University of Duisburg-Essen
Chair of Dynamics and Control
Univ.-Prof. Dr.-Ing. Dirk Söffker

Project work

Improvement of object detection candidate certainty using redundant tracklets and situational information

Mohammed Albhaisi

Advisor

Waldemar Boschmann, M.Sc.
Univ.-Prof. Dr.-Ing. Dirk Söffker

April 2023

Erklärung

Ich erkläre, dass die Arbeit bis auf die offizielle Betreuung durch den Aufgabensteller selbstständig verfasst wurde. Die verwendeten Quellen sowie die verwendeten Hilfsmittel sind vollständig angegeben. Wörtlich übernommene Textteile und übernommene Bilder und Zeichnungen sind in jedem Fall kenntlich gemacht.

Ort, Datum

Unterschrift

Urheberrechtsvereinbarung

Die vorliegende Arbeit entstand unter intensiver Mitwirkung der genannten, betreuenden Personen im Rahmen von Forschungsarbeiten im Lehrstuhl Steuerung, Regelung und Systemdynamik (SRS) der Universität Duisburg-Essen. Die in dieser Arbeit enthaltenen Lösungen und Ideen unterliegen daher nicht nur dem Urheberrecht der zu qualifizierenden Person, sondern dem genannten Personenkreis gemeinsam. Die persönliche, private sowie die Nutzung im Forschungskontext des Lehrstuhls SRS ist hiervon unbenommen.

Ort, Datum

Unterschrift

Contents

List of Figures

List of Tables

1 Introduction

The improvement of object detection has become one of the most important topics for research. Different methods are used for this task, but a recent method will be introduced in this master thesis. The main goal of this project is to increase the certainty of detection approaches, in order to increase the safety of autonomous vehicles.

The thesis is under the name "Improvement of object detection candidate certainty using redundant tracklets and situational information", which aims to improve results for different detection approaches, which is done by providing redundant tracklets. The first part of the project deals with object detection methods, while the second part explains the tracking algorithm process, and how it can aid to improve the object detection results.

The autonomous car needs to locate itself in the environment as well as detect and keep track of objects in the surrounding environment to avoid collision. Many methods aim to improve the safety of autonomous vehicles. Redundant systems are a recent method within this range that is essential to increasing safety and reliability for autonomous vehicles. Because this work is always related to reliability, which is usually established by redundancy, it is important to first define and understand what the meaning of redundancy is, how redundant systems can improve overall system safety in autonomous vehicles, and how detection and tracking can work together.

Redundant tracklets can be introduced in detection systems to improve their performance. The meaning of redundancy is the fact that multiple detection systems are used at the same time to detect a particular instance, and then the tracker is applied to the results of the different detection systems individually, which means that for each detection system, there is a set of tracklets. At this point, it can be said that the tracking information has been obtained and the results are ready for confusion. This method can significantly improve the reliability of the system. Based on the last concept, object detection and tracking will be introduced. It is also important to make some comparisons between the results in order to see which ones give a better performance. However, in the result comparison, it will be focused on all categories of objects, including pedestrians and cars.

This work follows the tracking-by-detection paradigm. Therefore, it is necessary to explain the process behind this paradigm. The most successful tracking methods at present follow the tracking-by-detection paradigm [Imm22]. The paradigm is done as follows: First, bounding boxes of objects will be obtained in every frame, and then the detected objects between frames will be associated as tracklets. However, for the first part of object detection, this work utilizes four different detection approaches based on the lidar system. To explain why it is needed to use different detection approaches for the same instance, the answer to this question was introduced in [BS22] as follows: Because using

a combination of diverse approaches can compensate for each other's drawbacks and lead to better and more robust predictions, improving the reliability of predictions and final decisions. Furthermore, in cases of redundant systems, the matching processing is considered so more information can be given based on agreement or disagreement in the available multiple predictions for an instance.

After the results for object detection are obtained, the next part is to track the detected objects using Immortal Tracker, which gives very good performance for object tracking. It is necessary to understand why this method is considered one of the best methods at present time and how it can successfully aid in solving The most important problem appears in object tracking, which is called identity switching. This will be discussed in this thesis.

1.1 Motivation

Robust detection and tracking of objects are critical for the development of autonomous vehicles to work safely and avoid accidents. So, it makes sense to keep improving the detection and tracking performance continuously. Self-driving cars have the potential to provide a number of benefits to society, such as accident prevention, optimal fuel usage, comfort, and convenience [Pet09]. Improving safety is done based on collision avoidance, which is often based on multiple systems detecting the surroundings. This project works on developing such systems that can improve the detection and avoidance of collisions based on lidar data.

Different object detection approaches are used to detect different classes. All detected objects will be tracked using the Immortal Tracker algorithm. The idea of joint detection and tracking is to increase safety and precision. This project focuses on object detection and tracking and how different approaches can work together, aiding each other in order to improve performance. To make a general overview of all stages that are followed in this project, see figure 1.1.

- At the first stage, the detection results will be obtained using different object detection approaches.
- At the second stage, all detected objects will be tracked using Immortal Tracker.
- At the third stage, all detection and tracking results will be evaluated using different metrics.
- At the final stage, based on evaluation results and which result is better than another, the detection approaches results can be improved.

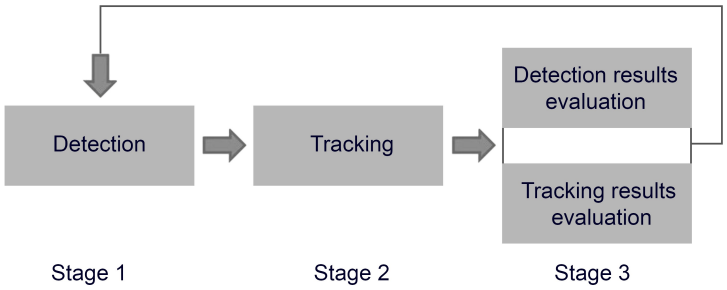

Figure 1.1: Overview of the project diagram

Overall, redundant tracklets are a powerful method for improving the accuracy and reliability of detection systems. The motivation behind using redundant tracklets to improve detection systems is to increase their reliability based on the additional information obtained from the redundant tracklets.

1.2 Problem description

Autonomous vehicles face many challenges, and there is an increasing number of accidents due to incorrect decision-making during autonomous or assisted operation. This project will design an improved object detection and tracking system. This improved detection certainty can lead to better decision-making and, therefore, accident avoidance. This project shows how object detection candidate certainty can be improved using individual tracklets. The project design is done as follows: Four detection approaches based on pointpillar and centerpoint with different voxels and pillars are trained over different cross-validations, resulting in a total of 64 models. The tool used for this task is MMDetection3D with the nuScenes dataset.

Three different splits are used: Leave-One-Out cross-validation (5 folds, 5 models), exhaustive 5 cross-validation (5 folds, 10 models), and the original split from nuScenes (1 model). After that, the detected objects are tracked using the Immortal Tracker algorithm. The detection results will be evaluated using different metrics. Then redundant tracklets will be obtained. These tracklets will also be evaluated using different metrics. Then it is important to start comparing the results and make comparisons to see which results give better performance than the others. From the object detection and tracking evaluation, a wide range of options will be obtained. Then, by using confusion between the results, the detection approaches can be improved.

To further clarify the method and how object detection can be improved, take the following example: Consider the confidence score to be a quantification of the certainty, which is given as information about certain detection approaches. Now assume that there is an instance, which should be observed over a timeframe to see how certain this one situation is figure 1.2.

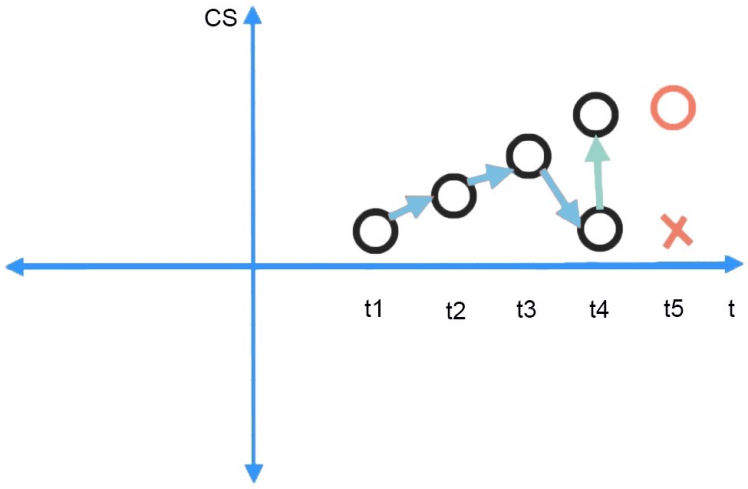

Figure 1.2: Object detection improving senario

At first timeframe, when this instance is detected, it will have a low confidence score because there is no information about it yet. At the next timeframe, when the same object is detected another time, the confidence score will increase based on the comparison between the different things like translation, scale, orientation, velocity, and attribute in the current frame and the last frames. In other words, more information will be obtained about the object, so the degree of certainty can increase. After some frames, it can be noted that the same object has been detected many times on the row, so the certainty will increase and there will never be uncertainty.

On the other hand, there is the tracking algorithm. In this task, association plays an important role. So, all that have been detected at different timeframes for the same object can be associated with each other. Now the important question arises itself, what will happen for the certainty, in case this object can not be seen in some frames due to being occluded like at frame t5, or uncertainty prediction is gotten that this object should be there at t4? It can be observed from the last frames that the object should be there, and now there is an uncertainty prediction about that. So, it is confirmed that this object should be there, then the certainty about that will be maintained and become higher again. At some points it will give certain like a good track on this object, then

if something wrong happens, it will be maintained at once using redundant tracklets. Finally, if there is an object that should be tracked over 40 frames, it can be assumed that the object should be tracked over all 40 frames.

Using the previous information and methods, wide options are obtained to merge the detection results over a timeframe or confidence score, and then to apply some cross-validation. From this process, individual trackers will be obtained on each line of detection. These redundant tracklets can be used later to improve the object detection results.

1.3 Thesis outline

In chapter 1, introduction, motivation, and problem description were presented. This chapter makes it clear why improvements to object detection are needed and the method that is used for this task.

In chapter 2, important concepts and methods that are used throughout the project are introduced under the name theoretical background. This is required in order to help the readers gain ideas and knowledge on all topics considered. The project is divided into detection and tracking tasks. Both of them will be included in the theoretical background. Firstly, object detection contains explanations for the detection methods used and important definitions used in the detection task. Secondly, object tracking contains important definitions for understanding the tracking mechanisms and the tool used for the task. Thirdly, the cross-validation methods, which are used in data split tasks, are introduced. Finally, the evaluation metrics for detection as well as tracking, which are available on the nuScenes dataset, will be explained in detail, including why every metric is needed for evaluation.

Chapter 3 explains the evaluation strategy. Firstly, the used dataset is introduced at the beginning. After that, the toolbox used for data training is presented. At the end, the experimental setup will be described, along with how the experiment was done, and the steps of the project and the preprocessing for evaluation will be summarized.

In chapter 4, the tracking and detection results will be discussed to see which result is better than the others. In the detection results, all splits for the detection results are evaluated using different metrics. On the other hand, tracking results, which will be divided into three parts, The first part will include an overall evaluation of all scenes from the original splits. Second part of the evaluation for every scene and see scenes with good results and scenes with bad results, and most identity switches. In the third section, some examples will be discussed, as well as the behavior of some instances.

Finally, chapter 5 contains a summary of this thesis.

2 Theoretical background

2.1 Object detection

Object detection is the field of computer vision that deals with the localization and classification of objects of interest in an image or video. It is an important component of many applications, including autonomous driving, surveillance, and robotics. In this section, the state-of-the-art approaches to object detection and object prediction will be presented.

2.1.1 State of the art

State-of-the-art object detection is the task of detecting and localizing objects within a video or image. The objective of object detection is to accurately locate the object in the image and classify it into a specific category. State-of-the-art object detection methods typically use deep learning techniques, such as convolutional neural networks (CNNs), in order to classify region proposals for object detection. It decomposes the detection problem into several stages, including bounding-box proposal, CNN pre-training, CNN fine tuning, SVM training, and bounding box regression [cao20].

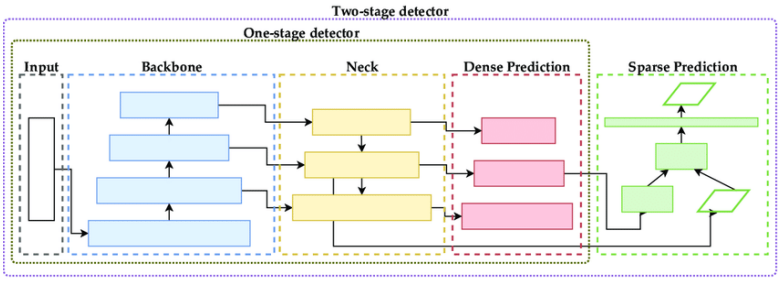

Figure 2.1: One-stage and two-stage object detection algorithm [Koay21]

The popular two approaches for object detection are one-stage and two-stage 2.1. One-stage methods contain a single feed-forward fully convolutional network that aims to directly predict bounding boxes and class probabilities for each object in a single shot. The Single Shot MultiBox Detector (SSD) and YOLO (You Only Look Once) were among the first to propose a single unified architecture, without requiring a per-proposal computation [gar20].

While two-stage frameworks divide the detection process into the region proposal and the classification stage. These methods first generate object proposals and then refine

them with classification and localization networks. In the other words, these models first propose several object candidates, known as regions of interest (RoI), using reference boxes (anchors). In the second step, the proposals are classified and their localization is refined [gar20]. The models are typically evaluated according to a mean-average precision metric.

The nuScenes dataset has been used extensively in the development of state-of-the-art lidar-based 3D object detection algorithms for autonomous driving. Here are some of the top-performing methods:

- Pointpillar is a popular 3D object detection method that uses a sparse voxel representation of lidar point clouds.
- SECOND is another popular 3D object detection method that uses a two-stage network to generate object proposals and refine object detections.
- Centerpoint is a single-stage 3D object detection method that uses a point-based approach to detect objects in lidar point clouds. It uses a standard lidar-based backbone network, for example Voxel-Net or PointPillars, to build a representation of the input point-cloud. It then flattens this representation into an overhead map view and uses a standard image-based keypoint detector to find object centers [cent20].
- PV-RCNN is a recently proposed 3D object detection method that combines a voxel-based feature encoder with a point-based feature encoder to achieve state-of-the-art performance.

Finally, the current state-of-the-art on nuScenes is BIRANet (RGB+Radar). It is a deep learning-based multimodal fusion approach for 3D object detection in autonomous driving scenarios [riax20].

In this thesis just pointpillar and centerpoint methods which are used in detection tasks.

2.1.2 Detection achitectures

This project utilizes detection systems implemented by four detection systems based on lidar sensors. Used detection architectures are pointpillar and centerpoint, using different pillar and voxel sizes. Pointpillar and CenterPoint are both state-of-the-art object detection algorithms for autonomous driving [Shi22]. They are both based on the idea of processing point cloud data captured by lidar sensors to identify objects in the environment. Both pointpillar and centerpoint are popular choices for object detection in autonomous driving applications, and they have their own strengths and weaknesses.

The choice between the two algorithms ultimately depends on the specific requirements of the application, such as speed, accuracy, and robustness. Some comparisons between pointpillar and centerpoint will be discussed.

Pointpillar is a 2D CNN-based approach that uses a voxel grid to discretize the point cloud data and then applies a CNN to generate feature maps. These feature maps are used to detect objects in the point cloud, and their 3D location is estimated using a regression network [Shi22]. Pointpillar is known for its high accuracy and speed, making it a popular choice for real-time object detection in autonomous driving scenarios [Lang19].

