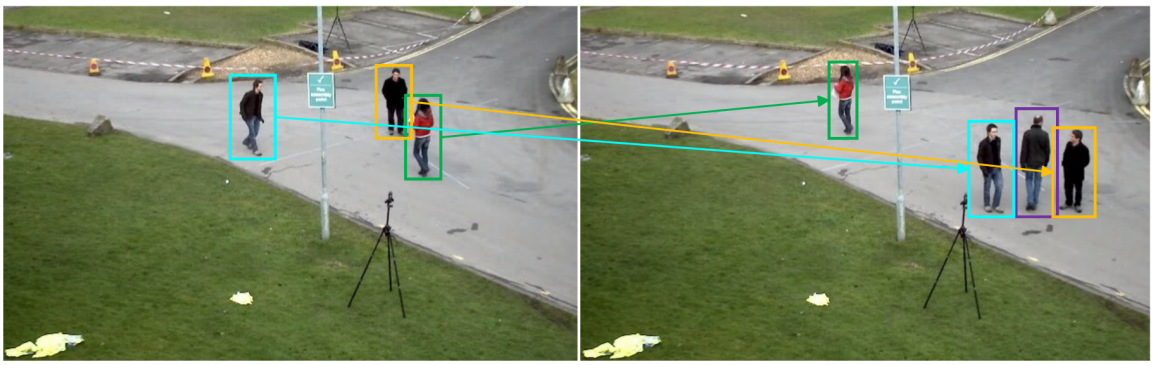


Figure 2.1: These two images from [12] illustrate the fundamental concept of Multi-Object Tracking. In the left image, three individuals are present. The right image shows four people, including the three from the left image and a new person. Multi-Object Tracking involves identifying which individuals in the right image correspond to those in the left image and determining the occurrence of new individuals, as indicated by the colored boxes.

Online and Offline Tracking

The difference between online and offline tracking lies in the information the tracker uses to perform the tracking task. Online Tracking only uses information from the current frame and past frames since, in real-world applications, it does not have information about future frames. Offline Tracking, also called Batched Tracking, uses information originating from all input frames. Processing one single frame can use previous and future frames for more accurate results. Online Tracking is used in applications or scenarios requiring a system's real-time performance or immediate reactions. For example, autonomous vehicles have to track the movement of other vehicles to be able to react to changes in the environment immediately. Also, video surveillance needs online tracking to get immediate information about objects or persons of interest. On the contrary, we use batched tracking when there are no timing constraints. For example, in performing video analysis or behavior analysis, the consecutive positions of objects need to be computed. Another use case for offline tracking can be post-processing of annotations or detections for dataset curation. However, the result is not immediately needed but consumed as a total in the analysis phase afterward.

Challenges

Multi-Object Tracking is a challenging task. It brings many problems and challenges one must consider in the solution approach for the problem. Some of the main challenges are:

- Occlusions: An occlusion occurs if a specific object is visible at frame F_t, is not visible at frame F_{t+1}, but is visible again at frame F_{t+2} or a later one, e.g., as illustrated in Figure 2.2. These occlusions can be partial or complete. For the tracking algorithm to deliver good performance results, the used algorithm has to handle these occlusions

to track the objects over extended periods. One way to realize this is, e.g., to model occlusion relationships between objects. Another possibility is to keep information on disappeared objects for a few following frames to be able to track them if they appear again in a later frame.

- Similarity of Objects: Another challenge for a tracking algorithm is the fact that multiple objects can look similar to each other in the current frame. More precisely, this problem mainly occurs when the tracking algorithm works with the appearance features of the considered objects. Then, two objects that look pretty similar could switch their IDs without the tracker recognizing the mismatch.

Figure 2.2: An occlusion scenario in crowded scenes with objects of varying sizes. Image 1 depicts a visible bicycle crossing the intersection, followed by the bus overtaking it in the second image. In the third image, the bicycle becomes visible again. Multi-Object Tracking aims to determine whether the bicycle in the third image corresponds to the one in the first image.

- Real-time constraints: Object Tracking must fulfill real-time requirements in many cases. In these cases, the tracking algorithm has to have a short inference time since multiple frames per second usually have to be processed by the tracking system. In the best case, the tracker should produce results in the frequency corresponding to the time difference between consecutive frames.
- Detector: Many approaches to Object Tracking utilize an object detector alongside the actual tracking algorithm. The tracker then uses the detections delivered by the detector. The first challenge is that the detector also needs a certain amount of time to complete its computations, making the real-time constraint even more challenging. Furthermore, the tracking algorithm works based on the detections delivered by a specific detector. In that case, the quality of the tracking results will depend significantly on how accurate the delivered detections are.

2.2 Pipeline

2.2.1 General

In most cases, when performing object tracking, object detection is also done or even required for tracking. There are different paradigms of how the two components of Object Detection and Object Tracking play together. It is possible to do Tracking before detection. In that case, the latter object detection typically utilizes the tracking results. A frequent reason for using this strategy are objects to track that are hard to observe. Another reason is to set the tracking task free from the fixed category that a detector gives its detected objects [37]. Often, tracking before detection is also used as a pre-processing step for object detection to improve the performance of the object detector.
Another possibility is to do joint Tracking and Detection. In that case, one algorithm considers object detection and object tracking simultaneously within a unified framework. Suppose

we use a learning-based approach for joint object detection and tracking. In that case, the whole pipeline, including detection and Tracking, can be optimized jointly, simplifying the training of the Multi-Object Tracker. Joint Tracking and detection aims to use complementary information and improve the overall performance and robustness of the Multi-Object Tracking systems in dynamic environments.
However, the predominant strategy at the time is to use a tracking-by-detection approach. The detections are delivered to the tracker by the detector, so the tracking task happens after the detection task. Therefore, the two tasks are mostly independent of each other. Since it is the most prominent approach at the time, the following section explains the concept in more detail.

2.2.2 Tracking by Detection

Generally, we can distinguish the Tracking-by-detection strategy into five main parts. The tracking algorithm then executes these five parts sequentially [20]:

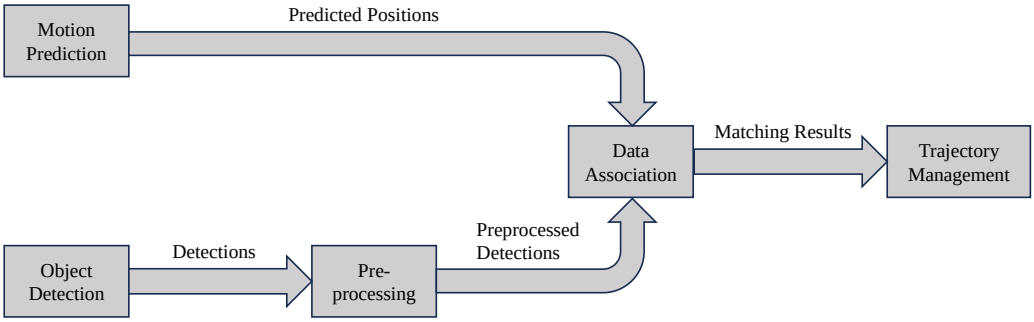


Figure 2.3: Illustration of the tracking-by-detection paradigm, depicting its individual steps.

1. Object Detection: While the primary focus of this work lies in 3D multi-object Tracking, it is essential to acknowledge the integral role of object detection within the overall pipeline. Typically, tracking-by-detection employs a well-established and effective detector. Significant to mention is that the detector operates independently of the subsequent tracking process. Once the detector completes the object detection task, it provides bounding boxes outlining the detected objects. These bounding boxes commonly include information such as position, dimensions, orientation, category, and confidence score. It is also possible that the detector delivers additional information about the bounding boxes in the form of appearance information, which can originate from image data as well as from LiDAR data.
2. Pre-Processing: The Pre-processing step is not strictly necessary but is often used to reduce the number of false positive detections delivered by the detector. Ultimately, the tracker chooses the bounding boxes used for tracking at this stage. Popular methods are, e.g., Non-Maximum-Suppression or a Score Filter. The latter checks the confidence score of a detection. If the score is below a certain fixed threshold, it discards the detection. Non-maximum-suppression aims to prevent the same object from being detected multiple times, resulting in multiple bounding boxes for one object. Ultimately, Non-Maximum-Suppression discards all bounding boxes with an Intersection over Union above a certain threshold.
3. Motion Prediction: A 3D Multi-Object Tracker usually uses a motion prediction module to predict the state by including the position of the objects tracked in the last frame for

the state update of the current frame. The state includes the bounding box information from the detector. It often uses the object's location, orientation and velocity from the last couple of frames to estimate where the object will be moving to and in which state it will be in. 3D object tracking commonly uses physical models like the constant acceleration model. Various methods, such as Extended and Unscented Kalman Filters, Particle Filtering, or Probabilistic approaches, can realize motion prediction. They provide a dynamic representation of how the objects move in consecutive frames.

4. Data association: After the predicted states of the tracked objects from the past frames are available, the goal is to associate these tracked objects with the object detections from the current frame. The objects get the same ID as the associated tracked objects from the past frames. Principally, two parts have to be considered [16]. First, the tracking algorithm computes the similarity of the detections and the tracked objects from past frames rationally. Secondly, based on the similarity of objects, the identification of the new detections has to be realized, i.e., the tracked objects must be matched with the detections optimally. The following Subsection 2.2.3 gives a more detailed classification of data association strategies.
5. Trajectory Management: The last part is to decide what trajectories are declared dead, which are newly created, and which ones will be output [20]. In the process of Multi-Object Tracking, declaring a trajectory as dead means that the object represented by the track is no longer present in the perception field of the used sensor and, therefore, will no longer be tracked and removed from the list of tracked objects. Creating a new track happens if a new detection is not very similar to any tracked object and, therefore, cannot be associated with an existing track. So, it will be assigned a new, to this point, unused ID representing a newly initialized trajectory. Concerning the Multi-Object Tracking system output, a decision about which trajectories are returned to the caller or output to a result file is required. For example, it could only return trajectories matched in the current frame or also ones matched in the past frame to the caller for a specific frame.

A problem that can occur while using this strategy is the dependence of the tracking performance on the detection performance, as previously mentioned. If the detector produces many incorrect or imprecise detections or misses many detections, the tracker must try to compensate for these errors somehow.

2.2.3 Learning and Learning-Free

An essential distinction of Multi-Object Trackers to make is if the tracker uses an approach depending on a machine learning model or if it is learning-free, meaning that it can be realized by only doing simple mathematical computations that do not require a lot of computational resources like, e.g., neural networks do. This distinction relates mainly to the process of motion prediction as well as the process of data association.

- Motion Prediction: To predict the motion of an object in future frames, one can either choose to use a physical model or a learning-based approach. An example of a physical model is the constant acceleration model. This simple linear motion model assumes that the object is moving with a velocity influenced by a constant acceleration in the same direction in one time step. It does not take into account the orientation of an object. On the other hand, another way to predict the state of an object in the future frame is the usage of, e.g., Recurrent Neural Networks. The input to this network would then be the object's location, orientation and other kinematic properties from the previous frames. The advantage of the physical models is that they are more interpretable since their results are explicitly defined based on physical principles. Often, they are also

faster to compute. On the other hand, learned models can learn complex patterns and adapt to many circumstances that the physical models could not represent.

- Data Association: As mentioned in Section 2.2.2, the data association in multi-object tracking is usually composed of computing the affinity of detections and tracks and afterward getting a good matching between the detections and the tracks.
  - Affinity Computation: In this part, the tracking algorithm computes the affinity or similarity between the past frames' tracks and the current frames' detections. 2D Tracking often uses a convolutional deep-learning model to get a representation of the objects. It can then compare these representations. Siamese networks, e.g. [2], are neural networks that can compute the similarity between two objects by using the images of the detections and the tracks. Their goal is to minimize the distance between pairs of detections and tracks that are very similar and maximize the distance between dissimilar pairs. Since the input data is often in LiDAR format in 3D Tracking, training Siamese networks is not a common choice for affinity computation. In that case, an often-used alternative are graph neural networks [27], which model the relations between detections and tracks as a graph. By that, they can model complex dependencies and relationships. Apart from the learned affinity computation, simple distance measurements between the bounding boxes can also check which detections and tracks are likely to correspond to the same object. For example, Euclidean distance or the intersection over union metric are common choices to check how far apart and how similar two bounding boxes are. These metrics usually can be applied in the 2D as well as in the 3D domain.
  - Matching: The matching between the old tracks and the latest detected objects usually happens without machine learning. It uses the computed affinities in the previous step to perform bipartite graph matching [16], leveraging known matching algorithms like the greedy or Hungarian algorithm. Most of the available multi-object tracking systems use the latter one. It guarantees an optimal solution that minimizes the total cost of the assignment.

2.3 Kalman Filter

2.3.1 General

Generally, a Kalman Filter is used to solve the filtering problem by estimating the state of a system. External inputs and measurements of outputs performed by devices or sensors principally determine the system's behavior. Based on this information, the Kalman Filter tries to estimate the current state of a system [23] by also considering the noise in the measurements that originate from a noisy environment. In the context of Multi-Object Tracking, the state of every tracked object is managed by a Kalman Filter. This Kalman Filter internally keeps the object's state updated, i.e., usually the position, dimension and rotation of the bounding box corresponding to the object. Using the state of the object and applying the previously set system model, e.g., the constant acceleration model, it tries to predict the internal state of the object, i.e., the position and rotation, for the following frame. These considerations constitute the prediction step of the Kalman Filter. After the tracked object was assigned a detection in the current frame, the Kalman Filter incorporates the information from the assigned detection as the noisy measurement to adjust its prediction. While doing that, it considers the uncertainties in the measurements and the system model, labeled as the update step. Since the object's state is now up-to-date for the current frame, the filter is then ready to process the next frame. If there is no assigned detection in a frame to the tracked object, the tracker

skips the update step of the filter, assuming the predicted state is the correct state. Motion prediction in 3D object tracking often uses the principle of the Kalman Filter.

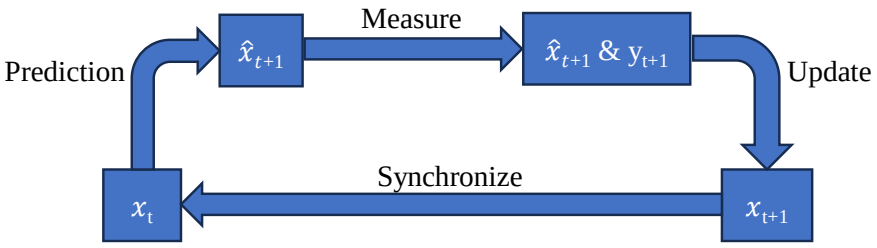


Figure 2.4: Illustration of the Kalman Filter's operating principle. The Filter predicts the system's next state ($\hat{x}_{t+1}$) based on its current state, then corrects this prediction using real-world measurements (y_t) to synchronize with the actual state (x_{t+1}).

2.3.2 Mathematical Computation

The general Kalman Filter assumes that the state of a system changes in a way such that a linear equation looking like this [23] can describe it:

$$Prediction: \hat{x}_{k+1} = A_k x_k + B_k u_k + G w_k$$

$$Update: x_{k+1} = \hat{x}_{k+1} - K_t * (y_t - H * \hat{x}_{k+1})$$

$$y_k = C_k x_k + v_k$$

where x_k is the state of the system at step k, y_k is the measurement given the state x_k and w_k and v_k is Gaussian noise with zero mean. A is the state transition matrix, B is the control input matrix and u is the control input. K_t is a timestep dependent computed factor and H is the measurement matrix.
One can see that the Equation 2.3.2 is a linear function. The Kalman Filter operates recursively, updating its estimate as new measurements become available at each time step. The result is a Gaussian density function that represents the current estimated state of the system.

2.3.3 Extended Kalman Filter

In the case of nonlinear system dynamics, including states and observations, we can no longer use the standard Kalman Filter. The problem is that the standard Kalman Filter relies on linear system dynamics and measurements. It assumes Gaussian distributions for the state and measurement noise. If the system dynamics are nonlinear, the resulting density function describing the estimated state is not Gaussian, and therefore, the preconditions for using the Kalman filter are violated [23]. Instead, extending the Kalman Filter to also represent nonlinear system models is possible. The Extended Kalman filter includes two steps:

1. Linearization: The nonlinear functions in the system dynamics and measurement models are locally approximated by linear functions using the Jacobian matrices. This approximation provides a linearized representation of the system in its current state.

2. Kalman Filter: The Extended Kalman Filter then uses the linearized models to execute the standard Kalman Filter. This involves predicting the state and covariance based on the linearized system dynamics, the update based on the linearized measurement model, and the computation of the Kalman gain, as mentioned before.

Since the Extended Kalman Filter uses an approximation of the system dynamics in the form of a linearized model, it introduces errors in the state estimation process, mainly if the system exhibits significant nonlinear behavior.

2.4 Generalized Intersection over Union

The Generalized Intersection over Union (GIoU) or the basic Intersection over Union (IoU) can compute the overlap between two spaces of elements. In the object detection and tracking context, these two spaces of elements are the bounding boxes of two detections. In 2D, the bounding boxes are rectangles at specified positions; in 3D, the bounding boxes are cuboids. So, all forms of Intersection over Union compute the overlap of two areas or two cuboids. The IoU is the most used metric, which can be computed by:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

where A and B would be the bounding boxes. The weakness of the standard Intersection over Union is that two elements that do not overlap with an observed third element are defined to have an equal distance to the target, also if one is near to it and the other not. Figure 2.5 illustrates this problem. To overcome the weakness [22] proposed the Generalized Intersection over Union (GIoU) to improve the standard Intersection over Union. The GIoU is defined as follows:

$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$$

Where C is the smallest convex shape enclosing A and B [22]. This principle was generalized to 3D by [20]. The value of the GIoU can be between -1 and 1, where -1 is the worst case where the two boxes are entirely separated, and 1 is a perfect overlap of the bounding boxes. This metric can avoid the previously mentioned problem, pictured in Figure 2.5.

2.5 Metrics

Meaningful metrics must be defined and used to evaluate the tracking system's performance. To do that and to understand how these metrics work, we present relevant error sources that occur in the tracking task. We also explain some more advanced metrics after some basic and straightforward metrics. Important to mention is that to evaluate a tracking system, we always require a consecutive sequence of frames.

2.5.1 Evaluation Process

To evaluate the performance of an algorithm, one has to possess the algorithm's results and the ground truth values. We assume the ground truth values to be the correct values representing real-world happenings. The tracking algorithm delivers the results of its algorithm as a set of hypotheses, which are the positions it thinks the objects with specified IDs are at

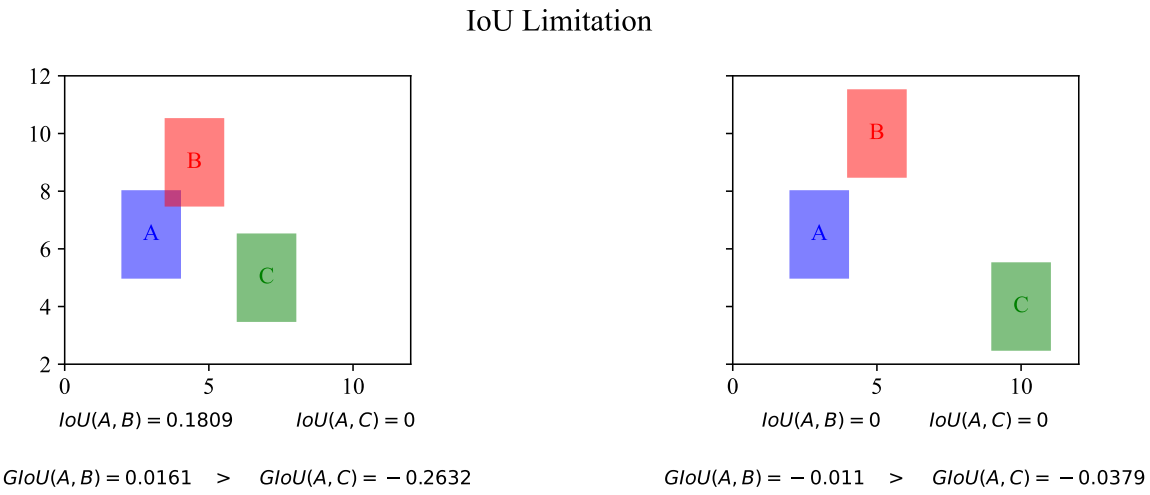


Figure 2.5: Illustration of the advantage of Generalized Intersection over Union (GIoU) over Intersection over Union (IoU). The goal is to determine the proximity of objects. In both images, objects A and B are closer than A and C." The left image shows an intersection between A and B, resulting in larger IoU and GIoU for A and B compared to A and C." In the right image with no intersection, IoU is 0 for both pairs, but GIoU considers the smallest area containing both objects, resulting in a non-zero GIoU for A and B and claiming them to be closer than A and C.