On the other hand, centerpoint is a 3D object detection algorithm that directly operates on the raw point cloud data without voxelization. It employs a multi-stage pipeline that first generates a set of 3D proposals, followed by a refinement stage to improve the accuracy of the proposals, and finally a classification stage to assign object labels to the proposals. CenterPoint is known for its robustness and high accuracy, and it has achieved state-of-the-art results on several benchmark datasets [cent20]. The center-based representation has many advantages: First, unlike bounding boxes, points have no intrinsic orientation. This dramatically reduces the object detector's search space while allowing the backbone to learn the rotational invariance of objects and the rotational equivariance of their relative rotation. Second, a center-based representation simplifies downstream tasks such as tracking. If objects are points, tracklets are paths in space and time. CenterPoint predicts the relative offset (velocity) of objects between consecutive frames, which are then linked up greedily. Thirdly, point-based feature extraction enables us to design an effective two-stage refinement module that is much faster than previous approaches [cent20].

2.1.3 Object predection

Object prediction refers to the task of predicting the category, presence and category of objects in a video or image. This is usually performed through the use of machine learning models like convolutional neural networks (CNNs), to identify objects within an image or video frame [cne20]. These algorithms produce a bounding box around each detected object, which can be used to determine its location within the image or frame.

In object prediction, the input of the model will be an image or a video frame, and it will produce a list of objects present in the image or frame along with their corresponding class labels and bounding box coordinates.

Once objects have been detected within an image or video frame, object prediction algorithms can be used to forecast their future locations and trajectories. These algorithms

use techniques such as Kalman filtering or particle filtering to estimate the future state of each object based on its previous positions and velocities.

Recent methods have used the Kalman filter in object prediction applications because it is able to account for uncertainty in the measurements and the underlying system dynamics. This makes it possible to produce more accurate and robust predictions even in the presence of noise and other sources of error. Uncertainty is an important factor that should be considered when addressing this problem. Uncertainty is a result of partial knowledge about the environment in which a robot moves and/or inaccurate information supplied by sensors. This problem is more evident in time-varying environments. [el01]. The system in object prediction using the Kalman filter is modeled as a state-space model, where the state is a vector containing the position and velocity of the object. The state estimate is updated using the measurements, which are obtained using sensors such as lidars. The filter then produces an estimate of the object's state at each time step along with an associated covariance matrix that describes the uncertainty in the estimate.

Object prediction is a challenging task due to the variability in object appearance, lighting conditions, and background clutter [neu13]. However, recent advances in deep learning have led to significant improvements in object prediction performance, making it possible to achieve near-human-level accuracy on certain datasets.

In this project, different predictions for multiple object classes are provided from different detection approaches.

2.2 Object tracking

2.2.1 Multi object tracking

The definition of multi-object tracking (MOT) is important in order to understand the tracking metrics that are defined in this thesis. MOT is the task of detecting the presence of multiple objects in video and associating these detections over time according to object identities [hota22]. The MOT task is one of the key pillars of computer vision research and is essential for many scene understanding tasks such as surveillance, robotics, or self-driving vehicles. It aims at estimating bounding boxes and identities of objects in videos [hota22]. An initially unknown number of targets from a known set of classes must be tracked as bounding boxes in a video in the MOT task. MOT has existed for a long time as a computer vision problem in which the goal is to keep track of the identities and locations of multiple objects throughout a video due to the complexity of the tracking task and the limitations of current computer vision algorithms [mot23].

In other words, MOT is the procedure of locating and following different objects in a video or image sequence over timeframe. The main goal of MOT is to correctly associate the same object across different frames, even if one object has been occluded or there is variation in appearance or change in the surrounding environment. It is one of the core tasks for understanding objects in video. The input to the MOT task is a single continuous video, split up into discrete frames at a certain frame rate. Each discrete frame could be an image or another representation such as a point cloud from a laser scanner.

The output of MOT is as follows: detection, association, and localization. The MOT task is a representation that encodes the information about what objects are present in each frame, which is called detection; where they are in each frame, which is called localization; and whether objects in different frames belong to the same or different objects, which is called association [hota22].

There are several approaches to multi-object tracking, including tracking-by-detection. It is a common MOT approach, in which an object detector is first run on every frame, and those detections are fed as input to a MOT algorithm [mot23].This approach will be defined in the next section.

Overall, MOT can be used to improve the accuracy of object detection by providing additional information about the movement and location of multiple objects to reduce missed detections. On the other words, it can improve the accuracy of object detection by using information about the motion and location of multiple objects to filter out false positives and reduce missed detections. By using information from multiple objects, MOT algorithms can also better differentiate between different classes and reduce confusion between identical classes.

2.2.2 Tracking by detection

In AVs, it is important to detect a different number of objects in the surrounding environment and keep tracking them in complicated scenes. This is a very challenging problem since there are many sources of uncertainty for the object locations, e.g., measurement noise, clutter, changing background, and significant occlusions [Bre09]. In order to deal with those problems, tracking by detection has become increasingly common for multi object tracking, which aims to acquire the moving trajectories of objects of interest in video or images [Rou17].

The process of this approach is as follows: In each frame, this approach involves first detecting objects using object detection algorithms and then tracking them across different frames [Bre09]. The process involves using an object detector to identify the location of

the object in each frame and then using this information to estimate the motion of objects. Furthermore, the tracking is done by matching object detections from one frame to the next one using different techniques such as the Hungarian algorithm and Kalman filtering.

Combining detection with tracking is a useful solution for solving the majority of problems that appear in AVS. In the tracking process, errors can accumulate, but a detector can be used to localize the object being tracked and re-initialize the tracker. It is evident that detection accuracy is essential, so a high decision threshold is set for the detector [Rou17].

In this project, the tracking by detection techniques that have been used are the Hungarian algorithm with Mahalanobis distance as the cost function and the Kalman filter.

2.2.3 Immortal Tracker

In this work, the tracking results are obtained using the Immortal Tracker 3DMOT Object Tracker method. One of the best performances on the nuScenes benchmark. This method achieves a mismatch ratio at the 0.0001 level, which is considered many times lower than any previously used method. ImmortalTracker was developed from a previous tracker called SimpleTrack. Some 3D MOT proposals propose to initialize and terminate tracks depending on their confidence score estimated from the confidence of their associated detections. However, they will still permanently terminate the tracklets that fail to be associated with new detections. In contrast, this method shows that through positively predicting and preserving the trajectories of objects, the tracklets that lost their targets can be properly maintained for possible association in the future [Imm22].

The main problem with the SimpleTrack method and other 3DMOT methods was that those methods terminated a tracklet prematurely for an object when it disappeared from the scene for a few frames, and it was considered to have left the scene. But in case this object appears and is detected another time in the scene, a new tracklet will be initialized, which will cause a problem called an identity switch. To address this problem, the developers have presented Immortal Tracker, a simple tracking framework based on trajectory prediction for invisible objects. Ideally, even if an object goes dark, like being occluded by other objects or obstacles for a few frames, a tracklet should be maintained till its target goes out of the scene and avoid premature termination. Therefore, Immortal Tracker is needed for this task and to maintain the identity switch problem. This method uses the Kalman filter for trajectory prediction to preserve the tracklet by prediction when an object is invisible. So instead of termination, the tracklet is maintained on its predicted trajectory [Imm22].

For example, see figure 2.2, one comparison between the Immortal Tracker and another tracker called CenterPoint++, used to track a vehicle over frames from 98 to 114. It can

be seen that the vehicle is not observed in frames 100–112. While CenterPoint++ failed to preserve the tracker as the object was not detected there and as this object appeared again in frames 112-114, a new tracker was initialized, which led to an identity switch, but Immortal Tracker succeeded in preserving the tracker until the object reappeared in the future with the same identity [Imm22].

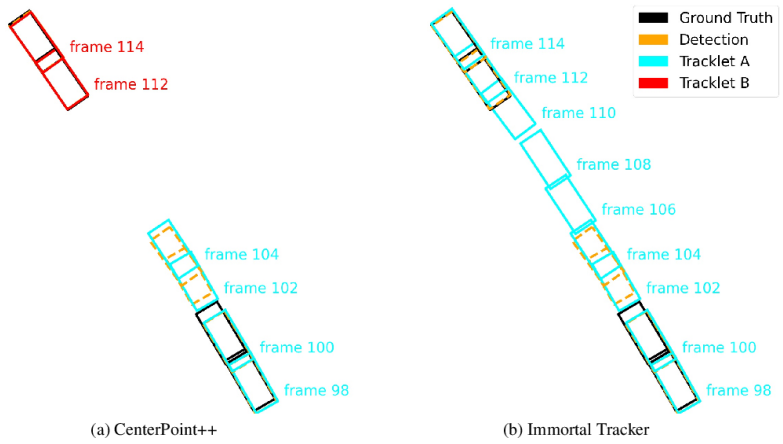

Figure 2.2: Visualization of a prevented premature termination. In this case the vehicle is not detected in frame 100-112 [Imm22]

After the advantage of Immortal Tracker is discussed, the algorithm of Immortal Tracker will be explained (see figure 1). Immortal Tracker is an online tracking algorithm that takes sequential detection results as input. But before that, Non-Maximum Suppression (NMS), which combines overlapped boxes with similar yaw, is applied to remove redundant boxes before feeding them to Immortal Tracker. The object locations for all tracklets will be predicted using a vanilla 3D Kalman Filter (3DKF). When no detection is provided, the vanilla 3D Kalman filter for trajectory prediction assumes velocity to be constant at 1, but for unmatched detections, velocity will be a multiple of the time lag between the frames. Initially, the value was set to 0. For the association, 3D IoU or generalized GIoU is computed to ensure better recall between detected 3D bounding boxes and boxes predicted by Immortal Tracker.

The method performs bipartite matching between detections and predictions using a Hungarian algorithm based on 3D IoU or Generalized GIoU. If the 3D IoU or GIoU of matching is higher than the threshold IoU or GIoU, it will be considered; otherwise, it will be discarded. The output from Hungarian will be the matched pairs and the unmatched tracklets. (X_m,X_{um}) or detections (D_m,D_{um}). Where X_m and D_m are the k matched pairs of predictions and detections. X_{um} are the unmatched tracklets and D_{um}

are the unmatched detections. The matched detections D_m will be used to update the 3DKF states of corresponding tracklets X_m. The unmatched tracklets X_{um} will use their own predictions to update 3DKF [Imm22]. Finally, the unmatched detecions D_{um} will be initialized as new tracklets $Xnew$. This strategy is called Tracklet Birth and Preservation. This strategy basically based on initialize a set of tracklets $Xnew$ for detections in D_{um}. If these new tracklets are correctly associated with detections for The minimum birth count $Mhits$ times in the next frames, they will be marked as alive, else they remain at the birth stage. Note that initially all detections are D_{um} [Imm22].

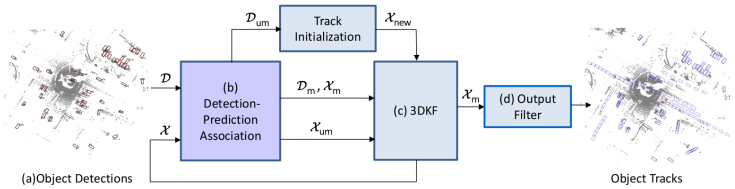

Figure 2.3: ImmortalTracker algorithm [Imm22]

Finally, when has Immortaltracker terminated a tracklet? Immortal Tracker preserves and maintains all tracklets for objects gone dark continuously. This method could successfully reduce identity switch cases by 96 % [Imm22]. So, there is no premature termination for any tracklet.

2.3 Cross-validation

Cross-validation is one of the most widely used data resampling methods to estimate the true prediction error of models and to tune model parameters in machine learning [Ber19]. It used statistical modeling to evaluate the performance of a predictive model on an independent dataset. The dataset that is available to evaluate a predictive model is referred to as the learning set [Ber19].

The basic idea of cross-validation is to split the available data into two sets, a training set and a validation set, and then repeatedly train and evaluate the model on different subsets of the data. It is commonly used to compare and select a model for a given predictive modeling problem because it makes it easier to understand and results in skill estimates that generally have a lower bias than other methods.

Cross-validation can be used to evaluate the performance of an object detection model on a dataset by splitting the dataset into multiple folds. It trains the model on one subset and tests it on the remaining subsets. There are different methods to perform cross-validation

in object detection, such as k-fold cross-validation, Leave-One-Out cross-validation, and exhaustive cross-validation. The choice of the method depends on the dataset and the specific research question being addressed.

In the next section, the two different cross-validations, LOOCV and E5CV, that were obtained in this project will be presented.

2.3.1 Types of cross-validation

There are different model evaluation techniques used for cross-validation, such as Leave-One-Out cross-validation (LOOCV) and exhaustive cross-validation (E5CV). The procedure as well as the advantages and disadvantages of both methods will be explained.

In the best case, a split validation subset is suitable to assess the model's performance if the obtained data is large enough. However, this is often not possible. So, to solve this problem, the K-Fold cross-validation algorithm is used. It splits the data equally into K folds of data to fit and validate the model. Usually, one of these folds is separated to validate the model and fits onto the remaining folds. This case is known as Leave-One-Out cross-validation [Gon21].

LOOCV is simply K-fold cross-validation, where K is the number of instances in the dataset. Each instance is left out in turn, and the learning method is trained on all other instances. For success, the result will be one, and for failure, zero. The results of all n judgments, one for each member of the dataset, are averaged, and that average represents the final error estimate [Wit11].

This procedure is suitable when there is a small dataset or when a precise estimate of model performance is more important than the computational cost of the method. The drawback of LOOCV is that it has extremely high computational complexity since training must be carried out as many times as there are objects in the training set [Mor21]. According to [Mor21The]The most popular cross-validation schemes, in particular Leave-One-Out, suffer from the necessity to multiply repeat the model estimation on different subsamples of the training set [Mor21].

The second cross-validation method is exhaustive cross-validation. This method basically learns and tests all possible combinations of hyperparameters, which are evaluated using 5-fold cross-validation. It is done by dividing the original sample into a training and a validation set [Sop21].

The difference between 5-fold cross-validation and exhaustive 5-fold cross-validation is that in 5-fold cross-validation, the process is repeated five times using different folds for evaluation each time. The data is divided into five equal folds. After that, the model is

trained on four folds and evaluated on the remaining fold. While exhaustive 5-fold cross-validation involves evaluating all possible combinations of hyperparameters using 5-fold cross-validation [Sop21]. For each time, the model is trained on 4 folds and evaluated on the remaining fold. This means that for each fold, the process is repeated five times for the remaining fold. So, for each fold, the process is repeated five times. This will result in 25 evaluations in total for each combination. This method is computationally expensive because it trains and evaluates the model in all possible ways using 5-fold cross-validation. But E5CV can be useful for selecting the optimal combination of hyperparameters and improving the performance of the mode.

2.4 Evaluation metrics

In this section, the metrics, which are used for detection and tracking results evaluation, will be introduced. The importance of each metric as well as the differences between them will be discussed. The criterion for determining whether a result is good or bad will also be explained. But first, it is important to explain the task of detection and tracking.

2.4.1 Detection task and metrics

Object detection is a computer technology affined to computer vision that works with instances detecting for different classes like humans, cars, or other classes which are mentioned in table 2.1.