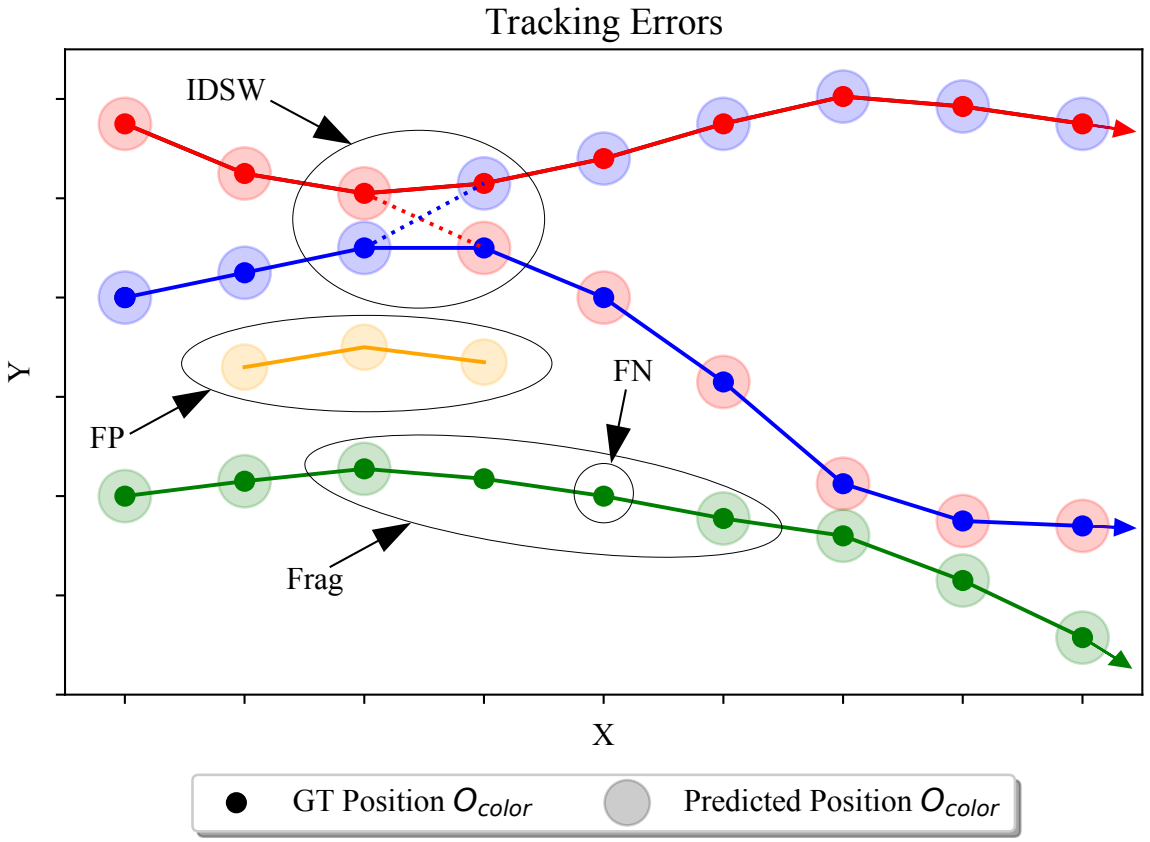


Figure 2.6: The paths of three objects are depicted, with ground truth positions at consecutive timesteps shown by dots and connected by colored lines. The transparent surroundings of the dots represent the tracker's prediction of the object ID. An ID-switch (IDSW) occurs when the tracker changes the IDs of the red and blue objects. False Positives (FP) arise when the tracker predicts a yellow object that does not exist. The tracker misses two predictions for the green object but later associates it with the correct color, resulting in fragmentation. One of these missed predictions is considered a False Negative (FN).

the current frame. The evaluation itself happens frame-by-frame. At each frame, we consider the results from the tracker in the current frame and the results from the ground truth in the corresponding frame. Given these two quantities of positions and bounding boxes, the evaluation functionality has to match the hypotheses with the elements of the ground truth. These correspondences between the predicted objects from the tracker and the ground truth objects are the basis for the following sections.

2.5.2 Common Errors

First of all, this section explains the typical mentioned and occurring errors to prevent misunderstandings; Figure2.6 provides a more intuitive explanation:

- False Positives (FP): The tracker predicts a hypothesis for the position of an object that could not be associated with an object from the ground truth.
- False Negatives (FN): The ground truth contains an object that is not matched to a predicted object from the tracker, meaning the tracker has not tracked this object in the current frame. Such a case is also called a miss.
- ID Switches (IDSW): An ID switch occurs if the same object that the tracker predicts in two or more consecutive frames switches ID from one frame to another, although it should have kept its ID according to the ground truth. It effectively counts the number of times an object got a new ID, although it should have maintained the old one. The source of an ID switch are often objects that look pretty similar or very crowded situations.
- Fragmentations (FRAG): An object's position with a consistent ID is tracked, but the track is lost, e.g., due to an occlusion. After some frames, the object is tracked again, but with a different ID. Such a scenario constitutes the definition of a fragmentation.

All these errors can also represent a metric. Keeping them as low as possible indicates a good tracking performance, achieving the ultimate goal.

2.5.3 Simple Metrics

Some metrics are rather simple and mainly used to get a rough overview of how good the results of the tracking system are. This work uses the following ones:

- Total Track Length: The sum of all single track lengths. We define one track as the positions at which one object with a consistent ID was tracked.
- Total number of tracks: The number of IDs given to objects across a whole sequence. It is the number of different objects in the real world. We can then compare the total number of tracks to the total in the ground truth. The nearer these two numbers are together, the better the performance of the tracker.
- Average Track Length: Total track length divided by the total number of tracks. It shows how long an object's track is on average. A higher value often corresponds to a better-performing tracking system.

2.5.4 Advanced Metrics for Performance measurement

As already said, the metrics above only give a rough impression of how well the tracking system performs. We use more advanced metrics to evaluate a tracker and compare the results to those from different trackers and benchmarks. These metrics use the error cases mentioned in Section 2.5.2. A set of widely used metrics that provide a comprehensive

assessment of the tracking performance are the CLEAR MOT metrics [1]. These metrics include the IDSW, FRAG, MOTA and MOTP. AMOTA and AMOTP are further advanced metrics originating from the CLEAR MOT metrics. Apart from that, the HOTA metric is also an important performance indicator that gives a more complete estimation of the performance than MOTA and MOTP. AMOTA and AMOTP are integral metrics proposed by [30] to deal with the problem that the other evaluation metrics do not consider the confidence score of a track.

- Multiple Object Tracking Accuracy (MOTA): MOTA focuses on evaluating the overall accuracy performance of a tracking system. It returns a comprehensive evaluation that reflects the ability to detect and track objects correctly and maintain the correct identity of objects over time. The definition of the MOTA computation is as follows:

$$MOTA = 1 - \frac{\sum_t |FN_t + |FP_t| + |IDSW_t|}{\sum_t |TP_t|}$$

  The MOTA metric is probably the most popular metric to evaluate the tracking performance, also if it may not be sufficient if used alone.
- Multiple Object Track Precision (MOTP): MOTP focuses more on the localization performance of the tracker. It is the average distance between the predicted and the matched ground-truth objects. The formula for its computation is:

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t |TP_t|}$$

  where d_t^i is the distance between a predicted object and its matched ground truth.
- Higher Order Tracking Accuracy (HOTA): This metric considers a multi-object tracking system's detection, identity and robustness performance. The computation comprises the computations of the localization, detection and association accuracy [15]. It generally is defined as follows:

$$HOTA = \int_0^1 HOTA_\alpha d\alpha \approx \frac{1}{19} \sum_{\alpha \in 0.05, 0.1, \ldots, 0.9, 0.95} HOTA_\alpha$$

$$HOTA_\alpha = \sqrt{\frac{\sum_{c \in \{TP_\alpha\}} A_\alpha(c)}{|TP|_\alpha + |FN|_\alpha + |FP|_\alpha}}$$

  $A(c)$ is a metric assessing the actual association performance in the predictions. More precisely, it measures the alignment between the ground truth trajectories and the predicted trajectories [15].
- Average Multiple Object Tracking Accuracy (AMOTA), Average Multiple Object Tracking Precision (AMOTP): AMOTA and AMOTP are MOTA and MOTP values integrated integrated over the recall value [30]. The integration is usually approximated by summing over a discrete set of recall values. Recall values in that context mean the tracking system's confidence score for output tracks. With this information, we compute the AMOTA like this [30]:

$$AMOTA = \frac{1}{L} \sum_{r \in \{\frac{1}{L}, \frac{2}{L}, \ldots, 1\}} 1 - \frac{|FP_r| + |FN_r| + |IDSW_r|}{|TP|}$$

  where L is the number of thresholds we integrate about. The formula under the sum is the MOTA value at the specified recall level. The computation of AMOTP works analogously with MOTP at the specified recall value instead of MOTA.

- Mostly Tracked(MT), Partially Tracked(PT), Mostly Lost(ML): these metrics keep the number of track IDs that were tracked for at least 80% (MT) or at most 20% (ML). We count the tracks in between of that range as partially tracked (PT).

Chapter 3

Related Work

As mentioned in Section 2, 3D Multi-Object Tracking approaches can be divided into the three categories of tracking-before-detection, joint-tracking and Detection, and tracking-by-detection (TBD). The chapter introduces some concrete works following the named theoretic principles to get an impression of how these topics are approached practically by researchers in the real world.

3.1 Tracking before Detection

Tracking before Detection is the approach that is used least often in the world of Multi-Object Tracking. Most of the works are relatively old; therefore, the following examples only try to glance at how to realize the tracking-before-detection approach. Works using the tracking-before-detection mostly use dynamic programming algorithms, multi-Bernoulli random set filter or particle-filter-based methods [34]. E.g., [34] uses a dynamic programming approach by performing Bayesian filtering after discretizing the state space. It tries to track an unknown number of objects by processing multiple consecutive frames simultaneously. However, this approach requires solving a high-dimensional optimization problem and, therefore, is computationally intensive, although they [34] proposed a method to reduce the computation complexity. [17] proposes another work pursuing the tracking-before-detection strategy. This work realized the track-before-detection approach to also track objects from unknown categories instead of objects originating only from known categories. They extract regions of interest using a stereo setup and segment them into candidate objects. These regions of interest are then independently tracked over consecutive frames using the iterative-closest-point method. After tracking a region of interest, the tracker passes its results on to an object detector to determine the category of the tracked object. The authors claim that their tracking algorithm performs well, also in very crowded scenes. Such a strategy has the positive side effect that a tracked object only has to be detected once, unlike in a tracking-by-detection approach in every frame, thereby saving computational resources.

3.2 Joint Tracking and Detection

Nowadays, many works also realize Multi-Object Tracking through the joint Tracking and Detection paradigm. It aims to improve the overall performance and accuracy of the tracking system. Different strategies exist, including using Transformer architectures, Kalman Filters, or Graphs. Many of the existing Joint Tracking and Detection approaches require image data as input, including the ones presented in the following:

PF-Track

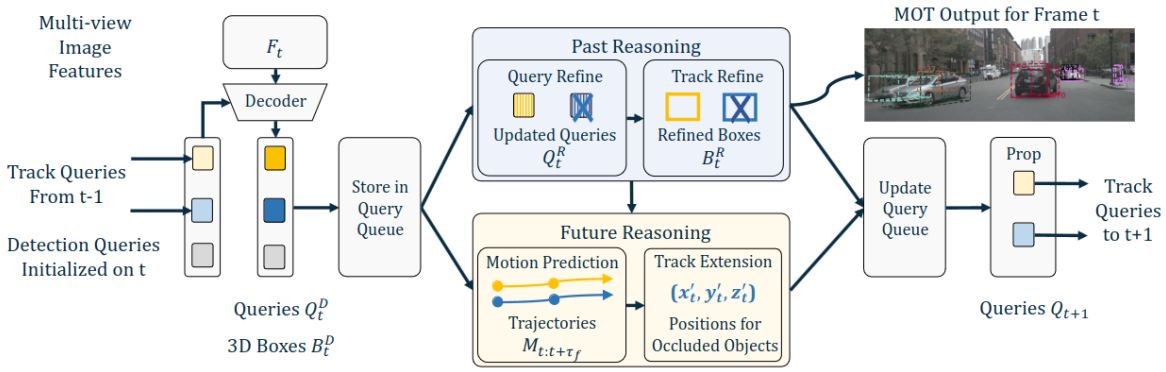


Figure 3.1: From [19], PF-Track represents objects, existing tracks, and new detections as queries. The approach employs Past and Future Reasoning to enhance spatio-temporal coherence. Past reasoning utilizes historical information to refine object queries, while Future Reasoning aims to improve motion prediction for future object states.

[19] propose an end-to-end multi-camera 3D multi-object tracking framework. Contrary to the target of this thesis, it uses image data only. Their approach, called PF-Track, uses a tracking-by-attention paradigm, i.e. utilizing the attention mechanism. An object query represents every object, the tracked objects from the past frame, and new detections in the current frame. These queries then depict the corresponding object by a feature vector and the 3D position of the object. The following parts subdivide the overall procedure:

1. To process a frame, the queries for all object tracks from previous frames are required, which are then extended by a fixed number of new detection queries to be able to track newly detected objects.
2. In the next step, the decoder of the framework uses the object queries to decode the image features of the current frame. The decoder outputs the detected bounding boxes for all tracks in the current frame and the updated query features kept internally.
3. To refine the detected bounding boxes and the query features in the current frame, PF-Track applies Past Reasoning. A combination of a cross-frame-attention module and a cross-object-attention module refines the queries. The cross-attention is applied across the time and the instances, setting the queries in relation to its tracks from previous frames and other objects in the current frame. A simple MLP refines the positions of the tracks' bounding boxes output in the current frame. The tracker writes these outputs to the result file.
4. Future reasoning is used to improve the propagation of track queries across frames. The authors realize this by applying a cross-frame attention module to get the motion embeddings of the tracks. An MLP then produces the decoded embeddings that represent the future trajectories of objects. Predictions from the trajectories resulting from future reasoning replace low-confidence detections to handle occlusions.

The PF-Track reaches an AMOTA value of 0.43 in the nuScenes tracking benchmark, ranking 78th in the nuScenes tracking challenge, see Table 3.1. It notably minimizes the number of identity switches and improving overall spatio-temporal coherence. They show that the approach can propagate the positions of occluded objects over several frames and re-associate them with the correct ID.

3D DetecTrack

[10] tries to solve the task of Multi-Object Tracking by realizing a joint object detection and multi-object tracking framework based on the fusion of camera and LiDAR data. The detector generates spatio-temporal features, which the tracker then uses to associate the objects from the previous frames with those in the current frame. Principally, the overall pipeline separates the detection stage from the tracking stage.

- Detection stage: They use an existing 3D detector which computes 2D and BEV features from LiDAR data, delivers spatiotemporal features. The 2D and BEV features are then fused to a single feature map using a feature aggregation network. After that, a Region Proposal Network detects the objects and their confidence scores. Another neural network refines the detection and the score. In both of these processes, by incorporating the track information from the previous frame into the computation for the current frame they want to improve the performance.
- Tracking stage: The spatiotemporal features delivered by the detector are used to match the detections with the tracks from previous frames. After computing RoI-aligned features for the detections in the current frame and the tracks from the previous frame, the features of each object are fed into the Spatial-temporal Gated GNN (SG-GNN). A node represents every object, and the graph itself is fully connected. The SG-GNN, in the end, outputs an affinity matrix, which the Hungarian algorithm then uses to generate the matching between detections and tracks.

This approach has some characteristics of a tracking-by-detection framework but with the critical difference that the detector incorporates the tracks from the previous frames to reconfigure the detection output. So, the detector and tracker collaborate to optimize the object detection and object tracking jointly. DetecTrack is only evaluated on the nuScenes validation set reaching an AMOTA value of 0.11. Therefore it is also not ranked in the nuScenes tracking benchmark.

3.3 Tracking by Detection

Tracking by Detection is by far the most popular and most used approach for tackling the problem of Multi-Object Tracking. Existing research and projects implement different approaches in the tracking component of the pipeline. The options for approaches to realize motion prediction or data association include Kalman Filters, Transformer architecture, Graph-based approaches, Deep-Learning approaches, and more. The following sections present a few existing works structured after their computation technique.

3.3.1 Based on Kalman Filters

SimpleTrack

SimpleTrack [20] breaks the tracking functionality down into the parts described in Section 2.2.2 and tries to improve the performance of the Tracking by investigating each of these functionalities individually for possible enhancements.

1. Pre-processing: SimpleTrack proposes to use stricter Non-Maximum-Suppression to the detections received from the detector to filter out false positives and prevent the tracker from selecting these false positives to form tracks.
2. Motion Prediction: A Kalman filter is applied to predict the motion of existing tracks since it performs better on high-frequency data that [20] uses and does not need velocity measurements performed by the detector to make the prediction.

3. Data association: For the association between the detections in a current frame and the tracks from the previous frame, they apply the generalized intersection over union metric as explained in Section 2.4. For the matching part, they argue that the Hungarian algorithm and the greedy algorithm perform with similar strength when working with an IoU-based association metric.
4. Trajectory Management: SimpleTrack uses a two-stage association strategy. In the first stage, they match only detections with a confidence score higher than a threshold. The matched detections of this stage are then also the ones that are output. If a detection is still successfully associated in stage two, it is not declared dead, but it is also not used for the update step of the Kalman filter that is realizing the motion prediction.

Integrating the mentioned improvements into the Multi-Object tracking framework mitigates some weaknesses occurring in other works. For example, the principle of two-stage data association significantly reduces the number of ID switches by reducing the number of early terminated trajectories. SimpleTrack reaches an AMOTA value of 0.67 on the nuScenes test set, ranking 32nd in the nuScenes tracking challenge, see Table 3.1.

ByteTrackV2

The approach proposed by [38] aims to improve the overall tracking performance by making use of all detection boxes independent of their confidence score, resulting in a detection-driven hierarchical data association strategy [38]. This approach is not limited to 2D or 3D Tracking but is applicable for both tasks. Once more, they divide the framework into the components explained in Section 2.2.2. The approach reaches an AMOTA value of 0.56 on the nuScenes test set, ranking 61st in the nuScenes tracking challenge, see Table 3.1.

1. Detection: ByteTrackV2 uses CenterPoint and TansFusion-L as its 3D detector processing LiDAR input. As a 3D camera-based detector, they mention the PETRv2 detector, while for 2D Detection, they use the YOLOX detector.
2. Motion Module: As the motion model, ByteTrackV2 uses a simple Kalman Filter, keeping an internal state vector whose dimensionality depends on the kind of Tracking performed, 2D or 3D. A complementary motion prediction is applied in 3D motion prediction to handle abrupt motions and object occlusions. Forward prediction uses the Kalman Filter for long-term prediction to recover from missing detections. In contrast, backward prediction, responsible for the short-term association of current tracks, uses the detected velocity.
3. Data association: The unified 2D and 3D data association strategy first associates the detection boxes with a confidence score above a certain threshold with the existing tracks. After that, the still unmatched tracks are associated with the detection boxes not surpassing the mentioned threshold.
4. Trajectory Management: tracks still unmatched after the second association stage are removed from the active tracks but kept for a fixed number of frames. If they are unmatched for that number of frames, they get removed finally. The boxes output at each frame are the matched existing tracks and those created by unmatched detections in the current frame.

This approach approaches the problem of missing objects and fragmented trajectories by using a generic data association and a complementary motion prediction. Furthermore, it can be used with various detectors since it showed no clear preference for a particular detector in the experiments done by [38].

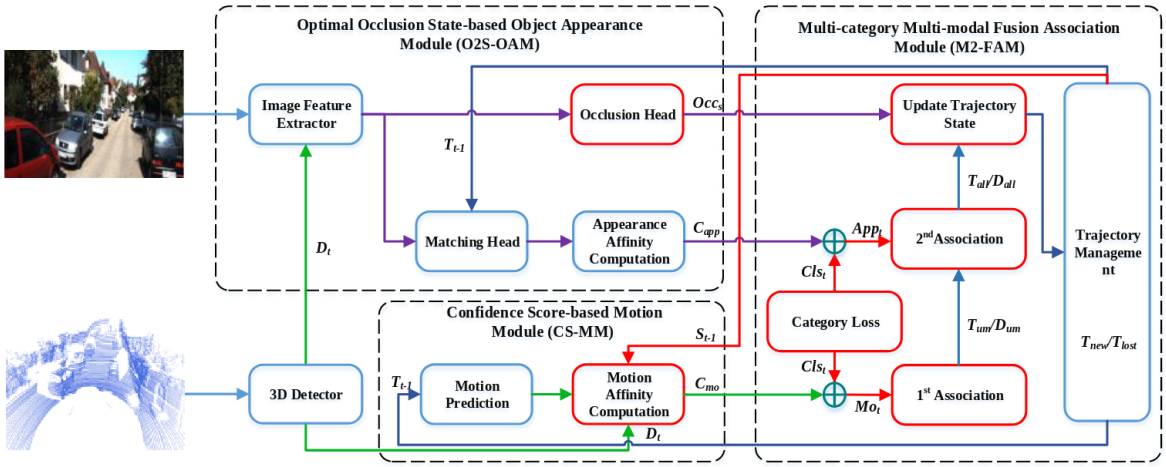


Figure 3.2: From [29]. The illustration showcases the three main modules of CAMO-MOT. The first module processes image features to estimate the occlusion state of objects, producing an affinity matrix based on object appearances. The second module creates object motion features, predicts the motion, and generates an affinity matrix based on these features. The third module combines appearance and motion affinities to establish an optimal matching between current detections and existing tracks.

CAMO-MOT

CAMOT-MOT, published by [29], is a Multi-Object Framework that uses both an appearance and a motion model, leveraging the advantages of images and point cloud data. The overall framework consists of three main modules:

1. Optimal Occlusion State-based Object Appearance Module: This includes the image feature extraction, which produces a fixed number of feature maps. After that, an occlusion head takes the feature maps and the bounding boxes of the objects detected by the LiDAR detector and tries to estimate the occlusion state of the objects. The subsequent matching head acts as an end-to-end module outputting the final cost matrix for the association between detections and existing tracks. It uses the differences between the object features in both frames and a CNN to output the final appearance-based association matrix.
2. Confidence Score-based Motion Module: CAMO-MOT uses a Kalman Filter to predict objects' trajectories. The output of this module is also an association cost matrix between the predicted trajectory states and the detections resulting from the 3D detector. The key idea is to extend the Kalman Filter to output a confidence score of their prediction together with the computed trajectory state. So, a lower confidence score in the prediction will also decrease the probability that this prediction will be matched with an incoming detection, resulting in a lower value at the corresponding entry in the motion-based association matrix.
3. Multi-category Multi-modal Fusion Association Module: This module uses cost matrices originating from the appearance and motion models to produce the final tracking results for the current frame. First of all, a category loss is added to both of the incoming cost matrices to prevent detections and tracks from different categories from being associated. After that, it uses the motion cost matrix to overcome the limitations of the 2D detector in the form of possible occlusions. Subsequently, the appearance cost matrix is used to correct tracking errors introduced, e.g., by unexpected motion of objects.

The essential new strategy introduced by this work, according to [29], is the use of the occlusion head and the introduction of confidence scores of trajectory predictions, as mentioned

above. The approach reaches a very high AMOTA value of 0.75 on the nuScenes test set, ranking second in the nuScenes tracking challenge, see Table 3.1. Due to the multi-modality of the explained approach, it is unsuitable for realizing this work's task. However, the introduced features are still exciting possibilities for improving performance.

3.3.2 Based on Attention Mechanism

InterTrack

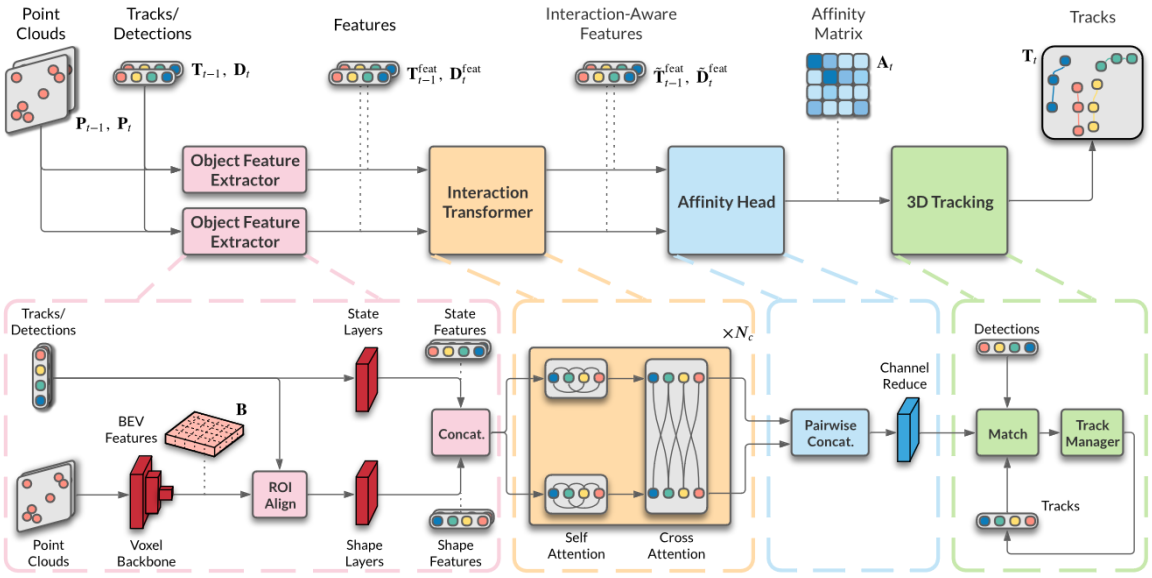


Figure 3.3: The figure provides a detailed view of the tracking process from [31]. Following the computation of features for all tracks and detections at the current frame, InterTrack employs the Interaction Transformer. This transformer applies self-attention for feature attenuation within a single frame and cross-attention to exchange information among multiple frames. To compute the affinity matrix crucial for the final matching, as described in Section 2.2.2, InterTrack utilizes the resulting features.

Intertrack [31] introduces the concept of an Interaction Transformer. [28] has proposed the general transformer architecture. The architecture of [31] computes characteristic representations of the detected and tracked objects. The focus here is on learning the affinity between existing trajectories and incoming detections to perform the data association effectively. The approach reaches an AMOTA value of 0.69 on the nuScenes test set, ranking 15th in the nuScenes tracking challenge, see Table 3.1. The following steps summarize the implemented tracking strategy:

1. Feature Extraction: The feature extractor takes the feature vectors of all tracks and detections with the frame-wise point clouds and computes new features for detections and tracks.
2. Interaction Transformer: The computed features are then fed into the Interaction Transformer to produce interaction-aware track and detection features [31]. It is composed of a self-attention block and a cross-attention block. The self-attention block computes the relation of object features within one frame, while the cross-attention block performs object feature attenuation across multiple frames. The result of the transformer are features for the objects and tracks that include intra-frame and inter-frame relations in their computation.

3. Affinity Head: The first step of calculating the output affinity matrix is concatenating each detection and track's state and shape features. A feed-forward network then consumes these features to produce probabilities representing the similarities between tracks and detections.

After that process, the optimal affinity matrix is computed and used to continue the tracking task. Now, 3D detections are projected onto the images to match them with the 2D detections in a greedy manner. InterTrack uses a two-stage data association. Based on the affinity matrix in the first stage, they compare all detections with all existing trajectories from the previous frame. The trajectories are computed by propagating them from the last frame using the motion prediction strategy proposed by [35]. After obtaining the assignment, matching pairs with an Euclidean distance over a threshold are removed. In the second stage, unmatched 2D detections and unmatched trajectories projected on the image are the inputs for the matching using IoU. After that, tracks are rejected based on 3D IoU with the other tracks.

TrajectoryFormer

[5] also uses the attention mechanism as its core part. It is a multi-object tracking framework that works only on LiDAR data. The work focuses on compensating missed detections delivered by the detector. It tries to recover these by generating multiple trajectory hypotheses with hybrid candidate boxes. The overall tracking is composed of the following steps:

1. Generation of multiple trajectory hypotheses: For each tracked object, the TrajectoryFormer generates multiple possible trajectories that try to predict the object's movement. Before doing that, an MLP that uses the history trajectory of a track is used to predict the states of the track for a fixed number of frames. TrajectoryFormer associates each existing track from the previous frames with its predicted boxes and the detections boxes from the current frame to generate Multiple Trajectory hypotheses.
2. Long Short-Term Hypothesis Feature Encoding: The multiple trajectory hypotheses are transformed into the feature space by fusing a long-term motion encoding with a short-term appearance encoding. The long-term motion encoding takes a sequence of the generated trajectory hypotheses for a track and computes its motion features utilizing a PointNet-like Neural Network. The short-term encoding encodes the appearance of each box of each trajectory hypothesis, incorporating points from the point cloud corresponding to the given box. This is done usig MPPNet. Self-attention achieves information interaction between all points of the box, and cross-attention generates the final appearance features. Another MLP takes the concatenation of motion embedding and appearance embedding as input to compute the long-short term embedding for each trajectory hypothesis of a track.
3. Global-local Feature Interaction of Multiple Trajectory Hypothesis: The computed feature encoding explained in the previous bullet point can capture each hypothesis's appearance and motion information but does not see the interaction between multiple hypotheses. A transformer with self-attention achieves this information interaction, which uses keys, values, and queries generated using the embeddings from the previous step.

After the last step, the confidence score computed by an MLP is extending the overall confidence score of each hypothesis feature. TrajectoryFormer chooses which trajectory hypothesis is best for each tracked object based on the hypothesis' confidence. The work by [5] is very interesting, considering that it only takes LiDAR data as input and still uses appearance information for performing the tracking task. TrajectoryFormer is only evaluated Waymo Open dataset [24] testing split where it achieves an MOTA value of 0.65 for tracking vehicles, a MOTA of 0.66 for tracking pedestrians and a MOTA of 0.65 for tracking Cyclists.

3.3.3 Based on Graphs

3DMOTFormer

The approach proposed by [8] called 3DMOTFormer is a 3D Multi-Object Tracking framework that combines Graphs and the transformer architecture to complete the tracking task. It models the tracking problem at a specific frame as a graph with the current detections and the existing tracks at nodes and the relations between them as edges. After the transformation operation, it updates the existing tracks by doing edge classification.
The **Graph Representation**, as already mentioned, is designed such that the nodes include all detections and all existing tracks. In the end, the work uses three graphs:

1. Detection graph: The detection graph models the interaction among all detections in the current frame; the state vector of the detection determines the initial node embedding.
2. Track graph: The track graph models the interaction among all existing tracks from the previous frames.
3. Association graph: The association graph models the relation between the current detections and the existing tracks. Here, an edge can only exist if a detection and a track are objects from the same category.

After building the graphs, the initial track features from the track graph, the initial edge features from the association graph, and the initial detection features from the detection graph are fed into the centerpiece of the framework, the Edge-augmented **Graph Transformer**. [36] explains the principle of Graph Transformers :

1. Encoder: The encoder models the interaction between the existing tracks using self-attention.
2. Decoder: The decoder uses graph self-attention to produce intermediate detection features. These intermediate detection features are used in the edge-augmented graph cross-attention as queries, while the keys and values originate from the encoder results. The computation of the cross-attention also includes the edges from the association graph.

In the end, the 3DMOTFormer tries to estimate the **affinity** between detections and existing tracks, represented as the association graph's final edge features. An MLP takes these features and performs edge classification to output the probability that the detection and the track refer to the same object. The tracks and detections are subsequently associated greedily according to the computed probability. Using the resulting track features, matched, inactive, and new tracks can be determined. The velocity computed by another MLP using the detection features can be used to predict the position of the center points in future frames.
This work is also very interesting for the task in this thesis because it only uses 3D data, as well, and does not rely on image data. Furthermore, the authors of the work claim that this tracking approach generalizes well across various 3D detectors; hardware accelerators like GPUs are necessary to use this approach for real-time applications. This approach reaches an AMOTA value of 0.68 on the nuScenes test set, ranking 29th in the nuScenes tracking challenge, see Table 3.1.

PolarMOT

[9] proposes another strategy to tackle the task of 3D Multi-Object Tracking based on graphs or, more exactly, graph neural networks. They perform tracking using only the geometric relationships between objects. The graph consists of the detections, current and from previous frames, as the nodes and edges as the relations between different objects. The edges are either inter-frame edges modeling the temporal relations or intra-frame edges modeling the

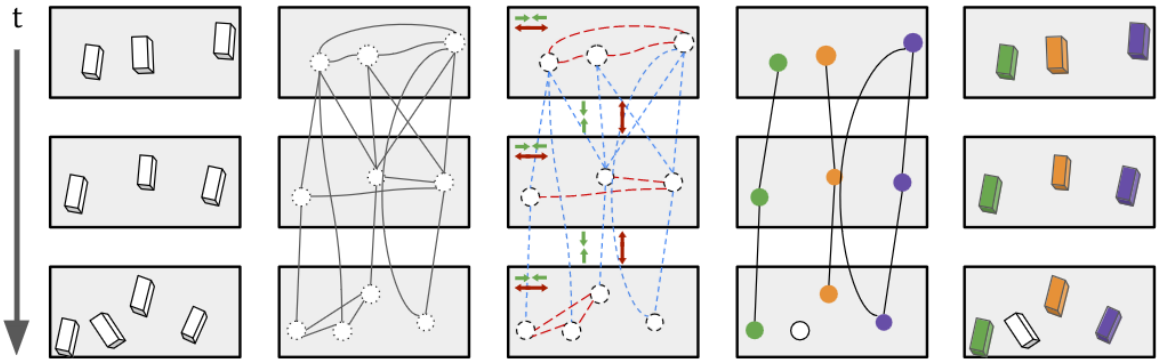


Figure 3.4: The figure elucidates the strategy employed by PolarMOT from [9]. The process begins with creating a graph, where detections from multiple consecutive frames serve as nodes, and the edges represent the number of pairs of these detections. Subsequently, Message Passing is applied to the graph to classify edges as active or inactive. An active edge can only connect two objects from different frames, indicating correspondence to the same real-world object.

spatial relation between detections. The following points describe the step-by-step procedure of this approach:

1. Graph Construction: During graph construction, nodes across any number of frames are connected, assuming that the geometrical distance between them can be covered in the interval between the number of frames in between for inter-frame edges. For intra-frame edges, they connect nodes if they can collide in the next frame.
2. Message Passing: A predetermined sequence of alternating updates on edges and nodes follows the general message-passing algorithm after constructing the graph. The edge update is executed by concatenating the edge features with the features of the end nodes and feeding the result into an MLP. The node update takes the max incoming message from the past, from the current time step, and, in the case of offline tracking, from the future time steps, concatenates them and uses an MLP to produce the wanted node update.
3. Edge Classification: After the message passing, they classify all edges, excluding intra-frame edges. If the classification result is positive, the edge represents a connection of the same object in two frames.

An important property of the framework is that localized polar coordinates instead of global cartesian coordinates represent the relative poses between detections. These poses then represent the edge features. The pose difference depends on the two detections instead of an additional reference frame. Since only the geometric states of the detection are required, PolarMOT is not coupled to a specific object detector. However, there is a clear difference in the performance between online and offline tracking since the message passing can use geometric cues from future frames as an additional source of information. PolarMOT achieves an AMOTA value of 0.66 on the nuScenes test set, ranking 42nd in the nuScenes tracking challenge, see Table 3.1.

3.3.4 Further Approaches

GNN-PMB

The GNN-PMB tracker [14] is a Multi-Object Tracking framework that uses Bernoulli Processes and Poisson Point Processes to model the state of objects in the tracking task. The Bernoulli process estimates the probability that a tracked object exists in the current frame.

Otherwise, it would be occluded. The Poisson Point Process gives the probability that a new object appears in the current frame. First, the 3D detection boxes are projected into Birds Eye View to get 2D bounding boxes. The remaining steps of the tracking process performed by the GNN-PMB tracker can then be summarized as follows:

1. Hypothesis Management for Data Association: Using the detections in the current frame, the propagated tracks from the previous frame, and possible new tracks that the Poisson Point Process initiated, the association cost matrix is created, which contains the probability that a certain detection corresponds to a specific track. After applying the Hungarian algorithm, there is an optimal matching. Matching probabilities resulting from that matching below a certain threshold are pruned and no longer seen as associated.
2. Update: After the final matching, they update the states of the existing and newly created tracks using the matched detections using a classical Kalman Filter.
3. Track Maintenance: If a detection is matched with a newly created track, it gets a new ID and is output; if it is associated with an existing track, it maintains its ID. If a track is not matched with a detection at one frame, the probability parameter of the Bernoulli process for that track decreases. If the adapted probability falls below a certain threshold, the track may be pruned and no longer considered a track.

This approach pursues a different strategy than that of most other works mentioned. However, it also achieves pretty good results compared to other LiDAR-based and Fusion-based approaches. More precisely, it achieves an AMOTA value of 0.68 on the nuScenes test set, ranking 26th in the nuScenes tracking challenge, see Table 3.1.

Approach	Year	Ranking	AMOTA ↑	AMOTP ↓	IDSW ↓	FRAG ↓	FP ↓	FN ↓
SimpleTrack [20]	2021	32	0.67	0.55	575	591	17,514	23,451
ByteTrackV2 [38]	2023	61	0.56	1.00	704	1,016	18,939	33,531
Camo-MOT [29]	2022	2	0.75	0.47	324	511	17,269	18,192
Polar-MOT [9]	2022	42	0.66	0.57	242	332	17,856	21,414
3DMOTFormer [8]	2023	29	0.68	0.50	438	529	18,322	23,337
PF-Track [19]	2023	78	0.43	1.25	249	839	19,048	42,758
GNN-PMB [14]	2023	26	0.68	0.56	770	431	17,071	21,521
InterTrack [31]	2023	15	0.69	0.56	1402	710	16,663	23,755
PolyMOT [13]	2023	1	0.75	0.42	292	297	19,673	17,956

Table 3.1: The tracking results of the approaches on the nuScenes test set. The ones that were described but are not in the Table have not been evaluated on the nuScenes test set. PolyMOT will be described in detail in Section 4. The ranking refers to the ranking in the nuScenes tracking benchmark and is taken from the paperswithcode website on Feb, 11 2024.

3.4 Other Works

Apart from the mentioned general approaches, there are some other concrete interesting works based on the mentioned principles. These works are especially interesting when considering the project this thesis is part of.

DMSTrack

The study proposed by Chiu et al. [6] introduces an innovative approach to cooperative tracking within autonomous driving systems. Cooperative tracking involves conducting tracking tasks using data from multiple sensors. While much of the existing literature in cooperative

perception focuses on object detection, DMSTrack takes a different approach by leveraging detections from individual sensors and producing overall tracking results. The methodology is detailed as follows: Each sensor resides on a connected autonomous vehicle (CAV), with each CAV transmitting its sensor data to an object detector and a neural network responsible for predicting the mean and covariance of detected object states. The central computing unit then receives and processes all detections, including their object states and covariances. The central unit manages sequential data associations, updates states, and initializes tracks using detections and covariances from each Connected Autonomous Vehicle (CAV). Upon processing data from all CAVs, the central unit generates the final tracking results. The subsequent Kalman Filter uses the results in its update step to propagate tracked objects to the next time step. The functionalities of the submodules mentioned in the tracking by detection approach, see Section 2.2.2, include:

1. Object Detection: A late fusion PointPillar model serves as the 3D object detector in each CAV [26].
2. Motion Prediction: The Constant Velocity Model is employed for motion prediction, enabling the utilization of the standard Kalman Filter due to its linear nature.
3. Data Association: To compute the affinity between detections and tracks, they utilize the 3DIoU method, and for the matching task, they utilize the Hungarian Algorithm.

Upon matching completion, the multi-sensor Kalman Filter incorporates the predicted covariance matrix of each CAV into its update step. They repeat the process of matching existing tracks with detections and performing CAV-specific Kalman Updates for each CAV in the same order. After processing data from the final CAV, matched tracks and detections are output, while unmatched ones are handled similarly to previous methods, such as in the work by Weng et al. [30]. The neural network designed to predict object states' covariances is a significant aspect of the study. This Covariance Neural Network utilizes local and positional object features to estimate the covariance of detection states. Training this neural network involves computing a regression loss on the final tracking results obtained after processing data from the last CAV. Since the approach is working in a different setup as the approaches described before due to the usage of multiple sensors, it is also evaluated on another dataset. Overall, DMSTrack achieves a performance AMOTA of 0.44 on the test set of the V2V4Real dataset [33].