A multitude of task detection is provided by the multimodal of nuScenes, where the detection task is defined to run on sensor data between [t - 0.5 t] seconds for an object at time t and needs detecting 10 object categories. Each class has 3D bounding boxes, velocities, and attributes. From the 10 classes, a subset of all 23 classes is annotated in nuScenes. Each class has its own upper bound on the evaluated detection range, as shown in table 2.1. For each detection class, as the range from the ego vehicle increases, the number of annotations will decrease, but the number of annotations per range varies by class [nus20]. nuScenes detection metrics consist of mean average precision, average translation error, average scale error, average orientation error, average velocity error, and average attribute error, and propose a comprehensive metric, specifically a nuScenes detection score, to set up the benchmark and evaluate different methods

The AP metric is a common metric used to measure the accuracy of object detectors. It computes the average precision value for the recall value over the range of 0 to 1, so this metric should be as close to 1 as possible to say the results are good. The following definitions are necessary to understand this metric.

Table 2.1: Object classes with attributes and an upper bound on the evaluated detection range [nus20]

Classes	Attributes	Range
car	vehicle moving, parked, stopped	50
pedestrian	moving , standing , sitting lying down	40
bicycle	cycle with(out) rider	40
motorcycle	cycle with(out) rider	40
barrier	void	30
traffic cone	void	30
bus	vehicle moving, parked, stopped	50
construction vehicle	vehicle moving, parked, stopped	50
trailer	vehicle moving, parked, stopped	50
truck	vehicle moving, parked, stopped	50

Precision = $\frac{TP}{TP+FP}$ measures what proportion of the detected objects are true objects, i.e., it calculates the amount of true positives (TP) over the sum of true positives and false positives. This metric measures how accurate the predictions are, and it is calculated by dividing true positives over all detections [Hem22]. While Recall = $\frac{TP}{TP+FN}$ measures the ability to capture all the relevant cases for an image. It measures the proportion of true objects detected, i.e., it is calculated as the amount of true positives over the sum of true positives and false negatives, which gives all ground truths [Hem22].

The metric calculation is done based on finding the closest match among ground truth boxes, and if this is close enough according to threshold, then it is considered that there is a match, and it will be marked as TP. Otherwise, if there is no match, it will be marked as a false positive (FP). Results are sorted based on the confidence score, meaning that predictions with high confidence scores will be matched first. As an example, see figure 2.4, which contains a match between some ground truths and prediction boxes according to the center distance threshold, where different ground truth boxes in green color and prediction boxes with just a high score in blue color are appearing. As mentioned before, if the closest match is close enough according to threshold, then there is a match. The matching criterion in AP metric is defined by considering the 2D center distance on the ground plane instead of intersection over union (IOU). The main reason for using this criterion is to decouple detection from an object's size and orientation, and because predictions are matched with the ground truth objects that have the smallest center distance up to a definite threshold. Furthermore, if objects with small footprints are detected with a small translation error, this will result in a zero IOU. That makes the

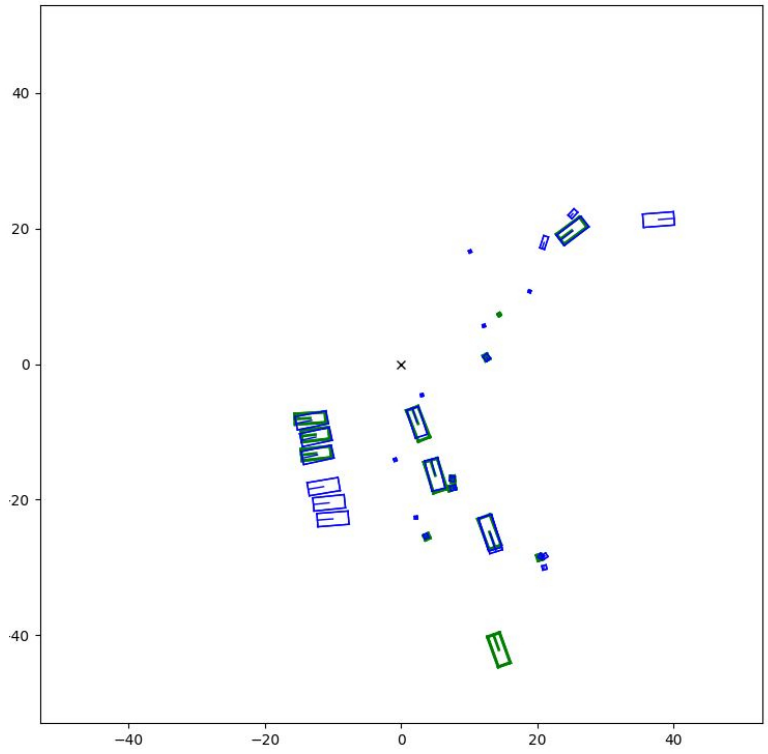

Figure 2.4: GT boxes in green, and prediction boxes with high score blue for one sample

comparison difficult to make between the performance of vision-only methods, which lead to large localization errors as shown in figure 2.5 [nus20].

After the matching criterion for AP is defined, the next step is to calculate AP as the normalized area under the precision recall curve for recall and precision over 10 %. The values of AP are between [0, 1], which is the area under the precision recall curve. For the recall, it is expected that 101 steps between [0 % - 100 %] will be obtained. The results from AP are considered good or bad as follows, AP will be compared for different methods, and for the higher AP, it is considered that it has a better detector.

Usually, the impact of noise can be noticed in low recall and precision areas, so in order to make the noise effect as minimal as possible, the operating points where recall or precision is less than 10 % will be removed. And the AP for that class is set to zero, in case no operating point is done in this area [nus20]. Finally, the average is a token over the match thresholds of values D = [0.5, 1, 2, 4] meters. The calculation of AP only involves one class. However, in object detection, there are usually many classes. The mean across classes mAP = $|C|\,|D|\,\Sigma_{c\in C}\,\Sigma_{d\in D}\,AP_{c,d}$. The second metric used in nuScenes is the TP

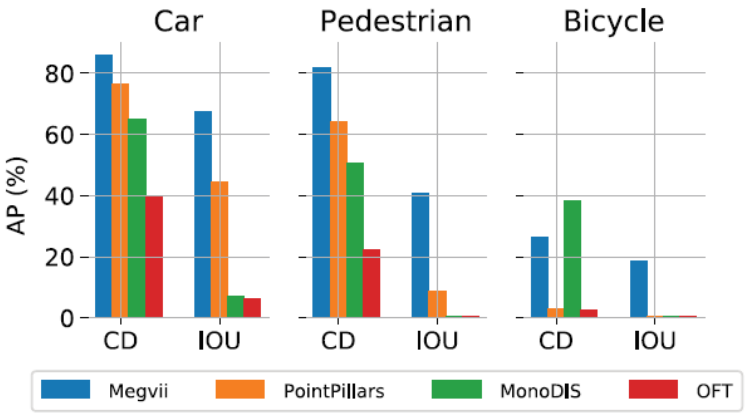

Figure 2.5: AP vs. matching function using CD and IOU. CD = 2m and IOU =.7 for car and IOU=0.5 for pedestrian [nus20].

metric. For each prediction and ground truth box where they are matched together, a set of TP metrics will be measured. TP measures translation, scale, orientation, velocity and attribute errors; all of these metrics are designed to be positive scalars and calculated during matching using the center distance of 2m, and they are in native units in order to make the results easy to compare and interpret. Per class, matching and scoring will occur independently.

For each metric, if the recall level doesn't reach 10 % for a specific class, all TP errors will be set to 1. But for each metric at each achieved recall level above 10 %, is the average of the cumulative mean.

The five TP metrics are calculated using different definitions, so here is the definition for each metric [nus20]:

- Average Translation Error (ATE) is the Euclidean center distance in 2D and its unit in meters.
- Average Scale Error (ASE) is the 3D intersection over union after aligning orientation and translation (1 - IOU).
- Average Orientation Error (AOE) is the smallest yaw angle difference between prediction and ground truth in radians. All angles are measured on a full 360◦ period except for barriers where they are measured on a 180◦ period.
- Average Velocity Error (AVE) is the absolute velocity error as the L2 norm of the velocity differences in 2D (m/s).

- Average Attribute Error (AAE) is defined as 1 minus attribute classification accuracy (1 - acc).

The following metrics are not defined for class cones AVE, AOE, and AAE because it is stationary, has a poorly defined orientation, and has no attributes. Metrics AVE and AAE are not defined for class barriers because they are stationary and have no attributes. Each TP metric is calculated for one class, but to calculate TP over all classes, the mean TP metric mTP = $\frac{1}{|C|} \sum_{c=C} TP_c$. The third metric used in nuScenes is called nuScenes detection score (NDS). To understand why this metric is important for nuScenes detection tasks, a comparison of different metrics is presented below.

The common metric used for object detection is mAP with a threshold IOU, but this metric is not the best way to capture all aspects of the nuScenes detection task, like velocity and attribute estimation, and it couples' location, size, and orientation estimates [nus20]. There are some methods for defining thresholds for each error type and recall threshold, but this makes this approach complex. Instead of that, in nuScenes consolidating the different error types onto a scalar score is proposed, which is given by NDS = $\frac{1}{10}$ [$5\, mAP + \Sigma_{mTP \in TP}\, (1 - min(1, mTP))$]. NDS is defined as a combination of mAP and mTP, the set of the five mean TP metrics. Half of NDS is thus based on the detection performance, while the other half quantifies the quality of the detections in terms of location, size, orientation, attributes, and velocity. First, the TP errors are converted to TP scores; after that, 1 is added to each of the 5 TP scores, giving a weight of 5 to the mAP. In order to ensure that each metric is bound between [0, 1], Finally, the normalized sum for NDS is calculated.

2.4.2 Tracking task and metrics

The tracking task is defined to operate on sensor data between [0 - t] seconds for an object at time t. The focus of the tracking task is to track the object classes from the detection classes, which were defined in the last section, except for the stationary classes like traffic cones, barriers, and construction. The main goal is to perform tracking on all moving objects in a scene. Similar to the detection task, only the recall level over 10 % will be considered; the recall level less than 10 % is removed.

The matching criterion for the tracking metrics is the same one used for the detection metrics: a match is made by thresholding the 2D center distance on the ground plane, and the center distance is set to 2m.

In nuScenes, different tracking metrics are defined. The first metrics are called the traditional MOT metrics, which contain MOTA and MOTP metrics. The second metrics are

called the main metrics AMOTA and AMOTP, and the third metrics are called the novel metrics TOD and LGD. Each metric from those will be defined, as well as their challenges and why every metric is needed.

Multi object tracking accuracy MOTA $= 1 - \frac{|FN| + |FP| + |IDSW|}{|gtDet|}$: measures combine three error sources: false positives, missed targets, and identity switches. The final MOTA score is calculated by summing up the total number of these errors, dividing by the number of ground detections, and subtracting from one [hota22]. Hier matching is done at detection level. In other words, it performs both matching and association scoring at a local detection level, which biases scores toward measuring detection.

A bijective one-to-one mapping is constructed between prediction detections and ground detections in each frame. Any prediction detection that is correct and GT detections that are matched become true positives. Any remaining prediction detections that are not matched, the extra predictions, become false positives. Any ground truth detections that are not matched or have missing predictions become false negatives [hota22]. On the other hand, association is measured by identity switching, which occurs when a track is lost and is reinitialized with a different identity. The range of this metric is between 0 and 1, which means that a value should be as close to 1 as possible to be a better result.

The challenge with this metric is that it doesn't include a measure of localization error, and detection performance significantly outweighs association performance. As mentioned before, MOTA doesn't include a measure of localization error. So a secondary metric, called MOTP, is defined.

Multi-Object Tracking Precision MOTP $= \frac{1}{|TP|} \sum_{TP} S$: measures localization accuracy. It is defined as the average distance between the predicted object location and the ground truth object location, over all predictions that are successfully matched to a ground truth. It is clear that the distance between them needs to be small, so the MOTP metric should be as close to zero as possible. The range of this metric is set in nuScenes from 0 to 2 meters. 2 means the result is the worst. Note that in some multiple object tracking benchmarks, such as the MOT Challenge benchmark, MOTP is listed as a percentage, and the goal is for it to be as high as possible. MOTP is simply the average similarity score S over the set of true positives [hota22]. MOTP measures the localization accuracy and averages the overlap between all matched predictions that are correct and their ground truth. It matches predictions with ground truths that have a similarity score greater than the threshold ($S \geq a$) and does not cause an ID switch. So, it can be noted that the threshold α has a big effect on the behavior of this metric.

The challenge of MOTP is that it quantifies the localization accuracy of the detector and, thus, gives little information about the actual performance of the tracker. These

traditional metrics don't take the confidence of a prediction into account. Therefore, two main metrics, AMOTA and AMOTP, are developed.

Average multi object tracking accuracy (AMOTA) takes the average of MOTA across all recall thresholds. Due to the difficulty of using nuScenes for detection and tracking, the traditional MOTA metric is often zero. Thus, an updated formulation is founded called sMOTAr $= max\ (\ 0, 1 - \frac{IDS_r + FP_r + FN_r - (1-r)\ P}{r\ P}\ )$; this result in MOTA is augmented by a term to adjust the respective recall. In sMOTAr include the recall-normalization term - (1 - r) * P in the nominator, the factor r in the denominator, and the maximum. This is done in order to guarantee that the value of sMOTAr is within the [0 , 1] range and bring the three error types into a similar value range. Note that P refers to the number of ground truth positives for the current class. nuScenes perform a 40-point interpolation in the recall range [0.1 , 1]. The resulting sAMOTA metric is the main metric for the tracking task AMOTA $= \frac{1}{|R|}\ \Sigma_{r \in R} sMOTA_r$, where the recall values are denoted as R [nus20]. The second main metric is average multi object tracking precision AMOTP $= \frac{1}{n-1}\ \Sigma_{r \in (\frac{1}{n-1}), \frac{2}{n-1}, .., 1)}\ \frac{\Sigma_{i,t} \in d_{i,t}}{\Sigma_{t \in TP_t}}$, this metric average MOTP across all recall thresholds. Here, `d_i,t` indicates the position error of track i at time t and `TP_t` indicates the number of matches at time t. There are also other traditional metrics defined in the nuScenes tracking task, the other metrics are [tra20]:

- FAF: The average number of false alarms per frame.
- MT (number of mostly tracked trajectories): The number of ground-truth trajectories that are covered by a track hypothesis for at least 80% of their respective life span.
- ML (number of mostly lost trajectories): The number of ground-truth trajectories that are covered by a track hypothesis for at most 20% of their respective life span.
- FP (number of false positives): The total number of false positives.
- FN (number of false negatives): The total number of false negatives (missed targets).
- IDS (number of identity switches): The total number of identity switches.
- Frag (number of track fragmentations): The total number of times a trajectory is fragmented (i.e., interrupted during tracking).
- FPS (tracker speed in frames per second): Processing speed in frames per second excluding the detector on the benchmark. Users report both the detector and tracking FPS separately as well as cumulative. This metric is self-reported and therefore not directly comparable.

The main tracking metrics are defined as well as the traditional metrics, but there are also new metrics added to nuScenes that are used to fix some challenges in the last metrics. The novel metrics are called track initialization duration in seconds (TID) and longest gap duration in seconds (LGD). Some trackers require a fixed window of past sensor readings. Trackers may also perform poorly without good initialization. The purpose of TID is to measure for each track the initialization duration until the first object is successfully detected. If an object is not tracked, the entire track duration is assigned as the initialization duration. Then the average over all tracks will be computed.

On the other hand, the task of LGD is to measure the number of fragmentations and the longest duration in case any detection gap is gotten. This metric is very important for the application of autonomous driving because, in the application of AVs, it is crucial to know how long an object has been missed. Both metrics measure duration for each track. After that For both metrics, the average will be computed over all tracks [nus20].

Finally, the table 2.2 shows the worst value for each metric. From this table, the tracking evaluation results can be compared to see which metrics give better results than the others.