Real-Time Multi-Object Tracking in Cloud-Based Environment

The approach proposed by [26] aims to address the challenges of real-time multi-object tracking in a cloud-based environment, mainly focusing on scalability and data synchronization from multiple sources. The Multi-Object Tracking framework achieves similar accuracy to the AB3DMOT tracking approach [30] while being 588%faster. It uses separate threads for all available sensors to increase inference speed. A Pre-Tracker thread is started for every sensor, keeping a local object list. The Pre-Tracker predicts the states of the objects from the previous frames, associates it with all current detections, updates the predicted states using a Kalman Filter, and passes the matched and new objects to a main fuser component. Each Pre-Tracker resembles a usual separate Multi-Object Tracking system. The main fuser maintains a global object list, which represents the aggregated information of all tracked objects across different sensors. Once the object lists are sorted and filtered based on age and quantity, the main fuser updates the global object list accordingly. This update involves integrating new object detections, associating them with existing tracks, and deleting tracks for objects that are no longer detected or are considered lost. Object association matches detections from different sensors to existing tracks in the global object list. This process helps in maintaining the identity of objects across different sensor views. Fusion combines information from multiple sensors about the same object to improve tracking accuracy. For example, the fusion com-

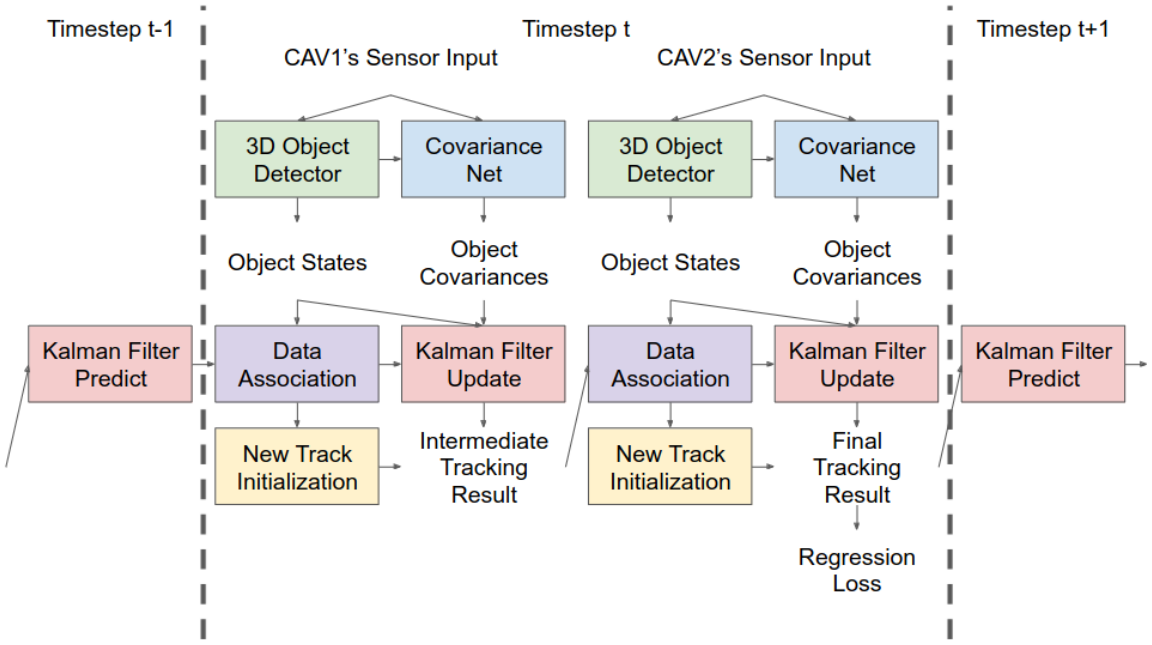


Figure 3.5: The figure depicts the DMSTrack procedure, illustrating the sequential processing of detections and covariances from all CAVs, along with the execution of Kalman Updates. Additionally, it highlights the point where the loss function, employed for training the Covariance Neural Networks, is computed.

bines position and velocity information from radar and lidar sensors. After association and fusion, the main fuser updates the state of existing tracks and initializes new tracks for newly detected objects. Lost tracks are deleted based on probabilistic estimations. The resulting algorithm runs at real-time speed, reaching very high frame rates. Applying the approach on the nuScenes test set results in an AMOTA value of 0.26 and 0.15 depending on which motion model is used internally by the tracking system.

Chapter 4

Tracking Approach

In this chapter, the tracking approach realized to solve the task of 3D Multi-Object Tracking is introduced and described in detail. Since the chapter is about the complete tracking approach, the Section 4.1 first describes the tracking pipeline proposed by [13] in detail. The Sections 4.1.1, 4.1.2, 4.1.3, 4.1.4 are only explanations of the already existing tracking approach of [13] as is, without any modifications. This is important to then better understand the features the baseline approach is extended with. Subsequent sections then present problems, failure patterns, and shortcomings of the adopted approach and analyze their impact on the tracking performance. Bringing our examination to a close, we introduce tailored solutions designed to address the identified issues, offering enhancements to the resilience and effectiveness of our tracking system. The Figure 4.1 shows the final overall architecture of the tracking system including the added features that are illustrated in the red boxes.

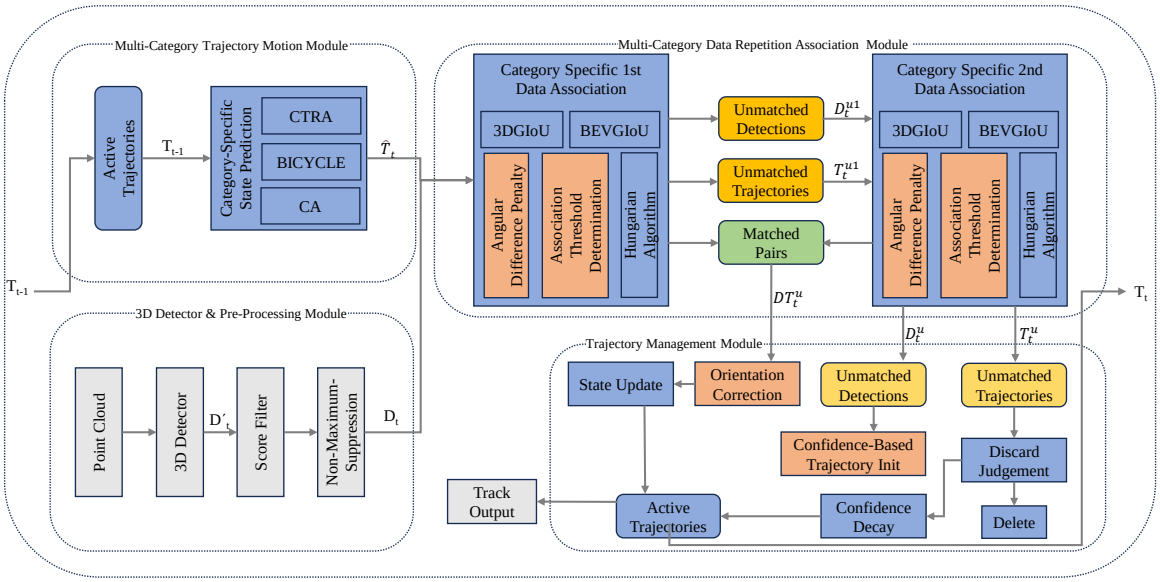


Figure 4.1: The figure shows the complete architecture of the realized tracking system. Most of it originates from the Baseline [13]. The added features are illustrated in the red boxes. T_{t-1} depict the existing tracks at frame $t-1$ while D_t and T_t depict the detections and existing tracks in the current frame t. $\hat{T}_t$ are the predicted states for the existing tracks at frame t.

4.1 Baseline

The tracking approach developed for 3D Multi-Object Tracking in this work is building upon the Polyhedral Multi-Object Tracking Framework proposed by [13]. We choose the baseline framework for multiple reasons:

- Efficiency through Learning-Free approach
- Best performance in the NuScenes Multi-Object Tracking challenge, see Table 3.1[4]
- Exclusive reliance on LiDAR data
- Precision in motion features through the use of LiDAR data

These attributes collectively contribute to the framework's effectiveness in meeting real-time constraints, making it a good choice for achieving our 3D Multi-Object Tracking objectives.

The mentioned approach is realizing the tracking-by-detection paradigm, explained in detail in Section 2.2.2. The core concept behind this approach involves the differentiation between various categories when performing the tracking tasks. Subsequent sections delve into the specifics of how this idea implements the tracking process. However, to further extend and improve the baseline work, additional features are added including a dynamic threshold determination and an angular difference penalty, explained in Section 4.2, compensating for some of the weaknesses observed during the execution of the basic approach.

4.1.1 3D Detector and Pre-Processing Module

As mentioned, the Polyhedral Multi-Object Tracking Framework is realizing the tracking-by-detection baseline. Therefore, an arbitrary 3D LiDAR detector delivers its detections to the tracking system. Each detection contains only geometric information about the object, like the position, the dimensions, or the orientation. Based on the information of the detections, the tracker then tracks them across multiple frames, assigning each object a consistent ID. As part of the pipeline, [13] realizes two preprocessing strategies to fortify and simplify the task. Standard object detectors used in autonomous driving output a confidence score in addition to the spatial information like position, dimensions, rotation, and the specification of the category of the detected object. The confidence score indicates how sure the detector is that the detection is indeed a True Positive. Based on the confidence, the PolyMOT tracker neglects all detections residing below a specified threshold as a preprocessing step and does not consider them in the tracking process. This threshold's magnitude depends on the object's concrete category following the essential idea of [13]. The main target of the Score Filter is to reduce the number of False Positive tracks. In addition, PolyMOT applies Non-Maximum-Suppression over all detection bounding boxes in one frame. This avoids tracking two objects, although only one of them exists in reality, resulting in a lower number of False Positives.

4.1.2 Multi-Category Trajectory Motion Module

The trajectory motion module's task is to predict objects' position. This happens by applying motion prediction models incorporating the last positions of the considered object. The tracking system's list of tracks includes all objects tracked over the last frames. Based on the motion history and a computed internal state, it uses specific physical motion models to predict where the object will move in the next time step. The exact choice of the motion model for an object is category-specific. The used model for each category is specified based on the typical motion patterns of an object of the respective category. Predicting the position constitutes the prediction step of the Kalman Filter, mentioned in 2.3. The plot 4.2 shows the used motion models. The following motion models are available:

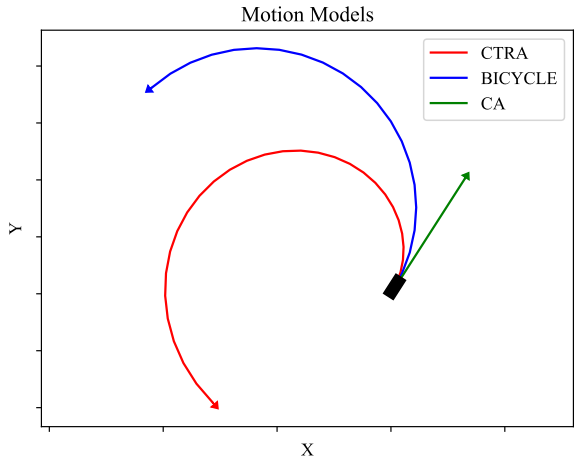


Figure 4.2: The figure illustrates a typical path predicted by three employed motion models. The initial yaw orientation of the object in radians is set to 1 in this illustration. The plots demonstrate that the Constant Turn Rate and Acceleration (CTRA) and Bicycle (BICYCLE) models account for orientation changes, allowing them to represent curved motion patterns. In contrast, the Constant Acceleration (CA) model does not have this capability.

- **Constant Turn Rate and Acceleration Model:** The CTRA model predicts the position, the velocity v, the orientation θ, the acceleration a, and the turn rate ω while a and ω stay constant [25]. The predicted orientation is computed by $\theta_{k+1} = \theta_k + \omega \cdot T$; the predicted velocity is $v_{k+1} = v_k + a \cdot T$, where T is the time difference between two frames. Sine and cosine are applied to the position and velocity in the preceding timestep to compute the position. By assuming a steady acceleration in rotation and position, the model can capture non-linear motion patterns of objects, which traffic participants in urban areas often perform.
- **Bicycle Model:** The bicycle model assumes the vehicle to have only one wheel in the front and one in the back, imitating a bicycle. As described in [21], the motion predicted by the Bicycle model depends on the steering angle of the vehicle as well as on the acceleration. Using the steering angle δ and the rotation θ to compute the direction of the velocity allows us to compute the change in position in x as well as in y-direction: $\dot{x} = v * \cos(\theta + \beta(\delta))$ and $\dot{y} = v * \cos(\theta + \beta(\delta))$. $\beta(\delta)$ is the slip angle at the center of gravity as explained in [21]. Like the CTRA model, the Bicycle model can model non-linear motion patterns of vehicles.
- **Constant Acceleration Model:** This model predicts the position of an object by assuming a constant acceleration between two consecutive timesteps. The prediction can be described by: $x_t = x_{t-1} + v_{t-1} \cdot \Delta t + 0.5 \cdot a_{t-1} \cdot (\Delta t)^2$. In contrast to the other two models, it cannot capture non-linear motion.

The tracking process's next step uses the Kalman Filter's predicted position. This step is elucidated in section 4.1.3. An important initialization step for the motion models is the velocity at the beginning of a track. This initialization is important and independent of the motion model used in the concrete case. Performing tracking in scenarios where the vehicles normally have high speed, e.g., on highways, the velocity initialization is far more important than in scenarios where the vehicles move rather slowly. This is because fast objects travel more meters in the same time interval as slow objects. Therefore, the distance between the position of an object at two consecutive timesteps is much bigger. If the position prediction assumes a far too low velocity, it will predict a position that is not close to the actual position. In the intersection scenario, a velocity initialization with a speed of $0\frac{m}{s}$ is sufficient, while for the high-speed scenarios, a relatively high initialization speed produces valuable results.

4.1.3 Multi-Category Data Repetition Association Module

The system performs the data association using the preprocessed detections from 4.1.1 and the predicted position for the currently tracked objects. More precisely, the goal is to associate each of the detections with an existing track by computing their affinity. This affinity resembles the probability that the detection and the track correspond to the same real-world object. There are various ways to measure the affinity between two objects. This work computes affinity by calculating distance metrics between detections and tracks. The distance metric used in the concrete case is category-specific to be as flexible as possible and accommodate different object characteristics. Possible metrics include the 3DGIoU and the BEVGIoU. These metrics are based on the Generalized Intersection over Union as explained in 2.4. BEVGIoU uses the BEV projected area as input to the 2D-IoU, while 3DGIoU directly uses the cuboid volumes. The affinity matrix C for one category is created by computing the distances between all pairs of detections and tracks within this category. A pair of detection d and track e is considered unreachable in C if the distance $d(d,e)$ exceeds the category-specific threshold. Each category's affinity matrix C is then fed into the Hungarian Algorithm to generate an optimal matching.

An additional feature is the two-stage data association. Its idea is to always compute the distance between a pair of detection and track with two different distance metrics resulting in C_1 and C_2. Furthermore, the thresholds for the two different stages can be different. The following order then summarizes the process of the data association:

1. Compute the affinity C_1 between all pairs of detections and tracks for all categories and use the first threshold to prevent wrong matches.
2. Apply the Hungarian algorithm to C_1 for all categories.
3. Take all detections and tracks not matched in the first stage, compute their affinity matrices C_2, and use the second threshold to prevent wrong matches.
4. The matched detections and tracks are now fixed, while the unmatched tracks and detections are handled by the next module explained in 4.1.4.

After that, the matched pairs between detections and tracks have been determined, and the information about the detection bounding box is used to update the state information of the tracked object. This state information update corresponds to the update step of the object's Kalman Filter 2.3. Consequently, the states of unmatched tracks cannot be updated. In these cases, the updated states correspond to the predicted state.

4.1.4 Trajectory Management Module

The Trajectory Management Module takes the matched pairs of detections and tracks, the unmatched detections and the unmatched tracks, and decides which tracks to remove, which to add, and which to output. [13] includes the following features:

- Tentative Trajectories: A track is only outputted at a specific frame if matched with a detection in the last N consecutive frames. N is a category-specific fixed parameter.
- Dead Trajectories: A track is only declared dead and no longer considered in the data association if the tracking algorithm has not matched it with a detection in the last M frames. Again, M is a handcrafted category-specific parameter.
- Valid Trajectories: Valid trajectories include all tracks that are no longer tentative but are not yet dead.
- Track Output: Additionally to the tracks matched in the current frame, it is possible to specify the number of frames for which the track is output after the last frame the tracker matched it with a detection. Therefore, the information on the object's bounding box is solely based on the prediction from the Kalman Filter.

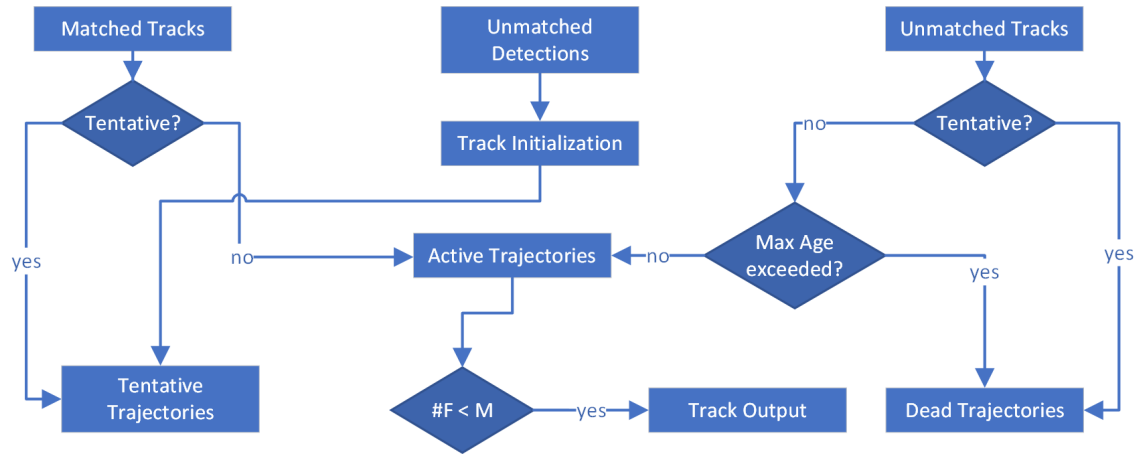


Figure 4.3: The figure illustrates the state management of tracked objects. Given the currently tracked objects and the detections in the current frame, the tracker attempts to establish a matching between them, as explained in Section 4.1.3. Subsequently, we categorize the objects into matched detections, unmatched detections, and existing tracks that have not been matched with a detection. The diagram outlines the process of determining which objects are no longer considered for tracking, which we still consider in the next frame, and which objects are output in the current frame. The $\#F$ mentioned in one box denotes the length of the track of a tracked object in frames. M is a constant representing the minimum length of the track of an object in frames to be considered for output.

- Post-NMS: PolyMOT also applies Non-Maximum-Suppression to all tracked bounding boxes output at the current frame.

4.2 Limitations

Several considerations became apparent after implementing the adopted tracking system and performing initial evaluations. This section explains some important aspects influencing the overall tracking performance in detail and how these aspects are interrelated with general tracking challenges elucidated in 2.1.

4.2.1 Imperfect Object Detection

As mentioned, the presented tracking system works based on detections predicted by the 3D LiDAR object detector integrated into the project. Like every other detector, its results could be better, leading to wrong information about the object's state or missed detection of objects. E.g., in the left column of Figure 5.5, one can see that if the detector detects an object in one frame, it does not necessarily detect it in the following frames. Another example is Figure 5.12, where object information, more precisely the object's orientation, is mispredicted in the upper images in Frame 414.

4.2.2 Influence of Orientation

The first conclusion drawn from that is the awareness of the intricate relationship between the orientation of detected objects and the performance of the presented multi-object tracking system. Understanding the impact of the orientation of the detected objects is crucial for

enhancing the robustness and accuracy of the tracking system. As elucidated in Section 4.1.2, the PolyMOT tracker employs specific motion models that leverage object orientation for predicting the position of known tracks. Two of the mentioned motion models require the object orientation of the detection to update the Kalman Filter's internal state. While the CA model disregards the object orientation, focusing solely on velocity prediction, the CTRA and the BICYCLE models use the orientation to predict the potential curve the object will move along. A wrongly assumed object orientation leads the motion prediction to produce a wrong state update following a wrong and lousy motion prediction. Using this incorrect prediction from the last frame for the affinity computation in the current frame can be difficult. Even if the detector detects the real-world object corresponding to the lousy prediction in the current frame, it is likely that the matching algorithm cannot match them since the distance might have gotten too big in the meantime. Hence, wrong object orientations make the data association immensely worse.

Problem

The importance and possible resulting failures become apparent based on the example shown in 4.4. The orientation of the detected car in the first image of frame 413 is correct, while it is incorrect in the following frame 414. The missing detection of the car in frame 415 leads to a wrongly predicted trajectory illustrated in the lower row in frame 415 4.4. The predicted motion takes a turn to the left, although the car follows a straight trajectory in the real world. If the car gets detected in a later frame again, comparing it to the predicted position can then be challenging. The reason is that the two objects are now very distanced from each other, resulting in a low affinity between them. Applying the association mechanism explained in Section 4.1.3 will not lead to a match, resulting in a new ID for the car. The frequency of occurrence of this pattern, characterized by instances of both missing detections and inaccurately predicted orientations, is notable.

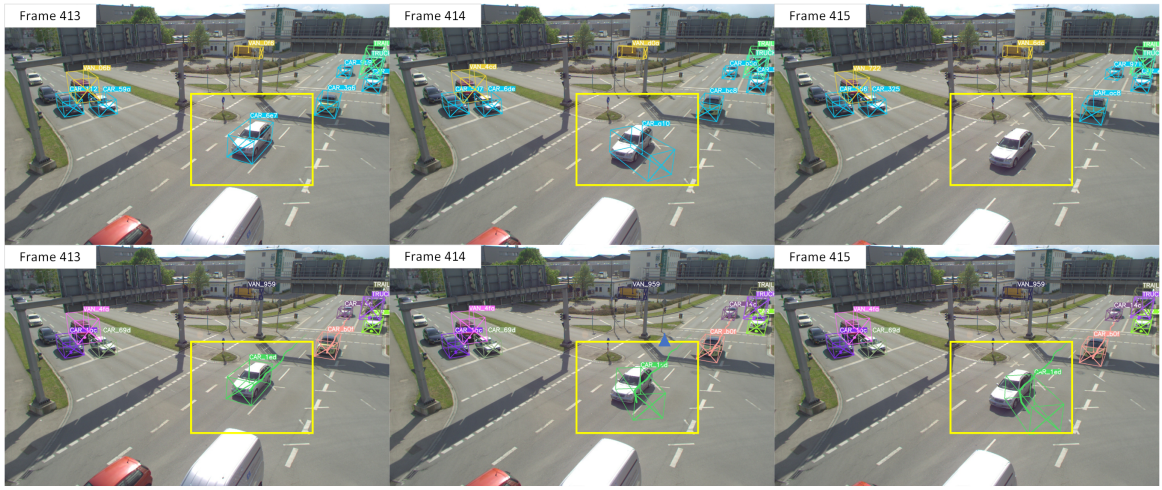


Figure 4.4: A sequence of three consecutive frames is presented. The three upper images display the detections delivered by the detector without tracking, while the three lower images depict the detections after being processed by the tracker. In frame 413, the detector detects the car correctly, but in frame 414, the detector predicts a wrong orientation. In frame 415, the detector completely misses the object. Consequently, the tracker attempts to predict the motion of the car for frame 418 based on prior information. Since the orientation was incorrect in frame 414, the tracker assumes the car to take a left turn, inaccurately capturing the object's actual motion.

Orientation Plausibility Check

To address the issue of wrongly predicted object orientations presented in Section 4.2.2, the addition of an orientation plausibility check is suggested. The basic idea to decrease the influence of incorrect detection orientations on the tracking performance is pretty straightforward. After the two-stage data association, all matches between detections and existing tracks are determined. Since the detection is now matched, it is associated with an existing track. Therefore, it has a history of states, including past positions, velocities, and orientations. These are now used to check the correctness of the current detection orientation. We only consider the orientation of the tracked object from the last frame θ_e and the orientation of the detection in the current frame θ_d. The added feature compares the two orientations to each other. If the difference between them differs more than a certain amount, i.e. more than 0.35 radians, the orientation of the detection box θ_{d_new} is assumed to be wrongly predicted. In that case, the bounding box of the detection is set to have the same orientation as the internal state of the tracked object proposed. A manual parameter search determines the exact maximal value of the angular difference. Concisely formulated, this idea looks as follows:

$$\theta_{d\_new} = \begin{cases} \theta_e, & \text{if } \theta_e - \theta_d > \alpha \\ \theta_d, & \text{otherwise} \end{cases}$$

The basis of this check is the assumption that an object cannot change its orientation over more than α in one timestep between two frames. Extreme orientation changes within one timestep are unlikely, especially since the difference in time is usually far smaller than one second. After updating the detection orientation, the Kalman Filter and the motion model should be able to produce more valuable predictions.

Moving Average Filter for Rotation

As an alternative solution to the solution proposed in Section 4.2.2 we added another way of handling the challenge of wrongly predicted orientations. The idea is to apply a simple moving average filter to the orientations of a tracked object in the last frames. The number of past considered frames is fixed. The resulting orientation of an object is then computed by:

$$\theta_{d\_new} = \frac{1}{p} * \sum_{i=t-p+1}^{t} \theta_i$$

θ_i is the orientation of the object in the frame at time i, p is the number of past orientations that is taken into account when applying the moving average filter and θ_{d_new} is the orientation which is then used as the update orientation of the object.

4.2.3 Occlusions and Missed Detections

The concept of occlusions elucidated in Section 2.1 is similar to the problem of missed detections by the detector. In both cases, the tracking system has to predict the object's position correctly, such that it can be associated with a frame where the object is detected again. In the case of an occlusion, this frame would be the third image of Figure 2.2 while in the example of missed detections, it would be at the left column of Figure 5.5 in frame 148. This association is usually very challenging since the time difference between the two frames where the object is detected can be multiple seconds, resulting in a multiple of tens of frames where the tracker has to predict the object's state. The goal is to avoid association failures due to lousy motion predictions, especially in occlusion situations. The orientation of the last detection

of the object is never wholly correct. Minor deviations already strongly influence the motion prediction for an object, resulting in a position prediction that does not correspond to the proper position. For every missed detection, the tracker skips the update step of the Kalman Filter. Consequently, the deviation is getting bigger and bigger with every frame for which the detector reports no detection for the tracked object. The reason for that is the classical principle of fault propagation. As already explained, lousy quality in position predictions affects the ability to generate a correct association in the frame where the object is first detected again. We propose the following two strategies to enable and enhance the required association.

Adaptive Threshold

The first idea is to use adaptive dynamic thresholds in the two-stage data association instead of fixed static ones. This idea was also realized by [32]. Figure 4.5 illustrates this strategy. With every time step where the Kalman Filter is only predicting but not updating its internal state, we decrease the association threshold, making it easier for objects to be associated. The fact that the track of an object is kept alive for a certain number of frames, although not being matched with a detection, enables us to realize that idea. For every consecutive frame where the matching algorithm can not match the track with a detection, the association threshold δ_{c_new} is dynamically altered and determined by:

$$\delta_{c\_new} = \lambda * \delta_c$$

where the factor λ is determined manually. Figure 4.5 shows the intended effect.

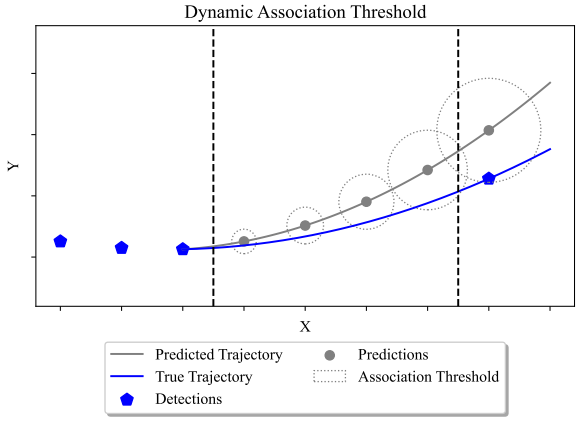


Figure 4.5: Adopted from [32]. As elaborated in Section 4.1.4, information about a tracked object is retained even when it goes undetected for several frames. During this period, the object's position relies solely on predictions from the motion model. However, as the motion model cannot accurately predict the object's motion without measurement information for the Kalman Update step for a long time, the prediction quality degrades with each missing measurement. Consequently, when a measurement becomes available after a certain number of timesteps, the predicted position may be so distant from the true position, making the correct data association challenging. Increasing the threshold with each missing measurement could enhance the likelihood of correctly re-establishing the match with the object.

Angular Difference

Another point to start is to directly modify the computation of the affinity used for the association. The key idea is to adjust the affinity based on an additional distance metric applied between the detection position and the last position of the tracked object associated with the

detection. When predicting a moving object's position, its uncertainty is more significant in the direction of motion than perpendicular to it. As the detector itself used in this work does not provide velocity information, the velocity for motion prediction is estimated or computed based on the object's last positions and orientations. Since an object primarily moves along the direction of the estimated velocity vector, the predicted position in that direction may not be completely accurate. Conversely, movements perpendicular to the motion direction are typically small, as objects usually do not exhibit such behavior. Consequently, penalizing movements perpendicular to the motion direction more heavily than movements along it makes sense. The resulting computation of the affinity measurement between an existing track e and a detected object d can be defined as:

$$C_{d,e} = GIoU(p_d, p_e) - \gamma * ang\_diff$$

where $C_{d,e}$ is the assignment cost between d and e, γ is a fixed pre-factor and ang_diff is computed as:

$$ang\_diff = |\theta_t - \theta_d|$$

where θ_d is computed by

$$\theta_d = atan2(x_d, y_d)$$

and x_d and y_d by

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \boldsymbol{p_d} - \boldsymbol{p_e}$$

Here, p_d is the position of the incoming detection, and p_e, θ_e are the position, and orientation of the predicted position of an existing track. The resulting $C_{d,e}$ is then compared against the same threshold as before to determine the association between detections and tracks. The factor γ is manually fixed through an extensive parameter search and remains constant across all categories. The accompanying Figure 4.6 provides a visual representation and further detailed explanation.

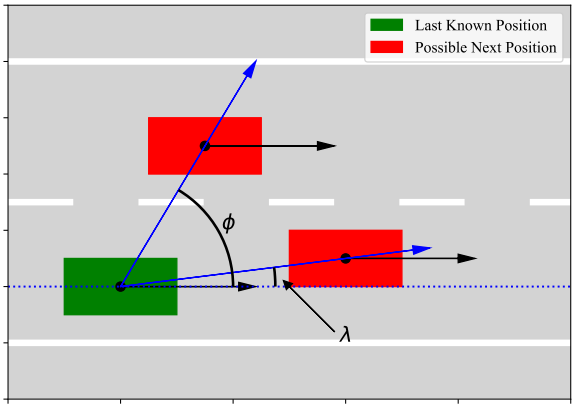


Figure 4.6: This illustration pictures the concept behind the angular difference penalty. The velocity vector of an object usually aligns with the direction of motion. Consequently, motion occurring perpendicular to this direction is unlikely, implying increased uncertainty in the predicted position along this perpendicular direction. The affinity between two objects is heightened if one lies in the motion direction of the other and penalized otherwise. While the red object in the left lane might have a higher affinity with the green object than the red object in the right lane, the latter is more likely to move toward the position of the green object due to the smaller value of λ compared to ϕ.

The just explained computation of the angular difference can also be used as part of an additional similarity metric itself. Combining the simple euclidean distance metric with

the angular difference results in a similarity metric which does not require to compute the Generalized Intersection over Union metric. That can be of advantage since the computation of it requires determining the convex hull which is a computationally intensive task. The following distance metric could then look like the following when we want to compute the similarity between an existing track e and a detected object d:

$$C_{d,e} = d_{eucl}(d,e) + d_{eucl}(d,e) * ang\_diff$$

The computation of the ang_diff is performed the same way as described above. The computation of the metric only requires the computation of the euclidean distance as well as the angular difference which are two quite easy computations.

Modification of Trajectory Lifecycle

In Section 4.1.4 the concept of tentative tracks as well as the concept of tracks that are not matched at a current frame but still output is introduced as intended by [13]. Tentative tracks have a number N_{min_hit} which states the number the track has to be matched with a detection to be output. In this way, false positive output tracks are avoided. However, if N_{min_hit} is set too high, the number of false Negative objects also increases. To mitigate the influence of a high number of false negatives, we propose to only use N_{min_hit} when initializing a track with a detection with a confidence score above a certain threshold γ. If a detection d that is used to initialize a new track t has a confidence score above a certain threshold, we set this track to active immediately:

$$N_{min\_hit} = \begin{cases} 1, & \text{if } \gamma < CS(d) \\ N_{min\_hit}, & \text{otherwise} \end{cases}$$

$CS(d)$ tells the confidence score of detection d. We set γ to 0.9. Therefore a detection has to have a confidence score of at least 0.9 to be output as a track immediately.
We propose another modification to the trajectory management module proposed by [13] concerning the output of tracks. In the basic adopted version of the tracking system a number of frames M can be specified for which a track t is output after it was last matched with a detection, see Section 4.1.4. Additionally, the trajectory management module of [13] decreases the confidence score of a track for each frame it was not matched with a certain factor. We use this confidence score decay to determine the concrete value of M. This happens similar like the principle of the tentative trajectories are modified. We output the track as long as its confidence score is above a certain threshold. So, an object which was last detected with a high confidence score will be output for more frames than an object with a low score:

$$Output(t) = \begin{cases} True, & \text{if } \nu < CS(t) \\ False, & \text{otherwise} \end{cases}$$

$CS(t)$ again depicts the current confidence score of track t while ν determines the confidence threshold for a track to be output.

Chapter 5

Evaluation and Results

In this chapter, the adopted and extended 3D Multi-Object Tracking system initiated by [13] is evaluated on two different datasets: 1) TUM Traffic A9 Highway dataset [7] and 2) TUM Traffic Intersection dataset [39]. The main focus of this work is on intersection data. However it is interesting to check if the tracker also performs well on highway data. The performance of the tracker in both scenarios is compared to the currently used tracker to check if this tracker outperforms it. This comparison gives us a more general ranking of the tracker's performance. The results are reported in quantitative and qualitative aspects. Additionally, we illustrate the search for the optimal hyperparameters necessary for tracking. After that, the added features to the tracking system are analyzed and checked to see if they can further improve the overall performance of the adopted tracker in specific more complex scenarios.

5.1 Setup

The evaluation of the tracking system works based on detections generated by a 3D detector. The detections for the intersection scenarios, as well as for the highway scenarios, are generated in advance and are used for every run of the tracking. The detector delivers the detections in OpenLABEL [18] format, and the tracking system outputs the detections in the same format but with rectified track IDs. The output detections are then used to compute the basic metrics described in Section 2.5.3 and the advanced metrics like MOTA or MOTP, explained in Section 2.5.4. The motmetrics Python library builds the base of the implementation of the evaluation. It is a prevalent Python library that facilitates the computation of standard Multi-Object Tracking metrics based on the CLEAR MOT metrics [1]. Initially determined for 2D Multi-Object Tracking, it can also perform the evaluation for 3D Tracking by feeding its central accumulator element with the list of IDs of the predictions, the list of IDs of the ground truth, and a cost assignment matrix at each frame. The computed metrics rank the quality of the tracking results in multiple aspects. We calculate the metrics for the newly adopted trackers and the currently used 3D tracker, i.e., a modified SORT tracker (SORT3D). The original SORT tracker [3] is used for 2D Tracking but was extended in previous work of the project to also handle 3D detections. Apart from comparing the performance results produced by the motmetrics library, given visualizations compare the tracking results in specific scenarios. These visualizations are generated using the tum-traffic-dataset-dev-kit. Since the tracker presented in Section 4 is only using physical computations, the configuration of the tracker includes many parameters that need to be optimized. The optimization can present an elaborate task, mainly because executing the tracking system in the urban intersection scenario requires different parameters than in the highway scenario. Section 5.5 illustrates the search for the optimal hyperparameters. The tracker uses category-specific parameters,

see Section 4, as well as general parameters. We use a M of 3, see Section 4.1.4, an angular difference factor λ of 0.3 and a threshold θ_{SA} of 1.1 in the second association stage for all categories; the parameter search concluded that the second stage association threshold is independent of the category. Furthermore we use 3DGIoU in the first association stage and BEVGIoU in the second stage. We use the orientation plausibility check instead of the moving average filter, see Section 4.2.2 since it produces better results. We also do not use the modified output strategy depending on the confidence score of the track but use the standard output decision process of [13] for the same reason. The confidence threshold ν for directly initializing a detection as an active track is set to 0.9. The category specific parameters including the first association threshold θ_{FA} which is a scalar value since the metric is Generalized Intersection over Union, Score Filter θ_{SF}, minimal hit number N_{min_hit}, maximal age number N_{max_age} are listed here:

	Bike	Bus	Car	M.Cycle	Ped.	Trailer	Truck	Van	Emerg.	Other
θ_{FA}	1.5	1.4	1.1	1.4	1.1	1.5	1.3	1.3	1.3	1.3
θ_{SF}	0.39	0.45	0.41	0.41	0.41	0.4	0.45	0.41	0.41	0.41
N_{min_hit}	3	2	3	2	2	3	3	3	3	3
N_{max_age}	15	14	15	15	15	12	12	15	20	10

Table 5.1: Used hyperparameters for the intersection data used for the evaluation of the given sequences.

5.2 Intersection Data

5.2.1 Data Description

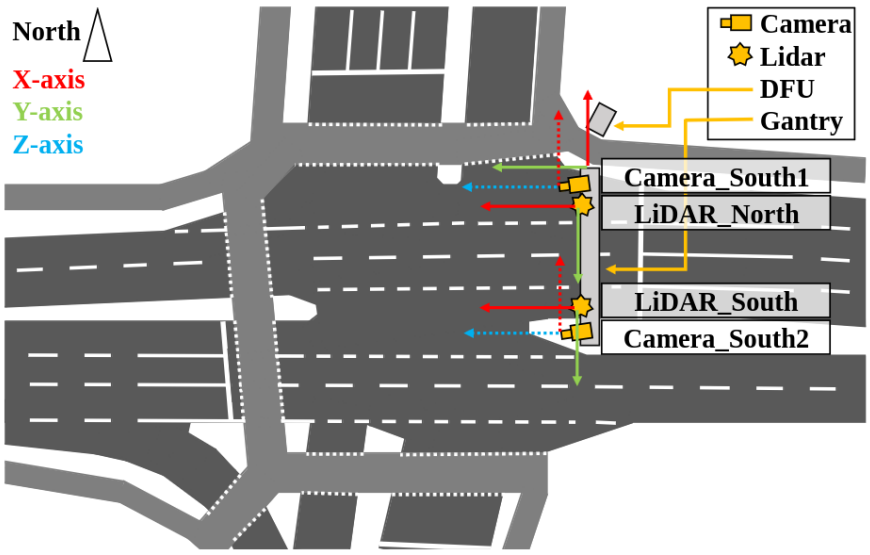


Figure 5.1: This image depicts the intersection from which the utilized data originates. It also illustrates the concept of the infrastructure view, where cameras and sensors are mounted on a sign gantry, consistently observing the same environment.

The tracker is applied to the sequences from the TUM Traffic Intersection Dataset [39] to test the tracking system in an urban environment. It contains sequences of different lengths recorded at a busy intersection at Garching near Munich. Figure 5.1 shows the setup at the

intersection and the intersection itself. The intersection equipment includes two LiDAR sensors and multiple cameras. Using the LiDAR data, a 3D detector, based on the principle of the primary PointPillar [11] detector used in the Providentia project, is applied to generate the detections for the tracking. The data is available at a frequency of 10 Hz. All four mentioned sequences include labeled tracks in the ground truth, making it very easy to measure the performance of the applied tracker. The first two sequences are relatively short and recorded in usual traffic situations. The third one is quite long, with 1,033 frames, including very crowded and relatively sparse scenarios, while the fourth is a sequence recorded at night. Hence, the third sequence is the most interesting one. More detailed information about the dataset including category specific numbers about the number of objects and track lengths can be found in [39].

5.2.2 Quantitative Results

In the following two subsections, we compute and compare the performance metrics of the modified PolyMOT tracker and the existing SORT3D tracker. The results are compared against each other and interpreted. The first subsection includes the comparison using basic tracking metrics, see Section 2.5.3. The basic metrics evaluation results are compared with the ground truth values. The second subsection shows the results from applying the advanced CLEAR-MOT metrics of Section 2.5.4.