Table 2.2: Tracking metrics worst values list

Metric	Worst	Metric	Worst
MOTA	0	MOTP	2
AMOTA	0	AMOTP	2
TID	20	LGD	20
MOTAR	0	Recall	0
MT	0	ML	high
TP	0	FAF	500
FP	high	FN	high
Frag	high	IDS	high

3 Evaluation strategy

The training data and results are obtained from the nuScenes dataset, which will be presented in this chapter. Next, the toolbox that is used to train the detection approaches will be introduced. In the end of this chapter, the experiment and procedures that were used to collect the data will be described.

3.1 Dataset

Nowadays, machine learning methods are becoming more widely used, so it's necessary to train and evaluate such methods on datasets that include range sensor data. However, most autonomous vehicles have sensors like lidar. For this task, the nuScenes dataset is used. nuScenes is a large-scale autonomous driving dataset. Researchers used it to study challenging urban driving situations using the full sensor suite of a real self-driving car on multiple tasks, such as object detection and tracking [nus20].

Since its release in 2019, nuScenes has become a common benchmark for evaluating perception, prediction, and planning algorithms for autonomous driving. Several state-of-the-art methods have been developed using the nuScenes dataset, including pointpillar and centerpoint.

nuScenes contains 1000 scenes, each with a duration of approximately 20 seconds, captured in urban areas around Boston and Singapore. Each scene has unique attributes and comes with a set of information. In other words, different times, different locations, different weather conditions, and different lidar visibility can be obtained. The training split in nuScenes consists of 850 scenes: 700 for training, 150 for validations, and 150 for testing. Furthermore, nuScenes the first dataset includes the full autonomous vehicle sensor suite with a total of 6 cameras, 1 lidar, and 5 radars with full 360-degree coverage. These multimodal datasets are complementary and provide redundancy against different conditions and failures. Considering the first sensor camera, it provides accurate measurements but it is difficult to determine the 3D localization from images. On the other hand, lidar gives extremely accurate localization but less information and sparse data with a range of 50–150 m. The nuScenes dataset contains a high-resolution lidar point cloud; the radar sensor provides a higher range between 200 and 300 m, but the returned data are sparser than lidar and less accurate for localization. It can be seen that the three sensors have different failure modes under different conditions [nus20].

In nuScenes, 3D detection and tracking metrics are defined. In this work, 10 object classes are used in a detection task using lidar. The classes are car, pedestrian, bus, truck, bicycle, barrier, motorcycle, construction vehicle, traffic cone, and trailer. In order

to improve object detection, 3D object detectors and trackers are trained as a baseline within a new approach of multiple lidar sweeps.

The full dataset contains approximately 1.4 million camera images, 390,000 LiDAR sweeps, 1.4 million radar sweeps, and 1.4 million object bounding boxes in 40K key frames, according to wang21. Object detection in the nuScenes dataset involves detecting and localizing objects in 3D using lidar and camera data. In this thesis, object detection is done based on just lidar data.

3.2 MMdetection3D framework and 3D lidar

In this section, the toolbox MMdetection3d, which is used for data training, is presented, as well as lidar-based 3D detection, one of the most basic tasks supported in this toolbox.

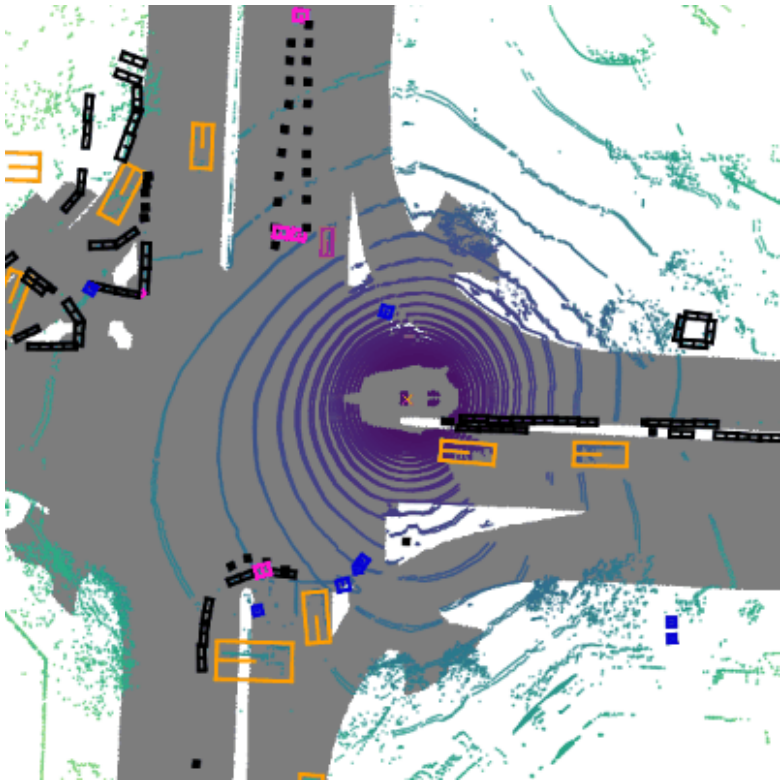

Figure 3.1: MMDetection 3D outdoor [mmdet20]

MMdetection3D toolbox is an open-source object detection framework based on PyTorch that is contributed by researchers and engineers from various colleges and companies. It's a part of the OpenMMLab project developed by MMLab, in respect of the next-generation

platform for general 3D detection, and it is available as open-source software under the Apache License 2.0 [mmdet20]. MMdetection3D trains faster than other codebases. It directly supports multi-modality and single-modality detectors, including pointpillars, centerpoints, etc. Using this codebase, more than 300 models can be trained and modules supported in MMDetection. Also, a model can be prepared, trained, and tested on a standard 3D detection benchmark; furthermore, it can be visualized and the results validated. It is built on top of tools for developing the PyTorch framework and gives a set of tools for developing and evaluating state-of-the-art 3D object detection. It also provides many types of 3D data, including point clouds and voxel grids. It also includes several pre-trained 3D object detection models like pointpillar, which achieved state-of-the-art results on the nuScenes benchmark.

The MMdetection3D framework has many features because it is easy to use; for example, it provides a user-friendly command interface for training models. It has also got multi-GPU training, which reduces the training time for large models. Furthermore, it is supported for multiple modalities like lidar, and it is designed to be modular with separate components, which make it easy to customize the framework for specific cases.

Lidar-based 3D detection is one of the most basic tasks supported in MMDetection3D. It uses a laser scanner to measure the distance to the environment, generating a sparse point cloud representation as a result. As input, it expects the given model to take any number of points with features collected by lidar and predict the 3D bounding boxes and category labels for each object of interest. Lidar is one sensor that provides the 3D information of the object in terms of a point cloud in order to localize the object [mmdet20]. Object detection in point clouds is an important aspect of many robotics applications, such as Avs. Recently, many novel approaches have transformed the point cloud into a pillar or voxel, which leads to a dense structure.

3.3 Experimental setup

In this section, the outline of the experimental setup for this project will be detailed. The main steps are dataset selection, object detection system selection, redundant tracklets generation, results evaluation, and finally state-of-the-art redundant tracklet comparison.

The used programming language is python. The other tools that are utilized in this project are nuScenes devkit, visual studio code, MMdetection3d and Immortal Tracker.

The experimental setup steps are done as follows:

Firstly, dataset selection: the chosen dataset is nuTonomy scenes (nuScenes), which is considered an autonomous driving dataset and suitable for training and evaluating the

performance of object detection algorithms. 10 different object classes are used in the detection task in different situations and conditions.

Secondly, select object detection system: Four different object detection systems are chosen. These systems are suitable and can produce tracklets over timeframes. The object detection are pointpillar, and centerpoint based on different sizes and voxels. The object detection architectures are pointpillar, centerpoint, voxel01, voxel0075, and pillar02. Combining the predictions of multiple models can often result in better performance than using a single model.

Thirdly, data splitting: three different splits are used for the selected object detection systems. The splits are Leave-One-Out cross-validation, exhaustive 5 cross-validation, and the original split from nuScenes, resulting in a total of 64 models.

Fourthly, data training: the selected model on the preprocessed dataset is trained using the MMdetection3D toolbox based on the PyTorch framework. This toolbox trains and evaluates the models at the same time. The sensor used is lidar for 3D detection.

Fifthly, the detection system evaluation: the performance of the trained models are evaluated on a separate validation dataset using different metrics such as average precision, recall, average translation error, average scale error, average orientation error, average velocity error, and average attribute error.

Sixthly, redundant tracklets generation: redundant tracklets by using different detection algorithms are created. All detected objects are tracked using the Immortal Tracker tool.

Seventhly, tracking algorithm evaluation: the performance of the tracklets is evaluated using different metrics such as MOTA, MOTP, AMOTA, AMOTP, TID, LGD and IDS.

Eighthly, from the redundant tracklets, additional information can be obtained, and it can be used to confuse the results with each other and improve the detection approaches.

There are some preprocessings steps before detection and tracking evaluation are started, such as ground truths and predictions that are removed if they exceed the class specific tracking range. Furthermore, all ground truth boxes without lidar points in them are removed, this is done due to it can be not guaranteed that they are actually visible in the frame. Predicted boxes does not be filtered based on number of points. Finally, all ground truth and predicted tracks are interpolated linearly to avoid excessive track fragmentation from lidar points [tra20].

4 Results

This chapter covers the results obtained for both object detection and object tracking. The detection results will cover all 10 object classes, which will be evaluated using different metrics, but in object tracking only class car is covered, and some chosen examples will be discussed.

4.1 Detection results

Object detection results cover all object detection classes, which are introduced in nuScenes. Results are provided for class cars, pedestrians, bicycles, motorcycles, barriers, traffic cones, buses, construction vehicles, trailers, and trucks. For mean average precision, matching is done using 2D center distance, and for a given match threshold, AP is calculated by integrating the recall precision curve for recalls and precisions larger than 0.1. Finally, the average over match thresholds of [0.5, 1, 2, 4] meters is calculated, and the mean across all classes is computed. All TP metrics for all classes are calculated using a 2-meter CD threshold during matching.

All detection results contain information about eight boxes, as follows:

- The first box is a sample token, which identifies the keyframe for which objects are detected in every scene.
- The second box is translation, which is considered the estimated bounding box location in meters in the global frame center-x, center-y, and center-z.
- The third box is size, which is considered the estimated bounding box size in meters: width, length, and height.
- The fourth box is rotation, which is considered the estimated bounding box orientation as a quaternion in the global frame w, x, y, and z.
- The fifth box is velocity, which is the estimated bounding box velocity in m/s in the global frames vx and vy.
- The sixth box is the detection name, which is the predicted class for the sample result, such as car, pedestrian, and so on.
- The seventh box is the detection score, which is considered an object prediction score between 0 and 1 for the class identified by the detection name.

- The eighth box is the attribute name that carries information about the name of the predicted attribute, or for classes without attributes, it will carry an empty string for them, which means that the predicted attributes for these cases will be ignored [det20].

Table 4.1: Mean average precision for all classes in detection results for Leave-One-Out split and original split

Detection	LOOCV 1	LOOCV 2	LOOCV 3	LOOCV 4	LOOCV 5	Original
PointPillar	0.3464	0.3753	0.3395	0.3554	0.3630	0.3487
Pillar 02	0.3464	0.3895	0.3966	0.3978	0.4012	0.3898
Voxel 01	0.4999	0.5010	0.5075	0.4788	0.5125	0.5144
Voxel 0075	0.5108	0.4942	0.5163	0.4970	0.5247	0.5076

Table 4.2: Mean average precision for all classes in detection results for exhaustive 5 cross-validation split

Detection	E5 1	E5 2	E5 3	E5 4	E5 5	E5 6	E5 7	E5 8	E5 9	E5 10
Pointpillar	0.352	0.318	0.325	0.332	0.319	0.344	0.341	0.332	0.300	0.344
Pillar 02	0.391	0.366	0.401	0.37	0.367	0.391	0.366	0.382	0.366	0.377
Voxel 01	0.478	0.487	0.507	0.467	0.454	0.478	0.483	0.5	0.477	0.485
Voxel 0075	0.485	0.487	0.507	0.468	0.464	0.486	0.496	0.512	0.493	0.485

The mean average precision and TP metrics are calculated for different detection architectures (pointpillar, centerpoint, voxel01, voxel0075, pillar02, and their splits), which means all 64 models are evaluated using different metrics, including the main metric mean average precision for Leave-One-Out split as well as the original split as seen in table 4.1. On the other hand, the evaluation for exhaustive 5 cross-validation is seen in table 4.2.

It is important to start comparing results to see which ones are better than others, which ones have fewer errors, and so on. Firstly, let's consider the results from the original split. The high mean average precision is for centerpoint voxel01 and the worst one is pointpillar. Now, from the LOOCV split, it can be observed that the best performance is for centerpoint voxel0075 LOOCV-5, but the worst performance belongs to pointpillar LOOCV-3. By the same logic, all results can be confused and compared. Another used metric in evaluation is the TP metrics table 4.3. From the four detection approaches, it can be seen which one gives better mAP, NDS and TP metrics.

Table 4.3: Mean average precision, the set of the five mean TP metrics and nuScenes detecton score

Detection	mAP	mATE	mASE	mAOE	mAVE	mAAE	NDS
PointPillar	0.3487	0.4402	0.2840	0.5124	0.3396	0.1779	0.4989
Pillar 02	0.3898	0.3240	0.2647	0.3693	0.3916	0.1962	0.5403
Voxel 01	0.5144	0.2979	0.2545	0.3170	0.2888	0.1929	0.6221
Voxel 0075	0.5076	0.2912	0.2536	0.3103	0.2678	0.1965	0.6218

Object detection results also include other information for each class, such as the size, translation, velocity, attribute, and orientation of the object. The detection results include metrics evaluations for AP, ATE, ASE, AOE, AVE, and AAE per class. Let's take an example from the results for the original splits as shown in tables 4.4, table 4.5, table 4.6 and table 4.7. These tables contain the five true positive metrics as well as the average precision metric for all classes for the different four detection architectures. From each split, such results are obtained. So, here also, the results from different splits for each class can be confusing, and it can be seen whether the performance is good or bad. For example, the average translation error for centerpoint-voxel 0075 in class car is the best, and the worst is pointpillar. On the other hand, the average attribute error for centerpoint-voxel01 is better than other approaches. From the results also, it can be observed, that the better results are for class car and pedestrian, which give higher average precision than other classes.

Table 4.4: Object detection result metrics per class pointpillar

Object class	AP	ATE	ASE	AOE	AVE	AAE
car	0.781	0.222	0.157	0.175	0.273	0.195
Truck	0.315	0.502	0.223	0.286	0.290	0.222
Bus	0.365	0.527	0.206	0.204	0.575	0.192
Trailer	0.183	0.818	0.248	0.682	0.256	0.173
Construction vehicle	0.016	0.865	0.530	1.460	0.144	0.350
pedestrian	0.705	0.164	0.280	0.359	0.254	0.066
Motorcycle	0.277	0.275	0.260	0.550	0.634	0.198
Bicycle	0.085	0.295	0.291	0.838	0.290	0.026
traffic cone	0.284	0.265	0.353	nan	nan	nan
Barrier	0.477	0.469	0.293	0.057	nan	nan

Table 4.5: Object detection result metrics per class centerpoint-pillar02

Object class	AP	ATE	ASE	AOE	AVE	AAE
car	0.796	0.190	0.157	0.180	0.359	0.219
Truck	0.458	0.347	0.193	0.146	0.308	0.265
Bus	0.607	0.340	0.184	0.080	0.733	0.342
Trailer	0.285	0.555	0.210	0.403	0.276	0.153
Construction vehicle	0.124	0.672	0.423	1.043	0.117	0.318
pedestrian	0.732	0.166	0.273	0.354	0.243	0.070
Motorcycle	0.359	0.220	0.239	0.422	0.856	0.167
Bicycle	0.094	0.202	0.284	0.550	0.243	0.035
traffic cone	0.360	0.206	0.368	nan	nan	nan
Barrier	0.084	0.342	0.316	0.145	nan	nan

Table 4.6: Object detection result metrics per class centerpoint-voxel0075

Object class	AP	ATE	ASE	AOE	AVE	AAE
car	0.820	0.178	0.156	0.111	0.272	0.194
Truck	0.509	0.323	0.178	0.103	0.261	0.244
Bus	0.667	0.317	0.179	0.052	0.437	0.276
Trailer	0.305	0.544	0.213	0.491	0.194	0.139
Construction vehicle	0.139	0.715	0.424	0.917	0.126	0.354
pedestrian	0.831	0.143	0.280	0.375	0.218	0.082
Motorcycle	0.545	0.185	0.237	0.260	0.439	0.267
Bicycle	0.315	0.155	0.266	0.397	0.195	0.018
traffic cone	0.616	0.136	0.332	nan	nan	nan
Barrier	0.329	0.217	0.271	0.088	nan	nan

Table 4.7: Object detection result metrics per class centerpoint-voxel01

Object class	AP	ATE	ASE	AOE	AVE	AAE
car	0.823	0.184	0.154	0.117	0.298	0.191
Truck car	0.518	0.318	0.180	0.104	0.264	0.244
Bus	0.672	0.346	0.183	0.048	0.471	0.265
Trailer	0.323	0.544	0.210	0.402	0.202	0.122
Construction vehicle	0.170	0.650	0.438	1.042	0.120	0.335
pedestrian	0.818	0.167	0.283	0.406	0.232	0.088
Motorcycle	0.514	0.196	0.239	0.263	0.511	0.276
Bicycle	0.311	0.161	0.267	0.391	0.212	0.023
traffic cone	0.608	0.156	0.332	nan	nan	nan
Barrier	0.387	0.259	0.261	0.079	nan	nan

Table 4.8 is one example of the difference between matching using center distance and intersection over union threshold and which one gives better average precision.

Table 4.8: Average precision using matching function IOU=[0.3, 0.5, 0.7, 0.9] for detection pointpillar

Object	Car	Truck	Bus	Trailer	Const	Pedestrain	Motor	Bicycle	traffic
AP	0.276	0.401	0.128	0.234	0.153	0.375	0.127	0.072	0.318

Considering one example for pointpillar architecture, where the matching threshold IOU is used in matching, As mentioned in chapter 2, figure 2.5, If objects with small footprints, like bikes, are detected with a small translation error, give a 0 IOU. This makes it hard to compare the performance of vision-only methods, which tend to have large localization errors. It also couples location, size, and orientation estimates. In this example, four different thresholds are used, and the average is calculated over the match thresholds [0.3, 0.5, 0.7, 0.9]. From the result, it is clear that the average precision for car and pedestrian using IOU is much lower than using the center distance threshold. Furthermore, the mean average precision using IOU for all classes is 0.24, but using center distance, the mean average precision is 0.3487, which is better than using IOU.

Now, all information about objects has been obtained and can be used to obtain redundant tracklets to get more information about them, which can help improve detection accuracy.

4.2 Tracking results

The tracking results will be divided into three parts. The first part will explain the overall performance of the four trackers: pointpillar and centerpoint voxel01, voxel0075 and pillar02. In the second part, the results are obtained for every scene using redundant trackers; this will be explained, and some examples are given for the best and worst scenes, as well as clear and complex scenes. In the third part, some interested instances will be considered and study their behaviours. In all cases, the tracking results contain information about eight boxes, as follows:

- The first box is a sample token, which identifies the keyframe for which objects are detected in every scene.
- The second box is translation, which is considered the estimated bounding box location in meters in the global frame center-x, center-y, and center-z.
- The third box is size, which is considered the estimated bounding box size in meters: width, length, and height.
- The fourth box is rotation, which is considered the estimated bounding box orientation as a quaternion in the global frame w, x, y, and z.
- The fifth box is velocity, which is the estimated bounding box velocity in m/s in the global frames vx and vy.
- The sixth box is the tracking-id, which is used to identify an object track across samples and is a unique object id.
- The seventh box is the tracking name, which is the predicted class for the sample result, such as car, pedestrian, and so on.
- The eighth box is the tracking score, which is an object prediction score between 0 and 1 for the class identified by the tracking name. All scores are averaged over frame level to compute the track level score, and the score will be used to determine positive and negative tracks via thresholding [tra20].

For all tracking result metrics, except false positives, they are computed per class and then averaged over all classes.

In the first part, the tracker results for the original splits are obtained using two methods using the Immortal Tracker tool. In the first method, all scenes are evaluated together in one file using main, secondary, and novel metrics in table 4.9 and table 4.10. The first

thing that is interesting in these results is the number of identity switches. As seen in table 4.9 the number of identity switches for voxel0075 is lower than for other approaches. The percentage of the number of identity switches related to the number of ground truths is about 0.02. This result is considered very good. The reason behind this good performance is that the tool used in tracking tasks is Immortal Tracker, which maintains trackelets that have lost their targets and can possibly reappear in the future.

Different metrics are used to evaluate each detection architecture; for example, it can be seen that MOTA for centerpoint-voxel01 is the best, but for pointpillar is the worst. MOTAR and MOTP for Centerpoint-voxel0075 are better than the other trackers. The highest AMOTP and AMOTA are for centerpoint-voxel01, but the lowest value is for pointpillar. In case evaluation using the novel metrics TID, which measures for each track the initialization duration until the first object is detected, and LGD which measures the number of fragmentations, both metrics indicate that centerpoint-voxel01 has the best performance, and pointpillar the worst. Now, going to table 4.10 , it can be observed that the highest fp and fn are for pointpillar. On the other hand, the highest tp number is for voxel01. From other metrics, mt and ml, information about the number of mostly tracked trajectories and the number of mostly lost trajectories can be obtained. So, the best mt and ml are for centerpoint-voxel01, and the worst ml and mt are for pointpillar.

Using different metrics in evaluation tasks is very important because it provides a clear image of the performance of models and helps to understand the behavior of trackers.

Table 4.9: Tracking results all scenes evaluation using different metrics

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.2892	0.6307	0.4413	0.3454	0.7793	408	0.7188	1.1497
Pillar02	0.4568	0.7669	0.3639	0.5375	0.6586	472	0.6565	0.9600
Voxel01	0.5530	0.7993	0.3709	0.6553	0.6139	463	0.5692	0.8122
Voxel0075	0.5446	0.8039	0.3573	0.6449	0.6204	332	0.5939	0.8126

Tracking results include a confidence score for trackers independently per class. This confidence score is selected for every class independently by picking the threshold that achieves the highest MOTA. Tracks with a score below the confidence threshold are ignored. The track level scores are determined by averaging the frame level scores. For example, in figure 4.1 selected confidence score for class car, it can be observed that the highest confidence is for voxel01, and the lowest confidence is for pointpillar. For each class, there is also such a confidence score.

Second part, every scene in a detection result is tracked independently from other scenes,

Table 4.10: Tracking results all scenes evaluation using additional metrics

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.4399	3313	2633	54.32	65936	16359	35553	443	14556.71
Pillar02	0.5869	3853	2044	47.88	73749	13853	27676	439	14556.71
Voxel01	0.6869	4206	1773	47.91	78240	13860	23194	428	14556.71
Voxel0075	0.6727	4132	1792	43.99	77922	12365	23643	426	14556.71

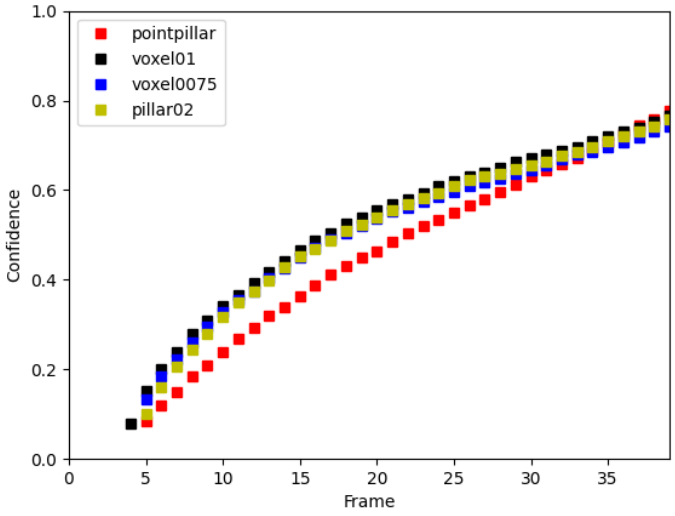

Figure 4.1: Confidence for overall system pointpillar and centerpoint architectures for class car

also for every class. This is done in order to be able to study every scene and see which ones give good results and which ones give bad results. Every scene comes with different situations. So, it will be easier to understand every scene and instance within it. All scenes, which are obtained from Immortal Tracker, are evaluated using different metrics. From the results, it can be clearly seen which scene has a good performance and which has the worst performance. Also, it can be observed where the most identity switches occur. After the critical scenes are determined, it can be started to choose some instances to study them, and to study different scene situations.

Some selected examples for the most critical scenes. The chosen cases are for class car. The selected scenes are the best and worst scenes, and scenes with the most identity

switches for the different four trackers. Starting with the best scenes for each tracker: pointpillar, centerpoint-voxel01, voxel0075 and pillar02.

Table 4.11: Tracking results evaluation best scene performance for pointpillar scene: 9709626638f5406f9f773348150d02fd

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.955	0.977	0.291	0.973	0.269	0	0	0.333
Pillar02	0.91	0.942	0.346	0.947	0.366	0	0.167	0.333
Voxel01	0.933	0.954	0.344	0.956	0.33	0	0.167	0.167
Voxel0075	0.944	0.966	0.316	0.972	0.322	0	0	0.333

Table 4.12: Tracking results evaluation best scene performance for pointpillar additional metrics scene: 9709626638f5406f9f773348150d02fd

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.978	2	0	5	87	2	2	0	89
Pillar02	0.966	2	0	12.5	86	5	3	0	89
Voxel01	0.978	2	0	10	87	4	2	0	89
Voxel0075	0.978	2	0	7.5	87	3	2	0	89

Table 4.13: Tracking results evaluation best scene performance for centerpoint-pillar02 scene: b4b82c4d338a4b6d86835388ce076345

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.673	0.856	0.384	0.766	0.429	2	0.156	0.281
Pillar02	0.876	0.973	0.289	0.949	0.284	1	0.1	0.125
Voxel01	0.849	0.876	0.395	0.915	0.365	2	0.19	0.19
Voxel0075	0.724	0.929	0.309	0.877	0.347	1	0.395	0.421

Third part: some examples from different scenes to understand the behavior of instances moving and see how identity switching can occur. Before starting with the results, it is important to know that all errors calcuation are sorted based on the annotation of instances. In nuScenes, each scene comes with about 40 samples. So, all results are calculated based on the annotation of the instances over the 40 samples. The important

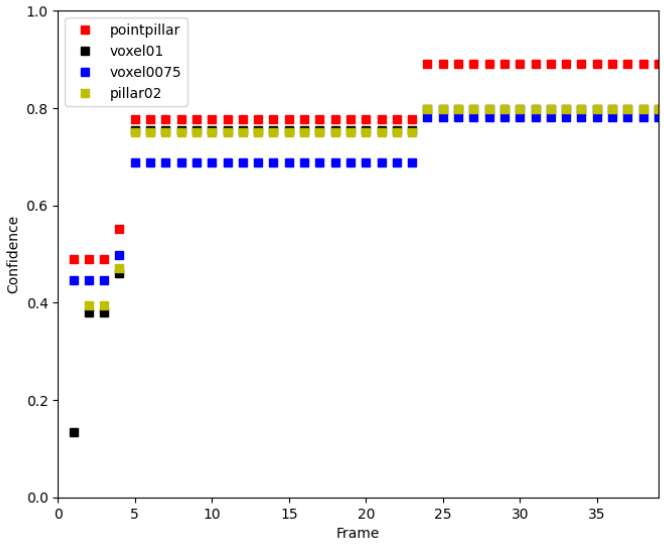

Figure 4.2: Best confidence scene for pointpillar

Table 4.14: Tracking results evaluation best scene performance for centerpoint-pillar02 additional metrics scene: b4b82c4d338a4b6d86835388ce076345

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.791	15	6	130.77	354	51	94	3	450
Pillar02	0.902	20	2	28.21	405	11	44	1	450
Voxel01	0.973	20	1	138.46	436	54	12	2	450
Voxel0075	0.782	17	4	64.1	351	25	98	2	450

Table 4.15: Tracking results evaluation best scene performance for centerpoint-voxel01 scene: efa5c96f05594f41a2498eb9f2e7ad99

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.956	0.97	0.23	0.947	0.304	0	0.125	0.125
Pillar02	0.985	0.985	0.189	0.99	0.167	0	0	0
Voxel01	0.985	0.985	0.185	0.996	0.17	0	0	0
Voxel0075	0.956	0.97	0.15	0.966	0.168	0	0.125	0.125

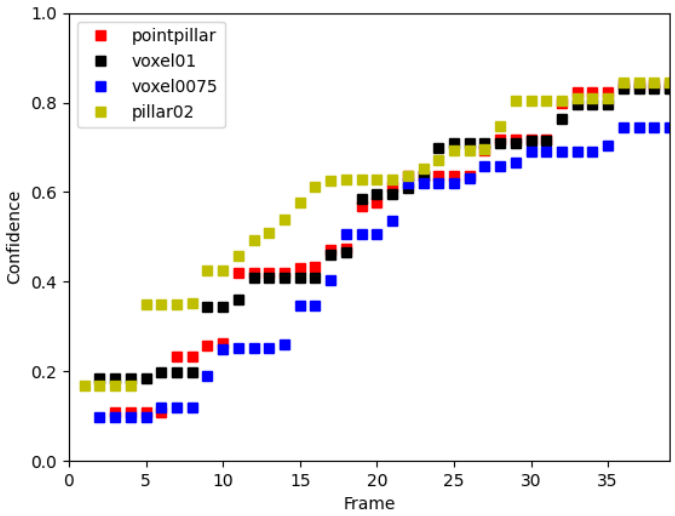

Figure 4.3: Best confidence scene for centerpoint-pillar02

Table 4.16: Tracking results evaluation best scene performance for centerpoint-voxel01 additional metrics scene: efa5c96f05594f41a2498eb9f2e7ad99

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.985	4	0	5	67	2	1	0	68
Pillar02	1	4	0	2.5	68	1	0	0	68
Voxel01	1	4	0	2.5	68	1	0	0	68
Voxel0075	0.985	4	0	5	67	2	1	0	68

Table 4.17: Tracking results evaluation best scene performance for centerpoint-voxel0075 scene: 9f1f69646d644e35be4fe0122a8b91ef

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.898	0.919	0.313	0.916	0.337	0	0.25	0.25
Pillar02	0.966	0.988	0.291	0.971	0.292	0	0.25	0.25
Voxel01	0.989	0.989	0.255	0.995	0.216	0	0	0
Voxel0075	1	1	0.222	1	0.197	0	0	0