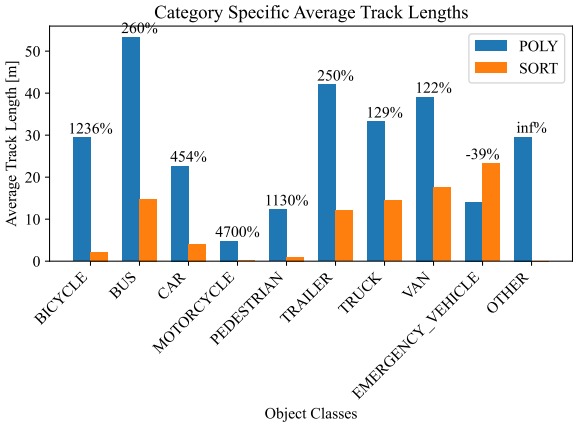
Basic Metrics

The tables 5.2, 5.3, 5.4, 5.5 show the evaluation results of the basic metrics evaluated on the four mentioned sequences. The goal is to keep the tracking results as close as possible to the values computed from the ground truth. Firstly noticeable is that the number of track IDs of the PolyMOT-Tracker is always quite similar to the number of ground truth IDs, very in contrast to the number of Track IDs returned by the SORT3D tracker, which is very high in each of the scenarios. Also, the average track length in meters and frames computed from the used tracker's results are much higher and more similar to the ground truth values compared to the results of the SORT3D tracker. This observation is accurate for each one of the four sequences. Using the Trajectory Management Module, see Section 4.1.4, is the reason for the better number of total track IDs of the PolyMOT-Tracker. The module judges the correctness of the object to be a new tracked object. This judgment also influences the total length of all tracks, leading to a better result for the SORT3D tracker. The SORT3D tracker assumes every delivered detection to be part of a correct track, leading to many different tracks that are partly incorrect and mostly incomplete.

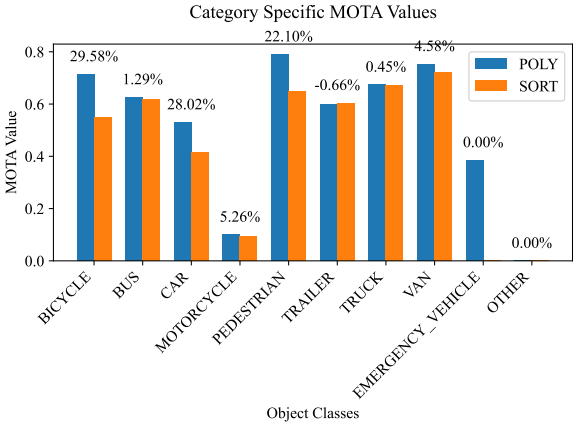
Figure 5.2(a) shows again the clear advantage in the average track length per object of the PolyMOT-Tracker. The difference is especially immense in the cases of small objects like bicycles, motorcycles, and pedestrians. One reason for that is likely to be the choice of the distance metric used for the affinity computation. The SORT3D tracker uses the Intersection over Union, which leads to the case of an affinity of zero if the objects do not have an intersection at all. At the same time, the PolyMOT-Tracker has a measurement of the affinity of two objects even if they do not have an intersection because of the Generalized Intersection over Union, see Section 2.4.

CLEAR-MOT Metrics

The Tables 5.6, 5.7, 5.8, 5.9 show the results of the CLEAR MOT metrics computation. Here, the arrows next to the metric's name indicate if the desired value for this metric is high or



(a) The percentage of the increase of the average track length of the SORT3D tracker to the PolyMOT-Tracker.



(b) The percentage of the increase of the MOTA metric of the SORT3D tracker to the PolyMOT-Tracker.

Figure 5.2: category specific tracking results originating from the evaluation on the third sequence $R02\_S03$.

	#Track IDs	Total Track Length[m]	Total Track Length[frame]	Avg. Track Length[m]	Avg Track Length[frame]
SORT3D	183	**702**	4466	3.8	24.4
PolyMOT	**41**	684	**4450**	**16.7**	**108.5**
GT	41	1088	5178	26.5	126.23

Table 5.2: Results Sequence R02_S01.

	#Track IDs	Total Track Length[m]	Total Track Length[frame]	Avg. Track Length[m]	Avg Track Length[frame]
SORT3D	188	1113	**3604**	5.9	19.2
PolyMOT	**47**	**1115**	3521	**23.7**	**74.9**
GT	57	3362	4953	59.0	86.9

Table 5.3: Results Sequence R02_S02.

	#Track IDs	Total Track Length[m]	Total Track Length[frame]	Avg. Track Length[m]	Avg Track Length[frame]
SORT3D	989	**4487**	**13843**	4.5	14.0
PolyMOT	**177**	4453	11336	**25.1**	**75.3**
GT	153	5453	15334	35.6	100.2

Table 5.4: Results Sequence R02_S03.

	#Track IDs	Total Track Length[m]	Total Track Length[frame]	Avg. Track Length[m]	Avg Track Length[frame]
SORT3D	181	**906**	3401	5.0	18.8
PolyMOT	**27**	844	**3407**	**31.2**	**126.1**
GT	26	1301	5018	50.0	193.0

Table 5.5: Results Sequence R02_S04.

low. The precision of the tracking algorithms given as MOTP is similar for both trackers in all sequences, while the PolyMOT-Tracker has clear advantages in the MOTA metric. Also, the HOTA metric, which evaluates the tracking performance more completely by incorporating the assessment of the localization accuracy and the association accuracy, is higher in all sequences for the results of the PolyMOT-Tracker. Reasons for the higher HOTA are, among others, the much lower number of ID-switches happening and the lower number of false negatives and false positives. Also, the number of fragmentations in the results of the SORT3D tracker is much higher. The low number of ID switches could already be assumed since the total number of track IDs, as explained in Section 5.2.2, is very low and comparable to the number of ground truth IDs. Also, the comparison of the category-specific MOTA values shown in Figure 5.2(b) shows the advantage of the PolyMOT-Tracker for almost every category. However, again, the difference between PolyMOT and SORT3D is especially noticeable for objects from categories with typically lower dimensions like bicycles, motorcycles, or pedestrians, as already mentioned in 5.2.2.

	FN ↓	FP ↓	MT ↑	PT ↑	ML ↓	IDSW ↓	FRAG ↓	HOTA ↑	MOTA ↑	MOTP ↓
SORT3D	1069	357	19	7	15	88	77	0.77	0.71	**0.25**
PolyMOT	**886**	**158**	**19**	**8**	**14**	**5**	**17**	**0.86**	**0.80**	0.26

Table 5.6: Results Sequence R02_S01.

	FN ↓	FP ↓	MT ↑	PT ↑	ML ↓	IDSW ↓	FRAG ↓	HOTA ↑	MOTA ↑	MOTP ↓
SORT3D	**1536**	187	13	**25**	**19**	96	84	0.64	0.63	0.34
PolyMOT	1547	**115**	**15**	20	22	**4**	**13**	**0.76**	**0.66**	**0.32**

Table 5.7: Results Sequence R02_S02.

	FN ↓	FP ↓	MT ↑	PT ↑	ML ↓	IDSW ↓	FRAG ↓	HOTA ↑	MOTA ↑	MOTP ↓
SORT3D	4388	2897	54	**61**	**38**	351	307	0.57	0.50	0.45
PolyMOT	**4177**	**2179**	**60**	53	40	**26**	**75**	**0.68**	**0.58**	**0.45**

Table 5.8: Results Sequence R02_S03.

	FN ↓	FP ↓	MT ↑	PT ↑	ML ↓	IDSW ↓	FRAG ↓	HOTA ↑	MOTA ↑	MOTP ↓
SORT3D	1820	203	9	**10**	**7**	105	107	0.52	0.57	0.59
PolyMOT	**1672**	**61**	**10**	8	8	**5**	**34**	**0.72**	**0.65**	**0.58**

Table 5.9: Results Sequence R02_S04.

5.2.3 Qualitative Results

In this section, we demonstrate some specific scenarios that illustrate the strong tracking performance of the PolyMOT-Tracker that has already been described in Section 5.2.2.

Tracking of Smaller Objects

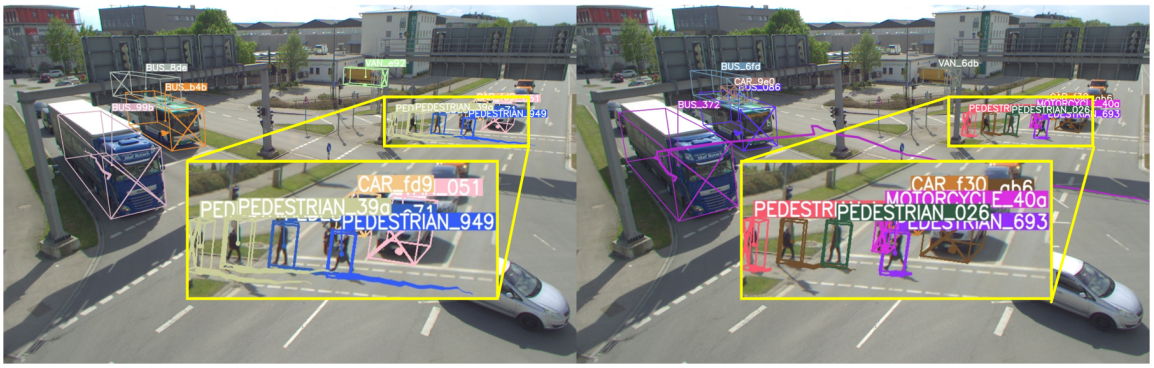


Figure 5.3: The tracking results of both considered trackers are compared in a concrete scenario involving the Tracking of numerous small objects, such as pedestrians. The left image displays the results from the PolyMOT-Tracker, while the right image shows the results of the SORT-Tracker.

Figure 5.3 shows the qualitative comparison between the PolyMOT-Tracker and the SORT3D tracker in a concrete case where many small objects like pedestrians are present. The tracking results from the PolyMOT-Tracker are illustrated in the left image, and the results from the SORT3D tracker are in the right image. The line behind each bounding box depicts the object's past position with the bounding box's concrete track ID. These lines are clearly

longer for objects in the left image than in the right image. This fact underlines the fact that PolyMOT-Tracker has significant advantages if it comes to the task of tracking pedestrians or generally smaller objects.

Occlusion Recovery

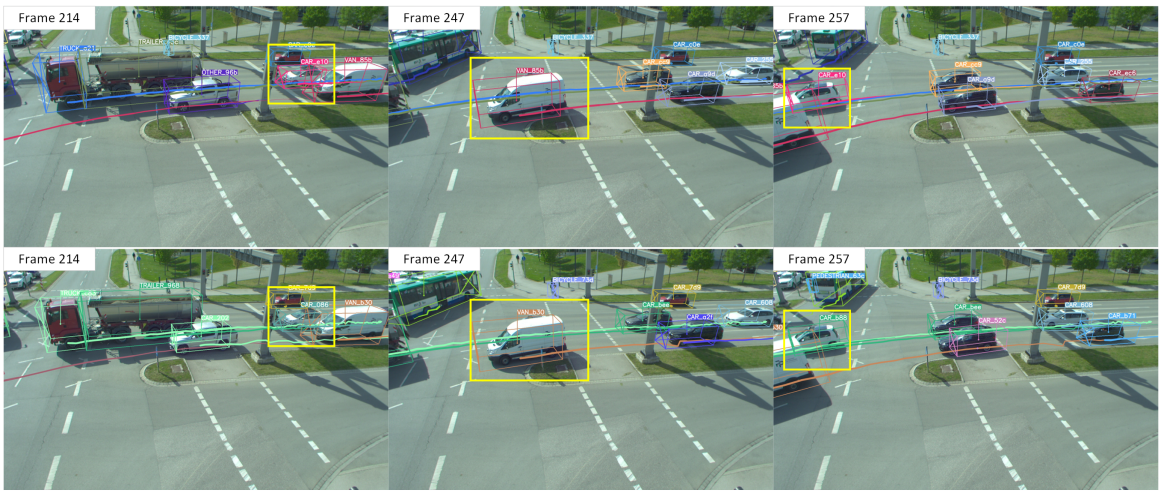


Figure 5.4: The images depict a time span during which a small car is occluded by a van driving next to it. The three upper images display the tracking results of the PolyMOT-Tracker, while the three lower pictures show the tracking results of the SORT3D tracker. The small car is occluded for 14 frames. PolyMOT still has the track of the small car in memory since 14 frames is lower than the used maximal age parameter of 15. However it does not outputs it only for the first couple frames of the occlusion. SORT3D has a maximal age of 1. After not being matched with a detection for more than one frame, a track is directly removed from memory.

Figure 5.4 shows another scenario where the PolyMOT-Tracker performs better than the SORT3D tracker. The upper three images visualized the results of the PolyMOT-Tracker, while the lower three originate from the results of the SORT3D tracker. The considered scenario includes a small car temporarily fully occluded by a van. Therefore, the detector cannot detect the small car for a few frames. The goal is to assign the same ID to the small car in frame 257 as it has in frame 214. The challenge is to correctly predict the object's motion for a couple of frames. One can see that the object keeps its ID in the upper images, while the small car in the lower images changes its ID after the occlusion.

Effect of the Trajectory Management Module

The trajectory management module explained in Section 4.1.4 enables the tracking system to compensate for missing detection by predicting their motion and keeping the objects in memory, also when the detector can not detect them for several frames. The example in Figure 5.5 shows a case where the detector misses a bicycle in 3 consecutive frames and then detects it again. The PolyMOT-Tracker can track the object over the whole sequence of 5 frames with a consistent ID, like in the ground truth, while the SORT3D tracker can keep the track for another frame but then loses it and assigns the bicycle a new ID in the frame where it is detected again. Cases like these lead to many ID switches in the tracking results of the SORT3D tracker mentioned in Section 5.2.2.

Overall, the tracking system presented in this work clearly outperforms the existing SORT3D tracker when interpreting the quantitative numbers and the qualitative illustrations in the in-

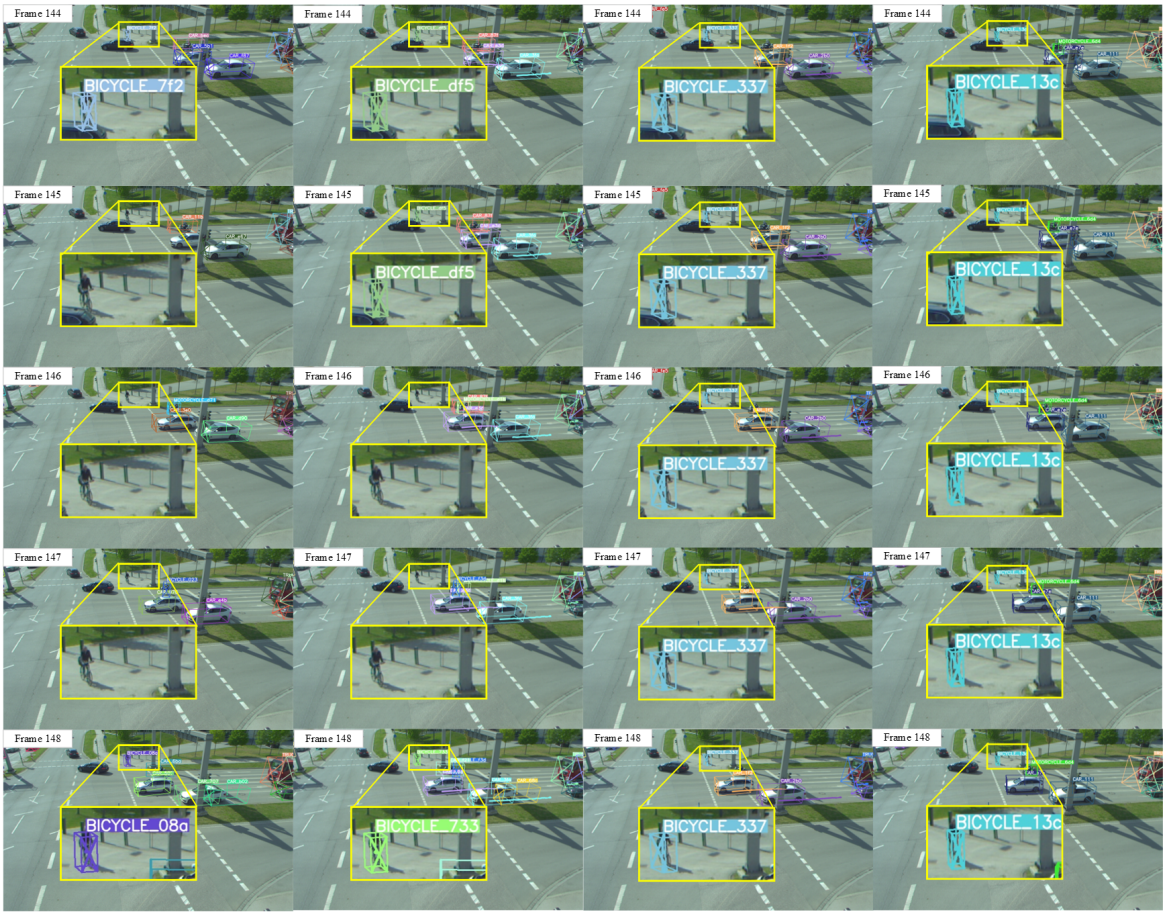


Figure 5.5: Presented is a sequence of five consecutive frames. The first column displays the detections, the second column showcases the tracked detections of SORT3D, the third column exhibits the tracked detections of PolyMOT and the fourth column are the ground truth objects.

tersection scenario. After examining the visualized tracking results, we searched the scenarios for cases where an object incorrectly changes its ID. The visualization of the untracked detections shows that the reason for most of these incorrect ID switches are noisy detections: either detections with wrongly predicted orientations or missing detections over several frames.

5.3 Highway Data

5.3.1 Data Description

The evaluation utilizes data from the TUM Traffic A9 highway dataset [7]; more specifically, it uses the $R0\_S02$ sequence of the dataset. This dataset captures real-world scenarios along a segment of the A9 highway near the exit of Garching Süd. The $R0\_S02$ sequence consists of 60 frames, each separated by a time interval of 0.4 seconds, resulting in a sampling frequency of 2.5 Hz. Although slightly slower compared to the data used in the intersection scenario, see Section 5.2, this sequence offers a rich source of information crucial for evaluating multi-object tracking algorithms. The highway section under consideration is equipped with LiDAR sensors and cameras, enabling the collection of point cloud and image data. However, for the $R0\_S02$ sequence, only image data is available. The dataset only contains the labels of the 3D object bounding boxes but does not include consistent track IDs across the sequences. The track IDs are necessary to evaluate the performance of the trackers. The tracker was executed on the ground truth detection boxes to generate the necessary consistent track IDs, thereby enabling the tracking performance evaluation. This process ensures that each object is assigned a unique identifier throughout the sequence, enabling an accurate assessment of the tracker's ability to maintain object identities over time. Without LiDAR point cloud data for the $R0\_S02$ sequence, we apply a monocular 3D detector to generate detections for the tracking evaluation. Despite the thesis' focus on LiDAR-based tracking, the monocular detector offers comparable geometric motion features, including object position, dimensions, and orientation. However, it still does not include appearance information about the objects gained from the images. As the data from the TUM Traffic A9 highway dataset originates from a highway, the orientation of the objects usually aligns with the direction of the lanes. Therefore, as an additional preprocessing step, the orientation of all detections is corrected to point in the right direction depending on which side of the highway the object is. Additionally, the initial velocity of the objects was not set to zero, as in the intersection scenario, but set to a fixed value, listed in Table 5.10. Modifications of the parameters are listed in Table 5.10. Also, we use N_{min_hit} of 1 and a second association threshold θ_{SA} of 1.5. The other parameters stay the same as in the intersection scenario. These modifications allow the tracker to produce improved tracking results. By leveraging the A9 highway dataset and using the aforementioned preprocessing technique, the evaluation aims to provide insights into the performance of the two trackers in highway scenarios.

	Bike	Bus	Car	M.Cycle	Ped.	Trailer	Truck	Van	Emerg.	Other
θ_{FA}	1.55	1.55	1.8	1.55	1.7	1.45	1.5	1.6	1.5	1.5
Velocity$[\frac{km}{h}]$	0	100	150	120	0	90	90	140	140	140

Table 5.10: Used hyperparameters for the intersection data used for the evaluation of the given sequences.

5.3.2 Quantitative Results

The following section considers the quantitative results of the PolyMOT-Tracker for the highway scenario. As before, we compute the metrics for the PolyMOT-Tracker output and the SORT3D tracker output. Again, the first subsection considers the basic metrics explained in Section 2.5.3 while the second one considers the CLEAR-MOT metrics explained in Section 2.5.4.

Basic Metrics

	#Track IDs	Total Track Length[m]	Total Track Length[frame]	Avg. Track Length[m]	Avg Track Length[frame]
SORT3D	1711	1845	**3676**	1.1	2.1
PolyMOT	**120**	**24357**	1748	**203.0**	**14.6**
GT	128	31335	2404	244.8	18.8

Table 5.11: Results Short Highway Sequence.