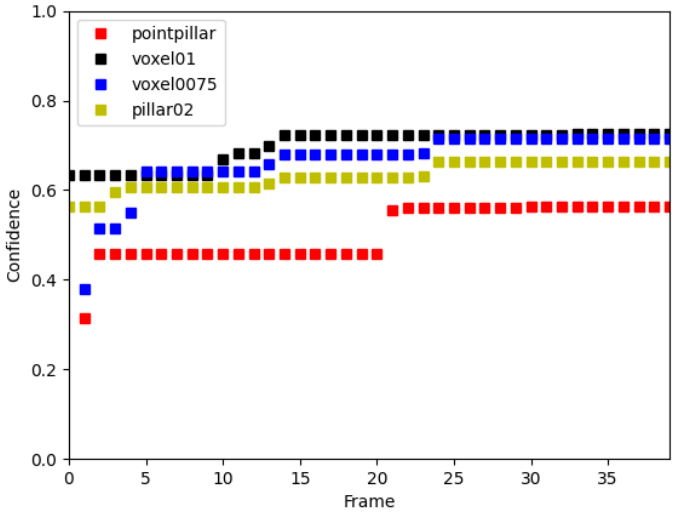

Figure 4.4: Best confidence scene for centerpoint-voxel01

Table 4.18: Tracking results evaluation best scene performance for centerpoint-voxel0075 additional metrics scene: 9f1f69646d644e35be4fe0122a8b91ef

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.977	4	0	17.5	86	7	2	0	88
Pillar02	0.977	4	0	2.5	86	1	2	0	88
Voxel01	1	4	0	2.5	88	1	0	0	88
Voxel0075	1	4	0	0	88	0	0	0	88

Table 4.19: Tracking results evaluation worst scene performance for all trackers scene: d1e57234fd6a463d963670938f9f556e

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0	0	1.055	0	1.561	0	0.25	0.5
Pillar02	0	0	0.903	0	1.424	0	0	1
Voxel01	0.286	0.5	0.63	0.263	1.281	0	0	0.5
Voxel0075	0.286	0.4	0.684	0.27	1.11	0	0	0

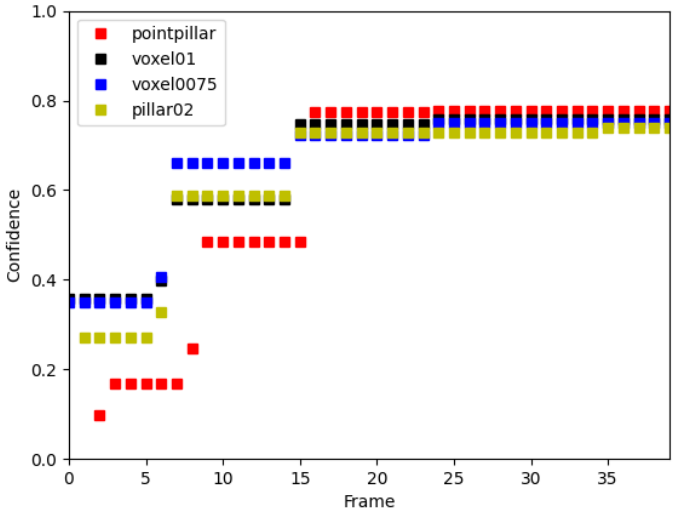

Figure 4.5: Best confidence scene for centerpoint-voxel0075

Table 4.20: Tracking results evaluation worst scene performance for all trackers additional metrics scene: d1e57234fd6a463d963670938f9f556e

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.714	1	0	155.17	5	45	2	0	7
Pillar02	0.429	0	1	78.57	3	11	4	0	7
Voxel01	0.571	1	1	28.57	4	2	3	0	7
Voxel0075	0.714	1	1	33.33	5	3	2	0	7

Table 4.21: Tracking results evaluation scene with most IDs for pointpillar metrics scene: d7bacba9119840f78f3c804134ceece0

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.584	0.704	0.455	0.662	0.699	17	0.383	0.65
Pillar02	0.652	0.788	0.358	0.632	0.619	1	0.75	0.893
Voxel01	0.737	0.845	0.364	0.713	0.543	3	0.45	0.633
Voxel0075	0.719	0.837	0.326	0.712	0.509	1	0.448	0.638

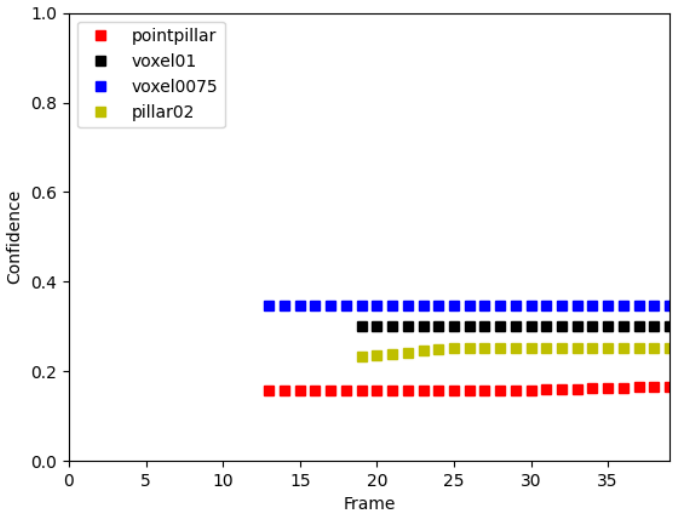

Figure 4.6: Confidence for the worst scene for the four approaches

Table 4.22: Tracking results evaluation scene with most IDs for pointpillar additional metrics scene: d7bacba9119840f78f3c804134ceece0

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.872	22	4	245	331	98	51	3	399
Pillar02	0.83	19	6	175	330	70	68	0	399
Voxel01	0.88	24	5	135	348	54	48	3	399
Voxel0075	0.862	21	5	140	343	56	55	2	399

Table 4.23: Tracking results evaluation scene with most IDs for centerpoint-pillar02 scene: 2ed0fcbfc214478ca3b3ce013e7723ba

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.573	0.77	0.362	0.708	0.511	15	0.389	0.717
Pillar02	0.691	0.796	0.320	0.819	0.420	11	0.316	0.457
Voxel01	0.675	0.825	0.29	0.804	0.418	7	0.203	0.316
Voxel0075	0.665	0.859	0.275	0.804	0.41	7	0.233	0.356

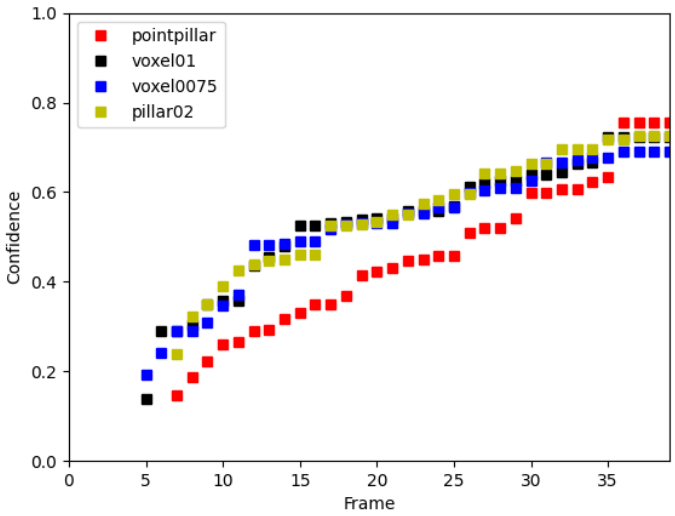

Figure 4.7: Confidence in case pointpillar with most IDs

Table 4.24: Tracking results evaluation scene with most IDs for pointpillar additional metrics scene: 2ed0fcbfc214478ca3b3ce013e7723ba

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.753	84	32	823.08	1396	321	462	13	1873
Pillar02	0.875	104	14	851.28	1627	332	235	15	1873
Voxel01	0.823	95	24	689.74	1534	269	332	14	1873
Voxel0075	0.778	94	30	525.64	1451	205	415	10	1873

Table 4.25: Tracking results evaluation scene with most IDs for centerpoint-voxel01 scene: e005041f659c47e194cd5b18ea6fc346

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.675	0.841	0.389	0.814	0.504	3	0.689	0.770
Pillar02	0.702	0.844	0.319	0.755	0.58	7	0.81	1.06
Voxel01	0.696	0.814	0.35	0.783	0.512	12	0.5	0.707
Voxel0075	0.735	0.841	0.311	0.801	0.475	2	0.655	0.75

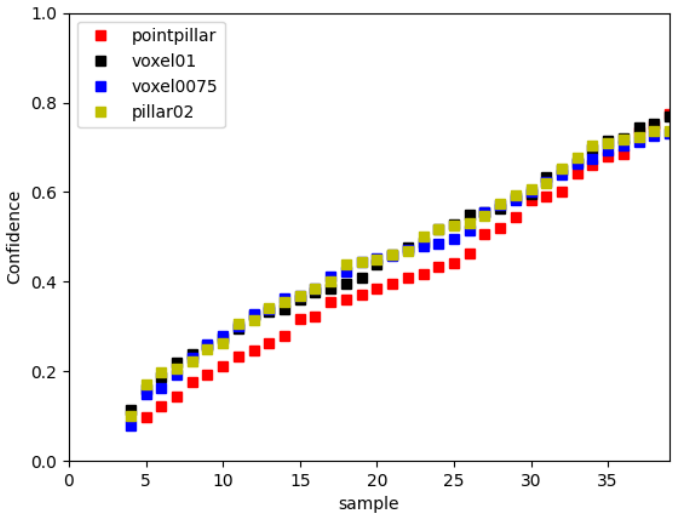

Figure 4.8: Confidence in case pillar02 with most IDs

Table 4.26: Tracking results evaluation scene with most IDs for centerpoint-voxel01 additional metrics scene: e005041f659c47e194cd5b18ea6fc346

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.808	31	9	224.39	577	92	138	3	718
Pillar02	0.841	31	2	226.83	597	93	114	8	718
Voxel01	0.872	33	3	278.05	614	114	92	11	718
Voxel0075	0.877	36	2	243.90	628	100	88	6	718

Table 4.27: Tracking results evaluation scene with most IDs for centerpoint-voxel0075 scene: 64bfc5edd71147858ce7446892d7f864

Detection	MOTA	MOTAR	MOTP	AMOTA	AMOTP	IDS	TID	LGD
PointPillar	0.741	0.834	0.39	0.81	0.438	2	0.59	0.8
Pillar02	0.765	0.86	0.387	0.832	0.444	9	0.37	0.707
Voxel01	0.778	0.863	0.371	0.831	0.446	5	0.456	0.717
Voxel0075	0.808	0.895	0.377	0.849	0.442	11	0.468	0.723

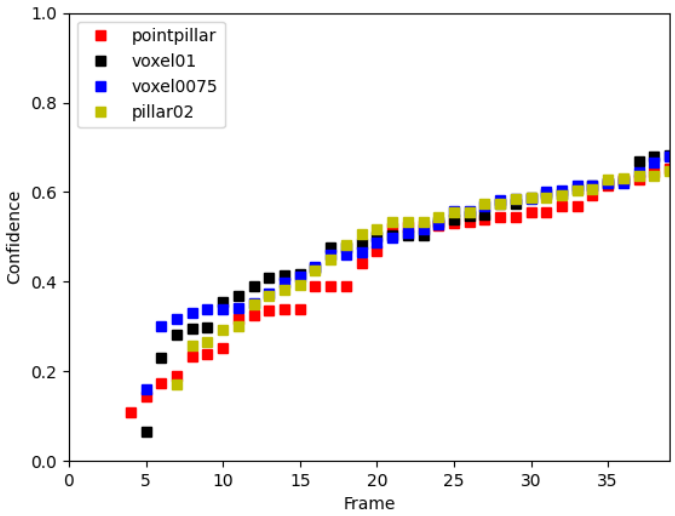

Figure 4.9: Confidence in case centerpoint-voxel01 with most IDs

Table 4.28: Tracking results evaluation scene with most IDs for centerpoint-voxel0075 additional metrics scene: 64bfc5edd71147858ce7446892d7f864

Detection	recall	mt	ml	faf	tp	fp	fn	frag	gt
PointPillar	0.889	40	4	377.5	913	151	114	6	1029
Pillar02	0.898	40	4	320	915	128	105	10	1029
Voxel01	0.907	40	4	317.5	928	127	96	3	1029
Voxel0075	0.914	39	2	245.00	929	98	89	9	1029

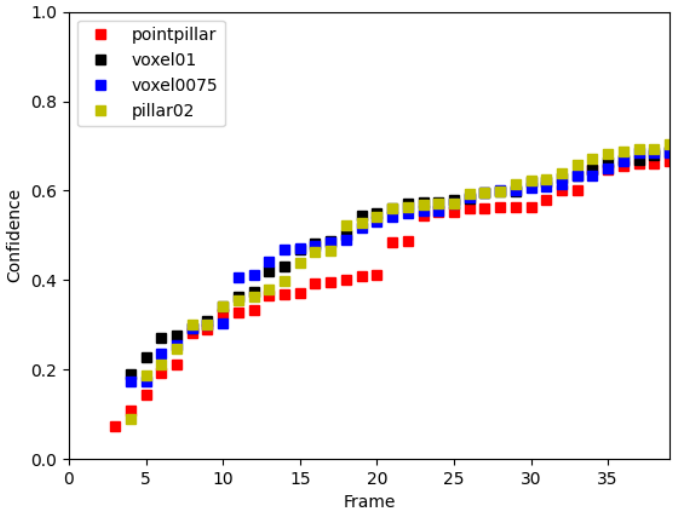

Figure 4.10: Confidence in case centerpoint-voxel0075 with most IDs

information that are calculated the number of lidar points, visibility, IOU, translation error, orientation error and tracking score.

The first example is, for instance, a car, which moved toward ego and then turned and went away. This instance is tracked using different trackers as shown in figure 4.11, figure 4.12, figure 4.13 and figure 4.14. In this example, it can be observed that pointpillar, centerpoint-pillar02 and voxel01 failed to keep track of this object until it left the scene and an identity switch occurred. Let's take pointpillar; at sample 22, this instance is tracked again using another tracking id. The same case is for pillar 02, where the identity switch occurred at sample 18. While centerpoint-voxel0075 has tracked this instance successfully to the end without an identity switch. This is illustrated in figure 4.15. Starting with the tracking score for pillar02 at sample 20, the tracking score for this tracker is very low, and the translation error is very high. This is due to be this instance occluded at this frame. But in case voxel0075, even though it is occluded, it keeps successfully tacking it. The same is true for pointpillar, where the translation error and orientation error are high at the moment of identity switching. The visibility for this instance was 2 over all samples; this means that the visibility is between 40 and 60 %, where the worst visibility is 1 level between 0 and 40 %, and the best visibility is 4 level between 80 and 100 %. And for visibility at three levels between 60 and 80, From this result also, it can be seen that the number of lidar points has been token for this instance. The highest number of lidar points occurs when this object is close to the ego pose.

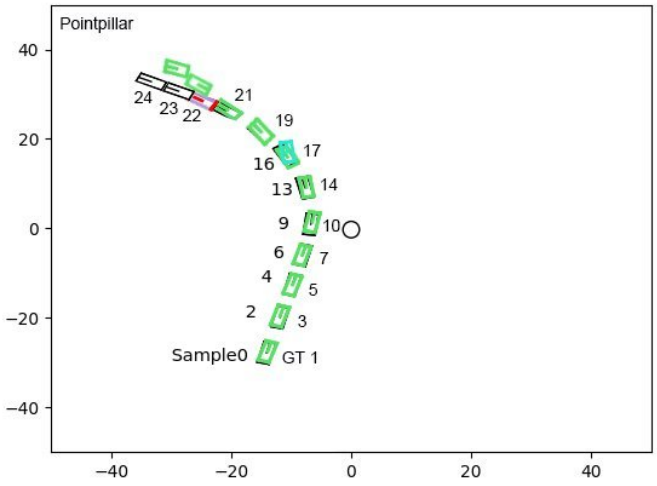

Figure 4.11: Example1: turned car tracking result for pointpillar tracker for instance: 52504c169efd4a0bb1437bfb8ed5bdb9

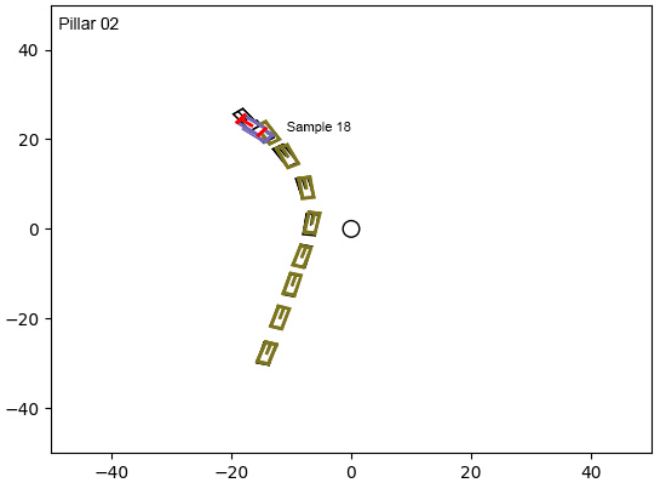

Figure 4.12: Example1: turned car tracking result for centerpoint-pillar02 tracker for instance: 52504c169efd4a0bb1437bfb8ed5bdb9

.

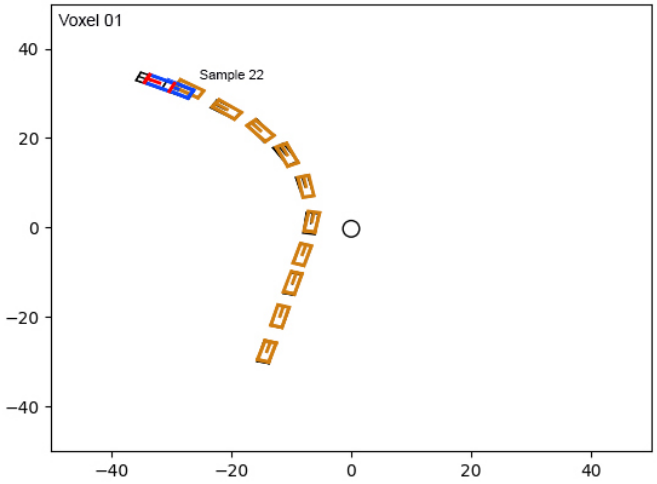

Figure 4.13: Example1: turned car tracking result for centerpoint-voxel01 tracker for instance: 52504c169efd4a0bb1437bfb8ed5bdb9

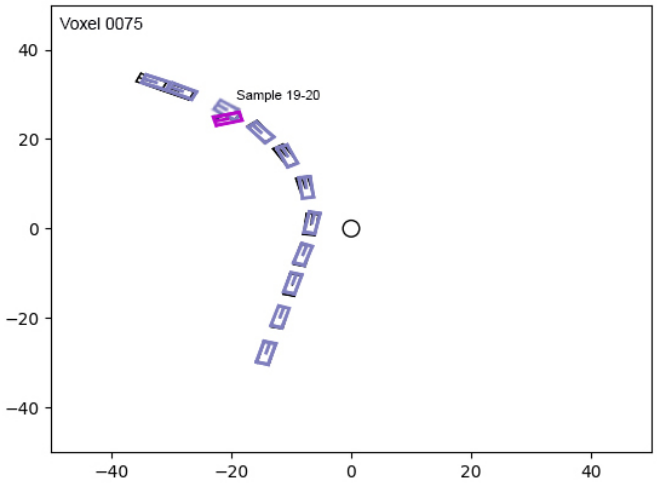

Figure 4.14: Example1: turned car tracking result for centerpoint-voxel0075 tracker for instance: 52504c169efd4a0bb1437bfb8ed5bdb9

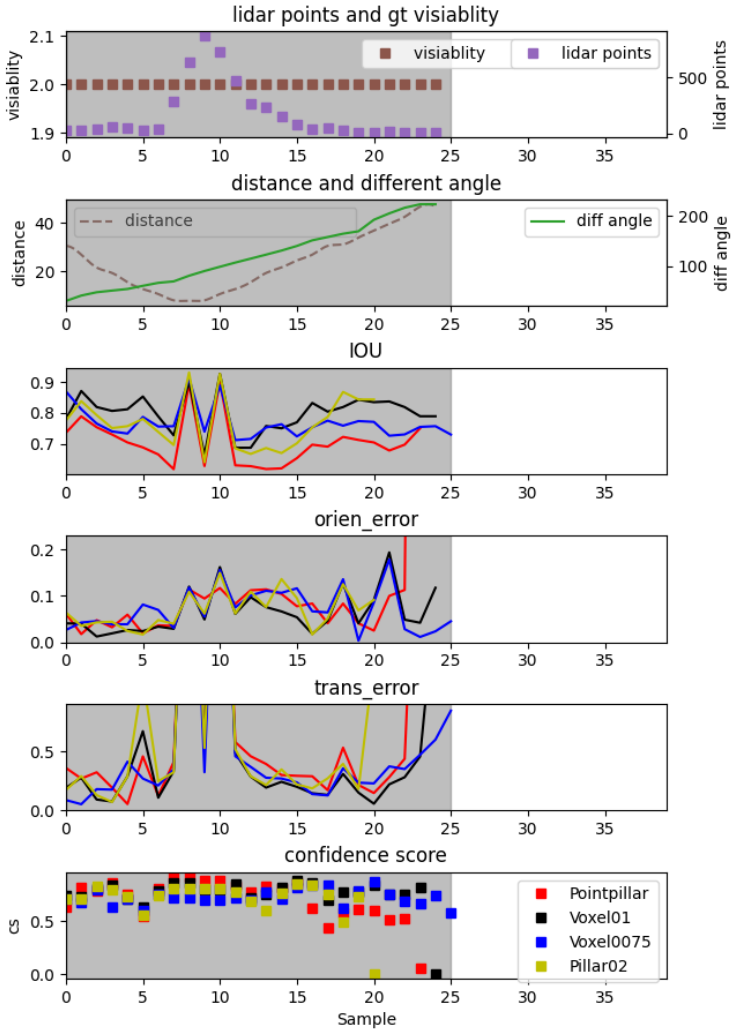

Figure 4.15: Example1: turned car errors calculation for pointpillar, centerpoint-voxel01, voxel0075 and pillar02 trackers for instance: 52504c169efd4a0bb1437bfb8ed5bdb9

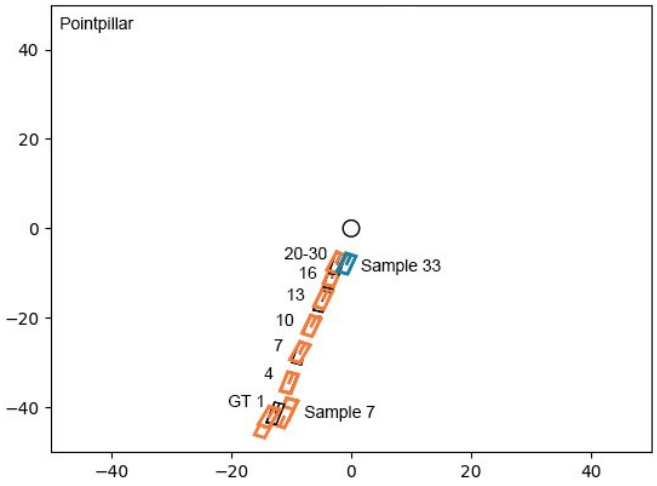

Figure 4.16: Example2: car moved toward ego and parked there tracking result for pointpillar tracker for instance: 165ac7e035e44c82af4856af7e993c9a sample 33

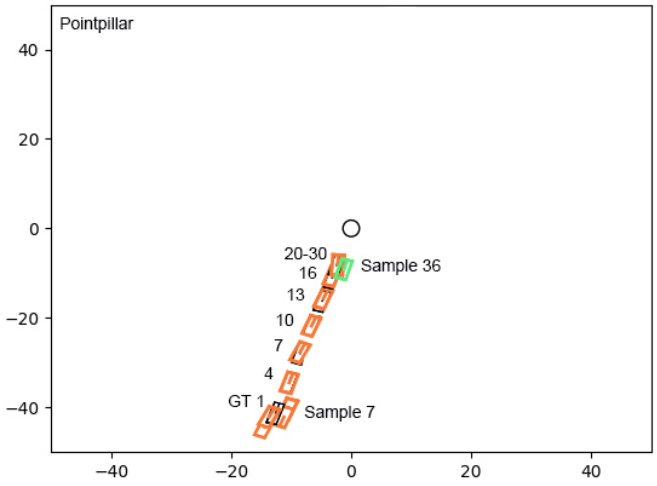

Figure 4.17: Example2: car moved toward ego and parked there tracking result for pointpillar tracker for instance: 165ac7e035e44c82af4856af7e993c9a sample 36

The second example is for instance car, where this car moved toward ego and then parked there from sample 30 to sample 39. This instance appears from samples 10 to 39, which is seen in figure 4.16. For this example, also, four different trackers are used to track it. Information about the instance is obtained and calculated, as shown in figure 4.18.

The visibility for this object was 4 over all frames, and the number of lidar points increased as it came closer. Let's start with pointpillar. It can be observed from figure 4.18 that the tracking score for the tracker decreased clearly in samples 33 and 36. The reason behind that, at samples 33 and sample 36, this instance is occluded by other objects, as shown in figure 4.16 and figure 4.17. Also, by looking to sample 33 and 36, it can be seen that orientation error increased and IOU decreased. Translation error went to being stable after this object parked near ego for all trackers after sample 30.

Different types of information are obtained and can be useful to understand the situations in every scene, as in this example. It can be observed where the confidence score is decreasing and where the higher orientation error or translation error is occurring, and if there is an identity switch, the behavior of different errors can be observed.

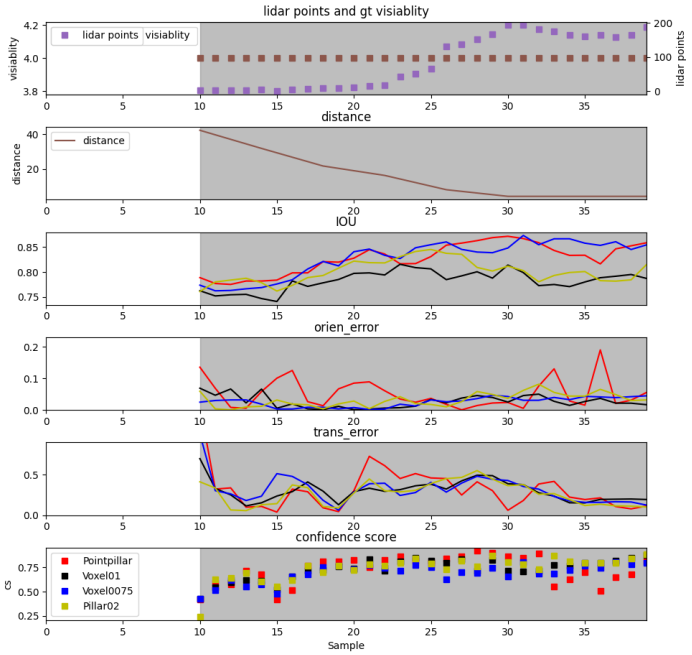

Figure 4.18: Example2: car moved toward ego, errors calculation for pointpillar, centerpoint-voxel01, voxel0075 and pillar02 trackers for instance: 165ac7e035e44c82af4856af7e993c9a

The third example is from the worst scene. This scene includes just two cars with just seven ground truths, but the results are very bad, as shown in 4.19. Pointpillar and pillar02 give 0 AMOTA. The description for this scene is "Follow bus, cross intersection, nature". So, to understand why the results are very bad for this scene, let's measure the errors for this instance and see the visibility as well as the number of lidar points for one instance. The number of lidar points is very low, although the visibility is very good. Figure 4.19 shows the high number of false positives in pointpillar. The first annotation for this instance is at sample 33, with black color near the cyan tracker figure 4.20, and the last annotation is at sample 37. By looking to 4.21, it can be observed that the translation and orientation errors are very high. These are some factors that can influence the tracking score for this instance.

The fourth example is for instance, with very good results in figure 4.22, Instance is parked near from ego and has very good visibility over all samples, and the number of lidar points is good. From figure 4.24, the results are very good for this instance; just at the first sample, the tracking score is very low as this instance tracked for the first time for centerpoint, but for pointpillar, it was better. Starting with pointpillar, it can be observed that orientation error at the first sample is very high for voxel01 and low for pointpillar.

Some suggested solutions to improve results are to work on handling occlusions, reduce the number of false positives and false negatives, handle changes in orientation, and handle other errors. This can be done by using multiple tracklets for the same object, as shown in the last examples. Where there are good trackers and bad trackers. So, based on the good tracker information, the other trackers can be handled.

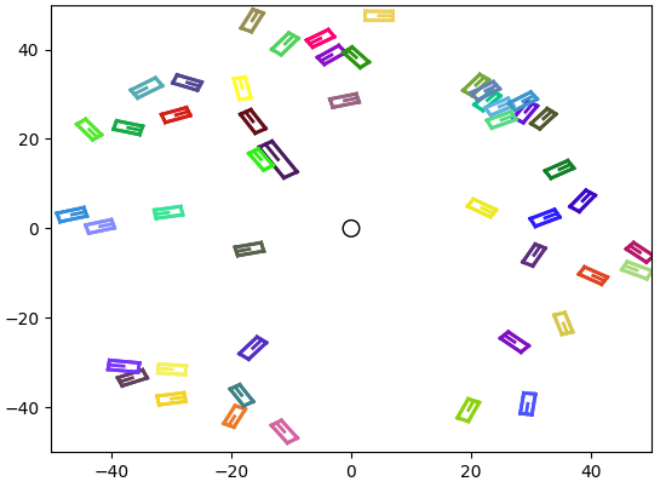

Figure 4.19: Render one sample from pointpillar for worst scene

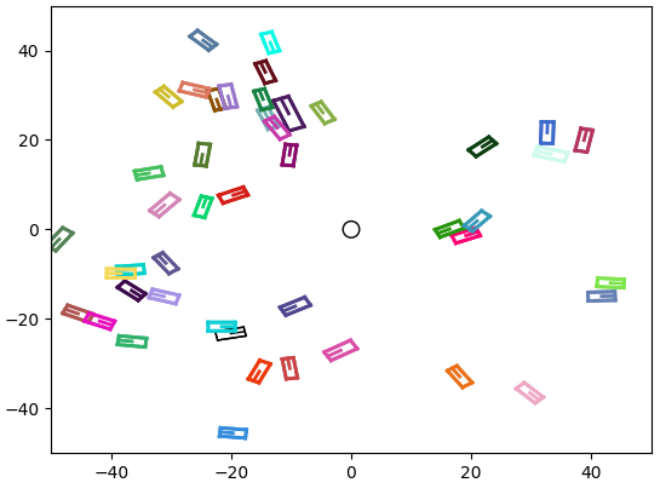

Figure 4.20: Render one sample from pointpillar for worst scene sample 33 for instance: f1305c22cb13447e955227022c3cdf4e