Table 5.11 shows the evaluation results of the basic metrics of the PolyMOT-Tracker and the SORT3D tracker compared to the ground truth. As mentioned, the goal is to have the calculated metrics as close as possible to the ground truth values. One can see that the results from the PolyMOT-Tracker are, again, pretty accurate compared to the ground truth. The number of track IDs differs only by eight IDs. Also, the average track length in meters and frames is in the same range as the ground truth values, indicating that the tracker also produces valid results on the highway. However, the SORT3D tracker is not able to produce valuable results. The very high number of track IDs of 1711 suggests that the SORT3D tracker can not track the detected objects correctly or at all. Additionally, the average track length in meters of 1.1 m and the average track length in frames of 2.1 frames shows that applying the SORT3D tracker in the highway scenario does not make sense. The cause for that is similar to the cause for the lousy tracking results on small objects of the SORT3D tracker mentioned in Section 5.2.2. Since the objects on the highway are typically faster than objects at an urban intersection, they put back more meters between two frames, especially in this concrete case, due to the lower frequency of frames. Because of the limited capability of the distance metric for the association used by the SORT3D tracker, i.e., the IoU, the SORT3D tracker is incapable of associating objects with an IoU of zero. This problem has been elucidated in Section 2.4. Because of this, the SORT3D tracker is not further considered in the following quantitative analysis since the results do not produce any valuable insights.

CLEAR-MOT Metrics

	FN ↓	FP ↓	MT ↑	PT ↑	ML ↓	IDSW ↓	FRAG ↓	HOTA↑	MOTA ↑	MOTP ↓
PolyMOT	1313	657	20	58	50	19	50	0.45	0.18	1.63

Table 5.12: Results Short Highway Sequence.

As stated in Section 5.3.2 in the following, only the CLEAR-MOT results for the PolyMOT-Tracker are enumerated and analyzed. The results in Table 5.12 show the calculated metrics. The first thing that stands out is that the results are generally worse than the evaluated results from the intersection scenario. However, we can not compare both results unconditionally

since different object detectors generated the detections used by the tracker to produce its results. Additionally, the same reason described in Section 5.3.2, that the objects' velocities are higher, influences the higher value of MOTP. We also perform the computation of the CLEAR-MOT metrics classwise to further analyze the reason for the difference in performance. The table 5.13 shows parts of the CLEAR-MOT results for every class separately.

	GT	**FN ↓**	**FP ↓**	**MOTA ↑**	**MOTP ↓**
BUS	8	6	15	-1.6	2.9
CAR	1641	591	339	0.43	1.60
PEDESTRIAN	0	0	37	-	-
TRUCK	179	140	266	-1.3	2.55
TRAILER	282	282	0	0	-
VAN	294	294	0	0	-

Table 5.13: Classwise Results Short Highway Sequence.

Only classes that occur at least once in the ground truth or the tracking results are listed. Also, the number of listed metrics is lower for simplification. One can see that the results for the car category, which is the most frequently present, are pretty good, with a MOTA of 0.43 and a MOTP of 1.60. However, the poorer performance in other categories drags down the good result for the car category. Many pedestrians are tracked, although there are no pedestrians in the ground truth. The reason for that is false positive detections of the used 3D detector. Furthermore, the PolyMOT-tracker does not track objects of the trailer and van classes at all in the considered sequence, leading to MOTAs of 0. The reason for this is missing detections for those objects. The computed MOTA values for the other two categories, bus and truck, are below -1. MOTA can be lower than zero due to how the MOTA metric calculation is performed, see Section 2.5.4. The reason for the bad results in the bus category could be the low number of occurrences of buses in the considered sequence. In the case of trucks, the problem is likely wrong detections. Since the MOTP can only be computed if a matched pair between a predicted object and a ground truth object exists, we cannot compute it for the mentioned classes. Concluding the quantitative analysis, we can say that the tracker also works well when applied to highway data, assuming a certain standard of detection quality.

5.3.3 Qualitative Results

As mentioned in Section 5.3.2, it does not make sense to consider the SORT3D tracker in the highway scenario. Therefore, we show the qualitative illustrations of the tracking results only for the PolyMOT-Tracker. Figure 5.6 shows the tracking results and the labels at one frame in the roadside view as an image and once in a BEV view. The visualization encourages the claim for the applicability of the PolyMOT-Tracker on highway data. On the image and the BEV plot, it is evident that the tracker returns smooth and long tracks for the objects. Also, lane changes, like for the object marked with the green square, are registered and smoothly tracked. Another point to emphasize is the nicely tracked curve taken by the car marked by the yellow square exiting the highway. The limited amount of ground truth tracks on the highway allows us to only give an impression of how well the tracker works on highway data. However, summarizing the qualitative analysis, the tracker also produces good results on the highway based on the illustration 5.6 and the preceding quantitative analysis in Section 5.3.2.

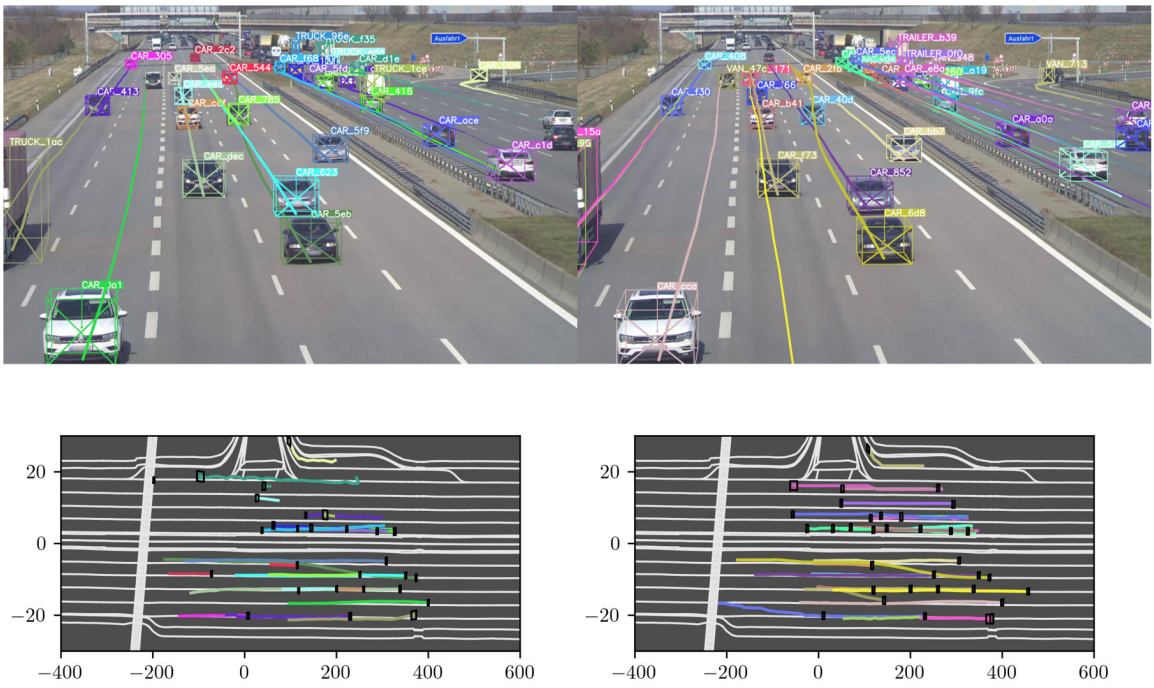


Figure 5.6: Visualized tracking results and labels at one frame of the highway data. The labels are visualized on the right while the tracked detections are visualized on the left. The roadside images are shown in the upper row, the time corresponding BEV plot is shown in the lower row.

5.4 Tracking Speed

All experiments were conducted on an i7-11370H Intel Processor. The inference speed of the tracking system varies depending on the scenario in which the tracker operates. In the intersection scenario, it achieves an average inference speed of around 30 FPS, whereas in the highway scenario, the speed is notably slower at around 8-10 FPS. This discrepancy is primarily attributed to the higher number of objects in a single frame in the highway scenario than in the intersection scenario. The primary computational bottleneck is attributed to the computation of the Generalized intersection over Union (GIoU), which involves the calculation of the convex hull in 3D, a highly computationally intensive task. We compute the GIoU at one frame between every pair of detections and the existing tracks in that frame. So, the computational intensity depends on the number of such pairs. Figure 5.7 plots the tracking speed for all four intersection sequences and two highway sequences. It concludes that the number of detections and tracks determines the overall tracking speed. More exactly, the inference speed decreases with the rising number of tracks and detections. The high number of outliers is probably due to the background process running on the system.

In contrast, the SORT3D tracker excels in speed, requiring only a fraction of the time for its operations. It achieves an impressive inference speed in the range of 700-800 FPS. Consequently, the SORT3D tracker holds a distinct advantage over the evaluated 3D multi-object tracking system regarding speed. On the highway scenario, the SORT3D tracker also maintains a competitive speed of around 200 FPS, but as mentioned in Section 5.3.2, it does not produce helpful tracking results.

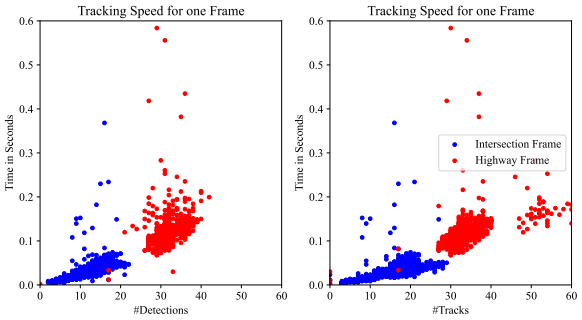


Figure 5.7: The tracking speed of the PolyMOT-Tracker as a function of the number of detection and and as a function of the number of tracks within a single frame. Each data point represents a unique frame. The figure visually depicts the correlation between tracking speed and the quantity of number of detections and number of tracks.

5.5 Hyperparameter Search

This section illustrates the elaborate task of the hyperparameter search. The resulting parameters are used in the final tracking algorithm. The search of the optimal parameters is based on the three metrics MOTA, MOTP and HOTA, described in Section 2.5.4. The rest of the parameters are kept constant during one evaluation, only the considered parameter is altered. More, exactly we use the same parameters as defined in Section 5.1 for the remaining parameters. The first two of the following three parameters are category-specific. Since describing the search for all categories, we only handle the car category since it occurs most often. The parameters used are the same as stated in Section 5.1 apart from the parameter that is varied in the single searches.

Maximal Age Parameter

As elucidated in Section 4.1.4, the max_age parameter defines how long the state of a tracked object is kept in memory after it has last been matched with a detection. Important to mention is the difference to the punish_num parameter: the latter one defines how long a tracked object will be output, while the max_age only defines for how many frames it will be considered in the data association. The outcomes of the core metrics upon applying the tracker to sequence $R02\_S03$ are depicted in Figure 5.8. It's noticeable that the metrics' values stabilize after $N_{max_age} = 9$. To accommodate occlusions lasting beyond 0.9 second, we establish the N_{max_age} threshold at 15. Figure 5.8 shows that extending this threshold beyond 15 frames is not improving the performance and at some point even decreases the performance with increasing MOTP and decreasing MOTA and HOTA. The increase in MOTP is a logical consequence since when matching a track with a new detection after a higher number of frames the position of the predicted position will get more inaccurate with every missing update step of the Kalman Filter, see Section 2.3.

Minimal Hit Parameter

The min_hit parameter determines the number of consecutive frames in which a tracked object has to be matched with a detection before it is written to the output file, see Section 4.1.4. The results of the primary metrics when the tracker is applied to the sequence $R02\_S03$ are depicted in Figure 5.9, with varying values of the N_{min_hit} parameter for the car category. Following observation, the MOTA value ceases to improve beyond $N_{min\_hit} = 3$, while the HOTA value experiences a slight decrease. Consequently, we set the N_{min_hit} parameter for the

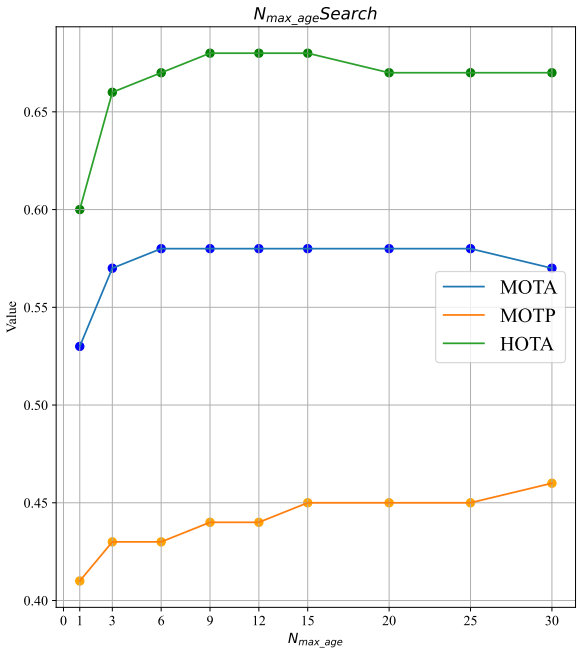


Figure 5.8: The figure shows the result of the MOTA, MOTP and HOTA metric when varying the N_{max_age} Parameter for the car category. One can see the results are not getting better after $N_{max\_age} = 15$. At some point MOTP and MOTA are even getting worse.

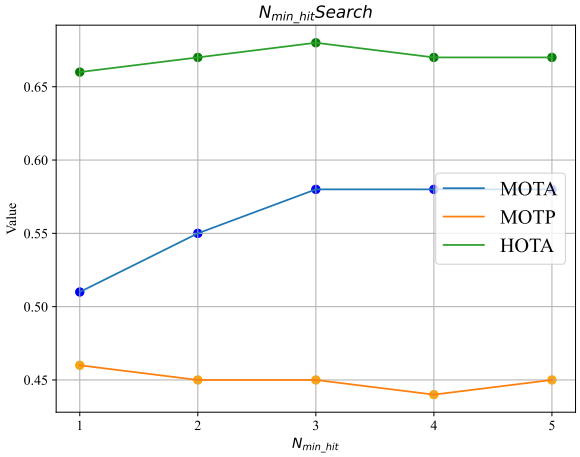


Figure 5.9: The figure shows the result of the MOTA, MOTP and HOTA metric when varying the N_{min_hit} Parameter for the car category. One can see the results are the best for $N_{min\_hit} = 3$

car category to 3. This decision is influenced by the increasing number of False Negatives with each increment in the N_{min_hit} parameter, as detections are only output after the fifth hit, even if they are True Positives from the first hit onward.

Angular Difference Penalty Parameter

The idea of the angular difference penalty is defined in Section 4.2.3. Here the search for the optimal λ parameter is illustrated in Figure 5.10. The parameter is the same for all categories. Figure 5.10 demonstrates that the optimal MOTA and HOTA scores are attained with an

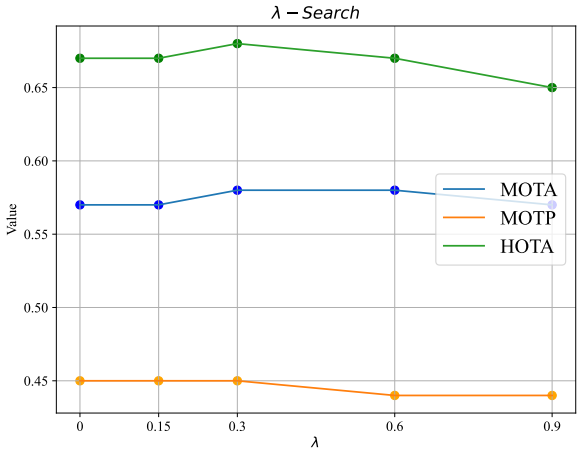


Figure 5.10: The figure shows the result of the MOTA, MOTP and HOTA metric when varying the general λ Parameter for the angular difference penalty computation 4.2.3. One can see the results are the best for $\lambda = 0.3$

angular difference penalty factor λ of 0.3. Larger factors elevate MOTP but simultaneously reduce MOTA and HOTA, as they render the matching of new detections with existing tracks more challenging. Hence, we set the parameter to $\lambda = 0.3$ for consistency.

The performance of the tracking system is significantly impacted by the variation of its parameters, making parameter optimization a crucial aspect of achieving optimal results. As such, conducting a comprehensive parameter search is essential. This search process entails exploring all necessary parameters across all categories utilized by the tracking system. The resulting optimized parameters are documented in Section 5.1.

5.6 Effect of Added Features

The tracking results of the PolyMOT-Tracker used in Section 5.2 and in Section 5.3 are produced using the added features elucidated in Section 4.2. The following two sections show these features' impact on the overall tracking performance using the $R2\_S03$ sequence as the sequence to track.

Quantitative Analysis

The tracking results with and without the added features are listed to show that the features have a quantitive advantage against the basic version of the tracker. First, we execute the tracker without the features. After that, the tracker's results using the orientation correction are captured. At last, the feature determining the association threshold dynamically, including the angular difference penalty, is applied. Table 5.2.2 compares the performance of both

results. Although the difference in the quantitive results of HOTA, MOTA, and MOTA is not

	FN ↓	FP ↓	IDSW ↓	FRAG ↓	HOTA↑	MOTA ↑	MOTP ↓
Without Features	4278	2292	22	**71**	0.67	0.57	0.45
Orientation Correct.	4248	2267	**21**	73	0.68	0.57	0.45
Dynamic Threshold	4253	2454	23	76	0.66	0.56	**0.44**
Angular Difference	4280	2198	23	76	0.67	0.58	0.45
Confidence Tentative	4214	2296	24	74	0.67	0.57	0.45
Combined Version	**4177**	**2179**	26	75	**0.68**	**0.58**	0.45
Improvement	101	113	4	4	0.01	0.01	0

that high, it clearly reduces the number of False Negatives and False Positives. The important metrics HOTA and MOTA are also improved. The small changes in the quantitative results are not surprising since the basic version of the tracking system already performs quite well. Most of the remaining failure cases can be traced back to wrong predictions by the detector. The added features, therefore, mainly aim to mitigate the influence of such predictions and solve single failure patterns observed during the evaluation.

Qualitative Analysis

Since the difference in the quantitative results is insignificant, the qualitative analysis shows single cutouts of the $R02\_S03$ sequence. These sections show problems in scenarios that the modified version can solve but the basic version cannot.

The scenario pictured in Figure 5.11 shows a van driving into the blind spot of the LiDAR sensor at the intersection. A blind spot in this context means a space where the LiDAR sensor cannot capture data. Therefore, objects located in that area are invisible to the sensor. The goal is to still be able to match the van to the correct ID when it leaves this spot again. The figures show that the basic version does not keep the ID, while the modified version accomplishes the goal precisely. The reason for this is the incorrectly predicted orientations of the van in the frames right before driving into the blind spot. These bad predictions lead to an incorrect motion prediction and failure in the data association. By trying to correct the orientation and loosen the association threshold in the motion direction of the van, the tracking system can correctly identify the van when it is first detected again.

Correct orientation predictions in the scenario pictured in Figure 5.12 also show possible consequences of wrong motion predictions and high association thresholds. The upper row shows the results from the tracker without the included features. The lower row shows them with the features applied. One can see the car's motion in the upper images is mispredicted starting at Frame 8. This results in the predicted position of the car in frame 14 not being very close to the object's actual position. Therefore, when detected again in frame 19, the detection can no longer be matched with the tracked object (there is no bounding box at all in frame 19 for the results without features because of the trajectory management module, which decided that the object will not be output at that frame). However, the version using the added features corrects the car's orientation correctly and, therefore, can produce a better motion prediction and, in the end, can associate the detection of the car in frame 19 with the initially tracked object. One can see in the lower images that the ID of the car in frame 8 is the same as the ID in frame 19.

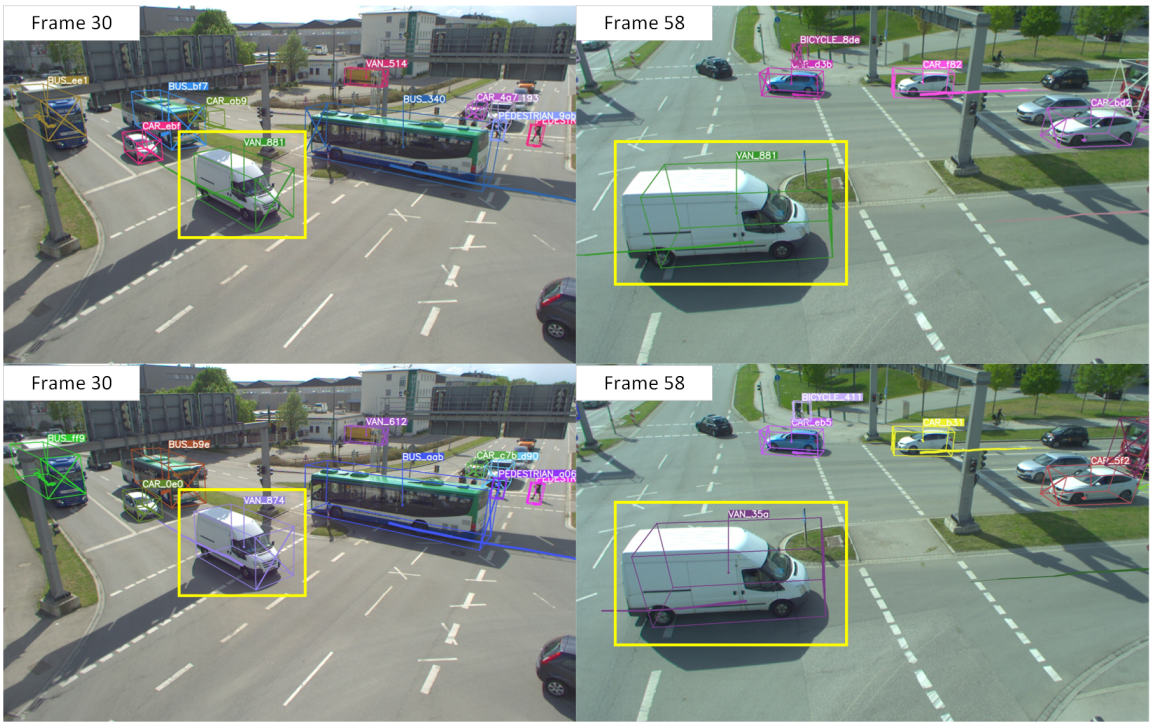


Figure 5.11: The images depict a scenario where a van traverses through the blind spot of the LiDAR sensor. The upper row illustrates the tracking results of the tracker executed with the features while the lower row shows the results of the basic PolyMOT-tracker. The basic version struggles to maintain track IDs, whereas the modified version demonstrates improved capability in preserving track continuity.