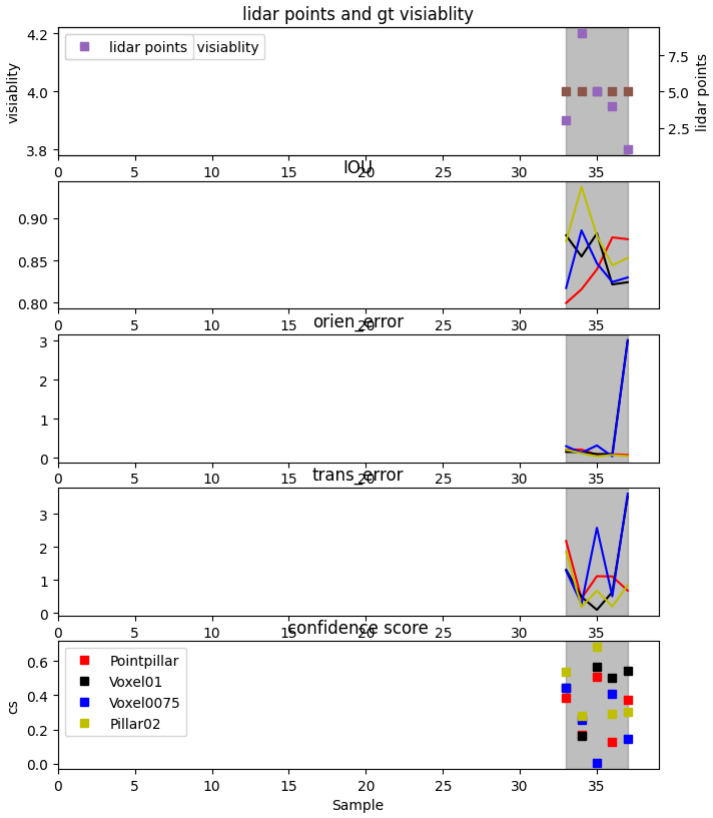

Figure 4.21: Result calculation for instance: f1305c22cb13447e955227022c3cdf4e

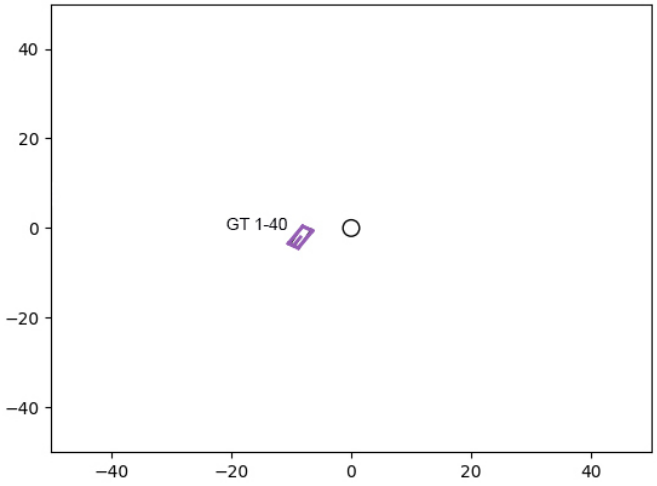

Figure 4.22: Stationary car near ego for instance: 2eb83a6a5aa143658a149e0f40b9a7ab

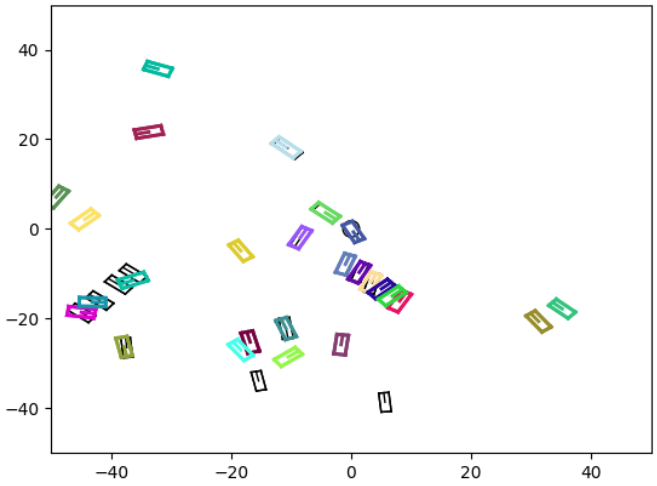

Figure 4.23: Stationary car near ego for pillar02 sample 12 for instance: 2eb83a6a5aa143658a149e0f40b9a7ab in purple color

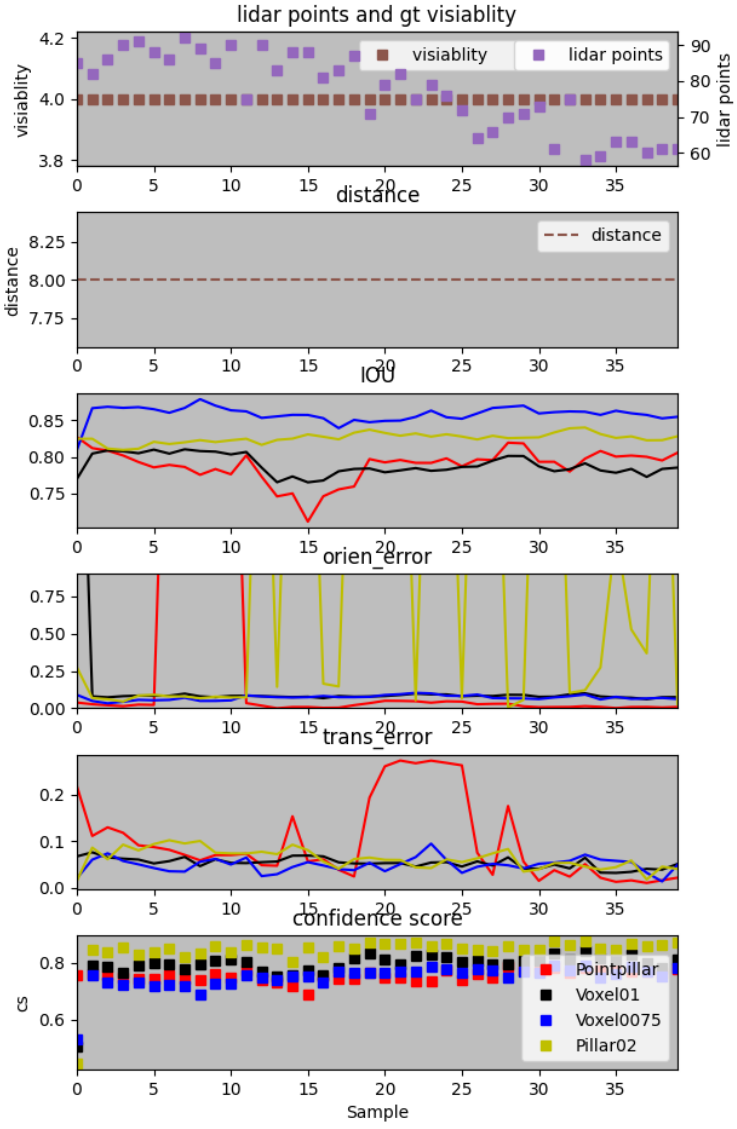

Figure 4.24: Result calculation for stationary car near ego for instance: 2eb83a6a5aa143658a149e0f40b9a7ab

5 Summary

The relationship between object detection and tracking is an essential component for many computer vision applications, and there is a necessity to improve their performance. Improvement and increasing the certainty of detection approaches are very important in order to raise the safety of autonomous vehicles. There are several methods to improve the performance of object detection systems; in this thesis, the redundant tracklets method is used for this task. Using redundant tracklets can aid in improving the performance of object detection approaches. Redundant tracklets mean that multiple tracklets and sequences of bounding boxes over time correspond to a particular object, which can be used to address difficult challenges such as changing orientation and occlusion. The goal of using redundant tracklets is to improve object detection and increase the confidence score. Additional to the confidence score and tracker information, other information can be provided about the object, such as distance information, resulting in another parameter that quantifies whether this is likely to be the particular object. This can help reduce the number of false positives and false negatives in the final result.

The selected dataset is nuScenes, which contains many scenes to get different conditions. The chosen object detection systems are pointpillar and centerpoint, where the data collected is based only on lidar sensors. For object tracking tasks, the immortal tracker is used to give very good performance, using the Kalman filter to handle occlusions and track all objects in complex environments.

Different evaluation metrics are used for object detection models evaluation, such as mean average precision, mean true positive metrics, and the nuScenes detection score. On the other hand, different evaluation metrics are utilized for object tracking evaluation, such as the MOTA, MOTP, TID, LGD, and IDs metrics. Using different metrics can provide a clear image of the model's performance.

Conclusively, improvement of object detection approaches is a long-term topic with different directions and a lot of challenges to treat different problems. As the requirement for reliable object detection systems continues to grow in fields such as autonomous driving and robotics, there will be a continued need for novel approaches to improve their performance. Overall, the utilization of redundant tracklets can lead to more accurate and robust object detection systems, especially in difficult and complex scenarios where objects could be temporarily occluded or disappear from visibility.

References

[BS22] Boschmann, Waldemar ; Söffker, Dirk: Complementary and situation sensitive object detection, review performance, and performance dependencies of common approaches. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, S. 888–892

[Pet09] Petrovskaya, Anna ; Thrun, Sebastian: Model Based Vehicle Tracking for Autonomous Driving in Urban Environments. In: *Autonomous Robots*, 2009, S. 123–139

[Hem22] Hemanth, Jude: Machine learning techniques for smart city applications: Trends and solutions (1st ed.) In: *Springer International Publishing*, 2022, S. 63

[nus20] Caesar, Holger ; Bankiti, Varun ; Lang, Alex H. ; Vora, Sourabh ; Liong, Venice E. ; Xu, Qiang ; Krishnan, Anush ; Pan, Yu ; Baldan, Giancarlo ; Beijbom, Oscar: nuScenes: A Multimodal Dataset for Autonomous Driving. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, S. 11618–11628

[tra20] nuScenes tracking task In: *Nuscenes.org*, 2020

[det20] nuScenes detection task In: *Nuscenes.org*, 2020

[hota22] Luiten, Jonathon ; Osep, Aljosa ; Dendorfer, Patrick ; Torr, Philip ; Geiger, Andreas ; Leal-Taixe, Laura ; Leibe, Bastian: HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking In: *International Journal of Computer Vision, 129(2))*, 2020, S. 548–578

[Imm22] Wang, Qitai ; Chen, Yuntao ; Pang, Ziqi ; Wang, Naiyan ; Zhang, Zhaoxiang: Immortal Tracker: Tracklet Never Dies In: *arXiv [cs.CV]. https://doi.org/10.31219/osf.io/am36b*, 2022

[mmdet20] MMDetection3D Contributors: MMDetection3D: OpenMMLab next-generation platform for general 3D object detection In: *https://github.com/open-mmlab/mmdetection3d*, 2020

[cao20] Cao, Danyang ; Chen, Zhixin ; Gao, Lei: An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks In: *Human-Centric Computing and Information Sciences, 10(1). https://doi.org/10.1186/s13673-020-00219-9*, 2020

[gar20] Carranza-García, Manuel ; Torres-Mateo, Jesús ; Lara-Benítez, Pedro ; García-Gutiérrez, Jorge: On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data In: *Remote Sensing, 13(1), 89. https://doi.org/10.3390/rs13010089*, 2020

[wang21] Wang, Yingjie ; Mao, Qiuyu ; Zhu, Hanqi ; Zhang, Yu ; Ji, Jianmin ; Zhang, Yanyong: Multi-modal 3D object detection in autonomous driving: A survey In: *arXiv [cs.CV]. http://arxiv.org/abs/2106.12735*, 2021

[riax20] Yadav, Ritu ; Vierling, Axel ; Berns, Karsten: RADAR+RGB ATTENTIVE FUSION FOR ROBUST OBJECT DETECTION IN AUTONOMOUS VEHICLES In: *arXiv [cs.CV]. http://arxiv.org/abs/2008.13642*, 2020

[cent20] Yin, Tianwei ; Zhou, Xingyi ; Krähenbühl, Philipp: Center-based 3D object detection and tracking In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021

[cne20] Li, Kaidong ; Ma, Wenchi ; Sajid, Usman; Wu, Yuanwei; Wang, Guanghui: Object detection with convolutional neural networks In: *Deep Learning in Computer Vision (pp. 41–62). CRC Press*, 2020

[fuj22] Fujitake, Masato ; Sugimoto, Akihiro: Video representation learning through prediction for online object detection In: *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2022

[neu13] Neubert, Peer ; Sünderhauf, Niko ; Protzel, Peter: Appearance Change Prediction for Long-Term Navigation Across Seasons In: *Proc. of European Conference on Mobile Robotics (ECMR). DOI: http://dx.doi.org/10.1109/ECMR.2013.6698842*, 2013

[el01] Elnagar, Ashraf: Prediction of moving objects in dynamic environments using Kalman filters In: *Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No.01EX515)*, 2002

[Shi22] Shi1, Guangsheng ; Li1, Ruifeng ; Ma2, Chao: PillarNet: Real-time and high-performance pillar-based 3D object detection In: *arXiv [cs.CV]. http://arxiv.org/abs/2205.07403*, 2022

[Lang19] Lang, Alex H ; Vora, Sourabh ; Caesar, Holger ; Zhou, Lubing ; Yang, Jiong ; Beijbom, Oscar: PointPillars: Fast encoders for object detection from point clouds In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019

[Koay21] Koay, Hong Vin ; Chuah, Joon Huang ; Chow, Chee-Onn ; Chang, Yang-Lang ; Yong, Keh Kok: YOLO-RTUAV: Towards real-time vehicle detection through aerial images with low-cost edge devices In: *Remote Sensing, 13(21), 4196. https://doi.org/10.3390/rs13214196*, 2021

[mot23] Mandel, Travis ; Jimenez, Mark ; Risley, Emily ; Nammoto, Taishi ; Williams, Rebekka ; Panoff, Max; Ballesteros, Meynard ; Suarez, Bobbie: Detection confidence driven multi-object tracking to recover reliable tracks from unreliable detections In: *Pattern Recognition, 135(109107), 109107. https://doi.org/10.1016/j.patcog.2022.109107*, 2023

[Bre09] D. Breitenstein1, Michael ; Reichlin, Fabian ; Leibe, Bastian ; Koller-Meier, Esther ; Van Gool, Luc: Robust tracking-by-detection using a detector confidence particle filter In: *IEEE 12th International Conference on Computer Vision*, 2009

[Rou17] Ruochen, Fan ; Zhang, Fang-Lue ; Zhang, Min: Robust tracking-by-detection using a selection and completion mechanism In: *Computational Visual Media, 3(3), 285–294. https://doi.org/10.1007/s41095-017-0083-7*, 2017

[Ber19] Berrar, Daniel: Cross-Validation In: *Encyclopedia of Bioinformatics and Computational Biology (pp. 542–545). Elsevier*, 2019

[Gon21] Gonzales–Martínez, Rosa ; Machacuay, Javier ; Rotta, Pedro ; Chinguel, César: Faster R-CNN with a cross-validation approach to object detection in radar images In: *IEEE International Conference on Aerospace and Signal Processing (INCAS)*, 2021

[Mor21] Morozov, Alexey ; Angulo, Brian ; Mottl, Vadim ; Tatarchuk, Alexander ; Krasotkina, Olga: Differential leave-one-out cross-validation for feature selection in generalized linear dependence models In: *3rd International Conference on Information Technology and Computer Communications*, 2021

[Wit11] Witten, Ian H ; Frank, Eibe: Data mining: Practical machine learning tools and techniques (3rd ed.) In: *Morgan Kaufmann*, 2011

[Sop21] Soppin, Shashidhar ; Ramachandra, Manjunath ; Chandrashekar, B N: Essentials of deep learning and AI: Experience unsupervised learning, autoencoders, feature engineering, and time series analysis with TensorFlow, keras, and scikit-learn In: *BPB Publications*, 2021