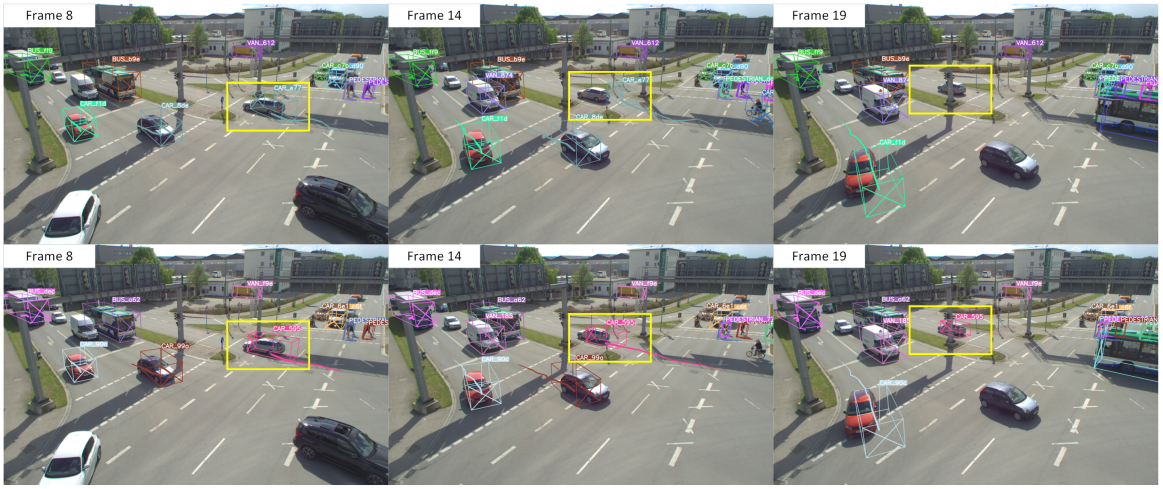


Figure 5.12: This scenario illustrates how incorrect predicted orientations of objects, coupled with missed detections, can result in ID switching. Videos of the four tracked sequences can be found here: $R02\_S01$ https://youtu.be/1PZsXaw3HXY, $R02\_S02$ https://youtu.be/_urCH_QfGCw, $R02\_S03$ https://youtu.be/c2NNmaYrcl4, $R02\_S04$ https://youtu.be/xvhGeZozXo0.

Chapter 6

Future Work

As we conclude the evaluation of the proposed 3D Multi-Object Tracking system and reflect on the presented findings, considering avenues for future enhancements and research directions is of the greatest importance. This chapter outlines potential enhancements, extensions, or novel investigations to improve the existing approach further.

6.1 Enhancements

Learning-based Modules

A way to improve the tracking performance of the tracker by pursuing the strategy proposed by [20] is to divide the tracking process into subtasks and consider each separately. An idea would be to replace existing non-learning-based modules with learning-based approaches. In this case, replacing the physical motion prediction models with trained models for trajectory prediction would be imaginable. Neural networks, primarily Recurrent Neural Networks or LSTMs, are widely used in trajectory prediction to capture complex motion patterns of objects from different categories. To use the idea introduced by [13], it also makes sense to have a separate motion model for different categories since a pedestrian carries out different motions than a car, for instance. Currently the motion of most categories are modelled using the CTRA model. The motion of bicycles and motorcycles is modelled using the BICYCLE model while the CA model models the motion of the trailer category. Another module to replace could be the data association. In general, Multi-Object Tracking research, as explained in Chapter 3 learning-based approaches like Graph Neural Networks or Transformer architectures, are an advanced method to compute the affinity between objects. Purely distance-based metrics show weaknesses in some cases since they do not consider the tracked object's past positions. Although these changes could improve the tracker's performance, they also influence the execution speed and hardware requirements for the execution. Neural networks generally require intensive computations and are usually executed on hardware accelerators like GPUs.

Bounding Box Refinement

As outlined before, in most cases, the tracking failure can be traced back to missing or flawed detections delivered by the detector. Therefore, to improve the tracker, it is helpful to try to improve the quality of detections. Missing detections cannot be recovered since that is the task of the detector. What is possible is to do a more advanced bounding box refinement as suggested in Section 4.2.2 to improve the quality of objects the detector detected. Especially in the concrete case of this project, because the tracking system has to perform well in a fixed environment, bounding box refinement is a promising feature. Bounding box refinement aims

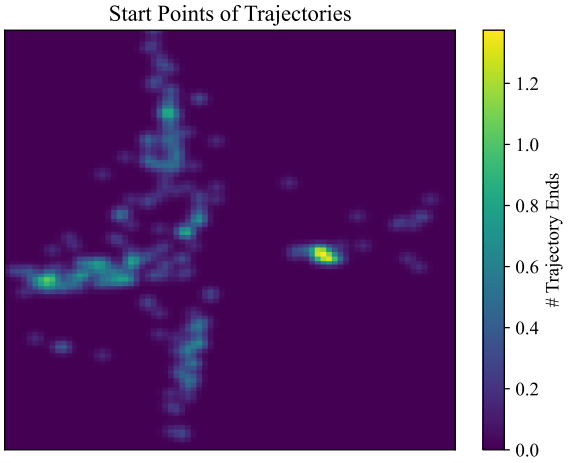


Figure 6.1: The plotted start points of trajectories in the ground truth.

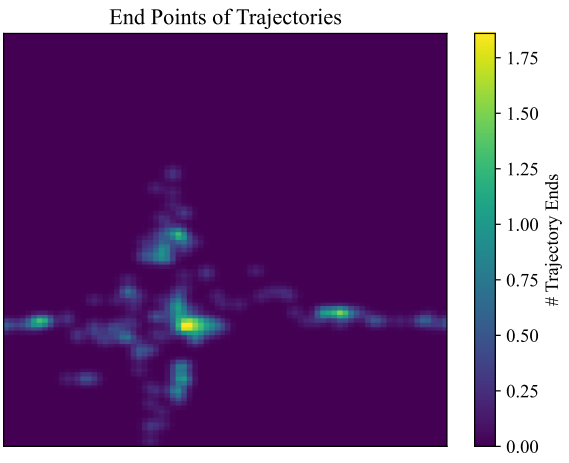


Figure 6.2: The plotted end points of trajectories in the ground truth.

to improve the accuracy of some of the bounding box's attributes. Feeding map information about the environment to the tracker is an idea to correct the orientation of an object to, e.g., point in the correct direction. Also, through training a model with data from the concrete environment the tracker is used in, we could create a deep-learning-based bounding box refinement feature.

Initial Velocity Estimation

As previously noted (see Section 5.3), employing an initial velocity estimation is essential for enhancing PolyMOT-Tracker's performance on highway data. However, relying on a fixed value for velocity estimation may introduce challenges, particularly in traffic jams, where velocities differ significantly from typical highway speeds. In such cases, the difference between the estimated and actual velocities can influence the tracking performance. Adopting a more sophisticated velocity estimation approach is warranted to address this issue. Two potential strategies could be applied. One approach involves dynamically adjusting the velocity estimation based on contextual factors, like the number of objects in the current frame. Given that the considered section is always the same, i.e., it always observes the same environment, a higher density of traffic participants might indicate a slower velocity of the single objects. Conversely, fewer traffic participants in the considered section might indicate a lower velocity of the objects. The second idea is to make the velocity estimation depending on the lane an object is currently moving in. Due to traffic regulations, vehicles typically gravitate toward specific lanes based on their velocity. Vehicles that want to overtake others have to do that by using the next lane on the left. Therefore, vehicles traveling on the left lanes can be assigned higher velocity estimates.

6.2 Extensions

Furthermore, the current approach can be extended by other promising techniques to improve the robustness and stability of the system. To accomplish this, e.g., future work could add a more advanced trajectory management module, taking knowledge about the environment of the scenario into account to decide whether trajectories should be newly initialized, declared as dead, or still kept in the memory. Usually, objects appear and disappear in the

same areas of the considered environment depending on the range that the LiDAR sensor can cover. The heatmaps 6.1 and 6.2 illustrate the distribution of geometric start and end points of trajectories derived from all ground truth data. Each trajectory's start and end points were individually plotted, and their densities were visualized. Brighter regions indicate higher concentrations of trajectory origins or destinations. However, an object finally disappearing in the middle of the intersection, e.g., is similarly as unlikely as one appearing in the middle, assuming no occlusions exist. In the highway scenario, tracks beginning in the middle of the observed section equally are less likely to represent the real beginning of a track for an object. So, including known map information in trajectory management can be a promising strategy. Another extension to the presented tracking system is to include appearance information in the tracking system. This appearance information can be acquired solely using LiDAR data. Also, the tracker could use appearance information resulting from image data to get additional information next to the already present motion features, resulting in multi-modal, multi-object tracking approaches.

While including map information in the tracking process might cause improvements, the algorithm would have to be adapted to each location where the tracking algorithm is used. Also, using multiple modalities comes with the same trade-off of deep-learning approaches as explained in Section 6.1.

Chapter 7

Conclusion

To enhance the capabilities of 3D multi-object tracking, this bachelor thesis incorporated a state-of-the-art 3D Multi-Object Tracking system into the already existing toolchain discussed in Chapter 1. The primary goal was to assess the performance of the chosen tracking system and conduct a detailed comparative analysis with its predecessor. The overall work involved thoroughly exploring the background knowledge required for understanding Multi-Object Tracking, including metrics for the evaluation and other relevant mathematical concepts. The literature review encompassed numerous tracking with diverse strategies. The approach from [13], identified as the top performer in the nuScenes tracking challenge with relatively low resource requirements, served as the baseline for this work. Initial evaluations identified and analyzed specific failure patterns and scenarios where the tracking system could not produce accurate results. Subsequently, the basic version was enhanced with additional features to improve its robustness and extend its capability to track objects in more complex scenarios. Features such as the correction of detection orientation and a dynamic determination of the association threshold in the data association stage aimed to improve the already high-performing tracker. Validation of the new tracker's superior performance compared to the existing SORT3D tracker was conducted in two distinct environments: a busy intersection and a highway. Highway objects usually have a higher velocity than objects at an urban intersection. Furthermore, there are usually more objects to track on a highway since the street is bigger. Therefore, it makes sense to differentiate between those two environments. After producing the tracking results for each scenario and computing the tracking metrics, it was possible to check if the new tracker outperformed the existing one. The quantitative results of the intersection scenario, see Section 5.2.2, conclusively demonstrate that the PolyMOT-Tracker outperforms the existing tracker, particularly for objects with relatively small dimensions. The qualitative comparison, see Section 5.2.3, revealed the PolyMOT-Tracker's ability to address challenges introduced by demanding scenarios, such as occluded objects or noisy detections, with whom the SORT3D tracker struggled. Considering the highway scenario, the quantitative analysis, see Section 5.3.2, showed the acceptable performance of the PolyMOT-Tracker. At the same time, we concluded the SORT3D tracker to be not applicable in the highway scenario due to the limitations of the distance metric used. The qualitative analysis, see Section 5.3.3, of the PolyMOT-Tracker encourages the mentioned quantitative results and shows some strengths of the tracker in the highway scenario. The added features were also found to contribute to correct tracking in these scenarios. However, it is crucial to note that the SORT3D tracker exhibits superior speed due to less computationally intensive mathematical computations compared to the PolyMOT-Tracker, i.e., the Generalized Intersection over Union.

In summary, the PolyMOT-Tracker delivers superior results to the SORT3D tracker: when tracking the most complex sequence $R02\_S03$ the PolyMOT-Tracker achieves a 16.2% higher HOTA value as well as a 13,8% higher MOTA value than the SORT3D tracker. Nevertheless,

the overall performance of a tracking system realizing the tracking-by-detection paradigm heavily relies on the quality of the available object detections. This work successfully mitigated the influence of poor detections to make the tracking more robust, which succeeded in some scenarios. However, the chosen tracker still needs to be optimal, opening certain aspects to be improved in future work. E.g., incorporating map information into the trajectory management module, replacing physical motion models with learning-based approaches, or adding appearance features of the objects to the affinity computation of detections and existing tracks.

Bibliography

[1] Bernardin, K. and Stiefelhagen, R. “Evaluating multiple object tracking performance: the clear mot metrics”. In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[2] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. “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*. Springer. 2016, pp. 850–865.

[3] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. “Simple online and realtime tracking”. In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[4] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[5] Chen, X., Shi, S., Zhang, C., Zhu, B., Wang, Q., Cheung, K. C., See, S., and Li, H. “TrajectoryFormer: 3D Object Tracking Transformer with Predictive Trajectory Hypotheses”. In: *arXiv preprint arXiv:2306.05888* (2023).

[6] Chiu, H.-k., Wang, C.-Y., Chen, M.-H., and Smith, S. F. “Probabilistic 3D Multi-Object Cooperative Tracking for Autonomous Driving via Differentiable Multi-Sensor Kalman Filter”. In: *arXiv preprint arXiv:2309.14655* (2023).

[7] Creß, C., Zimmer, W., Strand, L., Fortkord, M., Dai, S., Lakshminarasimhan, V., and Knoll, A. “A9-dataset: Multi-sensor infrastructure-based dataset for mobility research”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, pp. 965–970.

[8] Ding, S., Rehder, E., Schneider, L., Cordts, M., and Gall, J. “3dmotformer: Graph transformer for online 3d multi-object tracking”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 9784–9794.

[9] Kim, A., Brasó, G., Ošep, A., and Leal-Taixé, L. “PolarMOT: How far can geometric relations take us in 3D multi-object tracking?” In: *European Conference on Computer Vision*. Springer. 2022, pp. 41–58.

[10] Koh, J., Kim, J., Yoo, J. H., Kim, Y., Kum, D., and Choi, J. W. “Joint 3d object detection and tracking using spatio-temporal representation of camera image and lidar point clouds”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 1. 2022, pp. 1210–1218.

[11] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. “Pointpillars: Fast encoders for object detection from point clouds”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12697–12705.

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

[13] Li, X., Xie, T., Liu, D., Gao, J., Dai, K., Jiang, Z., Zhao, L., and Wang, K. "Poly-mot: A polyhedral framework for 3d multi-object tracking". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 9391–9398.

[14] Liu, J., Bai, L., Xia, Y., Huang, T., Zhu, B., and Han, Q.-L. "GNN-PMB: A simple but effective online 3D multi-object tracker without bells and whistles". In: *IEEE Transactions on Intelligent Vehicles* 8.2 (2022), pp. 1176–1189.

[15] Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., and Leibe, B. "Hota: A higher order metric for evaluating multi-object tracking". In: *International journal of computer vision* 129 (2021), pp. 548–578.

[16] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., and Kim, T.-K. "Multiple object tracking: A literature review". In: *Artificial intelligence* 293 (2021), p. 103448.

[17] Mitzel, D. and Leibe, B. "Taking mobile multi-object tracking to the next level: People, unknown objects, and carried items". In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*. Springer. 2012, pp. 566–579.

[18] *OpenLABEL Format for Object Detection*. https://www.asam.net/standards/detail/openlabel/. Accessed: February 12 2024.

[19] Pang, Z., Li, J., Tokmakov, P., Chen, D., Zagoruyko, S., and Wang, Y.-X. "Standing Between Past and Future: Spatio-Temporal Modeling for Multi-Camera 3D Multi-Object Tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17928–17938.

[20] Pang, Z., Li, Z., and Wang, N. "Simpletrack: Understanding and rethinking 3d multi-object tracking". In: *European Conference on Computer Vision*. Springer. 2022, pp. 680–696.

[21] Polack, P., Altché, F., d'Andréa-Novel, B., and La Fortelle, A. de. "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In: *2017 IEEE intelligent vehicles symposium (IV)*. IEEE. 2017, pp. 812–818.

[22] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. "Generalized intersection over union: A metric and a loss for bounding box regression". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 658–666.

[23] Ribeiro, M. I. "Kalman and extended kalman filters: Concept, derivation and properties". In: *Institute for Systems and Robotics* 43.46 (2004), pp. 3736–3741.

[24] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2446–2454.

[25] Tao, L., Watanabe, Y., Yamada, S., and Takada, H. "Comparative evaluation of Kalman filters and motion models in vehicular state estimation and path prediction". In: *The Journal of Navigation* 74.5 (2021), pp. 1142–1160.

[26] Thomsen, F., Ortmann, M., and Eckstein, L. "Scalable Real-Time Multi Object Tracking for Automated Driving Using Intelligent Infrastructure". In: *2023 International Conference on Electrical, Computer and Energy Technologies (ICECET)*. 2023, pp. 1–6. DOI: 10.1109/ICECET58911.2023.10389193.

[27] Touska, D., Gkountakos, K., Tsikrika, T., Ioannidis, K., Vrochidis, S., and Kompatsiaris, I. "Graph-Based Data Association in Multiple Object Tracking: A Survey". In: *International Conference on Multimedia Modeling*. Springer. 2023, pp. 386–398.

[28] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[29] Wang, L., Zhang, X., Qin, W., Li, X., Gao, J., Yang, L., Li, Z., Li, J., Zhu, L., Wang, H., et al. "Camo-mot: Combined appearance-motion optimization for 3d multi-object tracking with camera-lidar fusion". In: *IEEE Transactions on Intelligent Transportation Systems* (2023).

[30] Weng, X., Wang, J., Held, D., and Kitani, K. "3d multi-object tracking: A baseline and new evaluation metrics". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10359–10366.

[31] Willes, J., Reading, C., and Waslander, S. L. "Intertrack: Interaction transformer for 3d multi-object tracking". In: *2023 20th Conference on Robots and Vision (CRV)*. IEEE. 2023, pp. 73–80.

[32] Wu, H., Han, W., Wen, C., Li, X., and Wang, C. "3D Multi-Object Tracking in Point Clouds Based on Prediction Confidence-Guided Data Association". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 5668–5677. DOI: 10.1109/TITS.2021.3055616.

[33] Xu, R., Xia, X., Li, J., Li, H., Zhang, S., Tu, Z., Meng, Z., Xiang, H., Dong, X., Song, R., et al. "V2v4real: A real-world large-scale dataset for vehicle-to-vehicle cooperative perception". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13712–13722.

[34] Yi, W., Morelande, M. R., Kong, L., and Yang, J. "An efficient multi-frame track-before-detect algorithm for multi-target tracking". In: *IEEE Journal of Selected Topics in Signal Processing* 7.3 (2013), pp. 421–434.

[35] Yin, T., Zhou, X., and Krahenbuhl, P. "Center-based 3d object detection and tracking". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 11784–11793.

[36] Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. "Graph transformer networks". In: *Advances in neural information processing systems* 32 (2019).

[37] Zhang, J., Xiao, W., and Mills, J. P. "Optimizing moving object trajectories from roadside Lidar data by joint detection and tracking". In: *Remote Sensing* 14.9 (2022), p. 2124.

[38] Zhang, Y., Wang, X., Ye, X., Zhang, W., Lu, J., Tan, X., Ding, E., Sun, P., and Wang, J. "ByteTrackV2: 2D and 3D Multi-Object Tracking by Associating Every Detection Box". In: *arXiv preprint arXiv:2303.15334* (2023).

[39] Zimmer, W., Creß, C., Nguyen, H. T., and Knoll, A. C. "A9 Intersection Dataset: All You Need for Urban 3D Camera-LiDAR Roadside Perception". In: *arXiv preprint arXiv:2306.09266* (2023).